```
/*
   Project Group_29
  Project description : AUTOMATIC RAILWAY GATE CONTROL SYSTEM WITH ADDITIONAL FEATURES
 */

//defining various PORTS
#define LCD_RS  PORTA.B2   // RS
#define LCD_EN  PORTD.B6   //Enable
#define LCD_D4  PORTC.B4   //Data Bit 4
#define LCD_D5  PORTC.B5   //Data Bit 5
#define LCD_D6  PORTC.B6   //Data Bit 6
#define LCD_D7  PORTC.B7   //Data Bit 7
#define US_PORT PORTA
#define  US_PIN   PINA
 #define US_DDR    DDRA
#define US_POS PORTA.B0     //PORTA0
 #define US_ERROR 0xffff
 #define  US_NO_OBSTACLE 0xfffe

//variable declaration
unsigned int display[] = {0x01,0x4F,0x12,0x06,0x4C,0x24,0x20,0x0F,0x00,0x0C ,0x08 ,0x60, 0x31 ,0x42
,0x30,0x38};        //The display array consists of values which are used to display values from 0 to F in
single seven segment
unsigned int i,n;                //variable declaration

INT0_vect() org 0x002        //This is the function which is called when external interrupt request 0 occurs
{
 n=0x01;         //n is initialised to 0x01 which is used in the main function
}

INT1_vect() org 0x004        //This is the function which is called when external interrupt request 1 occurs
{
 n=0x02;         //n is initialised to 0x02 which is used in the main function
}


void LCD_data(unsigned char Data)          //function to print a character in LCD
{
PORTC=Data&0xF0; // Send Higher nibble (D7-D4)
LCD_RS=1;       // Register Select =1 (for data select register)
LCD_EN=1;      //Enable=1 for H to L pulse
delay_us(5);
LCD_EN=0;               //Enable=0 for H to L pulse


PORTC=((Data<<4)&0xF0); // Send Lower nibble (D3-D0)
LCD_EN=1;              //Enable=1 for H to L pulse
delay_us(5);
LCD_EN=0;                           //Enable=0 for H to L pulse

delay_us(100);
}


void LCD_Print(char * str)             //LCD Print
{
 unsigned char i=0;

 // Till NULL character is reached, take each character
 while((str[i])!=0)

 {
```

```
  LCD_data((str[i]));           // Data sent to LCD data register
  i++;                  // i is incremented
  delay_ms(10);   //delay of 10 ms
 }
}

void lcdcommand(unsigned char command)           //LCD Command
{
PORTC=command&0xF0; // Send Higher nibble (D7-D4)
LCD_RS=0; // Register Select =0 (for Command register)
LCD_EN=1; //Enable=1 for H to L pulse
delay_us(5);
LCD_EN=0;
delay_us(100);

PORTC=((command<<4)&0xF0);  // Send Lower nibble (D3-D0)
LCD_EN=1; //Enable=1 for H to L pulse
delay_us(5);
LCD_EN=0;
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));

}

// Clear the screen
void clear()
{
  lcdcommand(0x01);
  delay_ms(2);
}

 //LCD Iniatialize
void LCD_Initialize()
{
LCD_EN=0;

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
}

int getPulseWidth()          //code to get the pulse width through ultrasonic sensor
   {
      int i,result;

      //Wait for the rising edge
      for(i=0;i<600000;i++)
      {
         if(!(US_PIN & (1<<US_POS))) continue; else break;
     }
```

```
    if(i==600000)
      return 0xffff; //Indicates time out

    //High Edge Found

    //Setup Timer1
    TCCR1A=0X00;
    TCCR1B=(1<<CS11); //Prescaler = Fcpu/8
    TCNT1H=0x00;      //Init counter

    //Now wait for the falling edge
    for(i=0;i<600000;i++)
         {
      if(US_PIN & (1<<US_POS))
      {
        if(TCNT1H > 60000) break; else continue;
      }
      else
        break;
    }

    if(i==600000)
      return 0xffff;          //Indicates time out

    //Falling edge found

    result=TCNT1H;

    //Stop Timer
    TCCR1B=0x00;

    if(result > 60000)
      return 0xfffe;          //No obstacle
    else
      return (result>>1);
  }

  void Wait()
 {
   int i;
    for(i=0;i<10;i++)
      delay_ms(1000);
 }



void main()
{
int r;
DDRC=0xFF;  // For D3-D0
DDRA.B2=1;  //For RS
DDRD.B6=1;  //For Enable
DDRD.B0 = 1; //For buzzer
