COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY

CHENNAI 600025

EC5561 MICROPROCESSOR AND MICROCONTROLLER INTERFACING LABORATORY

SEMESTER V

(R-2019)

MINI PROJECT REPORT ON CAN SIMULATION USING 8051 MICROCONTROLLER

SUBMITTED BY:

POORNISHWAR M (2020105561)

SANTOSH M (2020105572)

SIVAPRAKASM D (2020105578)

ABSTRACT :

Microcontrollers communicate with each other using signals given from each other's ports for transmitting and receiving signals. Therefore, on the fulfillment of certain conditions or by following a certain protocol, the microprocessors can achieve various applications that require sophisticated networking. The water level of the tank and sump is measured using an ADC and given as input to the microcontrollers. The microcontrollers, depending upon the output control the working of the sump motor and the tank motor. The sump motor is turned on initially, when the level of water in the sump is zero. When it reaches it's 50% capacity, the tank motor is turned on and starts drawing water from the pump.

Once the tank reaches full capacity, both the motors are then turned off. For serial communication, the 8051 can use either asynchronous or synchronous types. To select which type will be used, we use a combination of registers. UART (Universal Asynchronous Reciever/Transmitter) as the name suggests is an asynchronous mode of data transmission. In the case of asynchronous data transmission, individual bits of data are transferred in the form of frames. These frames have gaps in between them so that the device can synchronize itself to receive data as it does not use an external clock. A driver circuit handles electric signaling levels between two circuits. A Universal asynchronous receiver-transmitter (UART) Communication is usually an individual component or part of an integrated circuit. We can use it for communications over a computer or its peripheral devices such as a mouse, monitor or printer.

The data coming in at the receiving end of the data line in a serial data transfer is all 0s and 1s, it is difficult to make sense of the data unless the sender and receiver agree on a set of rules, a protocol, on how the data is packed, how many bits constitute a character, and when the data begins and ends. In microcontroller chips, there are usually a number of dedicated UART hardware peripherals available. Asynchronous serial data communication is widely used for character oriented transmissions. In this method, each character is placed between start and stop bits. This is called framing.

In data framing for asynchronous communications, the data such as ASCII characters, are packed between a start bit and stop bit. The start bit is always one bit, but the stop bit can be one or two bits. The start bit is always a 0 (low) and the stop bit is 1(high). Look at below figure in which the ASCII character "A" (8-bit binary 0100 0001) is framed between the start bit and a stop bit. Notice that the LSB is sent out first.

8051 Microcontrollers:-

The AT89C51 is a low-power, high-performance CMOS 8-bitmicrocomputer with 4 Kbytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel's high density non-volatile memory technology and is compatible with the industry standard MCS-51Ô instruction set and pin out. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional non-volatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89C51 provides the following standard features: 4 Kbytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, five vector two-level interrupt architecture, a full duplex serial port, and on-chip oscillator and clock circuitry. In addition, the AT89C51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power down Mode saves the RAM contents but freezes the oscillator.

The following list gives the features of the 8051 architecture:-

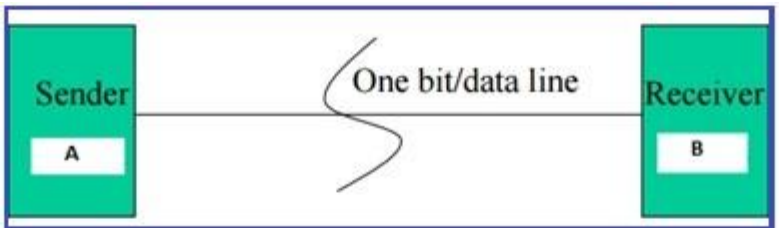
- Optimized 8-bit CPU for control applications.
- 64K Program Memory address space.
- 4K byte of ROM.
- 128 bytes of on chip Data Memory.
- 32 Bi-directional and individually addressable I/O lines.
- Two 16 bit timer/counters.
- Full Duplex UART.
- 5-vector interrupts structure with priority levels.
- On chip clock oscillator.

