EMBEDDED SYSTEMS DESIGN
PROJECT REPORT

AHMEDABAD UNIVERSITY

PROJECT REPORT

RFID CARD BASED ATTENDANCE AND A SMART CLASSROOM

Under the guidance of

Prof Anurag Lakhlani

Mentors

Ms. Bhavika Patel

Ms. Kavita Anjariya

Made By

Group 04

Dipen Kanjiya (1410016)

Harsh Dalal (1401022)

Chintan Saraviya (1401026)

Raj Derasari (1401029)

Index/Table of Contents

1. Introduction

Attendance can be taken by doing roll number calling or by passing attendance sheet. Both these techniques are manual, and have their own issues. Often, we have also seen that there are no people in the class but some lights or fans, or projector systems, may be left on because there is no one to maintain these. This has an environmental corner to it, we should be trying to avoid usage of electricity wherever possible.

2. Project Background

2.1 Project Idea:

We try to make the attendance system easy, automated, and quick, keeping in mind the price of the end product as well. If there are no entities in the classroom, then the system should automatically turn everything off. Provide a better, effortless, faster alternative for marking students' presence, and while it may not be perfect, it should be economic.

2.2 Our outcome/application:

To read RFID tags assigned to entities, and using RFID module interfaced with the microcontroller, keep track of each entity. Hence the application RFID-based attendance. In addition, by keeping track of the entities in the classroom, maintain the electronic systems of the classroom. Whenever the card of the professor is detected to be leaving the classroom, the attendance record of the students is written into the microcontroller EEPROM.

2.3 Market:

We have observed that in the market, there are a few areas; such as government offices, some campuses, and private company workplaces; where automated systems for attendance marking are implemented, which are either fingerprint based systems or id-card based systems.

3. Block Diagram

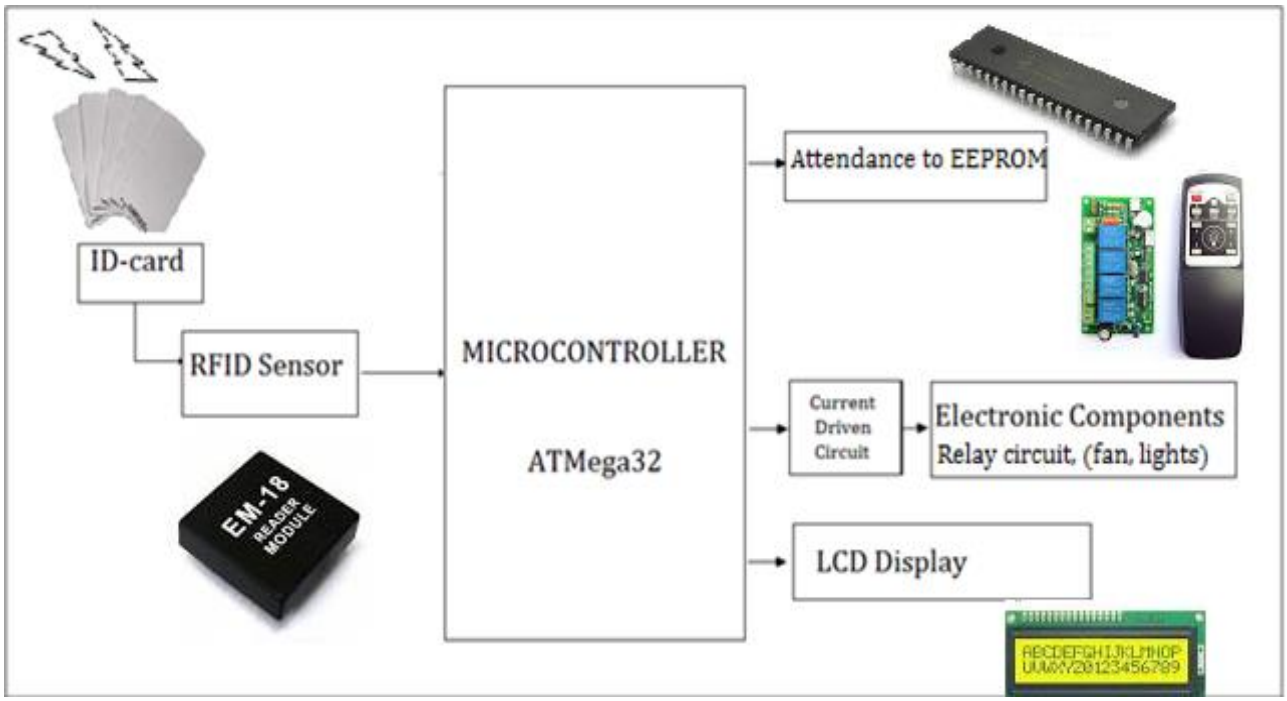

Figure 3.1 : Block Diagram Of Project

4. Peripherals used, selection criteria

4.1 Selection of RFID module

Based on findings, here are the RFID modules that we select from:

