SwitchDoc Labs

Quick Start Manual

July 2020
Version 1.2

SmartGarden System

Table of Contents

Cautions when building and using The Smart Garden System

1) **Keep all water away from the electronics and power supply at all times!**
2) **The Smart Garden System is designed for indoor use only and should be placed in a dry environment where no water or rain can reach to avoid short circuiting the electronics**
3) **Insert the moisture sensor into the CENTER of the flower pot, and keep it near the center of the plant and away from the water coming out of the holes cut in the watering pipe.**
4) **This is not a toy! Keep it out of reach of young children and pets.**
5) **SwitchDoc Labs assumes no liabilities in the use of this kit, beyond the refund of the purchase price.**

Errata

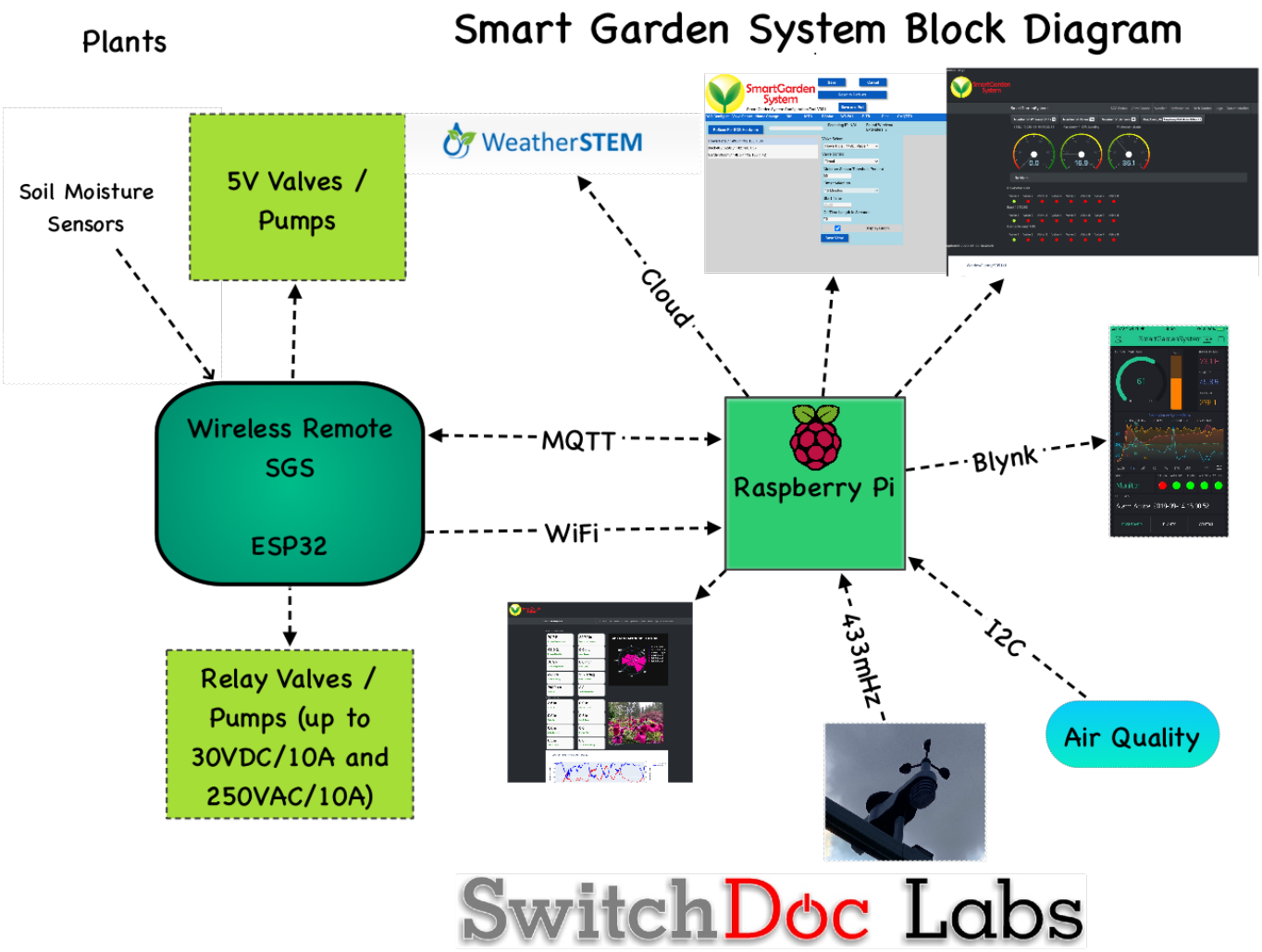
What is The Smart Garden System?