THEORY

When electronic devices communicate with each other, they can transfer data in two different ways. One is serial and the other one is parallel, When digital data is transferred serially, it is transmitted bit by bit, whereas in parallel transfer, many bits are transmitted at the same time. Though the parallel transfer of data is much faster but requires many wires. while a serial transfer is slower as compared to parallel transfer but requires few wires. Serial communication may be synchronous or asynchronous. In synchronous communication, the transmitter also transmits a clock along with data. This clock is used for synchronization between transmitter and receiver devices In an asynchronous transfer of data, there is no clock.

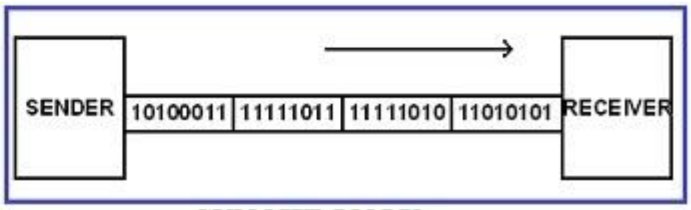
INTRODUCTION TO SERIAL COMMUNICATION WITH 8051 MICRO CONTROLLER

Serial communication may be simplex, half-duplex or full-duplex. simplex communication means that data will be transmitted only in one direction while half-duplex means data will be transmitted in both directions but at one time, only one device can transmit, whereas full-duplex means data may be transmitted in both directions at one time, while one device is transmitting, it can also receive data transmitted from other devices at same time. As I have mentioned before, the transmitter and receiver are configured to communicate at some data transfer rate before communication starts. This data transfer rate or a number of bits transmitted per second is called the baud rate for handling serial communication.
Parallel communication is fast but it is not applicable for long distances (for printers). Moreover, it is also expensive. Serial is not much fast as parallel communication but it can deal with transmission of data over longer distances (for telephone line, ADC, DAC). It is also cheaper and requires less physical wires, that's why we use serial communication. This article also deals with how to serially communicate in 8051 microcontrollers.

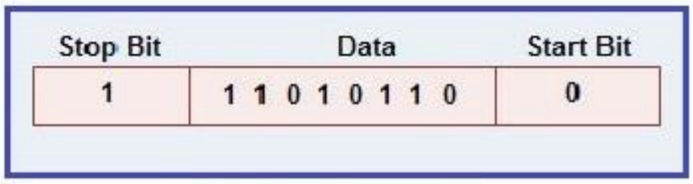


METHODS OF SERIAL COMMUNICATION

There are two methods of Serial Communication:



SYNCHRONOU



ASYNCHRONOUS

SYNCHRONOUS: Transfer the block of data (characters) between sender and receiver spaced by fixed time interval. This transmission is synchronized by an external clock.

ASYNCHRONOUS: There is no clock involved here and transmission is synchronized by special signals along the transmission medium. It transfers a single byte at a time between sender and receiver along with inserting a start bit before each data character and a stop bit at its termination so that to inform the receiver where the data begins and ends. An example is the interface between a keyboard and a computer. Keyboard is the transmitter and the computer is the receiver. We use USART and UART for serial communications. USART or UART is a microcontroller peripheral which converts incoming and outgoing bytes of data into a serial bit stream. Both have same work but with different methods which is explained below.

USART stands for Universal Synchronous/Asynchronous Receiver-Transmitter. USART uses external clock so it needs separate line to carry the clock signal. Sending peripheral generates a clock and the receiving peripheral recover from the data stream without knowing the baud rate ahead of time. By use of external clock, USART's data rate can be much higher (up to rates of 4 Mbps) than that of a standard UART.

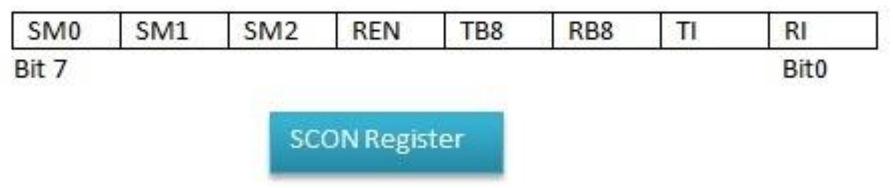
8051 SERIAL COMMUNICATION PROGRAMMABLE REGISTERS :

8051 microcontroller has a built-in serial port called UART. We can easily read and write values to the serial port. For using serial port we simply need to configure the serial port:

- Operation mode (how many data bits we want)
- Baud rate

There are 21 Special function registers (SFR) in 8051 microcontroller and 21 unique locations are for these 21 SFR. Each of these register is of 1 byte size. The “Serial Control” (SCON) is the SFR which is used to configure serial port.

