Sudan University of Sciences and Technology

College of Engineering

School of Electrical and Nuclear Engineering

An Elevator control system using microcontroller

نظام التحكم فى المصعد بإستخدام مايكروكنترول

A Project Submitted In Partial Fulfillment for the Requirements of the Degree of B.Sc. (Honor) In Electrical Engineering

Prepared By:

1. Abdullah Abdalmonem Abdullah Mohammed

2. Hamadnallah Gaber Hamadnallah Alasha

3. Mohammed Fadoul Alkreem Ahmed Essoi

4. Mohammed Towfeg Abdulraheim Hassan

Supervised By:

Dr. Awadalla Taifour Ali

November 2020

لآية

قال تعالي:

وَيَسْأَلُونَكَ عَنِ الرُّوحِ ۖ قُلِ الرُّوحُ مِنْ أَمْرِ رَبِّي وَمَا أُوتِيتُمْ مِنَ الْعِلْمِ إِلَّا قَلِيلًا (٨٥)

الإسراء: (85)

DEDICATION

To our great parents, who never stop giving us themselves in contless ways.

To our dearest friends, who leads us through the valley of darkness with light of hope and give us encourage and support.

To our beloved brothers and sisters who standy by us when things look bleak.

To all our family , the eymbol of love and giving.

To all the people in our life who touch our heart , we dedicate this research.

ACKNOWLEDGEMENT

We wish to express our profound gratitude to our Supervisor **Dr. Awadalla Taigour Ali** for his valuable guidance, continues encouragement ,worthwhile suggestions and constructive ideas throughout this project.

His support, pragmatic analysis and understanding made this study a success and knowledgeable experience for us.

ABSTRACT

In today`s society technology is growing at an exponential rate,
and elevator is very common example of technology for consuming time. An elevator is a platform that can move up and down in vertical direction by a mechanical mean. In the past elevators drive mechanism were powered by stream and water hydraulic piston. In today`s world, there are intricate governors and switching schemes to control elevators.

In our project, the arduino microcontroller based elevator system is constructed to simulate as an actual elevator, and it"s designed for multi-storage buildings.

المستخلص

فى مجتمع اليوم تنمو التكنولوجيا بمعدل متسارع ، والمصعد هو مثال شائع جدًا للتكنولوجيا لاستهلاك الوقت. مصعد عبارة عن منصة يمكن أن تتحرك لأعلى ولأسفل في اتجاه رأسي بواسطة وسيلة ميكانيكية. في الماضي ، كانت آلية دفع المصاعد تعمل بواسطة تيار ومكبس هيدروليكي للمياه .في عالم اليوم ، هناك متحكمات معقدة وانظمة تبديل للتحكم فى المصاعد .

فى هذا المشروع ، تم إنشاء ظام المصعد القائم على وحدة التحكم الدقيق . (Arduino) لمحاكاة المصعد الفعلي ، وقد صمم المبني على عدة طوابق

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

I/O	In put/Out put
IDE	Integrated Development Environment
SSD	Seven-Segment Display
PM	Permanent Magnet
VR	Variable Reluctance
HS	Hybrid Synchronous
GND	Ground
USB	Universal Serial Bus
DC	Direct Current
AC	Alternating Current
IC	Integrated Circuit
PM	Permanent Magnet
PCB	Printed Circuit Board
LED	Light Emitting Diode
PWM	Pulse Width Modulation
ICSP	In Circuit Serial Programming
FTDI	Future Technology Devices Interglobal
BCD	Binary Coded Decimal

CHAPTER ONE
INTRODUCTION

1.1 Background

Elevator, is a transport device that is very common nowadays, it used everyday to move goods or peoples vertically in a high building such as shopping center, working office, hotel and many more. It is a very useful device that moves people to the desired floor in the shortest time.

Elevators began as simple rope or chain hoists. An elevator is essentially a platform that is either pulled or pushed up by a mechanical means. A modern day elevator consists of a cab (also called a "cage" or "car") mounted on a platform within an enclosed space called a shaft or more correctly a hoist way. In the past elevator drive mechanisms were powered by steam and water hydraulic pistons.

In the 1800s, with the advent of electricity, the electric motor was integrated into elevator technology by German inventor Werner von Siemens. By 1903, this design had evolved into the gearless traction electric elevator, allowing hundred-plus story buildings to become possible and forever changing the urban landscape. Multi speed motors replaced the original single-speed models to help with landing-leveling and smoother overall operation. Electromagnet technology replaced manual rope-driven switching and braking. Besides, Push-button controls and various complex signal systems modernized the elevator even further. Safety improvements have been continual, including a notable development by Charles Otis.

Today, there are intricate governors and switching schemes to carefully control cab speeds in any situation. Buttons have been giving way to keypads. Virtually all commercial elevators operate automatically and the computer age has brought the microchip-based capability to operate vast banks of elevators with precise scheduling, maximized efficiency and extreme safety, {1}.

1.2 Problem Statement

electric elevators being powered and controlled by electromagnetic devices suffer much problems such as maintenance problems, difficulties inuse, any changes on software required changes on the hardware and system breaks.

1.3 Objectives

Microcontroller's use increased rapidly, they are cheap and very small in size, Programming of Microcontrollers is simple hence using Arduino Uno microcontroller give us advantages such maintenance is not required , Changes to be made on the software will not affect the components and flexibility in use

1.4 Methodology

Arduino Uno is used as the primary controller. The software for the system is written by C++, been designed according to the real elevator traffic management algorithm. The combination of the hardware and software perform the simulate function of a basic elevator system.

1.5 Layout

This project consist of five chapters, chapteronegives a background, problem statement, objectives and methodology. Chapter two definesthe elevators components, types and their working principles. Chapterthreeillustrates the control system in generally then coverselevator control system.Chapterfourgives abrief definitionfor the components which used in this project thendiscuss the project circuits and their analysis. Rather to the control system block diagram and its parts function and flow chart. Chapterfivecontains the conclusion and the recommendations.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

Elevator history begins several hundred years before Christ. The earliest elevators were called hoists. They were powered by human and animal power, or sometimes water-driven mechanisms. They were in use as early as the 3rd century BC.

Modern elevators were developed during the 1800s. These crude elevators slowly evolved from steam driven to hydraulic power. The first hydraulic elevators were designed using water pressure as the source of power.

They were used for conveying materials in factories, warehouses and mines. Hydraulic elevators were often used in European factories. In 1852, Elisha Graves Otis introduced the first safety contrivance for elevators.

Otis established a company for manufacturing elevators and went on to dominate the elevator industry. Today the Otis Elevator Factory is the world's largest manufacturer of vertical transport systems. Revolution in elevator technology began with the invention of hydraulic and electricity.

Motor technology and control methods evolved rapidly and electricity quickly became the accepted source of power. The safety and speed of these elevators were significantly enhanced. The first electric elevator was built by the German inventor WenerVon Siemens in 1880. In 1889, the first commercially successful electric elevator was installed. In 1887, an electric elevator with automatic doors that would close off the elevator shaft was patented. This invention made elevators safer.

Many changes in elevator design and installation was made by the great advances in electronic systems during World War II.

