# Local UI for OpenSprinkler – Design

## Introduction

Local UI for the OpenSprinkler is the UI available on the device itself. It uses on-device LCD screen and few control buttons OS has. The UI is capable of running on the basic 1602 LCD that is standard for OpenSprinkler, and requires minimum of 3 buttons (standard OS hardware), however the UI is capable of working with bigger displays and with more buttons. Functionally the UI is a super-set of the standard OS local UI.

Local UI for OpenSprinkler is intended to allow basic OS operation using device LCD and keys. Only a sub-set of OS functionality is exposed in the local UI, mainly focused on configuring the network and manually operating the device – which is important for irrigation system adjustments, tests etc.

There are two big complexities/constrains for the local UI – small size of LCD (2 lines 16 chars each) and small number of buttons (only 3 on the standard OS).

The UI is modeled after (an earlier) Orbit sprinkler timer. The UI is structured as two-layer model of “Modes” and “Pages” within modes. Each Mode is functionally dedicated to something – e.g. configuration display, valve control etc. Pages are providing necessary space within each Mode to accommodate more data and controls.

## Local UI Buttons

The UI is controlled by 3 (standard) or 4 (optional) buttons. Buttons are assigned as following:

1. Mode switch button. This button always switches between Modes.
2. Up/Change button. This button changes (increases) values or Pages within the Mode.
3. Down/Change button is optional. It compliments #2 button allowing more convenient up/down changes. Without this button (e.g. standard 3-button OS) the user must use Up button only iterating through all possible values in the loop to select required one.
4. Confirm/Enter button. This button confirms/saves changes made by button#2/#3. If not confirmed changes will be lost on Mode change or timeout.

The UI also supports “push and hold” operations on button#4, invoking “special” commands like reboot.

## Local UI Modes

Key Modes:

1. Status (Home Screen)

This is the standard OS Home screen, showing date/time, network and stations status

Pushing Up/Down buttons in this Mode switches between extension boards (like it happens in the existing OS).

Pushing Confirm/Enter button toggles Enabled/Disabled OS status.

Note: when status changes to Disabled, watering programs are suspended, and manually started job(s) are cancelled.

1. Manual Mode

This Mode allows manual start of valves. Manual start is time-limited, and when manually starting a valve user must specify the number of minutes to run.

In this Mode Up/Down buttons select the valve to control, and once Confirmed the user is asked to select the number of minutes to run (again using Up/Down and then Confirm buttons).

Once this data is entered the timer goes back to the Status/Home screen showing valves status etc.

1. “View Settings” Mode

This Mode displays IP address, network name, gateway IP and other configuration information. Functionally it is an expanded version of the standard OS functionality of showing IP and GW addresses.

1. Setup Mode

This Mode allows configuring IP address etc. Essentially, it is the standard OS setup just initiated via UI rather than reset.

## Local UI – Code Design

Local UI is running all the time (it is active all the time while OS is running), and because of that it must implement all its functionality in a wait-less manner. Local UI class is called twice – once to initialize (from setup), and also from the main loop(). All UI operation, including information display, keys input etc is handled in loop() with no delays – it must return as soon as possible to allow OS to handle other duties – such as WEB UI, schedules/programs etc.

Because of this, we cannot use any delay() calls within the Local UI, and all time-related things are handled using millis() and time counters. E.g. “push and hold” button operations are handled using millis counters.

The local UI is implemented mostly in code. There are three global state variables that store current UI mode/state:

static byte OSLocalUI::osUI\_State - current UI state. State could be enabled/suspended etc  
static byte OSLocalUI::osUI\_Mode - current UI Mode  
static byte OSLocalUI::osUI\_Page - current UI Page

Each UI Mode has a piece of code behind that handles that mode – a Handler. That code is responsible for displaying appropriate information on the screen and for reacting to button presses. That code also handles multiple pages within the Mode.

In theory it is possible to handle all modes in one piece of code (one function), but for code readability it makes sense to have a separate function for each Mode.