Property / Module	EM-18	NSKEDK125TTL	RHY1320 LFR	Robokit USB2.0
Read frequency	125 KHz	125 KHz	125 KHz	125 KHz
Range of reading	10 cm	10 cm	15 cm	20 cm
Interfacing	UART 48-bit	UART 64-bit	UART 64-bit	Only with PC
Operating V, I	5V, 50 mA	5V, 50mA	5V, 50mA	5V, 50mA
Indication	LED, Buzzer	LED, Buzzer	LED, Buzzer	LED, Buzzer
Cost	Rs. 500	Rs. 700	Rs. 990	Rs. 1000

5. Table of components

Component	Quantity	Total Cost
ATMega32	1	Rs. 500
RFID Module	2	Rs. 1000
RFID Cards	8+3 incl. with RFID	Rs. 240 (8 cards)
LCD Display	1	Rs. 150
LED Indicators	5	Rs. 10
Connectors	30F-F,15M-F,15M-M	Rs. 350
Fan + Relay	1	Rs. 600
Total	-	Rs. 2800

6. Circuit diagram of modules

6.1 Circuit Diagram of ATMega32

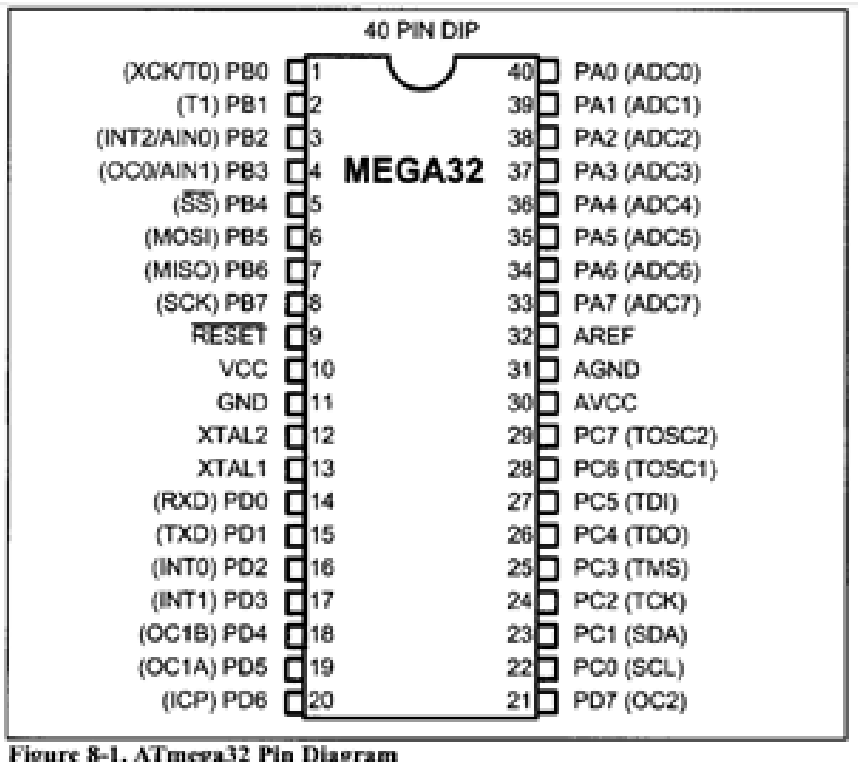

Figure 8-1. ATmega32 Pin Diagram

Figure 6.1 Circuit connection of ATMega32

6.2 Circuit Diagram of EM-18

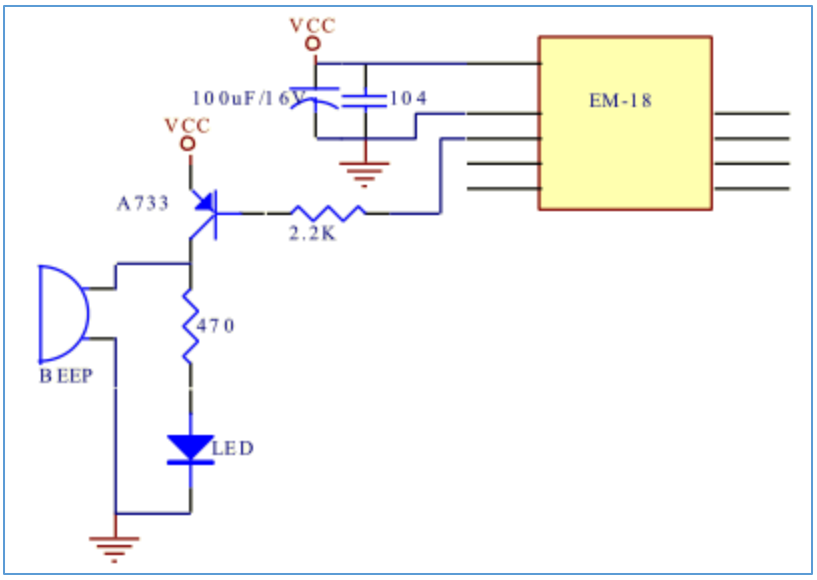

Figure 6.2.1 Circuit connection of EM-18

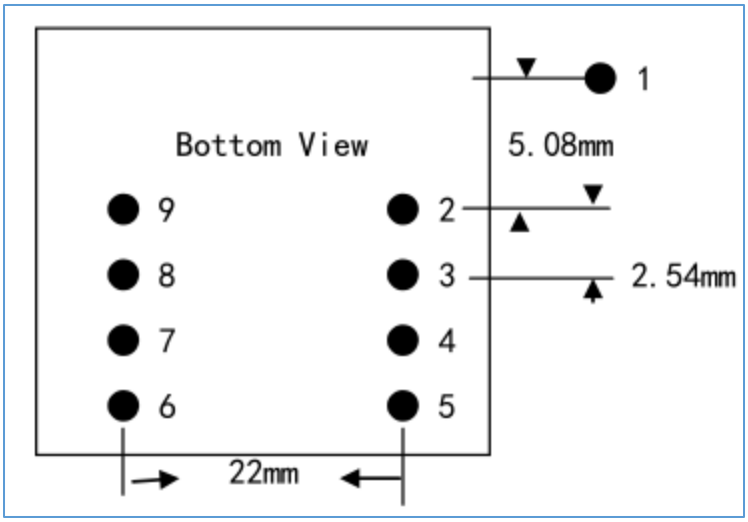

Figure 6.2.2 Pin connections of EM-18

