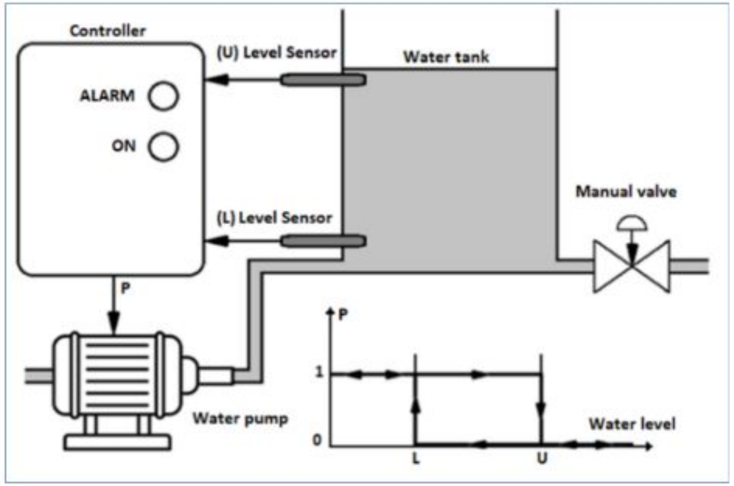
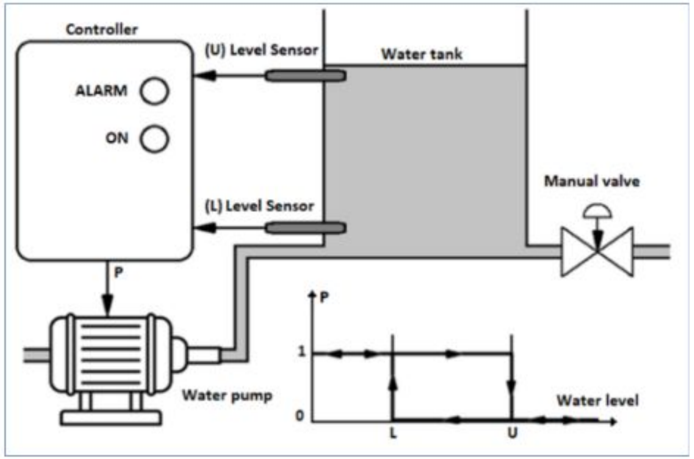
The figure to the right shows the connection between a water tank and its level Controller. The controller has two inputs: Upper-Level Sensor (U) and Lower-Level Sensor (L), which sense the level of the water inside the tank. Both sensors are connected through signal conditioning circuits to output logic high/low voltage levels. Controller output (P) is used to control a water pump. ON (Green) and ALARM (Red) LED indicators are used to show the state of the pump. The controller runs PROG1 (on the next page) on a TM4C microcontroller. The pump should be turned on (by applying a logic high on signal P), if the water level in the tank is below the low level until reaching the upper level, then it is turned off until the water level drops again below the lowerlevel.

If it happens that the pump is on for a specific time duration without having the water reaching the upper level, the pump is turned off and the ALARM LED is switched on for a specific duration. Then, the pump is switched on again (with ALARM LED off), continuing as normal.

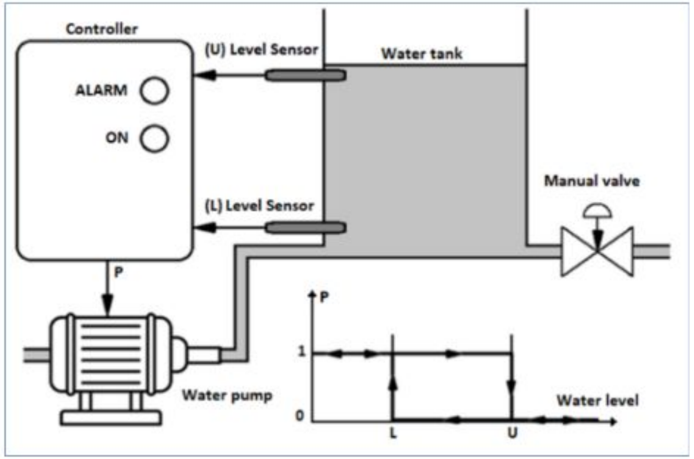


```
// defining constants
#define L_PIN 2
#define P_PIN 3
#define ALARM_LED_PIN 4
#define ON_LED_PIN 5
#define U_PIN 6
// defining pump states
#define PUMP_OFF 0
#define PUMP_ON 1
#define PUMP_ALARM 2
// counter threshold
#define THRESHOLD 10000
// global variables
int old_L , old_U , current_L , current_U , state;
int on_state_counter = 0;
state = PUMP_OFF;
```

