"Smart-shop"

Project Report

Embedded System Designing

in

ICT

By

Group no - 16

Maunil Vyas (1401007)
Deep C. Patel (1401010)
Shreyas Patel (1401025)

Under the guidance of

Course Instructor: Mr. Anurag Lakhlani

Mentored by: Bhavika Patel, Kavita Anjaria

Institute of Engineering and Technology

Ahmedabad University
Ahmedabad - 380009
May 2016

Index

1. Introduction

Motivation:

As we know that in modern days, time has become one of the most important thing for everyone and sudden change in the technology has changed the world. We are living in the society where people demand the new technology which saves their time and efforts.

Accepting above facts, and as we are an engineering students, it is our duty to serve the society with new technologies.

Description: -

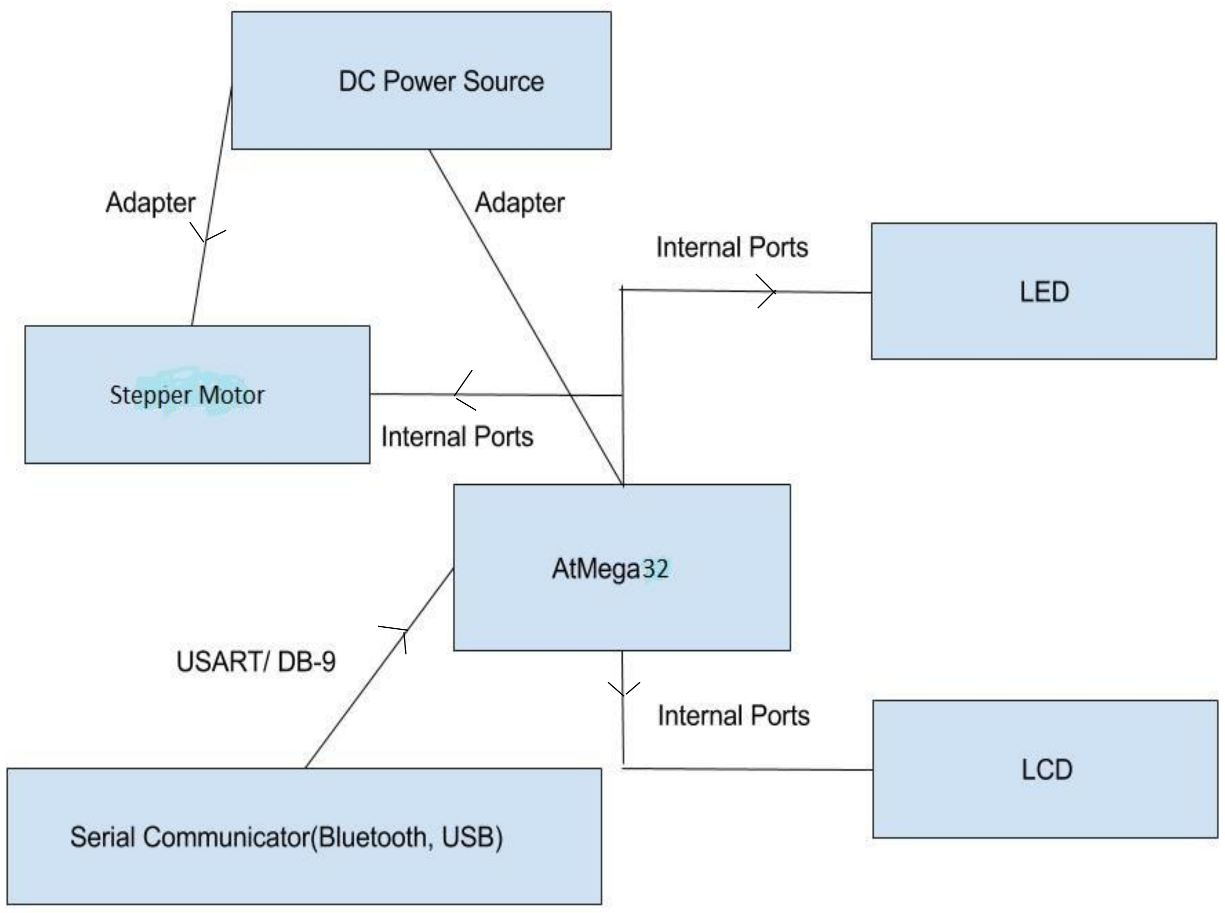
This project is a prototype of smart super market, In this project we are making shopping easier and advanced, basically in this system customer has to enter his or her desired items and all remaining tasks have been performed by our system (Like Collecting items from different racks and put it into customer's basket and after completing all tasks, customer has been delivered his or her own basket from where he or she can collect their items).

2. Market analysis

Market Analysis:-

We did the market analysis, and almost everywhere we have seen the backwardness of current shopping-system (like time consuming, not efficient).

