```
// 2025 july: pumpTime
// limit pump time per day,
// keep track of weekly pump time
// Note: unsigned long millis() correct time
difference after overflow
// july 31: display time runnig today (from
msStartDay)
// aug 04: delay after SSR( OFF) to reduce glitches to
pump
/*
  LCD circuit:
 * LCD RS pin to digital pin 7
 * LCD Enable pin to digital pin 8
 * LCD D4 pin to digital pin 9
 * LCD D5 pin to digital pin 10
 * LCD D6 pin to digital pin 11
 * LCD D7 pin to digital pin 12
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K resistor:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)

predefined constants
   HIGH,LOW, INPUT,OUTPUT, true/false
*/
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the
interface pins
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

// description
// use pressostat
// - pin 1 to trigger arduino checks
// - pin 2 to enable arduino pumpOn() thru to
SolidState relay
// this way hardware protects from software errors
//  (sure stop pump when pressostat opens)
// allow Firewater_On button to bypass timer limits
//  sure enable pump in case of fire

////////// global vars ////////////
boolean FALSE= 0, TRUE= 1;
boolean _bSSR= FALSE;      // pump not running
boolean _bLCD= FALSE;      // LCD backlight off
char    _strBuf[17]= "****************";   // sprintf
buffer

// define I/O pins
int _iDayLimUp= 2; // max minutes/day up
int _iDayLimDn= 3; // down
int _iPresLow = 4; // pressostat pin high if pressure
low

int _oLCD     = 5; // LCD backlight
int _oSSR     = 6; // switch SSR power on for pump
int _oBeep    =13; // sound buzzer (for time-out or
FwOn)

// for timers to LCD
unsigned long _minsDayLim= 2;   // default 2 mins day
limit
unsigned long _msTotalDay= 3600000*24;    // ulong
total ms per day
unsigned long _msTotalWeek= 7* _msTotalDay; // total
ms per week
unsigned long _msLCDdelay= 10000; // 3 sec for testing
// for auto reset each day/ week
unsigned long _msStartSec= 0;    // allow update dayTime
every second
unsigned long _msStartDay;       // set in Setup/ reset
unsigned long _msStartWeek;      // "
unsigned long _msPumpDay= 0;     // day total
unsigned long _msPumpWeek= 0;    // week total
unsigned long _msPumpStart= 0;   // after SSR= ON
unsigned long _msLCDstart= 0;    // after LCD= ON



void resetTimers()
  // call by Setup() and after reset
{
   _msPumpDay= 0;
   _msPumpWeek= 0;
   _msPumpStart= 0;

   // day and week reference
   _msLCDstart=
   _msStartSec=
   _msStartDay=
   _msStartWeek= millis();

   // show LCD
   set_LCD( TRUE);
}


void beep( unsigned long ms)
{
   buzzer( TRUE);
   delay( ms);
   buzzer( FALSE);
}


void delay_SSR( int nSec)
   // delay on and off by nSec, with message
{

   // sound notify
   beep( 100);
   char str[8]= "";
   // count down in status
   sho_LCD( TRUE);
   lcd.setCursor( 8, 1);
   while (nSec> 0) {
      printf( str, "%2d", nSec);
      lcd.print( str);
      delay( 1000);   // seconds
      nSec--;
   }

}


void set_SSR( boolean bVal)
   // set output to SSR and _bSSR
   // 2025aug02: check ON again after delay to suppress
glitches
{
   if (bVal) {
      // delay ON before start
      delay_SSR( 3);
      // check after delay
      if (lowPressure()) {
         _msPumpStart= millis();
         digitalWrite( _oSSR, HIGH);
         _bSSR= TRUE;
```

```
         // set ON, quit
         return;
      }
   }

   // request low or pressure already high
   {
      _msPumpStart= 0;
      digitalWrite( _oSSR, LOW);
      _bSSR= FALSE;
      // delay OFF after stop, before exit
      delay_SSR( 4);
   }
}

void sho_LCD( bool bVal)
// just backlight
{
   if (bVal)
      digitalWrite( _oLCD, HIGH);
   else
      digitalWrite( _oLCD, LOW);
}


bool set_LCD( boolean bVal)
   // sho backlight
   // reset timer when ON
   // for wakeup call by PushButton or SSR
{
   // remember prev value
   boolean prvVal= _bLCD;
   // remember new value
   _bLCD= bVal;
   // set backlight
   sho_LCD( bVal);
   // set timer if HIGH
   if (bVal)