DDRA = 0xEE;
DDRD.B2 = 0;
DDRD.B3 = 0;
DDRB = 0xFF;
SREG.B7 = 1;          //GLOBAL INTERRUPT
GICR.B7 = 1;         //To enable interrupt 1
GICR.B6 = 1;     //To enable interrupt 0
GICR.B5 = 1;
```

```
MCUCR = 0x0F;         //last four bits indicates that in interrrupt 0 and 1 the interrupt is generated on rising
edge
MCUCSR = 0x40;            //Interrupt 2 works on rising edge
n=0x00;             //n is initilally assigned to 0

 LCD_Initialize();             //Initialize
  Cursor_Position(1,3);
   LCD_Print("Safe ");        // Printing Hello at 1st row and 3rd columm
   Cursor_Position(2,5);
   LCD_Print("Journey ");        // Printing World at 2nd Row and 5th column

  while(1)            //loop works continously
   {

    if(PINA.B4==1)            //PINA.B4 checks whether LDR sensor receives light or not when PINA.B4 is 1 it
receives light and hence the LED connected to   PORTA.B6  and    PORTA.B6  are off
  {
        PORTA.B6 = 0;
       PORTA.B7 = 0;
  }

 else if(PINA.B4==0)        //PINA.B4 checks whether LDR sensor receives light or not when PINA.B4 is 0 it
doesn't receive light and hence the LED connected to   PORTA.B6  and    PORTA.B6  turned on
  {
       PORTA.B6 = 1;
       PORTA.B7 = 1;
       delay_ms(10000);   //LED remain on only for 10s
       PORTA.B6 = 0;
       PORTA.B7 = 0;
  }

  if(n==0x01)       //case to close the barrier
   {
          PORTB.B4 =1;                //The red LED connected to PORTB.B4 glows which indicates that train
is approaching

            OCR2=256;

          TCCR2 |= (1 << COM21);            // set none-inverting mode

          TCCR2 |= (1 << WGM21) | (1 << WGM20);   // set fast PWM Mode

          TCCR2 |= (1 << CS21);      // set prescaler to 8 and starts PWM
          PORTC = 0x01;               //enable for Motor
          PORTB = 0x1A;               //The last four bits of PORTB are 1010 in which 2 bits are for first motor
and next 2 are for second motor (10 indicates clockwise motion)

          delay_ms(5000);             //delay is provided so that the barrier connected to motor reaches its
accurate position
          PORTB = 0x10;               //The last four bits are 0000 which stops the motor
          n=0x00;                     //n is assigned to 0x00

           //The time for which the gate is closed is indicated by LCD
          LCD_Initialize(); //Initialize
           Cursor_Position(1,3);
           LCD_Print("Please stop");        // Printing Hello at 1st row and 3rd columm
           Cursor_Position(2,5);
           LCD_Print("6");        // Printing World at 2nd Row and 5th column
           delay_ms(1000);
           Cursor_Position(2,5);
           LCD_Print("5");        // Printing World at 2nd Row and 5th column
           delay_ms(1000);
```

```
          Cursor_Position(2,5);
          LCD_Print("4");        // Printing World at 2nd Row and 5th column
          delay_ms(1000);
           Cursor_Position(2,5);
          LCD_Print("3");        // Printing World at 2nd Row and 5th column
          delay_ms(1000);
           Cursor_Position(2,5);
          LCD_Print("2");        // Printing World at 2nd Row and 5th column
          delay_ms(1000);
           Cursor_Position(2,5);
          LCD_Print("1");        // Printing World at 2nd Row and 5th column
          delay_ms(1000);
          Cursor_Position(2,5);
          LCD_Print("0");        // Printing World at 2nd Row and 5th column
          delay_ms(1000);
          Cursor_Position(1,3);
          LCD_Print("                    ");        // Printing Hello at 1st row and 3rd columm
          Cursor_Position(2,5);

          LCD_Print("GO");        // Printing World at 2nd Row and 5th column
          delay_ms(1000);
          PORTB.B4 =0;        // The red LED is turned off.