3. Block Diagram



4.Selection criteria for listed components

1. AVR Microcontroller: -
    a. Faster execution and Affordable price
    b. UART Serial Port
    c. Internal and External interrupts
    d. 32 general purpose registers
    e. Availability of Timers and Counters
    f. Availability of C compiler

2. LCD (HD44780U)
    a. Lower power Consumption
    b. Sufficient RAM and ROM Memory
    c. Easy interface with existing Microcontroller
    d. Availability of various Interface functions (i.e, Clear, Shift Cursor, etc…)

3. LED
    a. Faster On/Off

4. ULN2003a
    a. Use as a driver of stepper motor
    b. Versatile device useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal print heads and high power buffers

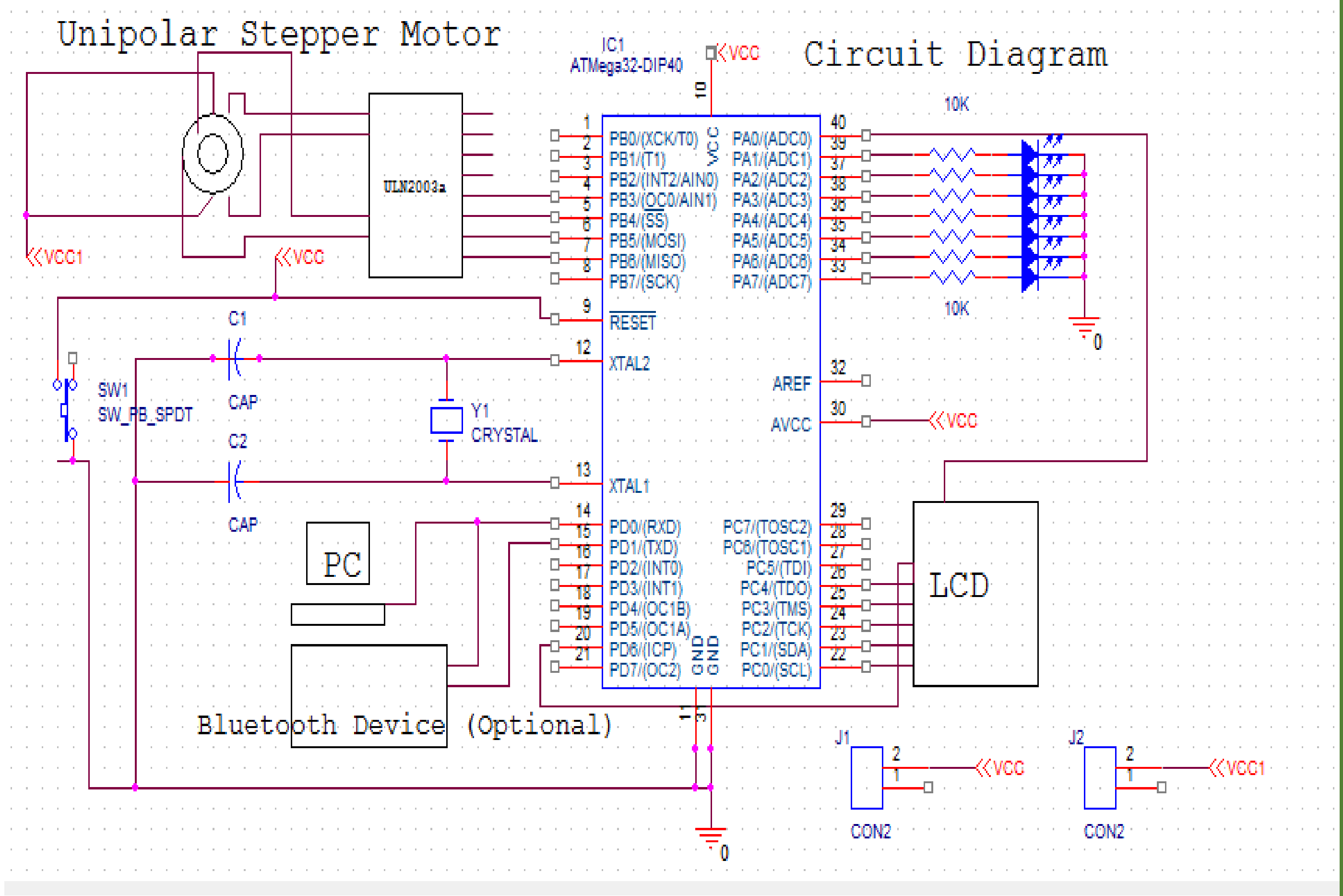
5. Stepper Motor
    a. Satisfy the functional Needs
    b. Low Power Consumption
    c. Durable
    d. Portable
    e. Affordable Price

5.Costing table

Component Name	Model	Company	Number of component	Description	Cost of component
AVR Microcontroller	ATmega32	Atmel[2]	1	For controlling whole system	500/-
LCD	HD44780U	Hitachi[3]	1	To display customer bill and other major things	140/-
LED	Regular LED	-	6-10	What is quantity for any product chosen by customer	1 rupee per LED
Bluetooth UART module(Optional)	RKI-1545	Robokits India[4]	1	To pass selected customer product related data to AVR	250/-
ULN2003a[5]	-	-	1	Driver for stepper motor	15/-
Stepper Motor	-	-	1	To rotate at particular angel to reach at product	250/-

6. ***Note: - Here we are using pc for serial communication.(USART)**

6.Circuit Diagram



LCD Interfacing reference[1]

7.Real problems encountered and their solutions

Problem: Problem of efficient design(means there would be no sensors and multiple users would be served with minimum cost by the system)

