Smart Room

Group Number – 29

Mihir Gajjar - 1401076

Vidit Shah - 1401078

Karan Patel - 1401113

Under the guidance of:

Course Instructor: Mr. Anurag Lakhlani

Mentored by: Bhavika Patel, Kavita Anjaria

IET



INSTITUTE OF  
ENGINEERING &  
TECHNOLOGY



**INDEX**

Introduction…………………………………………………...…... 3

Block Diagram……………………………………………………. 4

Selection Criteria for major Components………………………… 5

Costing Table……………………………………………………... 7

Circuit diagrams…………………………………………………... 8

Examples of debugging and troubleshooting…………….………. 10

Snapshots of your working model………………………………... 11

Flow chart………………………………………………………... 12

Code……………………………………………………………..... 15

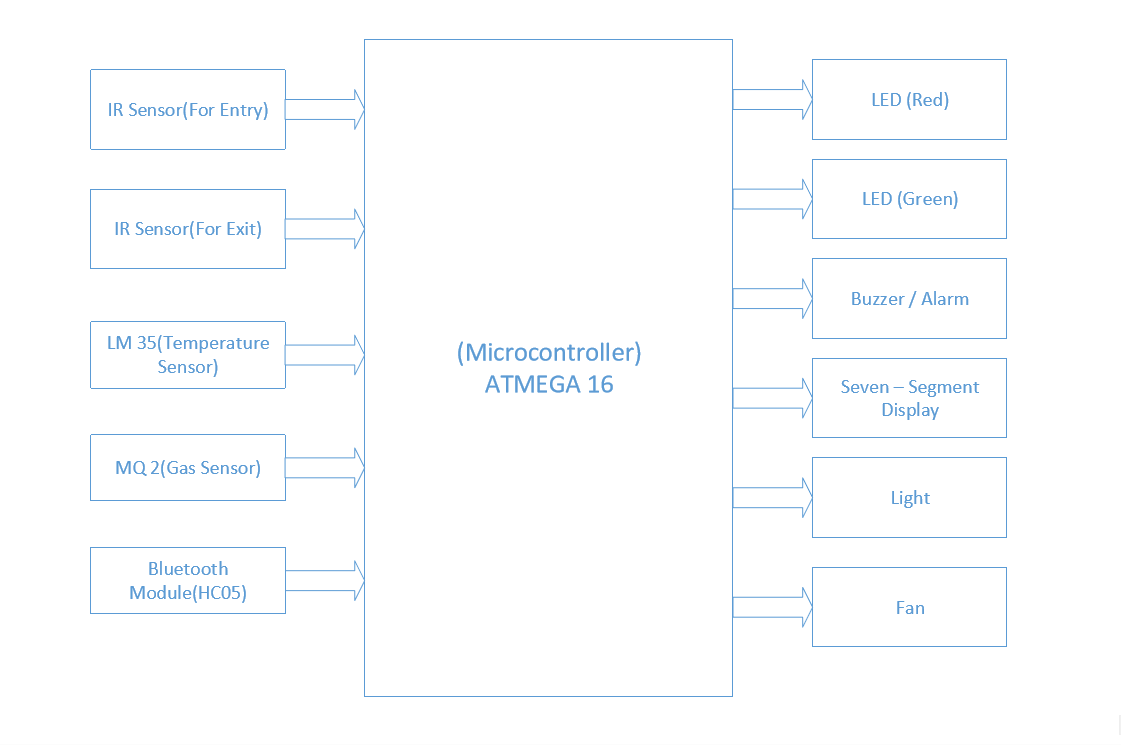
Conclusions……………………………………………………….. 27

Project Timeline…………………………………………………... 28

References……………………………………………………….... 30

**Introduction**

Nowadays computing technology research is focused on the development of Smart Environments. Many of the things are getting automated which led us to an idea for developing a smart room. Conserving the electricity is a major problem nowadays. In our smart room there are sensors which counts the number of people currently in the room. If there are non-zero person(s) in the room, then light will be turned on. If the count of the person in the room is zero, then the electricity supply will be stopped which will help in conserving the electricity. If the number of persons reaches the maximum limit say nine, then an LED will be turned on indicating that the limit for the persons in the room has reached and no more person should enter the room. There is password authentication if one wants to enter the room. The user has to enter the password via Bluetooth. If the entered password is right, then green LED will be turned on for ten seconds. The entry is allowed for 10 seconds after the password entered is correct and then the IR sensor will not detect the entry. If the user enters the password wrong for three times, then error will occur. There will be a temperature sensor in the room, if the temperature rises above a certain temperature (for eg. 49 °C) then the fire alarm will be turned on. Also, gas leakage has been taken care of. There is a gas sensor which detects the leakage of gas. If gas leakage is detected, then the gas alarm will be turned on.

