Cairo University
Faculty of Engineering
Mech. Power Department

ACC Virtual Labs
Automatic Control Circuits & Virtual Labs for Mechanical Power Systems

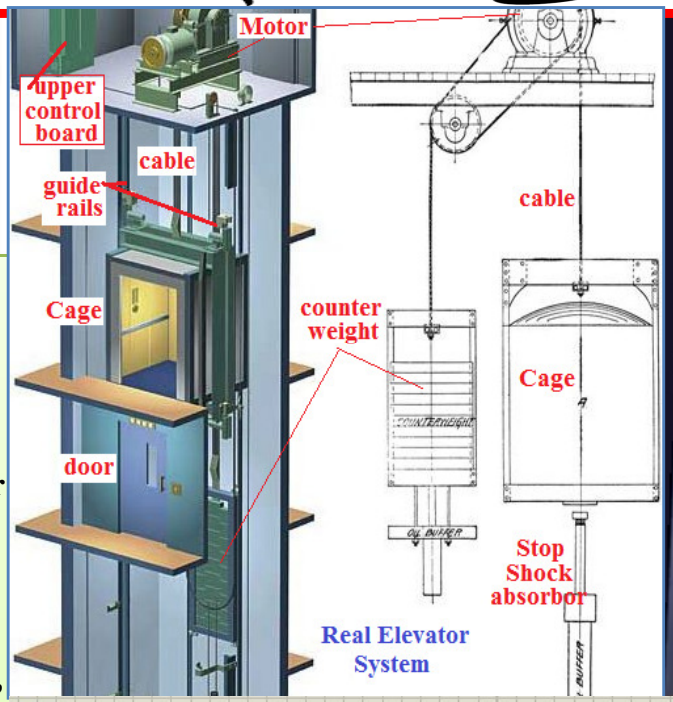
دبلوم تطبيقات التحكم الأوتوماتيكى فى نظم القوى الميكانيكية

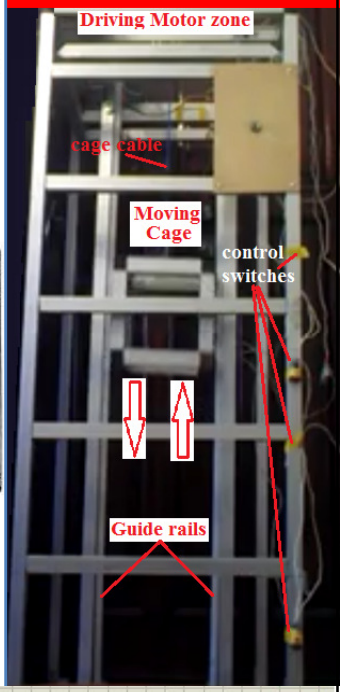
MEP 599 Diploma Design Project-Summer 2017/2018

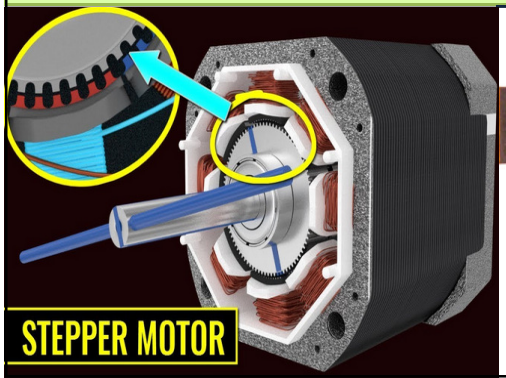
Elevator Simulator System [ESS]

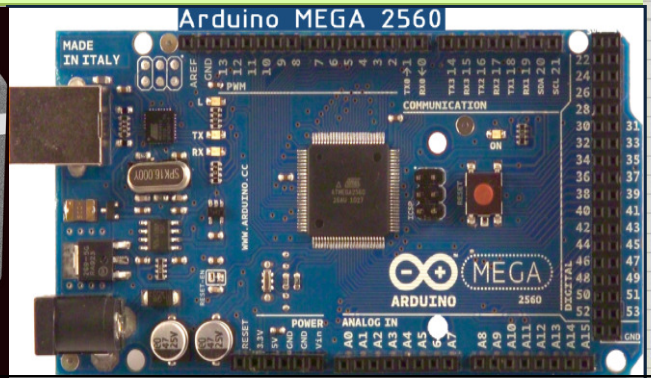
By Eng. Malek Mahmoud Al-Asem Ahmed Fahmy
Supervised by Assoc. Prof. Mohsen Soliman, ACC Manager
Mechanical Power Engineering Department

