Embedded System for Control and data acquisition of NLO experiments

A Thesis

Submitted to the Department of Physics as a Partial Fulfillment of the Requirement for the Degree of Master of Science

Submitted By

Syed Razwanul Haque

Reg. No.: 2011122011

Session: 2011-2012

Supervised By

Professor Yasmeen Haque

Department of Physics,

SUST

Department of Physics

Shahjalal University of Science & Technology Sylhet-3114, Bangladesh.

April, 2014

ACKNOWLEDGEMENTS

First of all I would like to express my sincere gratitude to **Dr. Yasmeen Haque**, Professor, Department of Physics, Shahjalal University of Science & Technology, who has been supervised me since the beginning of my present work. She gave me proper guidance, important advices and constant encouragement to successfully complete my thesis. Without her assistance I could not able to complete my thesis.

I also wish to express my appreciation to **Prof. Md. Zafar Iqbal**, Head, Department of EEE, Shahjalal University of Science & Technology, who gave me valuable suggestions and constructive advices throughout my thesis. He also encouraged me during my thesis work.

Special gratitude goes to Md. Enamul Hoque, Lecturer, Department of Physics, for providing numerous ideas and useful suggestions.

I also express my sincere gratitude to Dr. Sharif Md. Sharafuddin, Professor, Department of Physics, Shahjalal University of Science & Technology, for his constructive advices that improved the quality of this study.

My keen appreciation goes to Manash Kanti Biswas who has played key role throughout my thesis work. He has assisted me all the time.

I express my special thanks to Robi Kormokar and Md. Maruf Hussain Rahat, student, Department of Physics, Shahjalal University of Science & Technology, for their co-operation.

I also express my special thanks to all researchers of SUST Robotics, Aeronautics & Interfacing Research Group.

Thanks to the Department of physics of Shahjalal University of Science & Technology, faculty and staff members. During my MS research, the Department of physics provided primary financial support with partial funding through HEQEP.

Syed Razwanul Haque

April 2014

Dedicated
To
My Country Bangladesh

ABSTRACT

Highly precise devices are required for extensive, modern and sophisticated experimental research work. Two devices have been developed to control the laser and rotational stage. In those Embedded systems AVR Microcontroller has been used where the programming language was C++. The remote control has been developed for the laser controller and external mechanical device has been designed to mount a stepper motor with the rotational stage. The measured Accuracy for the controller of laser and rotational stage has been found to be 3% and 5% respectively.

Contents

Chapter 1

1.0 Introduction

Chapter 2

An Embedded System for Control of an Argon ion Laser

2.1 Introduction
2.2 Laser Controller
2.3 Specification of ModuLaser (Model: Steller-Pro)
2.4 Stellar-Pro Features
2.5 Available System Configuration
2.6 The Electronic Interface
2.7 System Architecture
  2.7.1 Laser Control Unit
  2.7.2 Remote Control Unit
2.8 Experiments Using LASER Controller
2.9 Future Works and Limitations.

Chapter 3

An Embedded System to Control Rotational Stage

2.0 Rotational Stage
2.1 Purpose of Rotational Stage
2.2 Embedded system to control rotational stage
2.3 Total System Architecture
2.3.1 Mechanical Unit
2.3.1.1 Newport RSP-1T Manual Rotational Stage
2.3.2 Motor Driver
2.3.3 Control Unit
2.4.3.1 Menu System
2.5 Algorithm

2.6 Coding
2.6.1 Some Techniques used in Coding
2.7 Experiments
2.8 Future work and Limitations

Chapter 3
3.1 References
3.2 Appendix

Chapter

Two

An Embedded System for Control of an Argon ion Laser

2.1. Introduction

A Laser controller is very important in laser operations especially for high powered laser for which user command and laser feedback data should be very accurate. Modu laser and a similar series of lasers used in a condition where giving commands using an analog switch and knobs can result in error and can consume a lot of time. A digital laser controller offers lots of facilities to users such as digital display to see laser operating conditions, wireless remote unit, password protection, data recording to flash drive etc. Moreover in case of an analog on board controller there are no safety features which could protect a wrong command from user and hence can increase laser lifetime. Laser power increasing or decreasing using analog on board controller is logarithmic and hard to control. A Microcontroller based digital laser controller can give input as percentage and can increase or decrease power linearly. Amicrocontroller based laser is also safe, user friendly and precise. The main circuit board has many ports for installs different kinds of modules.

2.2. Necessity of a Laser Controller

The laser used in the Nonlinear Optics lab is an Ar-ion Laser(ModuLaser ,Model: Steller-Pro) which has an on board analog controller. There are analog switches to control the shutter, laser power and laser on/off. The laser shutter control button is placed on the front side of the laser and the power control knob is located on the backside of the laser. During optical experiment a dark room situation is necessary and in most of the cases room lights remain off with many optical equipments placed on the optical table. In these conditions laser control by touching the laser physically is not convenient. So for accurate measurement it is imperative to control the laser using a remote control unit which is wireless. Laser power control using an analog controller is fully power meter dependent because analog knob is not precise and user cannot give numerical inputs. Laser has no display to show its operating data such as laser power, shutter state, tube current etc. The laser controller which has been designed has two luminous displays, one in remote control unit and the other in the laser control unit. So users can easily observe various laser data and don't need any external source to observe the laser data. The laser controller device has some extra ports to interface an SD card to save data and RTC to record time. It is also capable for interfacing other external devices with its main circuits such as WiFi module, GSM module, Xbee module , Ethernet module etc.

2.3 Specification of Stellar-Pro-CE Laser:

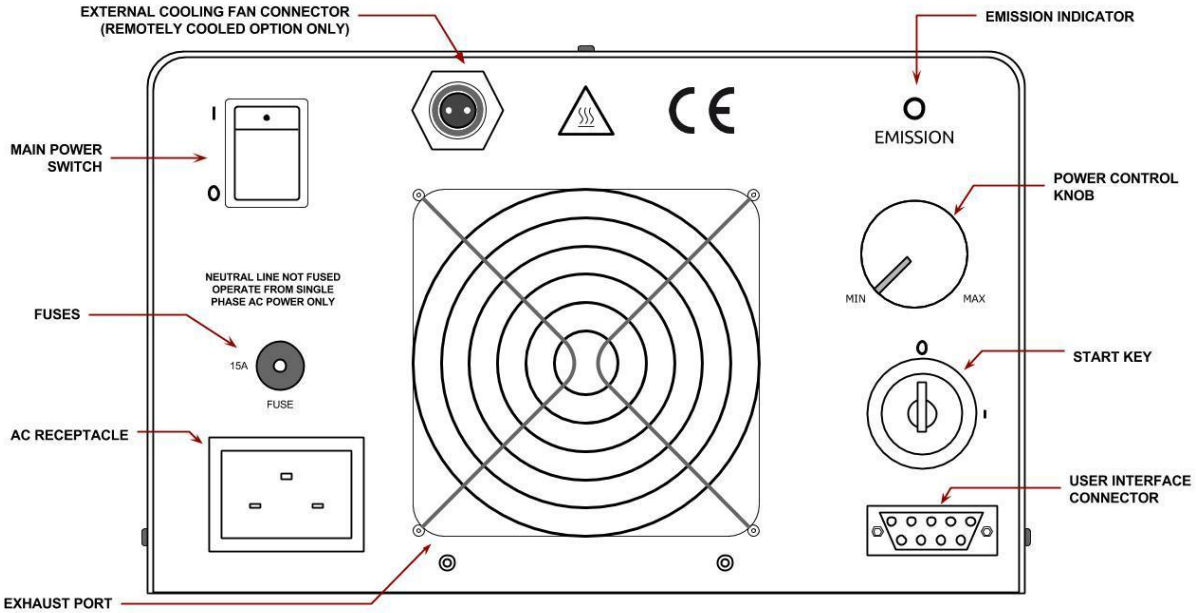
Table 2.1: Specifications of Stellar Pro-CE Laser

Beam Diameter 1/e2	0.65mm
Beam Divergence	0.95mrad
Full Angle Polarization	>100:1
Beam Pointing Stability(after warm-up)	<30urad
Beam Amplitude Noise	<1% RMS
Beam Output Power Drift(after warm-up)	<+/- 1%
Start Delay	30 sec. (approx)
Warm-up Time(from cold start)	5min
Beam Height	2.375"
AC Line Voltage (Universal)	100VAC-265VAC
Line Frequency	50Hz to 60Hz
Power Consumption	1500 Watt(max)

Case Width	7.60”
Case Height	5.06”
Case Length	13.0”
Total Weight	<13.5 lbs
Maximum Operational Ambient Temperature	35˚
Maximum Operational Relative Humidity	80%

2.4 Stellar-Pro Features

This is the rear view of Stellar-pro laser which is driven by our laser controller. In this diagram (rear view) there are Main Power Switch, Power Control Knob, Emission Indicator, Start Key, Fuses, External Cooling Fan Connector, Exhaust Port and User Interface Connector. We used User Interface Connector to interface our laser controller with this laser.



Rear View of ModuLaser (Steller-PRO)

1.5 Available System Configurations:

Beam Configuration Options:

Wave Length	Mode	Polarization	Maximum Power

457nm	TEM00	Yes	5mW
	TEM00	Yes	10mW
	Multi-Mode	No	20mW
488nm	TEM00	Yes	25mW
	TEM00	Yes	50mW
	Multi-Mode	No	100mW
514nm	TEM00	Yes	25mW
	TEM00	Yes	50mW
	Multi-Mode	No	100mW
Multi-line	TEM00	Yes	150mW
	Multi-Mode	No	300mW

Remotely Cooled Option:

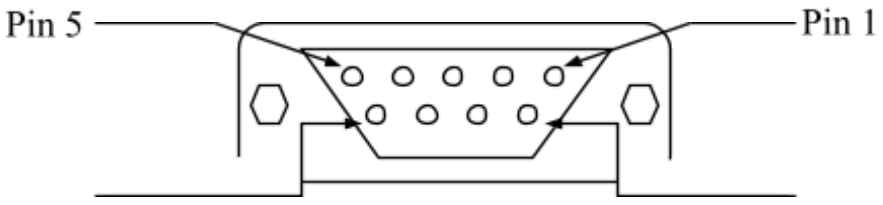
Internal cooling fans are removed and an external cooling fan assembly is connected to the rear of the laser via a rugged 3" hose. This configuration is typically used for critical scientific and OEM applications since it eliminates any effect of fan vibration on the laser beam and ducts the hot exhaust air away from the vicinity of the laser**.**

Remotely Controlled Options:

An analog remote control is available which allows the Key Switch ON/OFF, Shutter, and Output Power to be controlled remotely. Lamps on the remote control indicate when mains power is applied, laser emission status, and shutter OPEN/CLOSE status. The remote control connects to the Interface Connector of the laser via an 8' (standard longer lengths available). But there is no display and it is wired.

1.6 Electronic Interface to Control Stellar Pro CE Laser

The user Interface connector provides an interface port whereby the user can control and monitor the laser from a remote location. By correctly interfacing to this port, the user can start the laser, control the laser power, monitor the tube current, and immediately disable laser output all remotely. The correct pin # orientation for the user interface connector (as seen on the rear panel) is shown below



Pin 9 Pin 6

Fig 1.6: User Interface Connector

Pin 1: Power Control:

A DC voltage level between 0 and 5VDC applied between pins 1 & 2 may be used to linearly control the laser output power (0V= minimum power, 5V=maximum power). The source to drive this input must be a low impedance type capable of sourcing and sinking at least 25ma.

Pins 2 & 9: ISO-GND:

Each of these pins is an isolated GND. These pins should be used as common reference points to which the other signals within the user interface connector should be referenced.

Pin 3: Remote Start:

Connecting this pin to isolated GND (pin 2 or 9) will start the laser. This feature works in parallel with the start key. In order to have ON/OFF control using this feature, the start key must be set to the OFF position.

Pin 4: Tube Current:

A high impedance voltmeter may be used to monitor the laser tube current proportional voltage between pins 4 & 2. The voltage to laser tube current proportionality is 1:1

Pin 5: External Lamp:

This pin is used to control the emission indicator lamp of an attached remote control unit.

Pin 6: Shutter Control:

This pin is used by the remote control unit to control the shutter. Connecting this pin to the isolated +12V will open the shutter.

Pin 7: Isolated +12V:

This pin is used to provide power to the remote control unit.

Pin 8: Remote Interlock:

The remote interlock input at pin 8, in conjunction with isolated GND at pin 9, may be interfaced with a door interlock system so as to disable laser emission if the door of the compartment or room, wherein the laser is housed, is opened. In order for the laser to start, pin 8 must be electrically tied to isolated GND at pin 2 or pin 9. If this connection is broken during laser operation, the laser beam emission will be terminated immediately.

Pin #	Description	Expected Signal
1	Power Control	0 to 5VDC
2	Isolated GND	Isolated GND
3	Remote Start	Connect with GND to Start
4	Laser Tube Current	0 – 10VDC (1 Volt/Amp)
5	External Emission Lamp	Emission Indicator
6	Shutter Control	Connect with GND to Open Shutter
7	Isolated +12V	+12V +/- 1.5V
8	Remote Interlock	To start connect with GND
9	Isolated GND	Isolated GND

1.7 System Architecture of Laser Controller

Total laser controller system divided into two major sections. Laser control Unit and Remote Control Unit. Laser control unit connect with laser and remote control unit for user interface. Remote control unit send and receive data with laser control unit using wireless module.

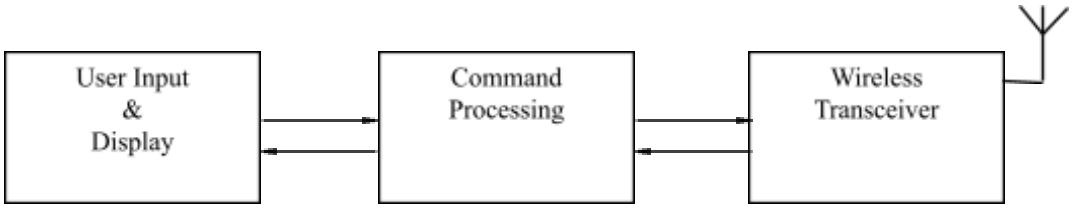


Fig 1.1: Remote Control Unit

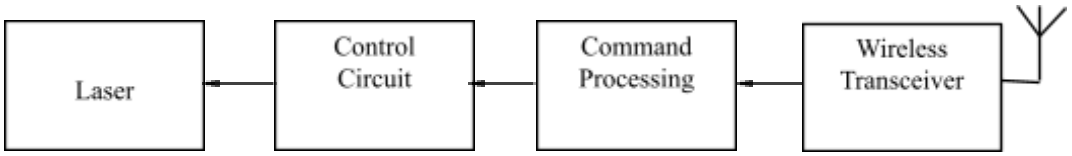


Fig 1.2: Laser Control Unit

Fig 1.2: Remote Control Unit (Left Side) and Laser Control Unit (Right)

1.7.1 Laser Control Unit

Laser control unit connected with laser via a wire. This unit receives digital command from remote control unit and converts it to analogue control mechanism. LCD display shows all necessary information such as shutter condition, laser power, duration etc to user and stored it to SD card.

1.7.1.1 Laser ON-OFF Control

Connecting pin 3 to GND will start (ON) the laser. When Laser Control Unit receives Laser ON command from Remote Control Unit a Microcontroller analyze this command and give a 5V to a Relay which makes connection between pin 3 and GND. First I tried with a CD4066 (quad bilateral switch) IC but it could not start the laser. This might be for its semiconductor property. Then I try with a electro mechanical switch (Relay). Relay is perfect for this type of laser switching. Laser OFF procedure is not so straight forward. When user press Laser OFF in Remote Control Unit then microcontroller check whether Laser power is zero or not. If laser power is zero then it give signal to Laser Control Unit to turns off the laser. If laser power is not zero it remind user to make laser power zero. Laser control device will not OFF the laser if power is not zero. This control mechanism increase laser safety because if user shutdown laser without making laser power zero it makes sudden temperature difference in laser tube which is harmful for laser.

1.7.1.2 Laser Shutter Control

Applying +12V to pin 6 will open the Shutter. When Laser Control Unit receives Shutter ON command from Remote Control Unit a Microcontroller analyze this command and give a 5V to a Relay which makes connection between pin 6 and +12V.

1.7.1.3 Laser Power Control

A DC voltage level between 0 and 5VDC applied between pins 1 & 2 may be used to linearly control the laser output power (0V= minimum power, 5V=maximum power). The source to drive this input must be a low impedance type capable of sourcing and sinking at least 25ma.

1.7.1.3.1 Varying voltage using PWM

An efficient way to varying voltage using a microcontroller circuit or digital circuit is PWM (Pulse Width Modulation). PWM can be converted into voltage.

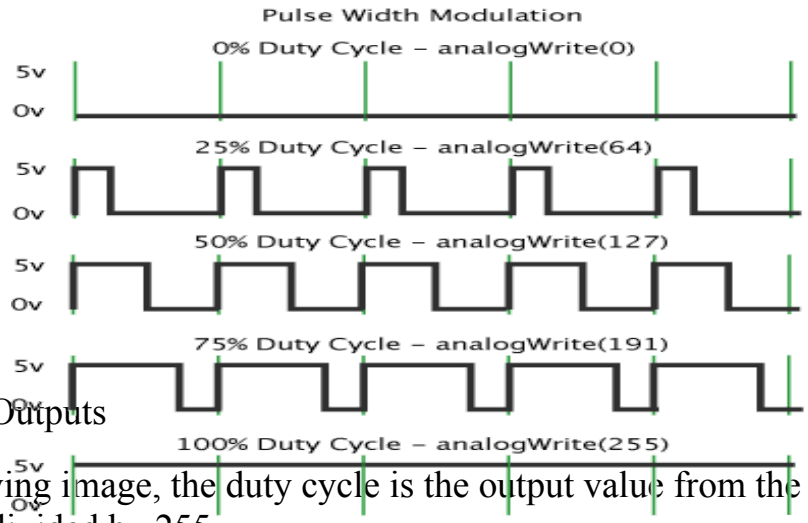


Fig : PWM Outputs

In the following image, the duty cycle is the output value from the PWM pin of an Arduino divided by 255.

For the Arduino, we can write a value from 0 to 255 on a PWM pin, and the Arduino library will cause the pin to output a PWM signal whose on time is in proportion to the value written.

When it comes time for us to actually write an output voltage, the 0-255 value lacks meaning. We want actually voltage. For our purposes, we will assume the Arduino is running at Vcc = 5 volts. In that case, a value of 255 will also be 5 volts. We can then easily convert the desired voltage to the digital value needed using simple division. We first divide the voltage we want by the 5 volts maximum. That gives us the percentage of our PWM signal. We then multiply this percentage by 255 to give us our pin value. Here is the formula:

Pin Value (0-255) = 255 * (AnalogVolts / 5);

In arduino *analogWrite* didn't actually output a voltage, but a PWM (pulse-width modulated) signal. After all, the ATmega had a A-D (analog to digital) converter along with Arduino's *analogRead*. The complementary *analogWrite* function was there, but no D-A (digital to analog) converter on the AVR chip itself. Fortunately, there is an easy way to convert a PWM signal to an analog voltage. To do so we only need to implement a simple single-pole low pass filter.

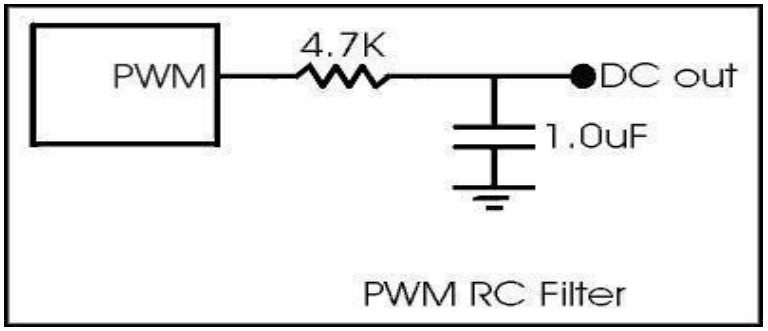


Fig: PWM RC Filter

1.7.1.4 SD Card Interfacing

SD card used in Laser Controller to log data which is produced as a result of laser operation. SD card saves laser operation time, date, RTC data, controlling data. So we can easily extract all information about laser control from SD card which is attached with Laser Controller.
The communication between the microcontroller and the SD card uses SPI, which takes place on digital pins 11, 12, and 13 (on most Arduino boards). Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. It can also be used for communication between two microcontrollers.
With an SPI connection there is always one master device (usually a microcontroller) which controls the peripheral devices. Typically there are three lines common to all the devices:

- **MISO** (Master In Slave Out) - The Slave line for sending data to the master,
- **MOSI** (Master Out Slave In) - The Master line for sending data to the peripherals,
- **SCK** (Serial Clock) - The clock pulses which synchronize data transmission generated by the master

and one line specific for every device:

- **SS** (Slave Select) - the pin on each device that the master can use to enable and disable specific devices.

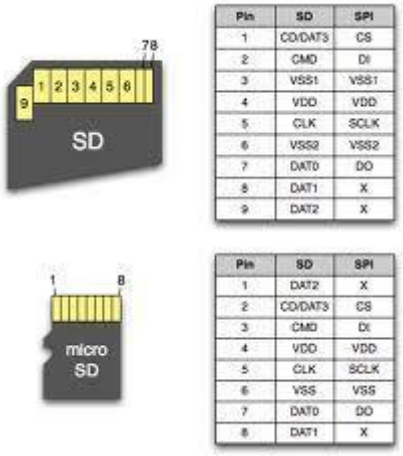
When a device's Slave Select pin is low, it communicates with the master. When it's high, it ignores the master. This allows you to have multiple SPI devices sharing the same MISO, MOSI, and CLK lines.

SD Library

The SD library allows for reading from and writing to SD cards, e.g. on the Arduino Ethernet Shield. It is built on sdfatlib by William Greiman. The library supports FAT16 and FAT32 file systems on standard SD cards and SDHC cards. It uses short 8.3 names for files. The file names passed to the SD library functions can include paths separated by forward-slashes, /, e.g. "directory/filename.txt". Because the working directory is always the root of the

SD card, a name refers to the same file whether or not it includes a leading slash (e.g. "/file.txt" is equivalent to "file.txt"). As of version 1.0, the library supports opening multiple files.

The communication between the microcontroller and the SD card uses SPI, which takes place on digital pins 11, 12, and 13 (on most Arduino boards) or 50, 51, and 52 (Arduino Mega). Additionally, another pin must be used to select the SD card. This can be the hardware SS pin - pin 10 (on most Arduino boards) or pin 53 (on the Mega) - or another pin specified in the call to SD.begin().



Pin	SD	SPI
1	CD/DAT3	CS
2	CMD	DI
3	VSS1	VSS1
4	VDD	VDD
5	CLK	SCLK
6	VSS2	VSS2
7	DAT0	DO
8	DAT1	X
9	DAT2	X

Pin	SD	SPI
1	DAT2	X
2	CD/DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	X

Fig Micro SD card pin out

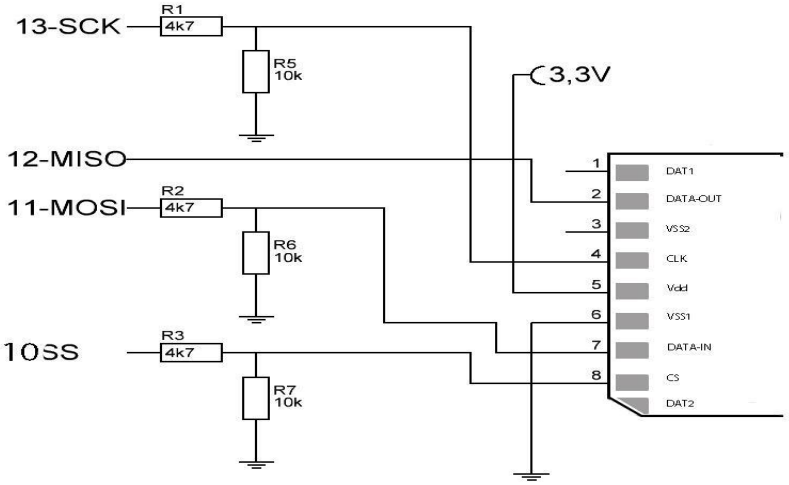


Fig SD card circuit diagram

1.7.1.5 RTC Interface

Duration of laser operation is very important to know because lifetime of a laser depends on how much hours it is used. It is also important to know which date

and time laser being used for a particular experiment which could give advantage for lab management.

I have used DS3231 RTC (Real time clock) module in laser controller for time and date tag and to count total hours it has used. The DS3231 RTC module integrates DS3231 Real Time Clock (RTC) IC which has an internal crystal and a switched bank of tuning capacitors. The temperature of the crystal is continuously monitored, and the capacitors are adjusted to maintain a stable frequency. Compared to other RTC solutions, this DS3231 module drifts less than a minute per year, even in extreme temperature ranges. This makes DS3231 module suits very well for time critical applications that cannot be regularly synchronized to an external clock.
The I2C interface to this module is very straightforward and virtually identical to the register address of the popular DS1337 and DS1307 RTCs.

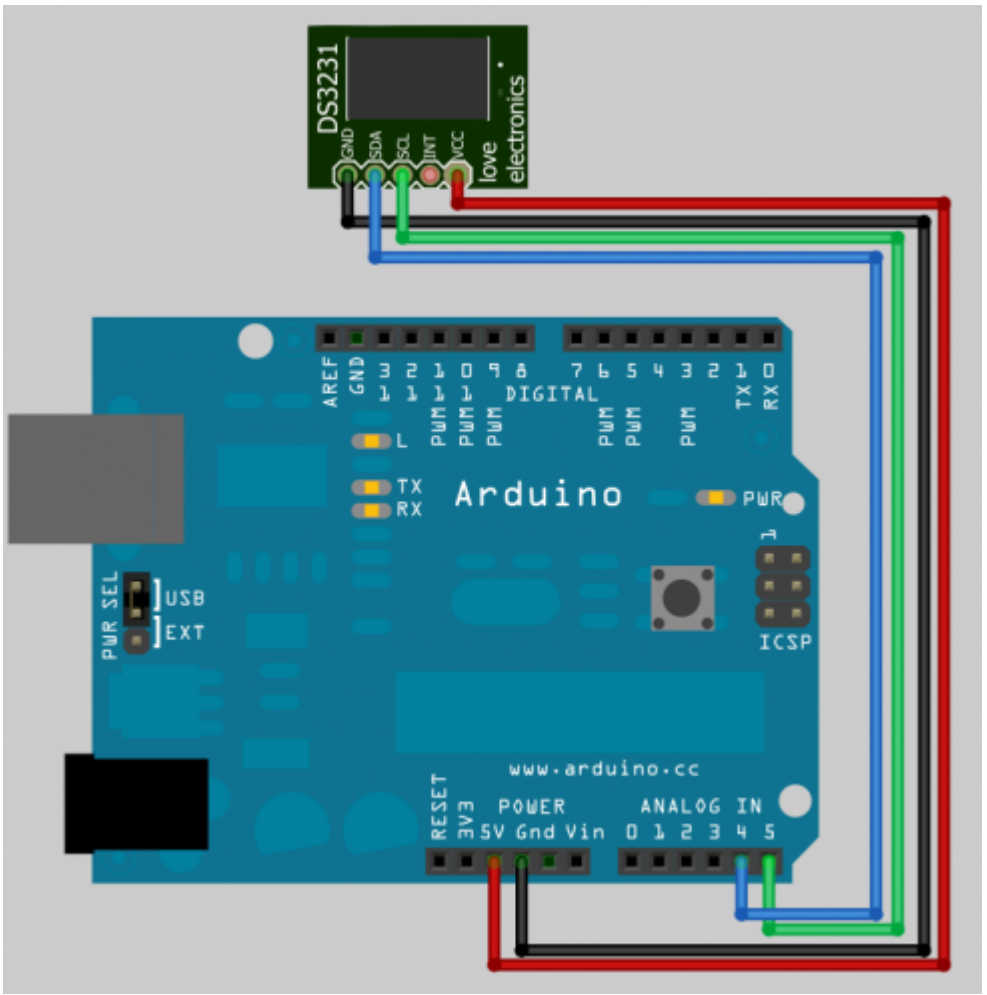
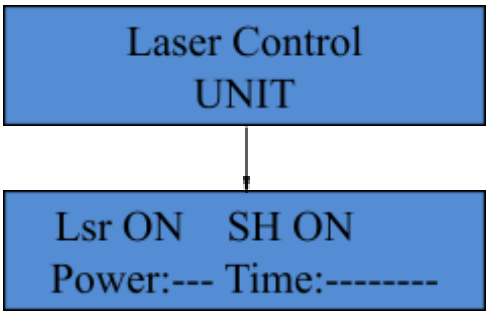


Fig: RTC Module with Arduino

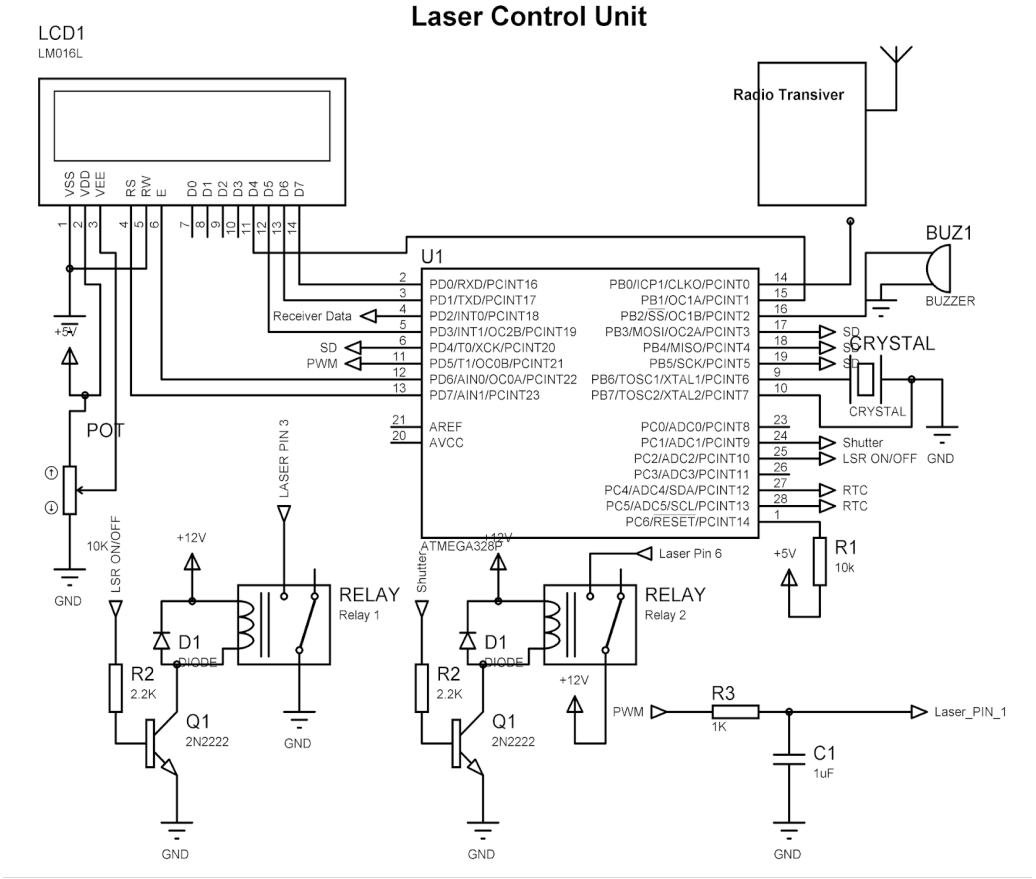
1.7.1.5 Menu Structure

Laser Control Unit:



(If Laser ON & SH ON command received from remote unit)

1.7.1.6 Control Unit Circuit Diagram



1.7.1.8 Control Unit Code

Please See Appendix

1.7.1.9 Power Supply

Total current consumption of laser control unit is only 0.3A. Any 12V power supply or adapter capable of delivering 0.5A is suitable for its operation. LM2576 voltage regulator IC inside the laser control unit distributes proper voltage at different sector of circuits. Laser should not be used as laser controller power source because it might not deliver enough current.