SCON - SPECIAL FUNCTION REGISTER :



SM0 : ***Serial port mode***
Bit 0 of serial port mode.
SM1 ***: Serial port mode***
Bit 1 of serial port mode.
SM2 : ***Enable multiprocessor***
Enable multiprocessor communication in modes 2 and 3 (for 9 bit UART).
REN : ***Receiver Enable***
Set/clear by software to enable/disable receive operation
TB8 : ***Transmit bit 8***
Set or clear by software. The 9 bits will be transmitted in mode 2 and 3.
TB8 = 1, a value is written to the serial port, 9th bit of data = 1.
TB8 = 0, 9th bit of data = 0, RI will not set.
RB8 :***Receive bit 8***
Set or clear by software. The 9 bits will be received in mode 2 and 3. First eight bits are the data received and 9th bit received will be placed in RB8.

TI :*Transmit Interrupt flag*
Set by hardware when a byte is transmitted completely. Now the port is free and ready to send the next byte. This bit must be cleared by software.

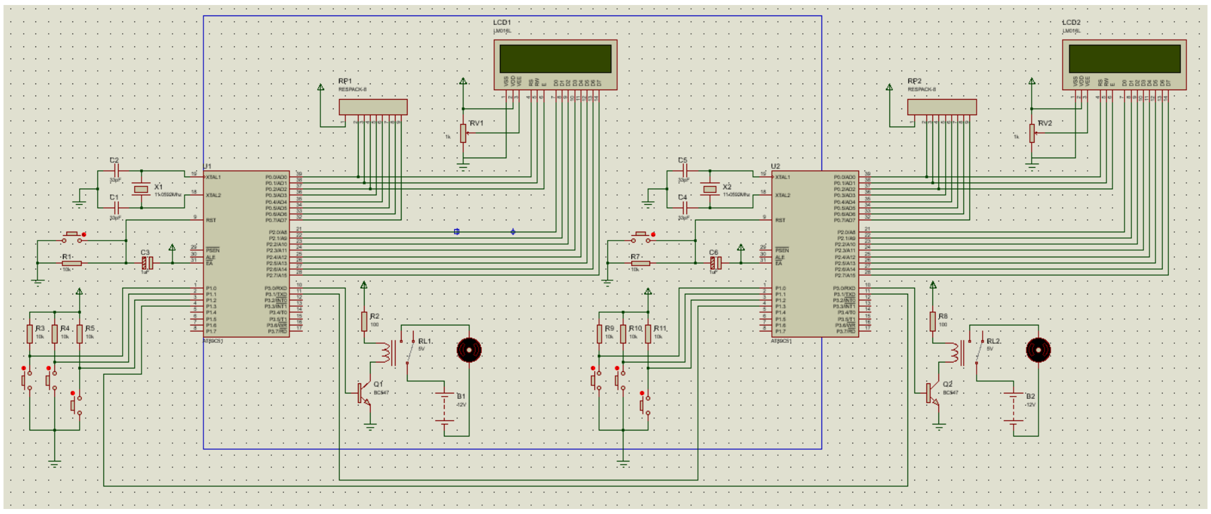
RI : *Receive Interrupt flag*
Set by hardware when a byte has been completely received. This lets the program to know that it needs to read the value quickly before another byte is read. This bit must be cleared by software

INTERFACING ADC WITH 8051 MICROCONTROLLER

ADC0804 is an 8 bit successive approximation analogue to digital converter from National semiconductors. The features of ADC0804 are differential analogue voltage inputs, 0-5V input voltage range, no zero adjustment, built in clock generator, reference voltage can be externally adjusted to convert smaller analogue voltage span to 8 bit resolution etc