**Block Diagram**

**Selection Criteria for major components**

1. Gas Sensor

|  |  |  |  |
| --- | --- | --- | --- |
| Sensor | MQ2 | MQ6 | MQ4 |
| Price | Rs.150 | Rs.383 | $4.96 = Rs.330.72 |
| Gases Detected | Methane, Butane, LPG, smoke | Butane, CNG | Methane, CNG |

The sensor we have selected is MQ2.The major factors being:

* Low Price.
* Detects the gas for LPG (i.e. It satisfies our purpose of detecting the gas leakage).
* Wide detecting scope.
* Stable and long life, Fast response and High sensitivity.

2.Temperature Sensor

|  |  |  |  |
| --- | --- | --- | --- |
| Sensor | LM35 | RTD | LM20 |
| Price | Rs.80 | Rs.995 | $ 0.32 = Rs.21.34 |
| Temp. Range | -55 to 150 °C | Up to 450°C | -55 to 120 °C |
| Accuracy | ±0.5°C (at 25°C) | + /- 0.3°C at 0°C | ±1.5 to ±4°C at 30°C |

The sensor we have selected is LM35. The major factors being:

* Cheaper than RTD.
* Accurate then LM20
* Satisfies the required temperature range.

3. Entry / Exit Sensor

|  |  |  |  |
| --- | --- | --- | --- |
| Sensor | •Adjustable Infrared Sensor | Sharp GP2Y0A02YK0F | Proximity sensor |
| Power supply | 5 V | 2.8 V | 4 V |
| Price | Rs. 650 | Rs. 997 | Rs. 79 |
| Range | 3-80 cm | 15 cm | 3-12 cm |

The sensor we have selected is Proximity. The major factors being:

* Relatively Cheaper.
* The detection range is enough for our purpose.

4.Bluetooth Module

HC 05:

* HC05 can use as master and slave mode.
* The HC-05 module can build a connection to other modules. E.g. a Robot being a master and connecting to slave Bluetooth module. Or in slave mode to make a wireless bridge to a notebook.

HC06:

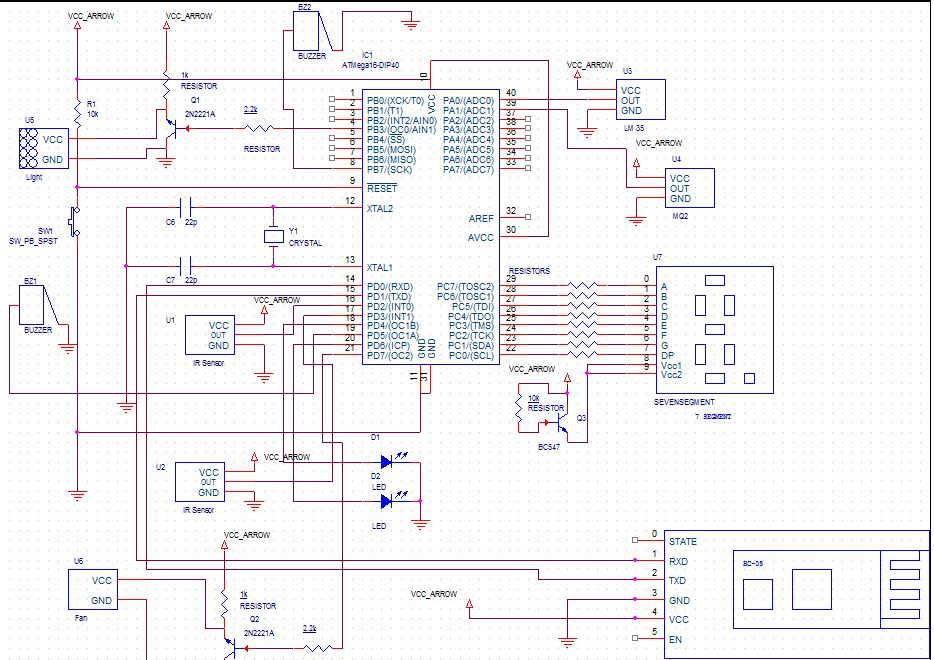
* HC06 can only use as slave mode.
* The HC-06 module only can be a slave. This makes it only useful for say connecting a notebook as a master to a robot with a slave module e.g. for a wireless serial bridge.

As we had HC – 05 module easily available we chose HC – 05.