1.7.2.1. Wireless Control Mechanism

RF(Radio Frequency) module is used for wireless communication between laser control unit and remote control unit. An RF module (radio frequency module) is a (usually) small electronic circuit used to transmit and/or receive radio signals on one of a number of carrier frequencies. RF modules are widely used in electronic design owing to the difficulty of designing radio circuitry. Good electronic radio design is notoriously complex because of the sensitivity of radio circuits and the accuracy of components and layouts required to achieve operation on a specific frequency. Design engineers will design a circuit for an application which requires radio communication and then "drop in" a radio module rather than attempt a discrete design, saving time and money on development. I have used 433 MHz RF module to built this laser controller. Which specification is-

Transmitter Specs:
Working voltage: 3V~12V
Working current: max≤40mA (12V), min≤9mA(3V)
Resonance mode: sound wave resonance (SAW)
Modulation mode: ASK /OOK
Working frequency: 433MHz
Frequency error: +150kHz (max)
Velocity: ≤10Kbps
Self-owned codes: negative
Aerial Length: 18cm

Receiver Specs:
Working voltage: 5.0VDC +0.5V
Working current: ≤2.5mA (5.0VDC)
Working principle: superheterodyne
Working method: OOK/ASK
Operating frequency: 433MHz

We can send string using this module. As we are using same module to communicate with many devices we must set prefix for each packet of data. So there is no chance of miscommunication. Each step of laser power has different tag so if any data misses it synchronies automatically with next data. Laser control unit has data feedback capability to remote control unit so users always see in remote control unit display which truly executed.

1.7.2.2. Intelligent safety

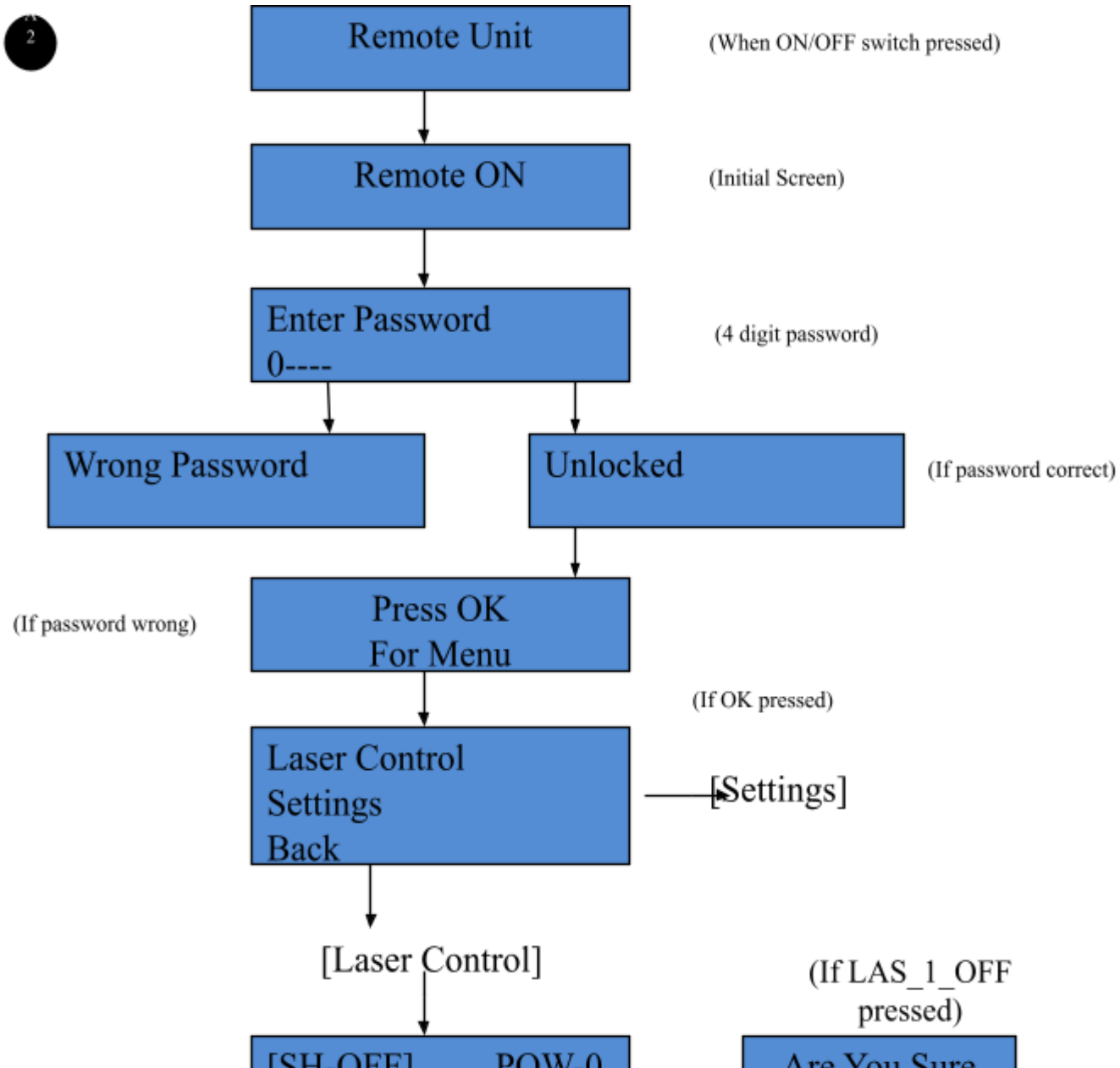
Each laser operates within some safety rules. User should maintain those rules strictly to conduct safe laser operation. This laser controller automatically maintains safety rules and do not allow user to skip safety rules. Control algorithm checks each command before execution. Such as if user wants to shut down laser without lowering laser power to zero laser control algorithm give warning to user and prevent laser from shutdown. Laser controller is password protected so unauthorized person will not operate laser. It has an option by which shutter can be controlled from external trigger.

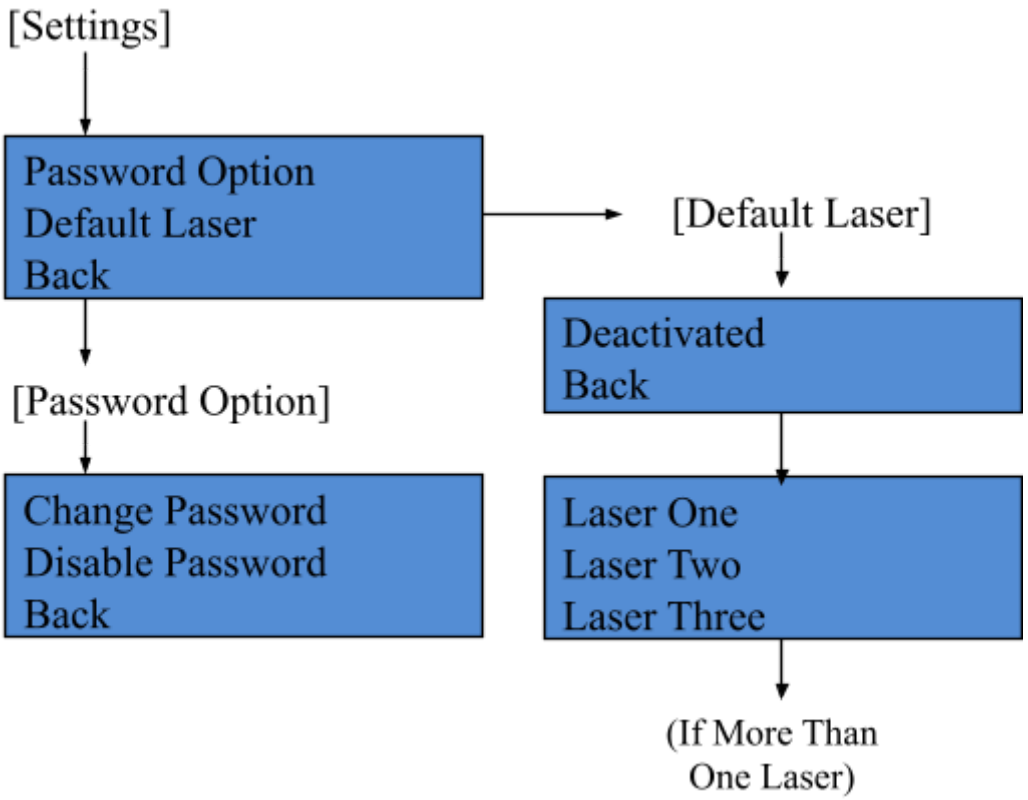
1.7.2.3 Multiple Laser Control

This laser controller has a unique feature which is multiple laser control system with same remote unit. It can connect with up to 3 laser control unit. User can lock any laser with password so that lengthy experiment will not interrupt by other user even though actual user not present in lab. During locking mode laser will run user pre selected mode.

1.7.2.4 Menu System

Remote Unit Menu Structure




[Settings]
Password Option
Default Laser
Back
[Default Laser]
Deactivated
Back
[Password Option]
Change Password
Disable Password
Back
Laser One
Laser Two
Laser Three
(If More Than
One Laser)

User can select any option by pressing UP or DOWN arrow key. Press OK button for Laser Shutter ON/OFF. Right arrow key will increase laser power and left arrow key decrease laser power. Press back to return previous menu.

1.7.2.5 Menu System and Algorithm

Menu is a system by which an electronic device can make with many features using only limited number of buttons. In our LASER controller, we have used a menu system or a unique operating system for making the device more configurable. We have added all the menus in the remote part only. We have added menus for controlling many LASERS using only one remote. In the menu system we also have added a password protection system. User can lock and unblock the device using the password lock system. An option has been added in the SETTINGS option in the MAIN menu to change the password. All the menu options are visualized in a LCD display. The display can display (16X2) = 32 characters (16 characters on each row) at a time. For that reason, only two options can be shown at a time in the display. The special property of the operating system is that, it takes very small amount of RAM memory when it operates. All the menus are controlled by using only five buttons. These are UP, DOWN, LEFT, RIGHT and OK. In the menu system we have used the blinking selection bar instead of solid square selection bar. It gives a beautiful graphical representation and also reduces the use of RAM memory.

1.7.2.6 Coding Technique

We have used here hardware programming for the controlling system. The device has two parts. One is the remote unit and the other is the LASER control

unit. All the menus have been given in the remote unit so that all the features of the LASER can be used by using the remote only. For that reason, all the difficulties have arisen in the coding part of the remote unit. We have made a unique operating system for the remote unit. Different menus have been given in the operating system for controlling different features of the LASER. Since there are many options in the main menu, we might be needed high RAM memory. But we made a special technique to reduce the RAM memory without reducing the menu options. The special technique is that we have made a virtual interrupt service routine. We did not use the conventional interrupt service routine because in that case the full control was not in our hand. All the menu options are controlled by five buttons and all the button states are checked in the virtual interrupt service routine (VISR). We have also reduced the RAM memory by using same variables for different menus. All the menu options have placed in different service routines so that any menus options can be execute from inside of any other menu option. The operating system is make by taking in mind that user can not do something wrong. Because of LASERS are very sensitive. One simple mistake can damage the LASER permanently. For instance, if a user shut down a LASER without regulating its power level to zero, then the bulb of the LASER could be fuse and hence cause a permanent damage of the LASER. For this reason we give an option in the LASER control menu that when a user trying to shut down the LASER using the remote without reducing the power level of the LASER, then the remote would not take any command and it also give an warning to the user to regulate the power to zero. The password system is given in the operating system for user protection. The user can only access the operating system using the password. The password system has been made by using EEPROM (Electrically Erasable Programmable Read Only Memory). To write the password, user does not need any keypad because he can write the password by using only UP and DOWN button. This system of password writing reduces the number of buttons, size, circuit and cost of the system.

1.7.2.7 Remote Unit Circuit Diagram

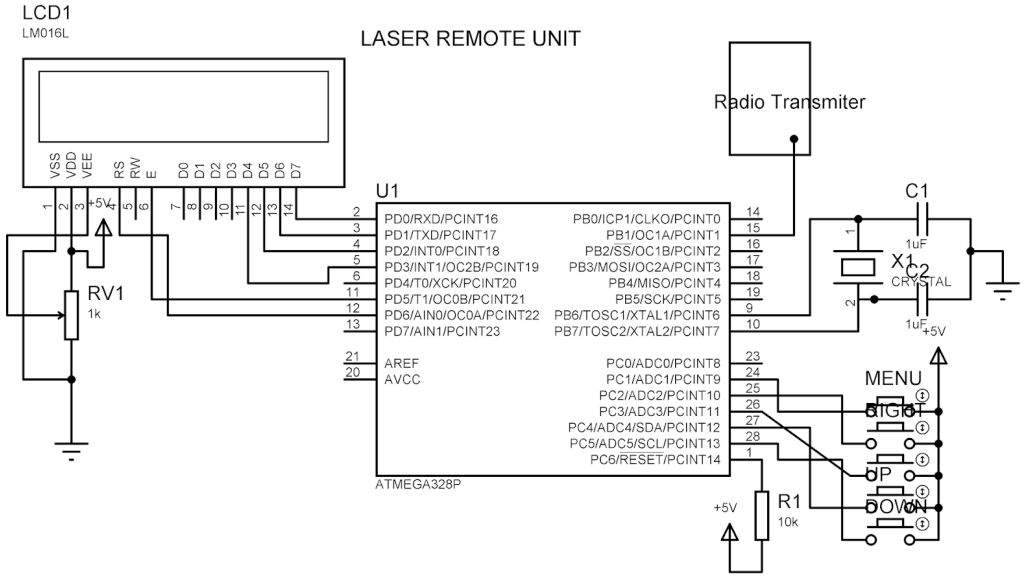


Fig 1.7.2.7 Remote Unit Circuit Diagram

1.7.2.8 Remote Unit Code

Plz See Appendix

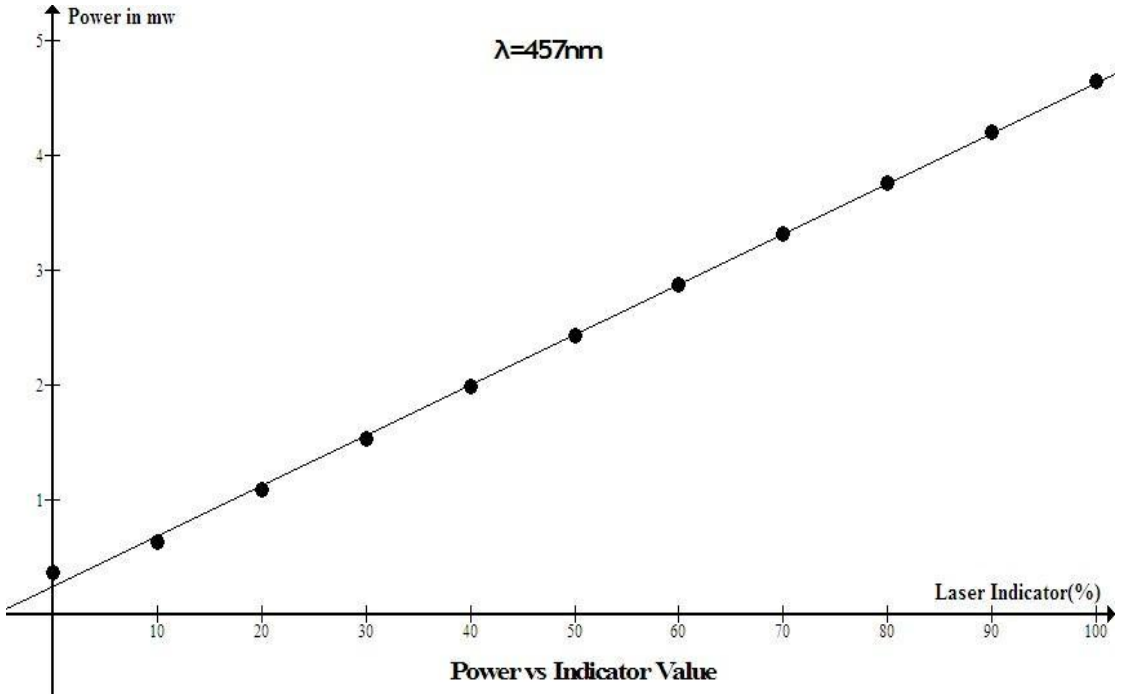
1.7.2.9 Power Supply

Remote Control Unit consumes less then 100ma current. So it can be operated with rechargeable Li-ion battery. Alternatively it has +12V power jack for adapter connection.

1.8 Experiments Using Laser Controller

Laser controller is tested with Ar-ion laser at three different wavelengths and graph is straight line as expected.
λ= 457nm

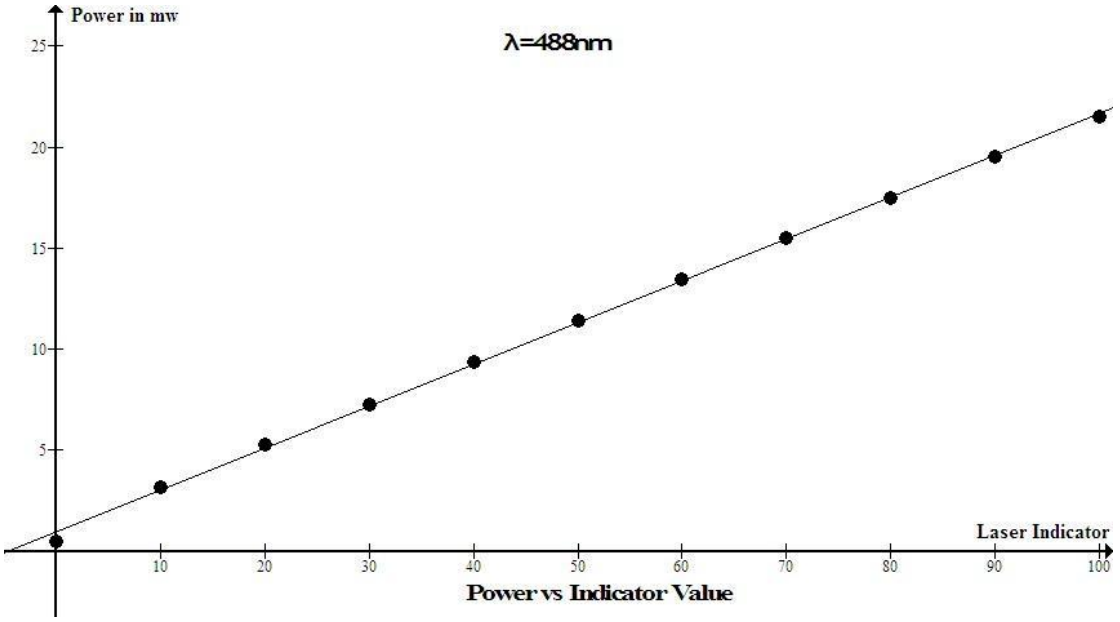
Controller Indicator (%)	Laser Power(mille watt)	Std. dev(micro watt)
0	0.368	2.9784
10	0.630	0.3265
20	1.086	1.8903
30	1.533	1.4437
40	1.982	2.2236
50	2.425	1.1493
60	2.868	1.0683
70	3.317	1.4082
80	3.761	1.9118
90	4.208	2.2376
100	4.643	2.0995



λ= 488nm

Controller Indicator (%)	Laser Power(mille watt)	Std. dev(micro watt)
0	0.460	41.645
10	3.195	15.822

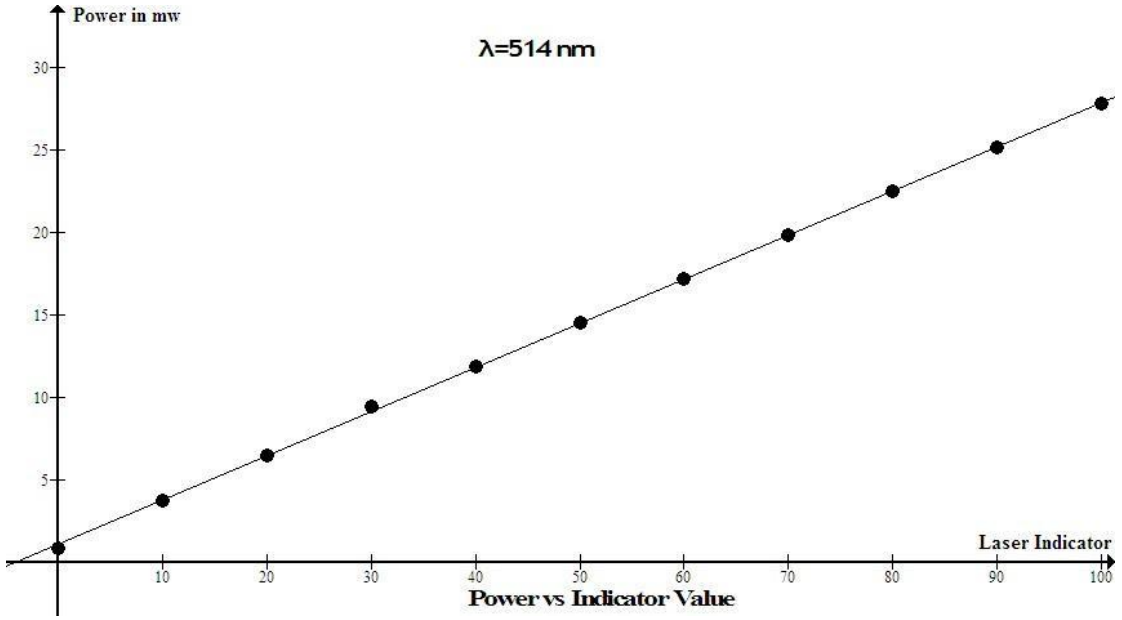
20	5.302	38.263
30	7.230	4.916
40	9.339	8.931
50	11.424	13.085
60	13.475	14.051
70	15.507	17.543
80	17.450	12.783
90	19.506	11.791
100	21.517	20.071



λ= 514nm

Controller Indicator (%)	Laser Power(mille watt)	Std. dev(micro watt)
0	0.887	1.524
10	3.760	3.882
20	6.469	2.897
30	9.59	2.897
40	11.844	6.079
50	14.524	6.460
60	17.183	5.251
70	19.801	6.986

80	22.461	7.565
90	25.128	5.792
100	27.815	1.977



1.9 Limitations and Future works

This version of laser controller device has some limitations. Such as wireless communication range is about 30 meters. Control unit has ports to install RTC and SD card but yet not installed. Controller do not support smart phone interfacing which could be a convenient way to control laser. External triggering system is limited only for shutter control.

So some steps should be taken in future such as

1) Xbee module for more reliable wireless communication.
2) Wifi module for smart phone interfacing.
3) Touch screen display to show graph in remote control unit.
4) Ethernet communication to interface laser with central network.
5) GSM/GPRS module to monitor and transfer data from remote distance.
6) Digital charging system
7) PC interfacing
8) Power meter interfacing with laser controller
9) Rotational stage controller interfacing with laser controller

10) Temperature sensor, humidity sensor, gas sensor for more safety.

Chapter Two
An Embedded System to Control Rotational Stage

2.0 Rotational Stage

Rotational stage is a mechanical device which can hold and rotate optical polarizer to perform many optical experiments. The optical element is secured inside central aperture of the platform of the holder by the threaded retaining ring. This enables the user to set the current position on the scale to any chosen angle independently of the position of the holder. Position is marked as degree from 0 to 360 degree. Angular position is read on 360° graduated **scale** on the platform, generally which is marked every **2°**. It can be rotate either manually or with motorized system. Motorized rotational stages are expensive but very convenient for experiment. In my thesis work I made a microcontroller based system to convert a manual rotational stage into a motorized rotational stage.

2.1 Purpose of Rotational Stage

Rotational stage is used in many experiments of physics to hold polarizer and to rotate polarizer in certain angel.

2.2 Embedded system to control rotational stage

Manual rotational stage has no control unit moreover it is not motorized. Rotating this stage with hand is a time consuming matter which waste huge time during experiment. Generally optical experiment performed in a dark or low light lab environment. So while we need to change the angel of polarizer we need to rotate knob of rotational stage which is not possible at low light condition. That time we need extra light source (i.e. Torch) to see the knob and change it to desire position. If we take data every **2°** rotation we need to touch the knob and click the torch 180 times. During sophisticated experiment light detector can detect light at nano watt range. So while we perform sophisticated experiment photon from torch light bounce in lab and detect by detector which makes the system unstable which causes error in experiment. So we need a motorized rotational stage and controller to solve this problem and make experiments better. I made a motorized rotational stage using a manual rotational stage (Newport, Model: RSP-1T) and develop a control unit to make it a complete embedded system to control rotational stage.

2.3 Total System Architecture

Total system consists of three major parts.

1) Mechanical Unit
2) Motor Driver
3) Control Unit

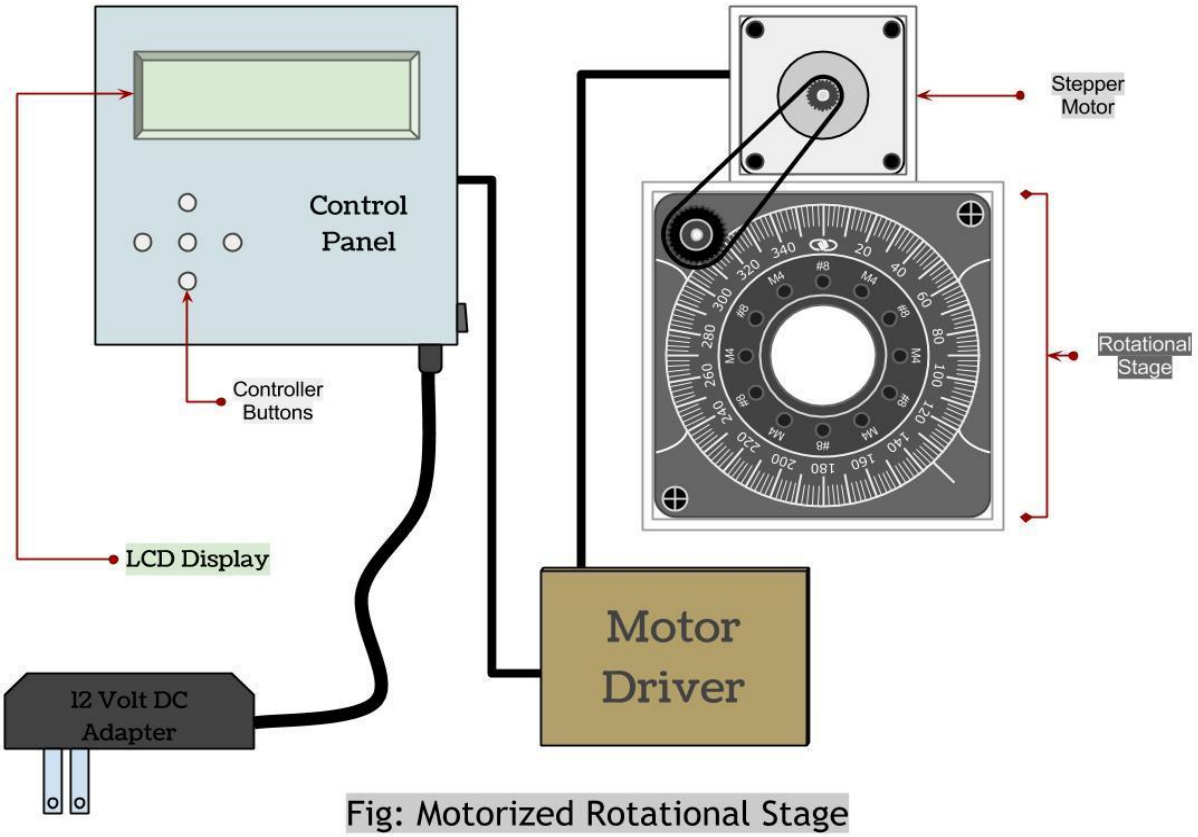


Fig: Motorized Rotational Stage

A brief description of three parts given below

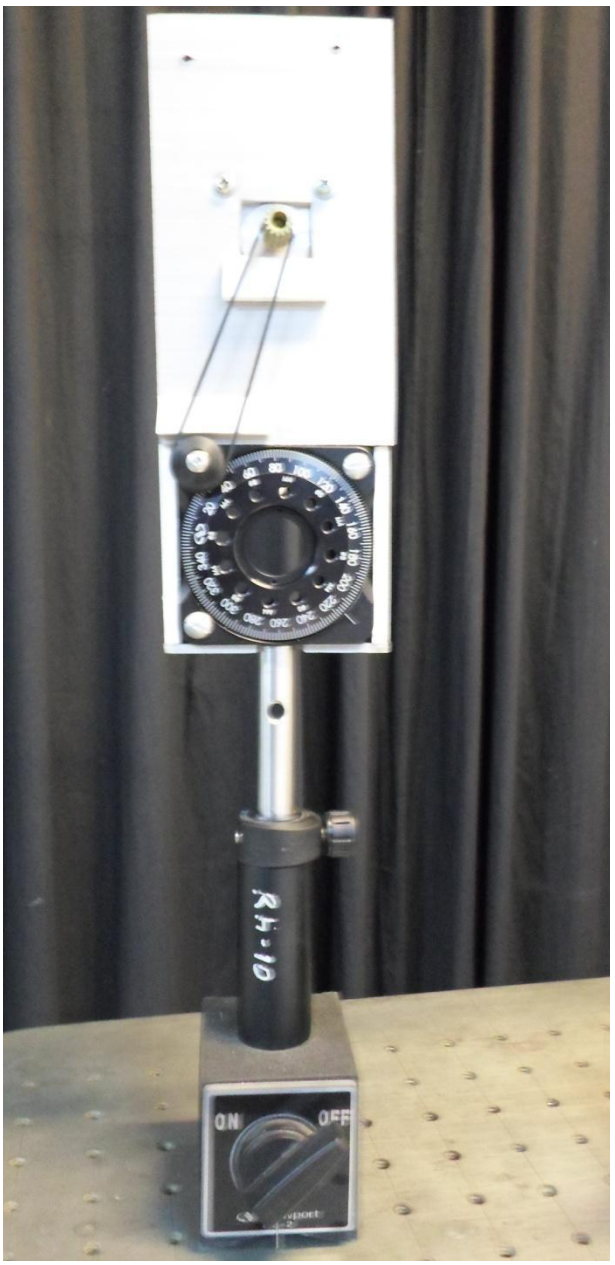


Fig 2.3 Motorized Rotational Stage in LAB

2.3.1 Mechanical Unit

I converted a Newport RSP 1T manual rotational stage into motorized rotational stage. A bipolar Stepper motor is used for precise control. Motor mounted above the main frame of rotational stage. RSP 1T manual rotational stage is very expensive so my target was made it motorized without changing its main mechanical design. 5mm PVC sheet is used to attach motor with main frame. Bipolar stepper motor connected with rotational stage by a belt. Engineering drawing before motorized is given below

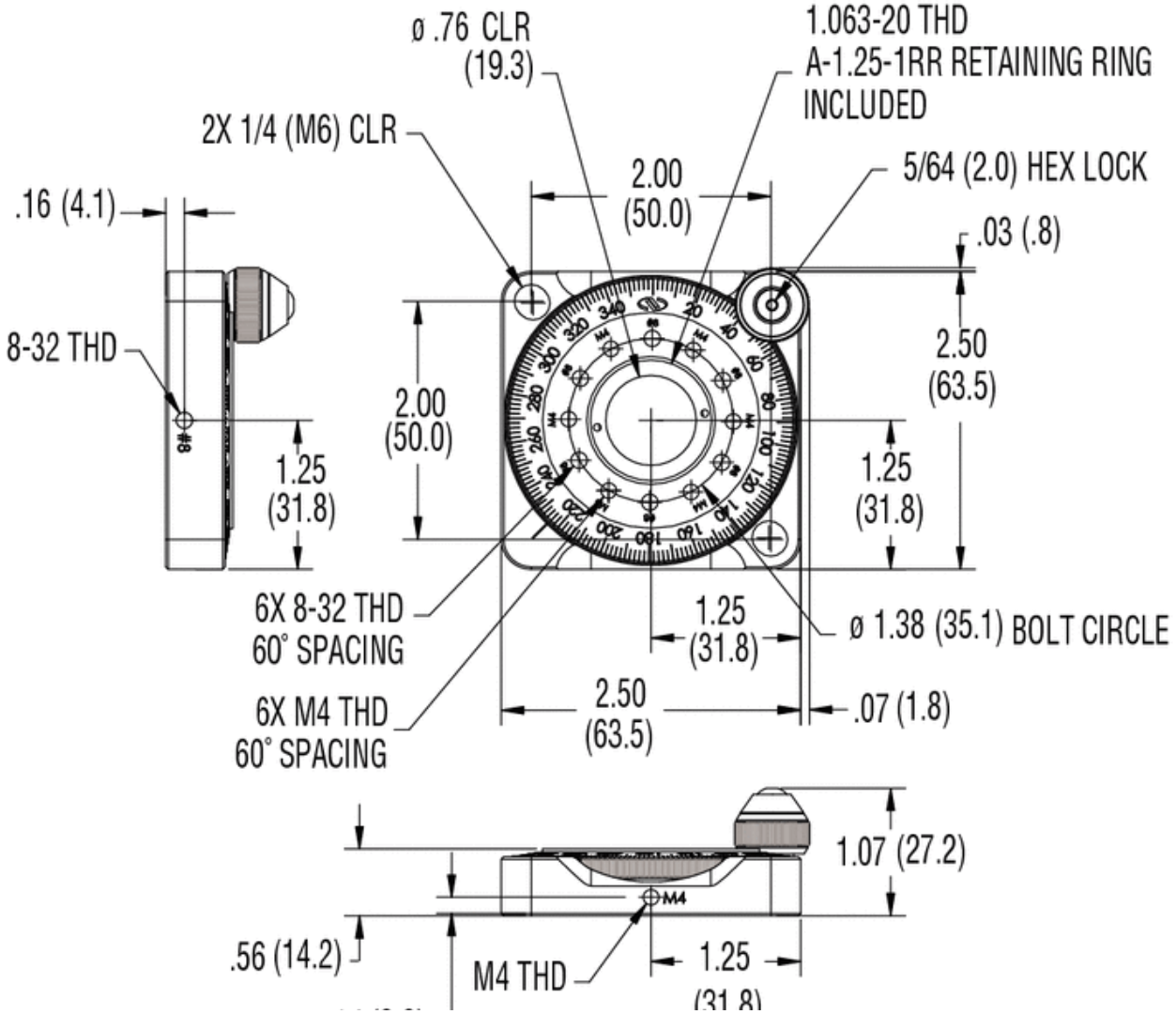


Fig 2.3(a) Engineering Drawing of Manual Rotational Stage

I just added a bipolar stepper motor model: STH-39D1126-06 which is mounted using PVC sheet and screw without changing main design of RSP-1T. Engineering drawing after motorized is given below

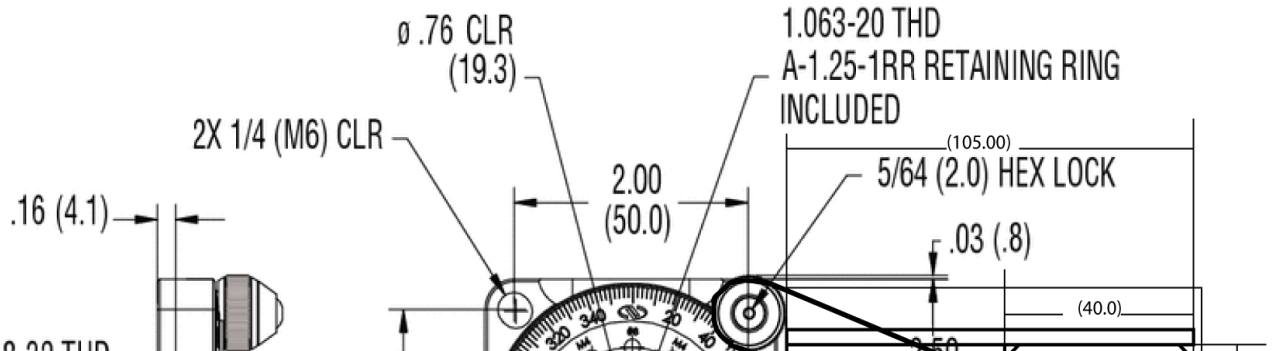


Fig 2.3(b) Engineering Drawing after Motorized

A brief description of RSP-1T Rotational Stage and Bipolar stepper motor is given below.