WORKING PROCEDURE WITH SIMULATION RESULTS

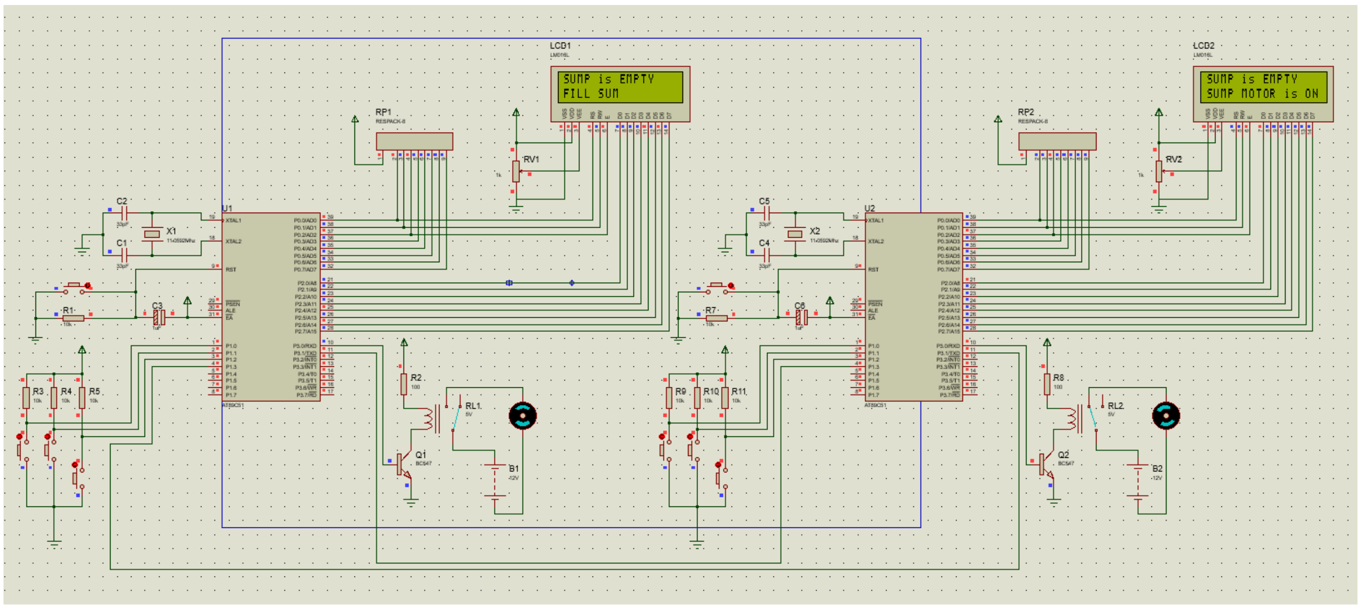
CIRCUIT SCHEMATIC :