Did you ever want to build your own remote monitoring and management system for your indoor or outdoor garden? Do you want to share your garden and the weather world wide? This project is for you. You can learn the Raspberry Pi and how to connect to the real world through this easy to build **no-soldering** kit. You can measure soil moisture and then use that as feedback to provide your plant or garden just the right about of water.
Highly expandable! You can have your Raspberry Pi Base Unit inside your house and have multiple wireless control units in your outdoor garden, greenhouse or in the upstairs bedroom. Up to 250 wireless control units can be connected up to one Raspberry Pi base unit. This allows you to control your truly MASSIVE garden. Or your small one. Either way!
This is a perfect highly expandable project kit for kids with some help from the adults and for adults trying to learn some new things. We have done this before with our successful OurWeather KickStarter so we know what we are talking about. People all over the world have built the OurWeather weather station with great success. This project has **no soldering** involved and uses Grove connectors to wire everything up! You can't reverse them and blow things up. Here is our tutorial on the Grove system.

The Smart Garden System Features

- Control Valves, Pumps and Lights
- Measure your Soil Moisture
- Measure your Sunlight
- Measure your Air Quality
- Measure your Weather
- Show your results on the Internet and your Phone

Easy to build. Easy to learn about the IOT (Internet Of Things) and the Raspberry Pi.



Quick Start

This is a quick start manual for use if you have purchased your Smart Garden System assembled and tested.

This manual assumes that you have either purchased an assembled and tested Smart Garden System or you have completed the assembly and testing of your Smart Garden System as described in the "Smart Garden System Assembly and Testing Manual". If you haven't completed the assembly and testing, go back and complete it now. Make sure your Wireless Extender unit(s) are plugged in and connected to your WiFi network.

The first thing to figure out is what your Raspberry Pi IP address is. Follow the procedures here:

https://learn.pimoroni.com/tutorial/raspberry-pi/finding-your-raspberry-pi

In a terminal window, change directories down to:

```
cd SDL_Pi_SmartGardenSystem2
```

Run SGSConfigure.py

```
sudo python3 SGSConfigure.py
```

You will see something like this:

```
pi@SwitchDocLabs:~/SDL_Pi_SmartGardenSystem2 $ sudo python3 SGSConfigure.py
SGSConfigure.py:24: DeprecationWarning: AppURLopener style of invoking requests is deprecated. Use
newer urlopen functions/methods
  myURLOpener = AppURLopener()
remi.server      INFO     Started httpserver http://0.0.0.0:8001/
remi.request     INFO     built UI (path=/)
SGS.JSON File does not exist
SGSConfiguration.JSON File does not exist
```