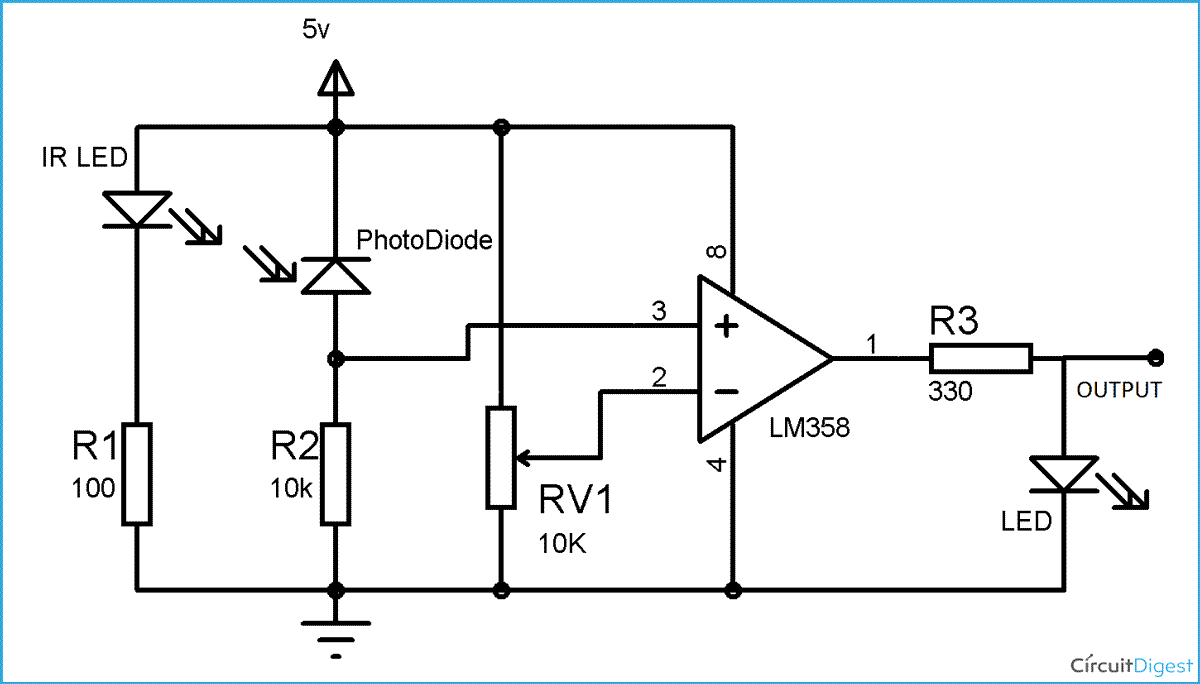
**Costing Table**

|  |  |  |
| --- | --- | --- |
| Name | Quantity | Total Cost  (Rs.) |
| Development kit | 1 | 400 |
| Atmega 32 | 1 | 120 |
| IR Sensor | 2 | 80 |
| HC-05 | 1 | 270 |
| MQ-2 | 1 | 150 |
| LM-35 | 1 | 50 |
| Relay Board | 1 | 100 |
| Battery | 1 | 15 |
| Motor | 1 | 20 |
| Buzzer | 2 | 20 |
| Fan | 1 | 10 |
| Led | 7 | 10 |
| Jumper Wires | 50 | 150 |
| Seven Segment | 3 | 15 |
| Total |  | 1410 |

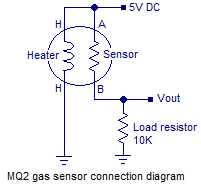
**Circuit diagrams of modules and major circuitry**

