DEPARTMENT OF PHYSICS, UNIVERSITY OF COLOMBO
PH3032-EMBEDDED SYSTEM LABORATORY

Mini Project Report

Gas Level Indicator and Leakage Detector

Name : I.E.Ranaweera

Index: s14365

Date: 14/08/2021

Abstract

This embedded system was gas level indicator and leakage detector. This project have two objectives. They are indicate the gas level in gas cylinder and also detects the gas leakage. The level of LPG is measured using the load sensor and it displayed on LCD. Gas level is display as a percentage of gas level in the gas cylinder. The gas level is measured every 30 minutes. The gas leakage is detected by the gas sensors (MQ-5). If gas leakage around the gas cylinder, a buzzer will be alarm. Then we can prevent the LPG gas burst accidents in the home. This system is for 2.3 kg gas cylinder. So, this project will be helped to all people who use gas cylinder.

TABLE OF CONTENTS

1 INTRODUCTION 6
2 THEORY 7
2.1 Atmega32A microcontroller 7
2.1.1 ADMUX- ADC Multiplexer Selection Register 7
2.1.2 ADCSRA- ADC Control and Status Register A 8
2.2 Interrupt in Microcontroller 8
2.2.1 External Interrupt 8
2.2.2 Internal Interrupt 11
2.3 Interrupt Service Routing 13
2.4 Loadcell (Weight sensor) 14
2.5 Gas sensor (MQ-5 gas sensor) 15
3 DIAGRAMS 16
3.1 Block Diagram 16
3.2 Circuit Diagram 17
4 APPARATUS AND ACCESSORIES 18
5 METHODOLOGY 22
6 RESULTS AND ANALYSIS 28
7 DISCUSSION 30
REFERENCE 31
APPENDIX 32

List of Figures

Figure 2. 1 - The pin diagram of ATmega32A microcontroller 7

Figure 2.1. 1 - ADMUX Register bit 7
Figure 2.1. 2- ADCSRA Register bit 8

Figure 2.2.1. 1- Bits of GICR Register 9
Figure 2.2.1. 2– Bits of MCURC Register 9
Figure 2.2.1. 3– Bits of MCUCSR Register 10

Figure 2.2.2. 1– Bits of TCCR0 Register 11
Figure 2.2.2. 2– Bits of OCR0 Register 13
Figure 2.3- Execution of program 14

Figure 2.4. 1-Loadcell 5kg 15
Figure 2.4. 2- Full bridge stain gauge circuit 15

Figure 2.5. 1- MQ-5 Gas sensor 15

Figure 3. 1 block diagram 16
Figure 3. 2 circuit diagram 17

Figure 5. 1 - Schematic diagram in Protues software 22
Figure 5. 2 circuit diagram for 12 V to 5 V voltage convertor 23
Figure 5. 3 – 230 V to 5 V voltage convertor 23
Figure 5. 4 – loadcell in the setup 26
Figure 5. 5 – The setup of the System 26
Figure 5. 6 – Circuit of the system (in Vero board) 27

Figure 6. 1 – LCD display of the system 28
Figure 6. 2 – The graph of ADC value Vs Weight 28
Figure 6.3 - Front view of the setup…………………………………………… *29*
Figure 6. 4- above view of the setup 29

List of Tables

Table 2.2.1. 1- Trigger level or edge of external interrupt..10
Table 2.2.1. 2- Trigger level or edge of external interrupt..10
Table 2.2.1. 3- Trigger edge of external interrupt...11

Table 2.2.2. 1- Modes of timer/counter0 interrupt 12
Table 2.2.2. 2- Clock source of timer/counter0 12

Table 4. 1 – Component for the System 18
Table 4. 2 – Component for the power supply..20
Table 4. 3 – Other components...20

Table 5. 1 - ATmega32 Pin out Configuration 22

1 INTRODUCTION

LP gas is one of the most widely used fuel in normal life. LP gas is stored in a variety of sizes cylinder. They are 2.3 kg, 5.5 kg, 12.9 kg and etc. One of the problems we face in our daily lives is the inability to measure the amount of gas in the gas cylinder. Now a days, there are some gas level indicators that need to be fitted to the gas cylinder. When gas cylinder is replaced, it must be removed. So it was very annoying.

To overcome those problems with some protection, this mini project introducing the gas level indicator and leakage detector. This gas level indicator displays the gas level as a percentage on the LCD display. The gas level is calculated every 30 minutes. If the button is pressed, the LCD display on and displayed it.

If a gas leakage happens around the gas tank, a gas sensor will detect the leakage. As soon as the leakage is detected, a buzzer will alarm the user about it.

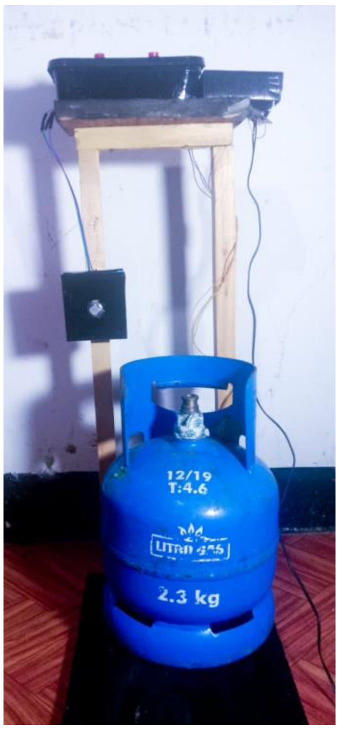


Figure 1. 1 Gas Level Indicator and Leakage Detector

2 THEORY

2.1 Atmega32A microcontroller

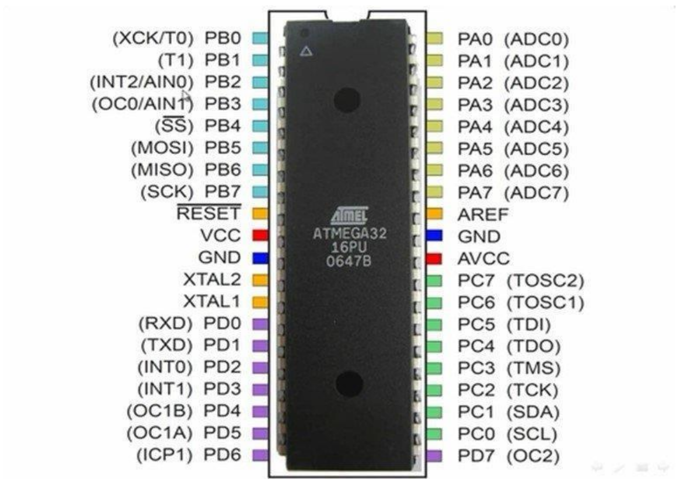
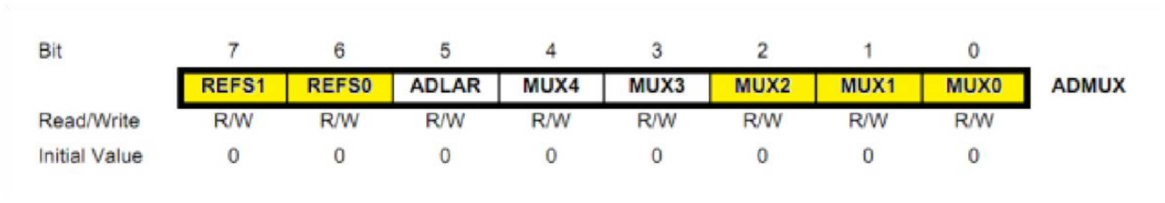


Figure 2. 1 - The pin diagram of ATmega32A microcontroller

ATmega32A microcontroller is a low power, high-performance microchip CMOS 8-bit based on AVR enhanced RISC. Program Memory type is flash. ATmega32A has 4 PORTS which are PORTA, PORTB, PORTC and PORTD with 32 I/O pins. PORTA is used for digital-to-analog converter. There are 32 registers associate with ALU in this microcontroller. Four of those registers are associate with ADC function. They are,

- ADMUX- ADC Multiplexer Selection Register
- ADCSRA- ADC Control and Status Register A
- ADCL and ADCH- ADC data Registers
- SFIOR- Special Function I/O Register

2.1.1 ADMUX- ADC Multiplexer Selection Register



Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

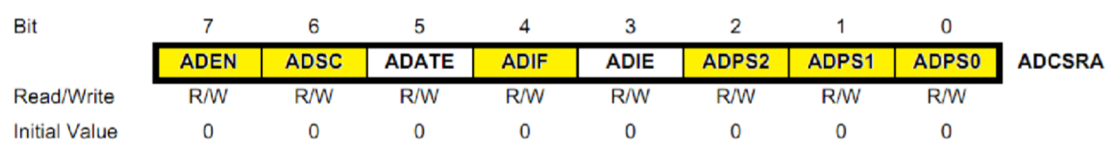
Figure 2.1. 1 - ADMUX Register bit

- Bit 7:6- Reference selection bit

- Bit 5- ADLAR – Left Adjust Result
- Bit 4:0- Analog channel and Gain Selection bit

REFS1 and REFS2 are used to select the reference voltage of the ADC. ADLR is done by adjusting result the ADC data register to the left, to access 10 bit resolution. Bit 4:0 is used to select gain for different channels.