6.3 Final Circuit Diagram Implemented

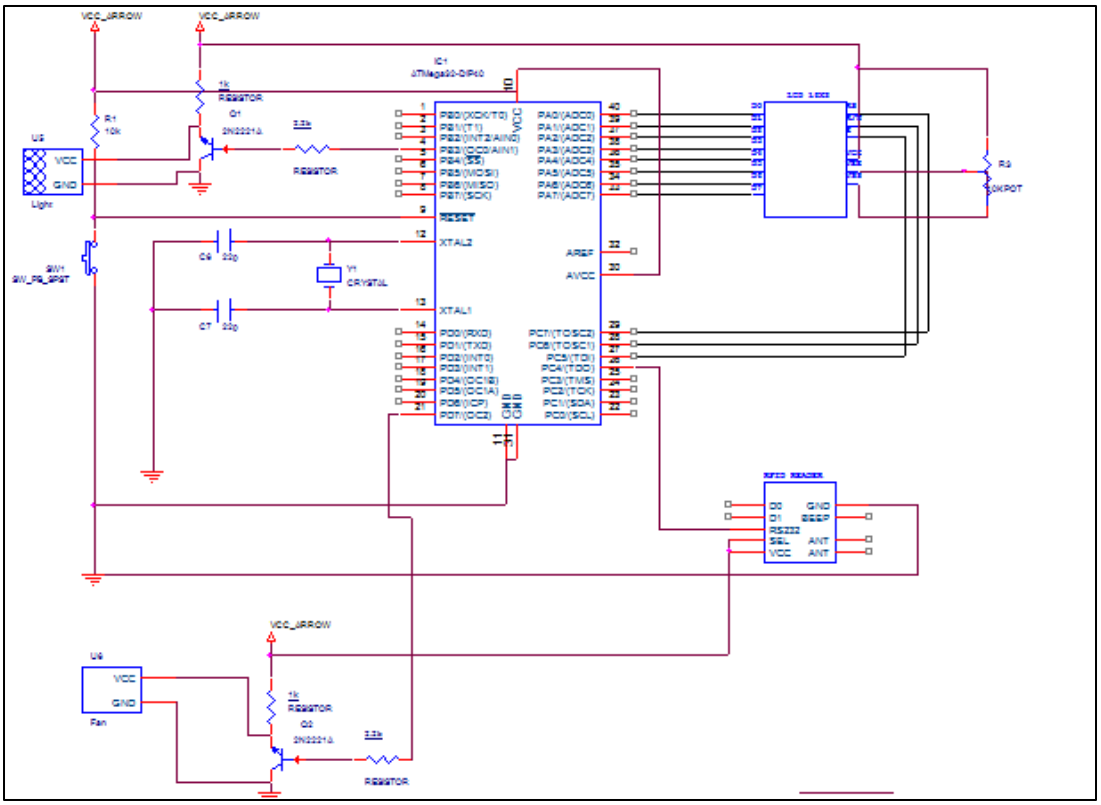

Figure 6.3 Final Circuit Diagram

7. Required debugging

7.1 Issues with EM-18:

We observed that if the card is put in the range of RFID but removed very quickly, some bad data is stored in the UDR of ATmega32 and hence the entire flowchart followed gets spoiled.

We tackled this issue by doing 2 steps before accepting reading of RFID:

1. We enabled USART in transmission mode and set interrupt enabled.
2. We transmitted 0x00 through USART and this essentially reset the UDR and we did not have to worry about bad data being stored in the UDR on reading of data.

7.2 Issues with relay circuit

So, we connected lights and fans using a relay circuit to a breadboard, which had VCC and GND connected to ATMega32 pins. But we observed that the voltage being supplied to the relay circuit from the breadboard was 4.5V instead of 5V. Because of this the relay circuit did not turn on and was not