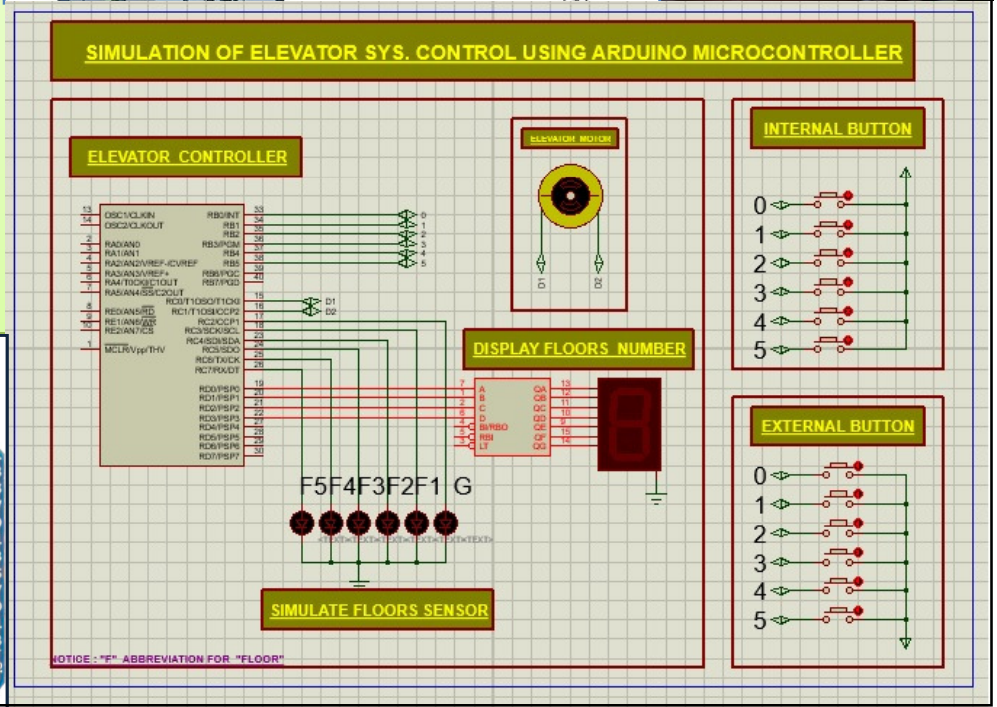
Abstract: **The main objective is to design& implement control logic for a 4 level elevator using a stepper motor. We had to study& understand types, operation principles, applications of elevators, stepper motors &PLC controller. To demonstrate our design on real model, we had to integrate, build & test experimental set shown on next fig. Using PLC leads to a cost effective & efficient system that assuressafety&simplicitywithoutcompromising onquality. In addition to stepper motor, we used Arduino mega2560 micro-controller/PLC board(shown below) which has 54digital input/output pins (of which 14 can be used as PWM outputs to drive the stepper motor), 16 analog inputs, 4 UARTs(hardware serial ports), 16MHz crystal oscillator, USB connection, power jack, ICSP header, & a reset button. Other electric parts to be used are: DC-power supply, NO-push buttons, 7Segment display, limit switches, LEDs, wires&steel cable to carry the carriage. After specifying all input&output signals needed to design control logic of the system, Flow chart/SFC (Sequential Function Chart) was done. Using Arduino Software, SFC was converted to program/code which was downloaded onto the Control Board. To test the code, a simulation of full elevator control system was made to run on PC (shown in next fig.). Both simulation& experimental model were tested to prove that they satisfy all design requirements & I/O commands as they were specified at start in the Flow Chart/SFC of the control logic.**