Space elevators use the same concept of classic elevator. They will be used to transport people to space station. This concept theoretically can considerably reduce the cost for putting a person into space.

Today, modern commercial buildings commonly have multiple elevators with a unified control system. In addition, all modern elevators have special override controls (to make elevators go directly to a specific floor without intermediate stops).

There are many usages of elevators in practical application such as: Passenger service, Freight elevators ,Stage elevators, Vehicle elevators, Boat elevator, Aircraft elevators, Residential elevator ,Paternoster and Scissor elevator, {2,3}.

- **Elevator Components generally:**
  The standard elevators will include the following basic components: Cab, Hoistway, Machine/drive system, Control system and Safety system.

2.2 Microcontroller

A microcontroller shown in figure 2.1 is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and Input/Output (I/O) peripherals on a single chip.

A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor. The temporary information that the microcontroller receives is stored in its data memory, where the processor accesses it and uses instructions stored in its program memory to decipher and apply the incoming data. It then uses its I/O peripherals to communicate and enact the appropriate action.

Microcontrollers are used in a wide array of systems and devices. Devices often utilize multiple microcontrollers that work together within the device to handle their respective tasks.For example, a car might have many microcontrollers that control various individual systems within, such as the anti-lock braking system, traction control, fuel injection or suspension control. All the microcontrollers communicate with each other to inform the correct actions. Some might communicate with a more complex central computer within the car, and others might only communicate with other microcontrollers. They send and receive data using their I/O peripherals and process that data to perform their designated tasks.Following are the most important facts about Microcontrollers, which causes rapid growth of their use:

-Microcontrollers are cheap and very small in size, therefore they can be embedded on any device.

-Programming of Microcontrollers is simple to learn. It’s not much complicated.

-We can use simulators on Computers to see the practical results of our program. Thus we can work on an Embedded project without even buying the required Components and Chips. Thus we can virtually see the working of our project or program, {4}.

Figure 2.1: Microcontroller

2.3 Arduino

Arduinoshown in Figure 2.2 is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package, {5}.

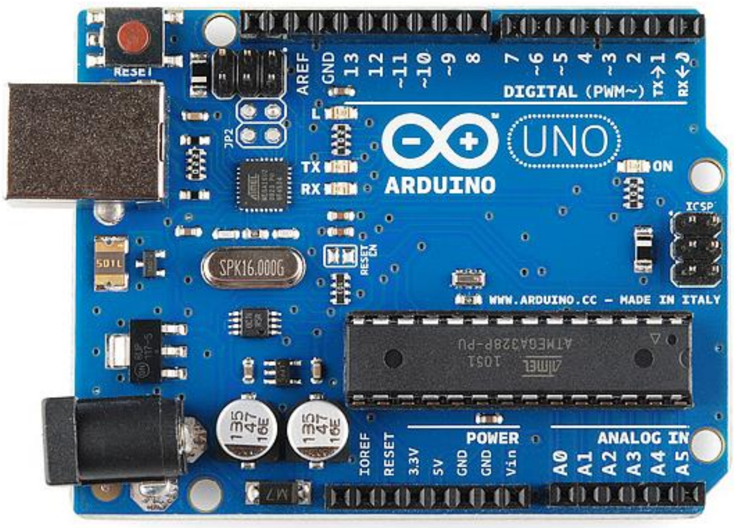

Figure 2.2 : Arduino

2.4 Seven Segment Display

A seven-segment display shown in Figure 2.3 is a form of electronic display device for displaying decimalnumerals that is an alternative to the more complex dot matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information, {6}.

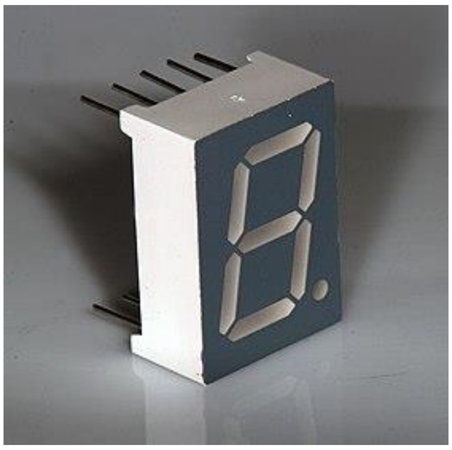

Figure 2.3 : 7-segement display

2.5 Stepper Motor

stepper motor shown in Figure 2.4 is an electromechanical device with ability to convert electrical pulse into discrete mechanical movements. The shaft of this motor rotates in a discrete stepincrement when electrical pulses are applied in a proper sequence. It is a brushless DC electric motor that divides a full rotation into a number of equal steps. Sizing of the motor depends on its torque and speed which determines its applications. This motor converts its train of input pulses into precisely defined increment in the shaft position with each pulses moving through a particular angle. Stepper motors have multiple „toothed" electromagnets arranged around center gear shaped piece of iron. An external drive circuit e.g. a microcontroller energizes the motor. To make the shaft rotate, one electromagnet is given power thus attracting the gear teeth. When these teeth are aligned to the first electromagnet, they are slightly offset from the next electromagnet. Thus when the next is turned on and the first turned off, the gear rotates slightly to align with the next one. The process is repeated with each rotation called a „step" with an integer number of steps making up a full rotation. Thus motor is turned with precise angle. This angle through which the motor shaft rotates for each command pulse is the step angle, {7}.

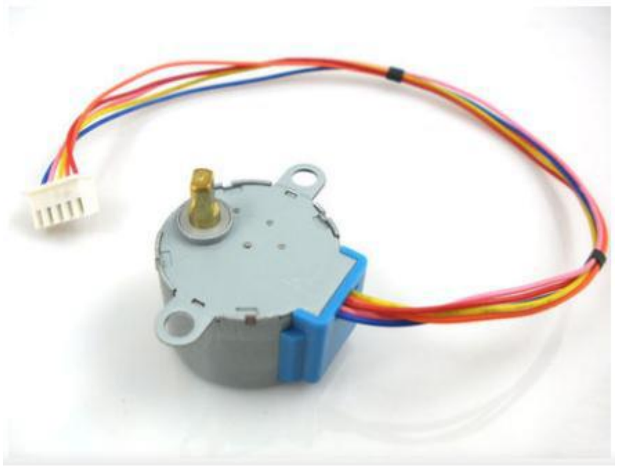
Figure 2.4 : Stepper Motor

Applications of Stepper Motors

This motors find applications in operation controls in computer peripherals, textile
industries, IC fabrications and robotics. They have advantage since positioninformation can be obtained by keeping count of pulses sent to the motor thuseliminating the need for expensive position sensors and feedback controls. They alsofind application in commercial, military and medical fields for mixing, cutting striking, metering and blending. However due to their good control over speed andposition, they are used today in elevators since have a robust design, no brushes and agood speed control.