able to supply any voltage to the fan or light. So for testing purposes we kept only LEDs connected to our circuit in replacement of the relay circuit.

8. Snapshots of implementation

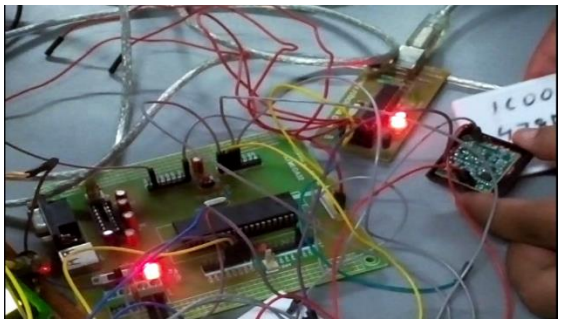

Figure 8.1: RFID card being read by module

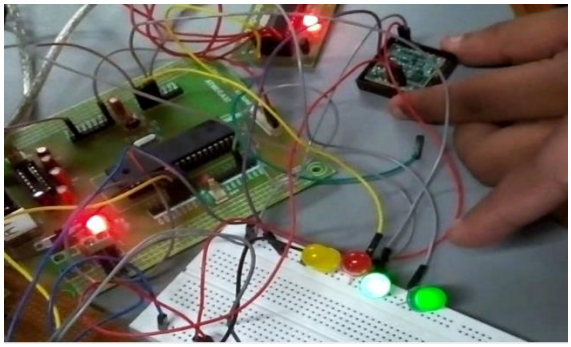

Figure 8.2: When the number of people in the class has increased, the LEDs indicating connection to light circuits have turned on

Figure 8.3: Initial, waiting for RFID

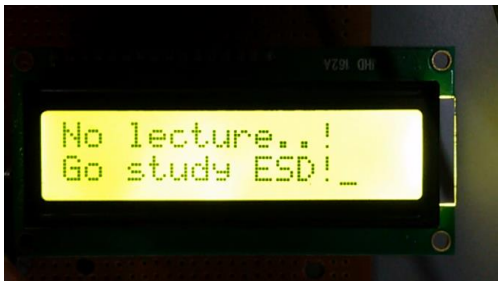

Figure 8.4: We swipe a student card (without any professor card swiped priory) so it denies entry and shows that no lecture is going on

Figure 8.5: We swiped the professor card which we matched ourselves to a subject “ESD”, so it shows that “ESD” lecture has started (that is, the professor of “ESD” had entered the classroom)

9. Flowchart

The final flowchart implementation had minor differences with the flowchart we determined by the submission of assignment-3.