      _msLCDstart= millis();

   // return previous value
   return prvVal;
}


void setup()
// init LCD and pins
{

   // set up the LCD's number of columns and rows:
   lcd.begin(16, 2);

   // show layout on LCD
   lcd.setCursor( 0, 0);
   lcd.print( "start 1234567890");
   lcd.setCursor( 0, 1);
   lcd.print( "* hello world! *");
   delay( 500);

   lcd.setCursor( 0, 0);
   lcd.print( "dd 0:00 ww 00:00");
   lcd.setCursor( 0, 1);
   lcd.print( "limit 2 mins  ");

   // set pins to read
   pinMode( _iDayLimUp, INPUT);
   pinMode( _iDayLimDn, INPUT);
   pinMode( _iPresLow , INPUT); // pressure is low:
enable pump

   // pins to write
   pinMode( _oLCD     , OUTPUT); // LCD backlight power
   pinMode( _oSSR     , OUTPUT); // pass thru pressostat
again
   pinMode( _oBeep    , OUTPUT); // buzzer

   // init timers
   resetTimers();

   // stop SSR (pump)
   set_SSR( FALSE);
   buzzer( FALSE);

   // show LCD backlight, start timer
   set_LCD( TRUE);

   // 2025july6: to be sure
   _msTotalDay= 3600000*24;
   _msTotalWeek= 7* _msTotalDay;
}


////////// LCD functions //////////////
//// LCD helpers
void LCD_str( int col, int row, String str)
// show string at col, row
{
   sho_LCD( TRUE);           // show LCD backlight
   lcd.setCursor( col, row); // leave timer
   lcd.print( str);
}

void LCD_int( int col, int row, int iVal)
// show integer at col,row
{
   sho_LCD( TRUE);            // show LCD backlight
   lcd.setCursor( col, row);
   lcd.print( iVal);
}


//// LCD output
void LCD_dayTime()
   // how long running since reset day
   // show LCD (leave timer)
   // count down
{
   // ms lapsed
   unsigned long tNow= millis();
   // seconds lapsed
   unsigned long dts= (tNow- _msStartDay) /1000;
   // get hh:mm:ss to display
   int hh= int (dts/ 3600);
   dts-= 3600*hh;
   int mm= dts/ 60;
   dts-= 60* mm;
   int ss= int(dts);
   // format
   sprintf( _strBuf, "%02d:%02d:%02d", hh,mm,ss);
   //sho_LCD( TRUE);
   LCD_status( _strBuf);
}
```

```
void LCD_minsDayLim()
// show day limit [min])
{
   // format string
   sprintf( _strBuf, "%2lu", _minsDayLim);
   set_LCD( TRUE);
   LCD_str( 5, 1, _strBuf);
}

void LCD_status( String str)
// show status
{
   LCD_str( 8, 1, str);
}


void ms_to_minSec( unsigned long ms, int& mins, int&
sec)
// calc minutes and seconds
// 2025july05: 60*1000 from mins to ms
{
   // get the whole minutes (max 10080 per week)
   unsigned long m= ms/ 60000;
   // find the leftover ms
   unsigned long ms_left= ms- 60000* m;
   // cast to int
   mins= (int) m;
   // get the first decimal value
   sec= (int) (ms_left/ 1000);
}


void LCD_minsPumpDay( unsigned long msPumpDay)
//show minutes and decimal [mm.d]
// 2025july02: from running lapse (not final
_msPumpDay)
{
   // get minutes and decimal time
   int mins, sec;
   ms_to_minSec( msPumpDay, mins, sec);

   // format string
   sprintf( _strBuf, "%2d:%02d", mins, sec);
   // show
   LCD_str( 2, 0, _strBuf);
}


void LCD_minsPumpWeek( unsigned long msPumpWeek)
// show minutes and decimal for this week's total
{
   // get minutes and decimal time
   int mm,ss;
   ms_to_minSec( msPumpWeek, mm, ss);
   // format string
   sprintf( _strBuf, "%3d:%02d", mm, ss);
   // show
    LCD_str( 10, 0, _strBuf);
}


///// timers ////////
void reset_day()
{
   unsigned long tNow= millis();
   // if pump running:
   if (_bSSR) {
       // get lapsed before reset
       unsigned long msDt= millis()- _msPumpStart;
       _msPumpWeek+= msDt; // add lapse to weektime
       _msPumpStart= tNow; // reset start for the day
   }

   _msPumpDay= 0;
   _msStartDay= tNow;
   set_LCD( TRUE);
   LCD_status( "reset day");
   LCD_minsPumpDay( _msPumpDay);           // show 0 day
mins
}


void chkTimeReset()
{
   // check daily reset
   unsigned long tNow= millis();

   // check second lapsed
   if (tNow- _msStartSec> 1000) {
      _msStartSec= tNow;
      // if dayTime needs update
      if (!_bSSR    // pump OFF
      && _bLCD)     // and backlight ON
         LCD_dayTime(); // show DayTime (since reset)