**Circuit Diagram:**

**IR Sensor:**



**MQ – 2:**



**Example of debugging and troubleshooting**

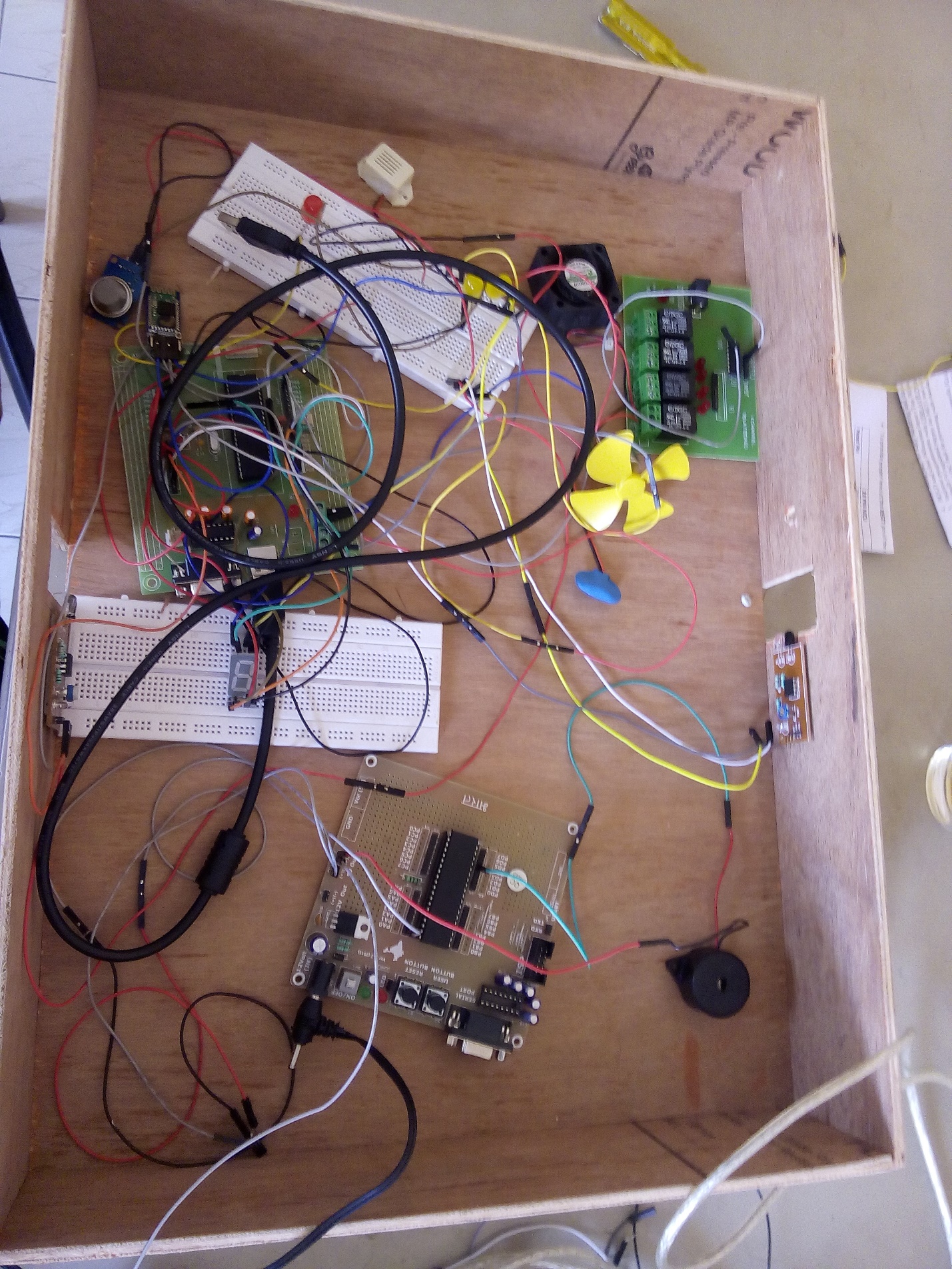
* The HC-05 Bluetooth module and other components such as IR sensor, MQ-2 sensor, LM-35 sensor worked perfectly individually. But when we tried to put together all the components then we were not able to connect with the HC-05 Bluetooth module. As we were not able to connect we were not able to enter the password. We had backup of the code of the HC-05(individually working) so we tried that again and HC-05 worked. But again we tried and it didn’t work with all other components. Then we supplied Voltage (Vcc) to the HC-05 from other source (not from the development board). Then the HC-05 wouldn’t function as programmed. In the app we received back whatever we sent to the HC-05. Hence we supplied Voltage to the HC-05 from that development board only and for other components we supplied voltage through different source. And then all worked fine. The reason that HC-05 wouldn’t work with all the components getting supply from the same Voltage source was that as the voltage got divided between HC-05 and other components the HC-05 was not getting the required voltage and hence we were not able to connect to it. After HC-05 was the only chip getting Vcc from the Development board and all other components getting Voltage from other source HC-05 got the required voltage and it started functioning normally.
* PORTC of the development board was not working for quite a while during the starting of the project. Some of the pins worked but some did not work and this was the case with many of the other groups as well. Then we found that we have to disable the JTAG first as it is enabled with the new AT mega chips by default. In the main function we have to include the below two lines:

MCUCSR |=(1<<JTD);