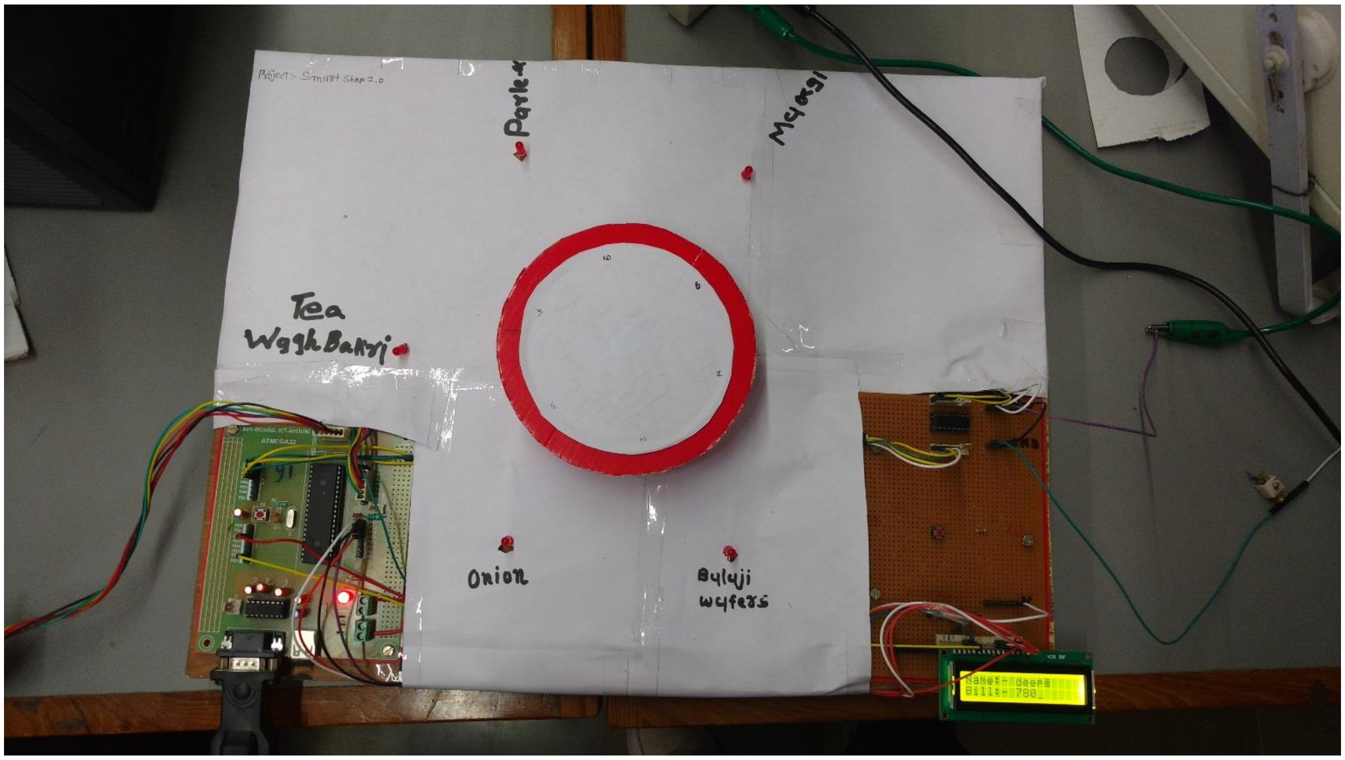
Solution: We model our system in a such a way that there would be no sensor at all, also applying our computer programming skills, we have successfully generated our own algorithm which solves our another concern of serving multiple customers with efficient manner.

Problem: Another problem is which one is the best platform for communication with microcontroller in our system(Run time Communication with micro controller for selecting items and their quantities from the shop).