2.1.2 ADCSRA- ADC Control and Status Register A



Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.1. 2- ADCSRA Register bit

- Bit 7- ADC enable
- Bit 6- ADC start conversion
- Bit 5- ADC auto trigger enable
- Bit4- ADC interrupt flag
- Bit3- ADC interrupt enable
- Bit2:0- ADC percale select bits

2.2 Interrupt in Microcontroller

An Interrupt is a signal emitted by hardware or software that requires immediate attention. It is the temporary suspension of the main program, giving control to external sources and resuming the main program from where it left off after its activation. There two types of interrupts. These are external interrupt and internal interrupt.

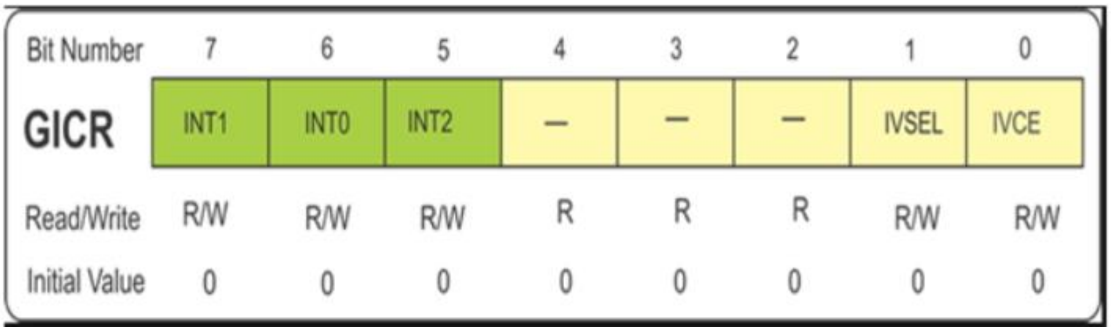
2.2.1 External Interrupt

External interrupt is an electronic signal sent to the processor from an external device. As an example press a button, press a switch and *etc*.

ATmega32A has three external interrupt pins which are PD2, PD3 and PB2. They are referred to as INT0 and INT1 and INT2 respectively. There several registers for external interrupts. They as follows.

- GICR Register
- MCUCR (MCU Control Register)
- MCUCSR (MCU Control and Status Register)

GICR Register



Bit Number	7	6	5	4	3	2	1	0
GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Figure 2.2.1. 1- Bits of GICR Register

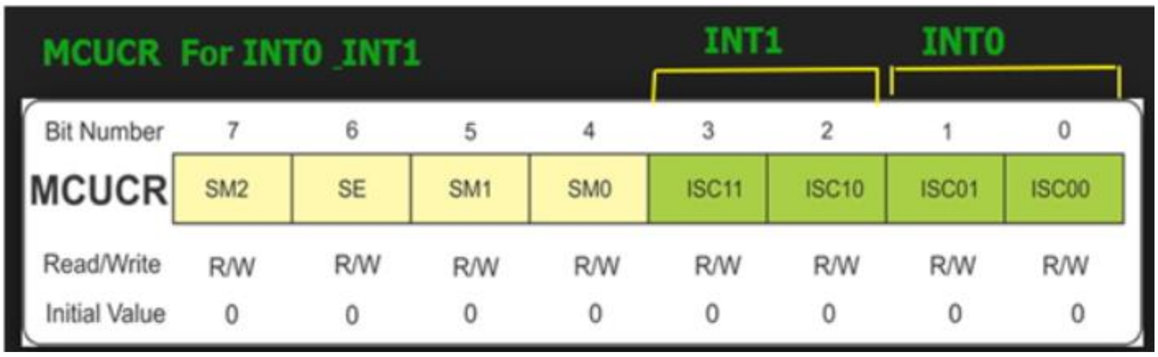
This register is used to enable or disable external interrupts.

Bit-7 : INT1- external interrupt enable/disable (0= disable interrupt , 1= enable interrupt).

Bit-6 : INT0- external interrupt enable/disable (0= disable interrupt , 1= enable interrupt).

Bit-5 : INT2- external interrupt enable/disable (0= disable interrupt , 1= enable interrupt).

MCUCR (MCU Control Register)



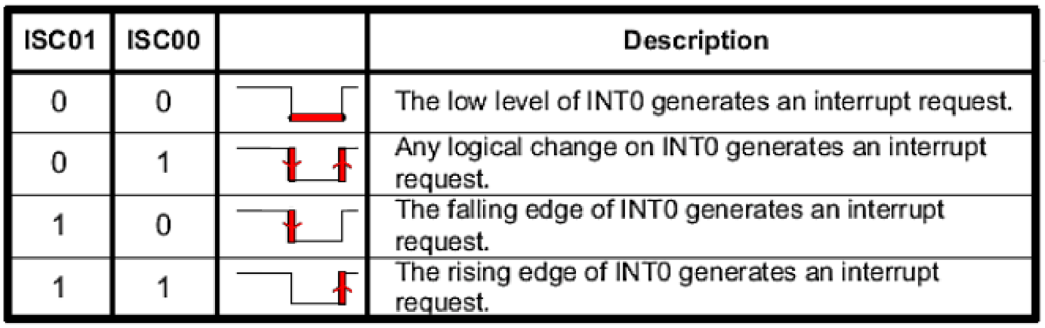
MCUCR For INT0 INT1

					INT1		INT0	
Bit Number	7	6	5	4	3	2	1	0
MCUCR	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Figure 2.2.1. 2– Bits of MCURC Register

This register is used to define a level or edge trigger in external interrupts of INT0 and INT1 pins. ISC11 and ISC10 bits are represented INT1 pin. ISC01 and ISC00 bits are represented INT0 pin. Therese bits are defined as follows.

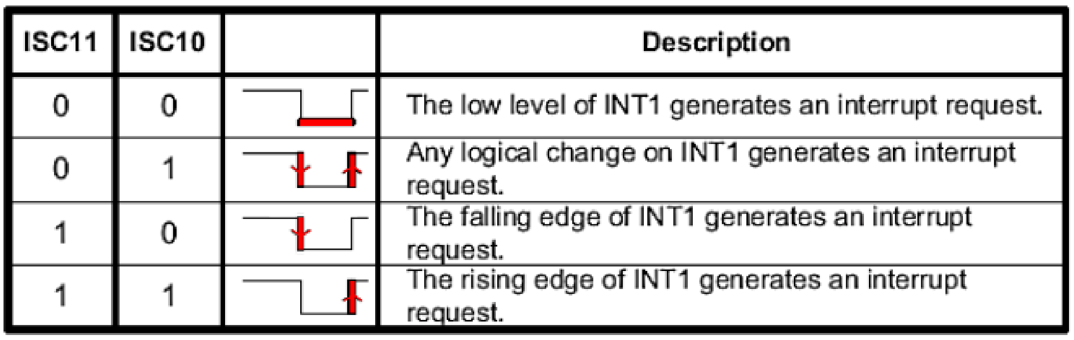
Table 2.2.1. 1- Trigger level or edge of external interrupt



ISC01	ISC00		Description
0	0		The low level of INT0 generates an interrupt request.
0	1		Any logical change on INT0 generates an interrupt request.
1	0		The falling edge of INT0 generates an interrupt request.
1	1		The rising edge of INT0 generates an interrupt request.

ISC11 and ISC10 bits are defined level or edge that selected the INT1 pin

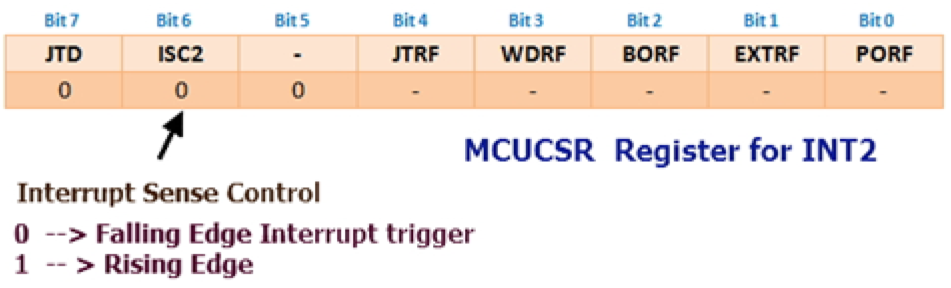
Table 2.2.1. 2- Trigger level or edge of external interrupt



ISC11	ISC10		Description
0	0		The low level of INT1 generates an interrupt request.
0	1		Any logical change on INT1 generates an interrupt request.
1	0		The falling edge of INT1 generates an interrupt request.
1	1		The rising edge of INT1 generates an interrupt request.

ISC11 and ISC10 bits are defined level or edge that selected the INT1 pin

MCUCSR (MCU Control and Status Register

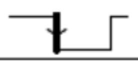
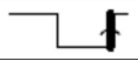


Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF
0	0	0	-	-	-	-	-

Figure 2.2.1. 3– Bits of MCUCSR Register

This register is used to define an edge trigger in external interrupts of INT2 pins.

Table 2.2.1. 3- Trigger edge of external interrupt

ISC2		Description
0		The falling edge of INT2 generates an interrupt request.
1		The rising edge of INT2 generates an interrupt request.