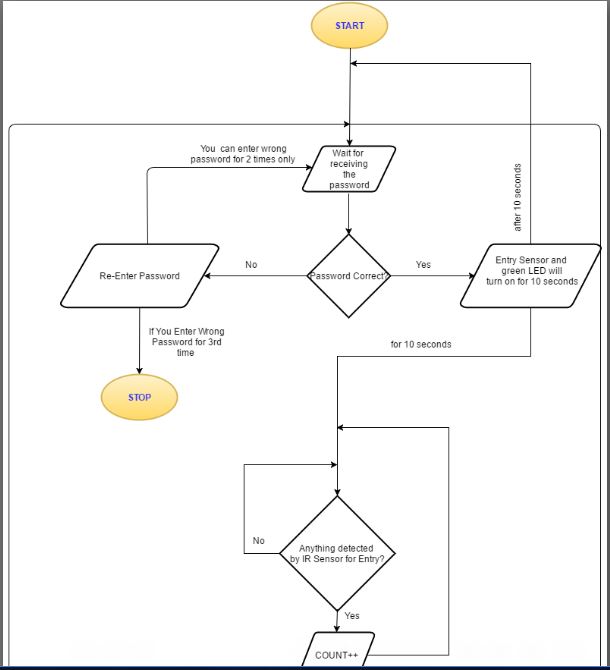
MCUCSR |=(1<<JTD);

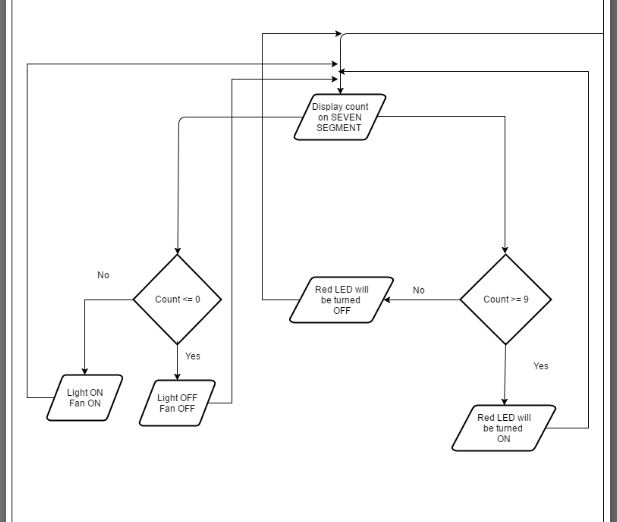
Yes, a bit strange but we have to write it twice. After writing the above two lines twice in the main function, PORTC started working as expected.

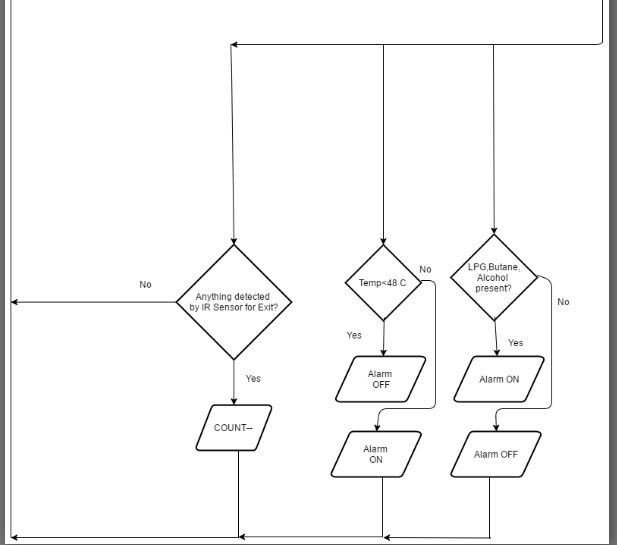
**Snapshots of your working model**

****

**Flow chart**

****

****

****

**Code**

/\*

SMART ROOM

Group 29

Mihir Gajjar (1401076)

Vidit Shah (1401078)

Karan Patel (1401113)

\*/

//Declaring Global Variables.

int SevenSegment\_Anode[]= {0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90}; //An array to store the values in hex which displays the numbers (0 - 9) on the seven segment.

char temperature = 0; //to store the temperature.

unsigned short int store[4]; //Storing hex values in an array for TIMER0\_COMP.

unsigned short int shift=1; //Used for TIMER0\_COMP.

unsigned char ch[100]; //for storing the received value from USART.

int temp,count=0; //Initializing temp and count to zero.

int number\_of\_persons; //For storing the number of persons in the room.

