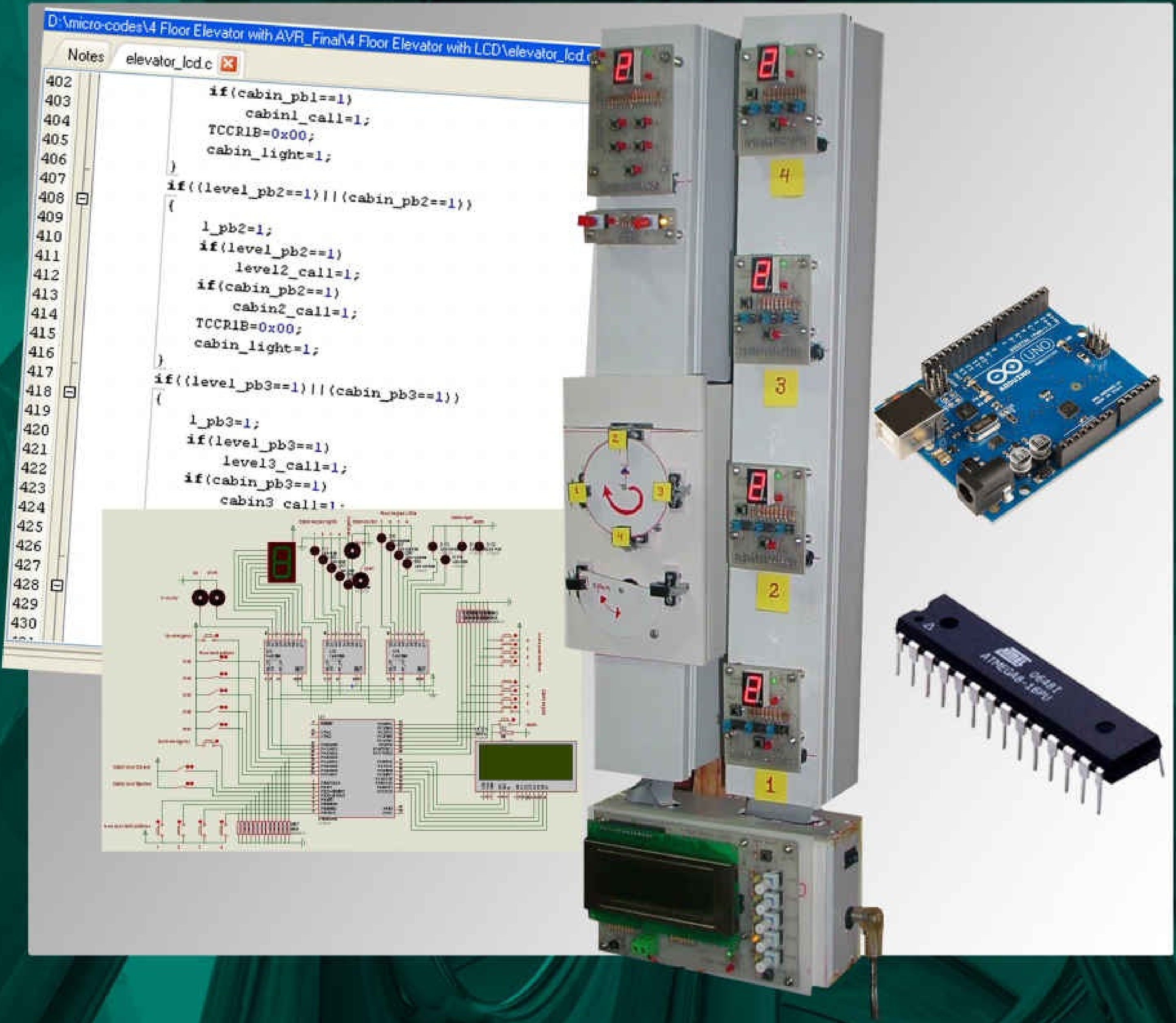
Simple Arduino based 4 Floor Elevator Project


D:\micro-codes\4 Floor Elevator with AVR_Final\4 Floor Elevator with LCD\elevator_lcd.c
Notes
elevator_lcd.c
4
3
2
1

```
            if(cabin_pb1==1)
                cabin1_call=1;
            TCCR1B=0x00;
            cabin_light=1;
        }
        if((level_pb2==1)||(cabin_pb2==1))
        {
            l_pb2=1;
            if(level_pb2==1)
                level2_call=1;
            if(cabin_pb2==1)
                cabin2_call=1;
            TCCR1B=0x00;
            cabin_light=1;
        }
        if((level_pb3==1)||(cabin_pb3==1))
        {
            l_pb3=1;
            if(level_pb3==1)
                level3_call=1;
            if(cabin_pb3==1)
                cabin3_call=1;
```

Circuits and programs for an AVR Microcontroller and Arduino board

Seyedreza Fattahzadeh

Simple Arduino based 4 Floor Elevator Project

Seyedreza Fattahzadeh

NOTICE TO THE READER

Please be informed that:

This book contains many images. Since eReaders don’t always display images well, upon request, I will provide you with the PDF file which contains the complete e-Book, including the images, if you need an easier-to-read format. To receive a PDF version of this e-book or any other software which is mentioned in the content of this text, you can email me the proof of your purchase of the kindle version of the book from Amazon.com to srfattah@yahoo.com. Upon receipt of that proof, a PDF version will be sent to your email address.

Table of Contents

Preface

How to use this book

Object and scope of the project

Files developed for this project

Chapter 1

Step 1: Generating schematic diagram of the project

Step 2: Generating the main control code

Step 3: Getting to manufacturing the PCBs of the prototype model

Hardware aspect of the project

Functional explanation of PCB #1 and PCB # 4

Functional explanation of PCB #5: Car Operating Panel

Functional explanation of part # 6: hoist and car door fixture

Simulating the hoist lift and car door motors

Functional explanation of PCB # 7: Main controller PCB

Functional explanation of PCB # 8: DC motor driver

Step 4: Building the PCBs of the prototype model lift system

Step 5: Uploading the control program into the ATmega16 IC

Chapter 2

Design and implementation of the 4 Floor Control System with Arduino

Step 1: Generating schematic diagram of the project

Step 2: Generating the main Control Code for the System

Step 3: Getting to manufacturing the PCBs of the prototype model

Step 4: Building the PCBs of the prototype model lift system

Step 5: Up loading the control program into the Arduino board

Appendix A: Description of ULN2803 Chip

Overview of the Arduino Mega 2560 board

Question and Answers related to the project

Conclusion

My e-book product list

Preface

In this project, we will learn the circuit design for a 4 Floor Elevator Control on how to develop a four stories passenger lift or cargo lift using an ATmega16 Microcontroller. This text consists of two chapters.

Chapter 1 is devoted to the design and implementation of a simple 4 Floor Elevator Control system using an AVR Microcontroller (e.g. ATmega16).

Chapter 2 is devoted to redesigning the same lift system using the Arduino Mega 2560 board, which is a microcontroller board based on the ATmega2560.

This project is about the model of a simple lift or elevator for a building, which is assumed to have four floors. Each floor has its own illuminated push button to call the elevator to go to the desired floor. The elevator door is emulated with the use of a small 5 V DC motor that opens, stays open for 5 seconds, and then closes the car door to be ready to move to the floor that is called. Also, a 5 V DC motor is used to emulate a hoist motor, which rotates a small disk clockwise or anticlockwise over four slotted optocouplers to generate stop signals. I used these optocoupler sensors to help the system detect if the elevator is at the floor, in a safe position to open. Next, the lift will go back to its standby status for further transportation.

The code generated for this small 4 Floor lift system has been adapted to control the same system for an Arduino board. This text provides a reader with steps he/she needs to take, to come up with a hardware and software to implement the lift system.

The reason I selected to author this text is that I already have written a few texts about the design and implementation of 4 and 8 Floor Elevator controls using a PLC.

I thought it was not a bad idea this time to show how the same design guides can be used to implement the same system, but this time with a Microcontroller instead. I love Elevator systems; they do a great job wherever they are installed. When a programmer can generate control software for a simple Elevator system, that will mean that he knows 'what he/she is talking about' when it comes to programming for any kind of CPU.

The final aspect of this experiment is to organize an intelligent main control code in C language, to control the whole system hardware. Hence, the purpose behind developing a manual like this is to make the learning about microcontroller\or a typical a Microcontroller based board more enjoyable!

How to use this Book

This book explains the generating of code for an ATmega8 Microcontroller chip and The Arduino Mega 2560 board to control a 4 Floor Elevator System in general. This book has been prepared for those who are already familiar with basic instructions related to any brand of Microcontroller, and may have already developed some Microcontroller based programs for different purposes. Therefore, this text is prepared for those who want to challenge their programming knowledge to generate even more complex programs with additional features

The project specification will be presented to you. It contains all goals, functionality, and details required for the development of this controller system to fulfill the project's typical minimum requirements. Study the project requirements presented in this text carefully. Your job is to generate the main control program to satisfy the project's requirements.

Take as much time as you think it is needed. You can use a pen and paper and any typical simulator software to develop the control program. When you have completed your project, review and compare it with the one presented in this text. If your solution works properly, then you can celebrate your accomplishment and perhaps go on to learn something new to improve your level of capability further, when it comes to Microcontroller programming. If the solution you come up with has some problem(s), and somehow it is not functioning as it is supposed to, then you can compare your solution against mine. In this case, you know that you did your best, and tried it on your own and then you can use my solution to find out what is wrong with yours. When you find it out what went wrong, then you can resolve the issue to make it function correctly. That is the whole purpose of this text; to give you enough motivation and make you enthusiastic to complete a relatively complex Microcontroller based program.

In Chapter 1, the main control program solution is prepared in C language using CodevisionAVR ® software. In addition, the schematic of all the hardware used in project is provided.

In Chapter 2, the same main control program which was prepared for the AVR Microcontroller, is redone for an Arduino based platform

The amount of effort you want to place on doing this project is completely up to you and your interest. If you wish, you can build your own controller hardware similar to the one presented in this text or even tailor it to the one with more features as you prefer. In either case, just by simulating the presented program and checking its performance, you can improve your programming knowledge and ultimately, from my side, this is the overall purpose behind developing the text.

Objective and scope of the project

In this project, lift control system is going to be produced by using an ATmega16 Microcontroller and the Arduino Mega 2560 board. Thus, the main objectives for this project is to design and construct a Microcontroller based lift control system. There are some scopes, which are needed to achieve the objective of this project:

1. To design a lift control system by using 1- ATmega16 microcontroller.
2. To design a control program for the overall system function based on the predefined basic lift system algorithm.
3. To integrate the hardware and software in order to simulate the function of a basic elevator system.
4. To build a lift prototype model to simulate the actual system with 4 floors, to show the concept with multiple floors.

This book was prepared for those who are familiar with programming a typical Microcontroller, and who want to challenge their knowledge by writing a more complex control program. By reading this book, you will be able to understand better on how to develop a control program for a four floor elevator system.

The schematics diagrams of all other hardware used in this project are given for this model- keyboard pad and displays, DC motor driver circuit, etc. If you wish, you can even build your own elevator hardware similar to mine, or even tailor it to the one with more features you like.

Programming of Microcontrollers is simple to learn. It is not much complicated and 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. Thus we can virtually see the working of our project or program. Later in the project, we will see how to use **Proteus** software to simulate 4 floor lift system, before the construction of the project.

Files developed for this project

Following table identifies the files developed for this project, as explained in this text. These files are not included in the text. To get these files emailed to your address, please contact me as you are directed in the **'Please be informed'** section of this text'.

Files related to Chapter 1
Simulation Proteus based file: elevator2.DSN
Codvision based generated file: elevator_lcd.prj
PCB layout files: (1) level1-4.PcbDoc (2) cabin.PcbDoc (car internal keypad) (3) main.PcbDoc (main controller layout) (4) motor.PcbDoc (2×DC motor drivers board)
Files related to Chapter 2
_4_floor_elevator.ino (Main controller file)
Elevator2.DSN
A_elevator.PcbDoc (Main controller board layout)

Chapter 1

Step 1:

Generating schematic diagram of the project

After determining what the project's scope is, the next step is to come up with Schematic diagram of the project. After all, this comprises the hardware and software parts of the project, which are the ones supposed to function in a way to carry out the main task of the project.

Let's draw schematic diagram of the project in a way that we think it can carry out the task of controlling all push buttons, 7-segment displays, 2 geared DC motors, bunch of LEDs, etc.

We might be able to come up with a circuit diagram as shown in figure 1-1 and try to figure out if that is capable of controlling different electronics of our project nicely. No problem to test the circuit and code at this stage. Our big brothers already have developed two great software- **CodevisionAVR** ® and **Proteus** ® that can help us to generate code and simulate the circuit respectively. So, at step 3, we may go ahead and start generating code after placing all components on the edit screen of the great **Proteus** software.