  }
  else if(n==0x02)            //case to open the barrier
  {

          PORTB.B5 =1;    //The green LED connected to PORTB.B5 glows which indicates that train has
departed.

         LCD_Initialize(); //Initialize
          Cursor_Position(1,3);
          LCD_Print("Safe ");        // Printing Hello at 1st row and 3rd columm
          Cursor_Position(2,5);
          LCD_Print("Journey ");        // Printing World at 2nd Row and 5th column
         OCR2=256;

         TCCR2 |= (1 << COM21);             // set none-inverting mode

         TCCR2 |= (1 << WGM21) | (1 << WGM20);   // set fast PWM Mode

         TCCR2 |= (1 << CS21);      // set prescaler to 8 and starts PWM
         PORTC = 0x01;               //enable for Motor
         PORTB = 0x25;               //The last four bits of PORTB are 0101 in which 2 bits are for first motor
and next 2 are for second motor (01 indicates anti clockwise motion)


           delay_ms(5000);            //delay is provided so that the barrier connected to motor reaches its
accurate position
           PORTB = 0x20;              //The last four bits are 0000 which stops the motor
           n=0x00;                    // n is assigned 0x00
           PORTB.B5 =0;               // The green LED is turned off.
  }


 if(n==0x03)       //Case for ultrasonic sensor
  {

   US_DDR|=(1<<US_POS);


       //Give the US pin a 15us High Pulse
       US_PORT|=(1<<US_POS);   //High
```

```
      US_PORT&=(~(1<<US_POS));//Low


      //Now make the pin input
      US_DDR&=(~(1<<US_POS));

      //Measure the width of pulse
      r=getPulseWidth();

      //Handle Errors
      if(r==US_ERROR)
     {

      }
      else  if(r==US_NO_OBSTACLE)
     {

      }
      else
      {

        int d;

        d=(r/58.0); //Convert to cm
        if(d<10)   //if the obstacle is at a distance of less than 10 cm
        {
           PORTB.B6 = 1;            //The buzzer which is connected to PORTB.B6 is turned on
           delay_ms(5000);          //The buzzer remains on for 5 seconds
           PORTB.B6 = 0;            //The buzzer which is connected to PORTB.B6 is turned off

        }
  }
      n=0x00;               //n is assigned to 0x00
 }
}
}
```

DESCRIPTION OF THE CODE

- INT0_vect() org 0x002 Is a function which is called when external interrupt request 0 occurs
- n is assigned 0x01 in the above function which is used in main function
- INT1_vect() org 0x004 is a function which is called when external interrupt request 1 occurs
- n is assigned 0x02 in the above function which is used in main function
- void LCD_data(unsigned char Data) //function to print a character in LCD
- PORTC is assigned Data&0xF0 which sends Higher nibble (D7-D4)
- LCD_RS is assigned 1 for Register Select =1 (for data select register)
- LCD_EN is assigned 1 Enable=1 for H to L pulse
- delay_us(5) provides a delay of 5 us
- LCD_EN is assigned 0 which makes Enable=0 for H to L pulse
- PORTCis assigned ((Data<<4)&0xF0) which sends Lower nibble (D3-D0)
- LCD_EN=1 is assigned 1 which makes Enable=1 for H to L pulse
- delay_us(5) provides a delay of 5 us
- LCD_EN=0 is assigned 0 which makes Enable=0 for H to L pulse

- delay_us(100) provides a delay of 100 us
- void LCD_Print(char * str) is a function for LCD Print
- unsigned char I is assigned to 0