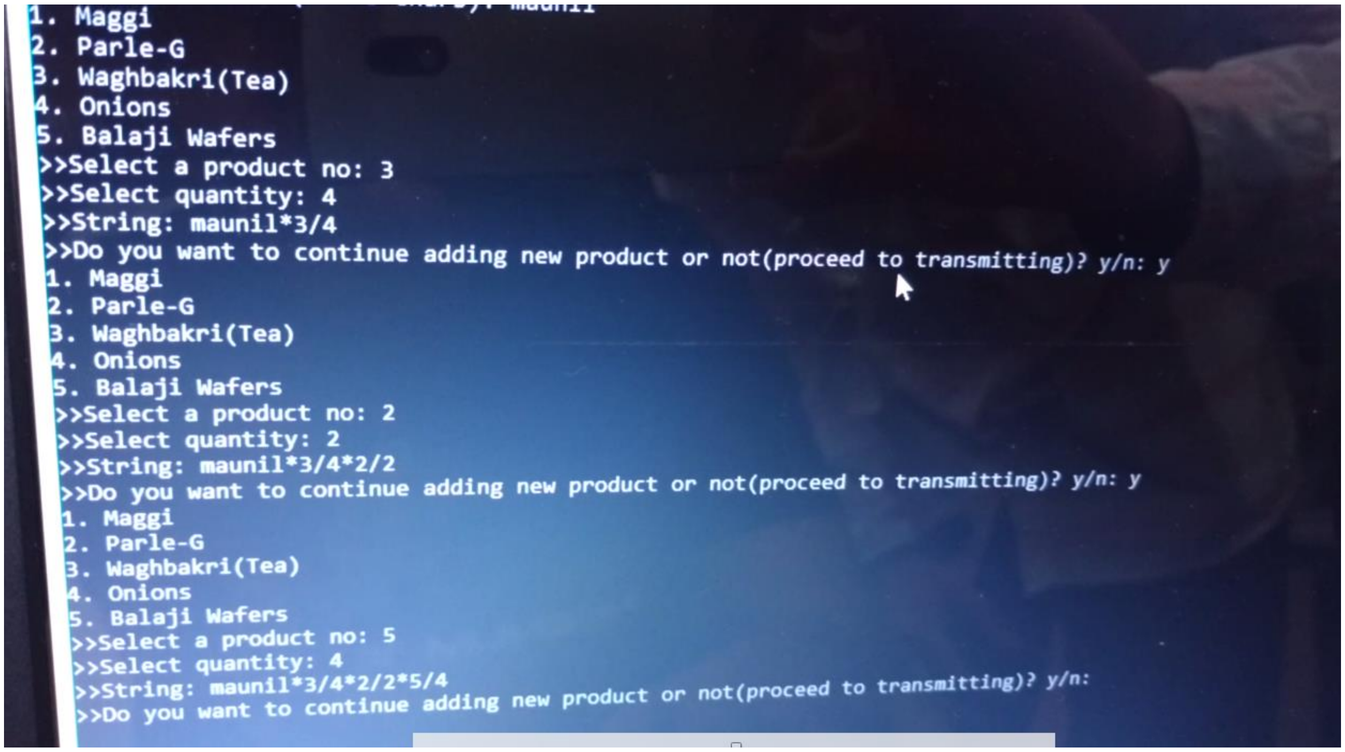
Solution: We have two options one is USART and another one is Bluetooth device, and finally we conclude to stay with USART, reason is obvious we all know that Bluetooth has short range so it won’t satisfy our aim for long range (one may say then wifi module is good one but due knowing the pros and cons of wifi module with Atmega 32 ,we do not think about it)

8. Snapshot of working Model

Final Model

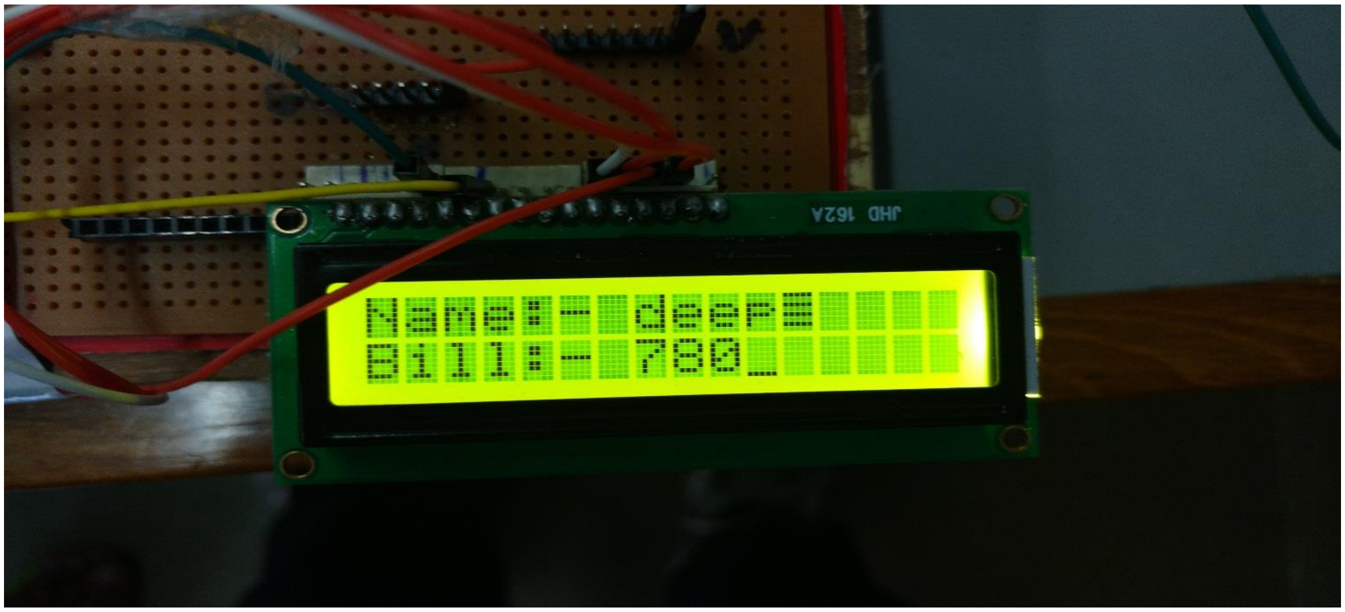


Serial Communication with Pc(Result of Python Script)