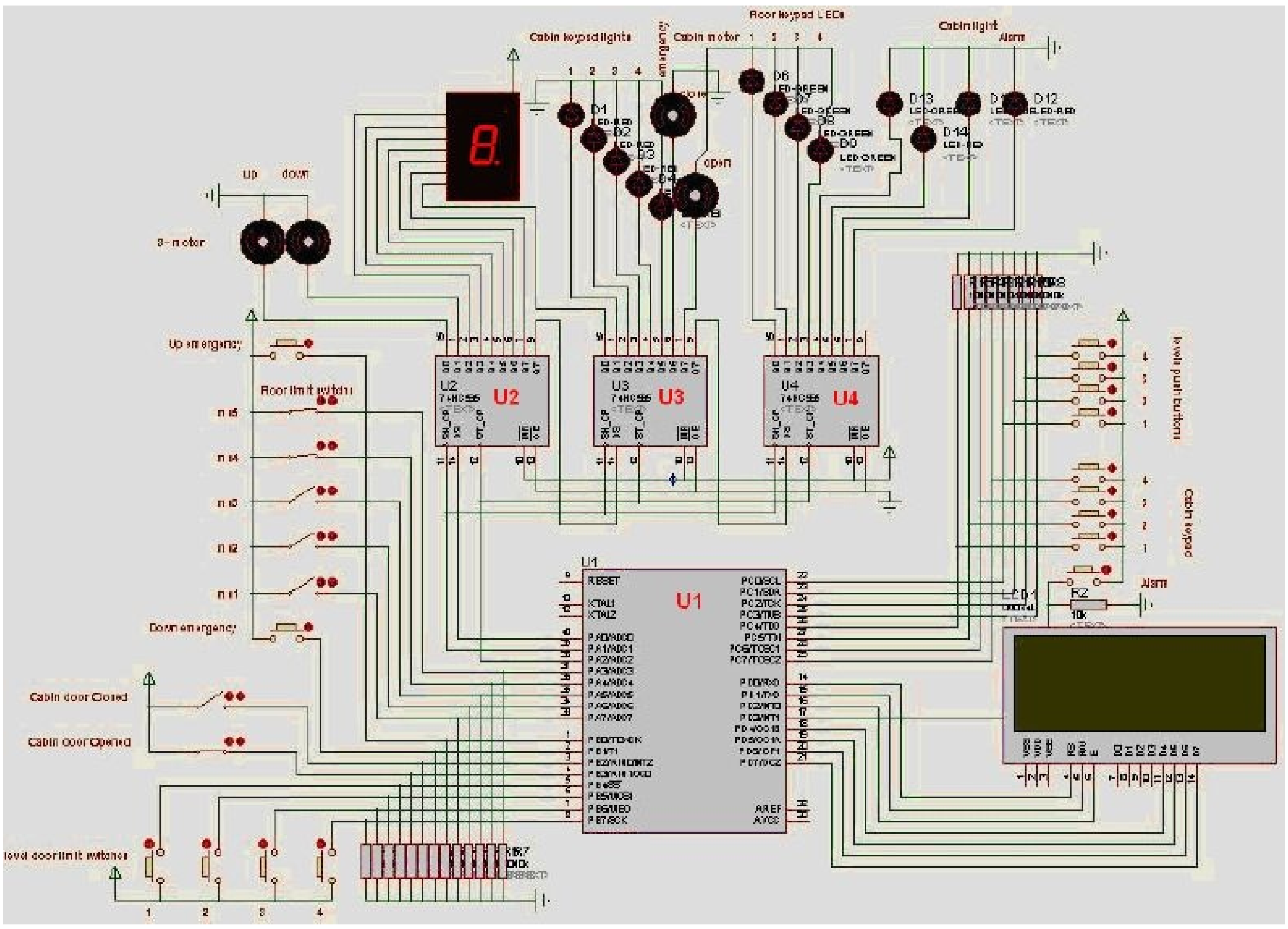


Figure 1-1 displays Proteus generated schematic diagram of the project.

Step 2:
Generating the main Control Code

Now that I managed to come up with the schematic diagram of my project, I do need to generate the control code to simulate the circuit shown in Figure 1-1 to ensure the components shown in Figure 1-1 are functioning properly. I go ahead and use my CodevisionAVR software to generate the main code and save it in a Hex format. In the next step, I will need to go back to Proteus and call the Hex file that I already have created, and start simulating the functioning of the schematic diagram, which is shown in Figure 1-1.

Program Explanations

In the next table, we explain the symbols and functions that are used in the program. You can see the full program in attendant CD.

Table 1-1: AVR Program

Initial library	

#include <mega16.h>
#include <delay.h>
// Alphanumeric LCD Module functions
#include <alcd.h>
	Defining needed library.

Define Symbols for AVR outputs and inputs	
/************ out ****************/ #define ds PORTA.0 #define sh_cp PORTA.1 #define st_cp PORTA.2	Defining symbols for three output pins that are connected to shift registers.
/************ in ****************/ #define level_ms5 PINA.3 #define level_ms4 PINA.4 #define level_ms3 PINA.5 #define level_ms2 PINA.6	Defining five limit switches (ms1 to ms5) which are connected to these pins (PINA.3 to PINA. 7)

#define level_ms1 PINA.7	
#define up_emergency PINB.0 #define down_emergency PINB.1	Defining the two micro switches at the top and bottom of the system
#define closed_door_ms PINB.2 #define opened_door_ms PINB.3	These two limit switches are used to detect the state of the door.
#define door_level_1 PINB.4 #define door_level_2 PINB.5 #define door_level_3 PINB.6 #define door_level_4 PINB.7	Inputs from 4 limit switches at each Floors door to detect if any door is open
#define level_pb1 PINC.0 #define level_pb2 PINC.1 #define level_pb3 PINC.2 #define level_pb4 PINC.3	Floor buttons to summon the car
#define cabin_pb1 PINC.4 #define cabin_pb2 PINC.5 #define cabin_pb3 PINC.6 #define cabin_pb4 PINC.7	Car keypad push buttons
Lcd_Show Function	

void lcd_show()
{
  switch(cabin)
  {
  case 1: lcd_gotoxy(0,0); lcd_putsf(“   First Floor   “); break;
  case 2: lcd_gotoxy(0,0); lcd_putsf(“   Second Floor   ”); break;
	First line of the LCD

case 3: lcd_gotoxy(0,0); lcd_putsf("   Third Floor   "); break;
case 4: lcd_gotoxy(0,0); lcd_putsf("   Fourth Floor   "); break;
}	

if((motor_up==0)&&(motor_down==0)&&(updownmotor_lcd==1))
{
updownmotor_lcd=0;
lcd_gotoxy(0,1);
lcd_putsf("   ");
}
if((cabinmotor_open==0)&&(cabinmotor_close==0)&&(cabinmotor_lcd==1))
{
cabinmotor_lcd=0;
lcd_gotoxy(0,1);
lcd_putsf("   ");
}
if(motor_up==1)
{
updownmotor_lcd=1;
lcd_gotoxy(0,1);
lcd_putsf(" Motor going Upward ");
}
if(motor_down==1)
…↓
	Second line of the LCD

if(((up_alarm==1)||(down_alarm==1)||(c_alarm==1)||(l_alarm==1))&&(count4==1))
{	

count4=0;
lcd_gotoxy(0,2);
if(up_alarm==1)
{
lcd_putsf(” Up Emergency is on “);
goto lable2;
}
if(down_alarm==1)
{
…↓
}
…↓
}
	Send Error to third line of the LCD

lcd_gotoxy(0,3);
lcd_putsf(“WWW.PLCGOODS.COM/NET”);
}
	Fourth line of the LCD

Show Function	

void show()
{
ds=alarm_light;
sh_cp=0;
sh_cp=1;
ds=cabin_light;
sh_cp=0;
sh_cp=1;
…
…↓
…
ds=cabin_led1;	

` sh_cp=0;`
` sh_cp=1;`
` for(i=0;i<7;i++)`
` {`
` ds=shift&seven_seg[cabin];`
` sh_cp=0;`
` sh_cp=1;`
` shift=shift<<1;`
` }`
` shift=2;`
` ds=motor_down;`
` sh_cp=0;`
` sh_cp=1;`
` ds=motor_up;`
` sh_cp=0;`
` sh_cp=1;`
`/*********** end *****************/`
` st_cp=0;`
` st_cp=1;`
`}`
	This function updates the outputs that are connected to three 74HC595. Notice that alarm light is the last outputs of these three 74HC595 (Error! Reference source not found.). It is sent first and it activates right pin of the 3rd 74HC595 by executing the Shift instruction.

find_cabin Function	

`void find_cabin()`
`{`
` //if((level_ms1==1)&&(level_ms2==1))    //1`
` if(level_ms1==1)`
` {`
` cabin=1;`
	This function detects the car current position and Updates the cabin parameter to use it in other places.

` if((level1_call==1)||(cabin1_call==1))`
` {`	

level1_call=0;
cabin1_call=0;
level_led1=0;
cabin_led1=0;
motor_up=0;
motor_down=0;
up_led=0;
down_led=0;
if(opened_door_ms==0)
cabinmotor_open=1;
}
	Also if the cabin has reached to a desired floor, it starts to update some parameters and outputs.

if(((level2_call==1)||(level3_call==1)||(level4_call==1)||(cabin2_call==1)||(cabin3_call==1)
||(cabin4_call==1))&&(opened_door_ms==1)&&(aa==0))
{
TCCR2=0x04;
if(wait==1)
{
aa=1;
wait=0;
cabinmotor_close=1;
TCCR2=0x00;
count_2=0;
}
}
bb=0;
cc=0;
dd=0;
show();
}
	If there is a request for other floors and door is open, after 3 second the door start is closed.
	
	We want this part of program to be executed only once so we use aa, bb, cc and dd parameters, especially in the startup condition.

…↓	

if(opened_door_ms==1)
{
cabinmotor_open=0;
}
	If the door is closed, motor is turned off

if((level1_call==0)&&(level2_call==0)&& (level3_call==0)&&(level4_call==0) && (cabin1_call==0)&&(cabin2_call==0)&& (cabin3_call==0)&&(cabin4_call==0)&& (cabin_light==1))
TCCR1B=0x03;
}
	Code turns on Timer3 if there is no any service call request from any floor and cabin light is on.

level_pushbuttons Function	

void level_pushbuttons()
{
if((level_pb1==1)||(cabin_pb1==1))
{
l_pb1=1;
if(level_pb1==1)
level1_call=1;
if(cabin_pb1==1)
cabin1_call=1;
TCCR1B=0x00;
cabin_light=1;
}
…↓
	This part stores all service calls information in a memory. Code turns cabin light on and turn timer1 off that will be used at the end of car destination.

if((l_pb1==1)&&(closed_door_ms==1)&& (cabinmotor_open==0)&&(motor_up==0)&& (motor_down==0))	

```
  {
  cabinmotor_close=0;
  //if((cabin>1)&&(b==1)&&(c==1)&&(d==1))
  //if((cabin>1)&&(b==1))
  if(cabin==2)
  {
  motor_up=0;
  up_led=0;
  motor_down=1;
  l_pb1=0;
  goto lable;
  }
  if((cabin>1)&&(l_pb2==0)&&(l_pb3==0))
  {
  motor_up=0;
  up_led=0;
  motor_down=1;
  l_pb1=0;
  }
  lable:
  }
  …↓
```

In this part the program, system detects which floor's push button is pressed and then controls the motor accordingly to make the car go up or down, and then turns off the motor.

level_doors_alarm Function

```
void level_doors_alarm()
{
  if((door_level_1==1)||(door_level_2==1)||(door_level_3==1)||(door_level_4==1))
  {
  switch(cabin)
  {
```

case 1:
if(door_level_2==1)
level2_alarm=1;
if(door_level_3==1)
level3_alarm=1;
if(door_level_4==1)
level4_alarm=1;
break;
…↓
}
…↓
}
	As an example, if cabin==1 and the doors of 2, 3 and the 4th are open, then we have an 'Alarm' case to display on the LCD. So this part of the code is responsible to set some parameters that will be used in lcd_show function.

updown_emergency Function	

void updown_emergency()
{
if(up_emergency==1)
{
perm=0;
if(motor_up==1)
{
level1_call=0; level2_call=0; level3_call=0; level4_call=0;
cabin1_call=0; cabin2_call=0; cabin3_call=0; cabin4_call=0;
l_pb1=0; l_pb2=0; l_pb3=0; l_pb4=0;
level_led1=0; level_led2=0; level_led3=0; level_led4=0;
cabin_led1=0; cabin_led2=0; cabin_led3=0; cabin_led4=0;
}
motor_up=0;
	If an emergency limit switch is activated, then the control program turns the system off and clears some parameters for the next start of the system.

```
  up_led=0;
  up_alarm=1;
  show();
  }
  if(down_emergency==1)
  {
  …↓
  }
}
```

alarm_off Function

```
void alarm_off()
{
if((door_level_1==0)&&
(door_level_2==0)&&(door_level_3==0)&&
(door_level_4==0))
  {
  switch(cabin)
  {
  case 1:
  if(door_level_2==0)
  level2_alarm=0;
  if(door_level_3==0)
  level3_alarm=0;
  if(door_level_4==0)
  level4_alarm=0;
  break;
  case2:
  …↓
  }
…↓
}
```

This function checks the states of system and if there are no more 'Alarm' cases, it clears all 'Alarms' cases which were turned previously for some reasons.

Other parts of program	

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
// Place your code here
  c_alarm=!c_alarm;
  cabin_emergency=c_alarm;
}
	Each time, that the internal 'emergency' button is pressed, this external interrupt is activated and it reverses the value of c-alarm variable.