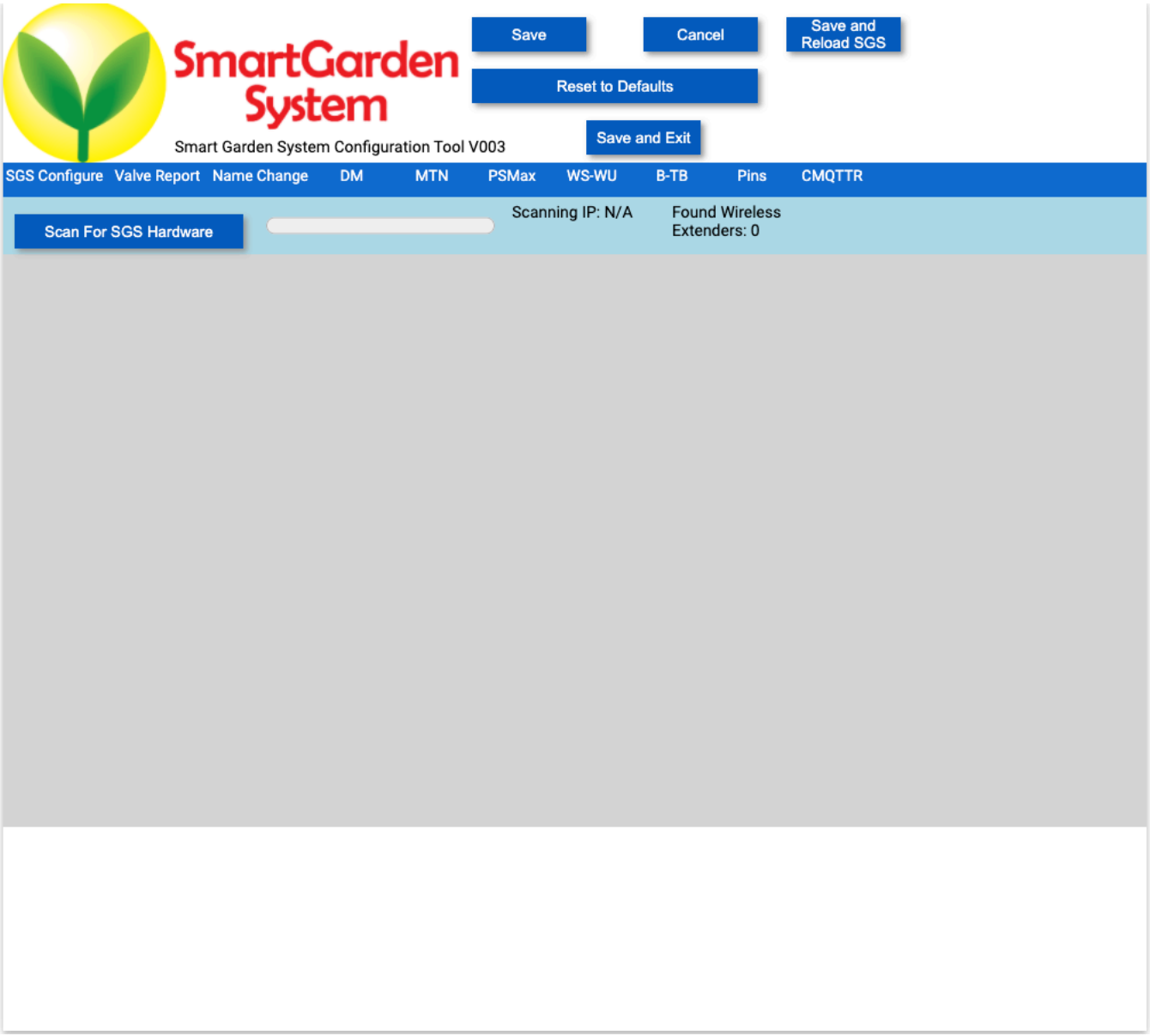
Now open up a browser window (either on the Raspberry Pi or on another computer on your local WiFi network) and enter this URL:

On your Raspberry Pi: http://127.0.0.1:8001/

On another computer type: http://xxx.xxx.xxx.xxx:8001/

Where “xxx.xxx.xxx.xxx” is the IP address of your Raspberry Pi that you wrote down above.

