GROUP 08

SECTION 10

WATER LEVEL MONITORING

Using 8051/PIC Microcontroller

Shakib Khan – 1711661642

Faria Rahman Khanam – 1711171042

Pervej Ahamed Joy – 1713076642

CSE 331 – Project Report

Course Instructor: Mohammad Sakib Mahmud (MoM).

Lab Instructor: Asif Ahmed Neloy.

Introduction

Our project's primary purpose is to save water and, Additionally, to monitor the water level in the tank. First of all, we will open the cap if it matches the input configuration, we set then it will display the percentage level in the LCD. If the input does not correspond with the design, then it displays nothing.

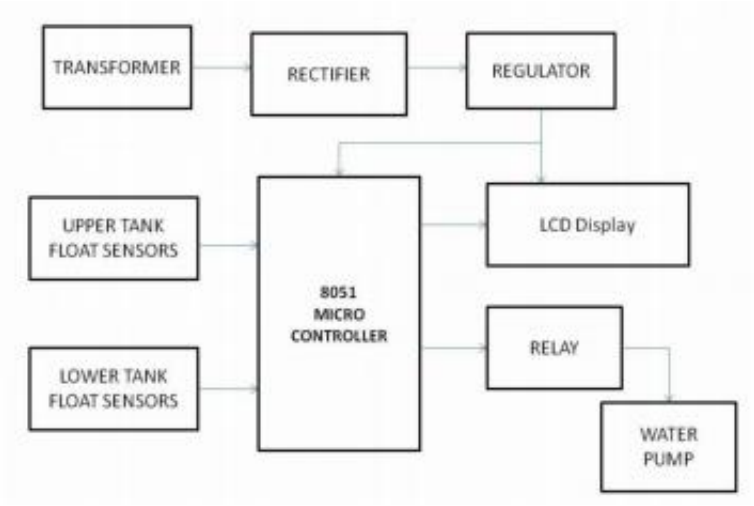
Project Application

1. It can be used in many fields like Agriculture, Plantation, etc.
2. It can be used in rooftop farming.
3. The most crucial role is It can help in reducing the waste of water.

Components

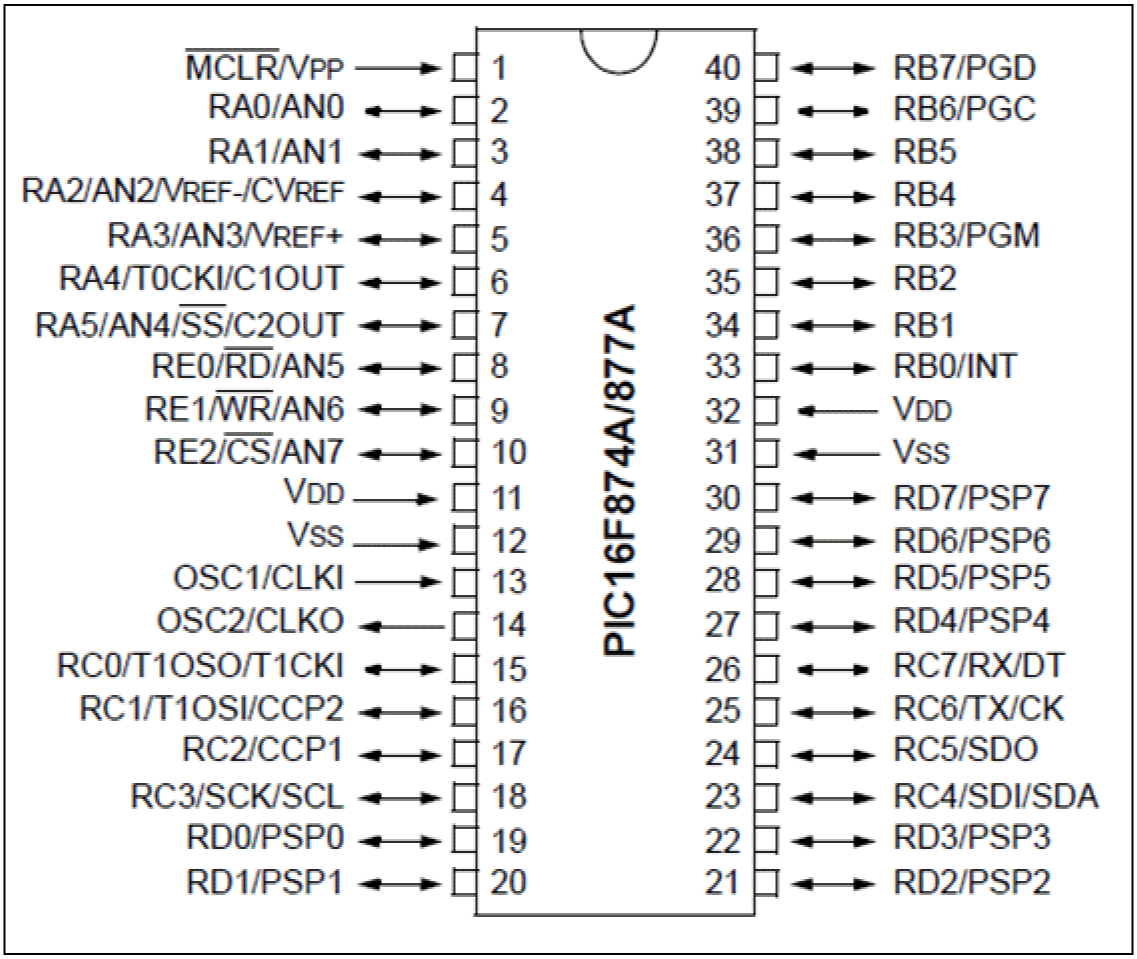
4. PIC Microcontroller (PIC16F877A),
5. Capacitors,
6. Resistors,
7. Battery,
8. Wires,
9. LED.