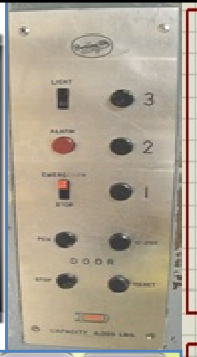


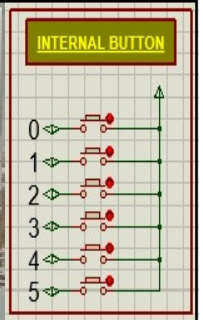




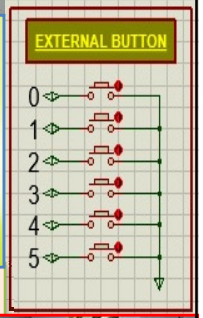
Introduction: **Elevators are normally powered by AC motors driving traction cables/counter-weight systems like hoist or by pumping hydraulic oil to raise a cylindrical piston like a jack. We used stepper motor (rather than AC motor) to reduce tasks involved with controlling speed & direction of AC motor. Stepper motor has advantage that position information is obtained from count of sent pulses thus eliminating need for expensive position sensors & feedback controls. Due to good control over speed &position, Stepper motors are used today in elevators since they have a robust design, no brushes & a good speed control.**

Project scope: **This project aims at building control logic that takes commands from users & acts on its own to enable operation of the elevator. It uses stepper motor and Arduino mega 2560PLC controller to implement an elevator system.**





CAR OPERATING PANEL AND LANDING DOOR PANEL INDICATORS



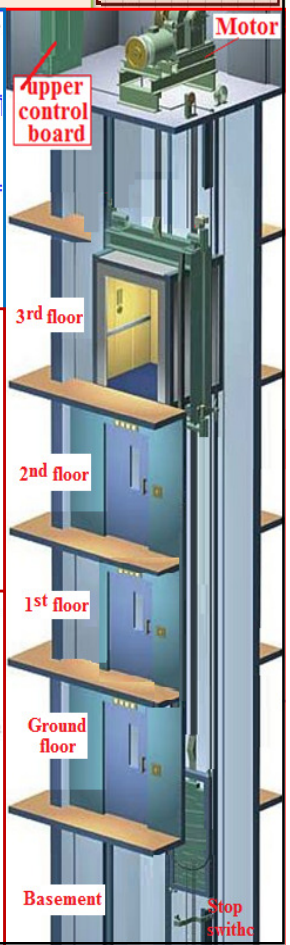
Stepper Motor & Elevator Controller: **It is responsible for coordinating all aspects of the elevator services such as travel, speed, accelerating, decelerating, door opening speed & delay, leveling & hall lantern signals. It accepts inputs (e.g. button signals) & produces outputs (e.g.elevator car moving &door opening).Elevator control system has to do (the minimum tasks): -To bring the lift car to the correct floor. -To minimize travel time. -To maximize passenger comfort bysmooth ride. -Toaccelerate, deceleratеat safe speed. Microcontroller is used to send pulses to stepper motor drive. It controls both speed & position of motor. Speed of motor depends on frequency of drive input pulses which is controllable with microcontroller. Motor rotation is proportional to # of output pulses. Microcontroller was preferred in project due to ease to connect with stepper motor to**

Control problems in simple elevator

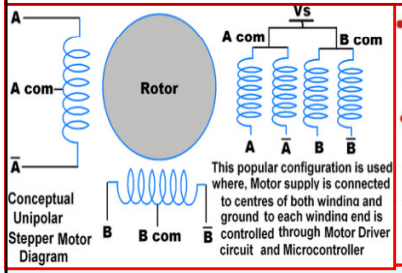
Logical:
1. The elevator must move towards a floor when a button is pushed.
2. The elevator must open a door when it is at a floor.
3. It must have the door closed before it moves. etc.

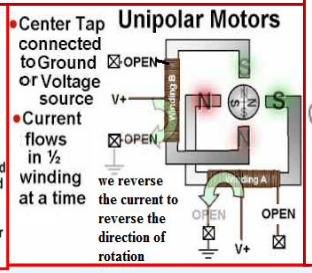
Linear:
1. If the desired position changes to a new value, accelerate quickly towards the new position.
2. As the elevator approaches the correct position, slow down.