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
…↓
}
	Timer 0, Timer1 and Timer2

void main(void)
{
…↓
}
	Main program to recall functions

Step 3:
Getting to manufacturing the PCBs of the prototype model lift system

It was very encouraging to see that simulation was carried out successfully and now we wish to get the PCBs built and test the lift system functionality manually in the first place. This might even be related to a project that we are working on as a college final year project, and our instructors have advised us that the course grade will not be give in full unless we show them the whole hardware of the system working properly!

No sweat again. Some other big brothers already have developed great software, and their application allows us to design and generate the PCB layout of our circuits. Altium Designer ® which is a registered trademark of Altiume Company is a sample of that software that can be used to create PCBs of any complexity.

It seems that I am going to need four PCBs: one to simulate keypad/display that is needed to be installed at floor (or stop), the second keypad/display to be installed inside the car, the third PCB for the controller section of the project and finally, the fourth one to be used to drive two geared DC motors. The layout of the PCBs needed to implement this project, will look similar to the one shown in Figure 1-2.

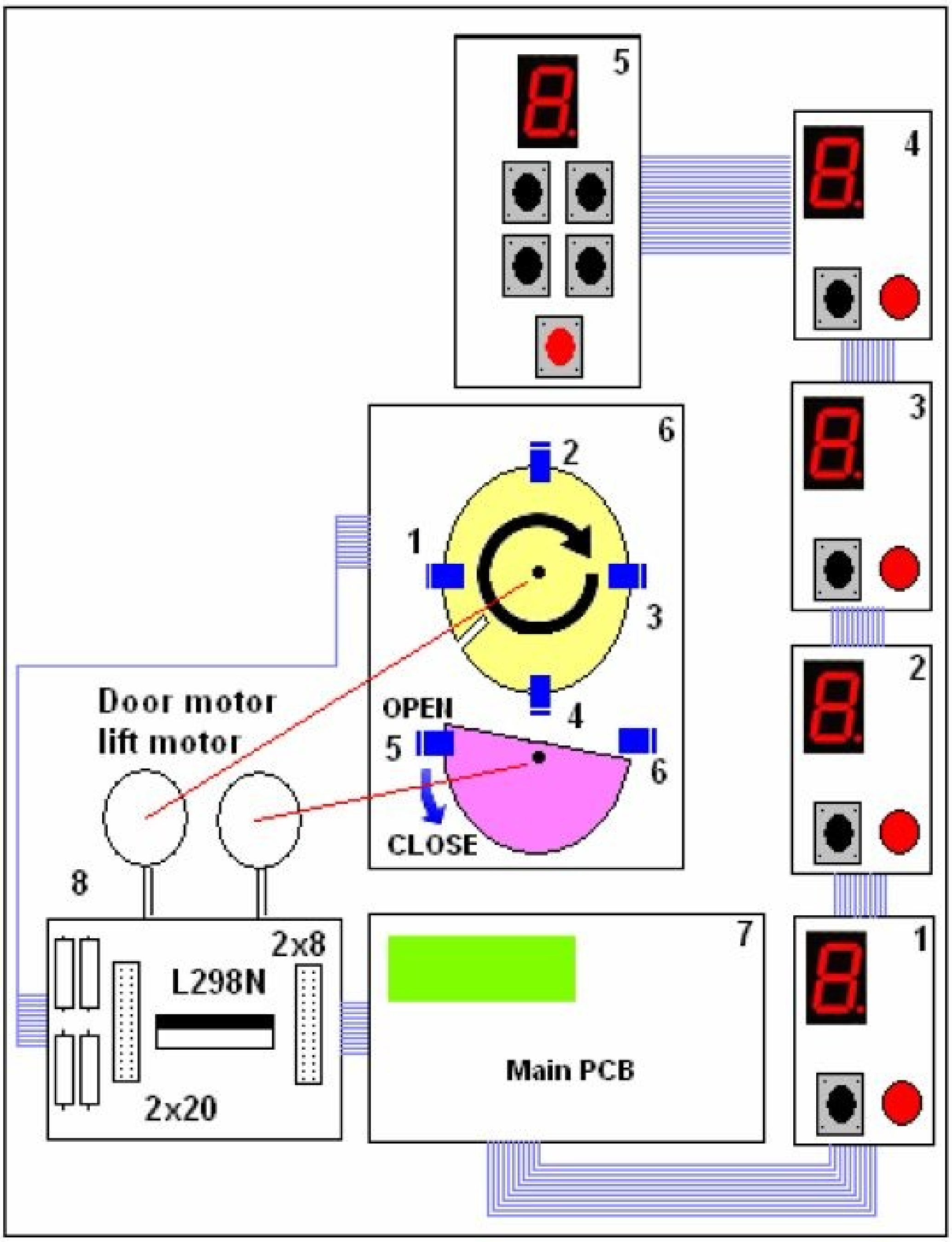


Figure 1-2

Figures 1-3 to 1-6 show the images of the PCBs designed for this project.

Figure 1-5 is related to the section that is holding two geared DC motors to simulate the hoist and car sliding door.

Hardware aspect of the project

From Step 1, we had an idea on how our project hardware should look like. It needs to have about five 7-segments to display the current stop #. It needs to have 5 keypads with displays (one installed inside the car and 4 others at each destination stop), it needs to have a mechanism to simulate the movement of the car. So let's put all the parts that we think are needed, on a paper and create a basic sketch of those sections of the lift system.

Figure 1-2 displays the simplified images of the PCBs I needed to design to implement the hardware of this project. As you can view in Figure 1-2, the project hardware is based on 7 PCBs design to implement the whole lift system.

Hardware of the project consists of eight PCBs. According to the Figure 1-2, 4 × 4.5 × 5.5 CM PCBs are designed to simulate each floor keypad and display (PCBs designated by number 1 to 4). One 5.5 cm ×8 cm PCB is designed to simulate the car internal keypad/display (PCB piece designated by number 4). One 13 cm× 8.5 cm PCB is designed to simulate the main elevator controller keypad (PCB # 7). PCB #8 drives two DC motors. PCB # 8 is designed to simulate both the hoist and car door motors.

Functional explanation of PCB # 1 to PCB #4

Each of these four PCBs is supposed to be installed in each floor in a way to be used to generate signals when the corresponding push button is depressed. Figure 1-3 displays more details of the components installed on each related PCB # 1 to PCB # 4. Notice that each PCB has one N.O push button to be used when there is a need to call for service. Upon depression of say P.B # 1, main control software detects a signal coming from first floor, and hence will direct the car to floor # 1. All 7-segments are connected in parallel and will display the current floor number of the car location automatically, and will be updated automatically as the car passes by each floor.

The LED indicator located at the right side of the 'service call' P.B starts flickering when the corresponding P.B is depressed, to indicate that the passenger at that particular floor needs a service.

Each PCB has three 2× 4 pin headers to set the address of each PCB to the controller. By using 3 jumpers (shown in red color), a user can define the address of each PCB for the controller.

Two LED indicators installed on the right side of the 7-segment display indicate if the car is going upward or downward, by flickering either a green or red LED respectively.

When push button designated with 'DL' is pressed, for the controller it means that a car door is left open. In practice, any car door besides the one in use, is not supposed to be left open. In our case, that is considered a 'fault', and hence when it is left open, the 'Alarm' LED located on the car keypad PCB (Figure 1-3) will start flickering and the hoist motor is not allowed to function as usual.

One 2×15 pin header at the bottom of the P.B is used to connect all these PCBs to the main controller PCB via a 2× 15 flat cable and related IDC connector.

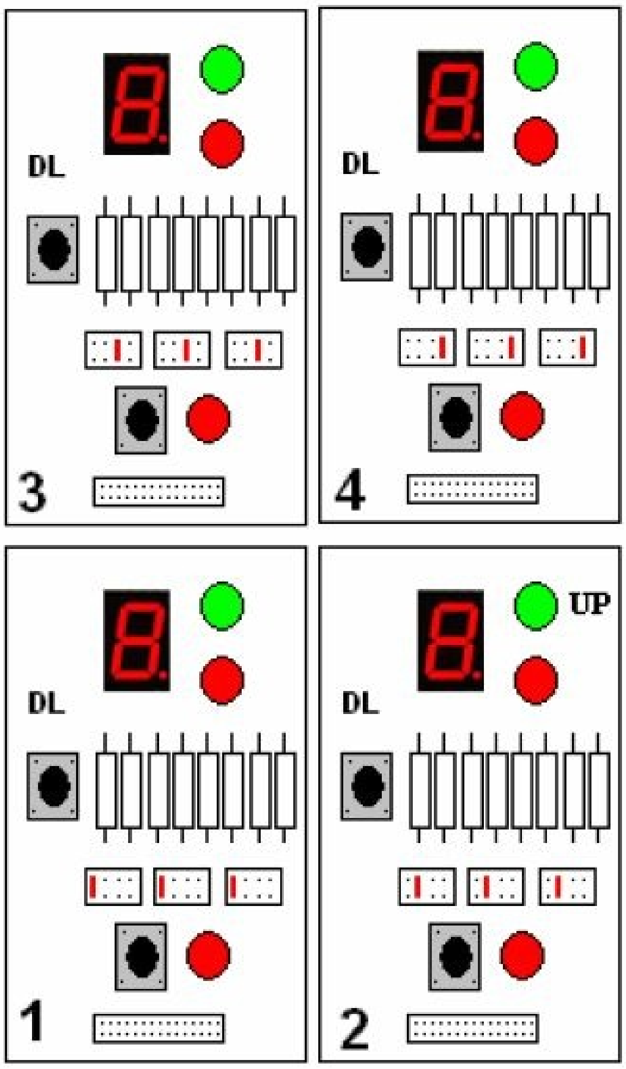


Figure 1-3

Functional explanation of PCB # 5: Car Operating Panel

PCB # 5 has a total of 5 P.B and five related red LEDs. Since this PCB is designed to be installed inside the lift car, it can be used to generate signals to cause the lift to have different stops at any of the four floors. As an example, when say P.Bs designated with numbers 4 and 1 are pressed, the related LED indicators 1- start flickering (to confirm that those P.B are pressed) 2- Yellow LED (installed on the left side of the 7-segment) is turned on 3- Lift starts moving the car to the first destination selected by the passengers inside the board (based on the priority integrated in the software) 4- Either UP or Down LED signal lamp starts flickering to indicate the direction of the car movement.

By pressing 'EMR.' push button momentarily, 'Alarm LED' lamp starts flickering to indicate that there exists a 'faulty' situation with the lift system, and it means that passengers inside the car are asking for 'help'. This is actually simulated by switching on an emergency light in the elevator, to avoid leaving trapped passengers in the dark as they wait for assistance. In practice, this 'Alarm' system must be installed outside the lift system, to get help from outsiders in case of any 'emergency case'. Pushing 'EMR.' button momentarily causes 'Alarm' indicator to be set its 'emergency status'. It functions similar to a Flip Flop, meaning that as long as it is not reset, it continues to turn the red LED on .

Two pin headers installed on the left and the one located on the right bottom of this PCB are used to connect the components to the main controller.

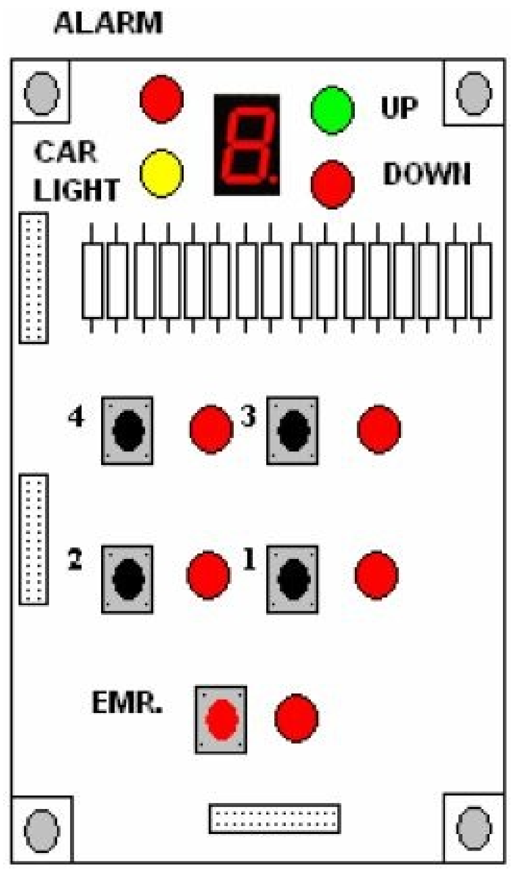


Figure 1-4

Functional explanation of part # 6: hoist and car door fixture

In Figure 1-2 the piece of fixture that is designated as # 6, is the mechanism designed to simulate Elevator’s hoist (at the top) and car door motors. To view this project in action, you may click on the following link:
http://youtu.be/UQbGcntcVP4

Simulating the host lift and car door motors

The disk on the top of the fixture (see Figure 1-5) simulates the up or down car movement of the elevator easily. There is a cabin of lift which will be moved up or down across different floors, with the help of a small DC motor which its shaft mounted to the center of a disk. In this project of lift, a single motor is used which will turn the disk clock or anticlockwise. When the disk is turning clockwise, it is simulating downward car movement. On the contrary, when it is being turned anticlockwise, it is simulating upward car movement.