ISC2 bit is defined as an edge that selected the INT2 pin

2.2.2 Internal Interrupt

An internal interrupt is an interruption of the processor to execute a specific condition or instruction. Internal interrupts are Timer, UART, SPI, ADC, etc. A timer is clocked with periodic signal and a counter is clocked with the event. It has three types of interrupt. These are,

- Timmer0/counter0: 8-bit counter
- Timmer1/counter1: 16-bit counter
- Timmer2/counter2: 8-bit counter

<u>Timmer0/counter0</u>

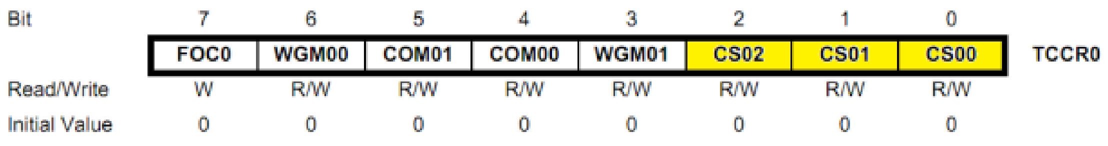
There several registers for Timmer0/counter0 interrupts. They as follows.

- TCNT0 – Timer/Counter Register 0
- TCCR0 – Timer/Counter Control Register 0
- TIMSK – Timer Counter Interrupt Mask
- OCR0 – Output Compare Register

TCNT0 – Timer/Counter Register 0

It is an 8-bit register that is used to counts up with each pulse.

TCCR0 – Timer/Counter Control Register 0



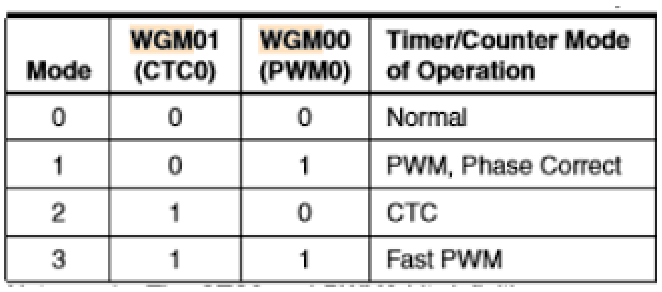
Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.2.2. 1– Bits of TCCR0 Register

This register is used to define for the operation mode and Prescaler selection.

- Bit 7: FOC0 (Force Output Compare) – this bit is used to determine the effect of the forced comparison. This bit always read as 0.
- Bit 6 and Bit 3: (WGM00 and WGM01) Waveform Generate Mode. These bits are used to select the mode of wave generation.

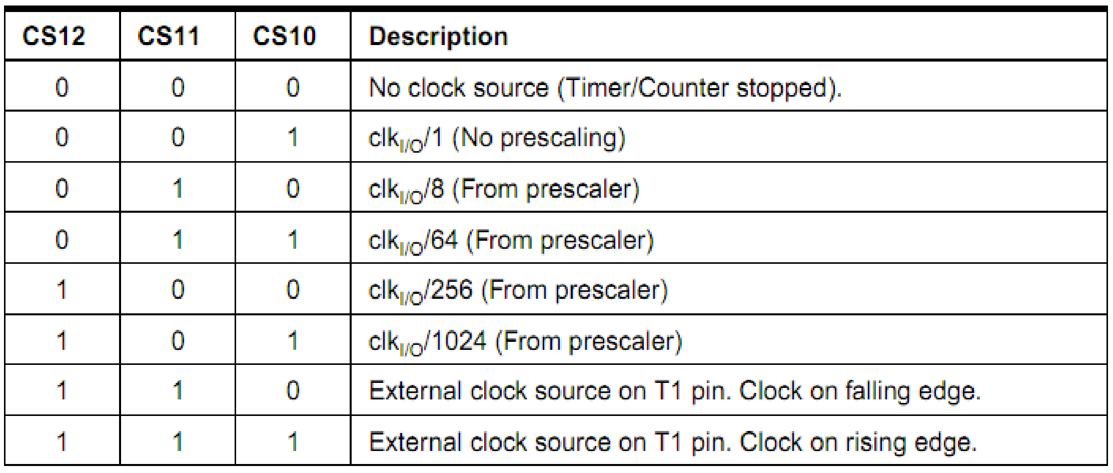
Table 2.2.2. 1- Modes of timer/counter0 interrupt



Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation
0	0	0	Normal
1	0	1	PWM, Phase Correct
2	1	0	CTC
3	1	1	Fast PWM

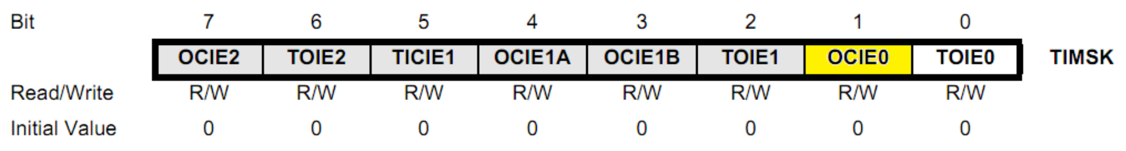
- Bit 5 and Bit 4 : (COM01 and COM00) Compare Mode Output Mode – this bit is controlled by the wave generator.
- Bit 2:0 - (Clock Source Select) - these bits are used to select the clock source (Prescaler) and start the timer. That bits are defined as follows.

Table 2.2.2. 2- Clock source of timer/counter0



CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

TIMSK – Timer Counter Interrupt Mask



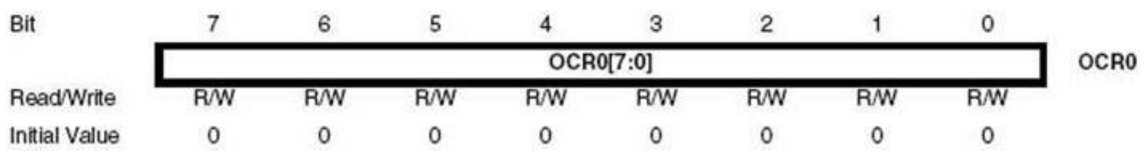
Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.1.1.5 – Bits of TIMSK Register

This register is mostly used to enable the compare match interrupt mode and the compare overflow interrupt mode.

Bit 1: OCIE (Timer/counter0 Output Compare Match Interrupt Enable) –this bit used to enable compare match interrupt.
Bit 0 : TOIE0 (Timer/counter0 Overflow Interrupt Enable) - this bit is used to enable overflow interrupt.

OCR0 – Output Compare Register



Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figure 2.2.2. 2– Bits of OCR0 Register

This register is an 8-bit register which used to continuously compare with the counter value of TCNT0. If they will be matched overflow the flag or toggle IO pins or clear TCNT0.

2.3 Interrupt Service Routing

Every interrupt must have an interrupt service routine. When an interruption occurs, the microcontroller processes the interrupt service routine. For it, every interrupt has a fixed location in the memory. That memory location is retained by interrupt vectors. They are called interrupted vectors. There are many different vectors for microcontrollers and interrupts. The following are some examples of such interrupt vectors.

- **INT0_vect**

This vector is used for external interrupt 0. That means, the external device is connected with the PD2 pin for doing interrupt.

- **TIMER0_OVF_vect**

This vector is used for internal interrupt time/counter0. it is handled when the timer/counter0(TCNT0) overflow. That means, it mange the uptime of the microcontroller. As an example, the TCNT0 is an 8-bit timer. It overflows at 256.

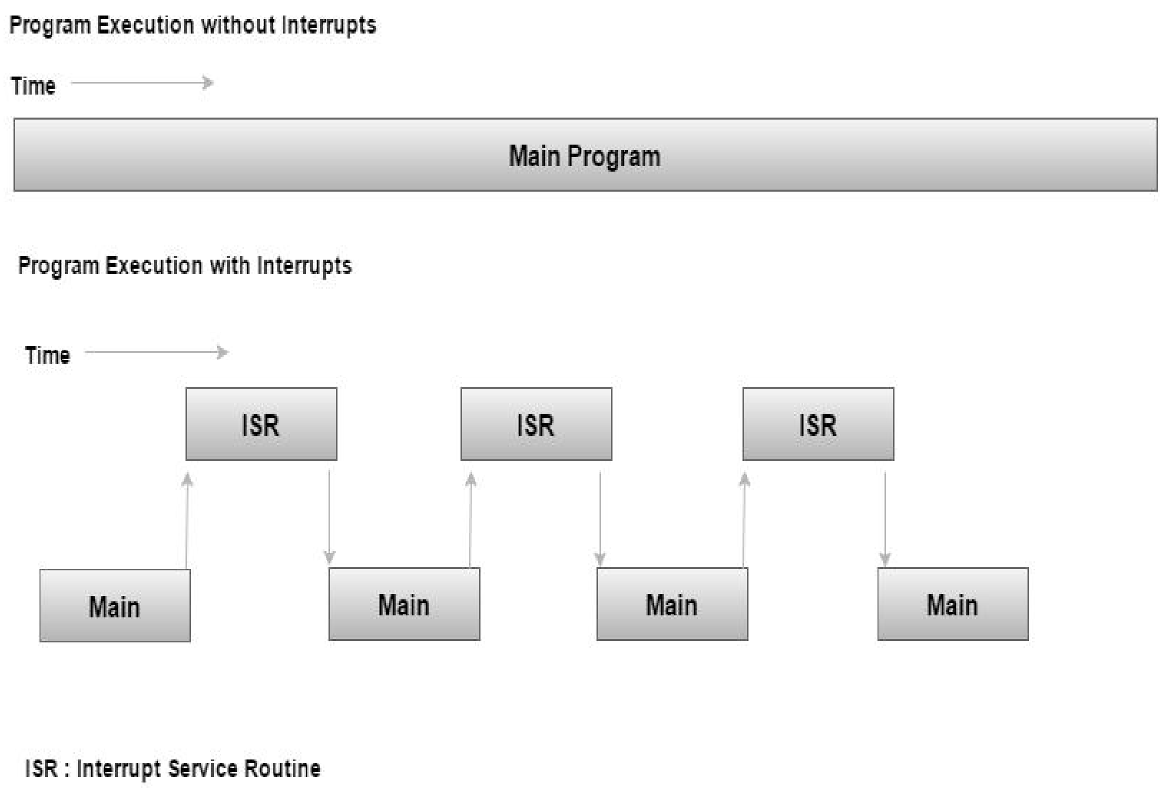

Program Execution without Interrupts
Time
Main Program
Program Execution with Interrupts
Time
ISR
ISR
ISR
Main
Main
Main
Main
ISR : Interrupt Service Routine

Figure 2.2.2. 3- Execution of program

2.4 Loadcell (Weight sensor)

A loadcell is a device that converts a force or weight acting on it into an electronic signal. The electronic signals here are causes by a change in voltage. Here, when a load act on the loadcell, there is tension at one end and compression at the other end. This causes a strain on the load on the load cell against the elastic. The strain gauge converts this strain into an electrical signal. The Weinstein Bridge is used to make the difference in resistance proportional to the weight. The output voltage varies by mV, due to the change in resistance.