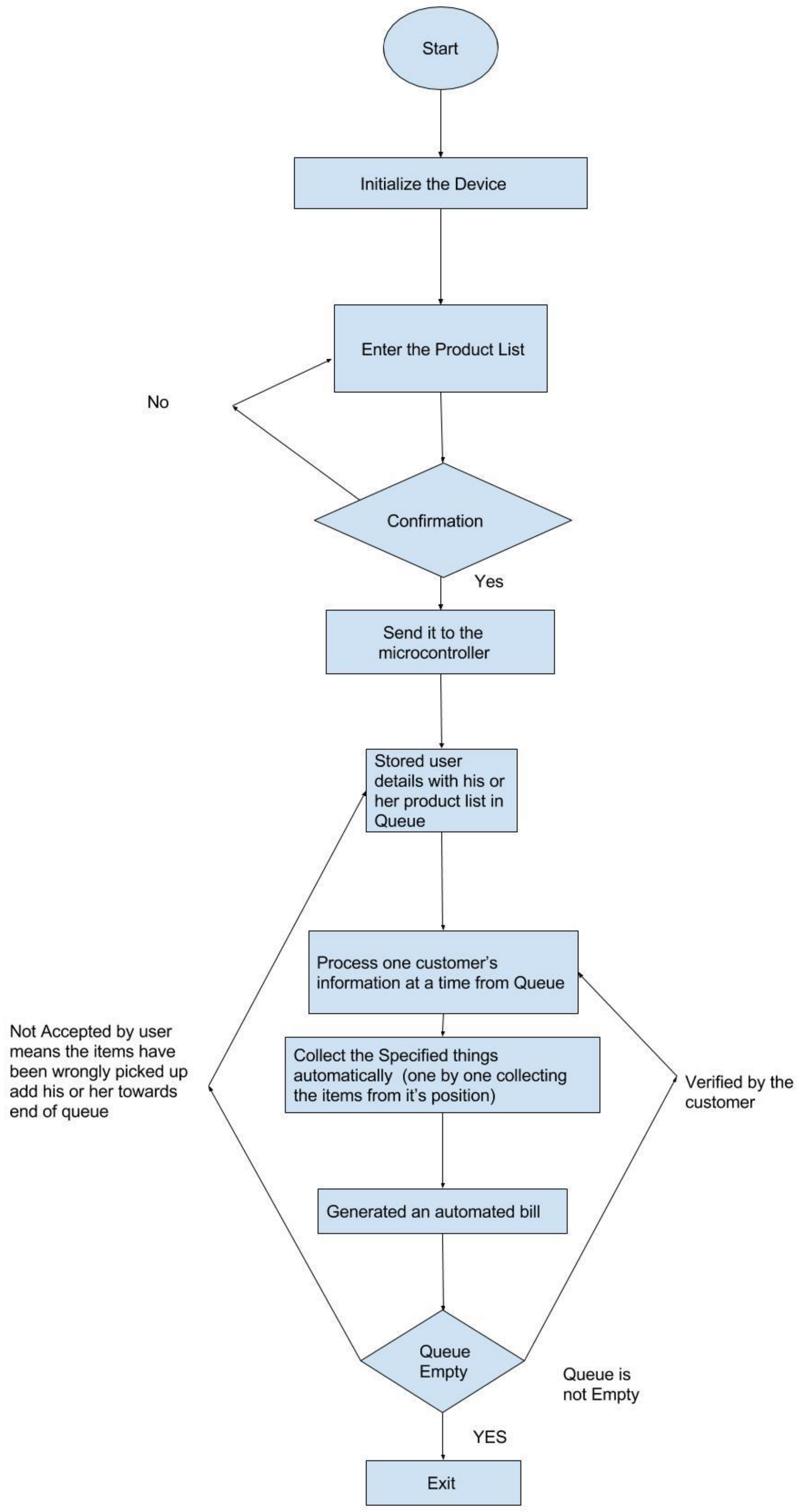
```
1. Maggi
2. Parle-G
3. Waghbakri(Tea)
4. Onions
5. Balaji Wafers
>>Select a product no: 3
>>Select quantity: 4
>>String: maunil*3/4
>>Do you want to continue adding new product or not(proceed to transmitting)? y/n: y
1. Maggi
2. Parle-G
3. Waghbakri(Tea)
4. Onions
5. Balaji Wafers
>>Select a product no: 2
>>Select quantity: 2
>>String: maunil*3/4*2/2
>>Do you want to continue adding new product or not(proceed to transmitting)? y/n: y
1. Maggi
2. Parle-G
3. Waghbakri(Tea)
4. Onions
5. Balaji Wafers
>>Select a product no: 5
>>Select quantity: 4
>>String: maunil*3/4*2/2*5/4
>>Do you want to continue adding new product or not(proceed to transmitting)? y/n:
```

Customer's bill displaying



```
Name:- deep
Bill:- 780_
```

9. Flowchart


Start
Initialize the Device
Enter the Product List
No
Confirmation
Yes
Send it to the microcontroller
Stored user details with his or her product list in Queue
Process one customer's information at a time from Queue
Collect the Specified things automatically (one by one collecting the items from it's position)
Not Accepted by user means the items have been wrongly picked up add his or her towards end of queue
Verified by the customer
Generated an automated bill
Queue Empty
Queue is not Empty
YES
Exit