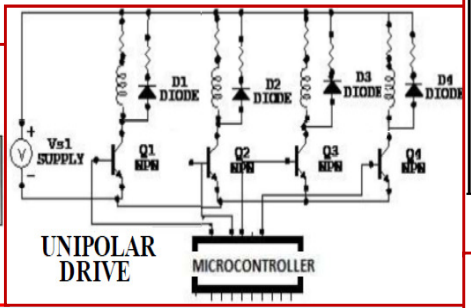
Non-linear:
1. Accelerate slowly to start.
2. Decelerate as you approach the final position.
3. Allow faster motion while moving.
4. Compensate for cable stretch, and changing spring constant.



UNIPOLAR DRIVE current flow is limited to one direction.The switch set is simpleand inexpensive. However drawbacks are limited capability to energize all windings at same time thus low efficiency. Here the torque is proportional to the square of the current, direction of the current in the phase windings being unimportant. Therefore, four windings A, B, C and D are excited in appropriate manner. The phasewindings have large inductance, therefore in series in each phase winding a forcing resistance is inserted so as to reduce circuit time constant. A resistance is also placed in series with freewheeling diode in order to dissipate the energy stored in the phase winding inductance.

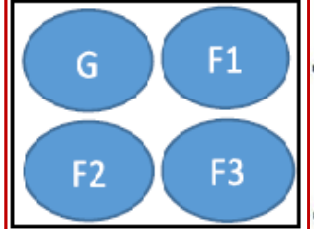




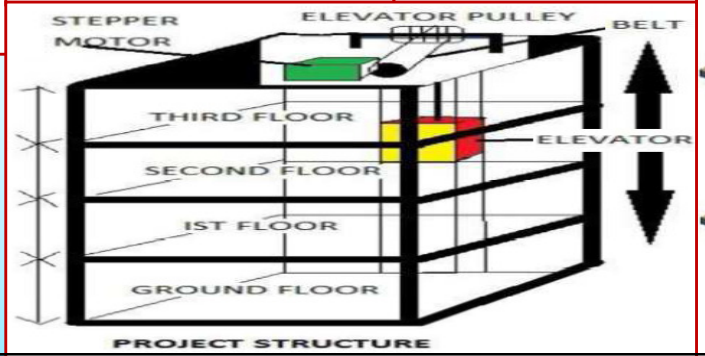


Third floor(D)
Second floor(C)
First floor(B)
Ground floor(A)

Buttons on outer doors for making requests &directions.



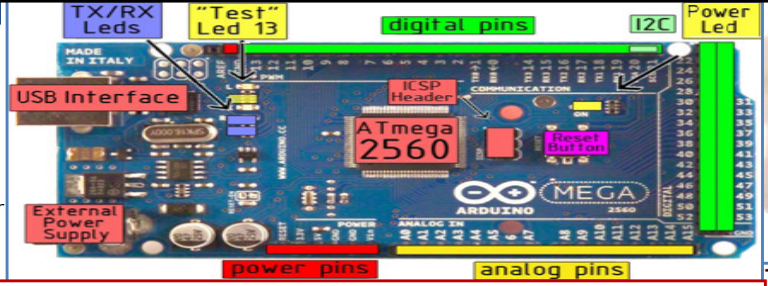
Internal Bottons



Control Overview: **The project is divided into 4 floors of equal distance: ground floor, 1st floor, 2nd floor & 3rd floor. Stepper motor is connected to elevator lifting pulley with belt. The pulley takes 2 full rotations to lift the elevator to cover distance from one floor to another. The 7 segment displays following information; floor position of carriage & whether its ascending or descending. External & Internal pushbuttons are used to select the desired location for elevator. After reaching at destination floor after some time delay, a green LED glows to represent opening of elevator door. Afterwards, a green LED is turned off and a red LED glows to represent closing of carriage door.**

Arduino MEGA 2560 Technical Specification

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz



Memory ATmega2560 has 256 KB of flash memory for storing code (8 KB is used for bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with EEPROM library).