   }

   // check LCD backlight timeout
   if (tNow- _msLCDstart> _msLCDdelay)
      set_LCD( FALSE);

   // check daily reset
   if (tNow- _msStartDay> _msTotalDay)
      reset_day();

   // check weekly reset
   if (tNow- _msStartWeek> _msTotalWeek) {
      reset_day();
      _msPumpWeek= 0;
      _msStartWeek= tNow;
      LCD_status( "reset wk ");
      LCD_minsPumpWeek( _msPumpWeek);    // show 0 week
mins
   }
}


//////// Buttons ////////////////////
bool resetPressed() {
   if (digitalRead( _iDayLimUp)
   &&  digitalRead( _iDayLimDn)) {
      if (!set_LCD( TRUE)) {  // show LCD backlight
         delay( 500);         // start timer
         return FALSE;        // no action
      }
      LCD_status( "reset");
      resetTimers();
      set_SSR( FALSE);
      LCD_minsPumpDay( 0);
      LCD_minsPumpWeek( 0);
      delay( 500);
      return TRUE;
   } else
      return FALSE;
}
```

```
bool dayLimUpPressed() {
   if (digitalRead( _iDayLimUp)
   &&  _minsDayLim< 99) {             // max 2 digits
      if (!set_LCD( TRUE)) {          // show LCD
backlight
         delay( 500);                 // suppress
double click
         return FALSE;                // just wake up,
and exit
      }
      _minsDayLim++;
      LCD_minsDayLim();
      delay( 500);
      return TRUE;
   } else
      return FALSE;
}


bool dayLimDnPressed() {
   if (digitalRead( _iDayLimDn)
   && _minsDayLim> 1) {                // min 1
      if (!set_LCD( TRUE)) {           // show LCD
backlight
         delay( 500);
         return FALSE;                 // no action
      }
      _minsDayLim--;
      LCD_minsDayLim();
      delay( 500);
      return TRUE;
   } else
      return FALSE;
}


void chkButtons()
// check up/dn/reset daylim
{
   if (resetPressed()
   || dayLimUpPressed()
   || dayLimDnPressed())
      ; // set_LCD already called
}


void buzzer( bool bVal){
   if (bVal)
      digitalWrite( _oBeep, HIGH);
   else
      digitalWrite( _oBeep, LOW);
}


bool pumpTimeOut()
   // true if pump time out
{
   unsigned long msDt= millis()- _msPumpStart;
   if (_msPumpDay+ msDt> _minsDayLim* 60000) {
      set_SSR( FALSE);         // stop pump
      LCD_status( "time-out");
      _msPumpDay+= msDt;       // count time
      _msPumpWeek+= msDt;      // .. oops
      _msPumpStart= 0;         // mark starter
      delay( 500);             // reduce flicker
      return TRUE;
   } else
      return FALSE;
}


bool lowPressure()
  // glitches: check after delay in caller
{
   return digitalRead( _iPresLow);
}


void chkPump(){
   // 2025july07: ignore time out when dayLim
   // is made less than msPumpDay when not running
   // check low pressure
   if (lowPressure()) {
      // not started pumping yet
      // 2025july05: set _msPumpStart before timeOut
      if (!_bSSR) {
         LCD_status( "pumping ");
         set_SSR( TRUE); // set _msPumpStart first
      // then check time out
      } else if (pumpTimeOut()) {
         while (!resetPressed()   // reset to clear
         && !dayLimUpPressed()) { // allow continue
            buzzer( TRUE);
         }
         // 2025july07: exit to avoid wrong LCD times
         buzzer( FALSE);
         return;
      }

      // OK continue pumping: get lapsed time
      unsigned long msDt= millis()- _msPumpStart;
      LCD_status( "pumping ");
      // show progress
      LCD_minsPumpDay( _msPumpDay+ msDt);
      LCD_minsPumpWeek( _msPumpWeek+ msDt);

   // high pressure: we're done
   } else {
      // if was on:
      if (_bSSR) {
         unsigned long msDt= millis()- _msPumpStart;
         // add run time
         _msPumpDay+= msDt;
         _msPumpWeek+= msDt;
         set_SSR( FALSE);
         LCD_dayTime();
         set_LCD( TRUE); // count down
         LCD_minsPumpDay( _msPumpDay);
         LCD_minsPumpWeek( _msPumpWeek);
      }
   }
} // chkPump()


void loop() {
   chkTimeReset();  // opt. reset day/week timers

   chkButtons(); // for daylimit up/down OR reset timers

   chkPump();
}
```