1. Assuming both sump and tank is empty initially :

   Then sump motor turns ON

   The tank motor remains OFF

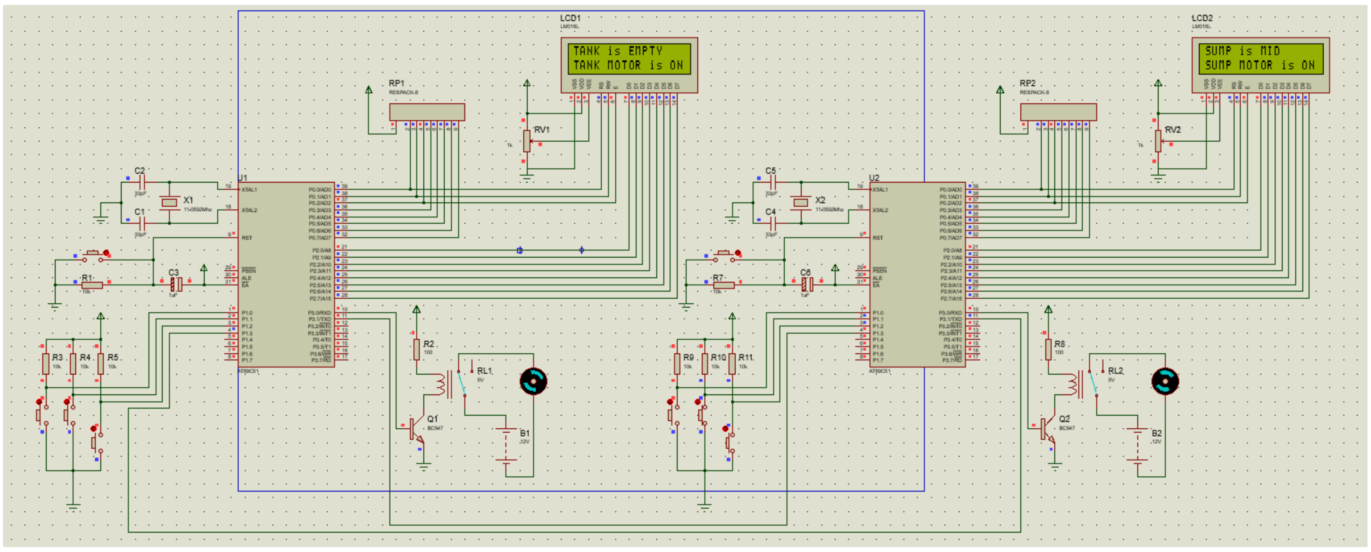


```
SUMP is EMPTY
FILL SUM
```

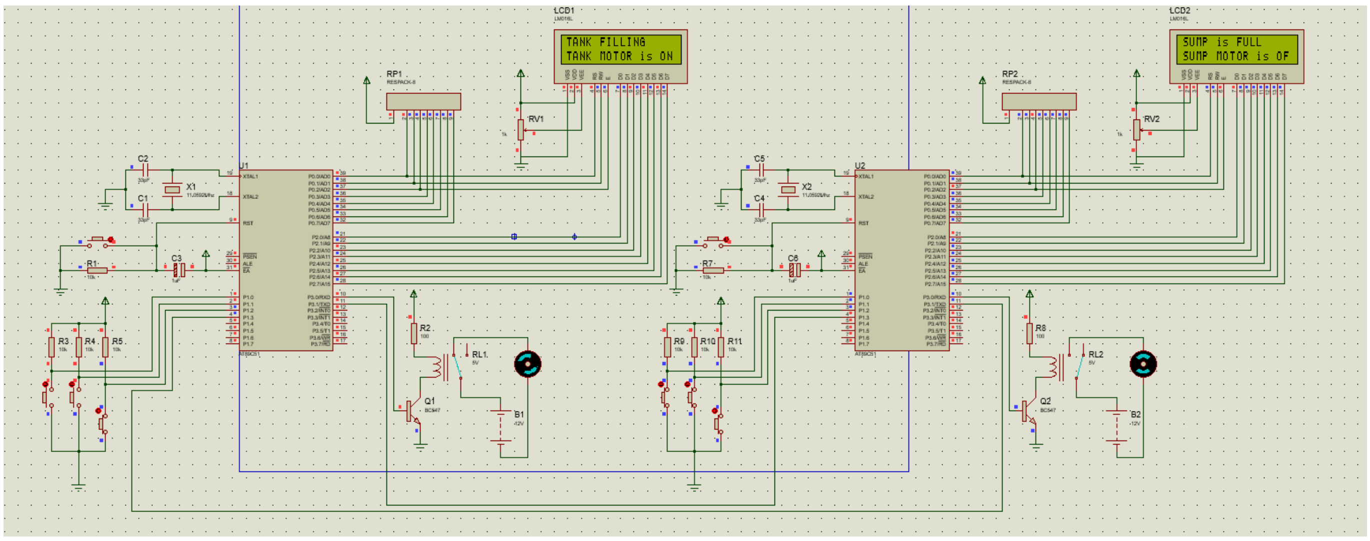
```
SUMP is EMPTY
SUMP MOTOR is ON
```