Power Arduino Mega2560 can be powered via the USB connection or with 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 a 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 Mega2560 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.

Input and Output Each of 54 digital pins can be used as input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** 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 attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **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.
- **I²C: 20 (SDA) and 21 (SCL).** Support I²C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I²C pins on the Duemilanove.

Communication Arduino Mega2560 has # of facilities for communicating with computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Automatic (Software) Reset Rather then physical press of reset button before upload, Arduino Mega2560 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 ATmega2560 via a 100 nanofarad 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 Mega2560 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 Mega2560. 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 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.

Mega2560 has 16 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 analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **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.

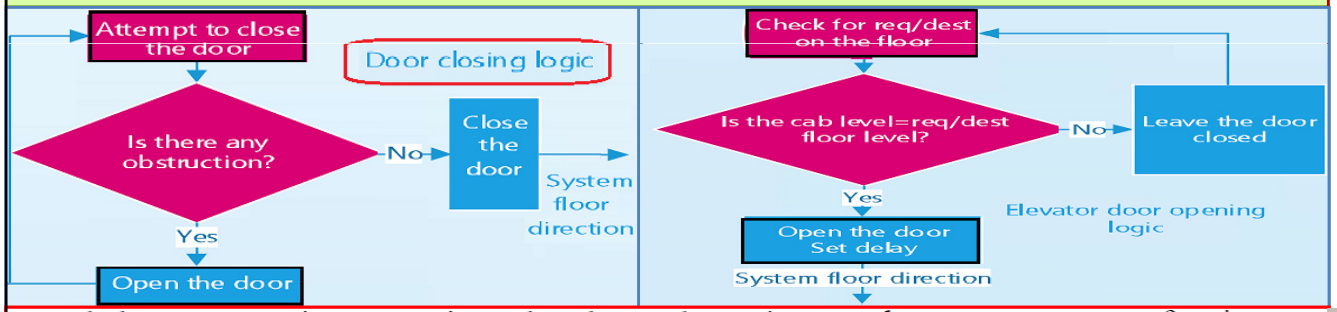
USB Overcurrent Protection Arduino Mega has resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

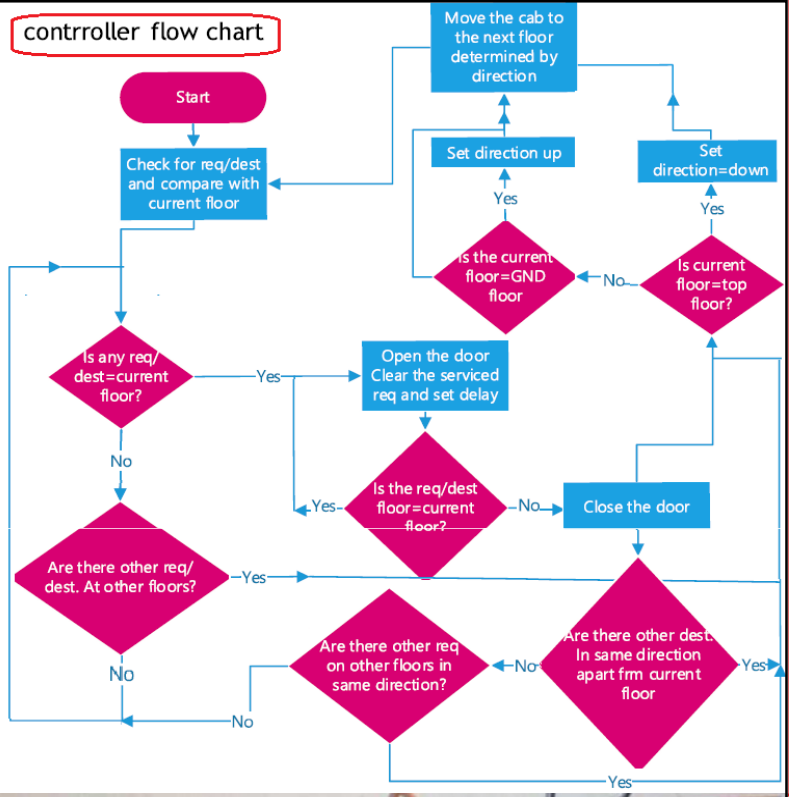
Physical Characteristics and Shield Compatibility The maximum length and width of Mega PCB are 4 and 2.1 inches respectively, with USB connector and power jack extending beyond former dimension. Three screw holes allow board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