2.3.1.1 Newport RSP-1T Rotational Stage

The RSP-1T Rotation Stage features full ball bearing races, provides exceptionally smooth and continuous angular positioning for 1 inch (25.4 mm) diameter optics or components. A delrin retaining ring is provided to secure the optic. The rotating platform has six 8-32 and six M4 threaded holes for added mounting flexibility. Like the RM25A, the RSP-1T has an internal thread that enables Newport Glan-laser calcite polarizer and CH Series cube beam splitter holders to be rotated in this stage. Polarizer or wave plates can be coarsely aligned using the knurled edge of the rotating platform while fine adjustment is achieved with the knob. Angular position is indicated on a 360 ° scale graduated in 2 ° increments. The adjustment knob can be securely locked by tightening the screw in the end of the knob.

Model	**RSP-1T**
Compatibility	Universal
Stage Type	Optic Rotator
Load Capacity	10 lb (45 N)

Sensitivity	4 arc min
Travel, Coarse	360 °
Bearings	Ball
Resolution	4 arc min
Diameter	2.5 in.
Diameter	63.5 mm
Material	Aluminum
Maximum Drive Torque	10 in-lb
Maximum Drive Torque	1.2 Nm
Graduations	2 °
Clear Aperture	0.75 in.
Optic Diameter	1.0 in. (25.4 mm)
Optic Thickness	0.33 in. (8.0 mm)
Sensitivity	240 arc sec

Table 1.1: Specification of **RSP-1T**

2.3.1.2 Motor Unit

Stepper motor

A **stepper motor** (or **step motor**) is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any feedback sensor (an open-loop controller), as long as the motor is carefully sized to the application.

DC brush motors rotate continuously when voltage is applied to their terminals. Stepper motors, on the other hand, effectively have multiple "toothed" electromagnets arranged around a central gear-shaped piece of iron. The electromagnets are energized by an external control circuit, such as a microcontroller. To make the motor shaft turn, first, one electromagnet is given power, which magnetically attracts the gear's teeth. When the gear's teeth are aligned to the first electromagnet, they are slightly offset from the next electromagnet. So when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated. Each of those rotations is called a "step", with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle.

Bipolar Stepper Motor

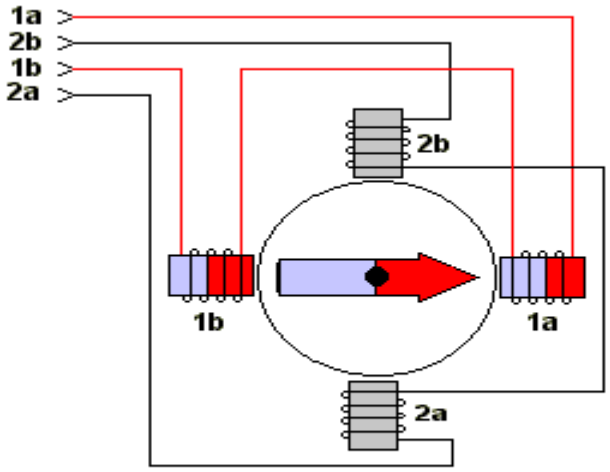
Bipolar motors have a single winding per phase. The current in a winding needs to be reversed in order to reverse a magnetic pole, so the driving circuit must be more complicated, typically with an H-bridge arrangement (however there are

several off-the-shelf driver chips available to make this a simple affair). There are two leads per phase, none are common.

Static friction effects using an H-bridge have been observed with certain drive topologies.

Dithering the stepper signal at a higher frequency than the motor can respond to will reduce this "static friction" effect.

Because windings are better utilized, they are more powerful than a unipolar motor of the same weight. This is due to the physical space occupied by the windings. A unipolar motor has twice the amount of wire in the same space, but only half used at any point in time, hence is 50% efficient (or approximately 70% of the torque output available). Though a bipolar stepper motor is more complicated to drive, the abundance of driver chips means this is much less difficult to achieve.



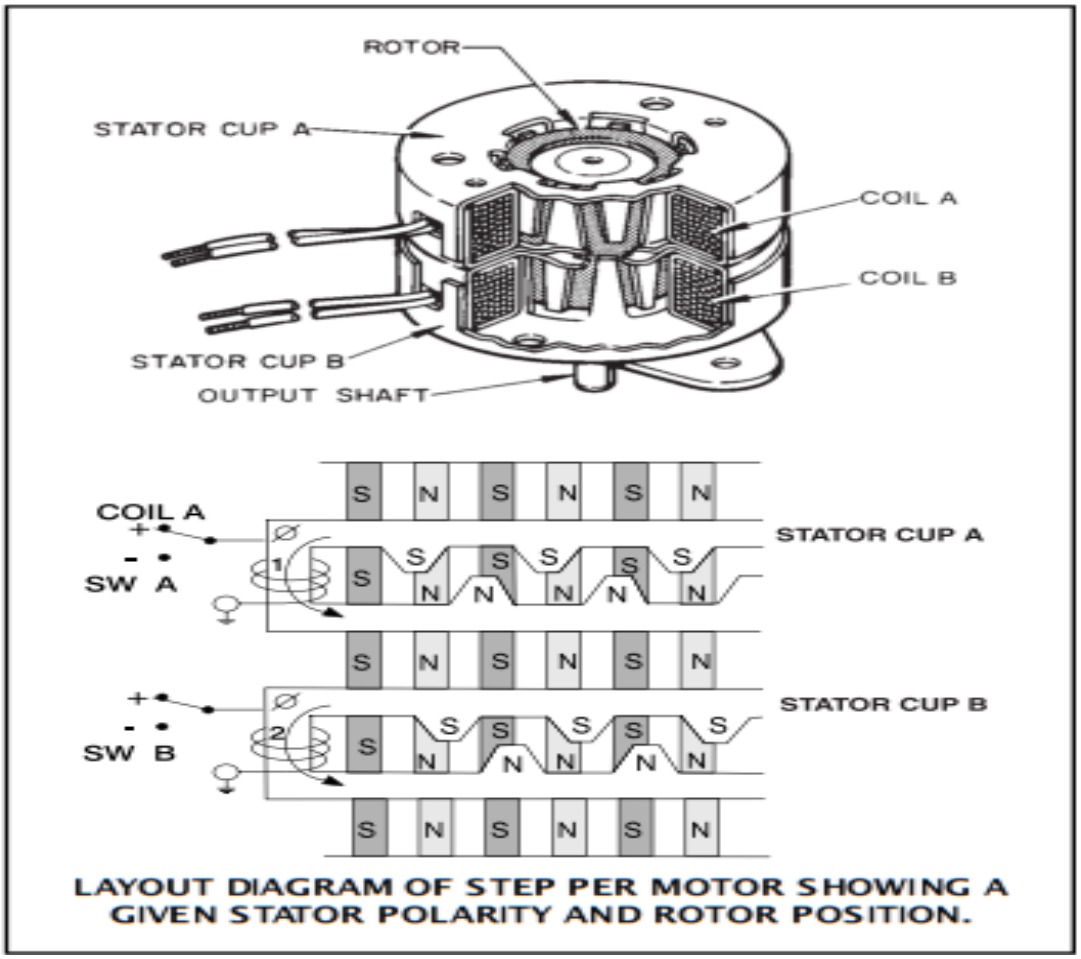
Conceptual Model of Bipolar Stepper Motor

Fig 2.3.1.2(a) Conceptual Model of Bipolar Stepper Motor

CONSTRUCTION

In a typical motor, electrical power is applied to two coils. Two stator cups formed around each of these coils, with pole pairs mechanically displaced by 1/2 a pole pitch, become alternately energized North and South magnetic poles. Between the two stator-coil pairs, the displacement is 1/4 of a pole pitch. The permanent magnet rotor is magnetized with the same number
Of pole pairs as contained by the stator-coil section. Interaction between the rotor and stator (opposite poles attracting and likes repelling) causes the rotor to

move 1/4 of a pole pitch per winding Polarity change. A 2-phase motor with 12 pole pairs per stator-coil section would thus move 48 steps per revolution or 7.5° per step.



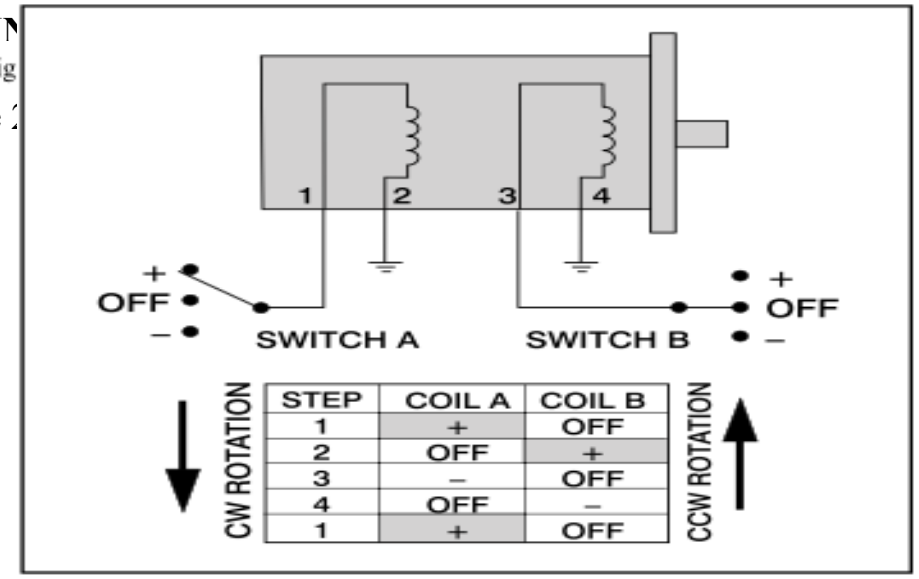
LAYOUT DIAGRAM OF STEP PER MOTOR SHOWING A GIVEN STATOR POLARITY AND ROTOR POSITION.

ELCTRICAL IN

The normal e

Fig

shown in Figure



STEP	COIL A	COIL B
1	+	OFF
2	OFF	+
3	–	OFF
4	OFF	–
1	+	OFF

Fig 2.3.2.1(c) Motorized Rotational Stage in LAB

Continuing the sequence causes the rotor to rotate forward. Reversing the sequence reverses the direction of rotation. Thus, the stepper motor can be easily controlled by a pulse input drive which can be a 2-flip-flop logic circuit operated either open or closed loop. Operated at a fixed frequency, the electrical input to the motor is a 2-phase 90° shifted square wave as shown below in Fig. 2.3.2.1(c). Since each step of the rotor can be controlled by a pulse input to a drive circuit, the stepper motor used with modern digital circuits, microprocessors and transistors provides accurate speed and position control along with long life and reliability.

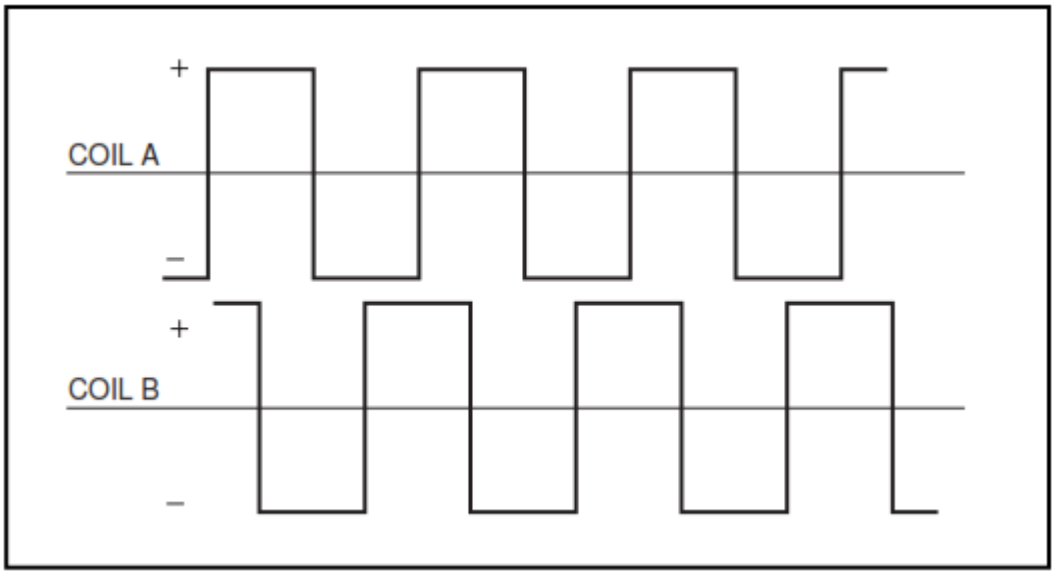


Fig 2.3.2.1(c): Voltage Wave Form- Fixed Frequency- 4 Step Sequence

Step angles for steppers are available in a range from .72° to 90°.
Standard step angles for Thomson Airpax steppers are:
1.8º — 200 steps per rev.
3.6º — 100 steps per rev.
7.5° — 48 steps per rev.
15° — 24 steps per rev.
18° — 20 steps per rev.
A movement of any multiple of these angles is possible. For example, six steps of a 15° stepper motor would give a movement of 90°.

ACCURACY

A 7.5° stepper motor, either under a load or no load condition, will have a step-to-step accuracy of 6.6% or 0.5º. This error is noncumulative so that even after making a full revolution, the position of the rotor shaft will be 360º ±0.5º.

The step error is noncumulative. It averages out to zero within a 4-step sequence which corresponds to 360 electrical degrees. A particular step characteristic of the 4-step is to sequence repeatedly using the same coil, magnetic polarity and flux path. Thus, the most accurate movement would be to step in multiples of four, since electrical and magnetic imbalances are eliminated. Increased accuracy also results from movements which are multiples of two steps. Keeping this in mind, positioning applications should use 2 or 4 steps (or multiples thereof) for each desired measured increment, wherever possible.

TORQUE

The torque produced by a specific stepper motor depends on
Several factors:

1) The Step Rate
2) The Drive Current Supplied to the Windings
3) The Drive Design

HOLDING TORQUE

At standstill (zero steps/sec and rated current), the torque required to deflect the rotor a full step is called the holding torque. Normally, the holding torque is higher than the running torque and, thus, acts as a strong brake in holding a load. Since deflection varies with load, the higher the holding torque the more accurate the position will be held. Note in the curve below in Fig. 2.3.2.1(d), that a 2-step deflection corresponding to a phase displacement of 180, results in zero torque. A 1-step plus or minus displacement represents the initial lag that occurs when the motor is given a step command.

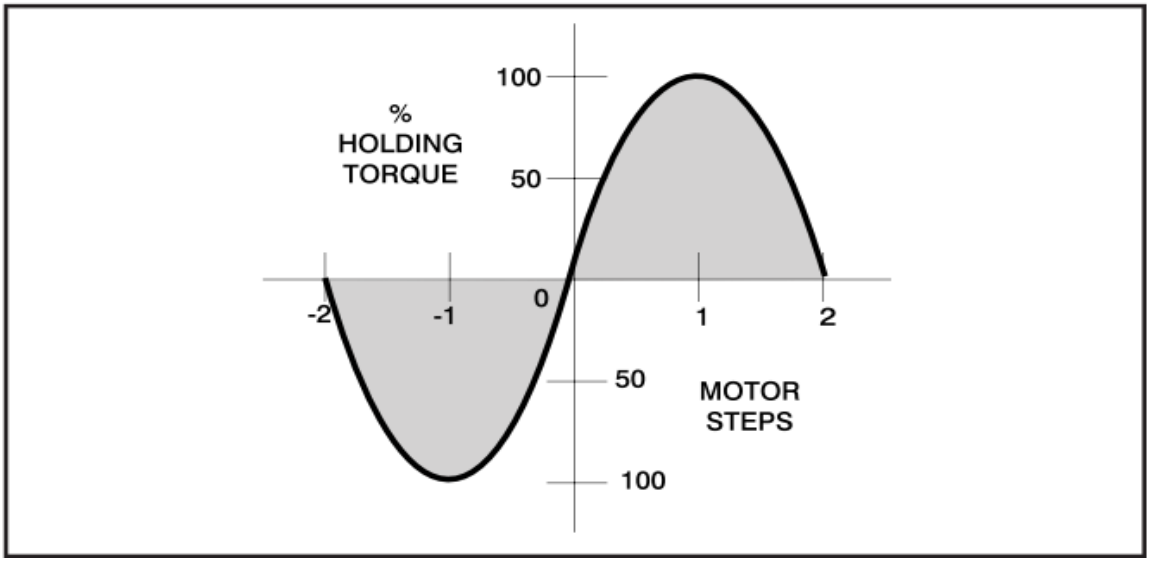


Fig 2.3.2.1(d): Voltage Wave Form- Fixed Frequency- 4 Step Sequence

RI

The non-energized detent torque of a permanent magnet stepper motor is called residual torque. A result of the permanent magnet flux and bearing friction, it has a value of approximately 1/10 the holding torque. This characteristic of PM steppers is useful in holding a load in the proper position even when the motor is de-energized. The position, however, will not be held as accurately as when the motor is energized.

DYNAMIC TORQUE

A typical torque versus step rate (speed) characteristic curve is shown in Figure 2.3.2.2(d). The PULL IN curve shows what torque load the motor can start and stop without loss of a step when started and stopped at a constant step or pulse rate. The PULL OUT curve is the torque available when the motor is slowly accelerated to the operating rate. It is thus the actual dynamic torque produced by the motor. The difference between the PULL IN and PULL OUT torque curves is the torque lost due to accelerating the motor rotor inertia. The torque or speed characteristics curves are key to selecting the right motor and control drive method for a specific application.

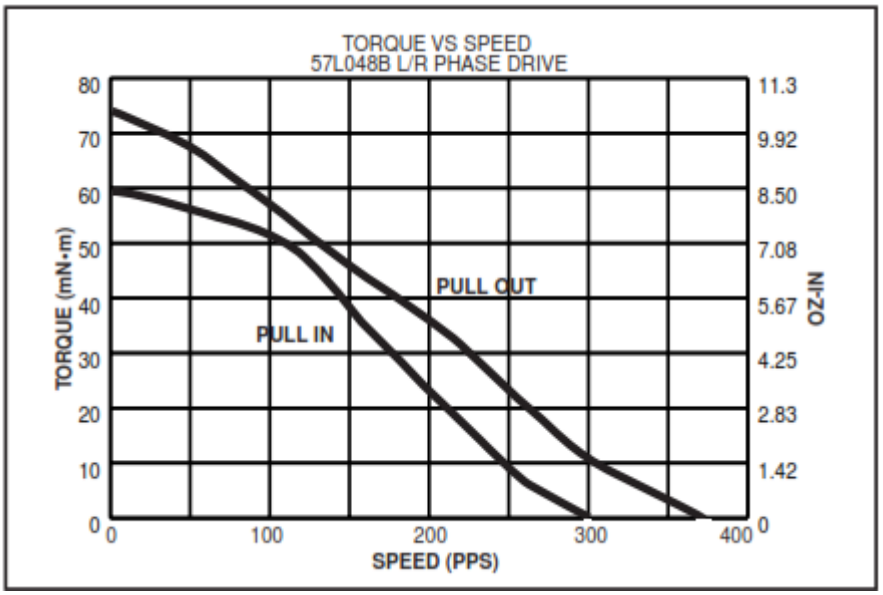


Fig 2.3.2.1(d): Torque vs Speed Curve (Thomson Airpex 57L048B L/R Stepper)

Various Stepper Motor Operations (Unipolar and Bipolar)

The stator flux with a Bipolar winding is reversed by reversing the current in the winding. It requires a push-pull Bipolar drive as shown in Fig. 2.3.2.1(e). Care must be taken to design the circuit so that the transistors in series do not short the power supply by coming on at the same time. Properly operated, the

Bipolar winding gives the optimum motor performance at low-to-medium step rates.
A Unipolar winding has two coils wound on the same bobbin (one bobbin resides in each stator half) per stator half. Flux is reversed in each coil bobbin assembly by sequentially grounding ends of each half of the coil winding. The use of a Unipolar winding, sometimes called a bifilar winding, allows the drive circuit to be simplified. Not only are half as many power switches required (4 vs.8), but the timing is not as critical to prevent a current short through two transistors as is possible with a Bipolar drive. For a Unipolar motor to have the same number of turns per winding as a Bipolar motor, the wire diameter must be decreased and the resistance increased. As a result, Unipolar motors have 30% less torque at low step rates. However, at higher rates the torque outputs are equivalent.

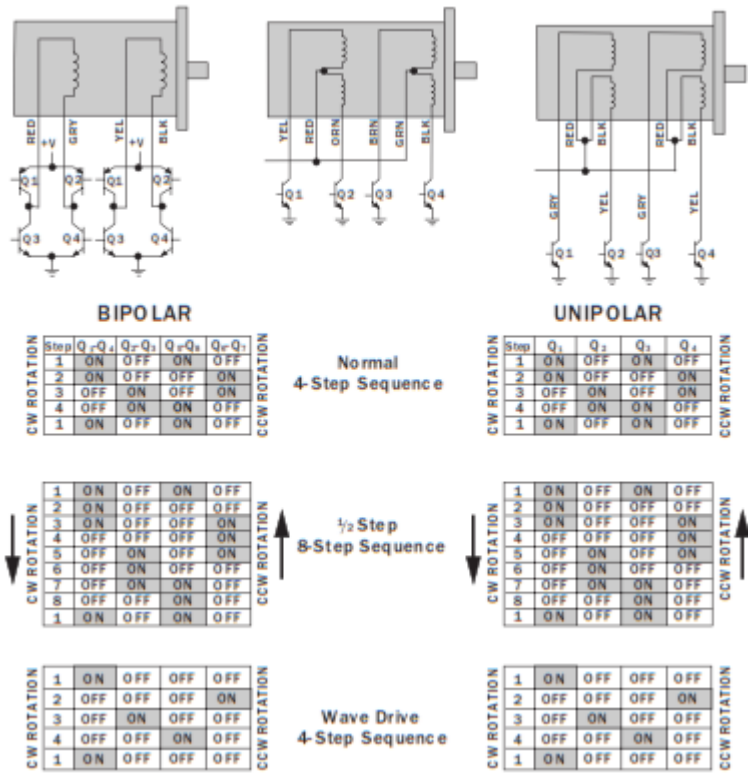


Fig 2.3.2.1(d): Bipolar and Unipolar Switching Sequence

2.3.1 Motor Driver

To control bipolar stepper motor I have used conventional H-bridge motor driver. Bipolar stepper motor Model: STH-39D1126-06 has 4.1ohms resistance each coil and it need 12Volts to run. So we need 2.93 Amps current for each coil

so almost 6 Amps current needed for total operation. First I made a H-bridge motor driver using TIP-126 and TIP-127 transistor but they terribally heated. Success came when I used MOSFET. MOSFET is excellent for high power operation. A brief description of H-bridge MOSFET driver is given below.

2.3.2.1 H bridge

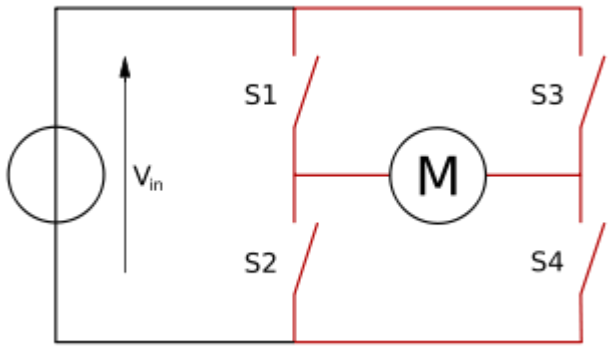


Fig 1.9: Basic H-bridge Circuits

An H bridge is an electronic circuit that enables a voltage to be applied across a load in either direction. These circuits are often used in robotics and other applications to allow DC motors to run forwards and backwards.

Most DC-to-AC converters (power inverters), most AC/AC converters, the DC-to-DC push–pull converter, most motor controllers, and many other kinds of power electronics use H bridges. In particular, a bipolar stepper motor is almost invariably driven by a motor controller containing Two H Bridges.

2.3.2.2 MOSFET Driven H-bridge Circuit

I used both P-Channel and N-Channel MOSFET. Four IRF540Z and four IRF9640 is used. Additional four npn transistor (BC547) is used to drive IRF 9640 P-Channel MOSFET. Protius schematic of MOSFET H-bridge is given below.

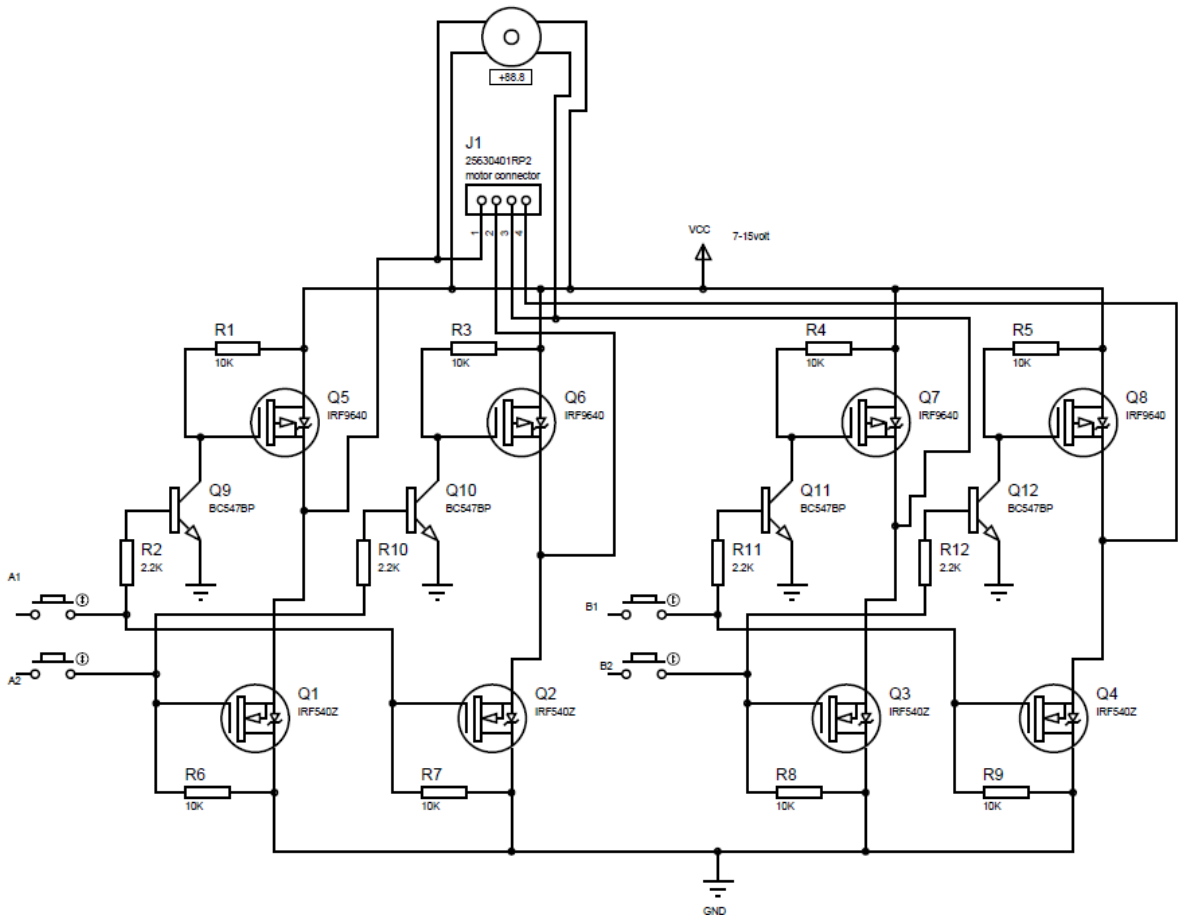


Fig 2.3.3.2: Implemented H-bridge Circuit for Rotational Stage

2.3.3 Control Unit

Control unit is the brain of a motorized mechanical stage. Control algorithm is installed in control unit. Microcontroller does all jobs here. Atmel Atmega328 microcontroller is used. A 16*2 LCD display used for menu browsing and for show result. Several options added in MENU. From control unit we can select amount of angle we want to rotate. If we need to calibrate there is a reset option so that we can manually rotate it and adjust it. Rotation is possible towards both directions. Microcontroller receives input from user, process it and send it to motor driver for particular task. Control unit has memory so it saves settings and how much degree it moves. It allow user to choose how much angle of rotation user needs for single button press. Control unit circuit diagram is given below.