- while((str[i])!=0) takes each character till NULL is reached
- LCD_data((str[i]) through this data is sent to LCD data register
- i is incremented
- delay_us(10) provides a delay of 10 us
- void lcdcommand(unsigned char command) is a function for LCD Command
- Assigning PORTC as command&0xF0 Sends Higher nibble (D7-D4)
- LCD_RS=0 is for Register Select =0 (for Command register)
- LCD_EN=1 is Enable=1 for H to L pulse
- delay_us(5) provides a delay of 5 us
- delay_us(100) provides a delay of 100 us
- Assigning PORTC=((command<<4)&0xF0) Sends Lower nibble (D3-D0)
- LCD_EN=1 is for Enable=1 for H to L pulse
- delay_us(5) provides a delay of 5 us
- delay_us(40) provides a delay of 40 us
- void Cursor_Position(unsigned short int x,unsigned short int y) is a function for cursor position
- unsigned char firstcharadd[] ={0x80,0xC0} makes First line address 0X80
- Second line address 0XC0
- lcdcommand((firstcharadd[x-1]+y-1)) lcd command function is called
- void clear() is a function to Clear the screen
- lcdcommand(0x01) lcd command 0x01
- delay_us(2) provides a delay of 2 us
- void LCD_Initialize() is a function for LCD Iniatialize
- LCD_EN is set to 0
-
- lcdCommand(0x33) is to Initialize LCD for 4 bit mode
- lcdCommand(0x32) is to Initialize LCD for 4 bit mode
- lcdCommand(0x28)is to Initialize LCD for 5X7 matrix mode
- lcdCommand(0x0E) is to Display on,cursor blinking
- lcdCommand(0x06) is to Shift cursor to right
- int getPulseWidth() is functoin to get the pulse width through ultrasonic sensor
- for(i=0;i<600000;i++) is loop to Wait for the rising edge
- if(!(US_PIN & (1<<US_POS))) continue; else break;
- if(i==600000) then return 0xffff it indicates Indicates time out
- TCCR1A is assigned to 0X00
- TCCR1B=(1<<CS11) for Prescaler = Fcpu/8
- TCNT1H=0x00 for Init counter
- for(i=0;i<600000;i++)
- if(US_PIN & (1<<US_POS))
- if(TCNT1H > 60000) break; else continue;
- if(i==600000) indicates time out
- return 0xffff; //Indicates time out
- Result is assigned TCNT1H
- TCCR1B is assigned 0x00.
- if(result > 60000) indicates no obstacle
- return 0xfffe; //No obstacle
- Else return (result>>1).
- In the main function following steps takes place
- SREG.B7 is assigned 1 for GLOBAL INTERRUPT
- GICR.B7 is assigned 1 To enable interrupt 1
- GICR.B6 is assigned 1 To enable interrupt 0
- GICR.B5 = 1 is assigned 1 to enable interrupt 2
- MCUCR is assigned 0x0F in which the last four bits indicates that in interrrupt 0 and 1 the interrupt is generated on rising edge
- MCUCSR is assigned 0x40 for Interrupt 2 to work on rising edge.
- n is initilally assigned to 0x00
- LCD_Initialize() is to Initialize LCD
- LCD_Print("Safe ") Prints Hello at 1st row and 3rd columm
- LCD_Print("Journey ") PrintsWorld at 2nd Row and 5th column
- while(1)//loop works continously

- if(PINA.B4==1) :PINA.B4 checks whether LDR sensor receives light or not when PINA.B4 is 1 it receives light and hence the LED connected to PORTA.B6 and PORTA.B6 are off
- else if(PINA.B4==0):PINA.B4 checks whether LDR sensor receives light or not when PINA.B4 is 0 it doesn't receive light and hence the LED connected to PORTA.B6 and PORTA.B6 turned on
- if(n==0x01) it is the case to close the barrier
- PORTB.B4 is assigned 1 due to which the red LED connected to PORTB.B4 glows which indicates that train is approaching.