// For multiplexing, data and selection lines are changed at very small time intervals. In this case 8 msec.

void TIMER0\_COMP() org 0x014

{

PORTA=0x00; //PORTA cleared

PORTA=shift; //Select data selection line

PORTC=store[count]; //Assign data to PORTC i.e. seven segment

shift=shift<<1; //Left shift 'shift' till it reaches 0x80

if(shift==0x10)

{

shift=1; //Initialize once 0x80 is reached

}

count++; //Increment count till it reaches 4

if (count==4)

{

count=0; //Initialize once it reaches 4

}

}

void interrupt0() org 0x02 //The address of interrupt 0 is 0x02.Int0 is used for Entry Sensor in the room.

{

if(PORTB.B5 != 1) //If the maximum limit for the persons in the room i.e. 9 has not reached then we will increment the number\_of\_persons.

number\_of\_persons++; //Incrementing the number\_of\_persons.

if(number\_of\_persons >= 9) //The number\_of\_persons cant exceed 9 and hence if more than 9 person enters in the room then number\_of\_persons will remain 9 only.

{

number\_of\_persons = 9;

}

}

void interrupt1() org 0x04 //The address of interrupt 0 is 0x04.Int1 is used for Exit Sensor in the room.

{

number\_of\_persons--; //If someone exits the room number\_of\_persons is decremented.

if(number\_of\_persons < 0) //The count will not go to negative numbers and hence if it goes in negative then number\_of\_persons will remain zero only.

number\_of\_persons = 0;

}

void receive\_complete() org 0x1A //When receiving from the USART is complete.

{

ch[count] = UDR; //Storing the entered password character by character in the "ch" array using the count variable to set the index.

count++;

}

void usart\_initialize() //Initializing the Serial Port.

{

UCSRB = (1 << TXEN | 1 << RXEN); //Enabling usart Transmission and Reception.

UCSRC = 0x86; //Data Size : 8-bit, Stop Bit:1,No parity

UBRRL = 0x33; //X = (Fosc/(16(Desired Baud Rate)))-1

} // = (8\*10^6/(16 \*9600))-1

// = 52.08-1

// = 51 (Dec) i.e. 33(hex).

//Here, URSEl=0, so Fosc is divided by 16 if it was 1, Fosc would have been divided by 8.

void usart\_send(unsigned char ch) //Send the information via USART.

{

while(UCSRA.B5==0); // Wait till UDR is empty

UDR=ch; //The UDR register is loaded with the value that is to be transmitted i.e. ch.

}

void main()

{

unsigned char flag=0,count=0; //flag and count variables - used for receiving information from the PC and storing them in "ch".

unsigned long temp; //for giving 10 seconds delay when the user has entered the right password.

unsigned char pass[] = "123"; //"pass" char array contains the original password i.e. "123".

/\*

The lengths are used in the for loop below to know the size of the array.The below arrays are used to store the messages that are to be sent to the PC in different situations as described in the question.

\*/

unsigned char error[] = "Wrong Password.Please re - enter the password."; //Length : 46

unsigned char hang[] = "Your System has been hanged."; //Length : 28

unsigned char welcome[] = "Welcome to the Room .!"; //Length : 22

int compare; //It is used to compare the entered characters(password) with the stored characters(password) (character by character using for loop.)

int re\_enter = 0; //For counting the number of times the user entered the wrong password.

/\*

For checking whether the entered password is equal to the stored password.If they are equal then match will be equal to 1 else the match will be zero.Initially the match will be 1 and both the character arrays will be compared starting from bit 0, if any of the bit is different then it will be changed to zero.

\*/

int match;

int i; //Used in the for loop.

number\_of\_persons = 0; //Initializing the number of persons initially in the room to zero.

//DDRA

DDRA=0x0F; //Configure selection lines @ PORTA

PORTA=0x01; //For TIMER0\_COMP.

DDRA.B1 = 0; //For MQ - 2 Gas sensor.

DDRA.B7 = 0; //For lm35 temperature sensor.

//DDRB

DDRB.B0 = 1; //Output for Fan

DDRB.B4 = 1; //Output for Light

DDRB.B5 = 1; //Output for Red LED for informing that the maximum person limit has been reached.

DDRB.B7 = 1; //Output for Fire Alarm

//DDRC

DDRC=0xFF; //Output for Seven Segment Display.

//DDRD

DDRD.B2 = 0; //For Entry Sensor.

DDRD.B3 = 0; //For Exit Sensor.

DDRD.B4 = 1; //Red LED for maximum persons.

DDRD.B5 = 1; //For MQ - 2 Sensor.

DDRD.B6 = 1; //If the password is true then Green LED will turn on for 10 seconds.

PORTD = 1<<2 | 1 << 3; //Declaring both the pins of PORT D as Pull Up for IR sensors.

// Timer 0 CTC mode, pre-scaler 256 , count: 250

TCCR0=0x0B;

OCR0=0xF9;

MCUCSR |=(1<<JTD); // To Disable JTAG.