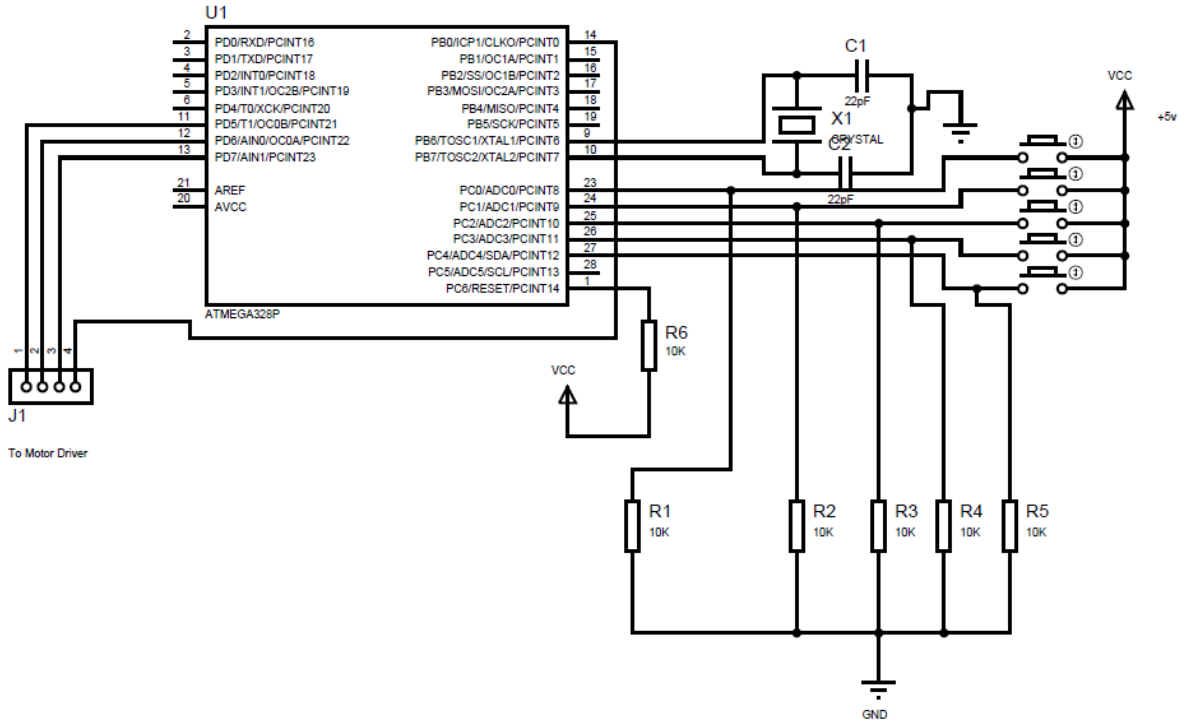
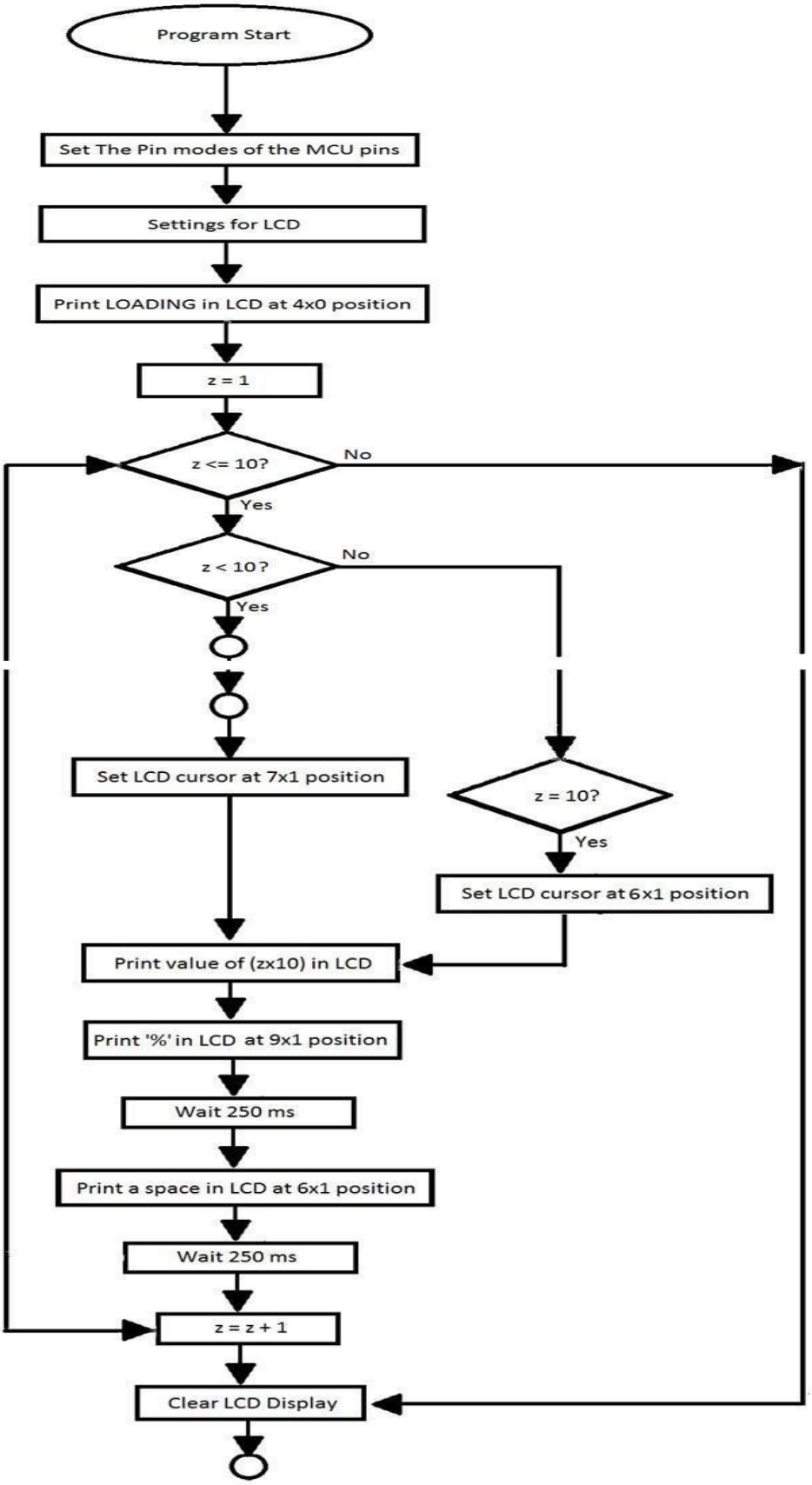
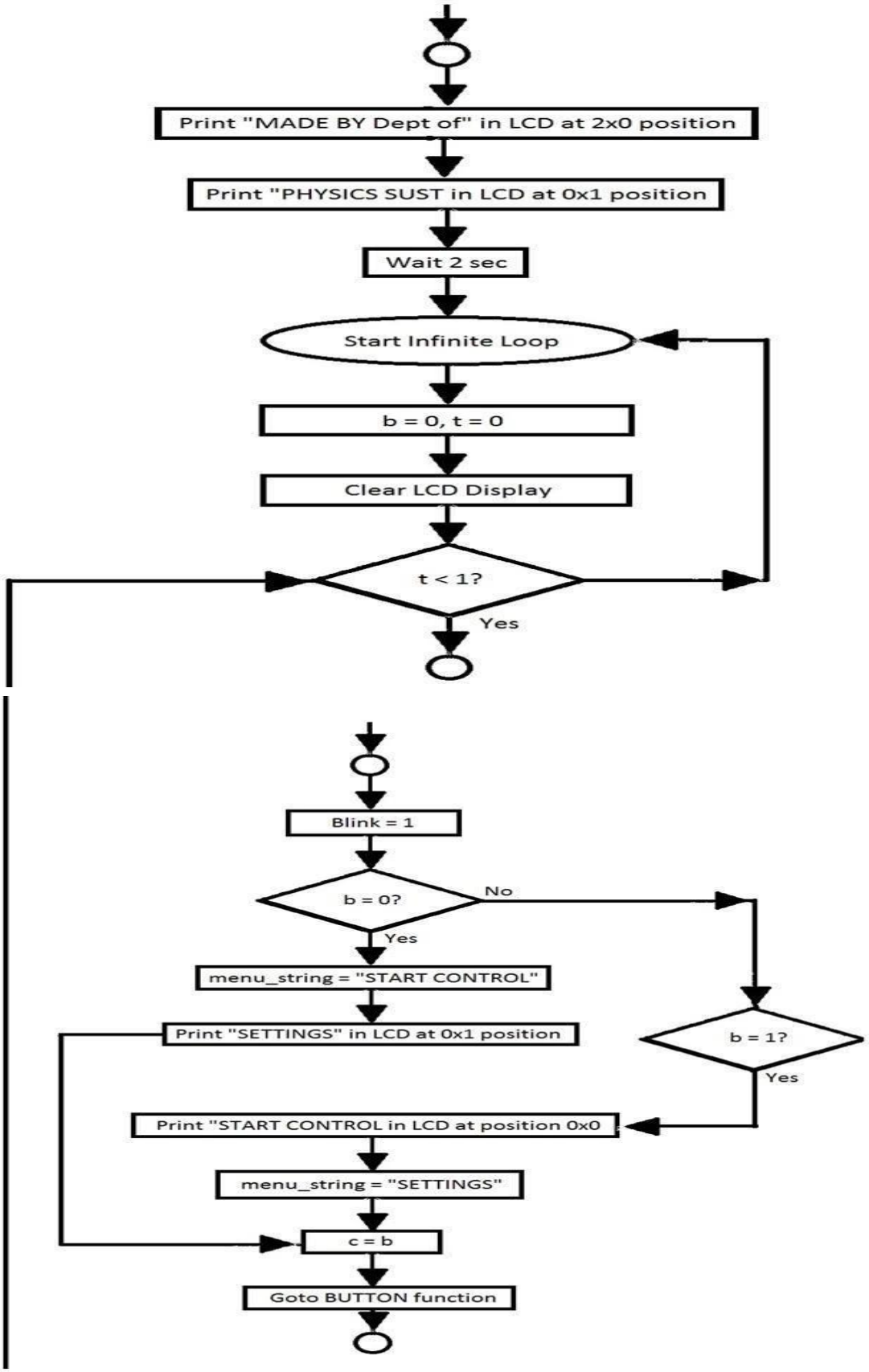
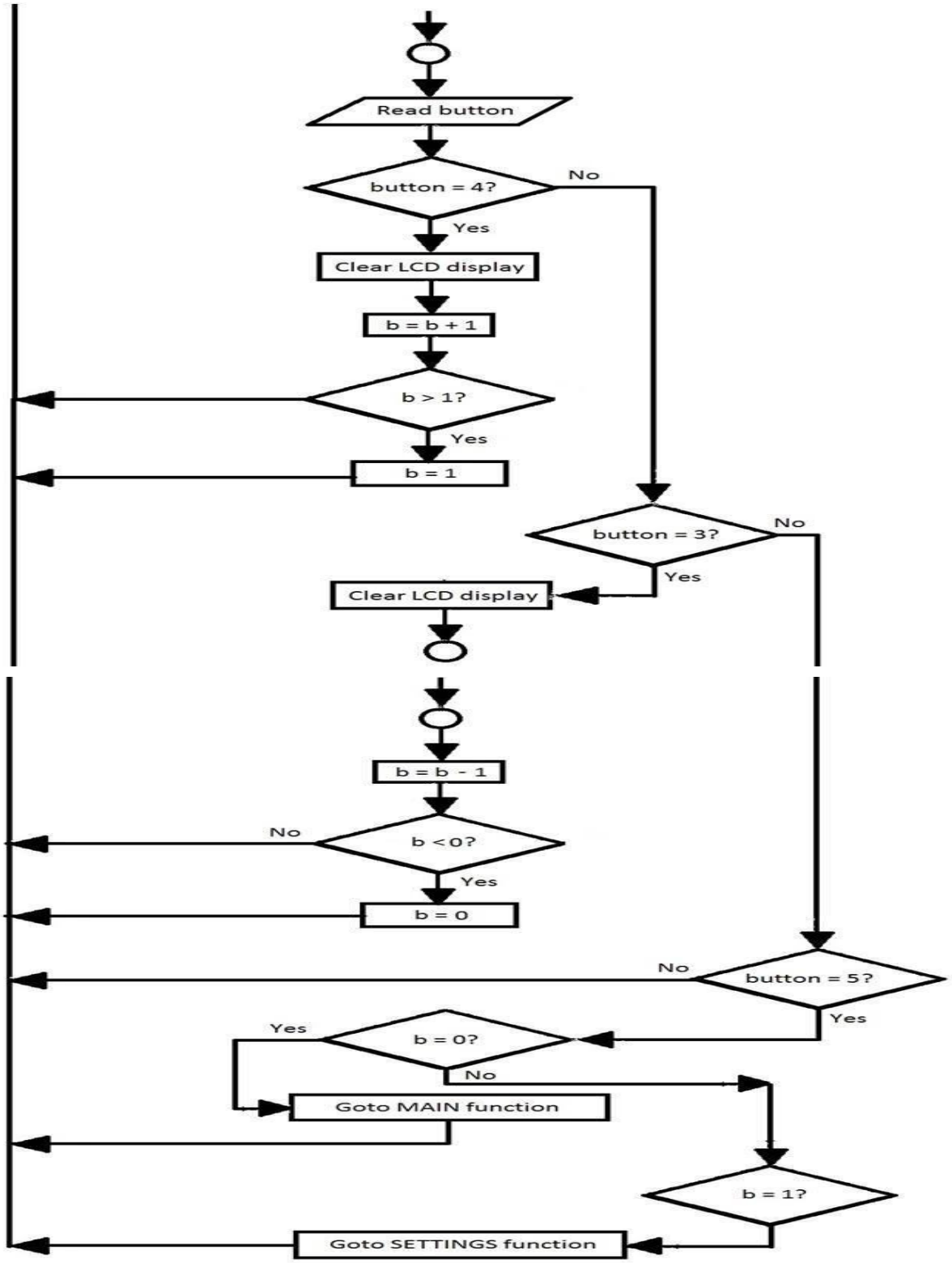


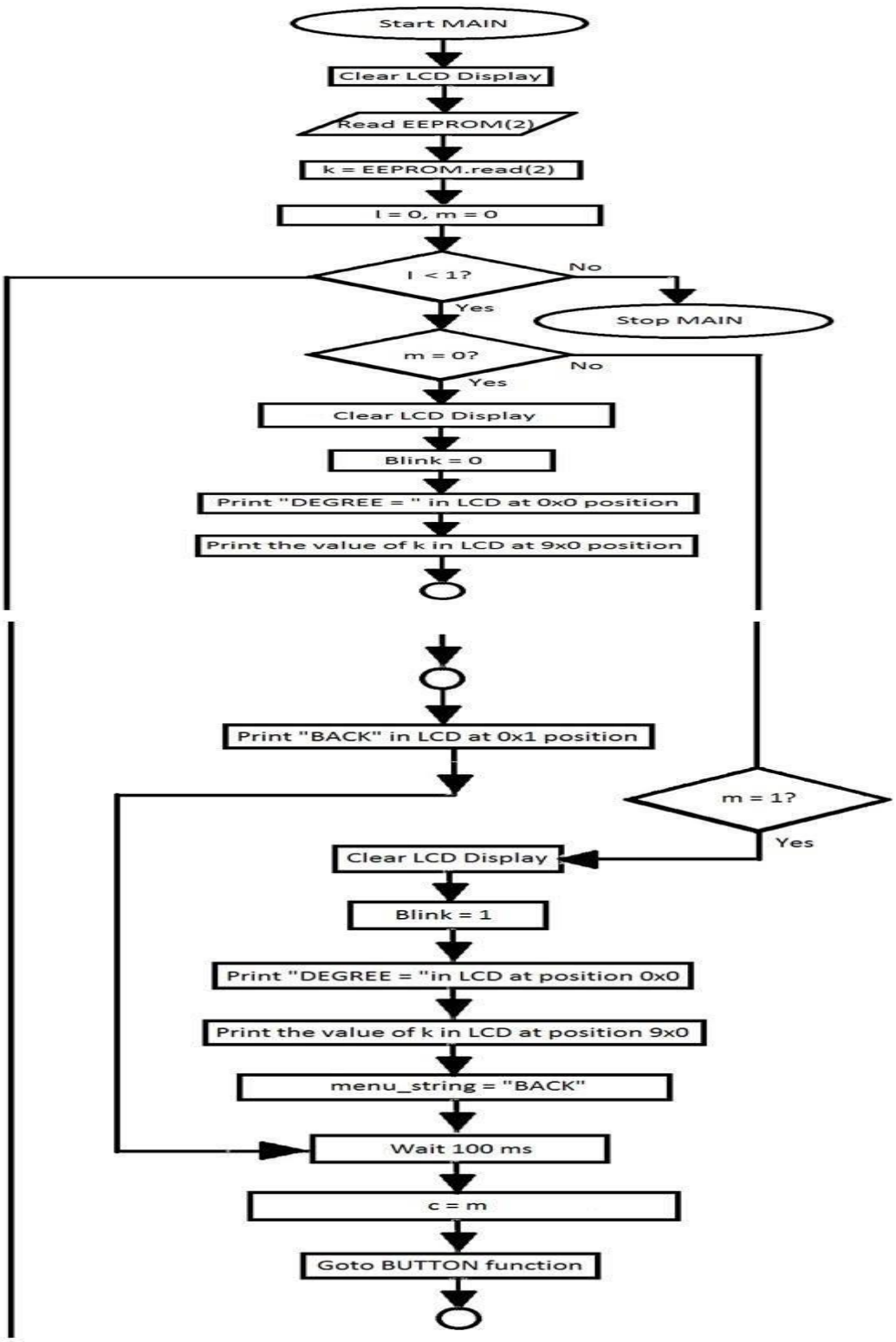
Fig 2.3.3: Control Unit Circuit

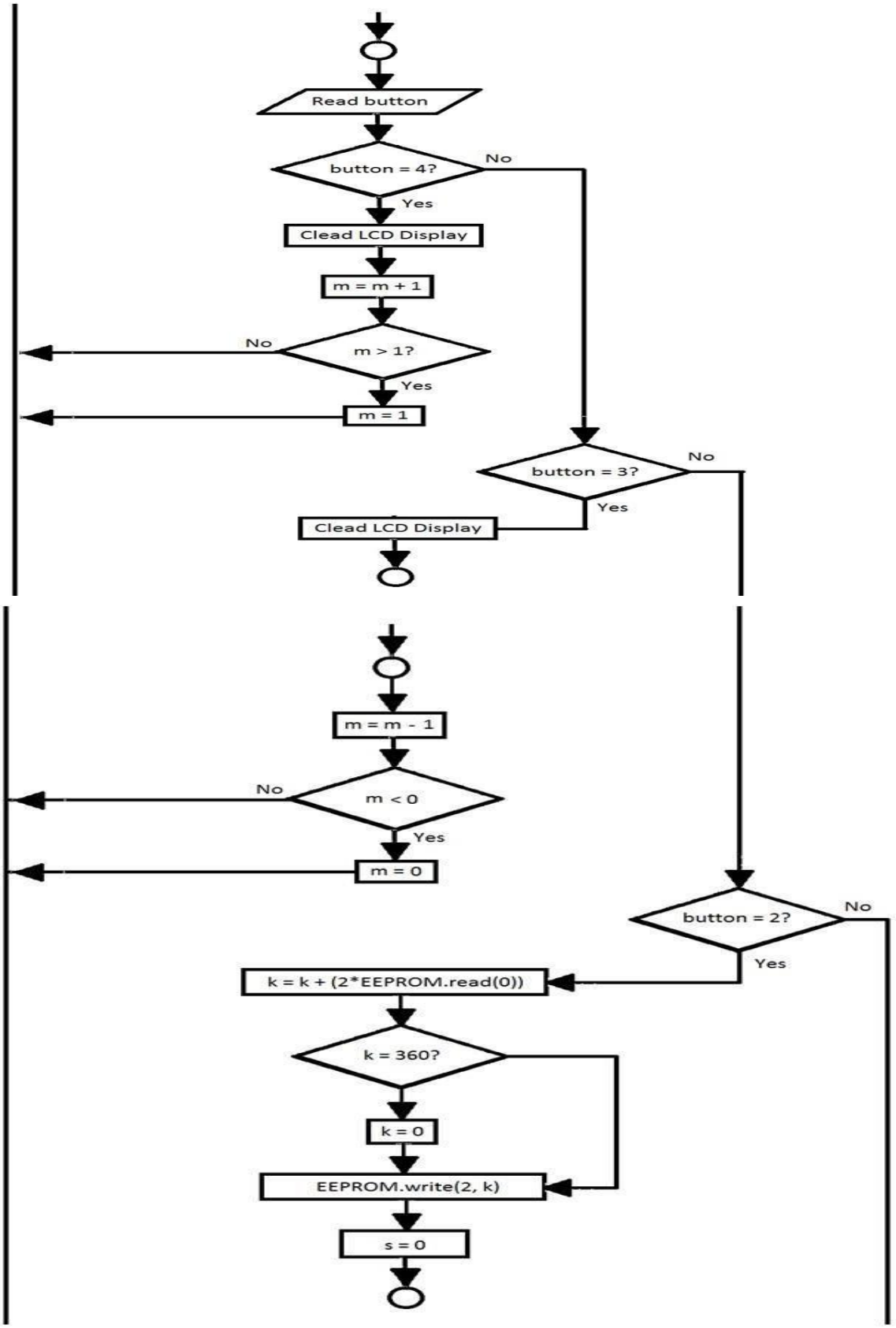
2.4 Flow Chart

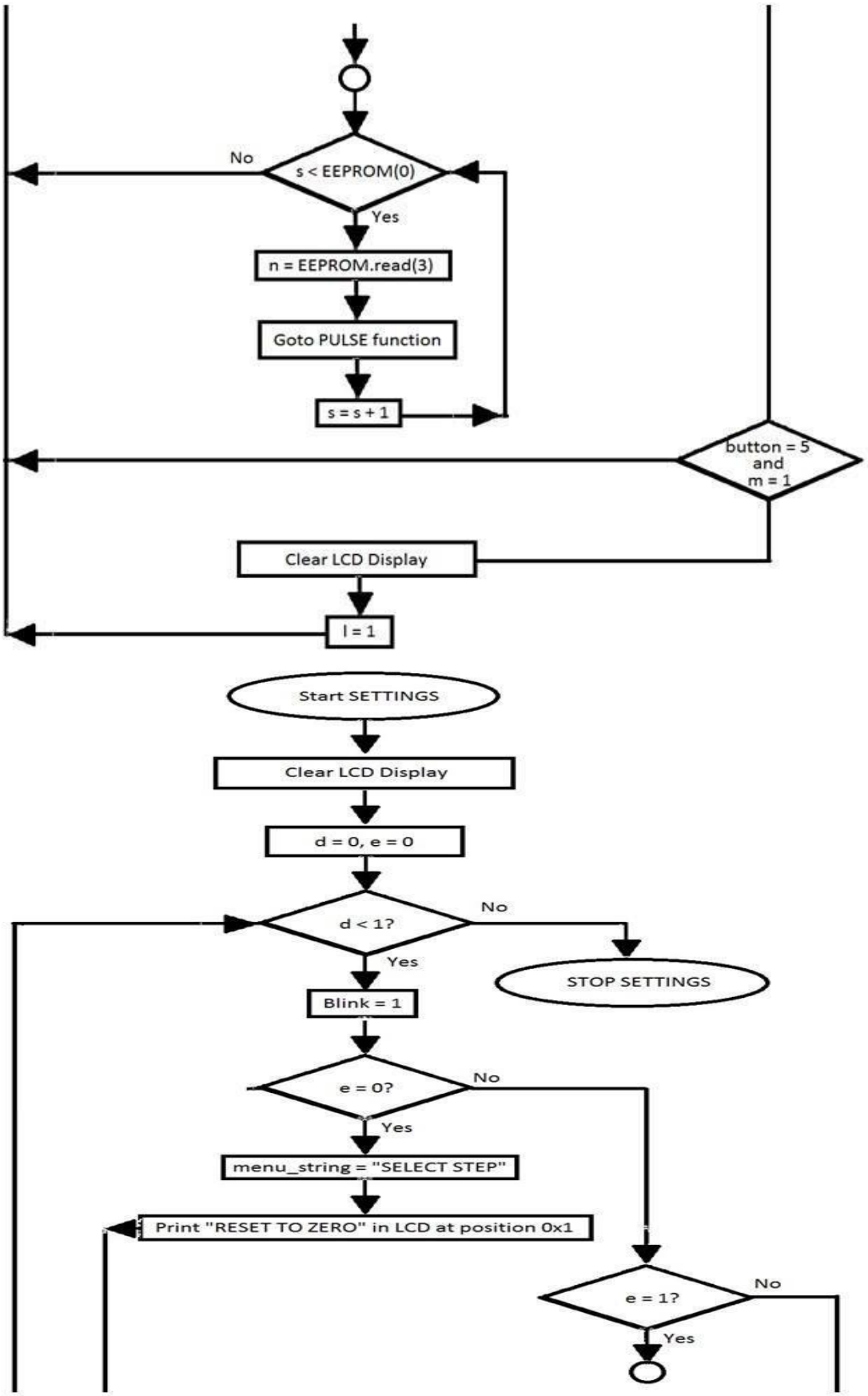


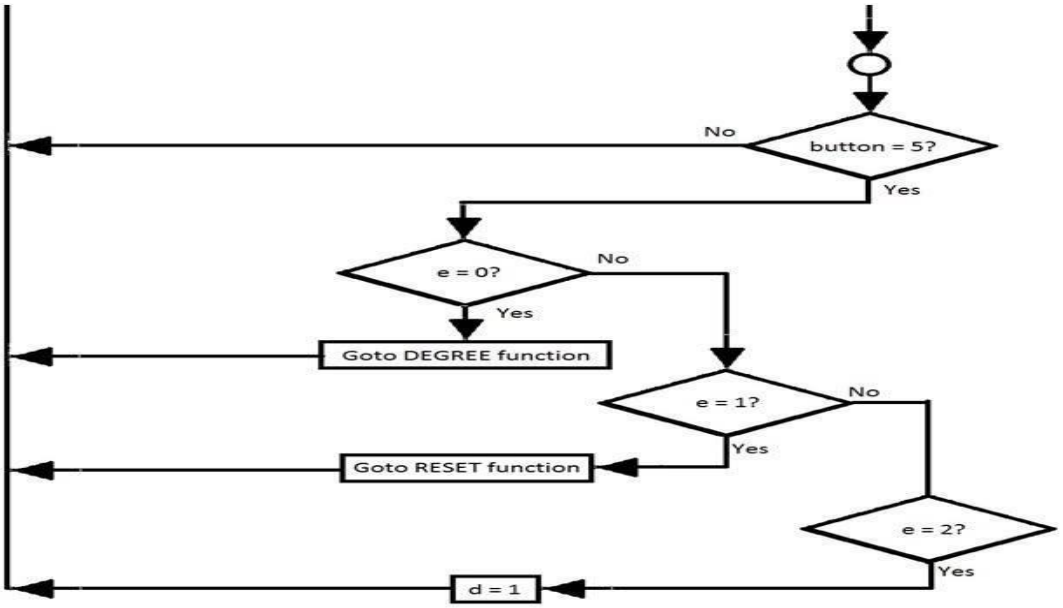

Print "MADE BY Dept of" in LCD at 2x0 position
Print "PHYSICS SUST in LCD at 0x1 position
Wait 2 sec
Start Infinite Loop
b = 0, t = 0
Clear LCD Display
t < 1?
Yes
Blink = 1
b = 0?
No
Yes
menu_string = "START CONTROL"
Print "SETTINGS" in LCD at 0x1 position
b = 1?
Yes
Print "START CONTROL in LCD at position 0x0
menu_string = "SETTINGS"
c = b
Goto BUTTON function

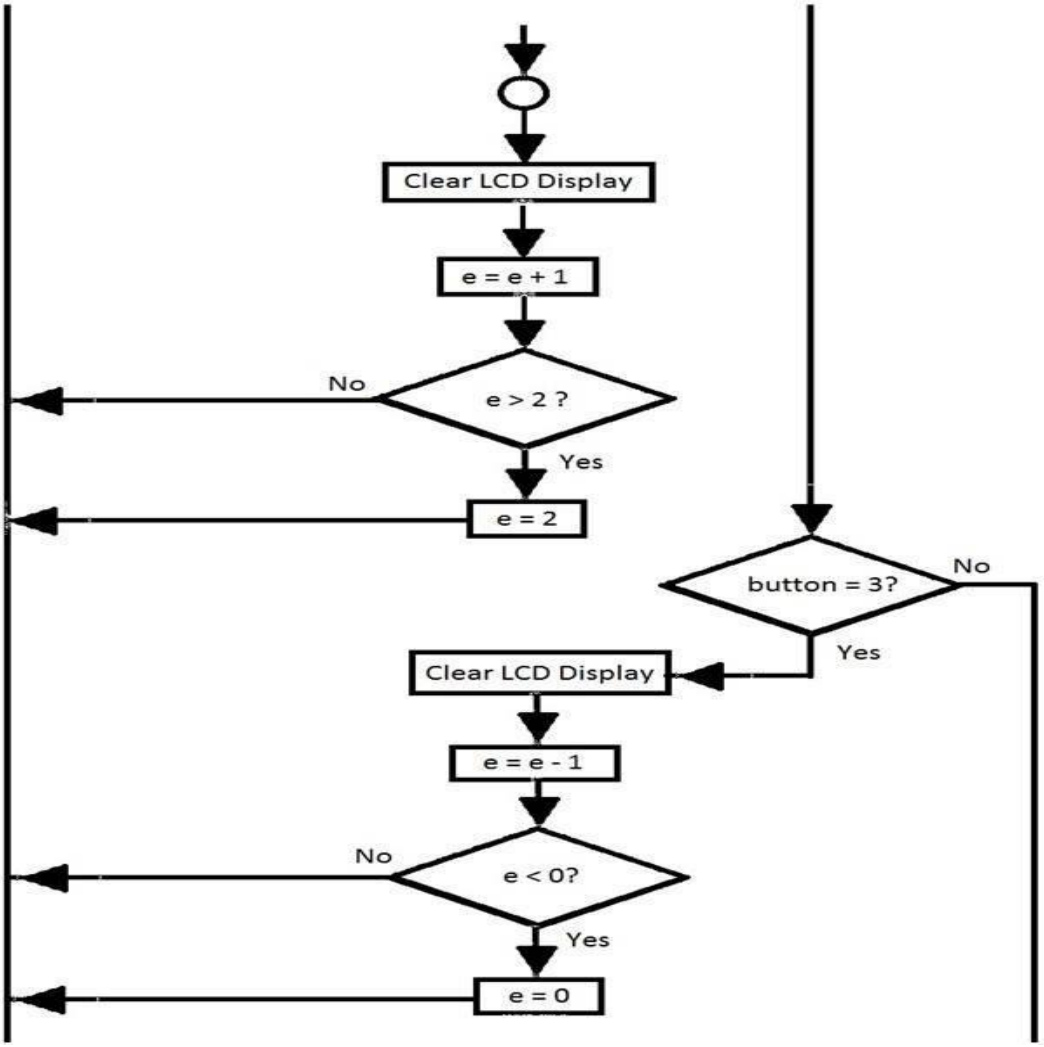

Read button
button = 4?
No
Yes
Clear LCD display
b = b + 1
b > 1?
Yes
b = 1
button = 3?
No
Yes
Clear LCD display
b = b - 1
No
b < 0?
Yes
b = 0
No
button = 5?
Yes
Yes
b = 0?
No
Goto MAIN function
b = 1?
Goto SETTINGS function