2. As soon as the sump reaches HALF , The tank motor turns ON and tank starts filling.

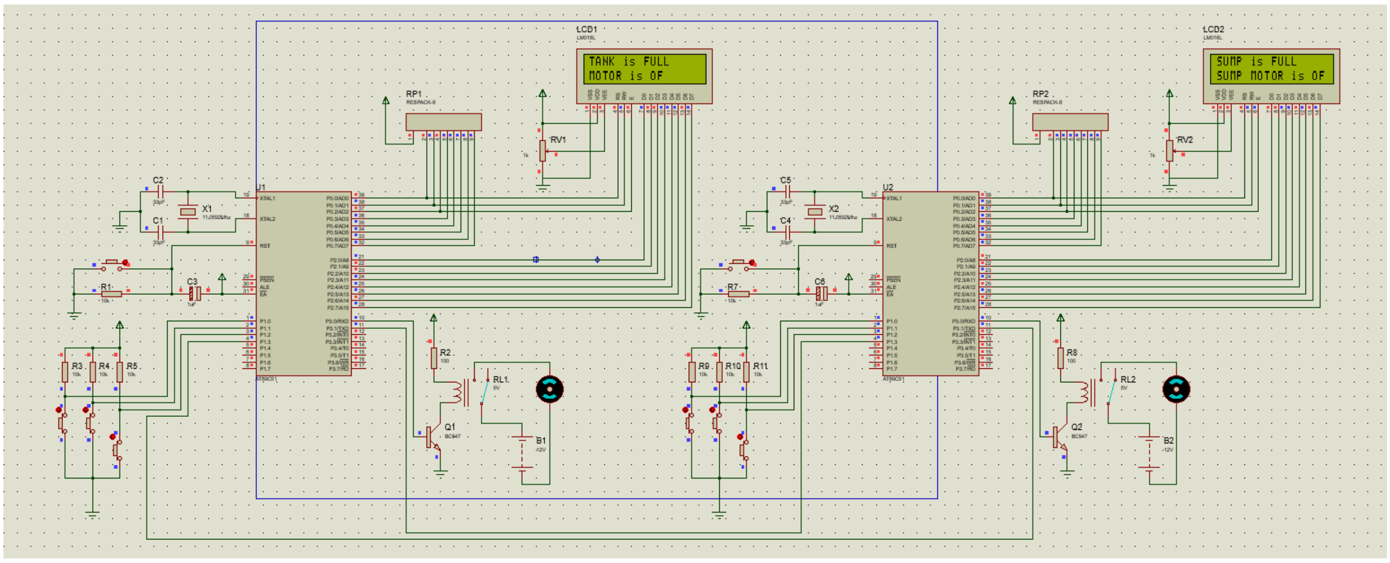
   50 Sump is fixed as threshold to turn ON the Tank motor.



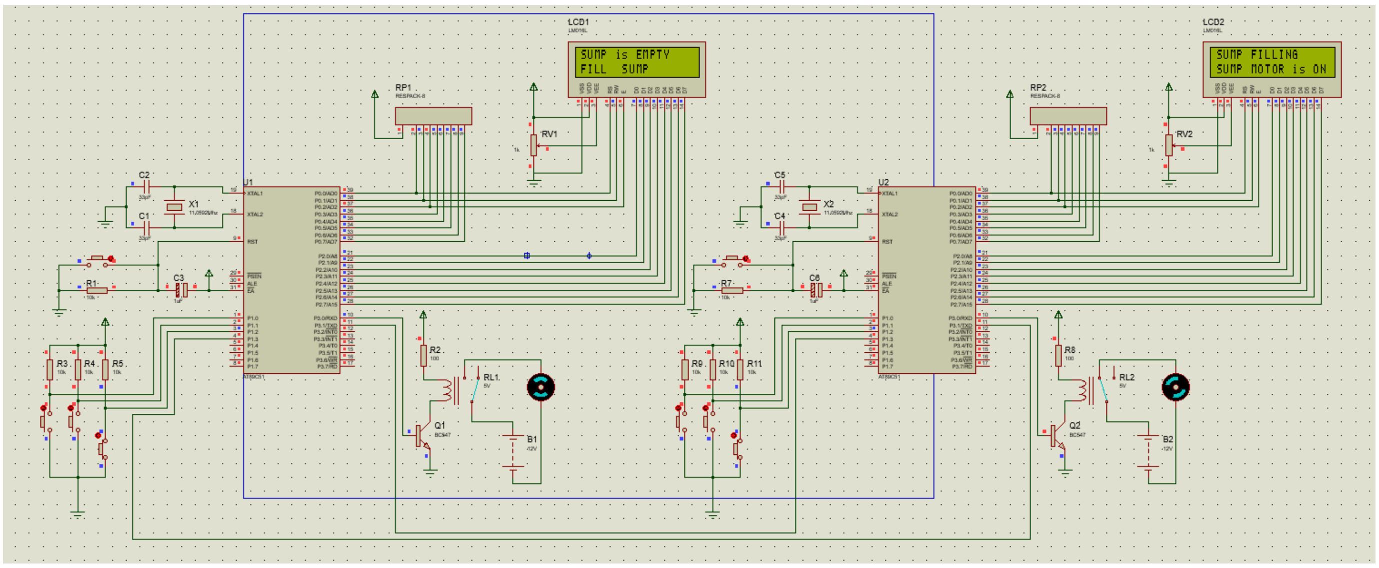
3. Now that the Sump is Full the Sump motor turns OFF and the Tank motor is still ON until it’s FULL.