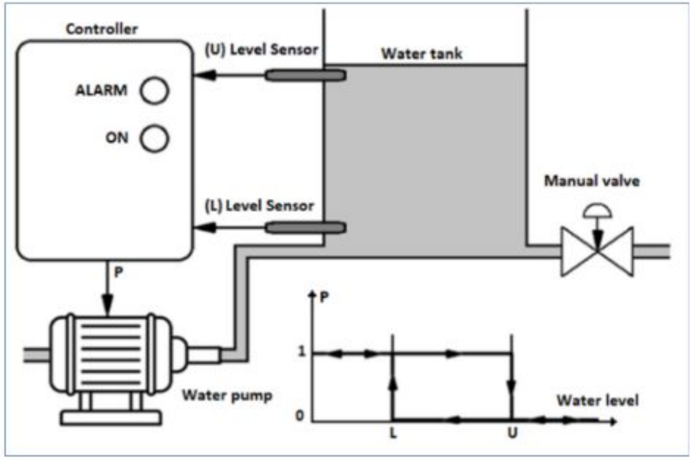


```
void setup()
{


}
```


Controller
ALARM
ON
(U) Level Sensor
Water tank
(L) Level Sensor
Manual valve
P
Water pump
P
1
0
L
U
Water level

```
void setup()
{
    pinMode(L_PIN,INPUT);
    pinMode(U_PIN,INPUT);
    pinMode(P_PIN,OUTPUT);
    pinMode(ON_LED_PIN,OUTPUT);
    pinMode(ALARM_LED_PIN,OUTPUT);
}
```



```
void loop()
{
      current_L = digitalRead(L_PIN);
      current_U = digitalRead(U_PIN);
      if(state==PUMP_OFF)
      {
            if(<C1>)  state = PUMP_ON;
            digitalWrite(P_PIN,V1);
            digitalWrite(ON_LED_PIN,V2);
            digitalWrite(ALARM_LED_PIN,V3);
      }
}
```

V1	V2	V3
LOW / HIGH	LOW / HIGH	LOW / HIGH

C1	
!current_L && !old_L	!current_L && old_L
current_L && !old_L	current_L && old_L

```
void loop()
{
    else if(state==PUMP_ON)
    {
        if(<C2>) state = PUMP_OFF;
        if(<C3>)
        {
            state = PUMP_ALARM;
            on_state_counter = 0;
        }
        if(!current_L) on_state_counter+=1;
        digitalWrite(P_PIN,V4);
        digitalWrite(ON_LED_PIN,V5);
        digitalWrite(ALARM_LED_PIN,V6);
    }
}
```

V4	V5	V6
LOW / HIGH	LOW / HIGH	LOW / HIGH

C2	
!current_U && !old_U	!current_U && old_U
current_U && !old_U	current_U && old_U