Block Diagram



PIC16F877A Microcontroller

This powerful (200 nanosecond instruction execution) yet easy-to-program (only 35 single word instructions) CMOS FLASH-based 8-bit microcontroller packs Microchip's powerful PIC® architecture into a 40 package. It is upwards compatible with the PIC16C5X, PIC12CXXX, and PIC16C7X devices. The PIC16F877A features 256 bytes of EEPROM data memory, self-programming, an ICD, 2 Comparators, 8 channels of 10-bit Analog-to-Digital (A/D) converter, 2 capture/compare/PWM functions, the synchronous serial port can be configured as either 3-wire Serial Peripheral Interface (SPI™) or the 2-wire Inter-Integrated Circuit (I²C™) bus and a Universal Asynchronous Receiver Transmitter (USART).



PIC16F877A Pin Configuration

Pin Number	Pin Name	Description
1	MCLR/Vpp	MCLR is used during programming, mostly connected to a programmer like PicKit.
2	RA0/AN0	Analog pin 0 or 0th pin of PORTA
3	RA1/AN1	Analog pin 1 or 1st pin of PORTA
4	RA2/AN2/Vref-	Analog pin 2 or 2nd pin of PORTA
5	RA3/AN3/Vref+	Analog pin 3 or 3rd pin of PORTA
6	RA4/T0CKI/C1out	4th pin of PORTA
7	RA5/AN4/SS/C2out	Analog pin 4 or 5th pin of PORTA
8	RE0/RD/AN5	Analog pin 5 or 0th pin of PORTE
9	RE1/WR/AN6	Analog pin 6 or 1st pin of PORTE
10	RE2/CS/AN7	7th pin of PORTE

Pin Number	Pin Name	Description
11	Vdd	Ground pin of MCU
12	Vss	Positive pin of MCU (+5V)
13	OSC1/CLKI	External Oscillator/clock input pin
14	OSC2/CLKO	External Oscillator/clock output pin
15	RC0/T1OSO/T1CKI	0th pin of PORT C
16	RC1/T1OSI/CCP2	1st pin of POCTC or Timer/PWM pin
17	RC2/CCP1	2nd pin of POCTC or Timer/PWM pin
18	RC3/SCK/SCL	3rd pin of POCTC
19	RD0/PSP0	0th pin of POCTD
20	RD1/PSPI	1st pin of POCTD