This project is simulating operation of a lift with four stops. For floor detection and to generate an arrival signal of the car to any selected stop, four slotted optocouplers are used (4 pieces shown in blue color). For this project, I needed four photo interrupters- each is a small device that shines a light from one side to the other, where there is a detector.

The idea is that if anything breaks the light beam, you take note of it, and use the information (signal) as you wish. In my case, I am using a disk which has a small rectangular area cut off from it, thus the beam will be broken except for that one 'spot'. I have installed four optocouplers around the disk corresponding to four floors. Thus, when the cut in section creates an interrupt at any related optocoupler, Microprocessor deletes the stop number and performs appropriate action accordingly. I used a total of six slotted optocouplers in this lift implementing project, one sensor for each stop and two more to detect if the car door is open or closed.

The half disk located at the bottom of the piece # 6 is related to opening and closing mechanism of the car door. Notice again that I have used two more optocouplers to detect if the door is open or closed. When sensor # 5 is interrupted (= 0) but sensor # 6 is uninterrupted (= 1), the status of the door is 'open'. To close the door, controller has to activate the door motor in anticlockwise (in the direction shown with the blue arrow) to cause Sensor # 5 = 1and Sensor # 6 = 0.

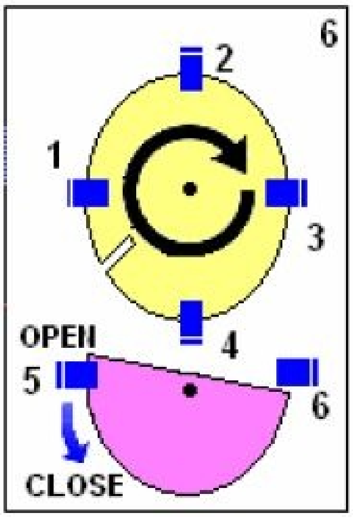


Figure 1-5

Functional explanation of PCB # 7: Main controller PCB

Five signals will be generated from the lift cabin (in Figure 1-2, PCB # 5): these are push buttons and will be used to request the lift to go to the desire floor. Notice that the fifth push button is related to 'Emergency case' (the push button with a red dot at its center). By momentary pushing this push button, a red indicator on the top left side of the cabin keypad starts flickering to inform others outside of the elevator that there is an 'emergency' case with the lift. In practice, this indicator must be installed outside of the lift system. In figure 1-3, Alarm light is indicated on the top left side of the PCB as red LED. Thus the lift cabin will have a total of five push switches for this purpose, one for each floor and the fifth one for the 'emergency' case.

Four signals will be generated outside the lift cabin, one from each floor. Again, these would be push buttons and their purpose will be to call the lift on desired floor, if it is not stationed there currently. The status of lift will be indicated with the help of 5 seven segment displays, which are all connected to the controller in parallel. One seven segment display is installed on each floor; which will show the current location of the lift. Following display pattern would be visible on seven segment display of lift project:

1. It means lift is on floor # 1 which is first floor.
2. It means lift is on floor # 2 which is second floor.
3. It means lift is on floor # 3 which is third floor.
4. It means lift is on floor # 4 which is forth floor.

Besides the 5 seven segment displays which always show the car's current stop number, a LCD which is wired to Microcontroller port also provides some more information such as if any lift door is open, the direction of lift movement (e.g. car is being moved up or down) etc.

When the lift is just powered up, it will go to its home position, which is the first floor. On reaching the first floor, display will show the current floor number.

From figure 1-2, it is obvious that the controller PCB is responsible to receive all input signals generated from all other field devices (all push buttons, optocouplers, jumper, switches etc) and to send output signals to the related devices (e.g. two DC motors, 5 seven segment displays, LEDs, LCD, etc).

Figure 1-5 displays more detailed image of the controller PCB, which is designed to control all I/O signals regarding this project. The main ingredient of the control Board is an ATmega16 Microcontroller, which is programmed in a way to do the control task easily.

In Figure 1-6, notice that switches S1 to S5 play the roll of 5 mechanical limit switches which can be used to simulate four opotocouplers, to detect location of car when stopped at any floor.

In Figure 1-7, notice how five switches S1 to S5 can be used to simulate 4 optocouplers to generate stop number signal. For example when S1 and S2 are depressed manually, the related LEDs are turned on and the output signals are in 'High' status and Microcontroller assumes that the car is parked at forth floor. Since in this project we are using opotocouplers instead of switches, just ignore their existence.

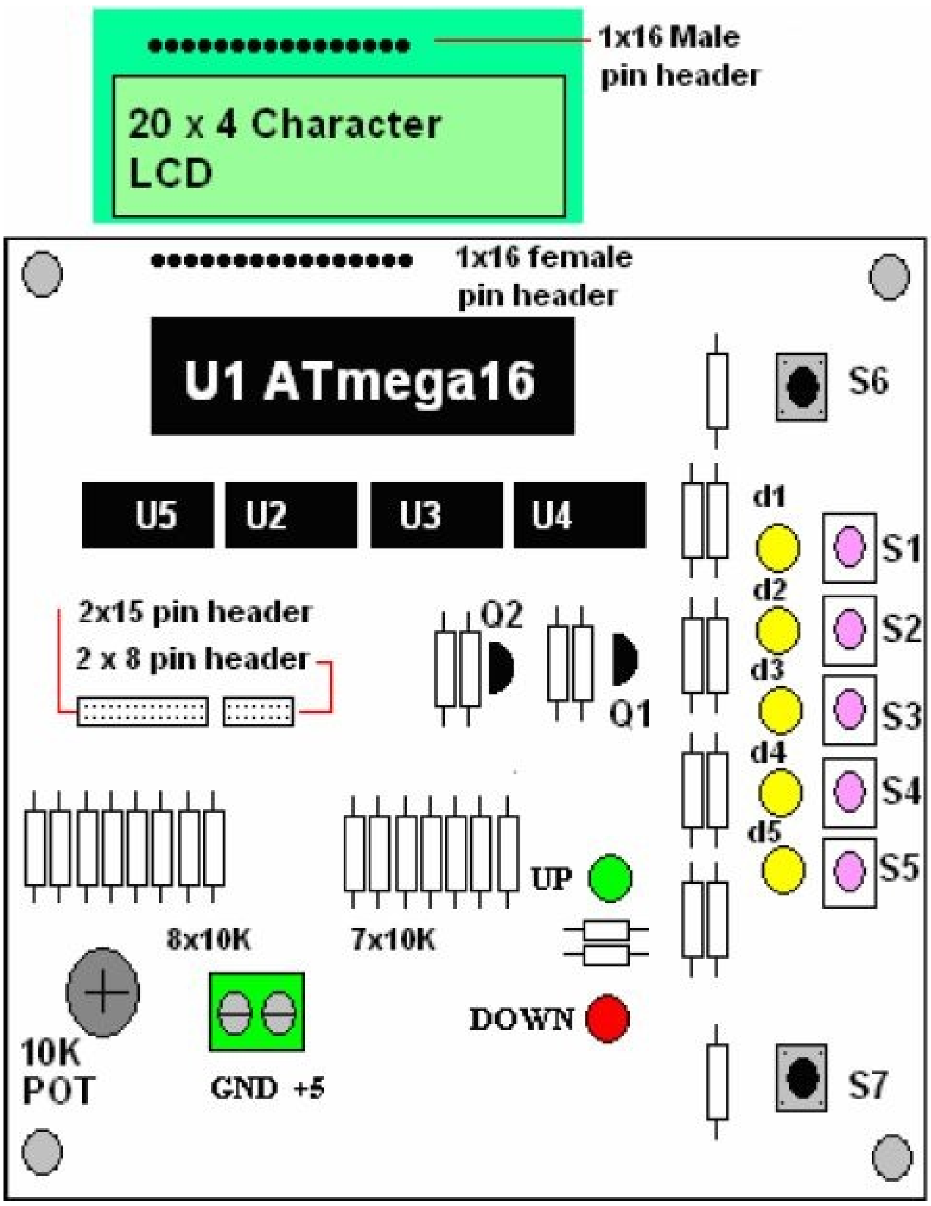


Figure 1-6

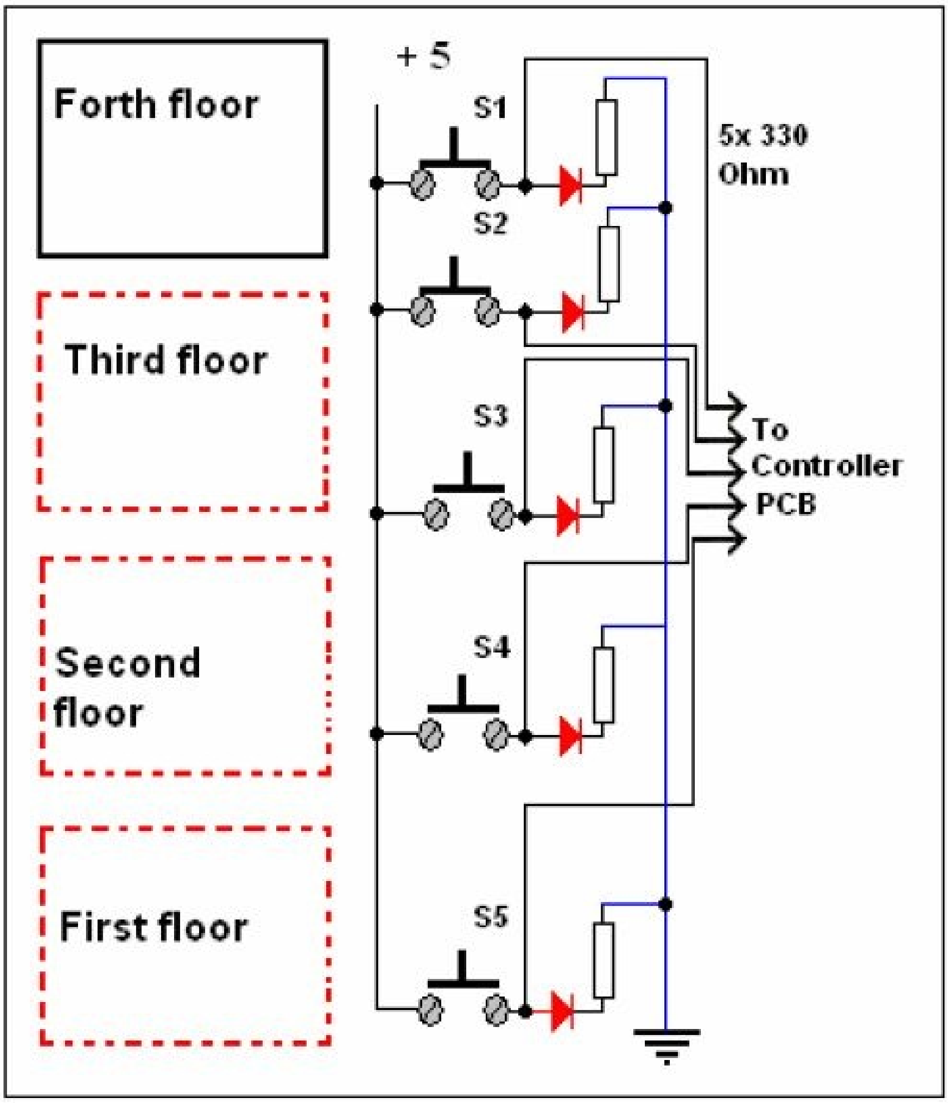

Forth floor
Third floor
Second floor
First floor
+ 5
S1
S2
S3
S4
S5
5x 330 Ohm
To Controller PCB

Figure 1-7

Functional explanation of PCB # 8: DC motor driver

It was mentioned previously that we have two DC small motors in our project for the hoist and the car door. To control these two small DC motors, I used a **L298N IC** which is a Full-Bridge Motor Driver. It can source/sink up to four amperes at about 40 volts with proper heat sinking. 2 DC motors can be connected to the output of the driver. Microcontroller can control the inputs so that the motor terminals are connected to Ground or Vcc. So, voltage can be applied in either direction through the motors, or both terminals can be connected to Ground to stop the motor. Internally the L298N consists of four independent power amps with 5-volt digital inputs and four high current, high voltage power amplifiers capable of driving single DC motors, and both unipolar and bi-polar stepper motors. I used two male pin header connectors and related flat cable to connect the DC motor driver to controller and DC motors. Figure 1-7 shows the actual image of the prototype made for this project.

Figure 1-8 illustrates a typical schematic diagram to activate two DC motors.