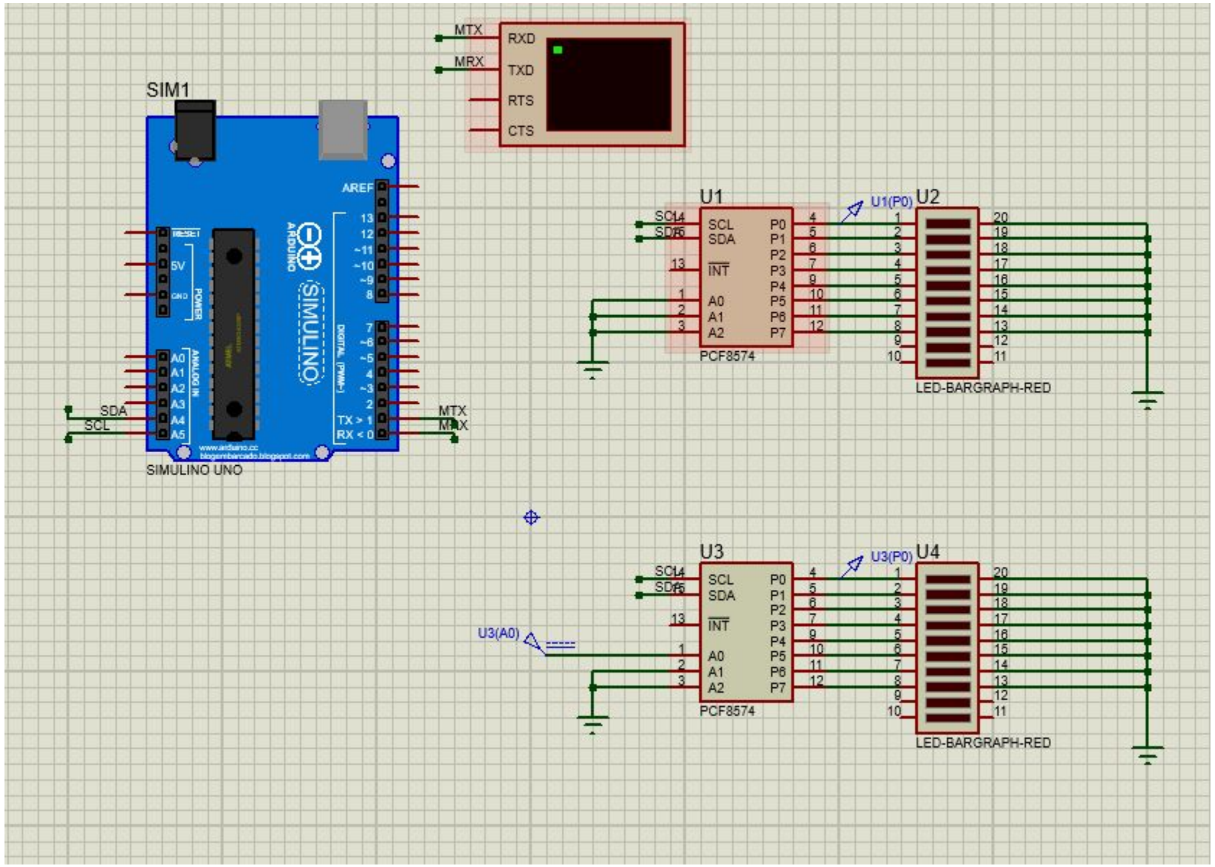
C3
!current_L && on_state_counter>THRESHOLD
!current_U && on_state_counter>THRESHOLD
current_L && on_state_counter<THRESHOLD
current_U && on_state_counter<THRESHOLD

```
void loop()
{
        else if(state==PUMP_ALARM)
        {
                on_state_counter+=1;
                if(<C4>)
                {
                        state = PUMP_ON;
                        On_state_counter = 0;
                }
                digitalWrite(P_PIN,V7);
                digitalWrite(ON_LED_PIN,V8);
                digitalWrite(ALARM_LED_PIN,V9);
        }
        old_L = current_L;
        old_U = current_U;
}
```