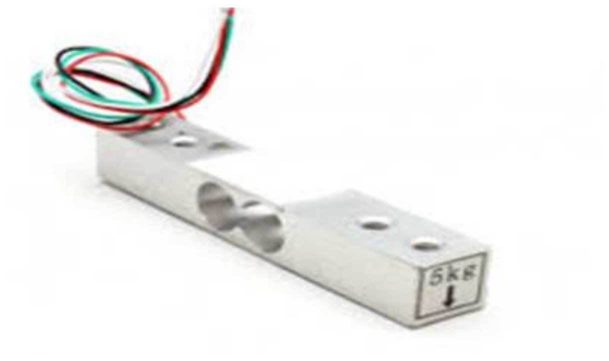


Figure 2.4. 1-Loadcell 5kg

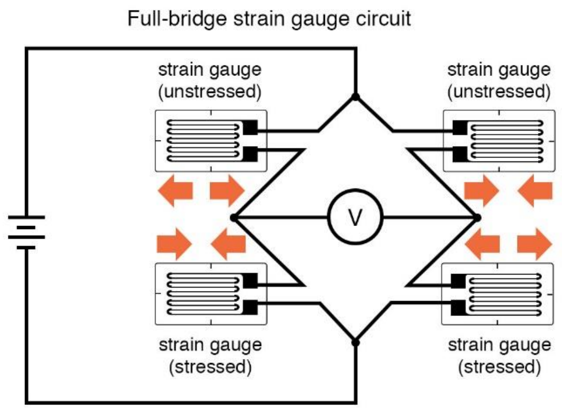

Full-bridge strain gauge circuit
strain gauge (unstressed)
strain gauge (unstressed)
V
strain gauge (stressed)
strain gauge (stressed)

Figure 2.4. 2- Full bridge stain gauge circuit

2.5 Gas sensor (MQ-5 gas sensor)

Gas sensor is an electronic component which can detector or measure gases. MQ-5 gas sensor is measured and detect LPG, CH_4 and C_3H_8. When gas leak, the heating element inside MQ5 gets heated up and output voltage varies (voltage is LOW = 0 - digital reading). The other time voltage is HIGH (1- digital reading).