MCUCSR |=(1<<JTD);

MCUCR = 0X0A; //With the falling edge of INT0 and INT1 the interrupt is generated.

GICR &=~ (1<<INT0); //Disabling the INT0 for entry as it will be enabled only when the user enters the right password for 10 seconds and then it will be disabled.

GICR |= (1<<INT1); //Enabling the INT1 Interrupt.

SREG.B7 = 1; //Enabling Global Interrupt.

ADCSRA = 0x87; //Enabling ADC and for selecting prescalar clk/128.

ADMUX = 0x47; //For selecting AVCC pin and selecting the 7th pin of the ADC for lm 35 temperature sensor.

TIMSK.B1=1; //Enable Timer O Comp interrupt OCEI0

usart\_initialize(); //Initializing the Serial Port.

while(1)

{

flag=0,count=0; //Initially setting the flag and count to zero.

match=1; //Initializing match to 1 as described above.

/\*

For checking whether the entered password is equal to the stored password.If they are equal then match will be equal to 1 else the match will be zero.Initially the match will be 1 and both the character arrays will be compared starting from bit 0, if any of the bit is different then it will be changed to zero.

\*/

while(flag!=1)

{

while(UCSRA.B7==0) //Waiting when the data is received and the flag is set.

{

if(PINA.B1 == 0) //For MQ - 2 Sensor the logic is negative and hence when the input is zero then the gas is detected and we are giving output 1 which will turn on the buzzer.

{

PORTD.B5 = 1;

}

else if(PINA.B1 == 1) //Else the buzzer will be turned off.

{

PORTD.B5 = 0;

}

if(number\_of\_persons == 0) //If the number\_of\_persons in the room are zero then the fan and the light will be turned off which will conserve electricity.

{

PORTB.B0 = 0;

PORTB.B4 = 0;

}

else //if(number\_of\_persons != 0) //if there are non zero number\_of\_persons in the room then the light and fan will be turned on.

{

PORTB.B0 = 1;

PORTB.B4 = 1;

}

if(number\_of\_persons==9) //if the number\_of\_persons in the room are 9 that is the limit then Red LED will be turned on.

{

PORTB.B5=1;

}

else //else the red LED will be turned off.

{

PORTB.B5=0;

}

ADCSRA |= (1<<ADSC); //For Starting the ADC conversion.

while((ADCSRA &(1 << ADIF)) == 0); //Waiting for the conversion to complete.

temperature = ADCL; //Storing the converted value.

if(temperature >= 100) //If the it is greated then 100 which is around 49 C then the temperature buzzer will be turned on for some time.

{

PORTB.B7 = 1;

delay\_ms(1000);

temperature = 0; //Setting to zero because in the next round it will again take the new converted value and the value is going to change.

}

else

{

PORTB.B7 = 0; //If the temperature doesnt exceed the threshold then the buzzer will be kept off.

}

delay\_ms(100);

PORTC = SevenSegment\_Anode[number\_of\_persons]; //For displaying the number\_of\_persons on the seven segment.

}

{PORTC = SevenSegment\_Anode[number\_of\_persons];}

ch[count] = UDR; //Storing the entered password character by character in the "ch" array using the count variable to set the index.

count++; //Incrementing the count variable.

/\*

In the hyper terminal while entering the password we have to select the option "Append New Line".

So it sends '\r' at the end of the entered password automatically.

So here we are detecting that '\r' and deciding that the end of the password has been reached or not.

When the end of the password has been reached i.e. \r has been detected we are setting the flag variable to 1 and hence the loop condition will be false and it will get out of the loop and stop receiving the data.

\*/

if(UDR=='\r') //Checking that the end of the entered password has been reached or not.

flag = 1; //Setting the flag to 1 and hence it will get out of the while loop and discontinue receiving the data.

}

if(count==5) //Here our password is "123". So length the password is 3. so 3+1(of \r) and + 1 (count++ is below so it will also increment once) hence this way we are checking the length of the the entered password with the stored password.

{

for(compare = 0 ; compare < 3 ; compare++)

{

if(ch[compare] != pass[compare]) //Using "compare" as an index and comparing the entered password with the stored password character by character.If any of the character is different then "match" becomes zero and "Invalid Password" will be displayed.

{

match = 0;

}

}

if(match == 1) //If the password's are same then match will be 1.(as explained above.)

{

for(i = 0 ; i < 22 ; i++) //Sending the message "Welcome to the Room .!".

{

usart\_send(welcome[i]);

}

usart\_send(0x0D); //For sending the message to the HC - 05 module.