Types Of Stepper Motor :

- Permanent Magnet Stepper. PM steppers have rotors that are constructed with permanent magnets, which interact with the electromagnets of the stator to create rotation and torque. PM steppers usually have comparatively low power requirements and can produce more torque per unit of input power.
- Variable Reluctance Stepper. VR stepper rotors are not built with permanent magnets. Rather, they are constructed with plain iron and resemble a gear, with protrusions or “teeth” around the circumference of the rotor. The teeth lead to VR steppers that have a very high degree of angular resolution; however, this accuracy usually comes at the expense of torque.
- Hybrid Synchronous Stepper. HS stepper rotors use the best features of both PM and VR steppers. The rotor in an HS motor has a permanent magnet core, while the circumference is built from plain iron and has teeth. A hybrid synchronous motor, therefore, has both high angular resolution *and* high torque.

2.6 Driver ULN2003a

The ULN2003 stepper motor driver PCB shown in Figure 2.5 provides a direct drive interface between your microcontroller and stepper motor. The PCB provides 4 inputs for connection to your microcontroller, power supply connection for the stepper motor voltage, and ON/OFF jumper, a direct connect stepper motor header and 4 LEDs to indicate stepping state.

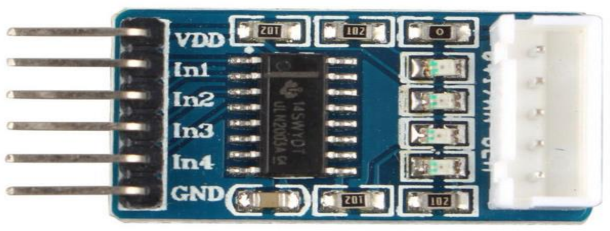

Figure 2.5 : DRIVER ULN200a

This driver acts as a buffer providing protection for the circuit from high voltage and taking the current to a definite range.

2.7 Elevator's Work Principle

The elevator compartment is raised and lowered by a hoist and pulley system and a moving counterweight .The
elevator is moves smoothly between vertical guide bars: it doesn't just dangle stupidly from the rope. The cable that does all the lifting wraps around several pulleys and the main winding drum. Don't forget this
elevator was invented before anyone was really using electricity: it was raised and lowered by hand. At the top of the elevator car, there's a simple mechanism made up of spring-loaded arms and pivots. If the main cable breaks, the springs push out two sturdy bars called "pawls" so they lock into vertical racks of upward- pointing teeth on either side. This ratchet-like device clamps the elevator safely in place. Most elevators have an entirely separate speed-regulating system called a governor, which is a heavy flywheel
with massive mechanical arms built inside it. Normally the arms are held

inside the flywheel by hefty springs, but if the lift moves too fast, they fly outward, pushing a lever mechanism that trips one or more braking systems. First, they might cut power to the lift motor. If that fails and the lift continues to accelerate, the arms will fly out even further and trip a second mechanism, applying the brakes. Some governors are entirely mechanical; others are electromagnetic, still others use a mixture of mechanical and electronic components.

The following list describes all the safety components used in electrical traction elevator safety system:

- Device for locking landing doors (Hoistway Door Interlock).
- Progressive safety gear.
- Over speed governor.
- Buffers.
- Final limit switches.
- Other safety devices and switches.

(a) Devices For Locking Landing Doors:

It shall not be possible in normal operation to open the landing& door or any of the panels in the case of a multi panel door unless the car has stopped, or is on the point of stopping, in the unlocking zone of the door.
the unlocking zone shall not extend more than 5.2 meter above and below the landing level. the hoistway door locking mechanism provides a means to mechanically lock each hoistway door and the elevator cannot leave a landing unless the doors are fully closed and secured.

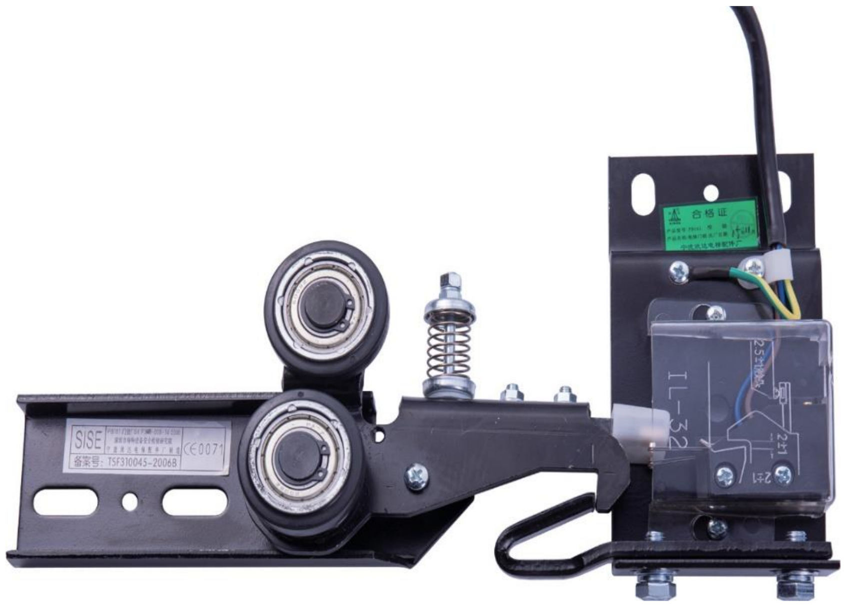
Figure 2.6 : Device for locking door

(b) Hoistway door interlock:

Hoistway door interlock shown in Figure 2.7 with the door been opened are also are interconnected electrically to prevent operation of the elevator if any of the elevators hoistway doors are open. hold the doors be forced open, the interlock circuit will be broken, causing the elevator to immediately stop. each landing door shall be provided with a locking device satisfying the previous conditions. This device shall be protected against deliberate misuse. landing doors shall be capable of being unlocked from the outside with the aid of key, which fit the unlocking triangle (Hoistway emergency door keys).

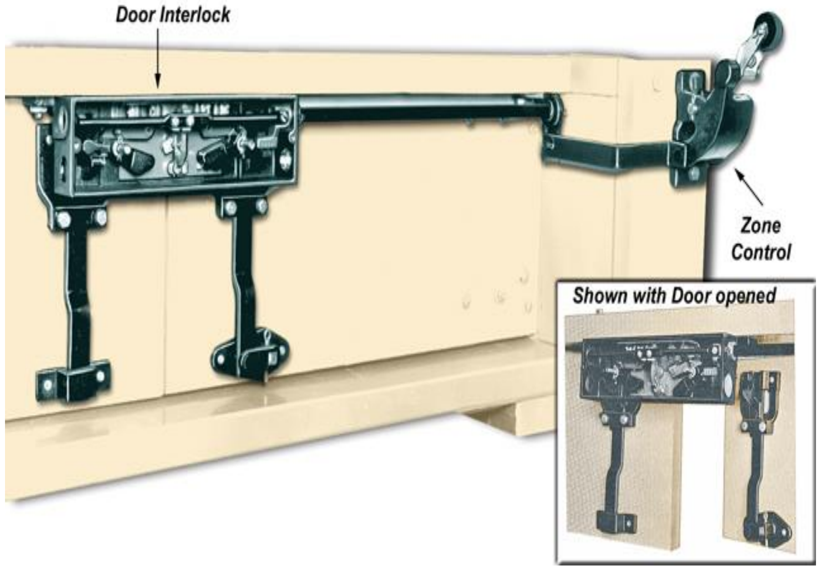

Fig ure 2.7 : Hoistway Door interlock

Hoistway Emergency Door keys:

Hoistway emergency door keys shown in figure 2.8 permit the unlocking of the hoistway door interlock

.