Designed flowchart is as follows:

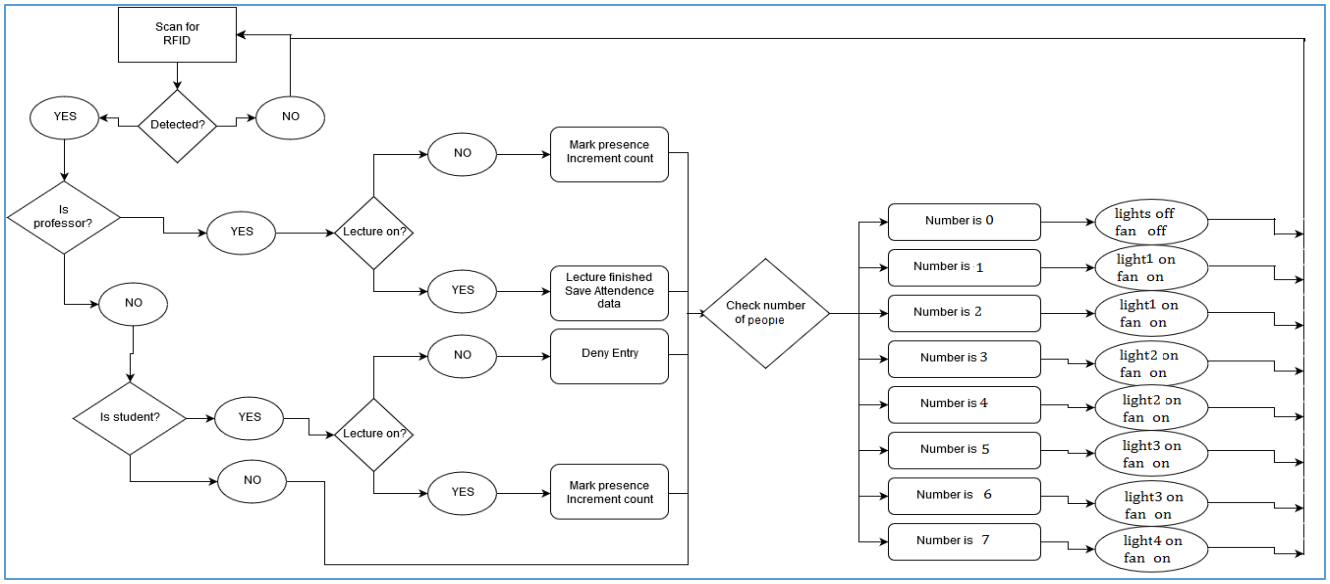

Figure 9.1: Flowchart model (final implementation)

10. Project code:

// 10.0 Preprocessor directive and variables

// 1. Preprocessor directives that we have used:

```
#define LCD_RS      PORTA.B2
#define LCD_EN      PORTD.B6
#define LCD_D4      PORTC.B4
#define LCD_D5      PORTC.B5
#define LCD_D6      PORTC.B6
#define LCD_D7      PORTC.B7
```

// 2. Usage of naming variables: The following variables are used, can be changed

```
char* name_of_lecture_1 = "ESD";
char* name_of_lecture_2 = "ADC";
char* name_of_lecture_3 = "PRP";
char* name_of_student_1="Student 1";
char* name_of_student_2="Student 2";
char* name_of_student_3="Student 3";
char* name_of_student_4="Student 4";
char* name_of_student_5="Student 5";
char* name_of_student_6="Student 6";
char* name_of_student_7="Student 7";
char* name_of_student_8="Student 8";
```

// 3. Stored RFID data

```
char* professor_1="19004B22A8D8";
char* professor_2="19004AB430D7";
char* professor_3="34001226A3A3";
char* student_1 = "1C007926FFBC";
char* student_2 = "1C0079AC478E";
char* student_3 = "1D003D853E9B";
char* student_4 = "1D003D78237B";
char* student_5 = "1D003D7B4A11";
char* student_6 = "1D003AD0F90E";
char* student_7 = "1D003D4FFB94";
char* student_8 = "1D003D050E2B";
```

// 4. Arrays given below store the attendance (0/1) of student/professor

```
char student_is_present[]  = {0,0,0,0,0,0,0,0};
char professor_is_present[]= {0,0,0,0};
unsigned char people_count=0;                   //for switching
unsigned char prof_memory_address=0;            //for EEPROM
unsigned char student_memory_address=1;         //for EEPROM
```

// 10.1 LCD Module

```
void LCD_data(unsigned char Data)
{
PORTC=Data&0xF0;
LCD_RS=1;LCD_EN=1;delay_us(5);LCD_EN=0;
PORTC=((Data<<4)&0xF0);
LCD_EN=1;delay_us(5);LCD_EN=0;delay_us(100);
}

void LCD_Print(char * str)
{
unsigned char i=0;
while(*(str+i)!=0)
    {
      LCD_data(*(str+i));
      i++;
      delay_ms(10);
    }
}

void lcdcommand(unsigned char command)
{
PORTC=command&0xF0;
LCD_RS=0;LCD_EN=1;delay_us(5);LCD_EN=0;delay_us(100);
PORTC=((command<<4)&0xF0);
LCD_EN=1;delay_us(5);LCD_EN=0;delay_us(40);
}

void Cursor_Position(unsigned short int x,unsigned short int y)
{
unsigned char firstcharadd[] = {0x80,0xC0};
lcdcommand( (firstcharadd[x-1]+y-1));
}

void clear()
{
lcdcommand(0x01); delay_ms(2);
}

void LCD_Initialize()
{
LCD_EN=0;
lcdcommand(0x33);lcdcommand(0x32);lcdcommand(0x28);
lcdcommand(0x0E);clear();
lcdcommand(0x06);lcdcommand(0x80);
}
```