4. Both Motors turns OFF when both Sump and Tank are FULL.



5. Irrespective of the state of the TANK’s water level , the Sump is ensured to have 50% for backup purposes.



SOURCE CODE :

FOR SUMP MOTOR :

```
#include <reg51.h>
sbit full=P1^0;
sbit mid=P1^1;
sbit emp=P1^2;
sbit t2=P1^3;
sbit rs=P0^0;
sbit rw=P0^1;
sbit en=P0^2;
sbit rly=P3^0;
sbit tank_wfull=P3^1;

void lcddta(unsigned char[],unsigned char);
void lcdcmd(unsigned char);
void msdelay(unsigned int);
void main(void){
    rly=0;
    P0=00;
    P2=00;
    full=1;
    mid=1;
    emp=1;
    t2=1;
          tank_wfull=1;

    lcdcmd(0x38);
    lcdcmd(0x0c);
    lcdcmd(0x06);
lcdcmd(0x01);
    while(1){
        if(t2==0){
             //lower motor condition

            if(emp==1 && mid==1 && full==1){
                rly=1;
                tank_wfull=1;
                lcdcmd(0x01);
                lcdcmd(0x80);
                lcddta("TANK is EMPTY",13);
                lcdcmd(0xc0);
                lcddta("TANK MOTOR is ON",17);
            }else if(emp==0 && mid==1 && full==1){
               tank_wfull=1;
                rly=1;
                lcdcmd(0x01);
                lcdcmd(0x80);
                lcddta("TANK FILLING",12);
                lcdcmd(0xc0);
                lcddta("TANK MOTOR is ONN",19);
            }else if(emp==0 && mid==0 && full==1){
                rly=1;
                 tank_wfull=1;
                lcdcmd(0x01);
                lcdcmd(0x80);
                lcddta("TANK is MID",12);
                lcdcmd(0xc0);
                lcddta("MOTOR is ON",12);
            }else if(emp==0 && mid==0 && full==0){
               tank_wfull=0;
                rly=0;
                lcdcmd(0x01);
                lcdcmd(0x80);
                lcddta("TANK is FULL",12);
                lcdcmd(0xc0);
                lcddta("MOTOR is OFF",12);
            }
            else{
                rly=0;
                lcdcmd(0x01);
                lcdcmd(0x80);
                lcddta("INVALID STATUS",15);
                lcdcmd(0xc0);
                lcddta("MOTOR is OFF",12);
               continue;
                            }

    }
      else{
        rly=0;
        lcdcmd(0x01);
        lcdcmd(0x80);
        lcddta("SUMP is EMPTY",13);
        lcdcmd(0xc0);
        lcddta("FILL  SUMP",11);
    }
```

```
    } //end of while
}//end of main

void lcdcmd(unsigned char cmd){
    P2=cmd;
    rs=0;//cmd
    rw=0;//write
    en=1;//latch
    msdelay(5);
    en=0;
}//end of lcdcmd

void lcddta(unsigned char a[],unsigned char len){
    unsigned char x;

    for(x=0;x<len;x++){
        P2=a[x];
        rs=1; //data
        rw=0;
        en=1;
        msdelay(5);
        en=0;
    }//end of for
}//end of lcddta

void msdelay(unsigned int a)
{
    unsigned int x,y;
    for(x=0;x<a;x++)
        for(y=0;y<1275;y++);
}//end of msdelay
```