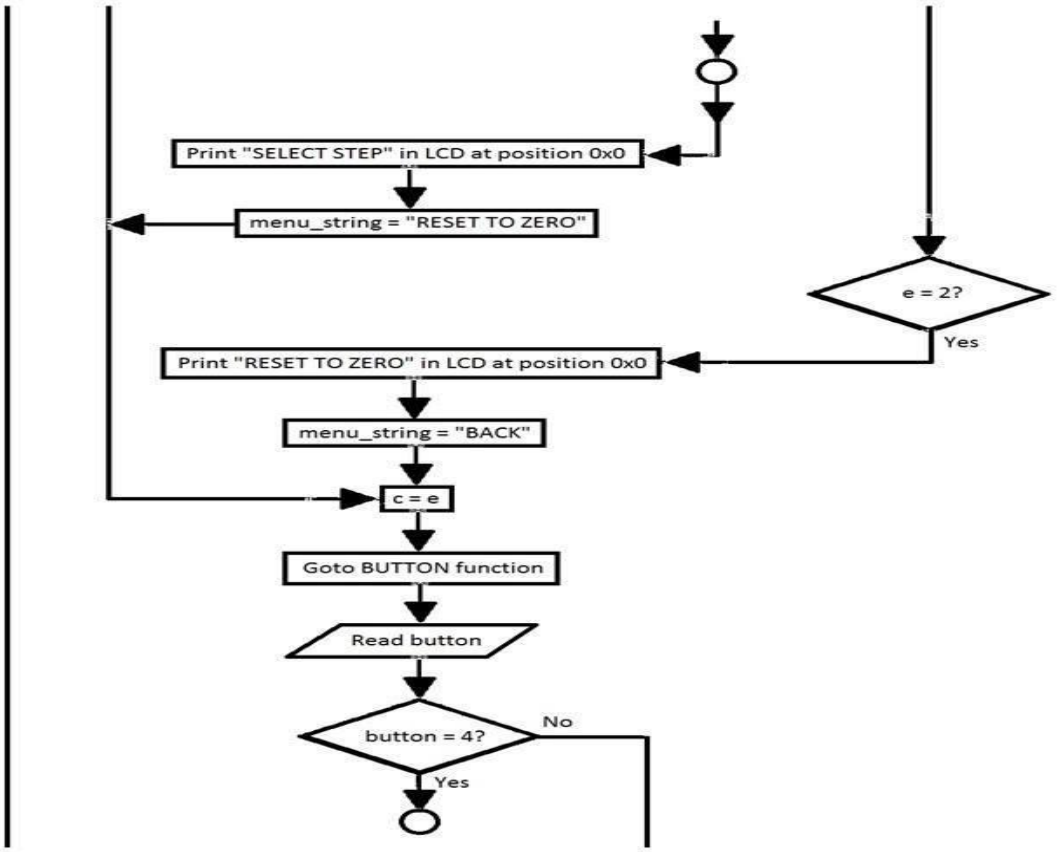

Start MAIN
Clear LCD Display
Read EEPROM(2)
k = EEPROM.read(2)
l = 0, m = 0
l < 1?
No
Yes
Stop MAIN
m = 0?
No
Yes
Clear LCD Display
Blink = 0
Print "DEGREE = " in LCD at 0x0 position
Print the value of k in LCD at 9x0 position
Print "BACK" in LCD at 0x1 position
m = 1?
Yes
Clear LCD Display
Blink = 1
Print "DEGREE = "in LCD at position 0x0
Print the value of k in LCD at position 9x0
menu_string = "BACK"
Wait 100 ms
c = m
Goto BUTTON function

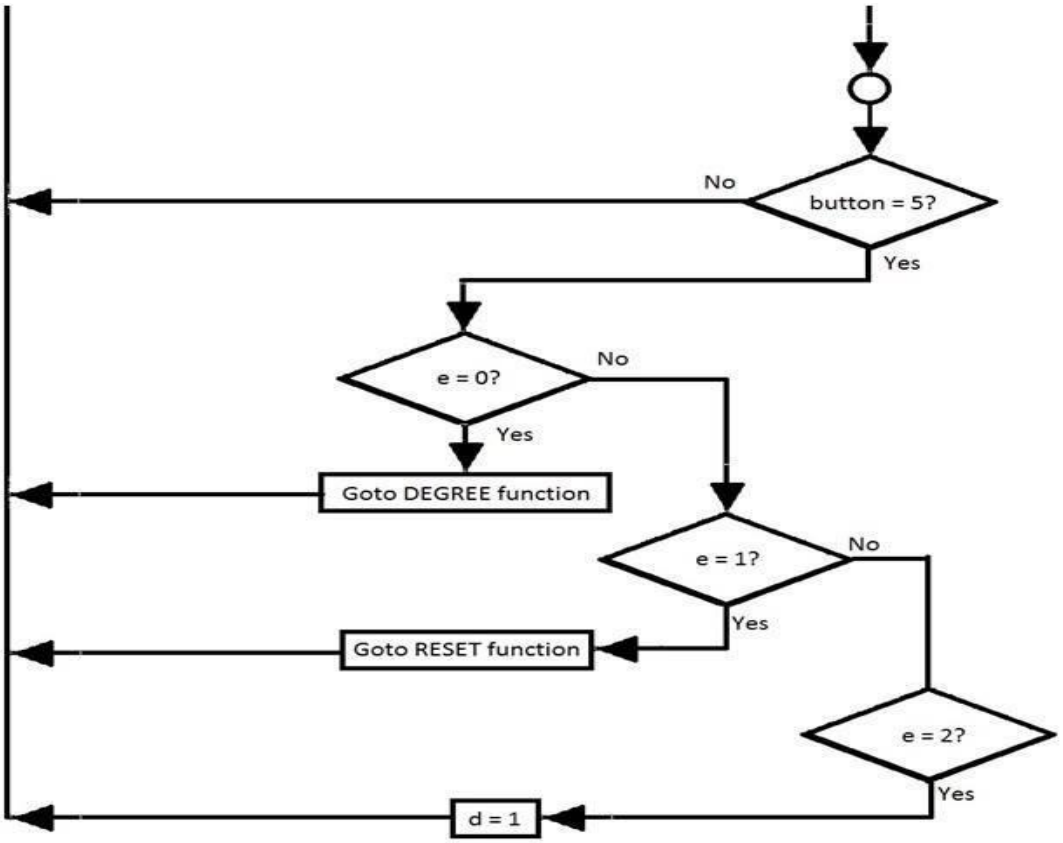

Read button
button = 4?
No
Yes
Clead LCD Display
m = m + 1
No
m > 1?
Yes
m = 1
button = 3?
No
Yes
Clead LCD Display
m = m - 1
No
m < 0
Yes
m = 0
button = 2?
No
Yes
k = k + (2*EEPROM.read(0))
k = 360?
k = 0
EEPROM.write(2, k)
s = 0

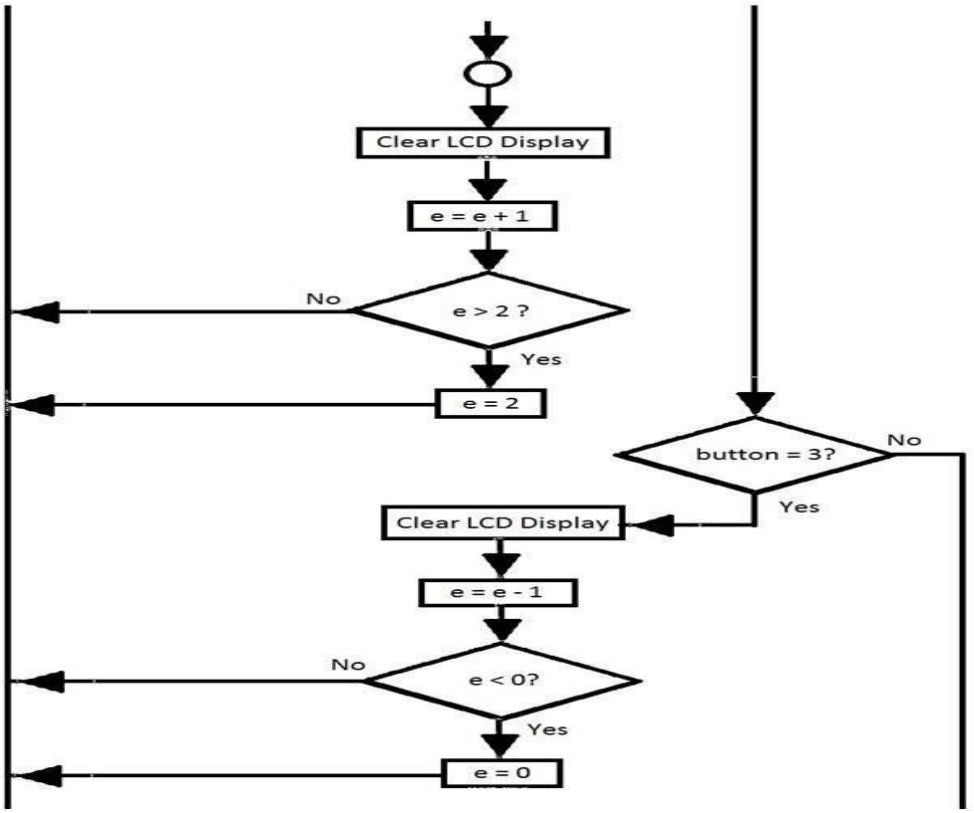

No
s < EEPROM(0)
Yes
n = EEPROM.read(3)
Goto PULSE function
s = s + 1
button = 5
and
m = 1
Clear LCD Display
l = 1
Start SETTINGS
Clear LCD Display
d = 0, e = 0
d < 1?
No
Yes
STOP SETTINGS
Blink = 1
e = 0?
No
Yes
menu_string = "SELECT STEP"
Print "RESET TO ZERO" in LCD at position 0x1
e = 1?
No
Yes

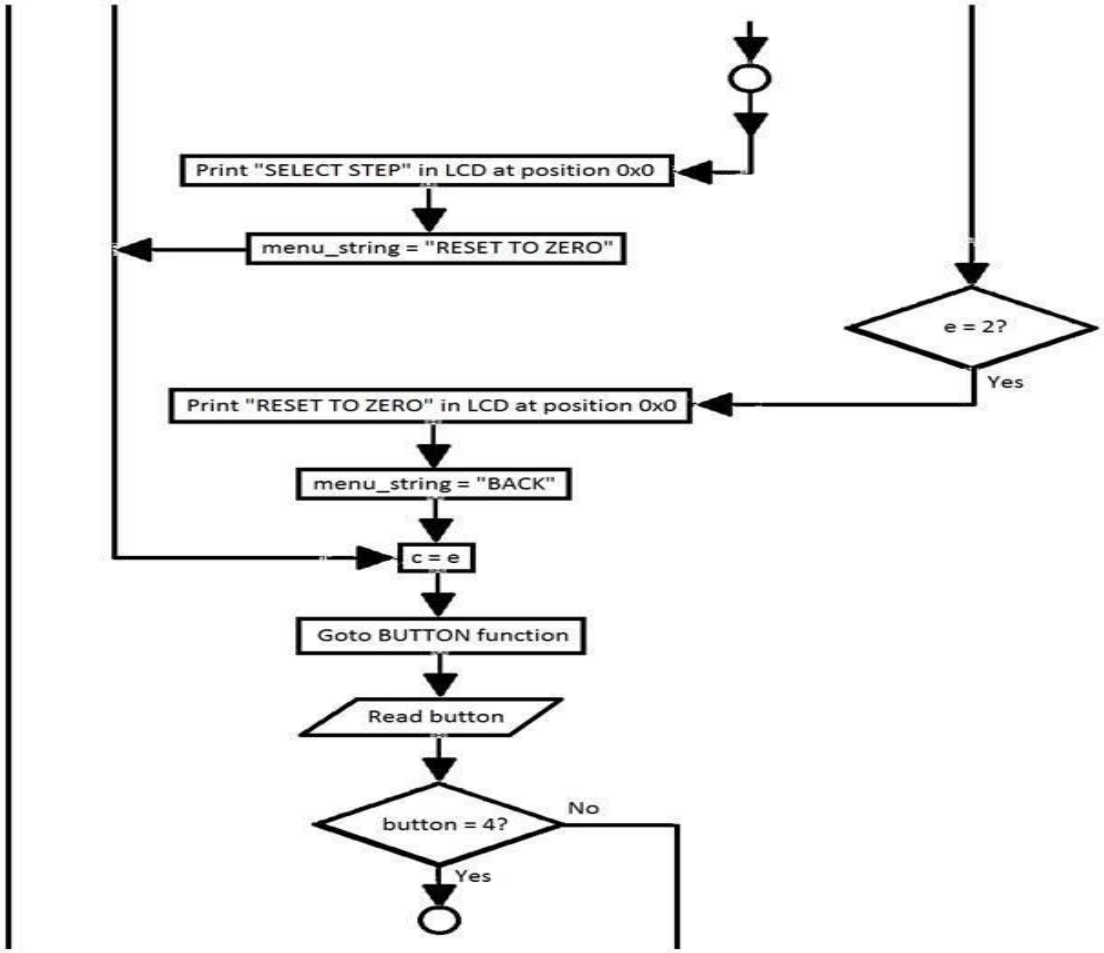

button = 5?
No
Yes
e = 0?
No
Yes
Goto DEGREE function
e = 1?
No
Yes
Goto RESET function
e = 2?
Yes
d = 1

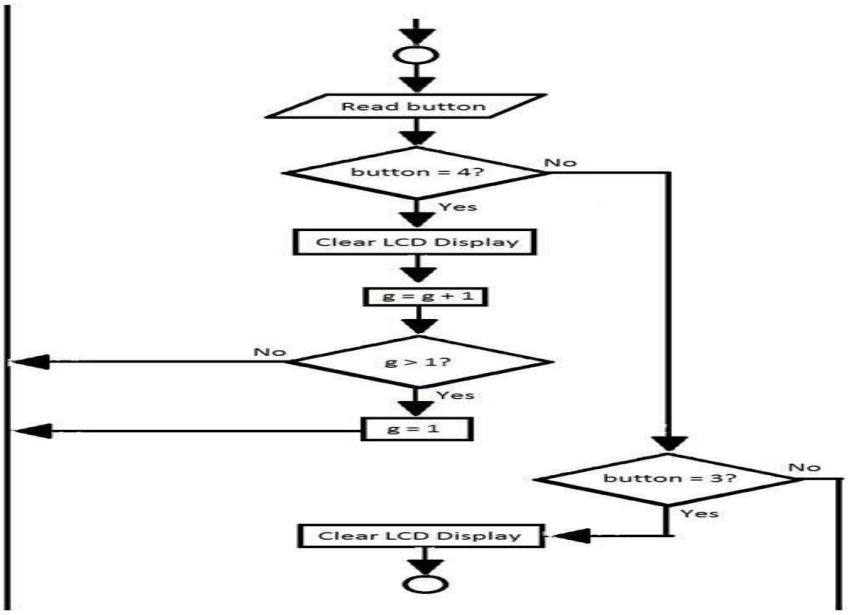

Clear LCD Display
e = e + 1
e > 2 ?
No
Yes
e = 2
button = 3?
No
Yes
Clear LCD Display
e = e - 1
e < 0?
No
Yes
e = 0

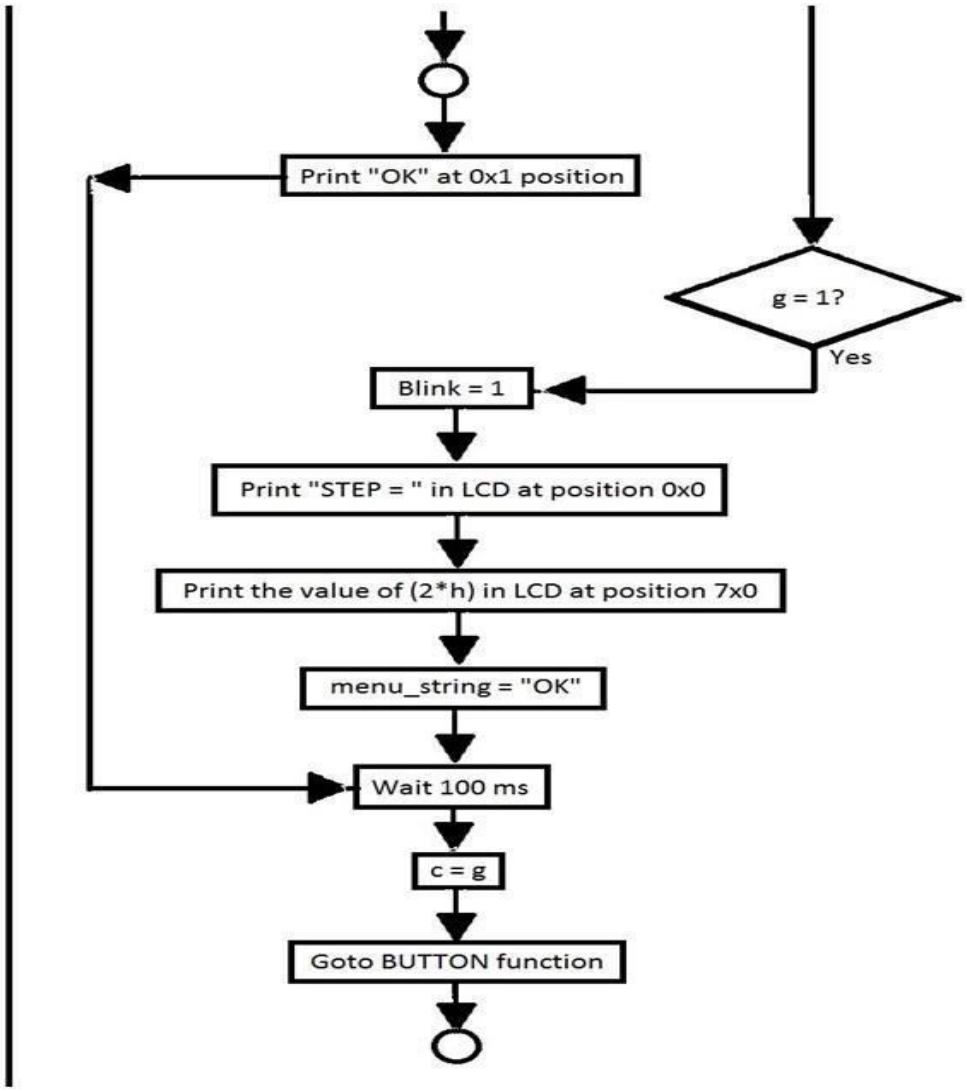

Print "SELECT STEP" in LCD at position 0x0
menu_string = "RESET TO ZERO"
e = 2?
Yes
Print "RESET TO ZERO" in LCD at position 0x0
menu_string = "BACK"
c = e
Goto BUTTON function
Read button
button = 4?
No
Yes

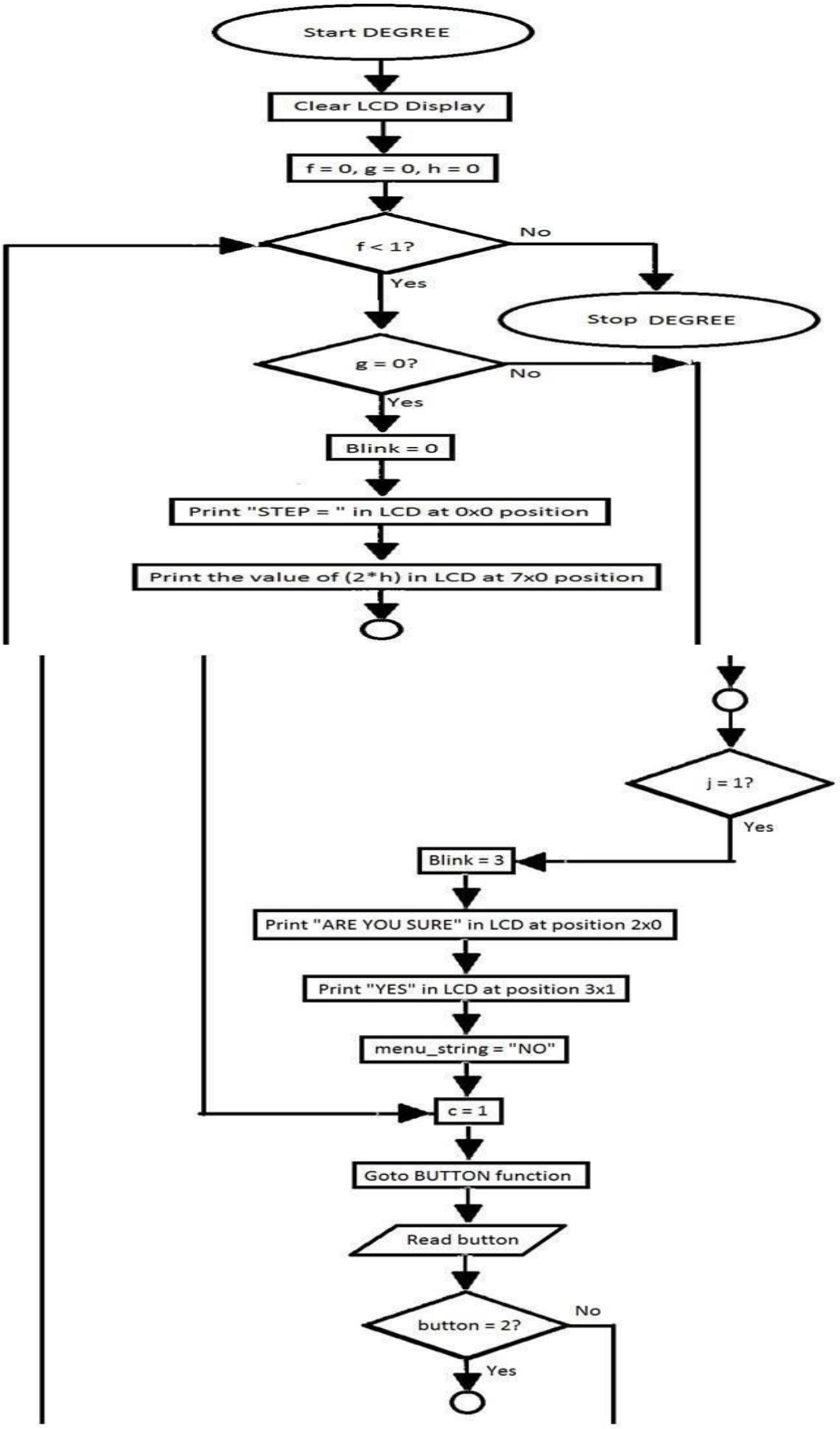

button = 5?
No
Yes
e = 0?
No
Yes
Goto DEGREE function
e = 1?
No
Yes
Goto RESET function
e = 2?
Yes
d = 1

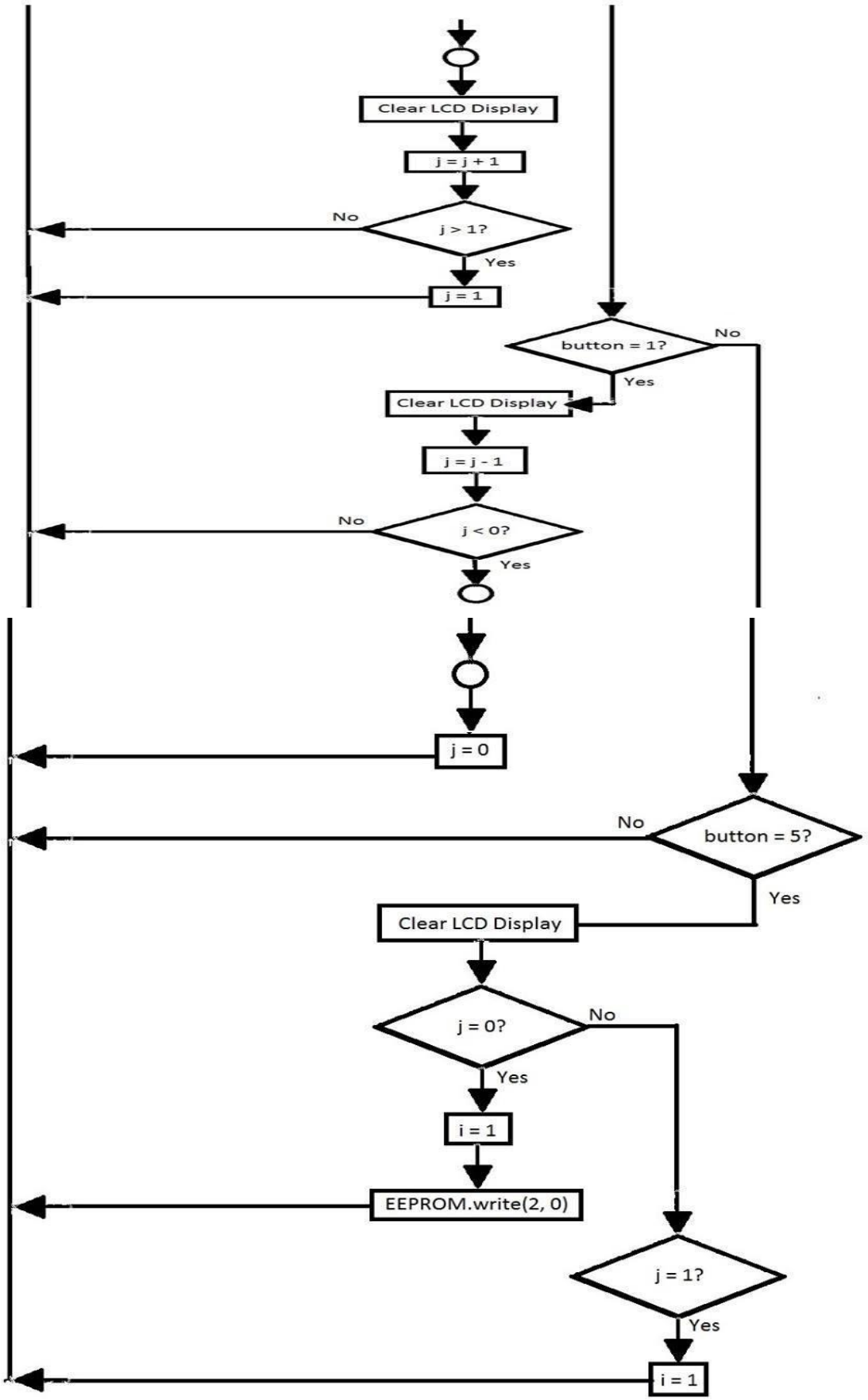

Clear LCD Display
e = e + 1
e > 2 ?
No
Yes
e = 2
button = 3?
No
Yes
Clear LCD Display
e = e - 1
e < 0?
No
Yes
e = 0

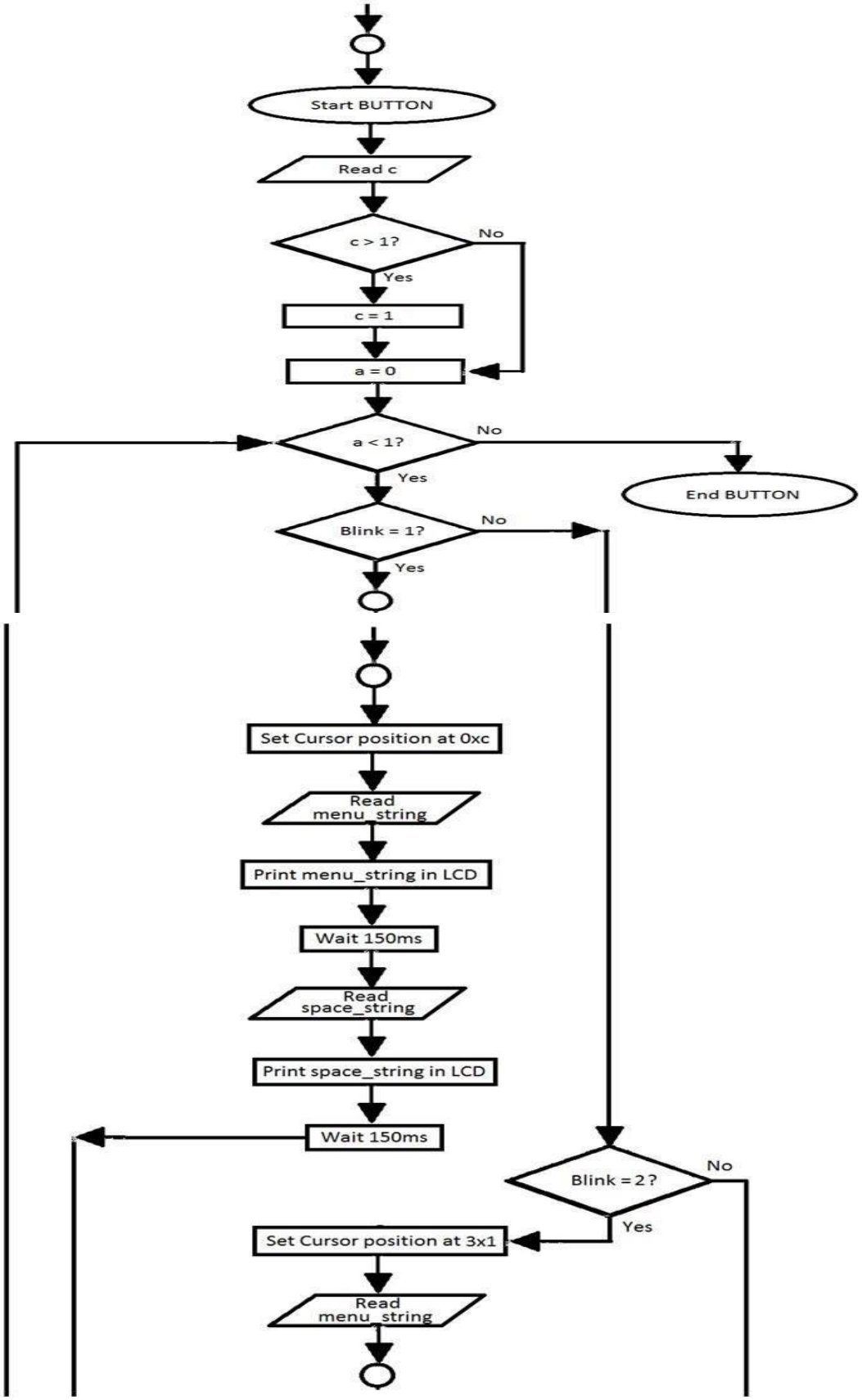

Print "SELECT STEP" in LCD at position 0x0
menu_string = "RESET TO ZERO"
e = 2?
Yes
Print "RESET TO ZERO" in LCD at position 0x0
menu_string = "BACK"
c = e
Goto BUTTON function
Read button
button = 4?
No
Yes