// 10.2 USART and EEPROM Modules

```
void usart_init()

{
UBRRH = 0;UBRRL = 51;
UCSRB |= (1<<RXEN)|(1<<TXEN);
UCSRC |= (1 << URSEL)|(3<<UCSZ0);
}

void EEPROM_write_data(unsigned char address, unsigned char datanew)
{
while(EECR & (1<<EEWE));
EEARH = 0;EEARL = address;
EEDR = datanew;
EECR |= 1<<eemwe;
EECR |= 1<<eewe;
}
```

// 10.3 EM-18 Module data

```
// 1. Comparing professor RFID tags

char return_professor_id(char* rfid_received_data)
{
  char i;
  for(i=0; i<12; i++)                    //if matches prof 1, return 1
  {
    if(*(rfid_received_data+i) != professor_1[i]) break;
    else if(i==11 && *(rfid_received_data+11) == professor_1[i]) return 1;
  }
  for(i=0; i<12; i++)                    //if matches prof 2, return 2
  {
    if(*(rfid_received_data+i) != professor_2[i]) break;
    else if(i==11 && *(rfid_received_data+11) == professor_2[i]) return 2;
  }
  for(i=0; i<12; i++)                    //if matches prof 3, return 3
  {
    if(*(rfid_received_data+i) != professor_3[i]) break;
    else if(i==11 && *(rfid_received_data+11) == professor_3[i]) return 3;
  }
return 0;                                //default, returns 0
}

//2. Comparing student RFID tags

char return_student_id(char* rfid_received_data)
{
    char i;                              // counter for the loop to compare RFIDs
    for(i=0; i<12; i++)                  //compare with student 1, return 1 if match
    {
      if(*(rfid_received_data+i) != student_1[i]) break;
      else if(i==11 && *(rfid_received_data+11) == student_1[i]) return 1;
    }
    for(i=0; i<12; i++)                  //compare with student 2, return 2 if match
    {
      if(*(rfid_received_data+i) != student_2[i]) break;
      else if(i==11 && *(rfid_received_data+11) == student_2[i]) return 2;
    }

  return 0;

}
```

// 10.4 Other functions used

//These functions were made by us to make the code as modular as possible. This means that the code can be modified to suit any classroom of any semester with any number of students and professors.

// 1. This function executes when professor enters class

```
void professor_has_entered(char professor_id)
{
  professor_is_present[professor_id]=1;
  people_count++;
  clear();
  Cursor_Position(1,1);  LCD_Print("Hello Professor");
  Cursor_Position(2,1);
    if(professor_id == 1)          LCD_Print(name_of_lecture_1);
    else if(professor_id == 2)    LCD_Print(name_of_lecture_2);
    else if(professor_id == 3)    LCD_Print(name_of_lecture_3);
  LCD_Print(" Has Started");
  delay_ms(1000);
}
```

// 2. this function checks if no lecture is going on, returns 1 if no other lecture is going on and returns 0 if some lecture is going on

```
char check_if_no_other_lecture(char professor_id)
{
  if(   (professor_id == 1 && (professor_is_present[2] == 1 ||
           professor_is_present[3] == 1))
     ||
        (professor_id == 2 && (professor_is_present[1] == 1 ||
           professor_is_present[3] == 1))
     ||
        (professor_id == 3 && (professor_is_present[1] == 1 ||
           professor_is_present[2] == 1)))
   {
     clear();
     Cursor_Position(1,1);
     LCD_Print("Class Is On.");
     Cursor_Position(2,1);
     LCD_Print("Please Wait.");
     delay_ms(1000);
     return 0;
  }
  return 1;
}
```