FOR TANK MOTOR :

```
#include <reg51.h>
sbit full=P1^0;
sbit mid=P1^1;
sbit emp=P1^2;
sbit t2=P1^3;
sbit rs=P0^0;
sbit rw=P0^1;
sbit en=P0^2;
sbit rly=P3^0;
sbit sump_en=P3^1;
sbit tank_wfull=P1^3;

void lcddta(unsigned char[],unsigned char);
void lcdcmd(unsigned char);
void msdelay(unsigned int);
void main(void){
   rly=0;
   P0=00;
   P2=00;
   full=1;
   mid=1;
   emp=1;
   t2=1;
   lcdcmd(0x38);
   lcdcmd(0x0c);
   lcdcmd(0x06);
   lcdcmd(0x01);
      while(1){
                  if(tank_wfull==0 && emp==0 && mid==0 && full==1)
               {   rly=0;
               lcdcmd(0x01);
               lcdcmd(0x80);
               lcddta("TANK is FULL",12);
               lcdcmd(0xc0);
               lcddta("SUMP MOTOR is OFF",18);


                                                         }
                  else if((tank_wfull==0 && emp==0 && mid==1 &&
full==1)||(tank_wfull==0 && emp==1 && mid==1 && full==1)) {
                  rly=1;
               lcdcmd(0x01);
               lcdcmd(0x80);
               lcddta("SUMP FILLING",12);
               lcdcmd(0xc0);
               lcddta("SUMP MOTOR is ON",17);
                                                         }
               else{


            if(emp==1 && mid==1 && full==1){
               rly=1;
                  sump_en=1;
               lcdcmd(0x01);
```

```
    lcdcmd(0x80);
    lcddta("SUMP is EMPTY",13);
    lcdcmd(0xc0);
    lcddta("SUMP MOTOR is ON",17);
}
else if(emp==0 && mid==1 && full==1){
    rly=1;
        sump_en=1;
    lcdcmd(0x01);
    lcdcmd(0x80);
    lcddta("SUMP FILLING",12);
    lcdcmd(0xc0);
    lcddta("SUMP MOTOR is ON",17);

}else if(emp==0 && mid==0 && full==1){
    rly=1;
        sump_en=0;
    lcdcmd(0x01);
    lcdcmd(0x80);
    lcddta("SUMP is MID",11);
    lcdcmd(0xc0);
    lcddta("SUMP MOTOR is ON",17);
}else if(emp==0 && mid==0 && full==0){
    rly=0;
        sump_en=0;
    lcdcmd(0x01);
    lcdcmd(0x80);
    lcddta("SUMP is FULL",12);
    lcdcmd(0xc0);
```

```
                    lcddta("SUMP MOTOR is OFF",19);

                }
                    else{
                        rly=0;
                        sump_en=1;
                    lcdcmd(0x01);
                    lcdcmd(0x80);
                    lcddta("INVALID SUMP STATUS",19);
                    lcdcmd(0xc0);
                    lcddta("SUMP MOTOR is OFF",18);
                                        }
                                                        } //end of else


    } //end of while
}//end of main

void lcdcmd(unsigned char cmd){
    P2=cmd;
    rs=0;//cmd
    rw=0;//write
    en=1;//latch
    msdelay(5);
    en=0;
}//end of lcdcmd

void lcddta(unsigned char a[],unsigned char len){
    unsigned char x;
    for(x=0;x<len;x++){
        P2=a[x];
        rs=1; //data
        rw=0;
        en=1;
        msdelay(5);
        en=0;
    }//end of for
}//end of lcddta

void msdelay(unsigned int a)
{
    unsigned int x,y;

    for(x=0;x<a;x++)
        for(y=0;y<1275;y++);
}//end of msdelay
```

CONCLUSION :

Thus communication between two 8051 microcontroller is established using CAN (Communication Area Network) protocol with a common household application of adaptive water level indicator and thus dry run prevention can be achieved.