10.Code

```
/*
 * SmartShop.c
 *
 *
*/
#ifndef F_CPU
#define F_CPU 8000000UL
#endif


#include<avr/io.h>
#include<stdio.h>
#include<stdlib.h>
#include<util/delay.h>
#include<avr/interrupt.h>


//LCD Connections

//////////////////////////////////////////////////////////////////////////////////////////////////// LCD CODE

ISR (INT0_vect) // Count LED on every interrupt
{
        int DataCharCount=0;
        char TempData[45];
        char p;
        char *Data;
        int i;

        while(1)
        {
                while((UCSRA & (1<<7))==0);
                p = UDR;

                if(p=='\r')
                {
                        break;
                }

                TempData[DataCharCount] = p;
                DataCharCount++;
        }

        Data=(char*)malloc((DataCharCount)*sizeof(char));

        for(i=0;i<DataCharCount;i++)
        {
                Data[i]=TempData[i];
        }

        interrupt(Data);
}



void LCD_data(unsigned char Data)// to provide data to LCD
{
    PORTC=Data&0xF0; // Send Higher nibble (D7-D4)

        PORTA|=(1<<0);       // Register Select =1 (for data select register)
    PORTD|=(1<<6);      //Enable=1 for H to L pulse

        _delay_us(5);    // Delay
        PORTD^=(1<<6);   //  PORT.D6=0

    PORTC=((Data<<4)&0xF0); // Send Lower nibble (D3-D0)
    PORTD|=(1<<6);               //Enable=1 for H to L pulse

        _delay_us(5);

        PORTD^=(1<<6);   //  PORT.D6=0

    _delay_us(100);
}


//LCD Print
void LCD_Print(char * str)// to print string to LCD
{
     unsigned char i=0;
     // Till NULL charecter is reached, take each character
     while(*(str+i)!=0)
     {
          LCD_data(*(str+i)); // Data sent to LCD data register
          i++;
          _delay_ms(10);
     }
}
//LCD Command
void lcdcommand(unsigned char command)// to provide command to LCD
{
    PORTC=command&0xF0; // Send Higher nibble (D7-D4)

    PORTA&=~(1<<0); // Register Select =0 (for Command register)
    PORTD|=(1<<6); //Enable=1 for H to L pulse

    _delay_us(5);
    PORTD^=(1<<6);
    _delay_us(100);

    PORTC=((command<<4)&0xF0);  // Send Lower nibble (D3-D0)
    PORTD|=(1<<6); //Enable=1 for H to L pulse
    _delay_us(5);
    PORTD^=(1<<6);

    _delay_us(40);
}

// Cursor Position
void Cursor_Position(unsigned short int x,unsigned short int y)
{
   unsigned char firstcharadd[] ={0x80,0xC0}; // First line address 0X80
                                              //Second line address 0XC0
   lcdcommand((firstcharadd[x-1]+y-1));// to get address line on LCD

}

void clear()// To clear LCD previous data
{
  lcdcommand(0x01); // to clear LCD
  _delay_ms(2);
}

 //LCD Iniatialize
void LCD_Initialize()
{
    PORTD&=~(1<<6);

    lcdcommand(0x33); // Initialize LCD for 4 bit mode
    lcdcommand(0x32); // Initialize LCD for 4 bit mode
    lcdcommand(0x28); // Initialize LCD for 5X7 matrix mode
    lcdcommand(0x0E); //Display on,cursor blinking
    clear();
    lcdcommand(0x06); //Shift cursor to right
    lcdcommand(0x80);
}


////////////////////////////////////////////////////////////////////////
// USART RECEIVE CODE


void usart_initialize()
{
        UCSRB=0x18; // Rx Enable
        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)
        //Here, URSEl=0, so Fosc is divided by 16 if it was 1 Fosc would
        //Have been diveded by 8
}

//////////////////////////////////////////////////////////////////////////////////
//Node

int ProductPrice[6] = {0,50,150,210,120,144};// store price of product
int Quant;// global quantity

struct node
{
        int item;
        int Quantity;
        struct node *link;
}*top;

struct nodeQ
{
        int UserNo;
        int CurrentPos;
        int TotalPrice;       //To be displayed at LCD
        char *Username;
        struct nodeQ *linkQ;
        struct node *Stack;
} *front,*rear;

int Qsize=1;
int Ssize=0;
//////////////////////////////////////////////////////////////////////
//Queue

void insertQ(int val,int totPrice,char *Name,int c)
{
        struct nodeQ *temp;
        int k;
        temp=(struct nodeQ*)malloc(sizeof(*temp));

        temp->UserNo = val;
        temp->CurrentPos = 0;
        temp->TotalPrice = totPrice;       //To be displayed at LCD

        //strcpy(temp->Username,Name);

        temp->Username = (char *)malloc((c) * sizeof(char));

        for(k=0;k<c;k++)
        {
                temp->Username[k] = Name[k];
        }
        temp->linkQ = '\0';
        temp->Stack = '\0';

        if (front == '\0')
        {
                front = temp;
                rear=temp;
        }
        else
        {
                rear->linkQ = temp;
                rear = temp;
        }

        Qsize++;
}

struct nodeQ* getElementAddr(int i) // Get the Position value form array
{
        struct nodeQ *temp;
        int j;
        temp = front;

        if(i>Qsize||i<1)
        {
                return '\0';
        }

        for(j=1;j<i;j++)
        {
                temp = temp->linkQ;
        }

        return temp;
}

void Delete(int i)        // Queue delete ith Node
{
        struct nodeQ *temp;
        struct nodeQ *temp1;
        int j;
        temp = front;


        if(i>Qsize||i<0)
        {
                //why
        }
        else if(i==0)
        {
                front=front->linkQ;
                Qsize--;
        }
        else
        {
                for(j=1;j<i;j++)
                {
                        temp = temp->linkQ;
                }

                temp1=temp->linkQ;
                temp->linkQ=temp1->linkQ;

                Qsize--;
        }
}
//////////////////////////////////////////////////////////////////