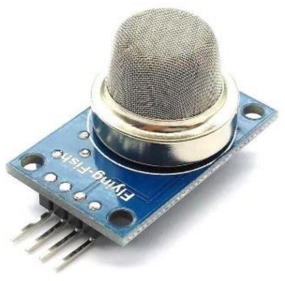


Figure 2.5. 1- MQ-5 Gas sensor

3 DIAGRAMS

3.1 Block Diagram

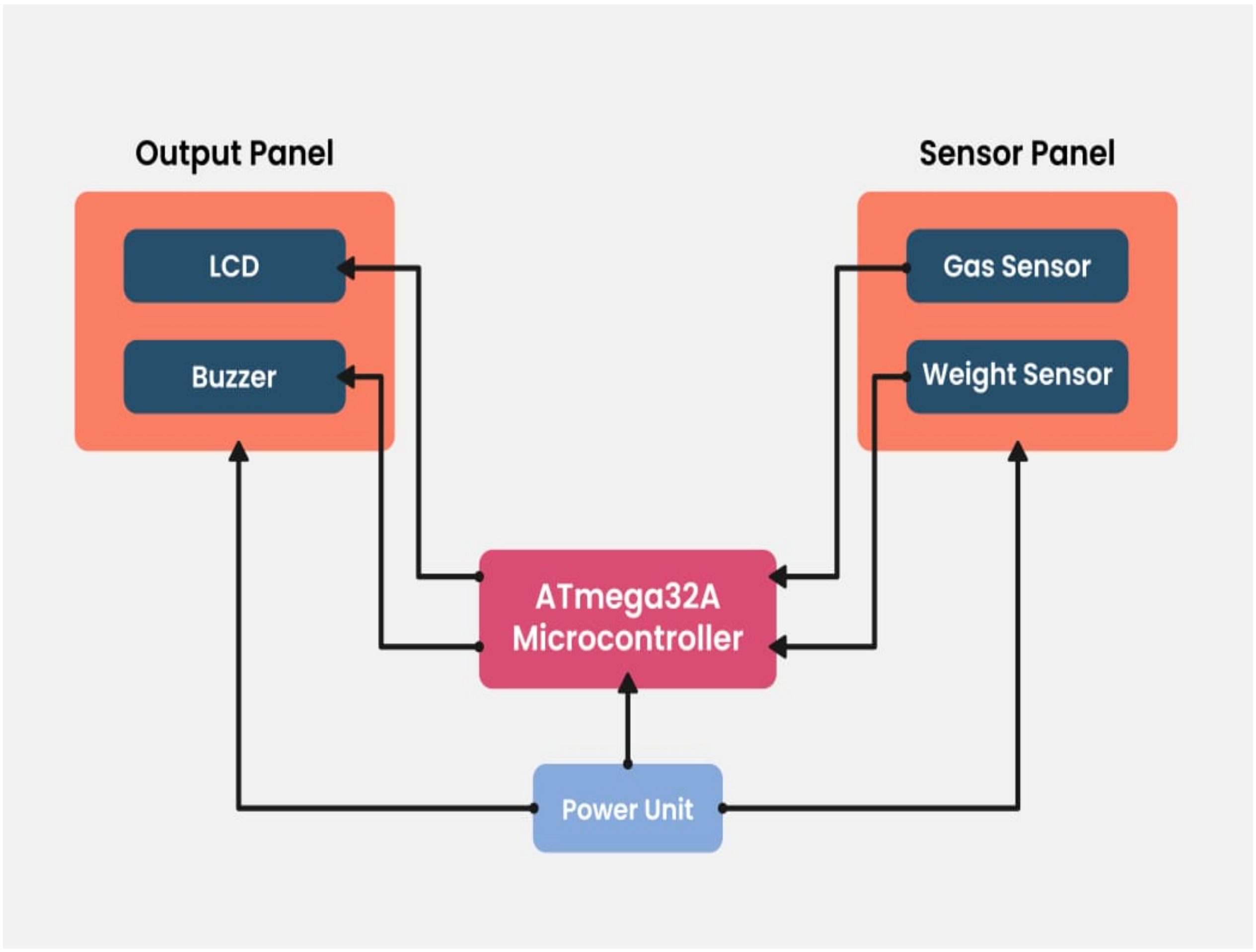


Figure 3. 1 block diagram

3.2 Circuit Diagram

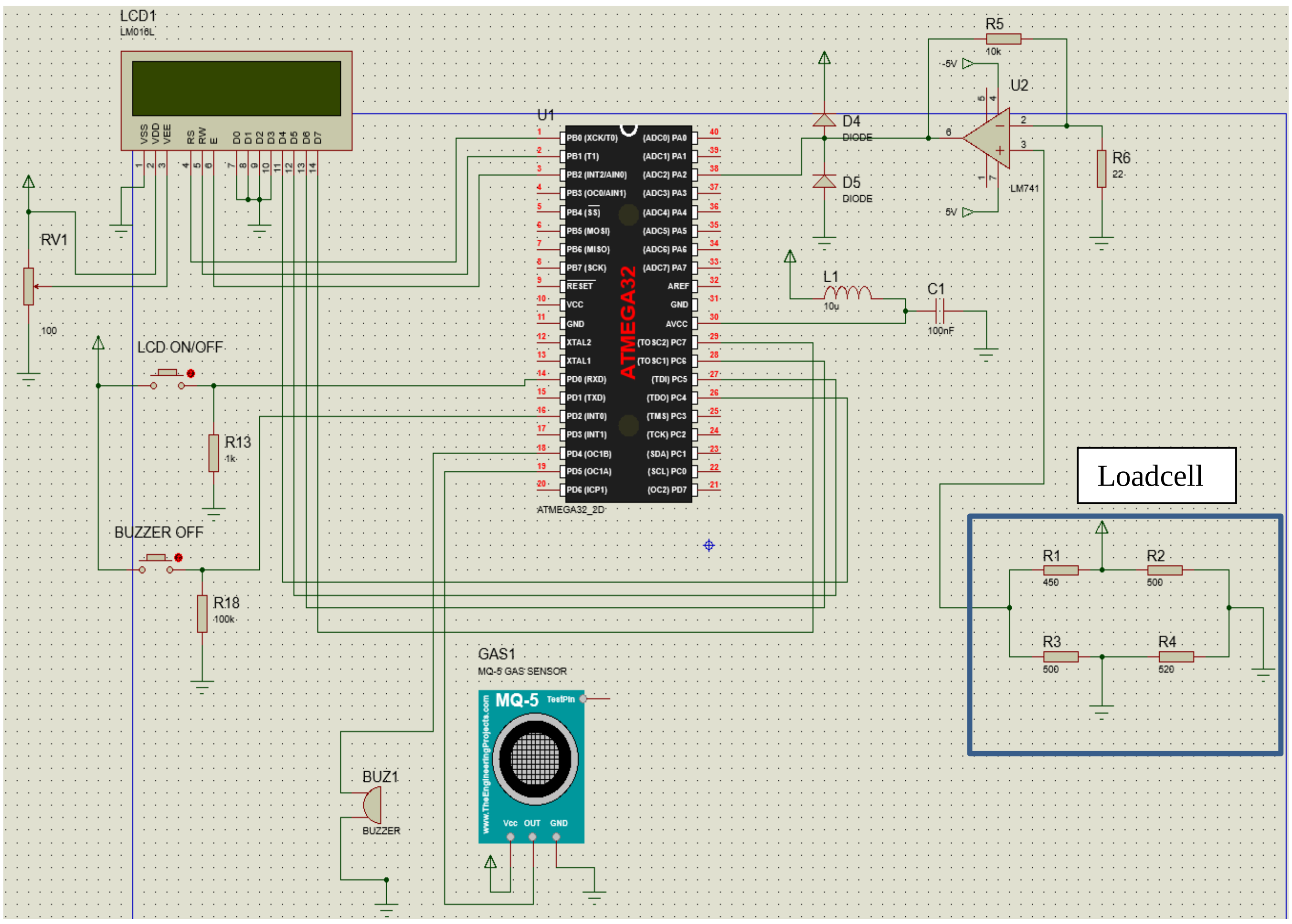
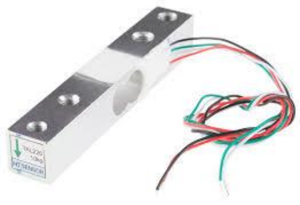
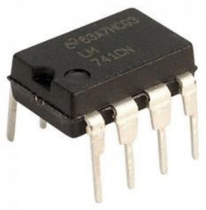


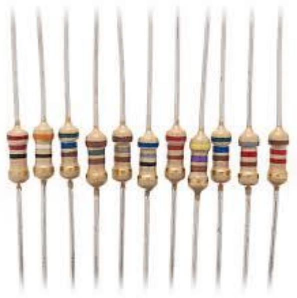
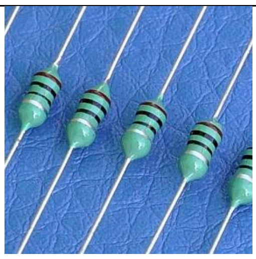
Figure 3. 2 circuit diagram

4 APPARATUS AND ACCESSORIES

Component for the System

Table 4. 1 – Component for the System

Component name	Component image	Quantity
Atmega32a Microcontroller		1
16×2 LCD Display		1
Gas sensor (MQ-5)		1
Loadcell (5 kg)		1
LM741 OpAmp IC		1
Buzzer		1

Capacitors		100 nF - 1 Ceramic capacitor – 1
Resistors		330 Ohm – 2 10k - 1 22 Ohm- 1
10 UH inductor		1
Diode		2
Variable resistor (10k)		1
Push button		2

Component for the power supply

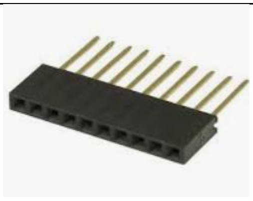
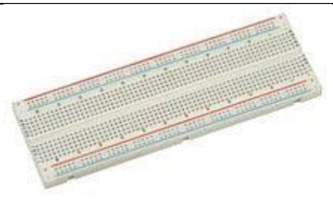
Table 4. 2 – Component for the power supply

Component name	Component image	Quantity
230V – 12V Adapter		1
LM7805		1
Capacitors		Ceramic capacitor (0.1 μF)- 2 470 μF – 2

Other Component

Table 4. 3 – Other components

Component name	Component image	Quantity
Development board (mini)		1

USBasp with cable		1
Wires		50
Header pin connector		4
Bread board		1

5 METHODOLOGY

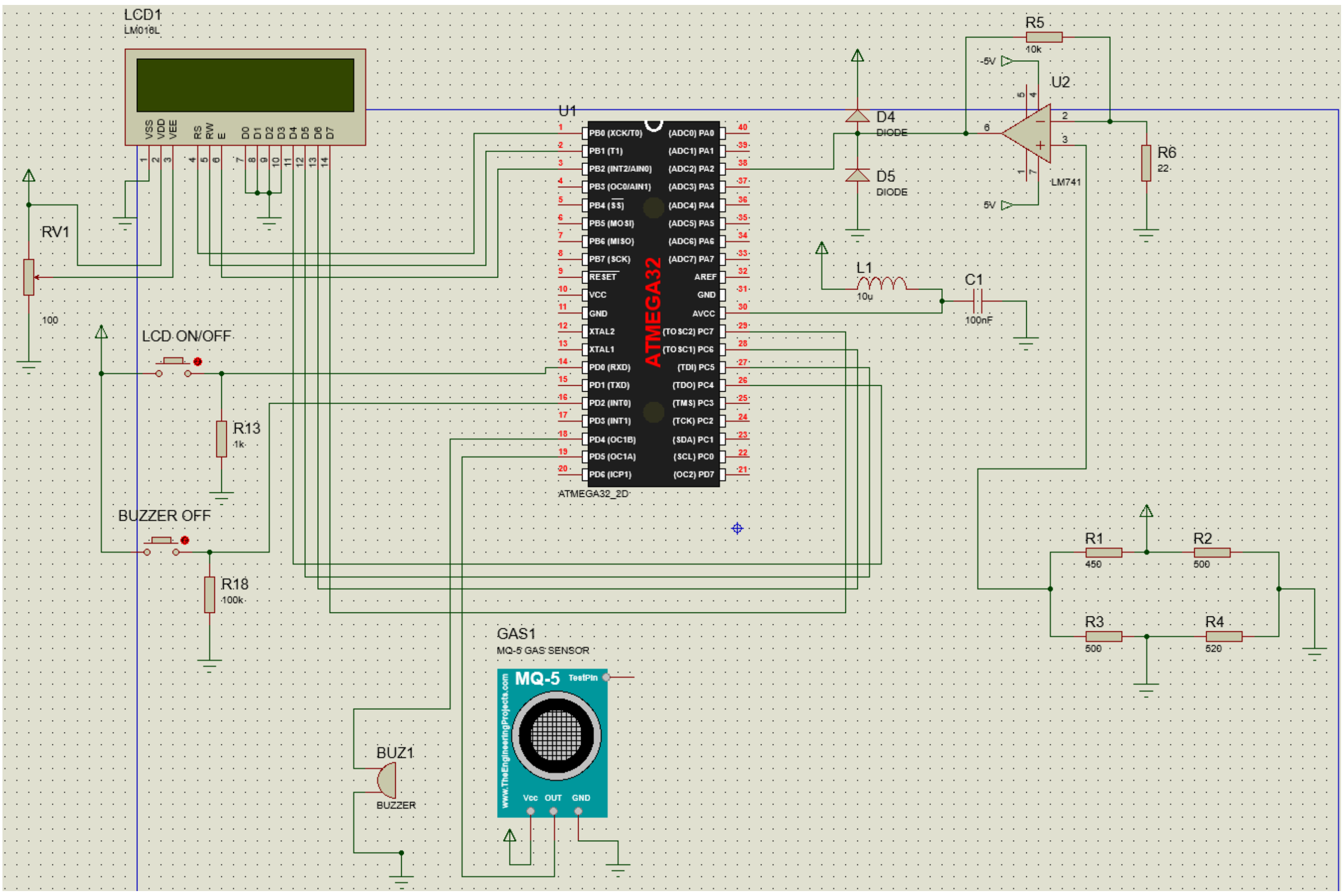


Figure 5. 1 - Schematic diagram in Protues software

- Firstly, the circuit was designed using Proteus Software as Figure 5.1 shown.
- The pin of the microcontroller is used as follows.

Table 5. 1 - ATmega32 Pin out Configuration

Pin	Description
PA2	Used to read analog value(voltage) from loadcell
PC4 – PC7	Used to output LCD data pins
PD5	Used to get input from gas sensor
PD4	Used to output the buzzer
PD2	Used to external interrupt
PD0	Used to connect the button for LCD on/off
PB0-PB2	Used to connect the control pins in LCD

1. All components were connected by using breadboard as Figure 5.1 shown.

2. Secondly, the power pack (power supply- 230V – 5V) was created as follows.

   - 12V – 5V voltage converter was created using two ceramic capacitors (0.1 μF), 470 μF two capacitor and LM705 regulator (on dote board).