Mega is designed to be compatible with most shields designed for Diecimila or Duemilanove. Digital pins 0 to 13 (and adjacent AREF and GND pins), analog inputs 0 to 5, power header, and ICSP header are all in equivalent locations. Further main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila.

Please note that I²C is not located on same pins on Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).

Design of Logic Control System: **A number of actions are performed to ensure proper system operation (as shown on next Flow Chart). Some points to be noted about the flowchart /SFC. 1-Elevator starts operating when power is turned on. 2-If it is at no use, elevator rest at last floor serviced (last destination floor). 3-Elevator keeps checking for requests during waiting time to ensure fast action of any request on any particular floor. 4-Carraige moves on each level & checks for requests or destinations in same direction i.e. UPWARDS/DOWNWRDS. 5-Elevator keeps checking both 4th& ground floor to ensure reverse of directions on those particular floors since they represent highest& lowest floors according to design scope (this to ensuressafety). 6-Closing&opening of doors is automated using delays& in this case LEDs (green&red)on simulation & indicate same on the model. 7-Request allows someone to get into carraige& indicates direction while destination request allows user to select particular floor of his interest. 8-Check on next floor happens when elevator is still on another floor.**



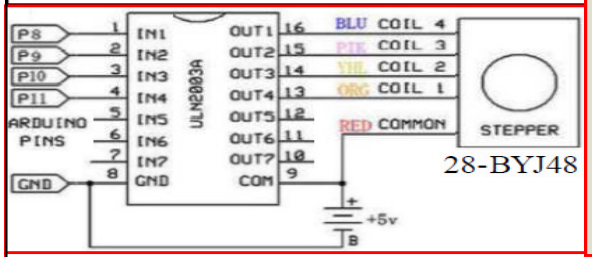


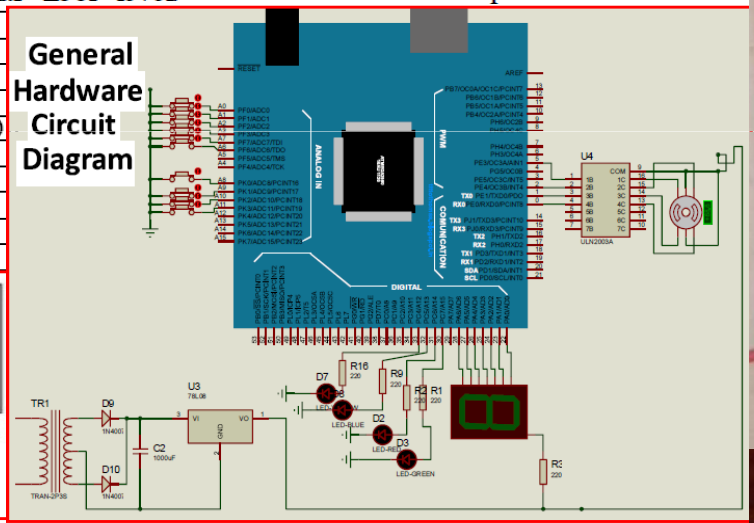
In real elevators a motion sensor is used to detect obstructions on door way to ensure safety i.e. door closes only when there is no obstacle on the way. However, in this project, door opening and closing mechanism will be achieved by led with a set delay.