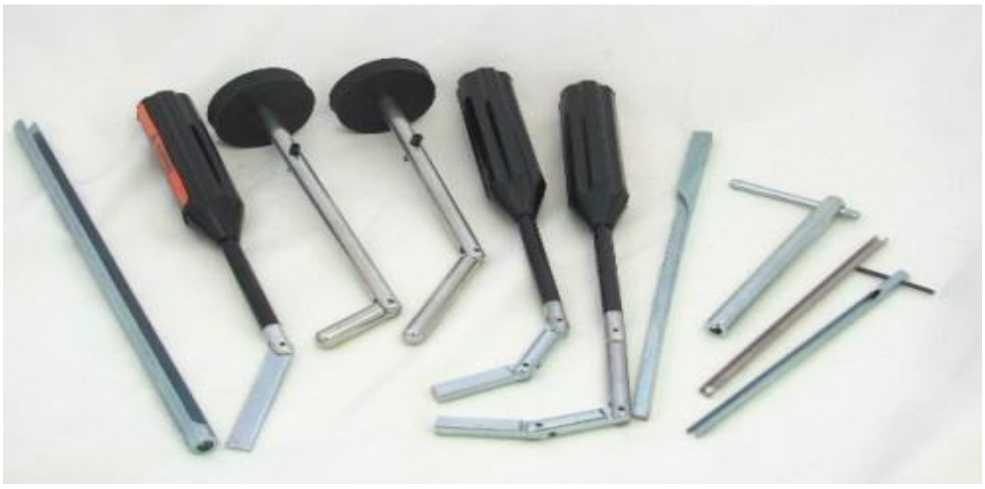

Figure 2.8 : Hoistway Door Emergency Keys

Escutcheon Tube:

The keyhole on the upper portion of a hoistway door that accepts a hoistway emergency door key and permits unlocking& of the hoistway door locking&

mechanism. these keyholes are usually located at the bottom and top floors, but may also be on other selected floors or all floors.

Elevator Safety Gear

Elevator safety gear is the part of elevator, which is the safety protection device for protecting elevator runs safely. Elevator safety gear is operated by elevator over speed governor, the function is that stops the elevator car and holds the elevator car at the elevator guide rail in an emergency when the running speed of elevator car exceeds the set speed by elevator over speed governor, or elevator suspension rope is in broken or loose situation. Elevator safety gear provides effective protection for elevator running in safe; usually it is installed on the elevator car or counterweight frame. In accordance with the different structure of its brake components, elevator safety gear can be divided into wedge-shaped block, eccentric gear, and roller. According to stop distance, it can be divided into instantaneous safety gear and progressive safety gear.

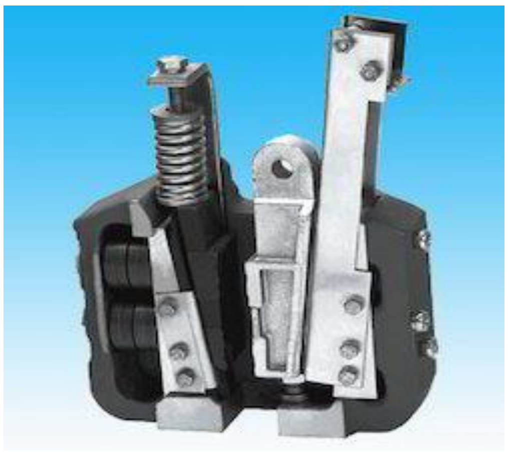

Figure 2.9 : Progressive Safety Gear

Over Speed Governor:

Over speed governor is one of the safety control components of elevator safety protection system. When the elevator in running, no matter what the speed governor in the car or the risk of falling or other situations of safety protection devices do not work, safety gear speed governor and linkage will start to work and make the elevator car stop running, to ensure the safety of the elevator. The speed governor is generally located in the elevator machine room. According to the requirements of the installation plan, it is generally installed on the floor of the machine room. However, the speed governor also can be directly installed on the bearing beam. The specific position of the speed governor is determined according to the construction drawing and the position of the car and guide rail after adjustment.

Buffers:

Elevator buffers are one of the most important items of the elevator safety system. Buffers work to minimize the danger when there is a loss of traction capability and not enough brake distance in cases such as rope breakage. So, we can think of the elevator buffer as a part of braking system. Thanks to the buffer, the cabin does not hit directly in a free fall. In this situation, the role of the buffers is to prevent the cabin from hitting hard when the cabin or counterweight is slipping toward the bottom of the shaft. The elevator buffer can be used in different structures, and can vary according to the elevators used; i.e., the elevator can be varied according to the number of people it will carry and its speed. For example, if the elevator car speed is high, energy-consuming buffers should be used and, if low, energy-conserving buffers should be used.

Final Limit Switches

Final limit switch is One of two mechanically operated switches mounted in an elevator hoistway, one at the top and one at the bottom, which if activated by the car, traveling more than a preset distance beyond a terminal landing, cuts off power to the elevator drive motor. The operation of this switch will prevent movement of the car by the normal operating devices in both directions of travel.

CHAPTER THREE

SYSTEM DESCIPTION AND CONTROL

3.1 Introduction

The elevator design process involves description of a series of
steps taken to make the logic for the 4 level elevators. Since design depends on the particular designer‟s interests and objectives, there is no single universally accepted design procedure thus each engineer has their own twist for how the process works. However, the most important thing is provision of a solution to a defined issue; in this case design of the elevator design process involves description of a series of steps taken to make the logic for the four level logic for a 4 level elevator. In this particular design, the following assumptions will be taken as a preference and also to simplify the project and keep it to the scope. The elevator starts and rests on the last floor of destination whenever there no more requests to be serviced. This is a one-way elevator thus it operates on a first come first serve basis ensuring that the first person to make request is attended to together with those in same direction but on the way of cab (UPWARD and DOWNWARD). Speed used is constant since floor number is small (4 levels). This is a sensor less elevator. Though sensors are used in the case of real life lifts to ensure safety for users. The stepper motor has given it this ability. v. This elevator accepts multiple requests and destinations while at rest on a floor.

3.2 System Definition

The input sends signals to the controller which processes the signal send to determine the necessary action. The action taken depends on the command issues at input and also the input signal received from the motor and the cab i.e. cab level/floor information which determines the necessary action by the controller. Thus all the logic operations are performed by the controller to enable up and down elevator motion, in order to determine the priorities in servicing the requests. The motor is connected to the cab through a belt-pulley system. This ensures movement of the cab to effect the logic operations of the elevator.