- OCR2=256;
-
- TCCR2 |= (1 << COM21) is to set none-inverting mode
-
- TCCR2 |= (1 << WGM21) | (1 << WGM20) is to set fast PWM Mode
-
- TCCR2 |= (1 << CS21) is to set prescaler to 8 and starts PWM
- PORTC = 0x01 is enable for Motor
- PORTB = 0x1A The last four bits of PORTB are 1010 in which 2 bits are for first motor and next 2 are for second motor (10 indicates clockwise motion)
- delay is provided so that the barrier connected to motor reaches its accurate position
- PORTB = 0x01 stops the motor
- n is assigned to 0x00
- After the LCD is initialised it indicates the seconds till which the barrier will remain off
- else if(n==0x02) is a case to open the barrier
- PORTB.B5 is set to 1 due to whichThe green LED connected to PORTB.B5 glows which indicates that train has departed.
- OCR2=256;
-
- TCCR2 |= (1 << COM21) is to set none-inverting mode
-
- TCCR2 |= (1 << WGM21) | (1 << WGM20) is to set fast PWM Mode
-
- TCCR2 |= (1 << CS21) is to set prescaler to 8 and starts PWM
- PORTC = 0x01 is enable for Motor
- PORTB = 0x25 The last four bits of PORTB are 0101 in which 2 bits are for first motor and next 2 are for second motor (01 indicates anti clockwise motion)
- delay is provided so that the barrier connected to motor reaches its accurate position
- PORTB = 0x20 The last four bits are 0000 which stops the motor
- n is assigned 0x00
- PORTB.B5 =0 by assigning this the green LED is turned off.
- if(n==0x03) is a Case for ultrasonic sensor
- US_PORT|=(1<<US_POS) is to Give the US pin a 15us High Pulse
- US_PORT&=(~(1<<US_POS)) is to give a low pulse
- US_DDR&=(~(1<<US_POS)) makes the pin input
- r=getPulseWidth() measure the width of the pulse
- if(r==US_ERROR) is a case to handle errors
- else if(r==US_NO_OBSTACLE)
- Else (it is executed when there is no error
- d=(r/58.0) Converts to cm
- if(d<10) if the obstacle is at a distance of less than 10 cm
  - PORTB.B6 = 1; //The buzzer which is connected to PORTB.B6 is turned on
  - delay_ms(5000); //The buzzer remains on for 5 seconds
  - PORTB.B6 = 0; //The buzzer which is connected to PORTB.B6 is turned off
- n is assigned to 0x00

CODE OF BLUETOOTH

```
void usart_initialize()
{
 UCSRB=0x18; // tx 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

}
void usart_send(unsigned char ch)
{
  while(UCSRA.B5==0); // Wait till UDR is empty

 UDR=ch; //Write the value to be Tx
}


int getPulseWidth()          //code to get the pulse width through ultrasonic sensor
   {
      int i,result;

      //Wait for the rising edge
      for(i=0;i<600000;i++)
      {
        if(!(US_PIN & (1<<US_POS))) continue; else break;
     }

      if(i==600000)
        return 0xffff; //Indicates time out

      //High Edge Found

      //Setup Timer1
      TCCR1A=0X00;
      TCCR1B=(1<<CS11); //Prescaler = Fcpu/8
      TCNT1H=0x00;       //Init counter

      //Now wait for the falling edge
      for(i=0;i<600000;i++)
            {
        if(US_PIN & (1<<US_POS))
        {
          if(TCNT1H > 60000) break; else continue;
        }
        else
          break;
      }

      if(i==600000)
        return 0xffff;           //Indicates time out

      //Falling edge found

      result=TCNT1H;
```

```
    //Stop Timer
    TCCR1B=0x00;

    if(result > 60000)
      return 0xfffe;          //No obstacle
    else
      return (result>>1);
  }

  void Wait()
 {
   int i;
   for(i=0;i<10;i++)
     delay_ms(1000);
 }



void main() {

  char string;
DDRB.B3=1;