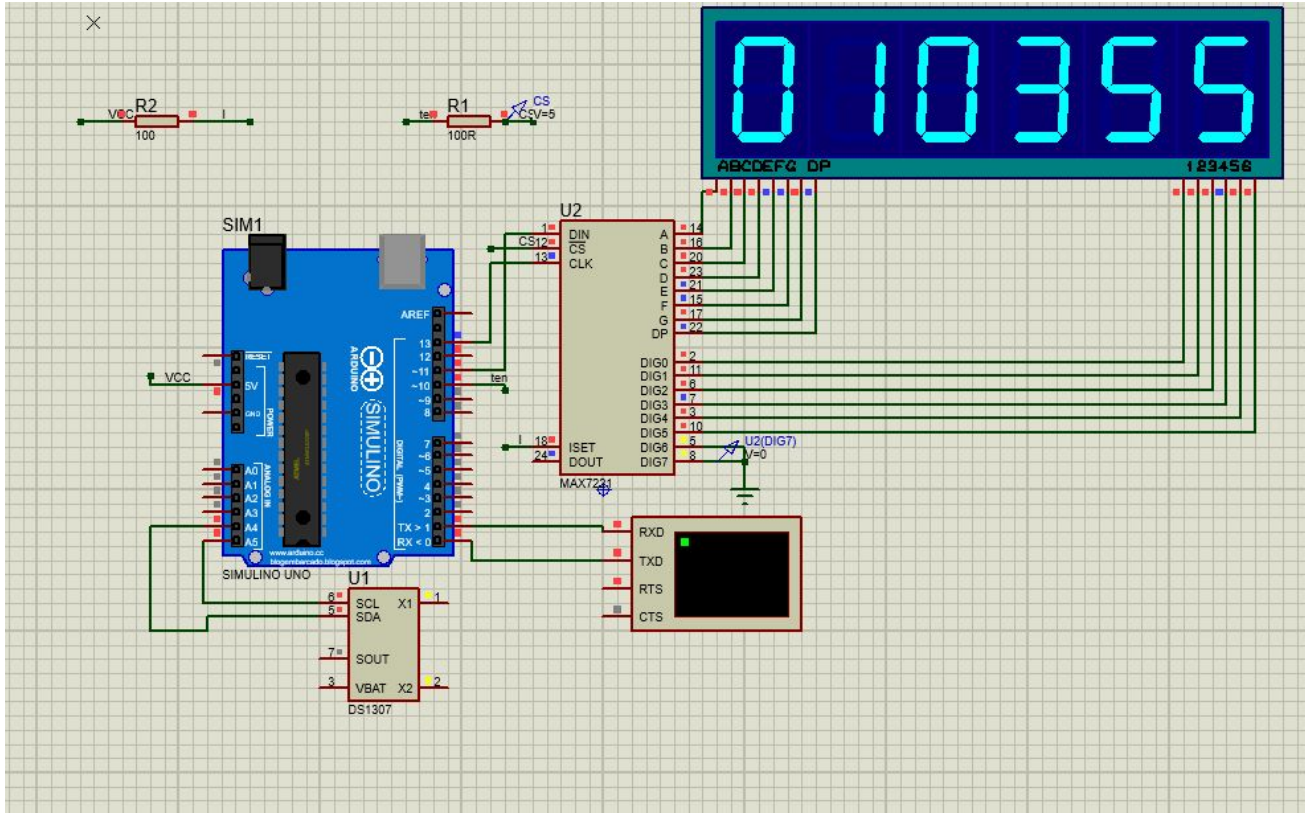
V7	V8	V9
[✓ LOW] / HIGH	[✓ LOW] / HIGH	LOW / [✓ HIGH]

C4
current_U == 1
old_U == 0
on_state_counter<THRESHOLD
[✓ on_state_counter>THRESHOLD]

Using I2C port expander to create a bouncing light system with 16 LED.



Develop a real time clock that shows (HH MM SS). Use following components: (I2C Real Time Clock, SPI 7-Segment Driver, Six 7-Segments Modules).


R2
100
VCC
R1
100R
CS
V=5
SIM1
SIMULINO UNO
U2
DIN
CS
CLK
ISET
DOUT
DIG0
DIG1
DIG2
DIG3
DIG4
DIG5
DIG6
DIG7
MAX7221
U2(DIG7)
V=0
U1
SCL
SDA
SOUT
VBAT
X1
X2
DS1307
RXD
TXD
RTS
CTS
ABCDEFG DP
123456
0 10355