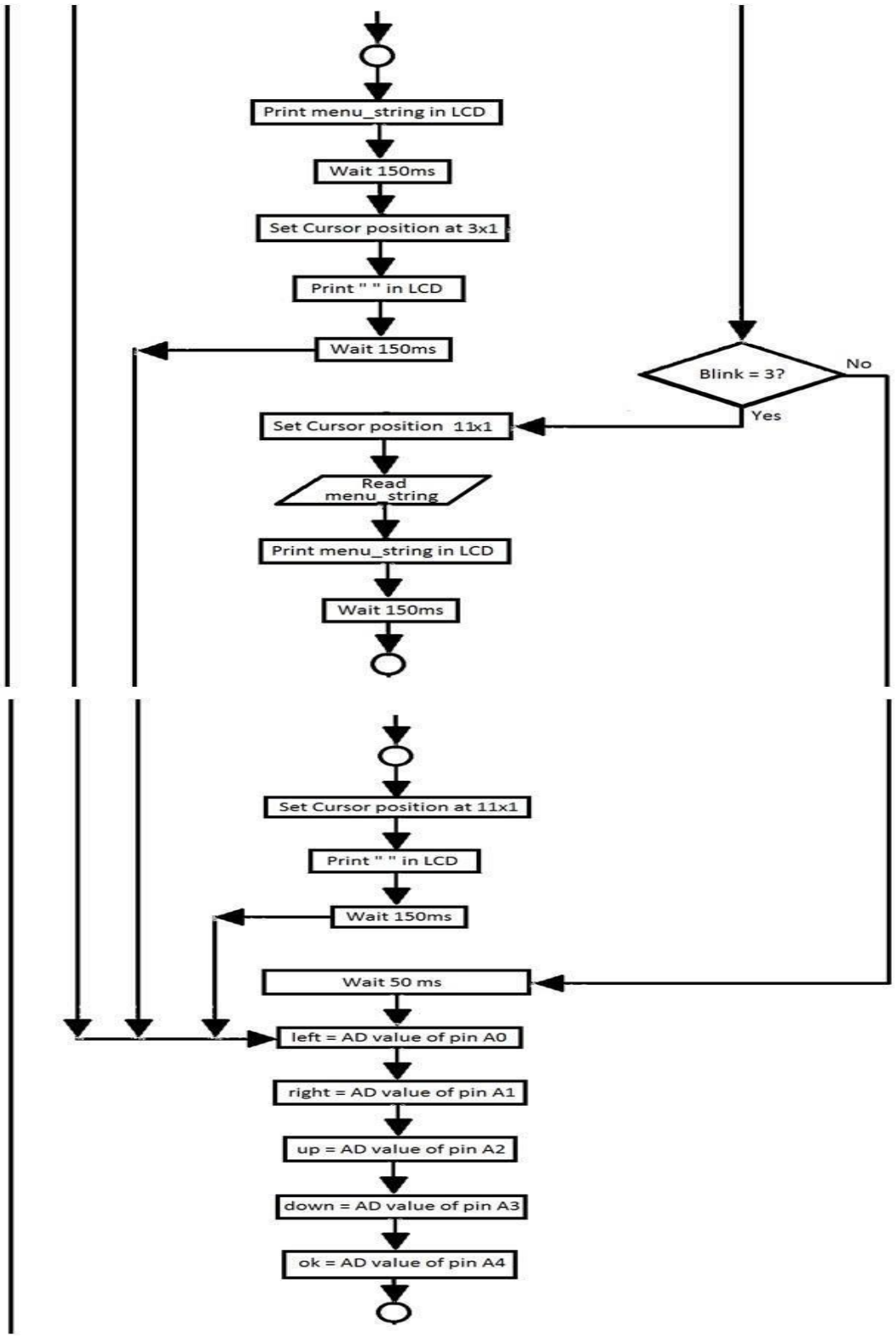

Read button
button = 4?
No
Yes
Clear LCD Display
g = g + 1
No
g > 1?
Yes
g = 1
button = 3?
No
Yes
Clear LCD Display

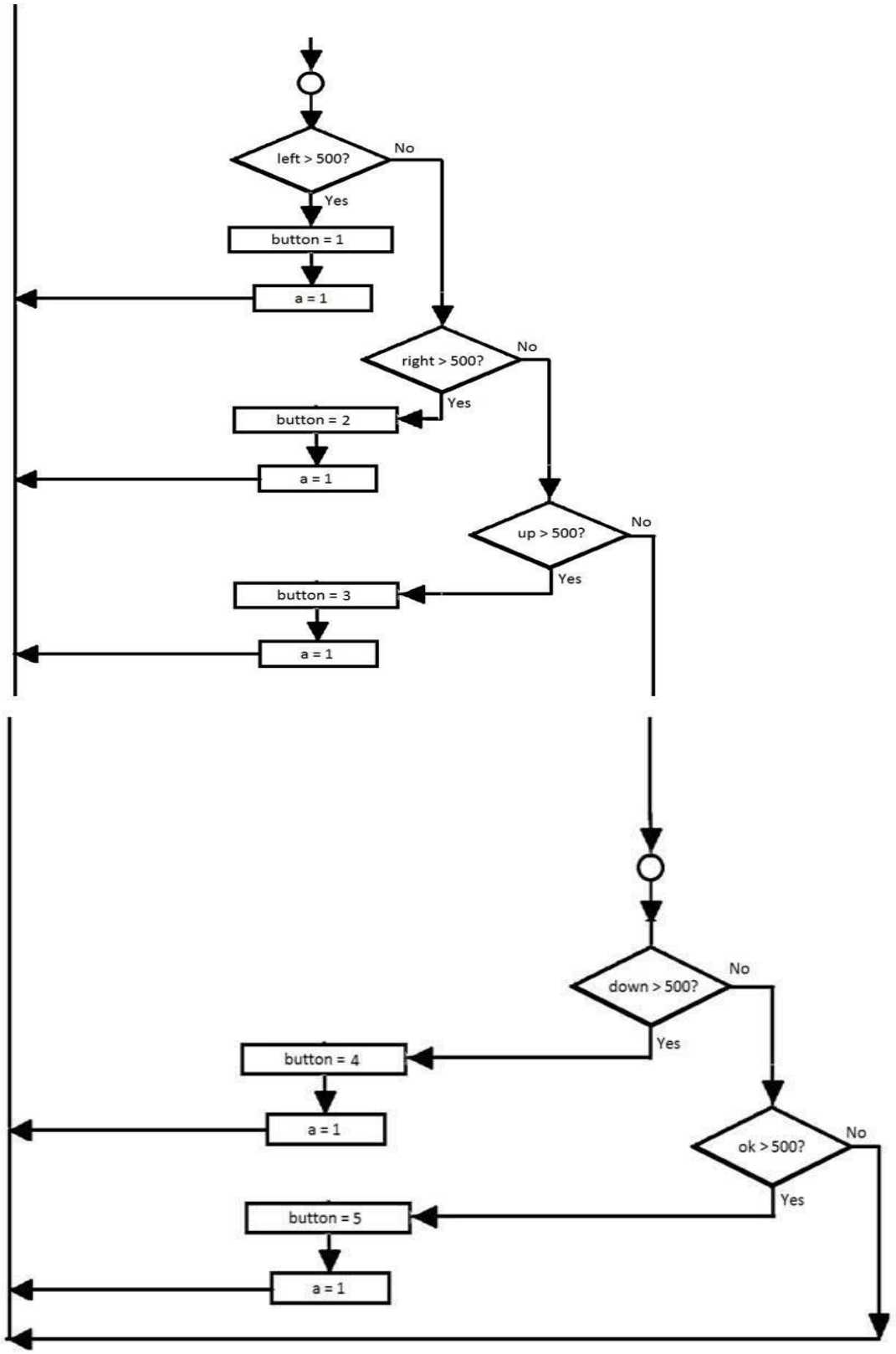

Print "OK" at 0x1 position
g = 1?
Yes
Blink = 1
Print "STEP = " in LCD at position 0x0
Print the value of (2*h) in LCD at position 7x0
menu_string = "OK"
Wait 100 ms
c = g
Goto BUTTON function

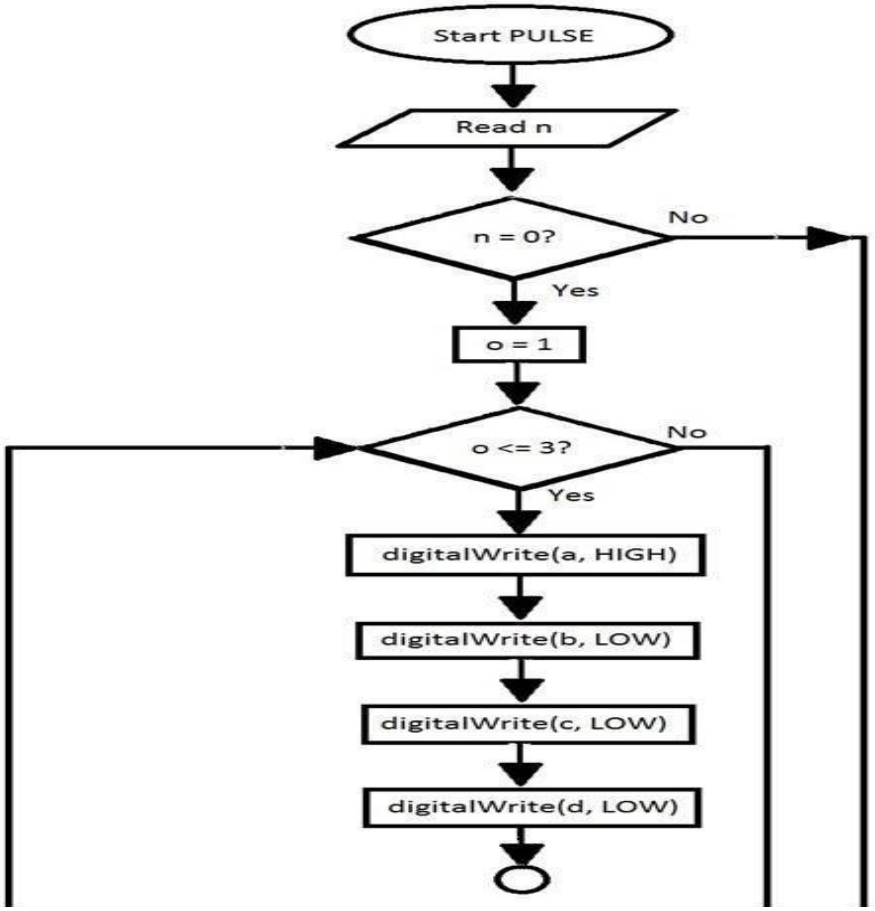

Start DEGREE
Clear LCD Display
f = 0, g = 0, h = 0
f < 1?
No
Yes
Stop DEGREE
g = 0?
No
Yes
Blink = 0
Print "STEP = " in LCD at 0x0 position
Print the value of (2*h) in LCD at 7x0 position
j = 1?
Yes
Blink = 3
Print "ARE YOU SURE" in LCD at position 2x0
Print "YES" in LCD at position 3x1
menu_string = "NO"
c = 1
Goto BUTTON function
Read button
button = 2?
No
Yes

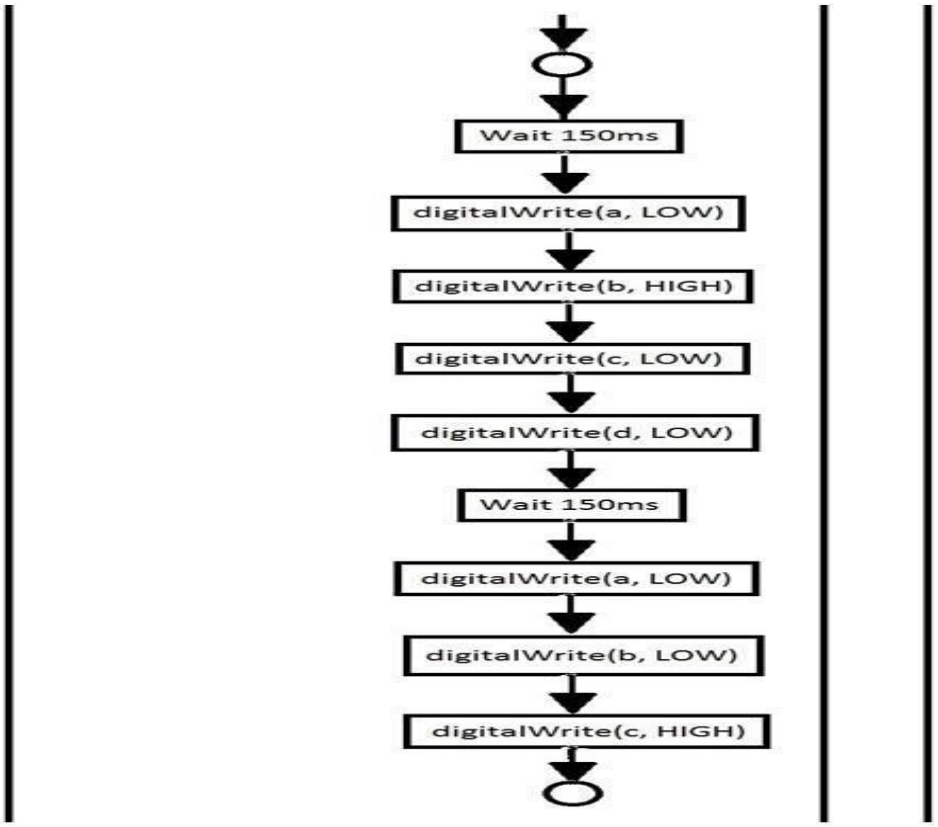

Clear LCD Display
j = j + 1
No
j > 1?
Yes
j = 1
button = 1?
No
Yes
Clear LCD Display
j = j - 1
No
j < 0?
Yes
j = 0
No
button = 5?
Yes
Clear LCD Display
j = 0?
No
Yes
i = 1
EEPROM.write(2, 0)
j = 1?
Yes
i = 1

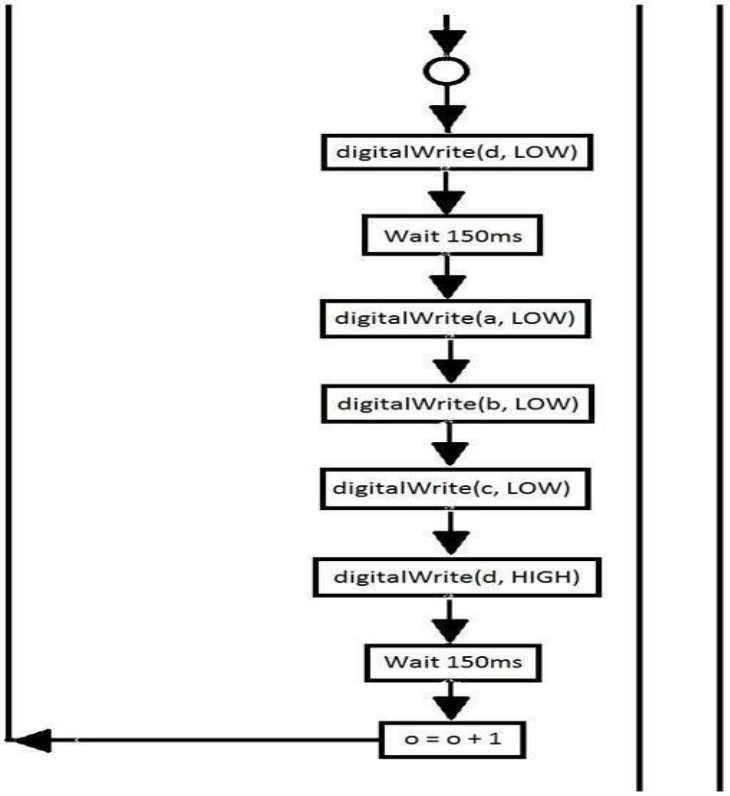

Start BUTTON
Read c
c > 1?
No
Yes
c = 1
a = 0
a < 1?
No
Yes
End BUTTON
Blink = 1?
No
Yes
Set Cursor position at 0xc
Read menu_string
Print menu_string in LCD
Wait 150ms
Read space_string
Print space_string in LCD
Wait 150ms
Blink = 2 ?
No
Yes
Set Cursor position at 3x1
Read menu_string

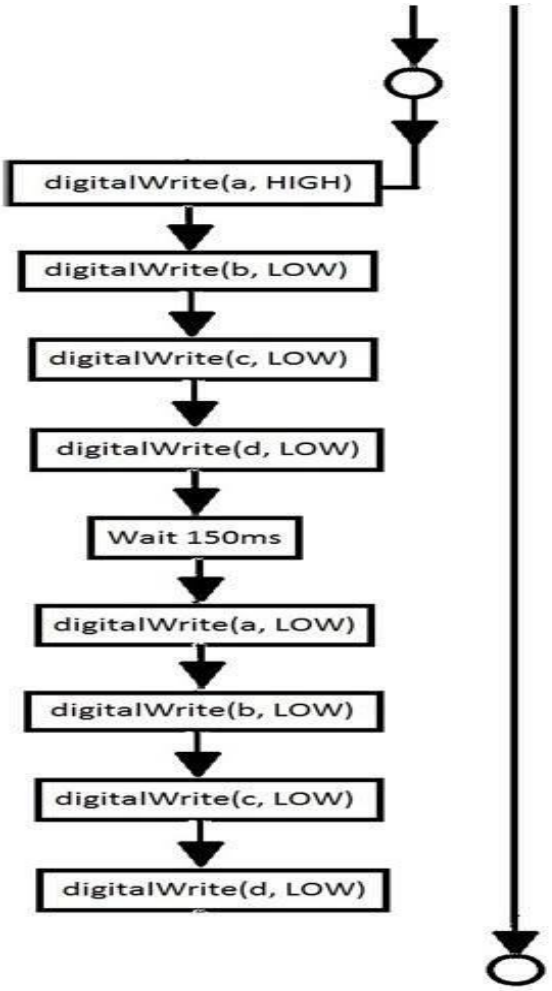

Print menu_string in LCD
Wait 150ms
Set Cursor position at 3x1
Print " " in LCD
Wait 150ms
Blink = 3?
No
Yes
Set Cursor position 11x1
Read menu_string
Print menu_string in LCD
Wait 150ms
Set Cursor position at 11x1
Print " " in LCD
Wait 150ms
Wait 50 ms
left = AD value of pin A0
right = AD value of pin A1
up = AD value of pin A2
down = AD value of pin A3
ok = AD value of pin A4

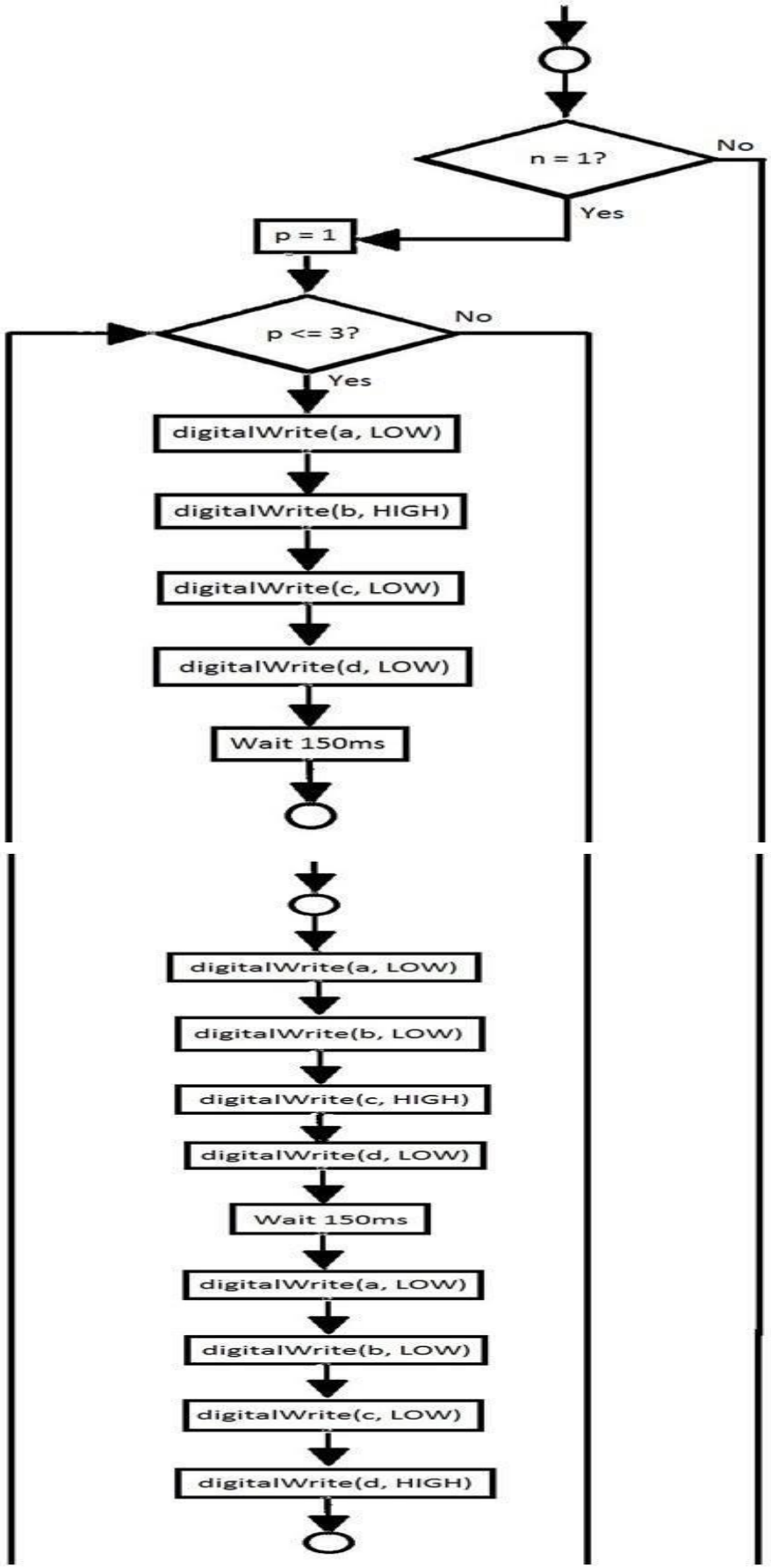

left > 500?
No
Yes
button = 1
a = 1
right > 500?
No
Yes
button = 2
a = 1
up > 500?
No
Yes
button = 3
a = 1
down > 500?
No
Yes
button = 4
a = 1
ok > 500?
No
Yes
button = 5
a = 1

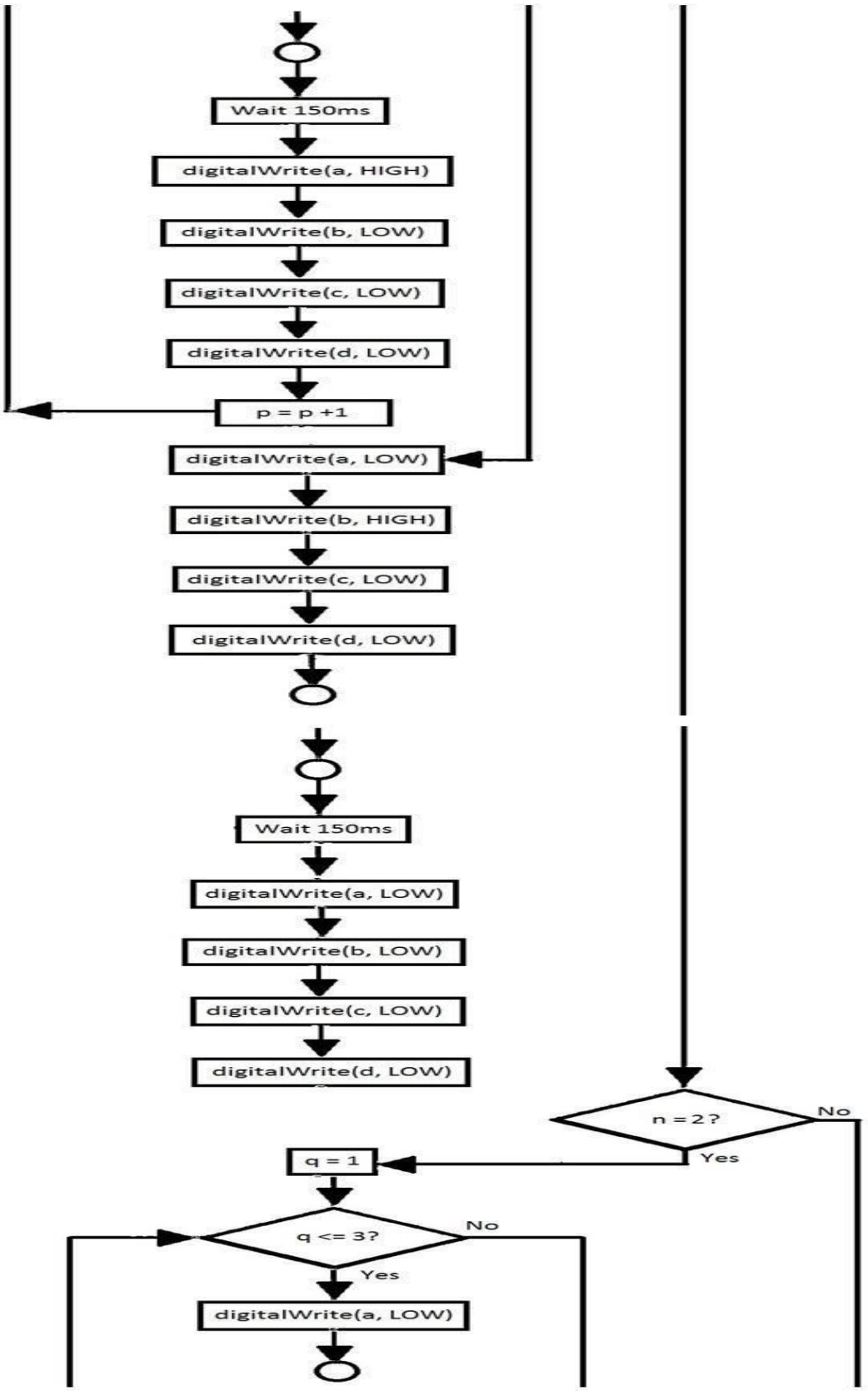

Start PULSE
Read n
n = 0?
No
Yes
o = 1
o <= 3?
No
Yes
digitalWrite(a, HIGH)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, LOW)

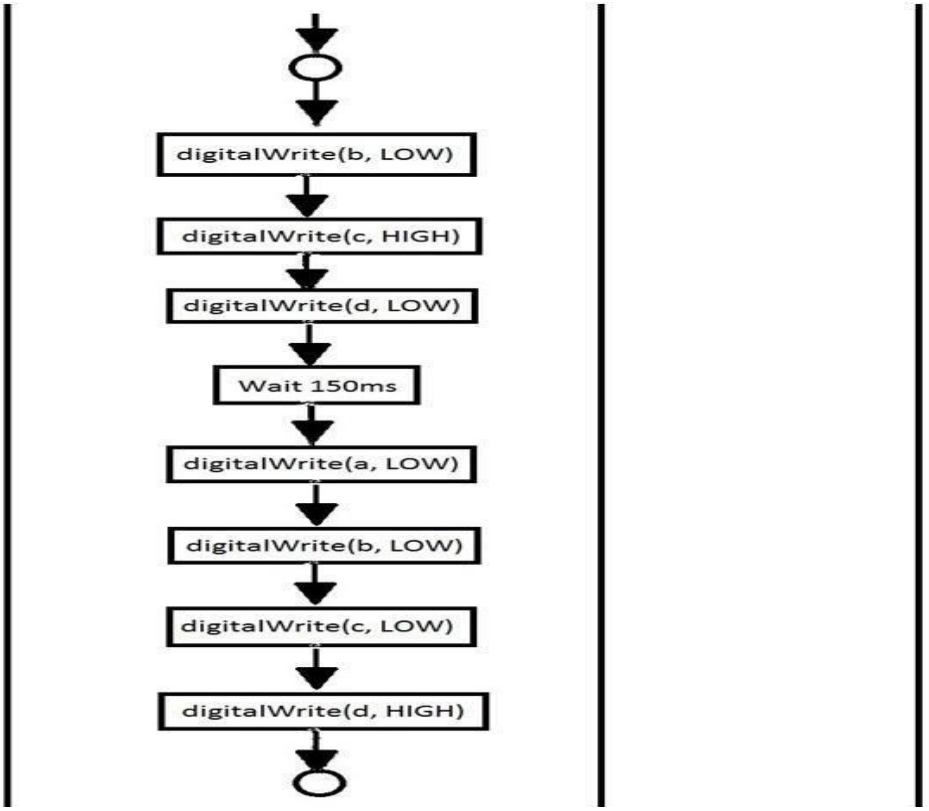

Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, HIGH)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, HIGH)

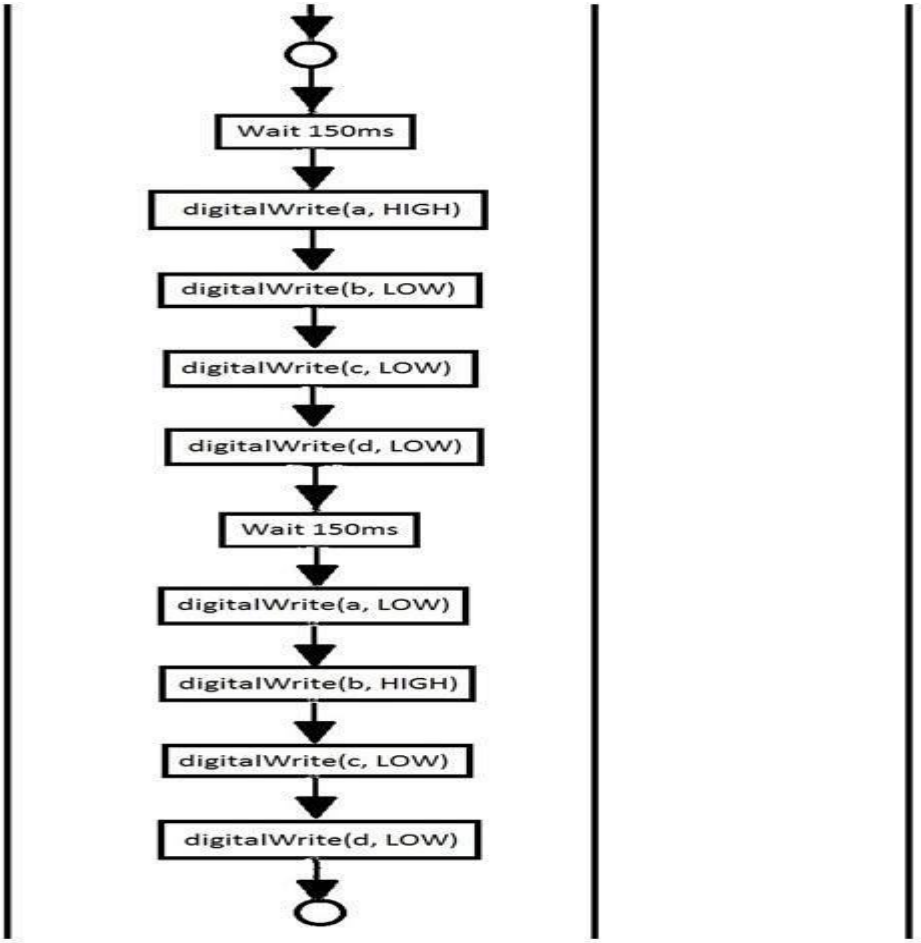

digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, HIGH)
Wait 150ms
o = o + 1