DDRB.B6=1;
DDRB.B7=1;
TCCR0=0x7D;  //set timer0 into ctc mode and clk prescaler= clk/1024
OCR0=0x00;   //count 0
usart_initialize();

while(1)
 {

   US_DDR|=(1<<US_POS);


      //Give the US pin a 15us High Pulse
      US_PORT|=(1<<US_POS);   //High


      US_PORT&=(~(1<<US_POS));//Low


      //Now make the pin input
      US_DDR&=(~(1<<US_POS));

      //Measure the width of pulse
      r=getPulseWidth();

      //Handle Errors
      if(r==US_ERROR)
     {

      }
      else  if(r==US_NO_OBSTACLE)
     {

      }
      else
      {

        int d;
```

```
          d=(r/58.0); //Convert to cm
          if(d<10)   //if the obstacle is at a distance of less than 10 cm
          {
             PORTB.B6 = 1;

         }
   }
  delay_ms(1000);

if(PORTB.B6)     //if PORTB.B6 is 1 then it means that an obstacle has been detected and hence a
message is sent
{
 usart_send('O');
 usart_send('B');
 usart_send('S');
 usart_send('T');
 usart_send('A');
 usart_send('C');
 usart_send('L');
 usart_send('E');
 usart_send(' ');
delay_ms(5000);
}
}
  }
}
```

DESCRIPTION OF THE BLUETOOTH CODE

- void usart_initialize() is a function to initialise usart
- UCSRB=0x18 is for tx Enable
- UCSRC=0x86 is for Data Size : 8-bit, Stop Bit:1,No parity
- UBRRL=0x33 is for 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
- void usart_send(unsigned char ch) is a function to send character
- while(UCSRA.B5==0) Waits till UDR is empty
- UDR=ch Writes the value to be Tx
- int getPulseWidth() is functoin to get the pulse width through ultrasonic sensor
- for(i=0;i<600000;i++) is loop to Wait for the rising edge
- if(!(US_PIN & (1<<US_POS))) continue; else break;
- if(i==600000) then return 0xffff it indicates Indicates time out
- TCCR1A is assigned to 0X00
- TCCR1B=(1<<CS11) for Prescaler = Fcpu/8
- TCNT1H=0x00 for Init counter
- for(i=0;i<600000;i++)
- if(US_PIN & (1<<US_POS))
- if(TCNT1H > 60000) break; else continue;
- if(i==600000) indicates time out
- return 0xffff; //Indicates time out
- Result is assigned TCNT1H
- TCCR1B is assigned 0x00.
- if(result > 60000) indicates no obstacle
- return 0xfffe; //No obstacle

- Else return (result>>1).
- if(n==0x03) is a Case for ultrasonic sensor
- US_PORT|=(1<<US_POS) is to Give the US pin a 15us High Pulse
- US_PORT&=(~(1<<US_POS)) is to give a low pulse
- US_DDR&=(~(1<<US_POS)) makes the pin input
- r=getPulseWidth() measure the width of the pulse
- if(r==US_ERROR) is a case to handle errors
- else if(r==US_NO_OBSTACLE)
- Else (it is executed when there is no error
- d=(r/58.0) Converts to cm
- if(d<10) if the obstacle is at a distance of less than 10 cm
- PORTB.B6 = 1 due to which the buzzer which is connected to PORTB.B6 is turned on
- delay_ms(5000) keeps the buzzer on for 5 seconds
- PORTB.B6 =0 due to which the buzzer which is connected to PORTB.B6 is turned off
- if(PORTB.B6) if PORTB.B6 is 1 then it means that an obstacle has been detected and hence a message is sent
- usart_send('O') usart_send is called to send ‘O’
- usart_send('B') usart_send is called to send ‘B’
- usart_send('S') usart_send is called to send ‘S’
- usart_send('T') usart_send is called to send ‘T’
- usart_send('A') usart_send is called to send ‘A’
- usart_send('C') usart_send is called to send ‘C’
- usart_send('L') usart_send is called to send ‘L’
- usart_send('E') usart_send is called to send ‘E’
- usart_send(' ') usart_send is called to send ‘ ’
- Delay of 5 seconds is provided.