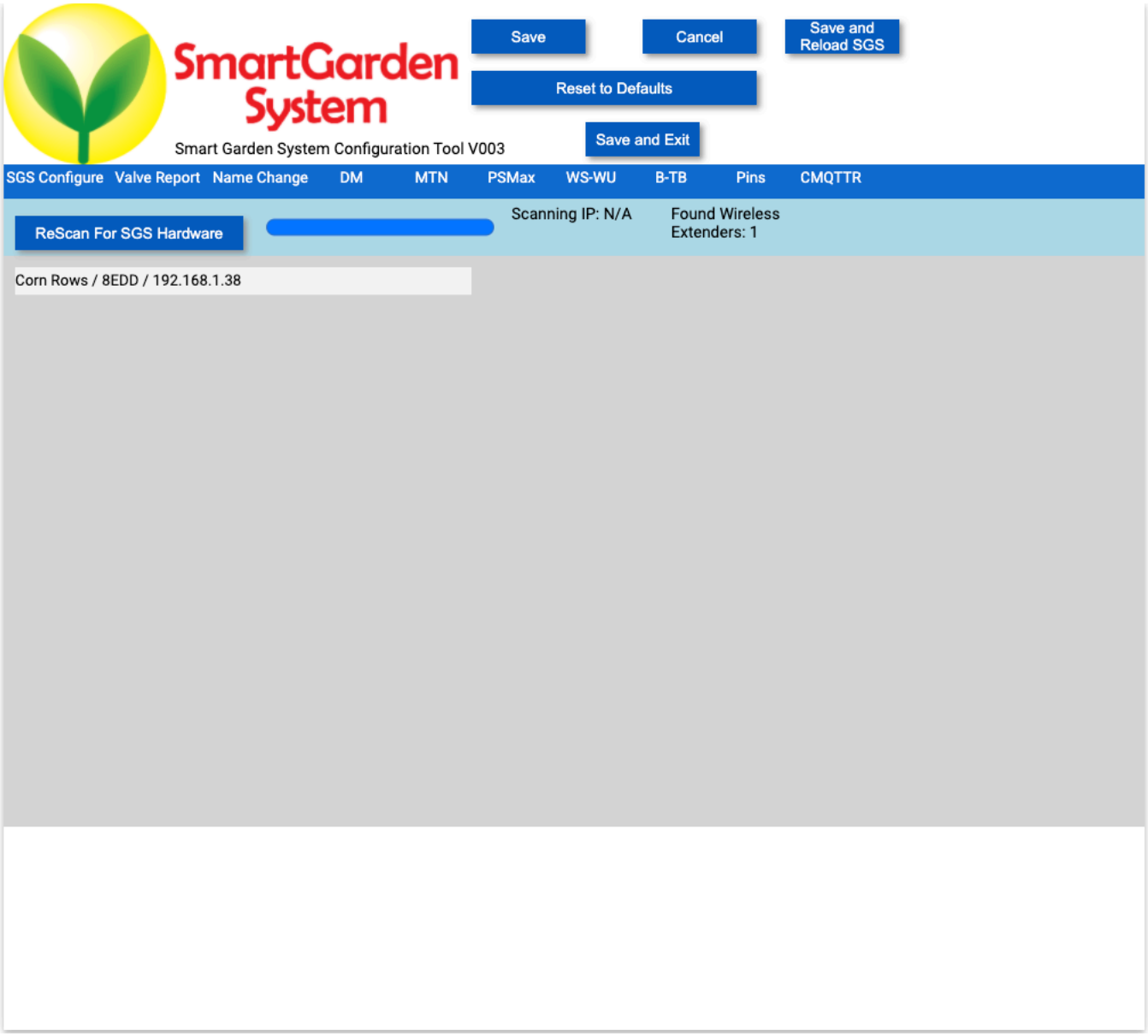
You will see this screen on your browser:


SmartGarden System
Smart Garden System Configuration Tool V003
Save
Cancel
Save and Reload SGS
Reset to Defaults
Save and Exit
SGS Configure
Valve Report
Name Change
DM
MTN
PSMax
WS-WU
B-TB
Pins
CMQTTR
Scan For SGS Hardware
Scanning IP: N/A
Found Wireless Extenders: 0

Scan for Wireless Units

The first thing we do is have the SGSConfigure software scan for all of your wireless units. Your base unit comes with one system.
 This takes a while (15 or 20 minutes).

Click on the “Scan For SGS Hardware Button”.



SGSConfigure found one SGS Wireless Unit: “Corn Rows/8EDD/192.168.1.38”. Note that we had named this uinit “Corn Rows” earlier. Yours will probably show up blank.