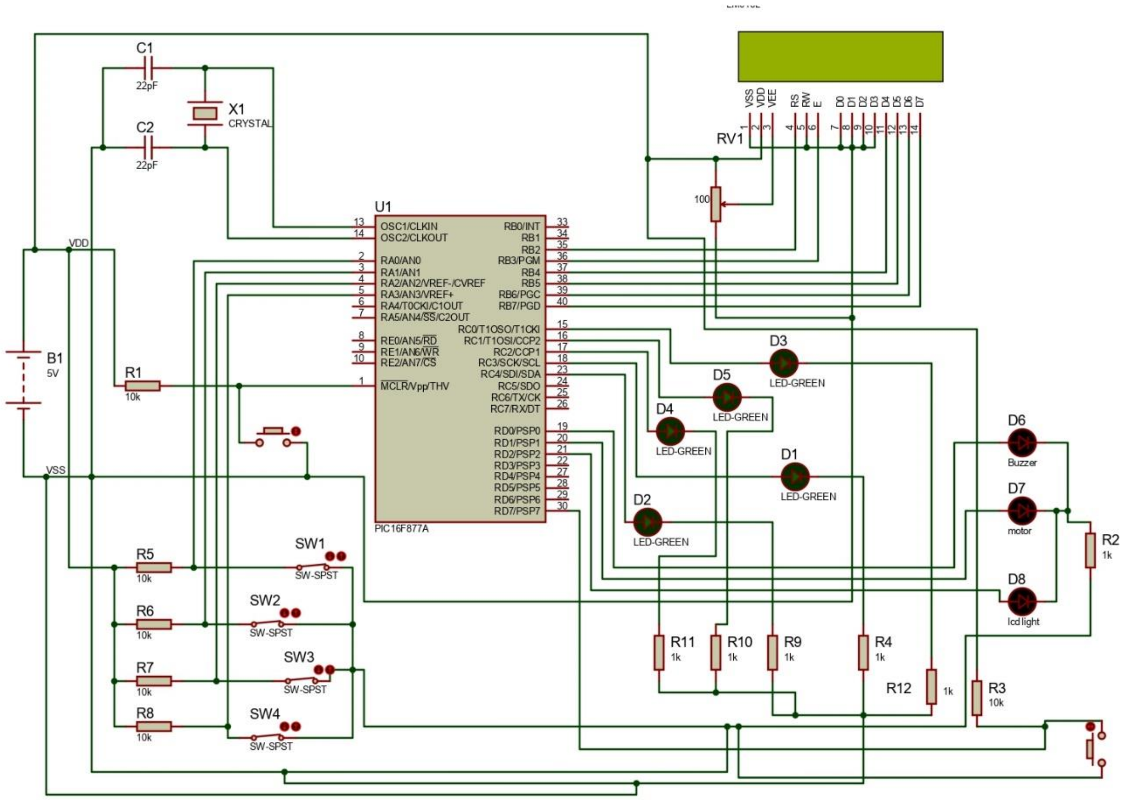
Pin Number	Pin Name	Description
21	RD2/PSP2	2nd pin of POCTD
22	RD3/PSP3	3rd pin of POCTD
23	RC4/SDI/SDA	4th pin of POCTC or Serial Data in pin
24	RC5/SDO	5th pin of POCTC or Serial Data Out pin
25	RC6/Tx/CK	6th pin of POCTC or Transmitter pin of Microcontroller
26	RC7/Rx/DT	7th pin of POCTC or Receiver pin of Microcontroller
27	RD4/PSP4	4th pin of POCTD
28	RD5/PSP5	5th pin of POCTD
29	RD6/PSP6	6th pin of POCTD
30	RD7/PSP7	7th pin of POCTD

Data Table

Here is the data table we create for monitoring water. When the input matches with table data, then it displays which states it currently is Very Low, Low, Medium, and Full

SW1	SW2	SW3	SW4	STATUS
0	0	0	0	All data pins are grounded, which indicates the tank is Full.
0	0	0	1	The water level is below D3 and above D2, which indicates High level.
0	0	1	1	The water level is below D2 and above D1, indicates Medium level.
0	1	1	1	The water level is below D1 and above D0, which indicates Low level.
1	1	1	1	The water level is below D0, which indicates a Very Low level.

Schematic Diagram



Basic Function

```
char txt1[] = "Project"
char txt2[] = "Developed By...."
char txt3[] = "GROUP 08";
char txt4[] = "----------------"

char mtr1[] = "Motor "
char mtr2[] = "OFF"
char mtr3[] = "ON"

char wtr1[] = "Level: "
char wtr2[] = "Very Low"
char wtr3[] = "Low"
char wtr4[] = "Medium"
char wtr5[] = "High"
char wtr6[] = "Full"
```

Code for this application:

We write the code C programming language using the mikroC IDE. We import necessary libraries. After writing the code, we compile the code and generate the.HEX file. Then upload the .hex file in Proteus simulation. Upload the hex file in PIC16F877A, then run the proteus file.

```
// LCD module connections
sbit LCD_RS at RB2_bit;
sbit LCD_EN at RB3_bit;
sbit LCD_D4 at RB4_bit;
sbit LCD_D5 at RB5_bit;
sbit LCD_D6 at RB6_bit;
sbit LCD_D7 at RB7_bit;

sbit LCD_RS_Direction at TRISB2_bit;
sbit LCD_EN_Direction at TRISB3_bit;
sbit LCD_D4_Direction at TRISB4_bit;
sbit LCD_D5_Direction at TRISB5_bit;
sbit LCD_D6_Direction at TRISB6_bit;
sbit LCD_D7_Direction at TRISB7_bit;
// End LCD module connections

char txt1[] = "Project";
char txt2[] = "Developed By....";
char txt3[] = "GROUP 08";
char txt4[] = "----------------";

char mtr1[] = "Motor ";
char mtr2[] = "OFF";
char mtr3[] = "ON";

char wtr1[] = "Level: ";
char wtr2[] = "Very Low";
char wtr3[] = "Low";
char wtr4[] = "Medium";
char wtr5[] = "High";
char wtr6[] = "Full";



void main()
{

  int i = 0;
  int c = 16;
  int b = 0;
  CMCON = 0x07;
  ADCON1 = 0x06;
  TRISA = 0x0F;           // set direction to be input
  PORTA = 0x00;
  PORTD = 0x00;
  PORTC = 0x00;
  TRISB = 0x00;           //  set direction to be output
  TRISC = 0x00;          // set direction to be output
  TRISD = 0x80;          // set direction to be output

  PORTD.F2 = 1;
  PORTD.F7 = 1;

  Lcd_Init();                        // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);               // Clear display
  Lcd_Cmd(_LCD_CURSOR_OFF);          // Cursor off
  Lcd_Out(1,1,txt1);                 // Write text in first row
  Lcd_Out(2,1,txt2);                 // Write text in second row
  Delay_ms(500);
  Lcd_Cmd(_LCD_CLEAR);               // Clear display
  Lcd_Out(1,1,txt3);                 // Write text in first row
  Lcd_Out(2,1,txt4);                 // Write text in second row
  Delay_ms(500);

// Moving text
for(i=0; i<15; i++)
{ // Move text to the right 16 times
  Lcd_Cmd(_LCD_SHIFT_RIGHT);
  Delay_ms(125);
}
i=0;

do
{
   Lcd_Cmd(_LCD_CLEAR);
   Lcd_Out(1,1,wtr1);
   Lcd_Out(2,1,mtr1);
   if(c>0)
   {
      PORTD.F2 = 1;
      c--;
   }
   else
      PORTD.F2 = 0;

   if(b>0)
   {
       PORTD.F0 = 1;
       Delay_ms(125);
       PORTD.F0 = 0;
       b--;
   }

   if(PORTD.F7 == 0)
     c = 16;

   if(PORTA == 0x0F)
   {
       PORTD.F1 = 1;
       Lcd_Out(1,8,wtr2);
       Lcd_Out(2,7,mtr3);
       PORTC = 1;
       if(i == 0)
       {
          c = 16;
          b = 3;
       }
       i=1;
   }
   else if(PORTA == 0x0E)
   {
       Lcd_Out(1,8,wtr3);
       if(i == 1)
          Lcd_Out(2,7,mtr3);
       else
          Lcd_Out(2,7,mtr2);
       PORTC = 3;
   }
   else if(PORTA == 0x0C)
   {
       Lcd_Out(1,8,wtr4);
       if(i == 1)
          Lcd_Out(2,7,mtr3);
       else
          Lcd_Out(2,7,mtr2);
       PORTC = 7;
   }
   else if(PORTA == 0x08)
   {
       Lcd_Out(1,8,wtr5);
       if(i == 1)
          Lcd_Out(2,7,mtr3);
       else
          Lcd_Out(2,7,mtr2);
          PORTC = 15;
   }

   else if(PORTA == 0x00)
   {
       Lcd_Out(1,8,wtr6);
       Lcd_Out(2,7,mtr2);
       PORTD.F1 = 0;
       if(i == 1)
       {
         c = 16;
         b = 3;
       }
       i=0;
       PORTC = 31;
   }
   else
       PORTA = 0x0F;
   Delay_ms(125);


}while(1);            // Endless loop
}
```

Overview

So, in Water Level Monitoring, we first need power. We used a 5v battery then used capacitor and quartz crystal, which controls the electric signals. The central brain of the monitoring system is the PIC16F877A IC, which has 40 pins. The diagram and pins configuration are given below.

For input, we use SW1, SW2, SW3, SW4 to select which level the water is. So when the water level full, the Buzzer light will be light up, and when the circuit first turns on, the LED light will light up for one time.

We connect the pins with the PIC16F877A and connect the output wires with the LCD board. And run the circuit.

From the table, we set the input, and the output will show in the LCD. If the input doesn't match the table's value, it shows nothing, which means it indicates the input is wrong.

So that's how the circuit is working, and we can monitor the water level.

THE END