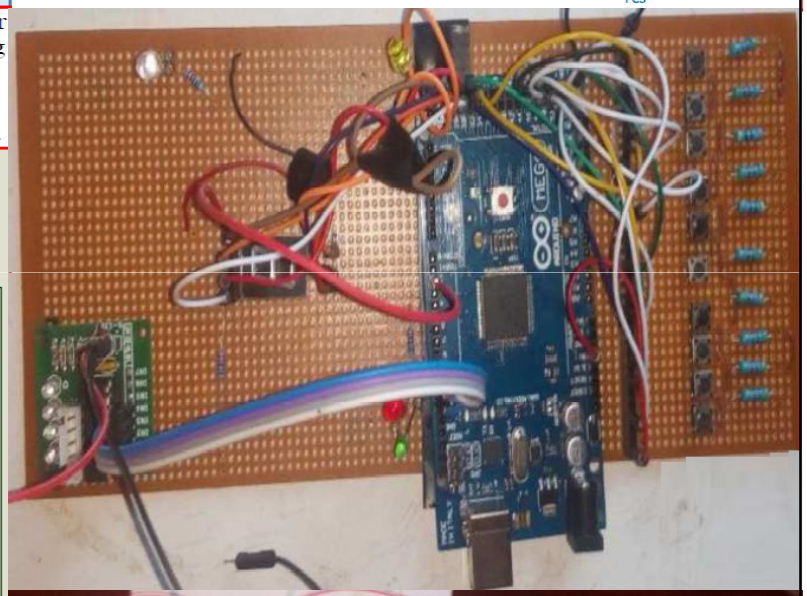
For elevator door to open a number of conditions need to be attained:

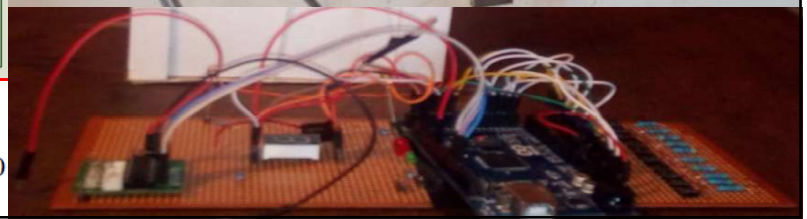
i. Cab must be same level with the particular floor level. ii. There must be a request for this action.

COMPONENTS USED
Arduino Mega 2560R3
7 segment floor display indicator (1 digit)
UNL2003 Stepper Motor Driver
4 ordinary LEDs(Green, Red, white, yellow)
Buzzer
Push-buttons
LM7805 and LM7809 voltage regulators
Stepper Motor 28BYJ48
resistors









Rated voltage : 5V DC Number of Phase 4, Speed Variation Ratio 1/64, Stride Angle 5.625° /64. Frequency 100Hz, DC resistance 50Ω±7%(25°C), Idle In-traction Frequency > 600Hz, Idle Out-traction Frequency > 1000Hz, In-traction Torque >34. 3mN.m(120Hz), Self-positioning Torque >34.3mN, Friction torque 600-1200 gf.cm, Pull in torque 300 gf.cm, Insulation grade A.