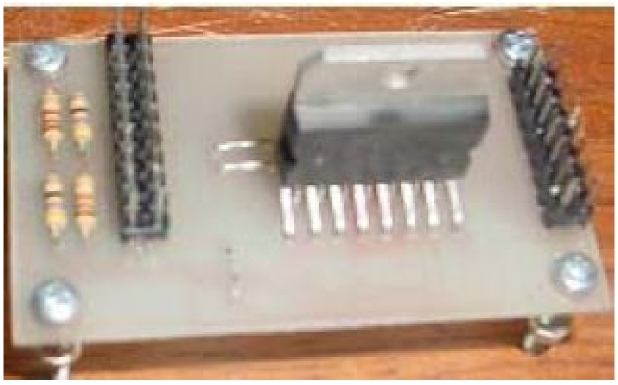


Figure 1-8

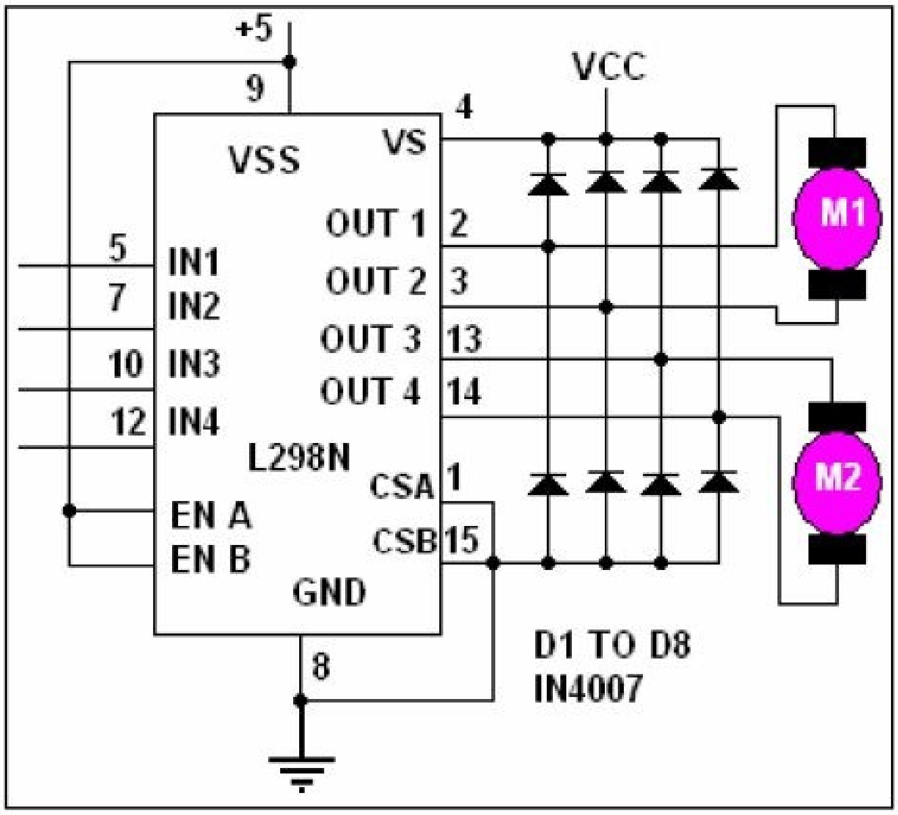


Figure 1-9

Step 4:
Building the PCBs of the prototype model lift system

It is very encouraging to see that simulation has been carried out successfully, and now we wish to get the PCBs built and test the lift system functionality manually in the first place. This might also be related to a project that we are working on it as a college final year project and our instructors have advised us the course grade will not be given in full, unless we show them the whole hardware of the system working properly!

No sweat again. Some other big brothers already have developed great software that allows us to generate the PCB layout of our circuit. Altium Designer ® which is a registered trademark of Altiume Company, is a sample of that software that can be used to create PCBs of any complexity.

Figures 1-10 to 1-14 display images of the PCBs designed for this project.

Figure 1-14 is related to the section that is holding two geared DC motors to simulate the hoist and car sliding door motors.

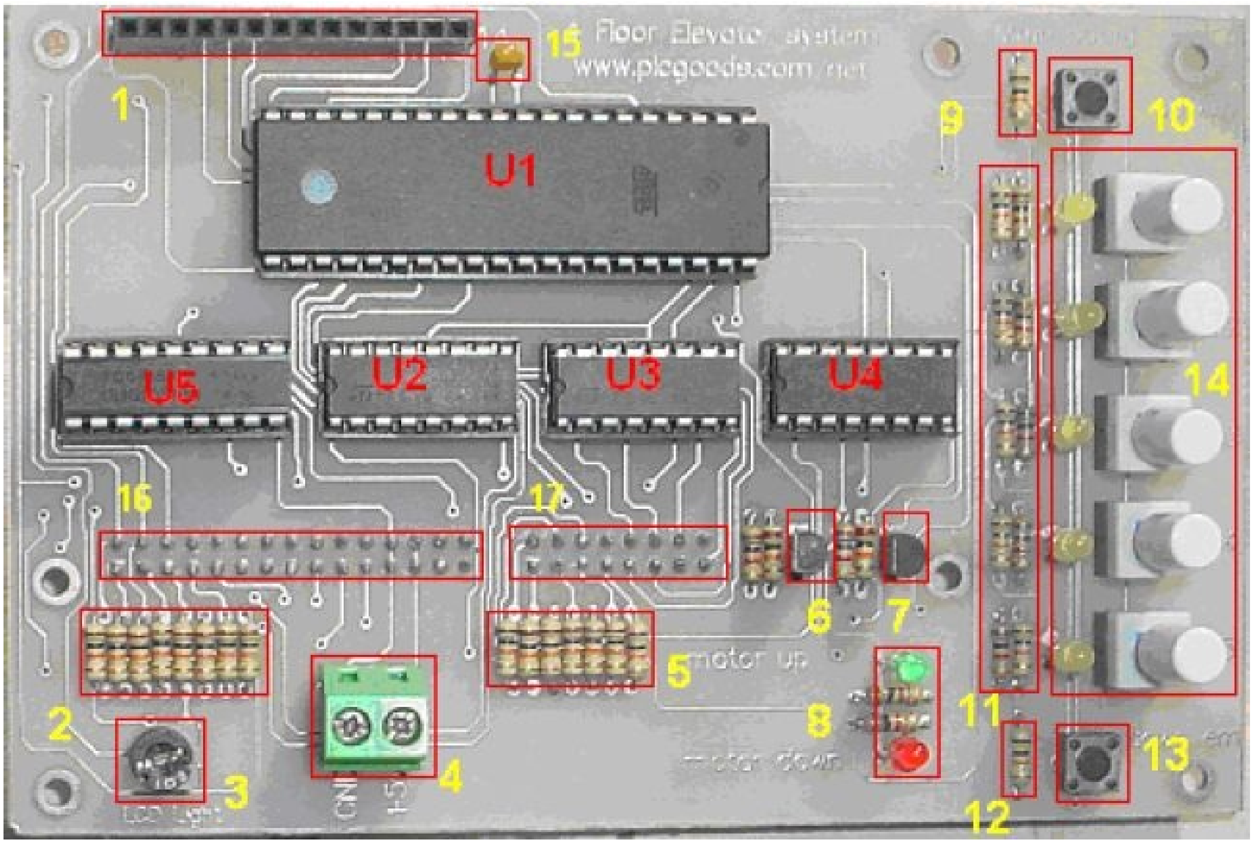


Figure 1-10

Figure 1-10 Notes

1. U1 ATmega16 Microcontroller
2. U2, U3, U4 = 3× 74HC595
3. U5 = ULN2803A

Parts designated with a yellow number

4. 1 = 1×14 female pin header

5. 2 = 8 × 10 K ohm 1/4 watt Resistors
6. 3 = 10 K ohm pot
7. 4 = Power in terminal
8. 5 = 7 × 10 K ohm 1/4 watt Resistors
9. Number 6 and 7 2 × C517 transistors
10. 8 = 2×3 millimeter green and red LEDs
11. 9 and 12 = 2×10 K ohm Resistors
12. Number 10 and 13 = 2 small push buttons
13. 11 = 5×1K and 5×10K ohm Resistors
14. 14= 5× small switch used to simulate opotocouplers
15. 15 = 1× 0.1 NMF Ceramic capacitor
16. 16 and 17 = 2×15 and 2×8 male pin header connectors to help to wire controller I/O signals to the other PCBs

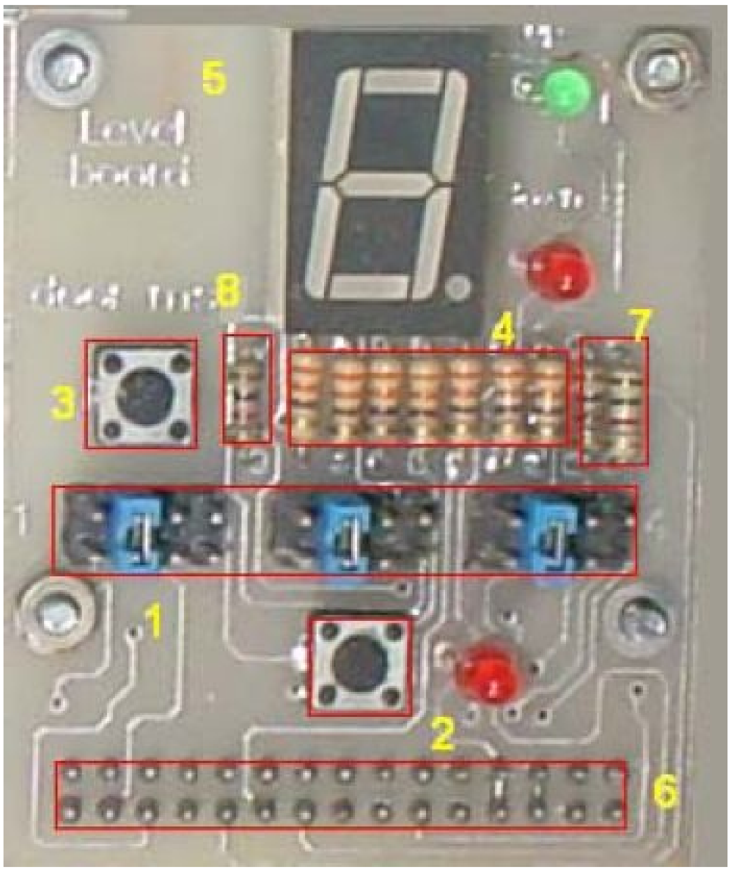


Figure 1-11

Figure 1-11 Notes

1. 3×2×4 male pin headers with related jumpers
2. = 1 small push buttons
3. = 1 small push buttons
4. 7×330 ohm Resistors

5. = 1 × 7-segment display (CC)
6. = 2×15 male pin header
7. 7 and 8 = 3×1K ohm Resistor
8. 3×3 millimeter green and red LEDs

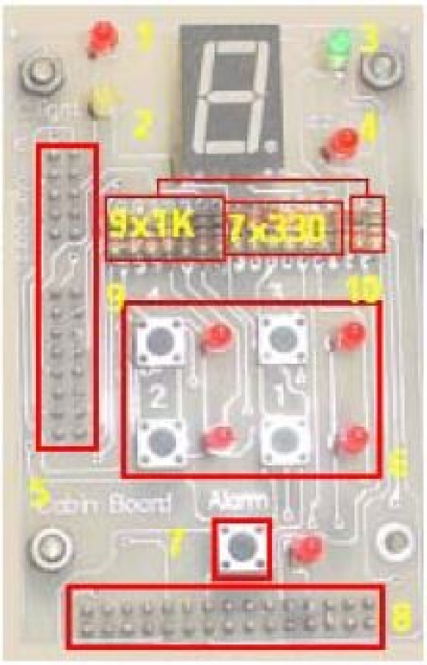


Figure 1-12

Figure 1-12 Notes

1. Alarm LED
2. Car internal light indicator
3. Green LED turns on when car is moving upward
4. Red LED turns on when car is moving downward
5. Two× 2 ×8 and 5 male pin header
6. Four push buttons and its related LEDs
7. Emergency push button and its related LEDs
8. 1×2×15 male pin header
9. Nine 1K ohm Resistors to bias 9 red LEDs
10. Seven×330 ohm Resistors to bias the 7-segment display

Figure 1-13

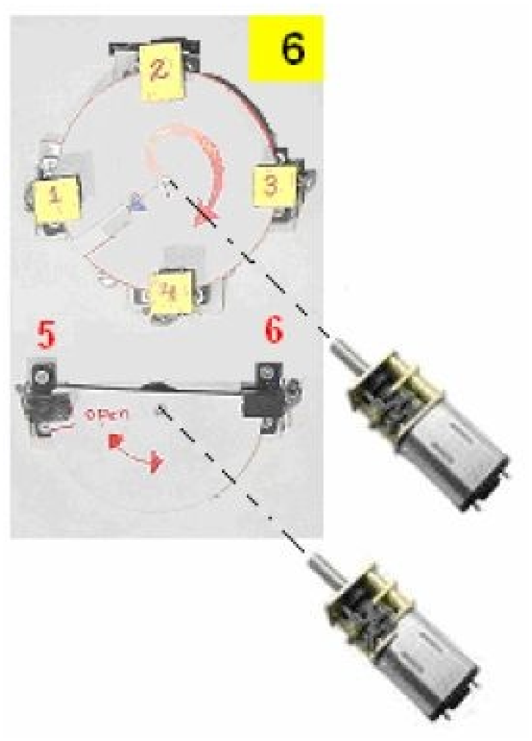


Figure 1-14 the locations where 6 slotted opotocuplers are installed

Figure 1-14 Notes

1. 1 to 6 = 6×slotted Opotocouplers
2. 2× Geared DC motors

Step 5:

Uploading the control program into the ATmega16 microcontroller (hex file)

Now that all needed PCBs are manufactured according to Step4, last step before powering up the system is to 1- assemble the PCBs 2- burn the code developed at step 1 into AVR ATmega16 controller user memory

I used my little mega16 programmer to upload the hex file developed in step 2 and complete project final step.