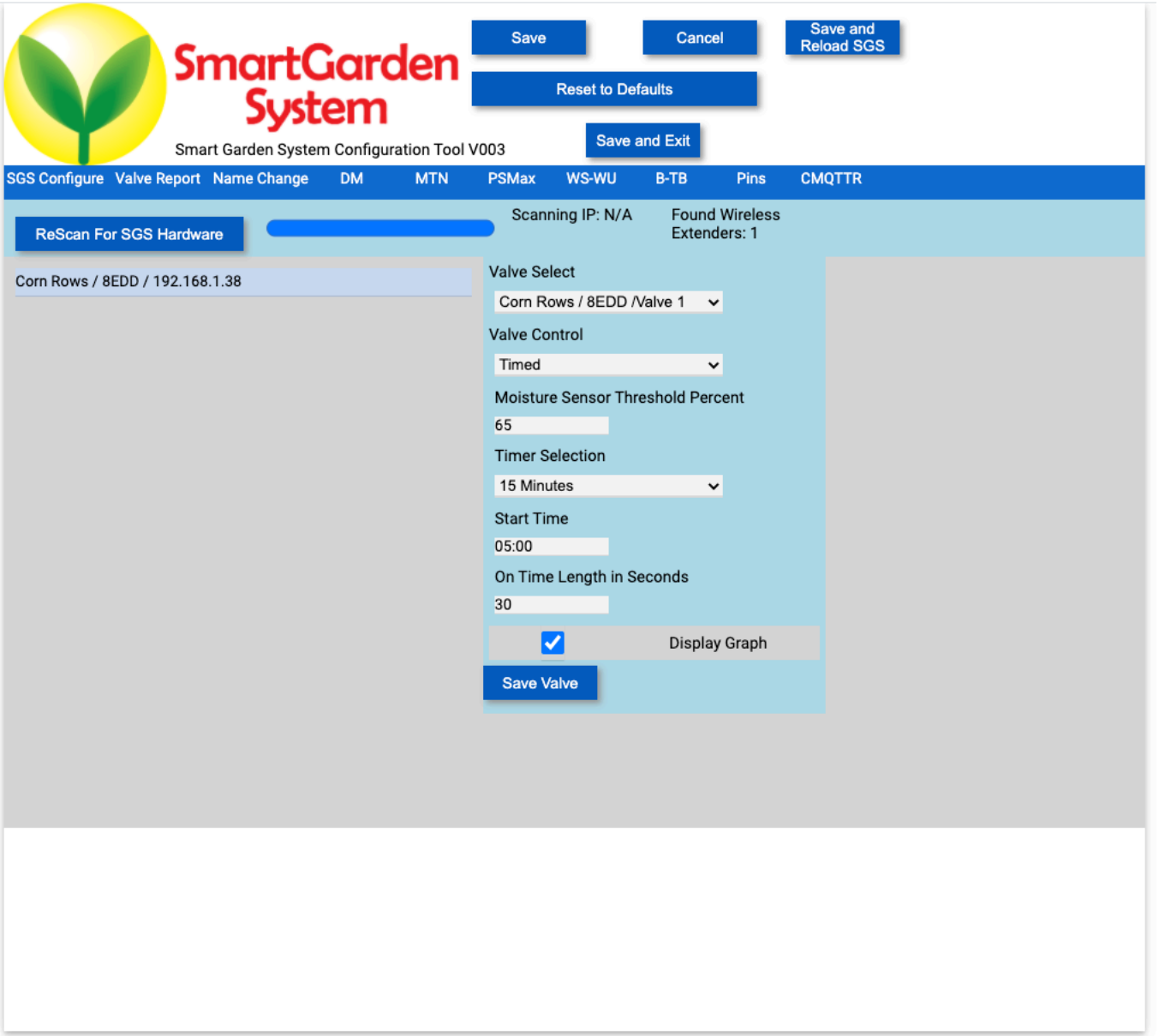
Setting a Test output on your SGS Wireless Unit

Click on the Wireless Unit on the screen to open up the Valve configuration menus. "Corn Rows/8EDD/192.168.1.38"in our example.

Select valve 1 – using the Valve Select menus (make sure you still have the USB Light Stick plugged into Valve 1) to turn on for 30 seconds every 15 minutes and click Show Graph.