//Stack
```

```
void push(int val,int no)// stack
{
        struct node *temp=(struct node*)malloc(sizeof(struct node));
        //You can also write node *temp=malloc(sizeof(node));

        temp->item = val;
        temp->Quantity = no;
        temp->link = '\0';

        if(top=='\0')
        {
                top = temp;
        }
        else
        {
                temp->link = top;
                top = temp;
        }
}


int pop() //Stack Pop
{
        struct node *temp;

        int val;

        if(top == '\0')
        {
                val=-1;
        }
        else
        {
                val = top->item;
                Quant= top->Quantity;

                temp = top;
                top = top->link;
                //free(temp);
        }

        return val;
}

int peak()
{
        int val;

        if(top == '\0')
        {
                val=-1;
        }
        else
        {
                val = top->item;
        }
        return val;
}

int getMinRotation()//Result Compute
{
        struct nodeQ *temp;
        int Min=5;// min =7
        int x=0;
        int m=0;
        int j;
        temp = front;

        if(temp=='\0')
        {
                Min=0;
        }
        else
        {
                for(j=1;j<Qsize;j++)
                {
                        top=temp->Stack;
                        m=peak();

                        x=m-(temp->CurrentPos);

                        if(x<Min)
                        {
                                Min=x;
                        }
                        temp = temp->linkQ;
                }

        }
        return Min;
}

void StackPop()
{
        ///////////////////////////////////////////////////////////////
        //////////// Variable declaration
        struct nodeQ *temp;
        int INTCount;
        int take;
        int TempPrice;
        int j,i;
        int LEDTemp[5][2];// for temporary storing data
        int **LED;// for taking data;
        int UserCount=0;// For count user who are going to pop
        int MAX_Rotation=0;// maximum time rotation possible
        int Rotation=0;// initial value for rotation
        unsigned char LEDDisplay=0X00;
        char *Str;
        temp = front;
        ///////////////////////////////////////////////////////////////
        ////////// Fetch Array of product and it's quantity and store in LEDTemp
        for(j=1;j<Qsize;j++)
        {
                top=temp->Stack;

                if(peak()==-1)
                {
                        clear();
                        INTCount=0;
                        Cursor_Position(1,1);
                        LCD_Print("Name:- ");
                        Cursor_Position(1,8);
                        LCD_Print(temp->Username);
                        Cursor_Position(2,1);
                        LCD_Print("Bill:- ");

                        TempPrice=temp->TotalPrice;
                        while(TempPrice!=0)
                        {
                                TempPrice=TempPrice/10;
                                INTCount++;
                        }

//                      Str = NULL;
                        Str=(char *)malloc(INTCount*sizeof(char));
                        sprintf(Str,"%d",temp->TotalPrice);
                        Cursor_Position(2,8);
                        LCD_Print(Str);
                        _delay_ms(300);
                        Delete(j-1);
                }
                else
                {
                        if(temp->CurrentPos==peak())
                        {
                                take=pop();        //Light the LEDs at "take" Position object no times
                                LEDDisplay=LEDDisplay | (1<<take);
                                LEDTemp[UserCount][0]=take;
                                LEDTemp[UserCount][1]=Quant;

                                if(Quant>MAX_Rotation)
                                {
                                        MAX_Rotation=Quant;
                                }

                                UserCount++;
                        }
                }
                temp->Stack=top;
                temp=temp->linkQ;
        }

        ///////////////////////////////////////////////////////
        //// Reallocation of Array

        LED=(int**)malloc((UserCount)*sizeof(int*));

        for(i=0;i<UserCount;i++)
        {
                LED[i]=(int*)malloc((2)*sizeof(int));
        }
        ////////////////////////////////////////////////////////

        for(i=0;i<UserCount;i++)
        {
                for(j=0;j<2;j++)
                {
                        LED[i][j]=LEDTemp[i][j];
                }
        }
        ////////////////////////////////////////////////////////
        //// Copy elements from above array to LED
        //free(LEDTemp);// free memory of temporary variable
        //////////////////////////////////////////////
        while(1)
        {
                if(Rotation==MAX_Rotation)
                {
                        break;
                }

                PORTA=LEDDisplay;
                _delay_ms(1000);
                PORTA=0X00;
                _delay_ms(1000);
                Rotation++;

                for(i=0;i<UserCount;i++)
                {
                        if(LED[i][1]==Rotation)
                        {
                                LEDDisplay = LEDDisplay ^ (1<<LED[i][0]);
                        }
                }
        }
        //////////// End of function
}
```

```
int updatePositionTill(int i,int rot)
{
        struct nodeQ *temp;
        int j;