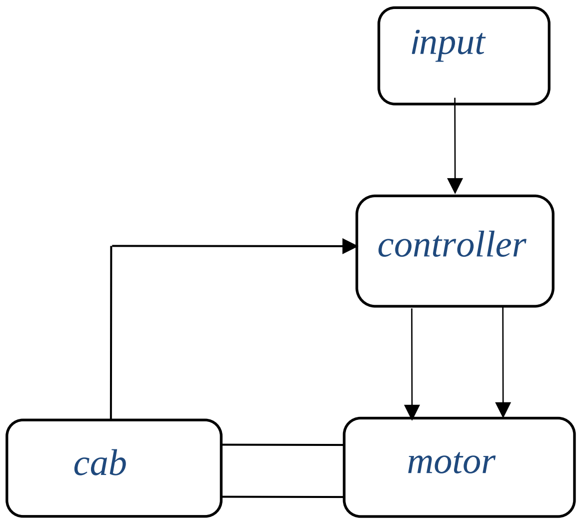

Figure 3.1: Flow Chart

Inputs

An input here refers to the data sent into the controller through a command issued

from a particular component device. In this case the commands sent to the controller are from push buttons described below.

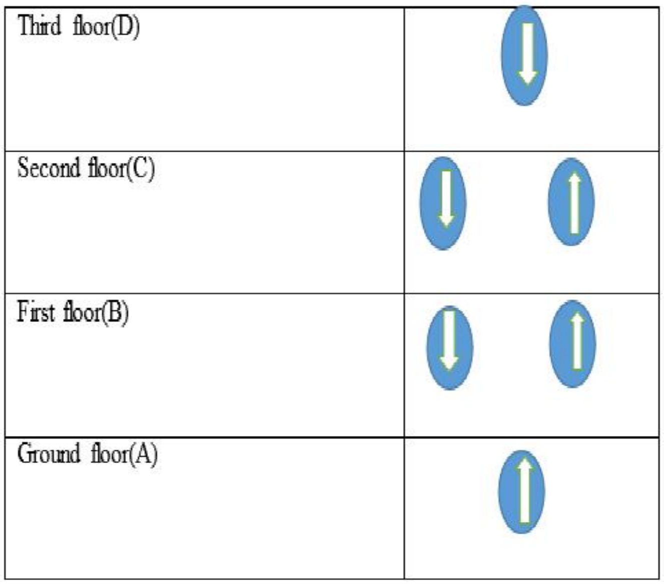

Figure 3.2: System Flow Chart

Motor

The motor i.e. the stepper motor runs depending on the pulses received from the controller. The number of steps to a particular flow levels are constant. The motor is coupled to the cab using a belt system to ensure movement up and down for servicing the floor request. Coupling is done by a belt drive mechanism.

Cab

This is the part that lifts the passengers up and down the particular flows of interests. Thus services the requests and destinations of passengers

The Controller

The Controller system is the system responsible for coordinating all aspects of elevator service such as travel, speed, and accelerating, decelerating, door opening speed and delay and levelling. It receives the inputs and responds to them by sending impulses to the motor via drives.

The controller also receives in formation like floor level, position and speed from the motor and the cab and process and displays the information on displays.

The controller is Arduino Uno. it acts as the heart of logic control in elevator system.

It accepts inputs (e.g. button signals) and produces outputs (e.g. elevator cars moving).

The main aims of the elevator control system:

- Is to bring the elevator cab to the correct floor.
- Is to minimize travel time.
- Is to maximize passenger comfort by providing a smooth ride.
- Is to ensure a safe speed limit for travel.

3.3 Controller Flow Chart

A number of actions are performed in the process to ensure proper elevator system operation. Below is a flow diagram describing a control logic design involved in operation of my elevator. The following points are to be noted about the flowchart representation of logic design indicated above. The elevator starts operating when the power is switched on. When under no use the lift rest at the last floor serviced (last destination floor).

The elevator keeps checking for the requests during the waiting to ensure fast serving of any request on any particular floor of request. The cab moves on each level and keeps checking for requests and destinations in the particular direction i.e. UPWARDS/DOWNWRDS depending on the direction of travel/motion. The elevator cab keeps checking the 4th and ground floor to ensure reverse of directions on those particular floors since they represent the highest and the lowest floors according to design scope. This ensures safety.

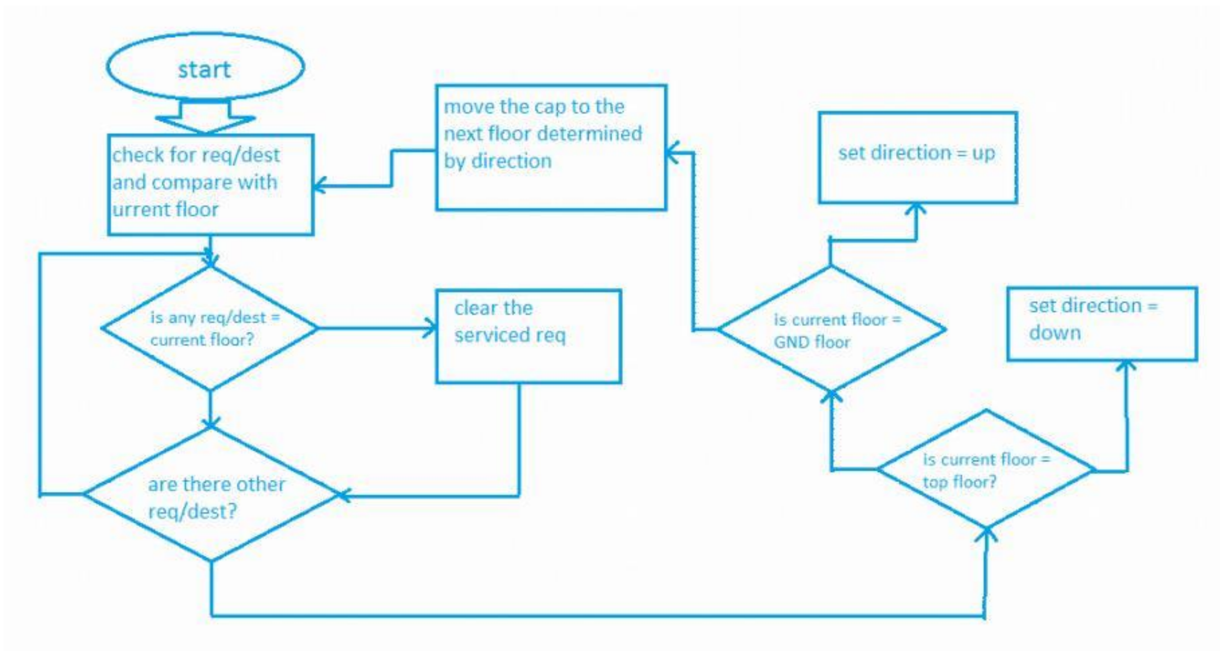
start
check for req/dest and compare with urrent floor
move the cap to the next floor determined by direction
set direction = up
is any req/dest = current floor?
clear the serviced req
is current floor = GND floor
set direction = down
are there other req/dest?
is current floor = top floor?

Figure 3.3 : Control flow Chart

CHAPTER FOUR

SYSTEM IMPLEMENTATION AND TESTING

4.1 components used

4.1.1 Arduino Uno

The Arduino Uno as shown in figure 4.1 is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, {8}.