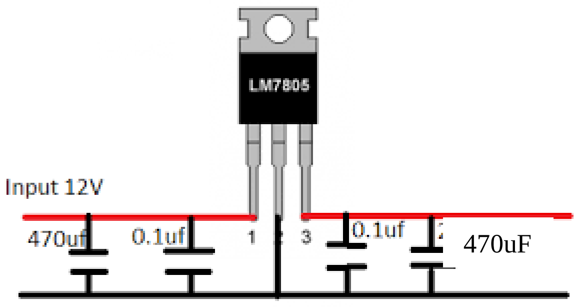


Figure 5. 2 circuit diagram for 12 V to 5 V voltage convertor

   - It was connected to the 230V -12V adapter.

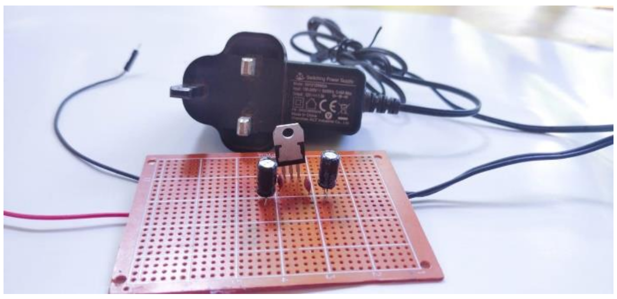


Figure 5. 3 – 230 V to 5 V voltage convertor

3. The code was written by using programmer's notepad software. The flow of the code is as follows.

   - Firstly, the needed libraries were included and PORTs were defined.
   - Next the functions were prototyped and initialized the variable.
   - Then initialized the internal interrupt's registers and external interrupt registers in main method.

```
 sei();

TCCR0 = 1<<WGM01 |1<<CS01 |1<<CS00;   //pre scler set 64
TIMSK = 1<<OCIE0;                       // every 1ms cll the interupt
OCR0= 124;
TCNT0=0;

GICR |= 1<<INT0;     // button of buzzer off set as external inturrupt
MCUCR |= 1<<ISC01 | 1<<ISC00;     // rising edge
```

- ❖ In this project, two interrupts were used. Firstly, the interrupt service enabled the calling sei() function.
- ❖ The clock pulse was starting using TCCR0 register.it set to enable the clear time to compare match mode and pre-scalar as clk/64.
- ❖ Then, TIMSK register was set for output compare match interrupt. Also, OCR0 register was set to 124. That means, it ORC0 and TCNT0 are matched at 124.
- ❖ GICR register was set to enable the INT0 external interrupts. Also, MCUCR register was used to defined rising edge that selected for INT0 pin.

- In while loop, the code was written for button press and gas sensor.

  - If gas leakage happen around the gas cylinder, the gas sensor digital pin was LOW. At that time buzzer will turn on.

```
   if ((PIND & (1<<5)) == 0 ){    // gas senseor digital pin in PD5
            PORTD|=(1<<4);                  //turn on the buzzer
       }
```

  - If the LCD ON/OFF button press once, LCD was on and display the gas level.

```
        if ((PIND & (1<<0))){// LCD ON/OFF buttton presss(hold on 
buuton)
             DDRC = 0xFF;
             DDRB = 0xFF;

             Lcd4_Init();
             Lcd4_Set_Cursor(1,1);
             Lcd4_Write_String("Gas Level Is");

             Lcd4_Set_Cursor(2,4);
             itoa(pecentage, buffer, 10)//display the gas lvel as pecentage
             Lcd4_Write_String(buffer);

             Lcd4_Set_Cursor(2,7);
             Lcd4_Write_String("%");

         }else{
             PORTC = 0x00;
             PORTB = 0x00;
         }
```

- The functions of the code as follows.

  - Interrupt service routing vectors were written to do Interrupts. This ISR(TIMER0_COMP_vect) interrupt was running the timer0. Every 30 min gas level is calculated.

```
ISR(TIMER0_COMP_vect){
    if(++Val_ms==100){
        Val_ms=0;
        if(++ms==10){
            ms=0;
            if(++sec==60){
                sec=0;
```

```
                    if(++mini==30){
                        mini=0;
                        pecentage = Weight();
                    }
                }
            }
        }
}
```

- ISR(INT0_vect) Vector was written to call the external interrupt, when the BUZZER OFF button pressed.

```
ISR(INT0_vect){
    PORTD &= ~(1<<4); //buzzerr off
}
```

- init_adc() method was used to select reference voltage(Using ADMUX register) ,ADEN set as 1 to ADC enable and ADPS2, ADPS1 set as 1 to assign factor as 64 in ADCSRA register.

```
void init_adc(void){
    DDRA &= ~(1<<2);        /* Make ADC port pinA2 as input */
    ADMUX = 1<< REFS0;
    ADCSRA = 1<<ADEN | 1<<ADPS2 | 1<<ADPS1; /* make pre scaler in 
64...*/
}
```

- read_adc() method was used to convert analog signal to digital signal and read ADC, ADSC set as 1 to ADC start conversion in ADCSA register.

```
unsigned int read_adc(unsigned char channel){

    ADMUX = ADMUX | (channel & 0x0f);     // set input chanel to read
    ADCSRA = ADCSRA | 1<<ADSC;            // start convercation
    while(ADCSRA & (1<<ADSC)){    //monitor end of convercasion intereupt
    }
    return ADC;
}
```

- weight() method was used to calculate the weight and indicate the gas level as percentage using weight.

```
unsigned int Weight(void){
    ad_val= read_adc(2);
    Voltage=((ad_val*5)/1024)*10000;

    z = ((Voltage- x)*100)/(y-x);
    return z;
}
```