Code used for Implementation

```
#include <Stepper.h>
//Global variables
int CF=0; // Current floor
int DF=0; //destination floor
int DF0=0; // Destination floor boolean
int DF1=0; // Destination floor boolean
int DF2=0; // Destination floor boolean
int DF3=0; // Destination floor boolean
int Direction=1;
int move_to_request=1;
int time_count=25;//5000/200 (5000msec delay with each
  key press cycle taking 200ms : 5000/200: 25 keypress
      cycle wait before door closes)
// Motor
const int stepsPerRevolution = 200; //
int intersteps = 2222;
Stepper myStepper(stepsPerRevolution, 8,10,9,11);
//Requests
int GFU=0; //boolean
int FFU=0; //boolean
int FFD=0; //bolean
int SFU=0;//bolean
int SFD=0;//bolean
int TFD=0;//bolean
// Declare pins
// Seven segment pins
int pina = 22;
int pinb = 23;
int pinc = 24;
int pind = 25;
int pine = 26;
int pinf = 27;
int ping = 28;

// Buttons pins
int pinGF = 53;//connected to ground
int pinFF = 52;
int pinSF = 51;
int pinTF = 50;
int pinGFU = 49;
int pinFFU = 48;
int pinFFD = 47;
int pinSFU = 46;
int pinSFD = 45;
int pinTFD = 44;

// LED pins
int pinLedOpen = 30;
int pinLedClose = 31;
int pinLedUp = 32;
int pinLedDown = 33;

// buzzer pins

int pinBuzzer=2;

void initButtons()
{
 pinMode(pinGF, INPUT);
 pinMode(pinFF, INPUT);
 pinMode(pinSF, INPUT);
 pinMode(pinTF, INPUT);
 pinMode(pinGFU, INPUT);
 pinMode(pinFFU, INPUT);
 pinMode(pinFFD, INPUT);
 pinMode(pinSFU, INPUT);
 pinMode(pinSFD, INPUT);
 pinMode(pinTFD, INPUT);
}

int keyScan()
{
 delay(200);
 if (digitalRead(pinGF) == HIGH)
 {
  return 0;
 }

 if (digitalRead(pinFF) == HIGH)
 {
  return 1;
 }
 if (digitalRead(pinSF) == HIGH)
 {
  return 2;
 }

 if (digitalRead(pinTF) == HIGH)
 {
  return 3;
 }

 if (digitalRead(pinGFU) == HIGH)
 {
  return 4;
 }
```

```
  pinMode(pina, OUTPUT);
  pinMode(pinb, OUTPUT);
  pinMode(pinc, OUTPUT);
  pinMode(pind, OUTPUT);
  pinMode(pine, OUTPUT);
  pinMode(pinf, OUTPUT);
  pinMode(ping, OUTPUT);

  //IndicatorLEDs
  pinMode(pinLedOpen, OUTPUT);
  pinMode(pinLedClose, OUTPUT);
  pinMode(pinLedDown, OUTPUT);
  pinMode(pinLedUp, OUTPUT);

  //Buzzer sound indicator
  pinMode(pinBuzzer,OUTPUT);

  //initialize the pins to HIGH<-- All of the off

  digitalWrite(pina, HIGH);
  digitalWrite(pinb, HIGH);
  digitalWrite(pinc, HIGH);
  digitalWrite(pind, HIGH);
  digitalWrite(pine, HIGH);
  digitalWrite(pinf, HIGH);
  digitalWrite(ping, HIGH);

  // Initialize LEDs to LOW
  digitalWrite(pinLedOpen, LOW);
  digitalWrite(pinLedClose, LOW);
  digitalWrite(pinLedUp, LOW);
```

ADXL3xx | Arduino 1.8.7

File Edit Sketch Tools Help

ADXL3xx

```
/*
  ADXL3xx

  Reads an Analog Devices ADXL3xx accelerometer and communicates the
  acceleration to the computer. The pins used are designed to be easily
  compatible with the breakout boards from SparkFun, available from:
  http://www.sparkfun.com/commerce/categories.php?c=80

  The circuit:
  - analog 0: accelerometer self test
  - analog 1: z-axis
  - analog 2: y-axis
  - analog 3: x-axis
  - analog 4: ground
  - analog 5: vcc

  created 2 Jul 2008
  by David A. Mellis
  modified 30 Aug 2011
  by Tom Igoe

  This example code is in the public domain.
  http://www.arduino.cc/en/Tutorial/ADXL3xx
*/
// these constants describe the pins. They won't change:
```

ADXL3xx | Arduino 1.8.7

File Edit Sketch Tools Help

ADXL3xx

```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/ADXL3xx
*/

// these constants describe the pins. They won't change:
const int groundpin = 18;             // analog input pin 4 -- ground
const int powerpin = 19;              // analog input pin 5 -- voltage
const int xpin = A3;                  // x-axis of the accelerometer
const int ypin = A2;                  // y-axis
const int zpin = A1;                  // z-axis (only on 3-axis models)

void setup() {
  // initialize the serial communications:
  Serial.begin(9600);

  // Provide ground and power by using the analog inputs as normal digital pins.
  // This makes it possible to directly connect the breakout board to the
  // Arduino. If you use the normal 5V and GND pins on the Arduino,
  // you can remove these lines.
  pinMode(groundpin, OUTPUT);
  pinMode(powerpin, OUTPUT);
  digitalWrite(groundpin, LOW);
  digitalWrite(powerpin, HIGH);
}

void loop() {
```