//3. Function that executes on professor exiting and writes to EEPROM

```
void professor_has_left(char professor_id)          //when professor exits
{
    professor_is_present[professor_id]=0;
    people_count--;
    clear();
    Cursor_Position(1,1);   LCD_Print("Bye Professor");
    Cursor_Position(2,1);
            if(professor_id == 1) LCD_Print(name_of_lecture_1);
       else if(professor_id == 2) LCD_Print(name_of_lecture_2);
       else if(professor_id == 3) LCD_Print(name_of_lecture_3);
    LCD_Print(" Has Finished");
    delay_ms(1000);

     EEPROM_write_data(prof_memory_address,professor_id);
     EEPROM_write_data(student_memory_address++,student_is_present[1]);
     EEPROM_write_data(student_memory_address++,student_is_present[2]);
     EEPROM_write_data(student_memory_address++,student_is_present[3]);
     EEPROM_write_data(student_memory_address++,student_is_present[4]);
     EEPROM_write_data(student_memory_address++,student_is_present[5]);
     EEPROM_write_data(student_memory_address++,student_is_present[6]);
     EEPROM_write_data(student_memory_address++,student_is_present[7]);
     EEPROM_write_data(student_memory_address++,student_is_present[8]);
     prof_memory_address = student_memory_address + 1;
     student_memory_address=student_memory_address+2;
     clear();
     Cursor_Position(1,1);      LCD_Print("Attendance data");
     Cursor_Position(2,1);      LCD_Print("written to rom");
     delay_ms(1000);
     }
```

//4. Executes when student "student_id" enters the class

```
void student_has_entered(char student_id)
{
  student_is_present[student_id] = 1;
  people_count++;
  clear();
  Cursor_Position(1,1);
  if(student_id == 1) LCD_Print(name_of_student_1);
  else if(student_id == 2) LCD_Print(name_of_student_2);
  else if(student_id == 3) LCD_Print(name_of_student_3);
  else if(student_id == 4) LCD_Print(name_of_student_4);
```

```
    else if(student_id == 5) LCD_Print(name_of_student_5);
    else if(student_id == 6) LCD_Print(name_of_student_6);
    else if(student_id == 7) LCD_Print(name_of_student_7);
    else if(student_id == 8) LCD_Print(name_of_student_8);
    Cursor_Position(2,1);
    LCD_Print("has entered");
    delay_ms(1000);
  }
```

//5. This executes when student "student_id" leaves the class

```
  void student_has_left(char student_id)
  {
    student_is_present[student_id] = 0;
    people_count--;
    clear();
    Cursor_Position(1,1);   LCD_Print(" ");
    if(student_id == 1) LCD_Print(name_of_student_1);
    else if(student_id == 2) LCD_Print(name_of_student_2);
    else if(student_id == 3) LCD_Print(name_of_student_3);
    else if(student_id == 4) LCD_Print(name_of_student_4);
    else if(student_id == 5) LCD_Print(name_of_student_5);
    else if(student_id == 6) LCD_Print(name_of_student_6);
    else if(student_id == 7) LCD_Print(name_of_student_7);
    else if(student_id == 8) LCD_Print(name_of_student_8);
    Cursor_Position(2,1);   LCD_Print("has left");
    delay_ms(1000);
  }
```

//6. This executes if a student tries to enter but there is no lecture

```
  void no_lecture_going_on()
  {
    clear();
    Cursor_Position(1,1);   LCD_Print("No lecture..!");
    Cursor_Position(2,1);   LCD_Print("Go study ESD!");
    delay_ms(1000);
  }
```

//7. This executes when RFID detected does not match any student

```
  void not_a_student()
  {
    clear();
    Cursor_Position(1,1);   LCD_Print("Unidentified.");
    Cursor_Position(2,1);   LCD_Print("..card detected!");
    delay_ms(1000);
  }
```