4. Then, checked that the circuit is correct by uploading the code.

5. Next, the setup was created as figure 5.4 shown.

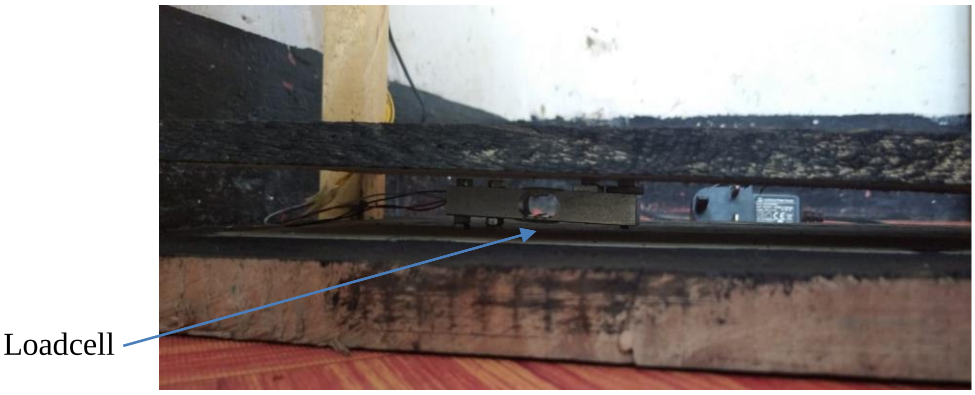


Figure 5. 4 – Loadcell in the setup

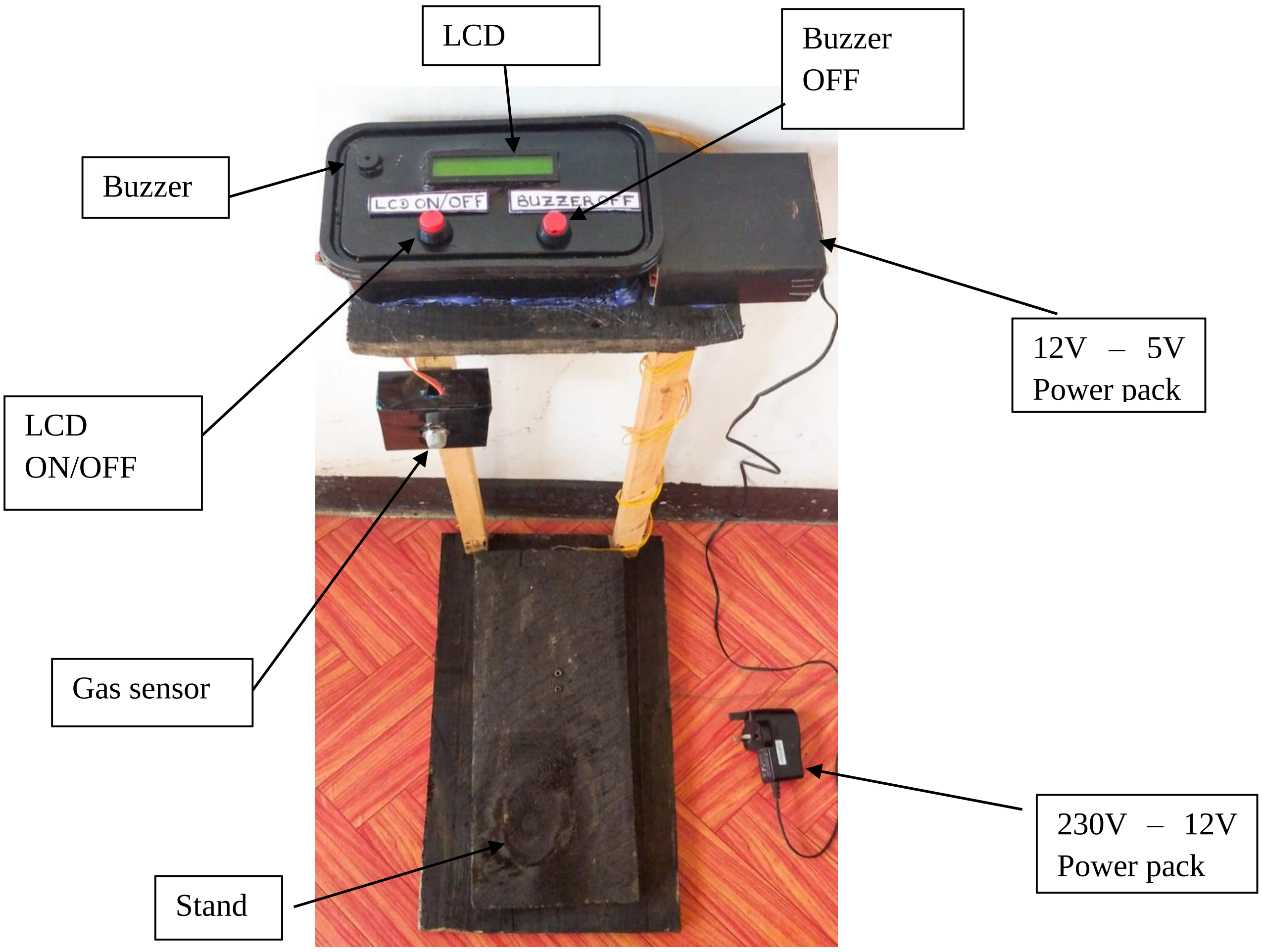


Figure 5. 5 – The setup of the System

6. Circuit was constructed in the Vero board as follows (soldering components).

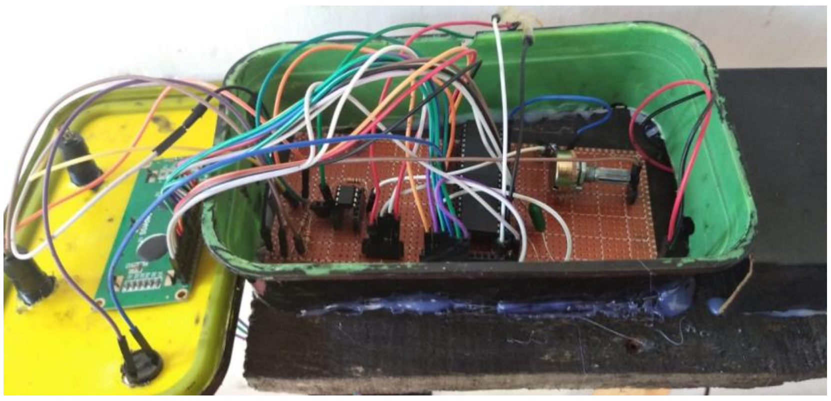


Figure 5. 6 – Circuit of the system (in Vero board)

7. Then, the loadcell was calibrated.
    - An empty gas cylinder (2.3kg) was placed on the stand and its ADC value was read.
    - A gas cylinder with gas (2.3kg) was placed on the stand and its ADC value was read.
    - Then, the scale was calibrated and edited code for get gas level as a percentage.

8. Finally, the gas tank was placed on the stand and read the gas level.

6 RESULTS AND ANALYSIS

Gas Level Indicator

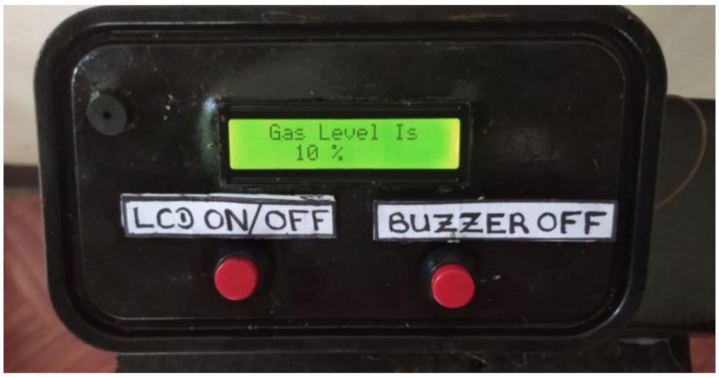


Figure 6. 1 – LCD display of the system

Gas level is represented as percentage. The gas level is calculated every 30 minutes.

The Weinstein Bridge is used to make the difference in resistance proportional to the weight. The output voltage varies by mV, due to the change in resistance. That voltage is amplified by using LM741 Op-Amp Ic. Then that value read by using PA2 pin. These values vary linearly.

- ADC reading for empty gas cylinder → 107
- ADC reading for full gas cylinder → 151

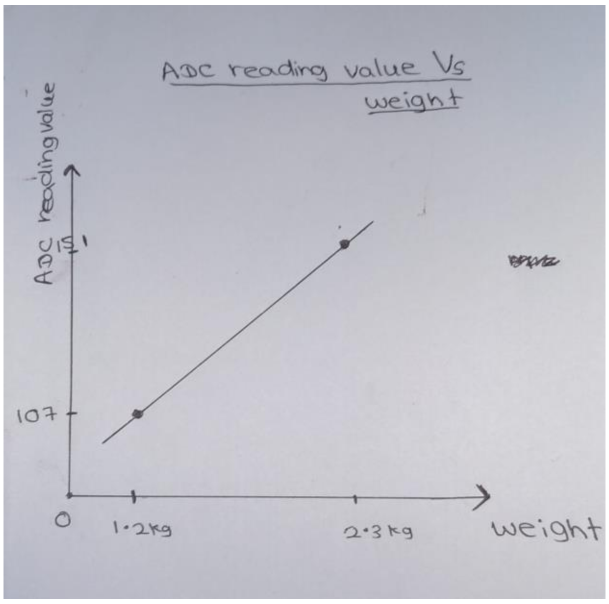


Figure 6. 2 – The graph of ADC value Vs Weight

So, this chart can be used to find the weight relative to an ADC value read using the gradient. From it, the gas level in the gas cylinder can be found as a percentage.

$$Z = \frac{(ADC\ reading - X) \times 100}{Y - X}$$

Z= the gas level as percentage
X= ADC reading for empty gas cylinder $\rightarrow$ 107
Y= ADC reading for full gas cylinder $\rightarrow$ 151

Gas leakage detector

If a gas leakage happens around the gas tank, a gas sensor will detect the leakage. As soon as the leakage is detected, a buzzer will alarm the user about it.

- If gas leakage happens around gas cylinder, gas sensor output voltage is Low, digital reading value is 0. Buzzer will alarm.

- If gas leakage does not happen around gas cylinder, gas sensor output voltage is High, digital reading value is 1. Buzzer does not alarm.

The view of setup

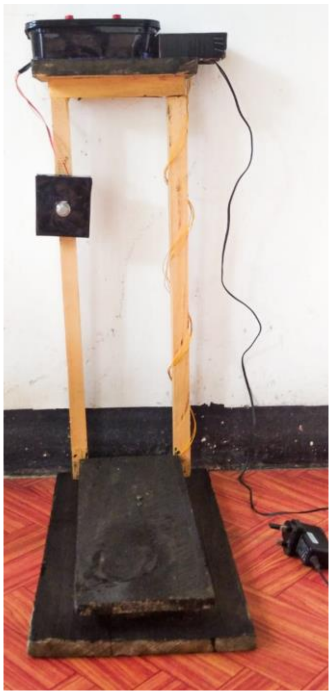


Figure 6. 3- Front view of the setup

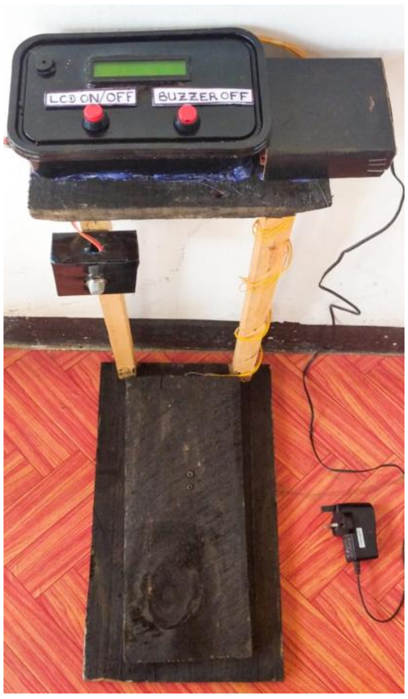


Figure 6. 4- above view of the setup

7 DISCUSSION

This mini project is performed by constructing a system of gas level indicator and leakage detector using an ATmega32 microcontroller. This project have two objectives. They are gas level indicator and leakage detector. This gas level indicator displays the gas level as a percentage on the LCD display. The gas level is calculated every 30 minutes. The gas leakage detector is detected the gas leakage around the gas cylinder. If a gas leakage happens around the gas tank, a gas sensor will detect the leakage. As soon as the leakage is detected, a buzzer will alarm the user about it.

The level of LPG gas is measured by loadcell sensor. That means using the weight, the gas vevel was measured. Also gas leakage is detected by MQ-5 gas sensor. Firstly designed the circuit by using Protues software. It was useful to plan the circuit diagram. Then, the circuit was implemented by using bread board and checked the system was run. After that all the component were soldered in the Vero board. The Atmega32 microcontroller was work only 5 V voltage as the power supply. Our domestic voltage is 230 V. So, it should convert to 5 V. The 230 V - 12V adapter was used to convert 230 V to 12 V. Then, 12 V to 5 V voltage converter was created using LM7805 regulator. Connecting both voltage converter system, 230 V – 5 V convertor was created. Also setup was constructed as figure 5.5 shown. Vero board and power supply was connected and the system was checked. The code was written by using Programmer’s notepad software. Code was uploaded and calibrate the loadcell.