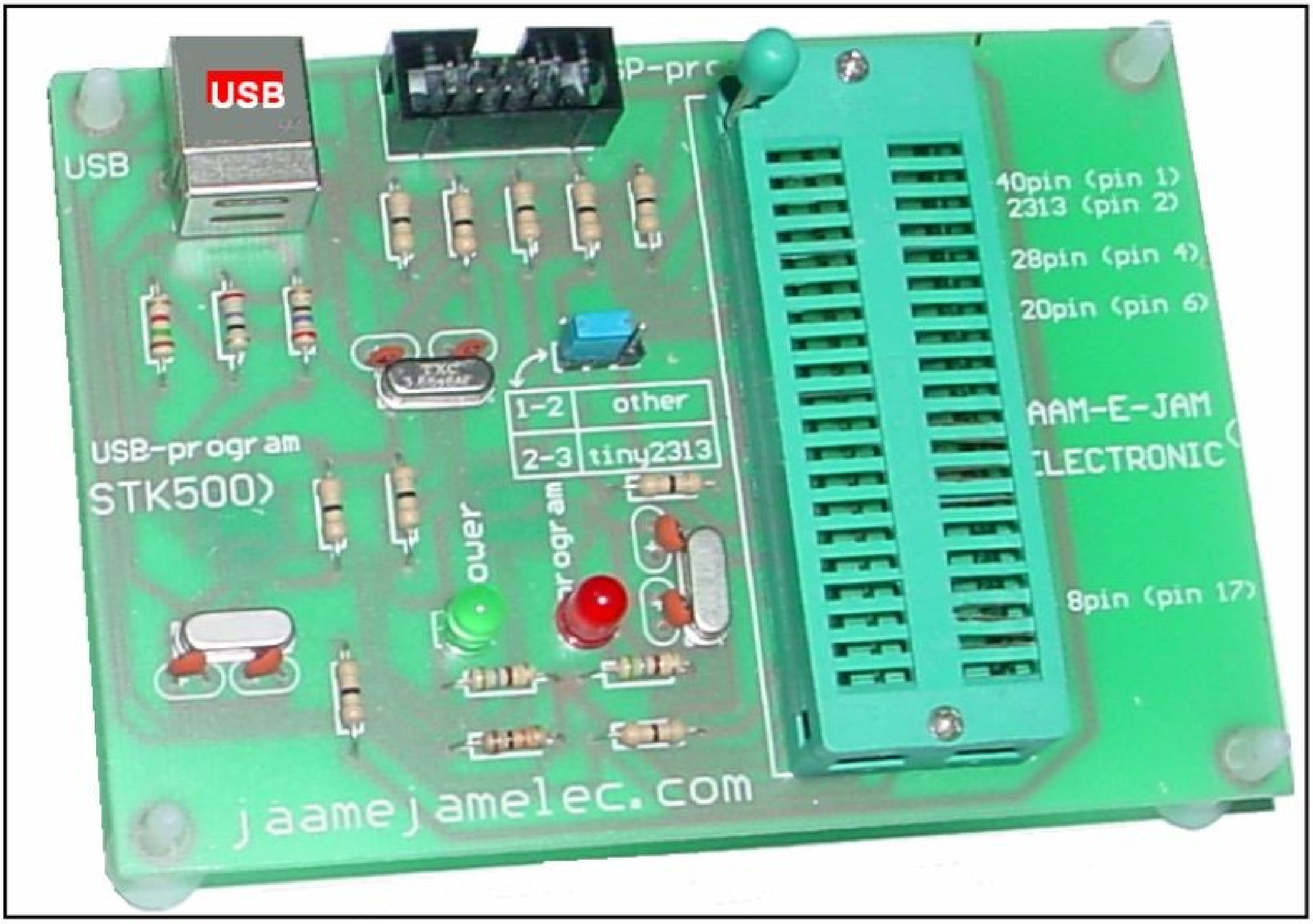


Figure 3.15 displays programmer used to program Atmega16 micro controller

Chapter 2

Design and Implementation of the 4 Floor Control System using an Arduino board

The Arduino board has a nearly limitless array of innovative applications for everything from robotics and lighting to games etc. It's a fun way to automate everything, enabling you to control simple devices or manage other complex projects.

In this text, I selected an Arduino to control a 4 Floor Control System and I think if you manage to do the same project using the Arduino Development Board, you can learn how to educate and inspire yourself to make great things with easily-available tools.

To view this project using an Arduino board in action, click the following link: http://youtu.be/WU2arv-yghw

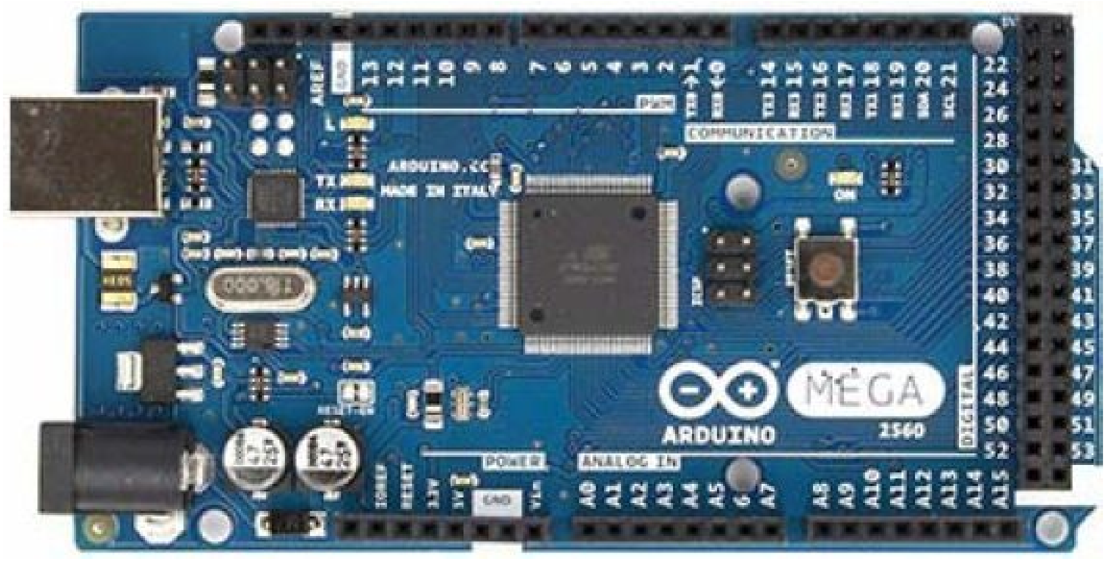


Figure 2-1 displays the Arduino MEGA 2560 Development Board

Step 1:

Generating schematic diagram of the project

The same schematic diagram (less few components) that was designed for AVR Microcontroller will be used in this section of the project. Notice that in Figure 2-2, ATmega16 Microcontroller is replaced with the Arduino Development Board.

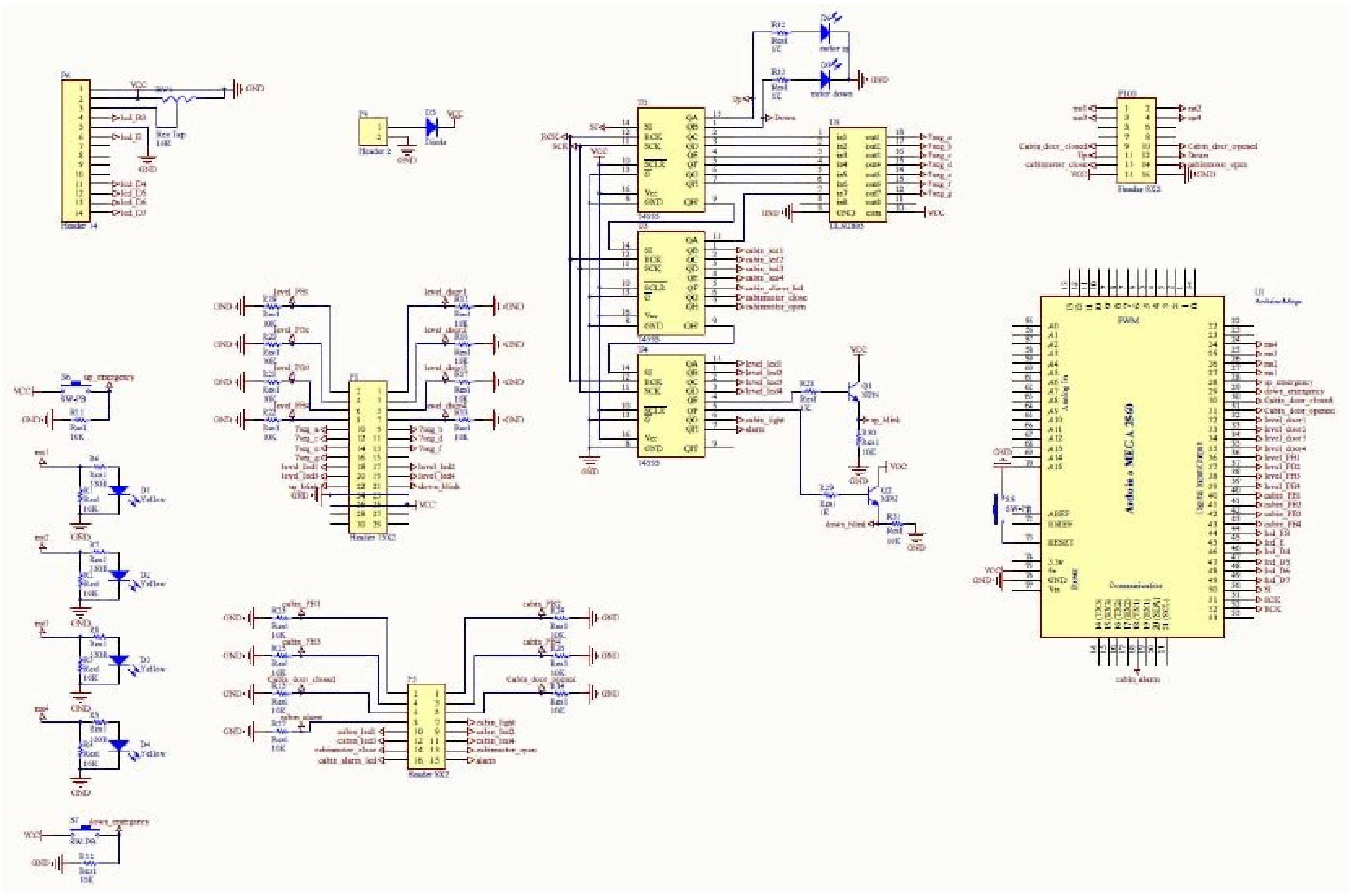


Figure 2-2 The wiring diagram of the project based on an Arduino board

Step 2:

Generating the main Control Code for the Arduino Microcontroller

Program Explanations

Now we’re going to explain each part of the control code adapted for the Arduino Board based on the logic that is developed already for AVR program presented in chapter 1, except for a few differences between these two codes. At first, we define symbols for Arduino pins. Table 2-1 shows definitions of symbols used in the presented code.

Program Explanations

Now we’re going to explain each part of the control code adapted for the Arduino Board and it can be a review of logical we already developed for AVR program presented in chapter 1 beside few differences between these two code. At first, we define symbols for Arduino pins. Table 2-1 shows definitions of symbols used in the presented code.

#include <LiquidCrystal.h>	Define needed library
/************ out ******/ #define ds 50 #define sh_cp 51 #define st_cp 52	Define symbols for 3 output pins that are connected to shift registers that will be used to update our loads: such as LEDs, Motors etc
/************ in *******/ #define level_ms4 24 #define level_ms3 25 #define level_ms2 26 #define level_ms1 27	In hardware of the project, four optocouplers (ms1 to ms4) send signals to these pins.
#define up_emergency 28 #define down_emergency 29	Two Emergency limit switches: EMS_UP and EMS_Down
#define closed_door_ms 30 #define opened_door_ms 31	Two optocoupleres are used to detect the state of the car door.