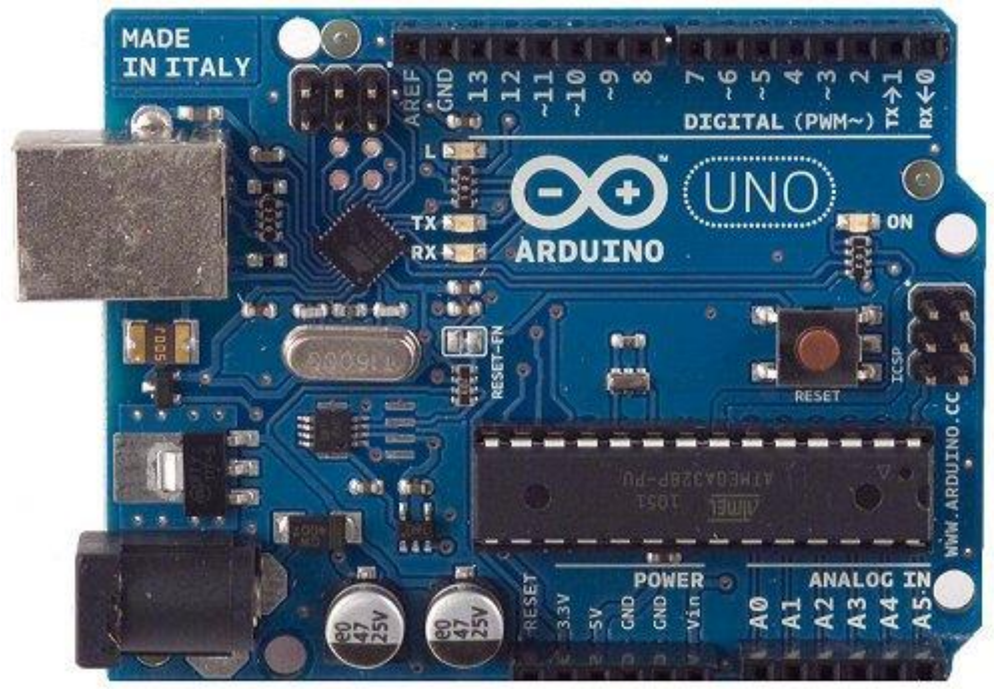

Figure 4.1 : Arduino Uno

Arduino Uno Properties

- **Power:** The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board

may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.
The power pins are as follows:

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins 12

- **Input and Output:** Each of the 14 digital pins on the Uno can be used as an input or output, using pin Mode(), digital Write( ), and digital Read( ) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 KOhms. In addition, some pins have specialized functions:
- **Serial: 0 (RX) and 1 (TX)**. Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attach Interrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11**. Provide 8-bit PWM output with the Analog Write() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not

currently included in the Arduino language.

- **LED: 13**. There is a built-in LED connected to digital pin 13. When the pin

is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024

different values). By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference () function. Additionally, some pins have specialized functionality:

- **I 2C: 4 (SDA) and 5 (SCL)**. Support I2C (TWI) communication using the

Wire library.

There are a couple of other pins on the board:

- **AREF**. Reference voltage for the analog inputs, used with

Analog Reference ().

- **Reset**. Bring this line LOW to reset the microcontroller, typically used to

add a reset button to shields which block the one on the board.

- **Communication:** The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to serial chip and

USB connection to the computer (but not for serial communication on pins 0 and 1). A Software 13 Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

- **Programming:** The Arduino Uno can be programmed with the Arduino software

(download). Select "Arduino Uno w/ ATmega328" from the Tools > Board menu
(according to the microcontroller on your board). For details, see the reference and
tutorials. The ATmega328 on the Arduino Uno comes pre burned with a
bootloader that allows you to upload new code to it without the use of an external
hardware programmer. It communicates using the original STK500 protocol
(reference, C header files). You can also bypass the bootloader and program the
microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega8U2 firmware source code is available.
The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and
then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the
DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use
the ISP header with an external programmer (overwriting the DFU bootloader).

- **Automatic Reset:** Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 Nano farad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data. The Uno contains a trace that can be cut to disable the auto reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110-ohm resistor from 5V to the reset line.

- Push button switches.
- BCD to seven segment decoder 7447.
- Seven segment display.
- Driver ULN2003a.
- Stepper motor.

4.2 The General Hardware Circuit Connection:

Explaining the connection of the circuit as simulation been made as in Figure 4.3 and the real circuit connection as in Figure 4.4

4.2.1 push buttons to microcontroller

First connection in this circuit is the four push buttons connected to the microcontroller through ports

• First floor push button is connected to output number 5 (~PD5/T1/OC0B) in the microcontroller.

• Second floor push button is connected to output number 4(PD4/T0/XCK).

• Third floor push button is connected to output number 3(~PD3/INT1/OC2B).

• Fourth floor push button is connected to output number 2 (PD2/INT0).

4.4 BCD to seven segment decoder 7447 to microcontroller The IC is connected to the Arduino through the four BCD inputs

• Output number 6 (~PD7/AIN1), and input number 7 (A) in the 7447 decoder.

• Output number 7 (PD7/AIN1), and input number 1 (B) in the 7447 decoder.

• Output number 8 (PB0/ICP1/CLKO), and input number 2 (C) in the 7447 decoder.

• Output number 9 (PB1/OC1A), and input number 6 (D) in the 7447 decoder

• Fourth floor push button is connected to output number 2 (PD2/INT0).

4.4 BCD to seven segment decoder 7447 to microcontroller The IC is connected to the Arduino through the four BCD inputs

•Output number 6 (~PD7/AIN1),and input number 7 (A) in the 7447 decoder.

• Output number 7 (PD7/AIN1), and input number 1 (B) in the 7447 decoder.

• Output number 8 (PB0/ICP1/CLKO), and input number 2 (C) in the 7447 decoder.

• Output number 9 (PB1/OC1A), and input number 6 (D) in the 7447 decoder.

4.2.2 BCD to decoder and display

In this connection the IC is connected to the seven segment display through the seven output lines, one line for each LED
segment.

4.2.3 ULN2003a Driver to the microcontroller

After the microcontroller receives the push button request, it sends control pulses to the stepper motor through the interface (driver uln2003a), which is connected to the Arduino through four ports.

• Output number 10 (~PB2/OC1B), to input number 1 (1B) in the uln2003a.

• Output number 11(~PB3/MOSI/OC2A), to input number 2 (2B).

• Output number 12 (PB4/MISO), to input number 3 (3B).

• Output number 13 (PB5/SCK), to input number 4 (4B).