// 10.5 Main Function

```
// main function which is the implementation of our flowchart

void main()
{
char rfid_data[13];                    //array to hold received RFID data
char i;                                //used in loop for receiving RFID
char get_professor_id;                 //returns number based on RFID data
char get_student_number;               //returns number based on RFID data
rfid_data[12] = '\0';                  //set the last bit as ending character
DDRD.B0=0;                             //input for USART receive
DDRB.B7=1;DDRB.B6=1;DDRB.B5=1;DDRB.B4=1;//light ports
DDRC=0xF0;                             //for LCD
DDRA.B2=1;                             //for LCD
DDRD.B6=1;                             //for LCD
DDRD.B4=1;PORTD.B4=0;                  //set fan port and initialize as 0
LCD_initialize(); clear();             //start LCD
usart_init();                          //set USART
while(1)
{
    clear();                                    //Clears the LCD
    Cursor_Position(1,1);
    LCD_Print(" Show RFID Tag");                //Print msg on LCD
    UCSRB|=(1 << UDRIE);                        //enable data empty interrupt
    UCSRB|=(1<<TXCIE);                          //enable USART transmit
    UDR=0x00;                                   //reset UDR
    usart_send(0x00);                           //reset UDR
    UCSRB=~(1<<TXCIE);                          //disable transmit
    UCSRB|=(1 << UDRIE);                        //enable data empty interrupt
    UCSRB|=(1 << RXCIE);                        //enable receive interrupt

    for(i=0; i<12; i++)
    {
      while(UCSRA.B7 == 0);                     //wait for RXC flag
      rfid_data[i] = UDR;                       //set bit of array
    }
    UCSRB=~(1 << RXCIE);                        //disable receive
    UCSRB=~(1 << UDRIE);                        //disable read

     get_professor_id = return_professor_id(rfid_data);          //get ID
     get_student_number = return_student_id(rfid_data);          //get ID

     if(get_professor_id!=0)          //means a professor was detected
     {
          if(check_if_no_other_lecture(get_professor_id) == 1)
```

```
            {
                if(professor_is_present[get_professor_id] == 0)
                    professor_has_entered(get_professor_id);
                else professor_has_left(get_professor_id);
            }
        }
        else
        {
            if(get_student_number == -1) not_a_student();
            else
            {
                if(student_is_present[get_student_number] == 1)
                    student_has_left(get_student_number);
                else
                {
                    if(professor_is_present[1] == 1 || professor_is_present[2] == 1
                                                    || professor_is_present[3]== 1)
                        student_has_entered(get_student_number);
                    else
                        no_lecture_going_on();
                 }
             }
         }
switch(people_count)                    //B7-B4 are lights, D4 is fan
{
   case 0: PORTB.B7=0;PORTB.B6=0;PORTB.B5=0;PORTB.B4=0;  PORTD.B4 = 0;
   case 1: PORTB.B7=0;PORTB.B6=0;PORTB.B5=0;PORTB.B4=1;  PORTD.B4 = 1;
   case 2: PORTB.B7=0;PORTB.B6=0;PORTB.B5=0;PORTB.B4=1;  PORTD.B4 = 1;
   case 3: PORTB.B7=0;PORTB.B6=0;PORTB.B5=1;PORTB.B4=1;  PORTD.B4 = 1;
   case 4: PORTB.B7=0;PORTB.B6=0;PORTB.B5=1;PORTB.B4=1;  PORTD.B4 = 1;
   case 5: PORTB.B7=0;PORTB.B6=1;PORTB.B5=1;PORTB.B4=1;  PORTD.B4 = 1;
   case 6: PORTB.B7=0;PORTB.B6=1;PORTB.B5=1;PORTB.B4=1;  PORTD.B4 = 1;
   case 7: PORTB.B7=1;PORTB.B6=1;PORTB.B5=1;PORTB.B4=1;  PORTD.B4 = 1;
   case 8: PORTB.B7=1;PORTB.B6=1;PORTB.B5=1;PORTB.B4=1;  PORTD.B4 = 1;
}
delay_ms(10);
clear();
}
```

11. Conclusions

- So, we had various learnings from the project.
  - We learnt how to implement our theoretical learning in the practical domain
  - We learnt how we simultaneously work with hardware and software, with required debugging
  - Which makes us learn that debugging of some parts of embedded hardware is tough, and requires thinking outside the box to debug
- Automation of tasks will be one of the prime objectives in the future. Everywhere we will be seeing good and bad implementations of embedded systems.
- Design drawbacks will always be seen (for eg. we had to change from relay based lights to LED lights because of relay circuit voltage requirement) and we sometimes need to change the implementation to deal with these.
- Learning:
  - Keeping faith in ourselves and in our software!
  - How to solder PCB with hardware peripherals like LCD.
  - Designing feasible embedded systems which fulfil all requirements
  - Documenting work done
  - Version control of software
  - Debugging processes.
  - Minimize wiring requirements and providing ease of use to the end user

12. Timeline

We have followed the following timeline

Task	Date	4/4/16	11/4/16	18/4/16	25/4/16
Implement algorithm in code		X			
Testing of code on chip		X	X		
Early implementation, bug fixing			X	X	
Final implementation on chip				X	X

13. References

[1.] EM-18 RFID datasheet:
www.tomsonelectronics.com/uploads/1430561217EM-18-RFID-Reader.pdf

[2.] LCD 16x2 datasheet:
www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet/

[3.] ATMega32 datasheet, which was part of Embedded Systems Design lab documents

[4.] The AVR Microcontroller and Embedded System using assembly and C, Mazidi, Naimi, and Naimi. (Embedded Systems Design course, textbook)