#define door_level_1 32 #define door_level_2 33 #define door_level_3 34 #define door_level_4 35	These inputs are four limit switches, one at each external stop door.
#define level_pb1 36 #define level_pb2 37 #define level_pb3 38 #define level_pb4 39	Stop buttons to call for the service of the car.
#define cabin_pb1 40 #define cabin_pb2 41 #define cabin_pb3 42 #define cabin_pb4 43	Bottoms located on car keypad\display

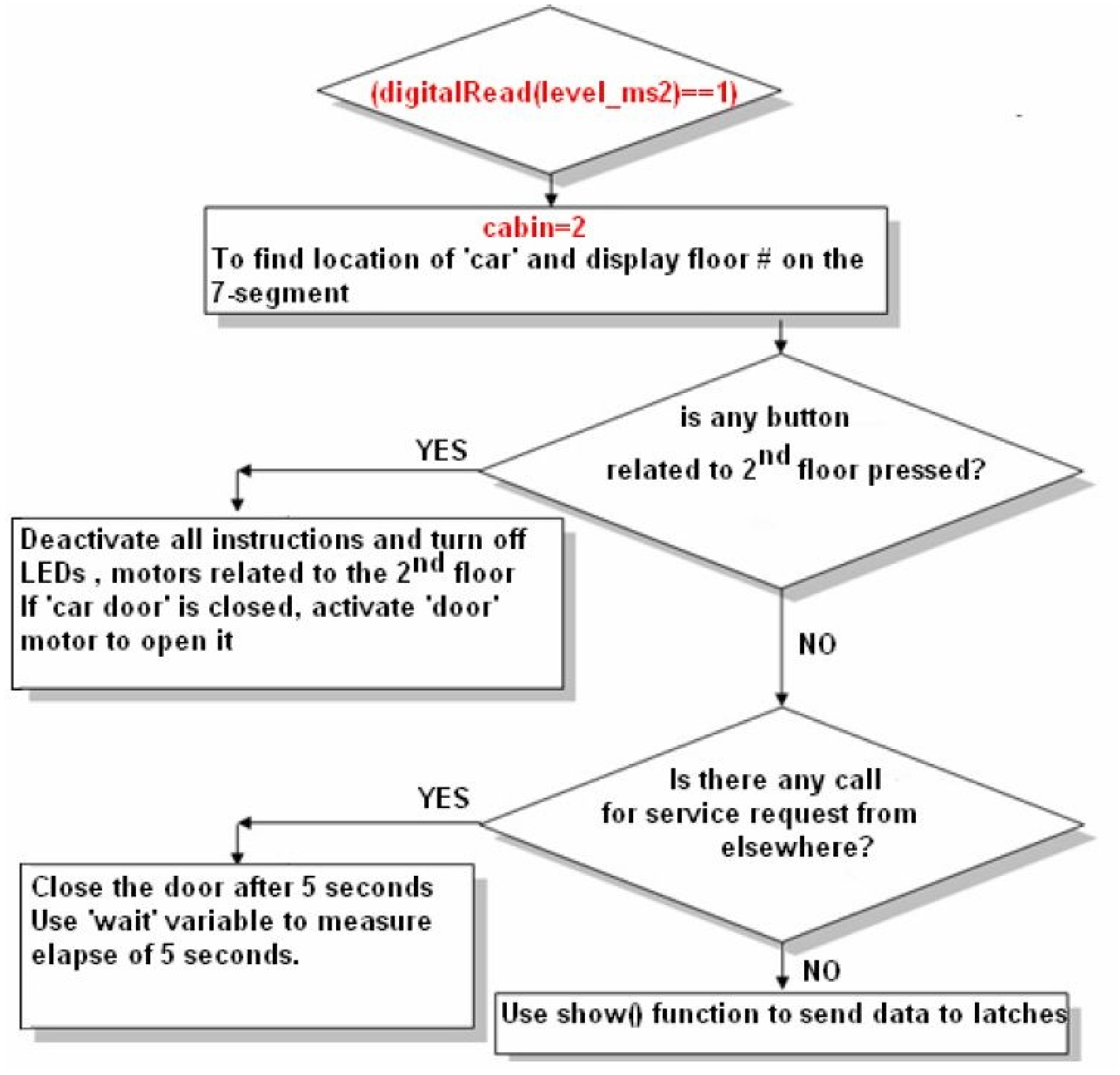
Table 2-1

Show() Function

By application of this function, we can use three pins of Arduino board to send data to three shifts register serially and then send out the outputs of these shift registers as parallel data using the Arduino **ST_CP** pin.

Find_cabin() Function

This function detects the current location of the 'car' to determine if to stop it from going up or downward further. The first issue that this function considers is to check signals coming from 4 optocoupleres. The current location of the car is where any of the four related optocouplers is interrupted (activated). Let's assume the car is located at the 2nd floor. In that case the condition (**digitalRead(level_ms2) ==1**) is true, and the lift will operate based on the following flowchart:


(digitalRead(level_ms2)==1)
cabin=2
To find location of 'car' and display floor # on the 7-segment
is any button related to 2nd floor pressed?
YES
Deactivate all instructions and turn off LEDs , motors related to the 2nd floor If 'car door' is closed, activate 'door' motor to open it
NO
Is there any call for service request from elsewhere?
YES
Close the door after 5 seconds Use 'wait' variable to measure elapse of 5 seconds.
NO
Use show() function to send data to latches

level_pushbuttons() function

When a push button is pressed from either a floor or 'car' internal keypad, following code is executed:

if((digitalRead(level_pb2) =1)||(digitalRead(cabin_pb2) =1))

Then l_pb2 becomes 'True' which is used to direct 'car' towards the 1st floor. Then main code checks to know if the pressed button is located on a floor or 'car' keypad. Based on the finding from the previous Step, one of the variables Level2_call or Cabin2_call becomes 'True' to turn on a related LED. **Light_perm** and **cabin_light** variables are used to turn on 'car light' as well.

When **l_pb2** state = 1, it means the 'car' is to be directed to the 1st floor. In this case, the following conditions must also be satisfied:

✓ (The limit switch behind of the door
is 'activated' (the 'car' external door is closed).

- ✓ (The 'car' door is closed (the related motor to be 'off').
- ✓ (Car isn’t being moved either up or down.

If the above conditions are satisfied, then the lift system turns off the 'door' motor (cabinmotor_close=0).

Then, if the car is in any floor below the 2nd floor (i.e. first floor), hoist motor rotates in anti-clockwise direction to move the car upward. But if the 'car' is located at any floor above the 2nd floor but button related to 3rd floor is not activated, then hoist motor must rotate in clockwise direction to move the car upward.

Level_doors_alarm() function

Another condition defined in the '**scope of the project operation**' is that 'Alarm' light is to be turned on if any of 4 doors is left open without the 'car' being parked at that open door. '**Level_doors_alarm()**' function checks to see if any door is open or not in the first place, and if there is any, it determines if that open door is related to the floor that the 'car' is parked there or not. Therefore, **level1_alarm**, **level2_alarm**, … variables are used to detect any floor with open door and **l-alram** is used to turn the 'Alarm' light on (**alarm_Led** variable).

Updown_emergency() function

If the 'car' is moved higher than 4th or lower than 1st floor stop respectively, any of the two limit switches is activated which means a 'fault' situation. When this type of 'fault' occurs, hoist motor is turned off. So this function performs the following task:

- ✓ (Execution of instructions are cancelled or stopped
- ✓ (Motors are turned off
- ✓ (The alarm (indicator) LED is turned on and the error case is sent to the 7-segment display using **show** function.

Alarm_off function

This function is used to turn off alarm LED if all 'emergency' related cases are resolved. Resolving 'Emergency' related faults mean: none of the doors may be left open, none of 'EMU_UP or 'EMU_down' limit switches may be activated for any reason and finally, 'Alarm' switch many not be activated as well.

Lcd_show() function

This function is used to send messages to LCD according to their priority.

In first line of LCD, current position of the car such as “**Second Floor**” is displayed. The current position of the car is determine from **cabin** variable which is explained when we were explaining **find_cabin()** function.

The second line is considered for other useful messages in the system. We don’t have enough space for displaying all the messages simultaneously. So, we use a priority for those. If there are more than one messages, first a message with higher priority is displayed. The messages displayed in the order of their priority are:

1. Up Emergency.
2. Down Emergency.
3. Cabin Alarm.
4. Door 1 is open.
5. Door 2 is open.
6. Door 3 is open.
7. Door 4 is open.
8. Going Upward.
9. Going Downward.
10. Door is opening.
11. Door is closing.

Alarm_pb() function

By using the instruction (“**attachinterrupt(5,alarm_pb,RISING)**” in **setup** function, whenever the external interup5 is activated (**pin 18**), the code within this function is executed and it reverses the value of **c_alarm** variable (from High to Low or vice versa) to turn the car internal **Alarm** light either off or on respectively.

Step 3:

Getting to manufacturing the PCBs of the prototype model lift system

For this Step, all previous PCBs are going to be used besides the main controller PCB. Since the Arduino Development Board has its own embedded Microcontroller, we need to design and implement only an Arduino driver Shield board to replace PCB # 7 in Figure 1-2. Figure 2-1 displays block diagram of the PCBs, which were used to implement the Elevator project using an Arduino board. In Figure 2-3, notice that PCB# 7 is replaced with an Arduino driver shield board but the other PCBs are the ones which were used in **Chapter 1** to implement the lift system with an AVR Microcontroller.

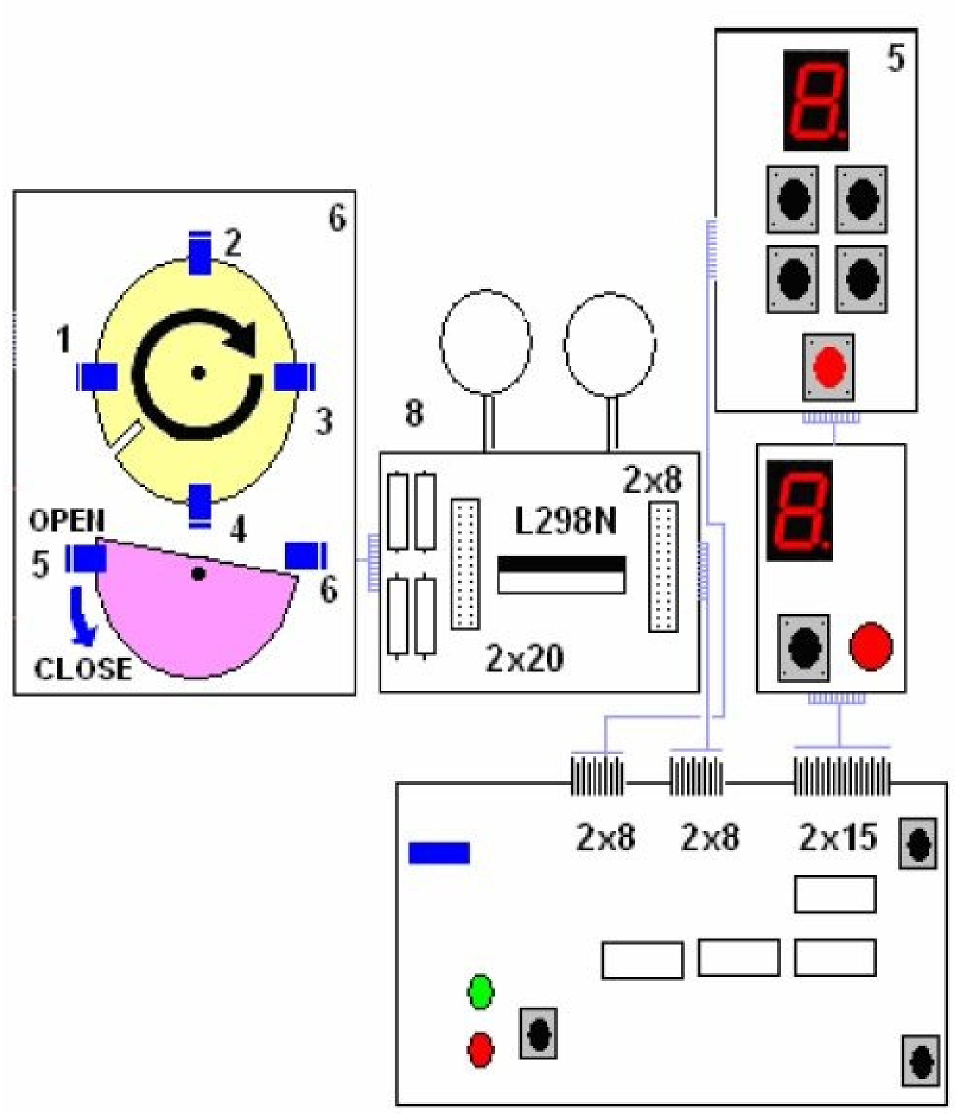


Figure 2-3

Figure 2-4 Notes

1. (1×) 2×8 male pin header
2. (1×) 2×8 male pin headerU2, U3, U4 = 3× 74HC595
3. (1×) 2×8 male pin header U5 = ULN2803A
4. Power in terminal
5. Reset push button 2 = 8 × 10 K ohm 1/4 watt Resistors

6. EMS_up limit switch3
7. EMS_down limit switch
8. 1×14 female pin header to connect LCD display
9. C517 transistor 6
10. C517 transistor
11. 16 character by 2 line LCD display