4.2.4 ULN2003a Driver to the stepper motor

The ULN2003a is connected to the stepper motor through the IC`s outputs (1C, 2C, 3C, 4C) and output number 9(COM) is ground. The uln2003a sends electrical pulses to the stepper motor after it receives the control pulses from the microcontroller; the stepper motor then converts these electrical pulses into discrete mechanical movements

- Fig4.2 represents the simulation of the control system

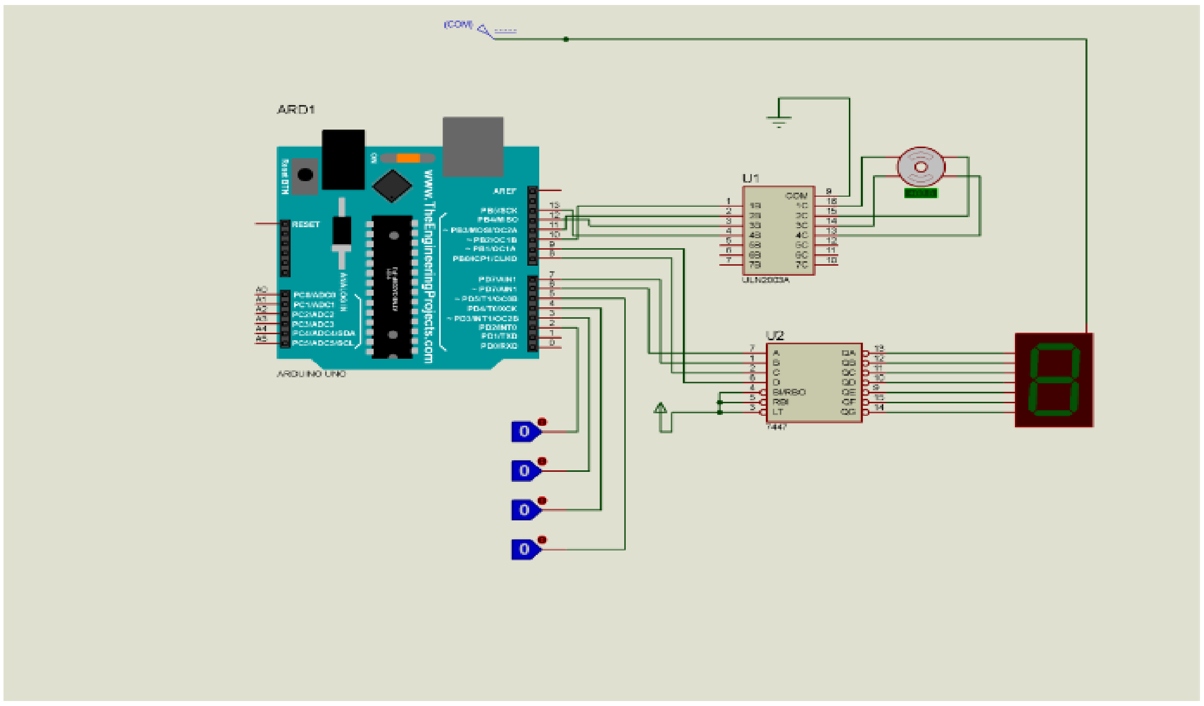

Figure 4.2 : Control Circuit Simulation

- Fig4.3 shows the control system implemented by Arduino Uno

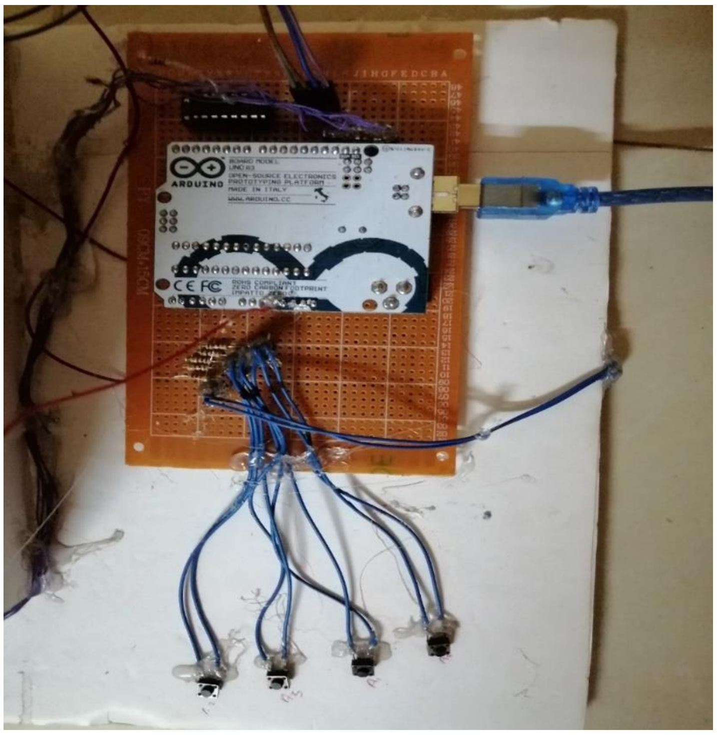

Figure 4.3 : control circuit

The model

Figure 4.4 shows a prototype design of Elevator Control System

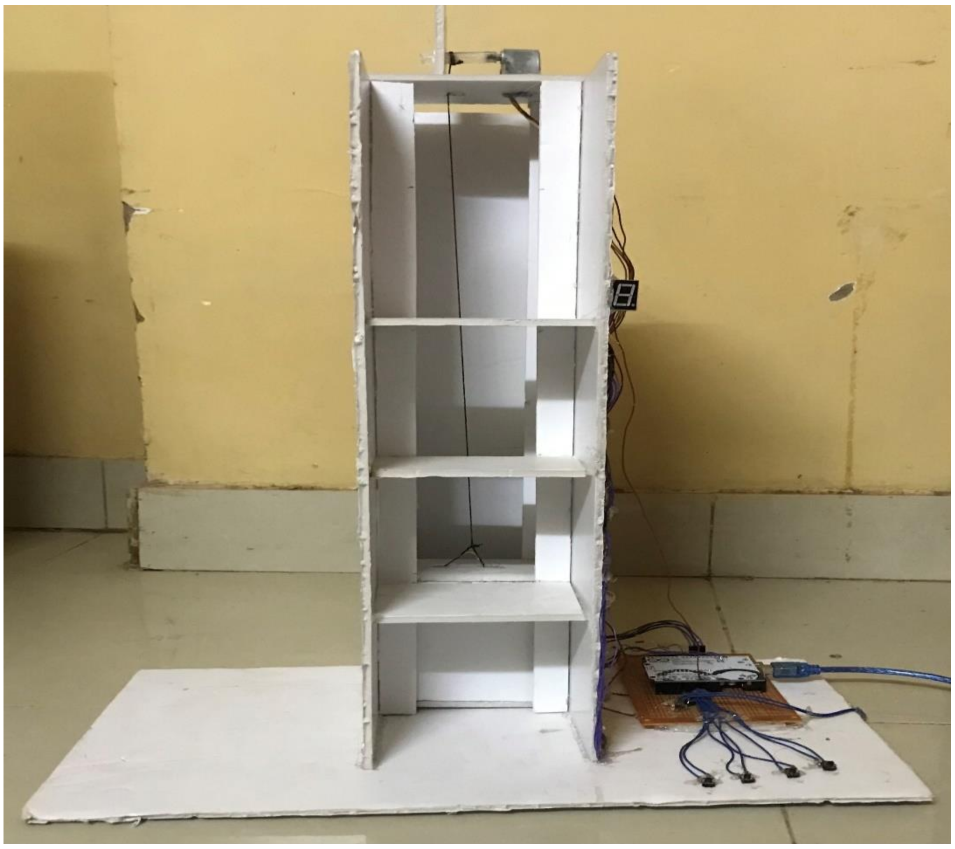

Figure 4.4 : The Model

4.3 Software implementation

The code

The code is written using C++, it implements the logic designed. The Arduino Uno receives destination requests from the push buttons to initiate the logic. The written code was loaded onto Arduino Uno controller using a USB cable, the different modules are also integrated allowing for particular response whenever the destination request commands are issued.