        temp = front;

        if(i>(Qsize-1)||i<1||rot<0)
        {
                return -1;
        }

        for(j=0;j<i;j++)
        {
                temp->CurrentPos=(temp->CurrentPos)+rot;
                temp = temp->linkQ;
        }

        return 1;
}

void stepper_move()      //30 degree
{
        //This should be in sequence means blue pink yellow and orange
        // sequence for stepper motor
        PORTB = 0x0C;
        _delay_ms(10);

        PORTB = 0x06;
        _delay_ms(10);

        PORTB = 0x03;
        _delay_ms(10);

        PORTB = 0x09;
        _delay_ms(10);
}

void insertIntoStack(int i,int val,int quanty)
{
        struct nodeQ*temp;

        temp=getElementAddr(i);

        top=temp->Stack;

        push(val,quanty);

        temp->Stack=top;

}

void Bubblesort(int **ar, int size)/// Sort product list
{
        int i,j,temp,temp1;

        for(i=0;i<size;i++)
        {
                for(j=0;j<size-1;j++)
                {
                        if(ar[j][0]<ar[j+1][0])
                        {
                                temp=ar[j][0];
                                ar[j][0]=ar[j+1][0];
                                ar[j+1][0]=temp;

                                temp1=ar[j][1];
                                ar[j][1]=ar[j+1][1];
                                ar[j+1][1]=temp1;
                        }
                }
        }
}

void Rotate()
{
        int user=(Qsize-1);
        int rot;
        int i=0,k;

        while(front!='\0')
        {
                rot=getMinRotation();

                if(rot>=0)
                {
                        for(k=0;k<rot;k++)
                        {
                                stepper_move();// 30 degree rotation
                                stepper_move();// 30 degree rotation
                                _delay_ms(500);
                        }
                        updatePositionTill(user,rot);
                }

                StackPop();

                i++;
                user=Qsize-1;

        }

}

void interrupt(char *Data)
{
        int i,j,k;
        int TwoArray[5][2];//for temporary use
        char Name[8];
        char *Fname;
        int pro;int Bill;
        int **NewArray;

        ////////////////////////////////////////////////////////////
        //// Fetch User name


        for(i=0;i<8;i++)
        {
                if(Data[i]=='*')
                {
                        break;
                }
                else
                {
                        Name[i]=Data[i];
                }
        }

        Fname = (char *)malloc((i) * sizeof(char));
        for(k=0;k<i;k++)
        {
                Fname[k]=Name[k];
        }
        //////////////////////////////////////////////////////////
        ////////// To calculate bill and to make array useful for sorting
        pro=0;Bill=0;
        for(j=i;Data[j]!='\0';j++)
        {
                if(Data[j]=='*')
                {
                        j++;
                        TwoArray[pro][0]=Data[j]-'0';// to convert char into int
                        j=j+2;
                        TwoArray[pro][1]=Data[j]-'0';// to convert char into int
                        Bill = Bill + (ProductPrice[TwoArray[pro][0]]*TwoArray[pro][1]);
                        pro++;
                }
        }
        ////////////////////////// Insert username and bill data into Queue
        insertQ(Qsize,Bill,Fname,i);

        NewArray=(int**)malloc((pro)*sizeof(int*));

        for(i=0;i<pro;i++)
        {
                NewArray[i]=(int*)malloc((2)*sizeof(int));
        }


        for(i=0;i<pro;i++)
        {
                for(j=0;j<2;j++)
                {
                        NewArray[i][j]=TwoArray[i][j];
                }
        }

        //////////////////////////Sort the List
        Bubblesort(NewArray,pro);
        ////////////////////////// Insert user item into stack
        for(i=0;i<pro;i++)
        {
                insertIntoStack(Qsize-1,NewArray[i][0],NewArray[i][1]);
        }
}

void Interrupt_Initializer()
{
        sei(); // For enable all interrupts
        MCUCR=0X02; // For lower level
        GICR = (1<<INT0);//enable INT0
}

int main()
{
        //Set-up PORTS for LCD
        DDRC=0xFF;  // For D7-D4 // LCD
        DDRA=0XFF;  // For A5-A0 // for LED
        DDRD=0XF0;  // D3-D0 for RX and INT0 where as D6 for LCD Enable
        DDRB=0XFF;  // for stepper motor

        front='\0';
        rear='\0';
        top='\0';

        PORTD = 0X00;//Pull down register

        Interrupt_Initializer();
        usart_initialize();
        LCD_Initialize(); //Initialize

        while(1)
        {
                Rotate();
        }
        return 0;
}
```

11.Conclusion

- We have successfully achieved our goal, and got an experience of building real time system.

12.Project timeline

Work	23/3/2016	4/4/2016	11/4/2016	18/4/2016	25/4/2016
Design of Model and circuit Paper	Done				
Mechanical Design	Running as project goes forward	Running	Running	Running	Final mechanical design
Interfacing		LED, LCD Interfacing Done Stepper Running			
Code Implementation			Done		
Code Testing			Running		
Code debugging				Running	
Final working output					Final errorless code

13.References

[1] http://www.ablab.in/16x2-alphanumeric-lcd-interfacing-with-avr-atmega32-microcontroller/
[2] http://www.atmel.com/images/doc2503.pdf
[3] https://www.sparkfun.com/datasheets/LCD/HD44780.pdf
[4] http://robokits.co.in/wireless-solutions/bluetooth-uart-module-based-on-hc-05
[5] http://www.alldatasheet.com/datasheet-pdf/pdf/25575/STMICROELECTRONICS/ULN2003.html