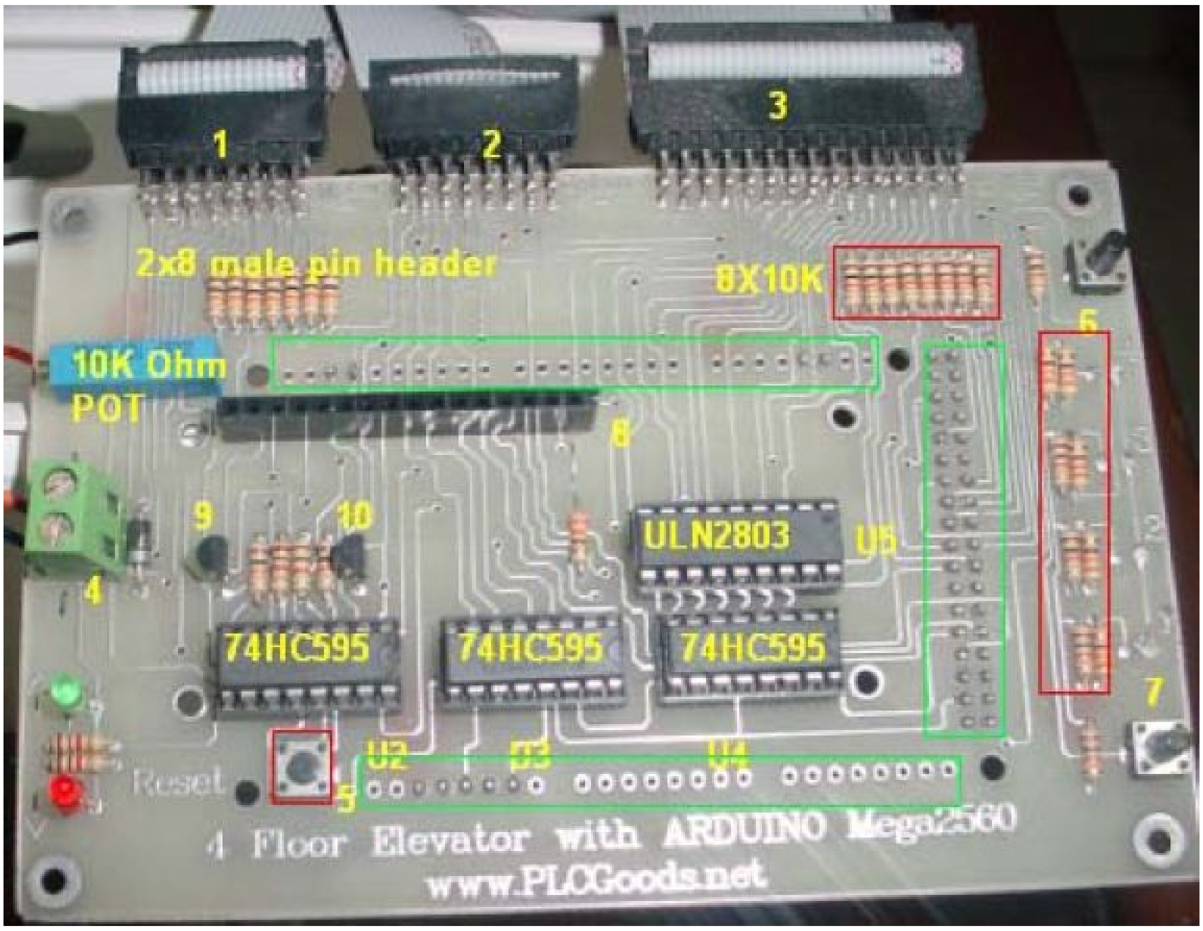


Figure 2-4

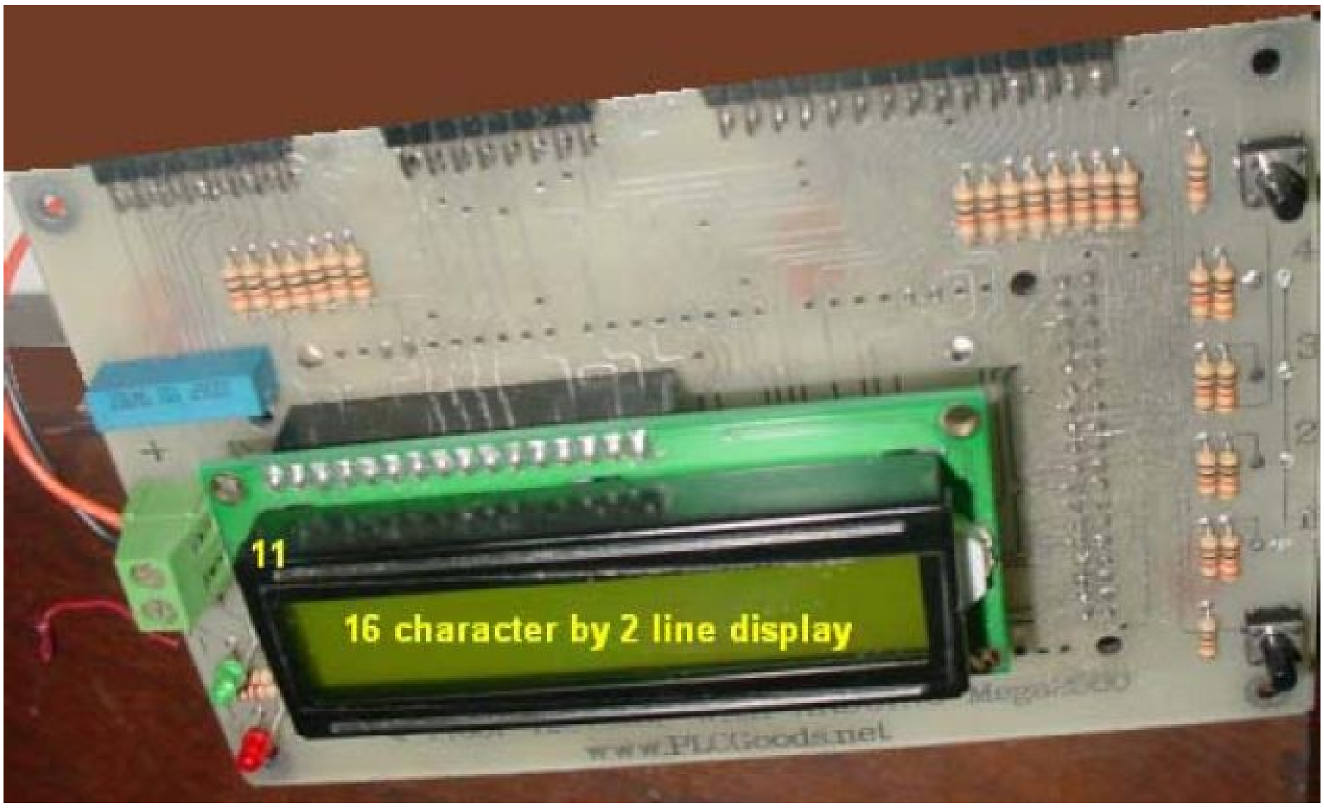
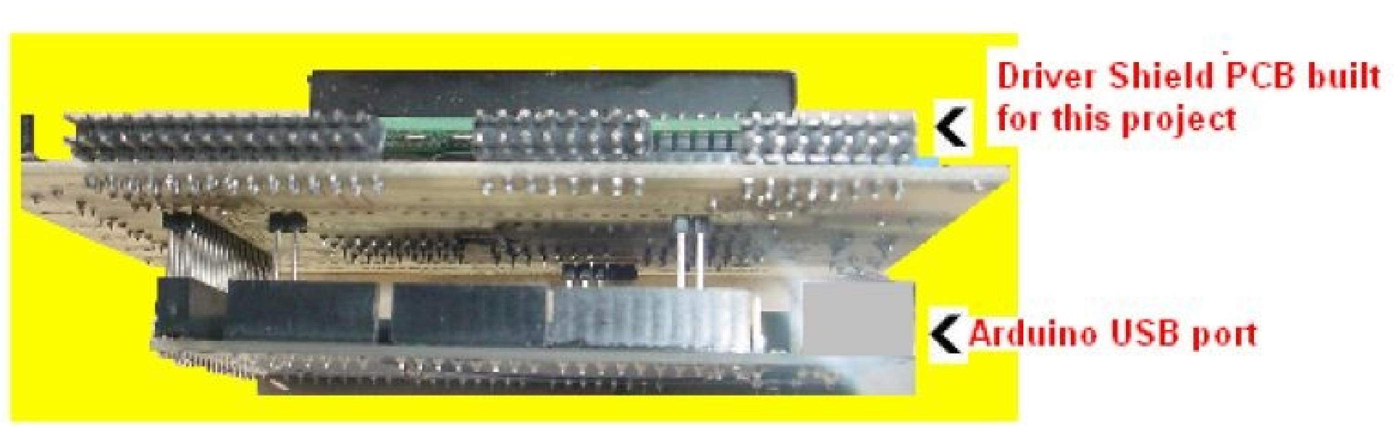


Figure 2-5

In Figure 2-4 notice that the pin headers shown in green rectangle boxes, which are soldered into the solder side of the Driver shield, are the ones that should be connected onto the female connecters of the Arduino board, to make the whole circuit functional.



Step 4:
Building the PCBs of the prototype model lift system

As it was already shown in Figure 2-4, for this Step of implementing the project, only one PCB related to the Arduino shield PCB is needed to be built. So, you may use any PCB layout designing program to do it.

Step 5:

Uploading the control program into the Arduino board

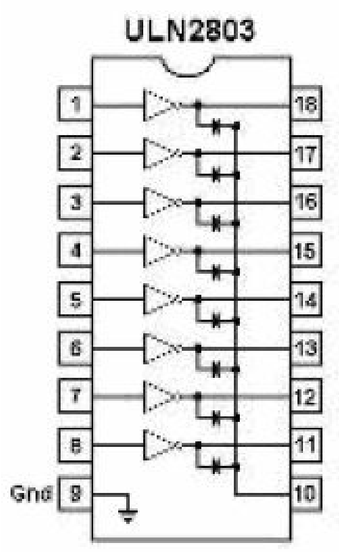
One great thing I like about Arduino board is that you do not need to have a programmer to upload your developed code. If this is the first time you are using an Arduino board for building your project, there is tons of information on the internet to teach you how to do it. You might try this link, for example:
http://www.ladyada.net/learn/arduino/lesson1.html

Appendix A

Description of ULN2803 Chip

Each ULN2803 contains eight Darlington transistors with common emitters and integral suppression diodes for inductive loads. So, you can bring in some muscle to your output pins with 8 mighty Darlingtons! This DIP chip contains 8 drivers that can sink 500mA from a 50V supply, and has kickback diodes included inside for driving the coils. This will let your little microcontroller or microcomputer power up the solenoids, DC motors (in one direction) and unipolar stepper motors.

Note that this is an 'open collector' driver - it can only be used to connect the load to ground and there will be a 1 Volt **drop** across the internal transistors. The inputs can be driven by 3.3V or 5V logic. Outputs may be connected in parallel for higher current capability. Five versions are available to simplify the interfacing to standard logic families.



Overview of the Arduino Mega 2560 board

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), 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 an AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Questions and Answers related to this project

This section of this manual is reserved for you. I would be very happy if you would like write to me with any questions (hardware or software wise) that you have related to this well designed industrial project.

I Will include your question in this section of the manual, as I believe that it might be also somebody else's question, as well.

Thanks for your comments and concerns regarding this design!

Conclusion

Now you have the information you need to become a successful a Microcontroller programmer. This is a moment for you to take the information you've been given and not be afraid to put it to use.

The primary thing that keeps people from becoming successful is fear. If you can overcome this fear then you will succeed and prosper.

All you need to do at this point is to stop putting things off and just do it. Start working and you will find it becomes easier as you go along.

Once you accomplish any new skill or task, you will realize just how easy it is. That is when you want to kick yourself for not starting sooner.

So don't hesitate. Start right now. Start today. Break that cycle of doubt and enjoy your newfound skills and succeed in exploiting them.

To your success and happiness,

Seyedreza

http://youtu.be/f-ZbP8juB20

2(A). Programming a **TRAFFIC LIGHT CONTROL** system using an **AVR** microcontroller

http://youtu.be/q0eWsX2kMeU

3. Simple Arduino based Automated Parking lot System

http://youtu.be/CNt2DIlwrso

3(A). Design and implementation of a Microcontroller-Based Car Parking System

http://youtu.be/7aIp2iCC_mU

4. Implementing an Arduino based temperature controller with PID algorithm
http://youtu.be/i2nxqLs9wBg

4(A). Temperature controller and monitoring with a microcontroller

http://youtu.be/mIyixludYaE

5. Analog LED clock with chime, alarm, calendar and temperature

display (not published yet)

http://youtu.be/Mi5zAdctp4g

Free microcontroller \Arduino based project Downloads from our www.plc-doc.com or www.plcgoods.net websites:

1. How to control a Stepper motor with an ATmega8 microcontroller
   http://youtu.be/XBWCsl512Lc
2. MATLAB/PC and Microcontroller based advanced Line following Robot
   http://youtu.be/2Y_qSzKL_2Y
3. 2 Zones microcontroller based weekly digital timer
   http://youtu.be/xbIEKI3qqNM

Purpose of the Book

The project is about development of a practical microcontroller based 4 floor elevator system. To develop the project, there are three objectives that must be accomplished which are:

1- Development of a 4 floor elevator system controlled by a Microcontroller. In that case, an Atmel AVR ATmega8 Microcontroller and an Arduino board will be used to control the operation of the 4 floor elevator system.

2- Arduino and Codevision softwares are used to program both microcontrollers in C language

3- Application of Proteus software to simulate the hardware.

The purpose behind of developing a manual like this is an attempt to make learning Programming Microcontrollers fun!

Simple Arduino based 4 Floor Elevator Project

Plcgoods
info@plcgoods.com/net
www.plcgoods.com/net
www.plc-doc.com