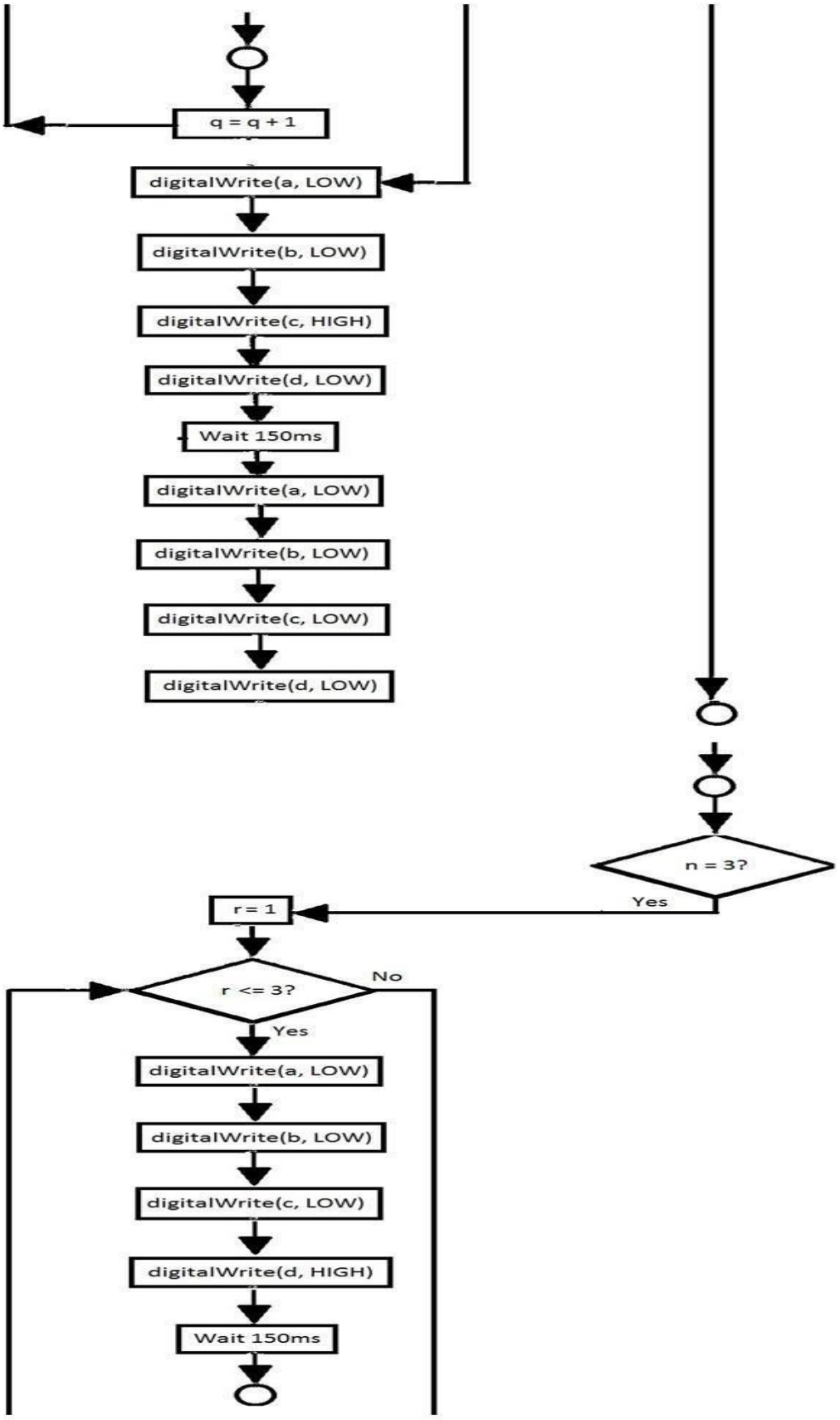

digitalWrite(a, HIGH)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, LOW)

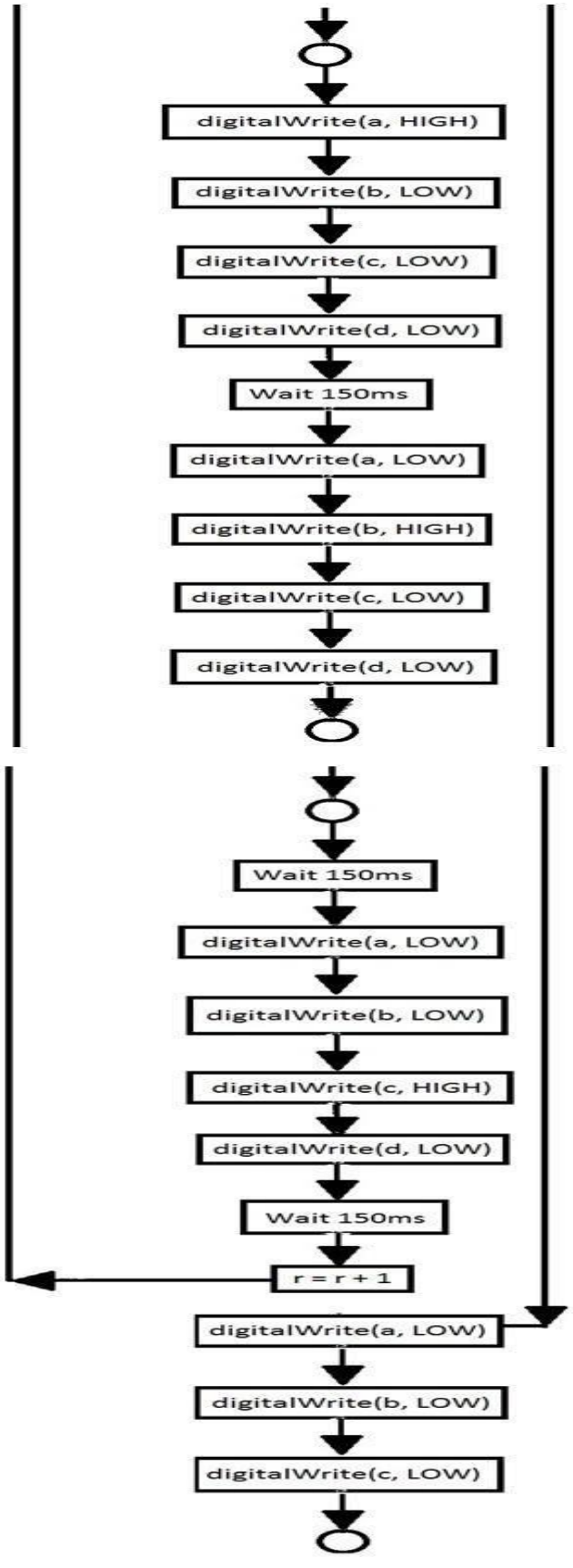

n = 1?
No
Yes
p = 1
p <= 3?
No
Yes
digitalWrite(a, LOW)
digitalWrite(b, HIGH)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, HIGH)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, HIGH)

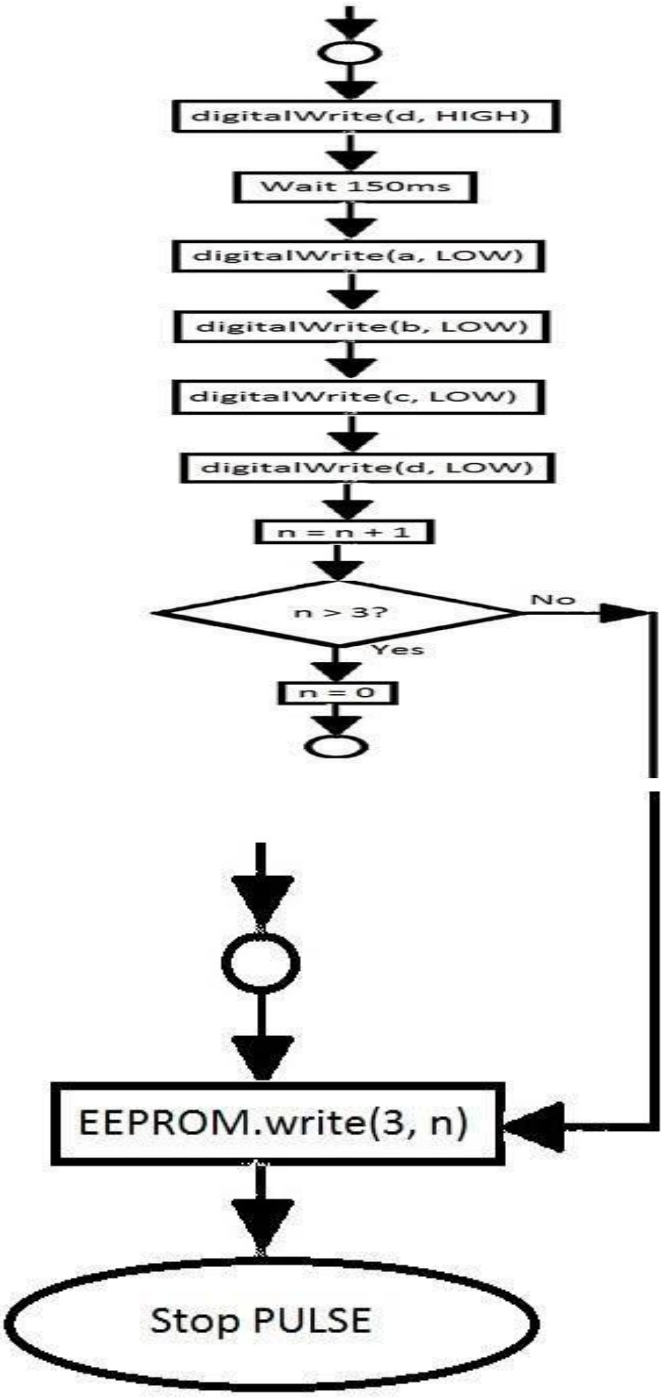

Wait 150ms
digitalWrite(a, HIGH)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
p = p +1
digitalWrite(a, LOW)
digitalWrite(b, HIGH)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
n = 2 ?
No
Yes
q = 1
q <= 3?
No
Yes
digitalWrite(a, LOW)


digitalWrite(b, LOW)
digitalWrite(c, HIGH)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, HIGH)


Wait 150ms
digitalWrite(a, HIGH)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, HIGH)
digitalWrite(c, LOW)
digitalWrite(d, LOW)


q = q + 1
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, HIGH)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
n = 3?
Yes
r = 1
r <= 3?
No
Yes
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, HIGH)
Wait 150ms


digitalWrite(a, HIGH)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, HIGH)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, HIGH)
digitalWrite(d, LOW)
Wait 150ms
r = r + 1
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, LOW)


digitalWrite(d, HIGH)
Wait 150ms
digitalWrite(a, LOW)
digitalWrite(b, LOW)
digitalWrite(c, LOW)
digitalWrite(d, LOW)
n = n + 1
n > 3?
No
Yes
n = 0
EEPROM.write(3, n)
Stop PULSE

2.5 Control Unit Coding
(Please See Appendix)

2.5.1 Some Techniques used in Coding

Stepper motor needs pulse sequence to rotate. The angle of rotation depends up to number of given pulses. Microcontroller generates pulses. For precise stepper motor rotation especially for discontinuous small degree of rotation step sequence should be started from just after previous step sequences. If our mechanical system need four step sequences for two degree of rotation (i.e. 1000, 0100, 0010, 0001). Now if microcontroller provides two step sequences for one degree of rotation then 1^{st} step sequence is 1000 and 2^{nd} step sequence is 0100. Let us consider user now allow rotational stage to hold this position. If we want to rotate next one degree we have to start from 0010 not from 1000. So in coding last step sequences always saved in EEPROM of Microcontroller. After particular angel of rotation LCD shows how much rotation occurs calculating previously saved angel of rotation in EEPROM.

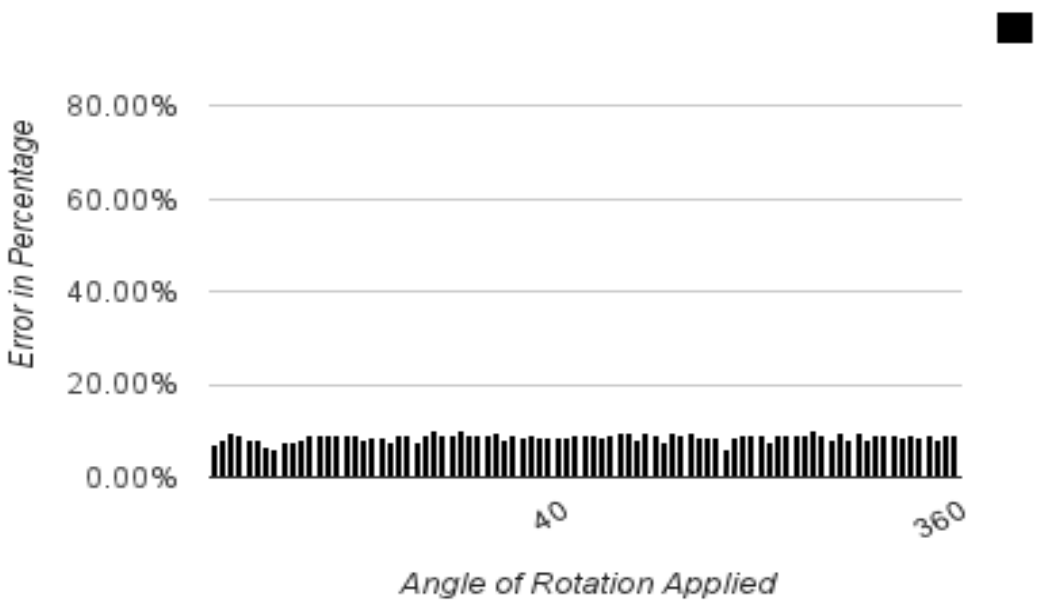
2.6 Technical Experimental

Angle of Rotation Applied	Actual Rotation	Error	Error Percentage
in degree	in degree	in degree	%
360	393.00	33.00	9.17%
	393.00	33.00	9.17%
	390.00	30.00	8.33%
	393.00	33.00	9.17%
	391.00	31.00	8.61%

270	294.00	24.00	8.89%
	293.00	23.00	8.52%
	294.00	24.00	8.89%
	294.00	24.00	8.89%
	294.00	24.00	8.89%
180	195.00	15.00	8.33%
	197.00	17.00	9.44%
	195.00	15.00	8.33%
	197.00	17.00	9.44%
	195.00	15.00	8.33%
120	131.00	11.00	9.17%
	132.00	12.00	10.00%
	131.00	11.00	9.17%
	131.00	11.00	9.17%
	131.00	11.00	9.17%
90	98.00	8.00	8.89%
	97.00	7.00	7.78%
	98.00	8.00	8.89%
	98.00	8.00	8.89%
	98.00	8.00	8.89%
80	87.00	7.00	8.75%
	85.00	5.00	6.25%
	87.00	7.00	8.75%
	87.00	7.00	8.75%
	87.00	7.00	8.75%
70	76.67	6.67	9.52%
	76.33	6.33	9.05%
	76.67	6.67	9.52%
	75.33	5.33	7.62%
	76.33	6.33	9.05%
60	65.67	5.67	9.44%
	65.00	5.00	8.33%
	65.67	5.67	9.44%
	65.67	5.67	9.44%
	65.33	5.33	8.89%
50	54.33	4.33	8.67%
	54.67	4.67	9.33%
	54.67	4.67	9.33%
	54.67	4.67	9.33%
	54.33	4.33	8.67%
40	43.50	3.50	8.75%
	43.50	3.50	8.75%
	43.50	3.50	8.75%
	43.75	3.75	9.38%
	43.50	3.50	8.75%
30	32.67	2.67	8.89%
	32.50	2.50	8.33%
	32.83	2.83	9.44%
	32.67	2.67	8.89%
	32.67	2.67	8.89%
20	21.83	1.83	9.17%

	22.00	2.00	10.00%
	21.83	1.83	9.17%
	21.83	1.83	9.17%
	22.00	2.00	10.00%
10	10.89	0.89	8.89%
	10.78	0.78	7.78%
	10.89	0.89	8.89%
	10.89	0.89	8.89%
	10.78	0.78	7.78%
8	8.70	0.70	8.75%
	8.70	0.70	8.75%
	8.65	0.65	8.13%
	8.75	0.75	9.38%
	8.75	0.75	9.38%
6	6.55	0.55	9.17%
	6.55	0.55	9.17%
	6.55	0.55	9.17%
	6.55	0.55	9.17%
	6.50	0.50	8.33%
4	4.31	0.31	7.69%
	4.31	0.31	7.69%
	4.25	0.25	6.25%
	4.27	0.27	6.82%
	4.33	0.33	8.33%
2	2.16	0.16	8.06%
	2.19	0.19	9.38%
	2.19	0.19	9.68%
	2.16	0.16	8.00%
	2.14	0.14	7.14%

Error W Fig 2.6 Error Graph



2.7 Limitations & Future Works

Current version of device has some limitations. Modified mechanical design made with PVC and belt. It should be replaced by metal and pinion to reduce error. There is no onboard sensor to read actual degree of rotation. Sensors should be installed so that PID algorithm could make this device more accurate and user friendly.

Some steps should be taken in future such as

1) Wireless control feature
2) Artificial Intelligence
3) Graphical LCD to show more information during operation
4) Automatic error reduction system
5) Computer interfacing option
6) Internet based control system
7) Interfacing with Laser Controller Device
8) Smartphone Interfacing

References

[1] Michael Margolis, *Arduino Cookbook,* O'Reilly Media, 2011
[2] Simon Monk, *30 Arduino Projects for the Evil Genious,* Mc Graw Hill, 2010
[3] Dogan Ibrahim, *Advanced PIC microcontroller projects in C,* Newnes, 2008
[4] D. W. Smith, *PIC in Practics, A Project Based Approach*, Newnes*,* 2006
[5] www.arduino.cc, Official Site of Arduino
[6] Dale Wheat, *Arduino Internals,* Apress,
[7] John David Warren, Josh Adams & Harald Molle, *Arduino Robotics,* Apress

Appendix

Laser Control Unit Code:

```
//LASER Control Device (Control Unit)
// Project NLO  Thesis Nabil
#include <VirtualWire.h>
#include <LiquidCrystal.h>
#undef int
#undef abs
#undef double
#undef float
#undef round
LiquidCrystal lcd(7, 6, 9, 3, 1, 0);
int indicator_pin = 13;
int receiver_pin = 2;
int i = 0;
int j = 0;
int k = 0;
char msg[3];

void setup() {
  //pinMode(indicator_pin, OUTPUT);
  pinMode(A1, OUTPUT);
  pinMode(A2, OUTPUT);
  pinMode(5, OUTPUT);
  delay(1000);
  lcd.begin(16,2);
  lcd.print(" LASER  CONTROL ");
  lcd.setCursor(6, 1); lcd.print("UNIT");
  vw_set_rx_pin(receiver_pin);
  vw_setup(2000);
  vw_rx_start();
}

void loop() {
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;
  digitalWrite(indicator_pin, HIGH);
  if(vw_get_message(buf, &buflen)) {
    for(j = 0; j < 3; j++) {
      msg[j] = buf[j];
    }
    if(msg[1] == '1') {
      if(msg[0] == 'L') {
        if(msg[2] == '1') {digitalWrite(A1, HIGH);
```

```
      lcd.clear(); lcd.setCursor(0,0);lcd.print("LSR ON");
      lcd.setCursor(0,1);lcd.print("Power=");
      lcd.setCursor(6,1);lcd.print(k*25);
      }
      if(msg[2] == '0') {digitalWrite(A1, LOW);
      lcd.clear(); lcd.setCursor(0,0);lcd.print("LSR OFF"); }
    }
    if(msg[0] == 'S') {
      if(msg[2] == '1') {
        digitalWrite(A2, HIGH);
        lcd.setCursor(8,0); lcd.print("      ");
        lcd.setCursor(8,0);lcd.print("SH ON");
      }
      if(msg[2] == '0') {
        digitalWrite(A2, LOW);
        lcd.setCursor(8,0); lcd.print("      ");
        lcd.setCursor(8,0); lcd.print("SH OFF");
      }
    }
    if(msg[0] == 'P') {
      k = msg[2] - 48;
      analogWrite(5, k*25);
      lcd.setCursor(0,1); lcd.print("Power=");
      lcd.setCursor(6,1); lcd.print(k*10);
      if(k < 10) {
        lcd.setCursor(8, 1); lcd.print("% ");
      }
      if(k == 10) {
        lcd.setCursor(9, 1); lcd.print("%");
      }
    }
   }
   for(j = 0; j < 3; j++) {
    msg[j] = ' ';
   }
  }
}
```