// number\_of\_persons--;

for(temp = 0 ; temp < 2000000 ; temp++) //delay of around 10 seconds.

{

PORTD.B6 = 1; //Green LED will be turned ON.

GICR |= (1<<INT0); //Enabling the entry sensor hence after entering the correct passrod during this 10 seconds a person can enter the room and the sensor will detect the entry.

PORTC = SevenSegment\_Anode[number\_of\_persons]; //Displaying the number\_of\_persons on Seven Segment i.e. PORTC.

}

PORTD.B6 = 0; //Turning the green LED off.

PORTC = SevenSegment\_Anode[number\_of\_persons]; //Displaying the number\_of\_persons on Seven Segment i.e. PORTC.

GICR &=~ (1<<INT0); //Disabling the entry sesor interrupt.

}

else //If the password's are not the same then we are sending error message using for loop.

{

for(i = 0 ; i < 46 ; i++)

{

usart\_send(error[i]); //Sending the message "Wrong Password.Please re - enter the password.".

}

usart\_send(0x0D); //For sending the message to the HC - 05 module.

re\_enter++; //The entered password is wrong and hence incrementing the re\_enter counter.

}

}

else //If the length of the passwords do not match then of course it is a wrong password.Hence, here also we are displaying the error message.

{

for(i = 0 ; i < 46 ; i++)

{

usart\_send(error[i]); //Sending the message "Wrong Password.Please re - enter the password.".

}

usart\_send(0x0D); //For sending the message to the HC - 05 module.

re\_enter++; //The entered password is wrong and hence incrementing the re\_enter counter.

}

/\*

Every time the user enters the wrong password we are incrementing the "re\_enter" counter.Hence , here we are checking the "re\_enter" variable which is initially zero.so when the user has entered the password incorrect for three times then the system will get hanged.

\*/

if(re\_enter >= 3)

{

for(i = 0 ; i < 28 ; i++)

usart\_send(hang[i]); //Sending the message "Your System has been hanged.", if the user has entered the incorrect password three times.

usart\_send(0x0D);

break; //Exits the loop.

}

}

}

**Conclusions**

We were successfully able to implement the features we wanted to include with slight variations. First we learned to interface every sensor with AT mega 32 individually. Then we merged together all the codes of different sensors which was a bit challenging task. We had used polling method for the individual components but when merged we had to keep some process running in the background along with waiting for some reading. Hence then we learned the interrupt method. Hence if we had started directly with the interrupt method for the individual components then it would be easy to merge all the codes of different components. We learned programming the AT mega 32 with Atmel Studio. We were not able to detect multiple persons going in at the same time. In that situation the sensors we used would detect only a single person while actually more than single person had taken entry which would display false count of the number of persons in the room. Working with HC – 05 was fun because you could actually send and receive the data to and from the AT mega and then code accordingly to control the pins. Interfacing with MQ – 2 sensor was an easy task. While working with the IR sensors we had some problem when we worked with it in the some light. When the IR sensor was exposed to sunlight, it would detect the sunlight and not the persons going in and out of the room. With LM – 35 we learned how to use the ADC feature of the ATmega 32. We had some drawbacks and we still have some future plans for improving the current features and adding more features to the smart room.

**Project Timeline**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date/Work | 23/03/16 | 04/04/16 | 11/04/16 | 18/04/16 | 25/04/16 |
| Assignment-2. | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg |  |  |  |  |
| Programming and Implementation on microchip of Entry-Exit. |  | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg |  |  |  |
| Programming and Implementation on microchip of Light and Fan. |  | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg |  |  |
| Programming and Implementation of gas sensor. |  |  | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg |  |
| Understand the working of Bluetooth module(HC05). |  |  |  | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg |  |
| Programming of  Temperature Sensor. |  |  |  | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg |  |
| Implementation of Bluetooth module. |  |  |  |  | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg |
| Implementation of temperature Sensor. |  |  |  |  | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg |
| Testing and removing bugs. |  |  |  |  | https://solidwize.com/wp-content/uploads/2012/04/Green-Check-Mark.jpg |

**References**

[1]M.A. Mazidi, S. Naimi, S. Naimi**,** "The AVR Microcontroller and Embedded Systems: Using Assembly and C " in Embedded System Design, PEARSON, 2015.

[2] <http://www.ablab.in/how-to-interface-a-hc-05-bluetooth-module-with-avr-atmega32-microcontroller/>

[3] <https://www.youtube.com/watch?v=cP-qq9M3CNc>

[4] <http://extremeelectronics.co.in/avr-tutorials/interfacing-temperature-sensor-lm35/>