CHAPTER FIVE

CONCULOTION AND RECOMMENDATIONS

5.1 Conclusion

Elevator control systems were widely used in most buildings. This thesis was based on design and implementation of Arduino microcontroller based elevator control system. In this work, we have designed and implemented a prototype elevator and its control systems using Arduino microcontroller based circuit. The elevator is operated by using Step motor.

Before implementation of the system, we have simulated the control circuit. It was found that the simulation results are satisfactory and practical systems work very well.

5.2 Recommendations

In the more complex elevator where there are more conditions to test, calculate and analysis it is better to use one of the well-known control theories like the fuzzy controlling or use the computer calculation methods.

More advanced security systems can be integrated like the fingerprint, face, voice, retina or any other kind of biological security system, it could be more secure but more expensive. The requesting methods can be more various by adding SMS remote request service or wireless requesting using the internet for the VIP or special cases The doors can be able to open as an automatic door for more flexibility and more advance control, but it is more financially disturbing.

References

[1] Robert Beyer, “ Specification Series: Elevators - First Things First ”, 1994.

[2] Lava Computer MFG Inc.,” IEEE 1284: Parallel Ports”, Lava Computer MFG Inc”, 2002

[3] Sajal K. Das, “Mobile Handset Design‟, John Wiley & Sons”, 2010.

[4] Steven F Barrett, Daniel Pack, Mitchell Thornton, “Atmel AVR Microcontroller Primer: Programming and Interfacing”, 2007.

[5] Programming Arduino Next Steps: Going Further with Sketches. 2nd Ed. Simon Monk.McGraw-Hill Education.

[6] Rogers, Warren O. 1910. Power Plant Signalling System. Power and the Engineer.

[7] B.L. Theraja, A.K. Theraja ‘A Text Book of Electrical Technology’, J.Ghand & Company LTD .RAM Nagar, New Delhi ”2005.

[8] Programming Arduino Next Steps: Going Further with Sketches. 2nd Ed. Simon Monk. McGraw-Hill Education.

Appendix

Projram Code

```
int i=1; // the value represents where elevator stay ( fisrt time elevator start in
the first floor
//int btn4=2; // represents the 4 th floor
int btn3=3; // 2 nd floor
int btn2=4; // 3 th floor
int btn1=5; // 1 st floor
int motorPin1=10;
int motorPin2=11;
int motorPin3=12;
int motorPin4=13;
int buttonState=0;
int buttonState2=0;
int buttonState3=0;
int buttonState4=0;
int m1 =8; //
int m2 =9;
int m3 =6;
int m4 =7; //
int ir;
int d1;
int d2;
int d3;
int d4;
void setup() {
Serial.begin(9600);
int delayTime = 50;
// pinMode(btn4,INPUT);
pinMode(btn2,INPUT);
pinMode(btn3,INPUT);
pinMode(btn1,INPUT);
pinMode(m1,OUTPUT);
pinMode(m2,OUTPUT);
pinMode(m3,OUTPUT);
pinMode(m4,OUTPUT);
```

```
}
void loop() {
// Serial.println("You are on the i st floor");
buttonState = digitalRead(btn1);
buttonState2 = digitalRead(btn2);
buttonState3 = digitalRead(btn3);
//buttonState4 = digitalRead(btn4);
//show(numbers[i]); //write number on the 7 segment display which floor
elevator stays
/*-----------------------------------------------------------------*/
//3 th floor required codes
if(buttonState3 == HIGH){
while(i<3){
i++;
up();
Serial.print("You are on the ");
Serial.print(i);
Serial.println("floor");
// show(numbers[i]);
}
i=3;
//show(numbers[i]);
//door();
}// end of the buttonState4
//------ end of the 4th floor ------
/*-----------------------------------------------------------------*/
/*-----------------------------------------------------------------*/
//start of the 2 td floor codes ----
if(buttonState2==HIGH){
if(i>2){
while(i>2) {
i--;
down();
Serial.print("You are on the ");
Serial.print(i);
Serial.println("floor");
//show(numbers[i]);
  }
}
if(i<2){ // person is waiting on the first or second floor. Call the elevater
while(i<2){
i++;
up();
Serial.print("You are on the ");
Serial.print(i);
Serial.println("floor");
// show(numbers[i]);
}
}
i=2; //assign the elevator value to the three
// show(numbers[i]);
delay(1000);
// door();
} // end of the buttonState3
//end of the 3 td floor codes ----
/*-----------------------------------------------------------------*/
// start of the 1 st floor codes ------
if(buttonState == HIGH){
if(i>1){
while(i>1){
i--;
down();
Serial.print("You are on the ");
Serial.print(i);
Serial.println("floor");
// show(numbers[i]);
}
}
i=1;
// show(numbers[i]);
delay(1000);
// door();
}// end of the buttonState1
//--- end of the first floor codes-----
/*-----------------------------------------------------------------*/
}
// ------- up function which is for to go up elevator
void up() { // each floor distance where it works to up
```

```
for(int b=0; b<2000; b++){
// show(numbers[i]);
digitalWrite(motorPin1, LOW);
digitalWrite(motorPin2, LOW);
digitalWrite(motorPin3, LOW);
digitalWrite(motorPin4, HIGH);
delay(3);
digitalWrite(motorPin1, LOW);
digitalWrite(motorPin2, LOW);
digitalWrite(motorPin3, HIGH);
digitalWrite(motorPin4, LOW);
delay(3);
digitalWrite(motorPin1, LOW);
digitalWrite(motorPin2, HIGH);
digitalWrite(motorPin3, LOW);
digitalWrite(motorPin4, LOW);
delay(3);
digitalWrite(motorPin1, HIGH);
digitalWrite(motorPin2, LOW);
digitalWrite(motorPin3, LOW);
digitalWrite(motorPin4, LOW);
delay(3);
}
}
void down() { // each floor distance where it works to down
for(int a=0;a<2000;a++){
// show(numbers[i]);
digitalWrite(motorPin1, HIGH);
digitalWrite(motorPin2, LOW);
digitalWrite(motorPin3, LOW);
digitalWrite(motorPin4, LOW);
delay(3);
digitalWrite(motorPin1, LOW);
digitalWrite(motorPin2, HIGH);
digitalWrite(motorPin3, LOW);
digitalWrite(motorPin4, LOW);
delay(3);
digitalWrite(motorPin1, LOW);
digitalWrite(motorPin2, LOW);
digitalWrite(motorPin3, HIGH);
digitalWrite(motorPin4, LOW);
delay(3);
digitalWrite(motorPin1, LOW);
digitalWrite(motorPin2, LOW);
digitalWrite(motorPin3, LOW);
digitalWrite(motorPin4, HIGH);
delay(3);
}
}
```