Remote Unit Code:

```
//Laser Controller , Remote Unit
//LASER OFF BLINKING, ARROW SIGN, LOCK SIGN
#include <EEPROM.h>
#include <LiquidCrystal.h>
#include <VirtualWire.h>
#undef int
#undef abs
#undef double
```

```
#undef float
#undef round
LiquidCrystal lcd(6, 5, 4, 3, 1, 0);
int up = 0;
int down = 0;
int right = 0;
int left = 0;
int ok = 0;

const char *menu_string = "";
const char *space_string = "                ";

int password[4];
int blnk = 0;  //blink off

int button = 0; int i = 0;
int s1 = 0; int p1 = 0;
int s2 = 0; int p2 = 0;
int s3 = 0; int p3 = 0;
int l1 = 0; //for laser on
int l2 = 0;
int l3 = 0;

int string_no = 0;
int x = 0;
int permanent_lock = 0;

int pass_state = EEPROM.read(5);
int default_state = EEPROM.read(6);

const int transmit_pin = 9;
byte space[8] = {0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000, 0b00000,
0b11111};

void setup() {
  lcd.createChar(2, space);
  lcd.begin(16, 2);
  lcd.print("Remote ON");
  delay(1000);
  vw_set_tx_pin(transmit_pin);
  vw_setup(2000);
  while(x < 1) {
    if(pass_state == 0) x = 1;
    if(pass_state == 1) {
      permanent_lock = permanent_lock + 1;
      if(permanent_lock < 4)
      ACCESS_CODE(1);
      if(permanent_lock == 4) {
        permanent_lock = 3;
        lcd.clear();
```

```
      lcd.print(" ACCESS DENIED");
      delay(1000);
     }
    }
   }
}

void loop() {
  blnk = 0;  //blink off
  button = 0;
  lcd.clear();
  delay(500);
  lcd.setCursor(4, 0); lcd.print("PRESS OK");
  lcd.setCursor(4, 1); lcd.print("FOR MENU");
  delay(1000);
  Button();
  if(button == 5) {
   SYSTEM();
  }
}


void SYSTEM() {

  int j = 0;
  int k = 0;
  lcd.clear();
  while(j < 1) {
   blnk = 1;  //blink on
   if(k == 0) {
    menu_string = "LASER CONTROL";
    lcd.setCursor(0, 1); lcd.print("SETTINGS");
   }
   if(k == 1) {
    menu_string = "SETTINGS";
    lcd.setCursor(0, 1); lcd.print("BACK");
   }
   if(k == 2) {
    menu_string = "BACK";
   }
   Button();
   if(button == 1) {  //down button
    lcd.clear();
    k = k + 1;
    if(k > 2) {
     k = 2;
    }
   }
   if(button == 2) { // up button
    lcd.clear();
```

```
      k = k - 1;
      if(k < 0) {
        k = 0;
      }
    }
    if(button == 5) {
      if(k == 0)
        LASER_CONTROL();
      if(k == 1)
        SETTINGS();
      if(k == 2)
        j = 1;
    }
  }  //while end
}  //SYSTEM() end

int Button() {
  i = 0;
  //delay(100);           //Could Change
  while(i < 1) {
    if(blnk == 1) {
      lcd.setCursor(0, 0); lcd.print(menu_string);
      delay(150);
      lcd.setCursor(0, 0); lcd.print(space_string);
      delay(150);
    }
    else if(blnk == 2) {
      lcd.setCursor(3, 1); lcd.print(menu_string);
      delay(150);
      lcd.setCursor(3, 1); lcd.print("    ");
      delay(150);
    }
    else if(blnk == 3) {
      lcd.setCursor(11, 1); lcd.print(menu_string);
      delay(150);
      lcd.setCursor(11, 1); lcd.print("  ");
      delay(150);
    }
    else if(blnk == 4) {
      lcd.setCursor(0, 1); lcd.print(menu_string);
      delay(150);
      lcd.setCursor(0, 1); lcd.print("         ");
      delay(150);
    }
    else if(blnk == 5) {
      lcd.setCursor(12, 1); lcd.print(menu_string);
      delay(150);
      lcd.setCursor(12, 1); lcd.print("    ");
      delay(150);
    }
```

```
    else delay(50);
    left = analogRead(A1);
    ok = analogRead(A5);
    up = analogRead(A4);
    down = analogRead(A2);
    right = analogRead(A3);

    if(up > 500) {
      button = 1;
      up = 0;
      i = 1;
    }
    if(down > 500) {
      button = 2;
      down = 0;
      i = 1;
    }
    if(left > 500) {
      button = 3;
      left = 0;
      i = 1;
    }
    if(right > 500) {
      button = 4;
      right = 0;
      i = 1;
    }
    if(ok > 500) {
      button = 5;
      ok = 0;
      i = 1;
    }
  }
  return (button);
}

void LASER_CONTROL() {
  lcd.clear();
  int m;
  if(default_state == 1)
    m = EEPROM.read(0);
  if(default_state == 0)
    m = 0;
  int l = 0;
  while(l < 1) {
    blnk = 1; //blink on
    if(m == 0 && default_state == 0) {
      menu_string = "LASER ONE";
      lcd.setCursor(0, 1); lcd.print("LASER TWO");
    }
```

```
    if(m == 1 && default_state == 0) {
      menu_string = "LASER TWO";
      lcd.setCursor(0, 1); lcd.print("LASER THREE");
    }
    if(m == 2 && default_state == 0) {
      menu_string = "LASER THREE";
      lcd.setCursor(0, 1); lcd.print("BACK");
    }
    if(m == 3 && default_state == 0) {
      menu_string = "BACK";
    }
    if(default_state == 0)
      Button();
    if(button == 1) {  //down button
      lcd.clear();
      m = m + 1;
      if(m > 3) {
        m = 3;
      }
    }
    if(button == 2) { // up button
      lcd.clear();
      m = m - 1;
      if(m < 0) {
        m = 0;
      }
    }
    if(button == 5 || default_state == 1) {
      if(m == 0)
        LASER_ONE();
      if(m == 1)
        LASER_TWO();
      if(m == 2)
        LASER_THREE();
      if(m == 3 || default_state == 1)
        l = 1;
    }
  }
}

void LASER_ONE() {
  if(l1 == 0) {
    string_no = 12;
    RF_Send(string_no);
    l1 = 1;
  }
  int a = 0; int b = 0;
  lcd.clear();
  while(a < 1) {
    //blnk = 0;
```

```
if(b == 0) {
  blnk = 0;
  lcd.setCursor(0, 0); lcd.print("[SH-");
  if(s1 == 0) {
    lcd.setCursor(4, 0); lcd.print("OFF]");
  }
  if(s1 == 1) {
    lcd.setCursor(5, 0); lcd.print("ON]");
  }
  lcd.setCursor(9, 0); lcd.print("POW-");
  lcd.setCursor(13, 0); lcd.print(p1);
  lcd.setCursor(0, 1); lcd.print("LAS_1_OFF |");
  lcd.setCursor(12, 1); lcd.print("LOCK");
  delay(400);
}
if(b == 1) {
  blnk = 4;
  lcd.setCursor(0, 0); lcd.print("SH-");
  if(s1 == 0) {
    lcd.setCursor(3, 0); lcd.print("OFF |");
  }
  if(s1 == 1) {
    lcd.setCursor(4, 0); lcd.print("ON |");
  }
  lcd.setCursor(9, 0); lcd.print("POW-");
  lcd.setCursor(13, 0); lcd.print(p1);
  menu_string = "LAS_1_OFF";
  lcd.setCursor(10, 1); lcd.print("| LOCK");
}
if(b == 2) {
  blnk = 5;
  lcd.setCursor(0, 0); lcd.print("SH-");
  if(s1 == 0) {
    lcd.setCursor(3, 0); lcd.print("OFF |");
  }
  if(s1 == 1) {
    lcd.setCursor(4, 0); lcd.print("ON |");
  }
  lcd.setCursor(9, 0); lcd.print("POW-");
  lcd.setCursor(13, 0); lcd.print(p1);
  lcd.setCursor(0, 1); lcd.print("LAS_1_OFF |");
  menu_string = "LOCK";
}
if(b == 3) {
  blnk = 4;
  lcd.setCursor(0, 0); lcd.print("LAS_1 OFF |");
  lcd.setCursor(12, 0); lcd.print("LOCK");
  menu_string = "BACK";
}
delay(100);
```

```
Button();
if(button == 1) {
  lcd.clear();
  b = b + 1;
  if(b > 3) b = 3;
}
if(button == 2) {
  lcd.clear();
  b = b - 1;
  if(b < 0) b = 0;
}
if(button == 4) {
  lcd.clear();
  //send RF data for power change
  p1 = p1 + 10;
  if(p1 > 100) p1 = 100;
  string_no = 40 + (p1/10);
  RF_Send(string_no);
}
if(button == 3) {
  lcd.clear();
  //send RF data for power change
  p1 = p1 - 10;
  if(p1 < 0) p1 = 0;
  string_no = 40 + (p1/10);
  RF_Send(string_no);
}
if(button == 5) {
  lcd.clear();
  if(b == 0) {
    //send RF code for sutter toggle
    s1 = ! s1;
    if(s1 == 0)
      string_no = 10;
    if(s1 == 1)
      string_no = 11;
    RF_Send(string_no);
  }
  if(b == 1) {
    int k1 = sure();
    //send RF code for LASER OFF
    if(k1 == 0) {
      if(p1 == 0) {
        string_no = 13;
        RF_Send(string_no);
        l1 = 0;
        a = 1;
        lcd.print("   LASER  OFF");
        delay(800);
      }
```

```
        else {
          lcd.print(" POWER MUST BE");
          lcd.setCursor(4, 1); lcd.print("!!ZERO!!");
          delay(1000);
          lcd.clear();
        }
       }
      }
      if(b == 2) {
       //for lock
       blnk = 0;
       delay(500);
       lcd.clear(); lcd.setCursor(0, 0); lcd.print("LASER 1  RUNNING");
       lcd.setCursor(3, 1); lcd.print("!!LOCKED!!");
       x = 0;
       while(x < 1) {
        Button();
        if(button == 5) {
         ACCESS_CODE(1);
         if(x == 0) {
          lcd.clear(); lcd.setCursor(0, 0); lcd.print("LASER 1 RUNNING");
          lcd.setCursor(3, 1); lcd.print("!!LOCKED!!");
         }
        }
       }
       string_no = 14;
       RF_Send(string_no);
       lcd.clear();
      }
      if(b == 3) {
       a = 1; //break while loop
      }
     }
   }//while end
}  //function end

void LASER_TWO() {
  if(l2 == 0){
    string_no = 22;
    RF_Send(string_no);
    l2 = 1;
  }
  int c = 0; int d = 0;
  lcd.clear();
  while(c < 1) {
    //blnk = 0;
    if(d == 0) {
      blnk = 0;
      lcd.setCursor(0, 0); lcd.print("[SH-");
      if(s2 == 0) {
```

```
    lcd.setCursor(4, 0); lcd.print("OFF]");
   }
   if(s2 == 1) {
    lcd.setCursor(5, 0); lcd.print("ON]");
   }
   lcd.setCursor(9, 0); lcd.print("POW-");
   lcd.setCursor(13, 0); lcd.print(p2);
   lcd.setCursor(0, 1); lcd.print("LAS_2_OFF |");
   lcd.setCursor(12, 1); lcd.print("LOCK");
   delay(400);
 }
 if(d == 1) {
   blnk = 4;
   lcd.setCursor(0, 0); lcd.print("SH-");
   if(s2 == 0) {
    lcd.setCursor(3, 0); lcd.print("OFF |");
   }
   if(s2 == 1) {
    lcd.setCursor(4, 0); lcd.print("ON |");
   }
   lcd.setCursor(9, 0); lcd.print("POW-");
   lcd.setCursor(13, 0); lcd.print(p2);
   menu_string = "LAS_2_OFF";
   lcd.setCursor(10, 1); lcd.print("| LOCK");
 }
 if(d == 2) {
   blnk = 5;
   lcd.setCursor(0, 0); lcd.print("SH-");
   if(s2 == 0) {
    lcd.setCursor(3, 0); lcd.print("OFF |");
   }
   if(s2 == 1) {
    lcd.setCursor(4, 0); lcd.print("ON |");
   }
   lcd.setCursor(9, 0); lcd.print("POW-");
   lcd.setCursor(13, 0); lcd.print(p2);
   lcd.setCursor(0, 1); lcd.print("LAS_2 OFF |");
   menu_string = "LOCK";
 }
 if(d == 3) {
   blnk = 4;
   lcd.setCursor(0, 0); lcd.print("LAS_2 OFF |");
   lcd.setCursor(12, 0); lcd.print("LOCK");
   menu_string = "BACK";
 }
 delay(100);
 Button();
 if(button == 1) {
   lcd.clear();
   d = d + 1;
```

```
  if(d > 3) d = 3;
 }
 if(button == 2) {
  lcd.clear();
  d = d - 1;
  if(d < 0) d = 0;
 }
 if(button == 4) {
  lcd.clear();
  //send RF data for power change
  p2 = p2 + 10;
  if(p2 > 100) p2 = 100;
  string_no = 60 + (p2/10);
  RF_Send(string_no);
 }
 if(button == 3) {
  lcd.clear();
  //send RF data for power change
  p2 = p2 - 10;
  if(p2 < 0) p2 = 0;
  string_no = 60 + (p2/10);
  RF_Send(string_no);
 }
 if(button == 5) {
  lcd.clear();
  if(d == 0) {
    //send RF code for sutter toggle
    s2 = ! s2;
    if(s2 == 0)
      string_no = 20;
    if(s2 == 1)
      string_no = 21;
    RF_Send(string_no);
  }
  if(d == 1) {
    //send RF code for LASER OFF
    int k2 = sure();
    if(k2 == 0) {
      if(p2 == 0) {
        string_no = 23;
        RF_Send(string_no);
        l2 = 0;
        c = 1;
        lcd.print("   LASER  OFF");
        delay(800);
      }
      else {
        lcd.print(" POWER MUST BE");
        lcd.setCursor(4, 1); lcd.print("!!ZERO!!");
        delay(1000);
```

```
          lcd.clear();
        }
       }
      }
      if(d == 2) {
       //for lock
       blnk = 0;
       delay(500);
       lcd.clear(); lcd.setCursor(0, 0); lcd.print("LASER 2  RUNNING");
       lcd.setCursor(3, 1); lcd.print("!!LOCKED!!");
       x = 0;
       while(x < 1) {
        Button();
        if(button == 5) {
         ACCESS_CODE(1);
         if(x == 0) {
          lcd.clear(); lcd.setCursor(1, 0); lcd.print("LASER 2  RUNNING");
          lcd.setCursor(3, 1); lcd.print("!!LOCKED!!");
         }
        }
       }
       string_no = 24;
       RF_Send(string_no);
       lcd.clear();
      }
      if(d == 3) {
       c = 1; //break while loop
      }
     }
    }//while end
}
void LASER_THREE() {
  if(l3 == 0) {
    string_no = 32;
    RF_Send(string_no);
    l3 = 1;
  }
  int e = 0; int f = 0;
  lcd.clear();
  while(e < 1) {
    //blnk = 0;
    if(f == 0) {
      blnk = 0;
      lcd.setCursor(0, 0); lcd.print("[SH-");
      if(s3 == 0) {
        lcd.setCursor(4, 0); lcd.print("OFF]");
      }
      if(s3 == 1) {
        lcd.setCursor(5, 0); lcd.print("ON]");
      }
```

```
  lcd.setCursor(9, 0); lcd.print("POW-");
  lcd.setCursor(13, 0); lcd.print(p3);
  lcd.setCursor(0, 1); lcd.print("LAS_3_OFF |");
  lcd.setCursor(12, 1); lcd.print("LOCK");
  delay(400);
 }
 if(f == 1) {
  blnk = 4;
  lcd.setCursor(0, 0); lcd.print("SH-");
  if(s3 == 0) {
    lcd.setCursor(3, 0); lcd.print("OFF |");
  }
  if(s3 == 1) {
    lcd.setCursor(4, 0); lcd.print("ON |");
  }
  lcd.setCursor(9, 0); lcd.print("POW-");
  lcd.setCursor(13, 0); lcd.print(p3);
  menu_string = "LAS_3_OFF";
  lcd.setCursor(10, 1); lcd.print("| LOCK");
 }
 if(f == 2) {
  blnk = 5;
  lcd.setCursor(0, 0); lcd.print("SH-");
  if(s3 == 0) {
    lcd.setCursor(3, 0); lcd.print("OFF |");
  }
  if(s3 == 1) {
    lcd.setCursor(4, 0); lcd.print("ON |");
  }
  lcd.setCursor(9, 0); lcd.print("POW-");
  lcd.setCursor(13, 0); lcd.print(p3);
  lcd.setCursor(0, 1); lcd.print("LAS_3_OFF |");
  menu_string = "LOCK";
 }
 if(f == 3) {
  blnk = 4;
  lcd.setCursor(0, 0); lcd.print("LAS_3_OFF |");
  lcd.setCursor(12, 0); lcd.print("LOCK");
  menu_string = "BACK";
 }
 delay(100);
 Button();
 if(button == 1) {
  lcd.clear();
  f = f + 1;
  if(f > 3) f = 3;
 }
 if(button == 2) {
  lcd.clear();
  f = f - 1;
```

```
  if(f < 0) f = 0;
 }
 if(button == 4) {
  lcd.clear();
  //send RF data for power change
  p3 = p3 + 10;
  if(p3 > 100) p3 = 100;
  string_no = 80 + (p3/10);
  RF_Send(string_no);
 }
 if(button == 3) {
  lcd.clear();
  //send RF data for power change
  p3 = p3 - 10;
  if(p3 < 0) p3 = 0;
  string_no = 80 + (p3/10);
  RF_Send(string_no);
 }
 if(button == 5) {
  lcd.clear();
  if(f == 0) {
   //send RF code for sutter toggle
   s3 = ! s3;
   if(s3 == 0)
    string_no = 30;
   if(s3 == 1)
    string_no = 31;
   RF_Send(string_no);
  }
  if(f == 1) {
   //send RF code for LASER OFF
   int k3 = sure();
   if(k3 == 0) {
    if(p3 == 0) {
     string_no = 33;
     RF_Send(string_no);
     l3 = 0;
     e = 1;
     lcd.print("   LASER  OFF");
     delay(800);
    }
    else {
     lcd.print(" POWER MUST BE");
     lcd.setCursor(4, 1); lcd.print("!!ZERO!!");
     delay(1000);
     lcd.clear();
    }
   }
  }
  if(f == 2) {
```

```
      //for lock
      blnk = 0;
      delay(500);
      lcd.clear(); lcd.setCursor(0, 0); lcd.print("LASER 3  RUNNING");
      lcd.setCursor(3, 1); lcd.print("!!LOCKED!!");
      x = 0;
      while(x < 1) {
        Button();
        if(button == 5) {
          ACCESS_CODE(1);
          if(x == 0) {
            lcd.clear(); lcd.setCursor(0, 0); lcd.print("LASER 3  RUNNING");
            lcd.setCursor(3, 1); lcd.print("!!LOCKED!!");
          }
        }
      }
      string_no = 34;
      RF_Send(string_no);
      lcd.clear();
    }
    if(f == 3) {
      e = 1; //break while loop
    }
  }
 }//while end
}

void RF_Send(int s) {
  const char *msg = " ";
  if(s == 10) msg = "S10";
  if(s == 11) msg = "S11";
  if(s == 12) msg = "L11";
  if(s == 13) msg = "L10";
  if(s == 14) msg = "K10";  //for remote 2, msg = K11 and for remote 3, msg = K12
  if(s == 40) msg = "P10";
  if(s == 41) msg = "P11";
  if(s == 42) msg = "P12";
  if(s == 43) msg = "P13";
  if(s == 44) msg = "P14";
  if(s == 45) msg = "P15";
  if(s == 46) msg = "P16";
  if(s == 47) msg = "P17";
  if(s == 48) msg = "P18";
  if(s == 49) msg = "P19";
  if(s == 50) msg = "P1:";

  if(s == 20) msg = "S20";
  if(s == 21) msg = "S21";
  if(s == 22) msg = "L21";
  if(s == 23) msg = "L20";
```

```
  if(s == 24) msg = "K20";
  if(s == 60) msg = "P20";
  if(s == 61) msg = "P21";
  if(s == 62) msg = "P22";
  if(s == 63) msg = "P23";
  if(s == 64) msg = "P24";
  if(s == 65) msg = "P25";
  if(s == 66) msg = "P26";
  if(s == 67) msg = "P27";
  if(s == 68) msg = "P28";
  if(s == 69) msg = "P29";
  if(s == 70) msg = "P2:";

  if(s == 30) msg = "S30";
  if(s == 31) msg = "S31";
  if(s == 32) msg = "L31";
  if(s == 33) msg = "L30";
  if(s == 34) msg = "K34";
  if(s == 80) msg = "P30";
  if(s == 81) msg = "P31";
  if(s == 82) msg = "P32";
  if(s == 83) msg = "P33";
  if(s == 84) msg = "P34";
  if(s == 85) msg = "P35";
  if(s == 86) msg = "P36";
  if(s == 87) msg = "P37";
  if(s == 88) msg = "P38";
  if(s == 89) msg = "P39";
  if(s == 90) msg = "P3:";

  //more if conditions will be added
  vw_send((uint8_t *)msg, strlen(msg));
  vw_wait_tx();
}

void SETTINGS(){
  int g = 0; int h = 0;
  lcd.clear();
  while(g < 1) {
    blnk = 1; //blink on
    if(h == 0) {
      menu_string = "PASSWORD OPTION";
      lcd.setCursor(0, 1); lcd.print("DEFAULT LASER");
    }
    if(h == 1) {
      menu_string = "DEFAULT LASER";
      lcd.setCursor(0, 1); lcd.print("BACK");
    }
    if(h == 2) {
      menu_string = "BACK";
```

```
    }
    Button();
    if(button == 1) {
      lcd.clear();
      h = h + 1;
      if(h > 2) h = 2;
    }
    if(button == 2) {
      lcd.clear();
      h = h - 1;
      if(h < 0) h = 0;
    }
    if(button == 5) {
      if(h == 0)
        PASSWORD_OPTION();
      if(h == 1)
        DEFAULT_COMMUNICATION();
      if(h == 2)
        g = 1; //break while loop
    }
  } //end while loop
}  //end SETTINGS

void PASSWORD_OPTION() {
  int p = 0; int q = 0;
  lcd.clear();
  while(p < 1) {
    blnk = 1;
    if(q == 0) {
      menu_string = "CHANGE PASSWORD";
      if(pass_state == 0) {
        lcd.setCursor(0, 1); lcd.print("ENABLE PASSWORD");
      }
      if(pass_state == 1) {
        lcd.setCursor(0, 1); lcd.print("DISABLE PASSWORD");
      }
    }
    if(q == 1) {
      if(pass_state == 0)
        menu_string = "ENABLE PASSWORD";
      if(pass_state == 1)
        menu_string = "DISABLE PASSWORD";
      lcd.setCursor(0, 1); lcd.print("BACK");
    }
    if(q == 2) {
      menu_string = "BACK";
    }
    Button();
    if(button == 1) {
      lcd.clear();
```

```
      q = q + 1;
      if(q > 2) q = 2;
    }
    if(button == 2) {
      lcd.clear();
      q = q - 1;
      if(q < 0) q = 0;
    }
    if(button == 5) {
      if(q == 0) {
        CHANGE_PASSWORD();
      }
      if(q == 1) {
        pass_state = ! pass_state;
        EEPROM.write(5, pass_state);
      }
      if(q == 2) {
        p = 1;  //break while
      }
    }
  }  // end while
}  //end PASSWORD OPTION

void CHANGE_PASSWORD() {
  ACCESS_CODE(2);
   if(x == 1) {
     char new_password[4]; char verify_password[4];
     int e1 = 0; int f1 = 0; int g1 = 0;
     lcd.clear();
     while(e1 < 1) {
       blnk = 0;
       if(g1 == 0) {
         lcd.setCursor(0, 0); lcd.print("NEW PASSWORD");
         lcd.setCursor(0, 1);
         lcd.print(f1); lcd.write(2); lcd.write(2); lcd.write(2);
       }
       if(g1 == 1) {
         lcd.setCursor(0, 0); lcd.print("NEW PASSWORD");
         lcd.setCursor(0, 1);
         lcd.print("*"); lcd.print(f1); lcd.write(2); lcd.write(2);
       }
       if(g1 == 2) {
         lcd.setCursor(0, 0); lcd.print("NEW PASSWORD");
         lcd.setCursor(0, 1);
         lcd.print("*"); lcd.print("*"); lcd.print(f1); lcd.write(2);
       }
       if(g1 == 3) {
         lcd.setCursor(0, 0); lcd.print("NEW PASSWORD");
         lcd.setCursor(0, 1);
         lcd.print("*"); lcd.print("*"); lcd.print("*"); lcd.print(f1);
```

```
  }
  delay(400);
  Button();
  if(button == 2) {
    lcd.clear();
    f1 = f1 + 1;
    if(f1 > 9) f1 = 9;
  }
  if(button == 1) {
    lcd.clear();
    f1 = f1 - 1;
    if(f1 < 0) f1 = 0;
  }
  if(button == 3) {
    f1 = 0;
    lcd.clear();
    g1 = g1 - 1;
    if(g1 < 0) g1 = 0;
    new_password[g1] = f1;
  }
  if(button == 4 && g1 != 3) {
    new_password[g1] = f1;
    lcd.clear();
    g1 = g1 + 1;
    f1 = 0;
  }
  if(button == 5 && g1 == 3) {
    new_password[g1] = f1;
    e1 = 1;
    lcd.clear();
  }
}
e1 = 0;
f1 = 0;
g1 = 0;
while(e1 < 1) {
  blnk = 0;
  if(g1 == 0) {
    lcd.setCursor(0, 0); lcd.print("VERIFY PASSWORD");
    lcd.setCursor(0, 1);
    lcd.print(f1); lcd.write(2); lcd.write(2); lcd.write(2);
  }
  if(g1 == 1) {
    lcd.setCursor(0, 0); lcd.print("VERIFY PASSWORD");
    lcd.setCursor(0, 1);
    lcd.print("*"); lcd.print(f1); lcd.write(2); lcd.write(2);
  }
  if(g1 == 2) {
    lcd.setCursor(0, 0); lcd.print("VERIFY PASSWORD");
    lcd.setCursor(0, 1);
```

```
  lcd.print("*"); lcd.print("*"); lcd.print(f1); lcd.write(2);
 }
 if(g1 == 3) {
  lcd.setCursor(0, 0); lcd.print("VERIFY PASSWORD");
  lcd.setCursor(0, 1);
  lcd.print("*"); lcd.print("*"); lcd.print("*"); lcd.print(f1);
 }
 delay(400);
 Button();
 if(button == 2) {
  lcd.clear();
  f1 = f1 + 1;
  if(f1 > 9) f1 = 9;
 }
 if(button == 1) {
  lcd.clear();
  f1 = f1 - 1;
  if(f1 < 0) f1 = 0;
 }
 if(button == 3) {
  f1 = 0;
  lcd.clear();
  g1 = g1 - 1;
  if(g1 < 0) g1 = 0;
  verify_password[g1] = f1;
 }
 if(button == 4 && g1 != 3) {
  verify_password[g1] = f1;
  lcd.clear();
  g1 = g1 + 1;
  f1 = 0;
 }
 if(button == 5 && g1 == 3) {
  verify_password[g1] = f1;
  e1 = 1;
  lcd.clear();
 }
}
 for(int h1 = 0; h1 < 4; h1++) {
  if(new_password[h1] == verify_password[h1]) {
    if(h1 == 3) {
      delay(100);
      for(h1 = 0; h1 < 4; h1++) {
        EEPROM.write((h1 + 1), new_password[h1]);
        lcd.clear(); lcd.print("PASSWORD CHANGED");
        delay(1000);
      }
      h1 = 3;
    }
  }
```

```
        if(new_password[h1] != verify_password[h1]) {
          lcd.clear(); lcd.setCursor(5, 0); lcd.print("FAILED");
          h1 = 4; //end for loop
          delay(1000);
        }
      }
    }
}

void DEFAULT_COMMUNICATION() {
  int r = 0; int t = 0;
  lcd.clear();
  while(r < 1) {
    blnk = 1;
    if(t == 0) {
      if(default_state == 0) menu_string = "ACTIVATE";
      if(default_state == 1) menu_string = "DEACTIVATE";
      lcd.setCursor(0, 1); lcd.print("BACK");
    }
    if(t == 1) {
      menu_string = "BACK";
    }
    Button();
    if(button == 1) {
      lcd.clear();
      t = t + 1;
      if(t > 1) t = 1;
    }
    if(button == 2) {
      lcd.clear();
      t = t - 1;
      if(t < 0) t = 0;
    }
    if(button == 5) {
      if(t == 0) {
        default_state = ! default_state;
        EEPROM.write(6, default_state);
        if(default_state == 1)
          ACTIVATE();
      }
      if(t == 1)
        r = 1;
    }
  } //end while
}  //end DEFAULT_COMMUNICATION

void ACTIVATE() {  //no back option
  int u = 0; int v = 0;
  lcd.clear();
  while(u < 1) {
```

```
    blnk = 1;
    if(v == 0) {
      menu_string = "LASER ONE";
      lcd.setCursor(0, 1); lcd.print("LASER TWO");
    }
    if(v == 1) {
      menu_string = "LASER TWO";
      lcd.setCursor(0, 1); lcd.print("LASER THREE");
    }
    if(v == 2) {
      menu_string = "LASER THREE";
    }
    Button();
    if(button == 1) {
      lcd.clear();
      v = v + 1;
      if(v > 2) v = 2;
    }
    if(button == 2) {
      lcd.clear();
      v = v - 1;
      if(v < 0) v = 0;
    }
    if(button == 5) {
      lcd.clear();
      EEPROM.write(0, v);
      u = 1;
    }
  } // end while
}  //end ACTIATE()

void ACCESS_CODE(int d1) {
  int a1 = 0; int b1 = 0; int c1 = 0;
  lcd.clear(); blnk = 0;
  while(a1 < 1) {
    if(c1 == 0) {
      lcd.setCursor(0, 0); if(d1 == 1) lcd.print("ENTER PASSWORD"); if(d1 == 2)
lcd.print("CURRENT PASSWORD");
      lcd.setCursor(0, 1);
      lcd.print(b1); lcd.write(2); lcd.write(2); lcd.write(2);
    }
    if(c1 == 1) {
      lcd.setCursor(0, 0); if(d1 == 1) lcd.print("ENTER PASSWORD"); if(d1 == 2)
lcd.print("CURRENT PASSWORD");
      lcd.setCursor(0, 1);
      lcd.print("*"); lcd.print(b1); lcd.write(2); lcd.write(2);
    }
    if(c1 == 2) {
      lcd.setCursor(0, 0); if(d1 == 1) lcd.print("ENTER PASSWORD"); if(d1 == 2)
lcd.print("CURRENT PASSWORD");
```

```
    lcd.setCursor(0, 1);
    lcd.print("*"); lcd.print("*"); lcd.print(b1); lcd.write(2);
   }
   if(c1 == 3) {
    lcd.setCursor(0, 0); if(d1 == 1) lcd.print("ENTER PASSWORD"); if(d1 == 2)
lcd.print("CURRENT PASSWORD");
    lcd.setCursor(0, 1);
    lcd.print("*"); lcd.print("*"); lcd.print("*"); lcd.print(b1);
   }
   delay(400);
   Button();
   if(button == 2) {
    lcd.clear();
    b1 = b1 + 1;
    if(b1 > 9) b1 = 9;
   }
   if(button == 1) {
    lcd.clear();
    b1 = b1 - 1;
    if(b1 < 0) b1 = 0;
   }
   if(button == 3) {
    b1 = 0;
    lcd.clear();
    c1 = c1 - 1;
    if(c1 < 0) c1 = 0;
    password[c1] = b1;
   }
   if(button == 4 && c1 != 3) {
    password[c1] = b1;
    lcd.clear();
    c1 = c1 + 1;
    b1 = 0;
   }
   if(button == 5 && c1 == 3) {
    password[c1] = b1;
    for(int z1 = 0; z1 < 4; z1++) {
     if(password[z1] == EEPROM.read(z1+1)) {
      if(z1 == 3) {
       x = 1;
       if(d1 == 1) {
        lcd.clear(); lcd.setCursor(4, 0); lcd.print("UNLOCKED");
        delay(1000);
       }
       a1 = 1;
      }
     }
     if(password[z1] != EEPROM.read(z1+1)) {
      x = 0;
      lcd.clear(); lcd.setCursor(1, 0); lcd.print("WRONG PASSWORD");
```

```
        a1 = 1;
        z1 = 4; //end for loop
        delay(1000);
       }
      }
    }
  }  //end while
} //end function

int sure() {
  int i2 = 0; int j2 = 0;
  lcd.clear();
  while(i2 < 1) {
    if(j2 == 0) {
      blnk = 2;
      lcd.setCursor(2, 0); lcd.print("ARE YOU SURE!");
      menu_string = "YES";
      lcd.setCursor(11, 1); lcd.print("NO");
    }
    if(j2 == 1) {
      blnk = 3;
      lcd.setCursor(2, 0); lcd.print("ARE YOU SURE!");
      lcd.setCursor(3, 1); lcd.print("YES");
      menu_string = "NO";
    }
    Button();
    if(button == 3) {
      lcd.clear();
      j2 = j2 - 1;
      if(j2 < 0) j2 = 0;
    }
    if(button == 4) {
      lcd.clear();
      j2 = j2 + 1;
      if(j2 > 1) j2 = 1;
    }
    if(button == 5) {
      i2 = 1;
    }
  }
  lcd.clear();
  return(j2);
}
```

Rotational Stage Code:

```
//Test Code for Rotational Stage
#include <EEPROM.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
```

```
int button = 0;
int Blink = 0;

int left = 0;
int right = 0;
int up = 0;
int down = 0;
int ok = 0;

int s = 0;

const char *menu_string = "";
const char *space_string = "                ";

int a = 1;
int b = 2;
int c = 3;
int d = 4;

void setup() {
  pinMode(a, OUTPUT);
  pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);

  lcd.begin(16, 2);
  lcd.setCursor(4, 0);
  lcd.print("LOADING");
  for(int z = 1; z <= 10; z++) {
    if(z < 10) lcd.setCursor(7, 1);
    if(z == 10) lcd.setCursor(6, 1);
    lcd.print(z*10);
    lcd.setCursor(9, 1);
    lcd.print("%");
    delay(250);
    lcd.setCursor(6, 1);
    lcd.print("   ");
    delay(250);
  }
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("MADE BY Dept. of");
  lcd.setCursor(0, 1);
  lcd.print("  PHYSICS SUST");
  delay(2000);
}

void loop() {
  int b = 0;
```

```
  int t = 0;
  lcd.clear();
  while(t < 1) {
    Blink = 1;
    if(b == 0) {
      menu_string = "START CONTROL";
      lcd.setCursor(0, 1); lcd.print("SETTINGS");
    }
    if(b == 1) {
      lcd.setCursor(0, 0); lcd.print("START CONTROL");
      menu_string = "SETTINGS";
    }
    BUTTON(b);

    if(button == 4) {
      lcd.clear();
      b = b + 1;
      if(b > 1) b = 1;
    }

    if(button == 3) {
      lcd.clear();
      b = b - 1;
      if(b < 0) b = 0;
    }

    if(button == 5) {
      if(b == 0) MAIN();
      if(b == 1) SETTINGS();
    }
  }
}

void MAIN() {
  lcd.clear();
  int k = EEPROM.read(2);
  int l = 0; int m = 0;
  while(l < 1) {
    if(m == 0) {
      lcd.clear();
      Blink = 0;
      lcd.setCursor(0, 0); lcd.print("DEGREE = "); lcd.setCursor(9, 0); lcd.print(k);
      lcd.setCursor(0, 1); lcd.print("BACK");
    }
    if(m == 1) {
      lcd.clear();
      Blink = 1;
      lcd.setCursor(0, 0); lcd.print("DEGREE = "); lcd.setCursor(9, 0); lcd.print(k);
      menu_string = "BACK";
    }
```

```
    delay(100);
    BUTTON(m);
    if(button == 4) {
      lcd.clear();
      m = m + 1;
      if(m > 1) m = 1;
    }
    if(button == 3) {
      lcd.clear();
      m = m - 1;
      if(m < 0) m = 0;
    }
    if(button == 2) {
      k = k + (2*EEPROM.read(0));
      if(k == 360) k = 0;
      EEPROM.write(2, k);
      for(s = 0; s < EEPROM.read(0); s++) {
        PULSE(EEPROM.read(3));
      }
    }
    if(button == 5 && m == 1) {
      lcd.clear();
      l = 1;
    }
  }
}

void SETTINGS() {
  lcd.clear();
  int d = 0; int e = 0;
  while(d < 1) {
    Blink = 1;
    if(e == 0) {
      menu_string = "SELECT STEP";
      lcd.setCursor(0, 1); lcd.print("RESET TO ZERO");
    }
    if(e == 1) {
      lcd.setCursor(0, 0); lcd.print("SELECT STEP");
      menu_string = "RESET TO ZERO";
    }
    if(e == 2) {
      lcd.setCursor(0, 0); lcd.print("RESET TO ZERO");
      menu_string = "BACK";
    }
    BUTTON(e);
    if(button == 4) {
      lcd.clear();
      e = e + 1;
      if(e > 2) e = 2;
    }
```

```
    if(button == 3) {
      lcd.clear();
      e = e - 1;
      if(e < 0) e = 0;
    }

    if(button == 5) {
      if(e == 0) DEGREE();
      if(e == 1) RESET();
      if(e == 2) d = 1;
    }
  }   //while d end
}

void DEGREE() {
  lcd.clear();
  int f = 0; int g = 0; int h = 0;
  while(f < 1) {
    if(g == 0) {
      Blink = 0;
      lcd.setCursor(0, 0); lcd.print("STEP = "); lcd.setCursor(7, 0); lcd.print(2*h);
      lcd.setCursor(0, 1); lcd.print("OK");
    }
    if(g == 1) {
      Blink = 1;
      lcd.setCursor(0, 0); lcd.print("STEP = "); lcd.setCursor(7, 0); lcd.print(2*h);
      menu_string = "OK";
    }
    delay(100);
    BUTTON(g);
    if(button == 4) {
      lcd.clear();
      g = g + 1;
      if(g > 1) g = 1;
    }
    if(button == 3) {
      lcd.clear();
      g = g - 1;
      if(g < 0) g = 0;
    }
    if(button == 2) {
      lcd.clear();
      h = h + 1;    //eaach step change by 2
      if(h > 180) h = 180;
    }
    if(button == 1) {
      lcd.clear();
      h = h - 1;
      if(h < 1) h = 1;
```

```
    }
    if(g == 1 && button == 5) {
      lcd.clear();
      EEPROM.write(0, h);
      f = 1;
    }
  }  //while f end
}   //BUTTON END

void RESET() {
  lcd.clear();
  int i = 0; int j = 0;
  while(i < 1) {
    if(j == 0) {
      Blink = 2;
      lcd.setCursor(2, 0); lcd.print("ARE YOU SURE");
      menu_string = "YES";
      lcd.setCursor(11, 1); lcd.print("NO");
    }
    if(j == 1) {
      Blink = 3;
      lcd.setCursor(2, 0); lcd.print("ARE YOU SURE");
      lcd.setCursor(3, 1); lcd.print("YES");
      menu_string = "NO";
    }
    BUTTON(1);
    if(button == 2) {
      lcd.clear();
      j = j + 1;
      if(j > 1) j = 1;
    }
    if(button == 1) {
      lcd.clear();
      j = j- 1;
      if(j < 0) j = 0;
    }
    if(button == 5) {
      lcd.clear();
      if(j == 0) {
        i = 1;
        EEPROM.write(2, 0);
      }
      if(j == 1) {
        i = 1;
      }
    }
  }     //while end
}      //reset end

void BUTTON(int c) {                          //BUTTON FUNCTION
```

```
if(c > 1) c = 1;
int a = 0;
while(a < 1) {
 if(Blink == 1) {
  lcd.setCursor(0, c); lcd.print(menu_string);
  delay(150);
  lcd.setCursor(0, c); lcd.print(space_string);
  delay(150);
 }
 if(Blink == 2) {
  lcd.setCursor(3, 1); lcd.print(menu_string);
  delay(150);
  lcd.setCursor(3, 1); lcd.print("   ");
  delay(150);
 }
 if(Blink == 3) {
  lcd.setCursor(11, 1); lcd.print(menu_string);
  delay(150);
  lcd.setCursor(11, 1); lcd.print("  ");
  delay(150);
 }

 else delay(50);

 left = analogRead(A0);
 right = analogRead(A1);
 up = analogRead(A2);
 down = analogRead(A3);
 ok = analogRead(A4);

 if(left > 500) {
  button = 1;
  a = 1;
 }

 if(right > 500) {
  button = 2;
  a = 1;
 }

 if(up > 500) {
  button = 3;
  a = 1;
 }

 if(down > 500) {
  button = 4;
  a = 1;
 }
```

```
    if(ok > 500) {
      button = 5;
      a = 1;
    }
  }  // while end
}

void PULSE(int n) {
  if(n == 0) {
    for(int o = 1; o <= 3; o++) {
      digitalWrite(a, HIGH);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      delay(150);

      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
      delay(150);

      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, HIGH);
      digitalWrite(d, LOW);
      delay(150);

      digitalWrite(a, LOW);
      digitalWrite(b, LOW);
      digitalWrite(c, LOW);
      digitalWrite(d, HIGH);
      delay(150);
    }
    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    delay(150);
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
  }
  if(n == 1) {
    for(int p = 1; p <= 3; p++) {
      digitalWrite(a, LOW);
      digitalWrite(b, HIGH);
      digitalWrite(c, LOW);
      digitalWrite(d, LOW);
```

```
    delay(150);

    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    delay(150);

    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, HIGH);
    delay(150);

    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    delay(150);
  }
  digitalWrite(a, LOW);
  digitalWrite(b, HIGH);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
  delay(150);
  digitalWrite(a, LOW);
  digitalWrite(b, LOW);
  digitalWrite(c, LOW);
  digitalWrite(d, LOW);
}
if(n == 2) {
  for(int q = 1; q <= 3; q++) {
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    delay(150);

    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, HIGH);
    delay(150);

    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    delay(150);
```

```
    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    delay(150);
   }
   digitalWrite(a, LOW);
   digitalWrite(b, LOW);
   digitalWrite(c, HIGH);
   digitalWrite(d, LOW);
   delay(150);
   digitalWrite(a, LOW);
   digitalWrite(b, LOW);
   digitalWrite(c, LOW);
   digitalWrite(d, LOW);
}
if(n == 3) {
  for(int r = 0; r <= 3; r ++) {
    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, HIGH);
    delay(150);

    digitalWrite(a, HIGH);
    digitalWrite(b, LOW);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    delay(150);

    digitalWrite(a, LOW);
    digitalWrite(b, HIGH);
    digitalWrite(c, LOW);
    digitalWrite(d, LOW);
    delay(150);

    digitalWrite(a, LOW);
    digitalWrite(b, LOW);
    digitalWrite(c, HIGH);
    digitalWrite(d, LOW);
    delay(150);
   }
   digitalWrite(a, LOW);
   digitalWrite(b, LOW);
   digitalWrite(c, LOW);
   digitalWrite(d, HIGH);
   delay(150);
   digitalWrite(a, LOW);
   digitalWrite(b, LOW);
   digitalWrite(c, LOW);
```

```
    digitalWrite(d, LOW);
  }
  n = n + 1;
  if(n > 3) n = 0;
  EEPROM.write(3, n);
}
```