The loadcell is usually used with the module hx711. Hx711 is an electronic component which is used to analog signal convert to digital signal. So in this project we used atmega32 microcontroller. It have in build registers to analog to digital convertor. Then loadcell can be used without the hx711 module. When a load act on the loadcell, there is tension at one end and compression at the other end. This causes a strain on the load on the load cell against the elastic. The strain gauge converts this strain into an electrical signal. The Weinstein Bridge is used to make the difference in resistance proportional to the weight. The output voltage varies by mV, due to the change in resistance. So, microcontroller can’t read the mV voltage value. Therefore the LM741 Op-Amp Ic was used to amplifier the voltage signal between 0V and 4.8 V. For that, the supply voltage was given as 5 V and used resistors Rf and Rin as 10 kΩ and 22 Ω separately. Then read the ADC value and calibrate it using empty gas cylinder and full gas cylinder. From it, we can find the LPD gas level.

In gas leakage detector system was used to MQ5 gas sensor. When gas leak, the heating element inside MQ5 gets heated up and output voltage varies (voltage is LOW = 0 - digital reading). The other time voltage is HIGH (1- digital reading). In this project, if happen gas leakage buzzer will alarm.

So, this project will be helped to all people who use gas cylinder.

Reference

- Electronicwings.com. 2021. *ADC in AVR ATmega16/ATmega32 | AVR ATmega Controllers*. [online] Available at: < https://www.electronicwings.com/avr-atmega/atmega1632-adc > [Accessed 12 August 2021].

- Electronicwings.com. 2021. *Timer in AVR ATmega16/ATmega32 | AVR ATmega Controllers*. [online] Available at: < https://www.electronicwings.com/avr-atmega/atmega1632-timer > [Accessed 12 August 2021].

- Electronicwings.com. 2021. *Interfacing LCD16x2 with AVR ATmega16/ATmega32 in 4-bit mode | AV...* [online] Available at: < https://www.electronicwings.com/avr-atmega/interfacing-lcd-16x2-in-4-bit-mode-with-atmega-16-32- > [Accessed 12 August 2021].

APPENDIX

The code for the system

```
#ifndef F_CPU
#define F_CPU 8000000UL // 8 MHz clock speed
#endif

#define D4 eS_PORTC4
#define D5 eS_PORTC5
#define D6 eS_PORTC6
#define D7 eS_PORTC7
#define RS eS_PORTB0
#define EN eS_PORTB2

#include <avr/io.h>
#include <util/delay.h>
#include<avr/interrupt.h>
#include "lcd.h"

#include <string.h>
#include <stdlib.h>

void init_adc(void);
unsigned int read_adc(unsigned char channel);
unsigned int Weight(void);

volatile unsigned char ms=0,mini=0,sec=0,hour=0,Val_ms=0;
char buffer[10];
int pecentage;
int Voltage;
float x = 107;            // when empty cylinder
float y = 151 ;          // when full cylinter
int z =0;               // precentage of the weight
float ad_val=0;

int main(void){

   DDRD |=(1<<4);        // DDRD 4th bit set as output
   DDRD &= ~(1<<5);      // DDRD 5th bit set as input

   sei();
   TCCR0 = 1<<WGM01 |1<<CS01 |1<<CS00  ;         //pre scler set 64
   TIMSK = 1<<OCIE0;                             // every 1ms cll the interupt
   OCR0= 124;
   TCNT0=0;

   GICR |= 1<<INT0;                  // button of buzzer off set as external
      inturrupt
   MCUCR |= 1<<ISC01 | 1<<ISC00;    // rising edge

   init_adc();
   pecentage = Weight();

   while(1){
        if ((PIND & (1<<5)) == 0 ){          // gas senseor digital pin in PD5
            PORTD|=(1<<4);                   //turn on the buzzer
        }
        if ((PIND & (1<<0))){                 // LCD ON/OFF buttton presss
      (hold on buuton)
            DDRC = 0xFF;
            DDRB = 0xFF;

            Lcd4_Init();
            Lcd4_Set_Cursor(1,1);
            Lcd4_Write_String("Gas Level Is");

            Lcd4_Set_Cursor(2,4);
            itoa(pecentage, buffer, 10);         // display the gas amount as
      pecentage
            Lcd4_Write_String(buffer);

            Lcd4_Set_Cursor(2,7);
            Lcd4_Write_String("%");
        }else{
            PORTC = 0x00;
            PORTB = 0x00;
        }
    }
    return 0;
}

//...timer.........................................................
ISR(TIMER0_COMP_vect){
    if(++Val_ms==100){
        Val_ms=0;
        if(++ms==10){
            ms=0;
            if(++sec==60){
                sec=0;
                if(++mini==30){
                    mini=0;
                    pecentage = Weight();
                }
            }
        }
    }
}

// when press the button buzzer
      off...........................................................
ISR(INT0_vect){
    PORTD &= ~(1<<4); //buzzerr off
}

// initilize the
      ADC...............................................................
      ............
void init_adc(void){
    DDRA &= ~(1<<2);                            /* Make ADC port pinA2 as input
      */
    ADMUX = 1<< REFS0;
    ADCSRA = 1<<ADEN | 1<<ADPS2 | 1<<ADPS1; /* make pre scaler in 64...*/
}

// read value from chanel
      2(PA2)............................................................
unsigned int read_adc(unsigned char channel){
    ADMUX = ADMUX | (channel & 0x0f);               // set input chanel to read
    ADCSRA = ADCSRA | 1<<ADSC;                    // start convercation
    while(ADCSRA & (1<<ADSC)){                     // monitor end of convercasion
      intereupt
    }
    return ADC;
}

// calculat the weight.................................
unsigned int Weight(void){
    ad_val= read_adc(2);
    Voltage=((ad_val*5)/1024)*10000;
    z = ((Voltage- x)*100)/(y-x); // the function of calculate the weight as
      precentage
    return z;
}
```

The code for the demonstrate (without internal interrupt)

```
#ifndef F_CPU
#define F_CPU 8000000UL // 8 MHz clock speed
#endif

#define D4 eS_PORTC4
#define D5 eS_PORTC5
#define D6 eS_PORTC6
#define D7 eS_PORTC7
#define RS eS_PORTB0
#define EN eS_PORTB2

#include <avr/io.h>
#include <util/delay.h>
#include<avr/interrupt.h>
#include "lcd.h"

#include <string.h>
#include <stdlib.h>

void init_adc(void);
unsigned int read_adc(unsigned char channel);
unsigned int Weight(void);

char buffer[10];
int pecentage;
int Voltage;
float x = 107;           // when empty cylinder
float y = 151 ;         // when full cylinter
int z =0;              // precentage of the weight
float ad_val=0;

int main(void){

   DDRD |=(1<<4);        // DDRD 4th bit set as output
   DDRD &= ~(1<<5);      // DDRD 5th bit set as input

   sei();
   GICR |= 1<<INT0;                   // button of buzzer off set as external
       inturrupt
   MCUCR |= 1<<ISC01 | 1<<ISC00;     // rising edge

   init_adc();

   while(1){
        pecentage = Weight();
        _delay_ms(1000);
        if ((PIND & (1<<5)) == 0 ){          // gas senseor digital pin in PD5
            PORTD|=(1<<4);                   //turn on the buzzer
        }
        if ((PIND & (1<<0))){                 // LCD ON/OFF buttton presss
       (hold on buuton)
            DDRC = 0xFF;
            DDRB = 0xFF;

            Lcd4_Init();
            Lcd4_Set_Cursor(1,1);
            Lcd4_Write_String("Gas Level Is");

            Lcd4_Set_Cursor(2,4);
            itoa(pecentage, buffer, 10);         // display the gas amount as
       pecentage
            Lcd4_Write_String(buffer);

            Lcd4_Set_Cursor(2,7);
            Lcd4_Write_String("%");
        }else{
            PORTC = 0x00;
            PORTB = 0x00;
        }
    }
   return 0;
}

// when press the button buzzer
       off.............................................................
ISR(INT0_vect){
    PORTD &= ~(1<<4); //buzzerr off
}

// initilize the
       ADC.......................................................................
       .............
void init_adc(void){
    DDRA &= ~(1<<2);                          /* Make ADC port pinA2 as input
       */
    ADMUX = 1<< REFS0;
    ADCSRA = 1<<ADEN | 1<<ADPS2 | 1<<ADPS1; /* make pre scaler in 64...*/
}

// read value from chanel
       2(PA2)................................................................
unsigned int read_adc(unsigned char channel){
    ADMUX = ADMUX | (channel & 0x0f);            // set input chanel to read
    ADCSRA = ADCSRA | 1<<ADSC;                   // start convercation
    while(ADCSRA & (1<<ADSC)){                  // monitor end of convercasion
       intereupt
    }
    return ADC;
}

// calculat the weight.................................
unsigned int Weight(void){
    ad_val= read_adc(2);
    Voltage=((ad_val*5)/1024)*10000;
    z = ((Voltage- x)*100)/(y-x); // the function of calculate the weight as
       precentage
    return z;
}
```