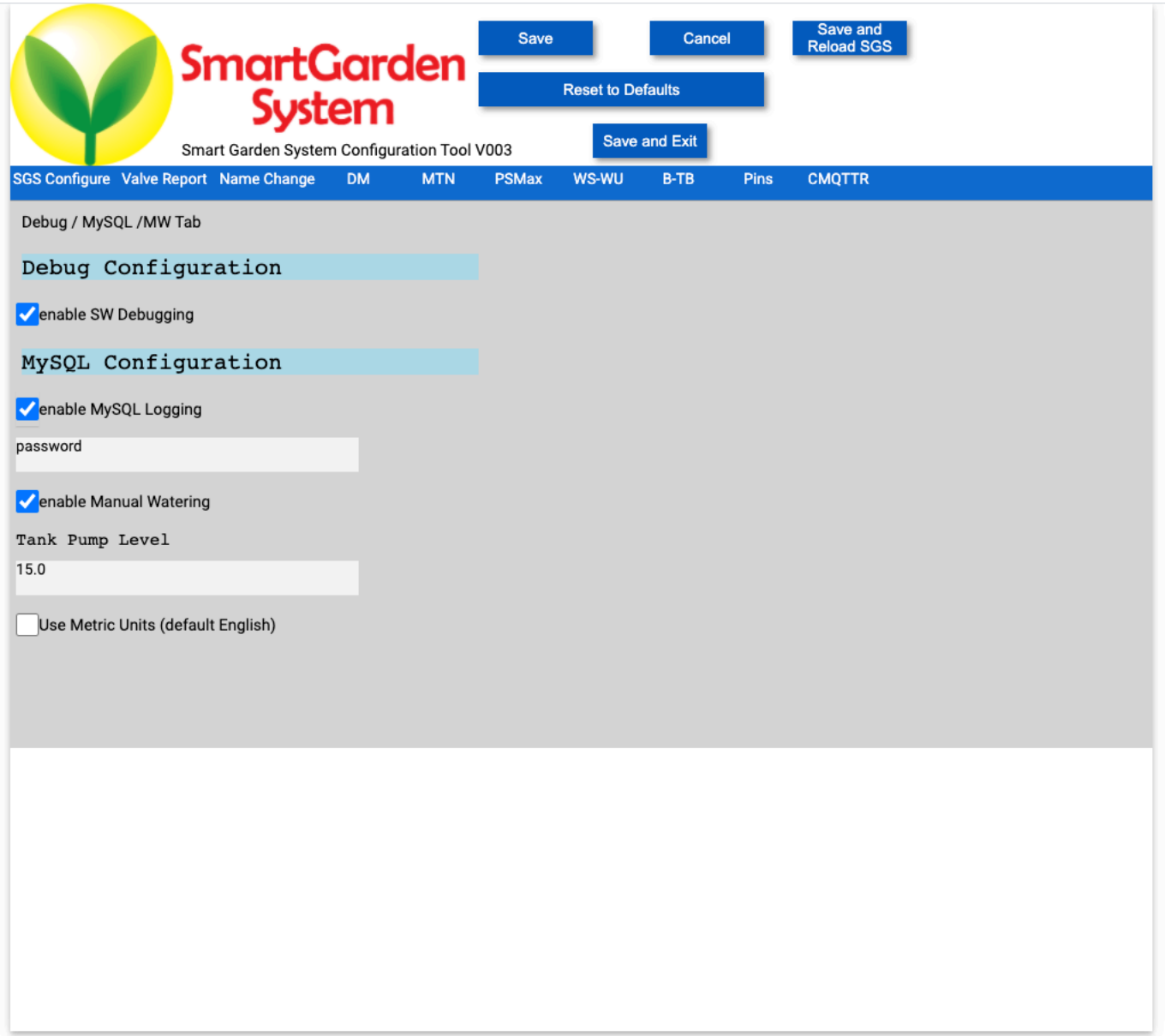
Note that you can only select the Timer Selection and Start Time if you have selected "Timed" in the Valve Control dropdown menu.

You will have this screen below:



Finally click the "Save Valve" button. If you don't do this, the valve changes are NOT saved.

Click on the “DM” Tab on your menu.



Click on enable SW Debugging (you can turn this off later).

Click on enable MySQL logging. The default MySQL password on the SDL SD Card is “password”.

Click on Save and Exit which saves your JSON files for SGS2 and quits the SGSConfigure program.

Note: On some systems, you may have to hit either “ctrl-c” or the “ctrl-|” to get the server to quit in the terminal window.

Look at the JSON files to see what you have done! SGS.JSON is the general configuration file, while SGSConfiguration.JSON is the valve/pump/timers configuration file for all the wireless extender units.

```
pi@SwitchDocLabs:~/SDL_Pi_SmartGardenSystem2 $ more SGS.JSON

{"key": "value", "ProgramName": "SmartGardenSystem2", "ConfigVersion": "001", "S
WDEBUG": true, "enable_MySQL_Logging": true, "English_Metric": false, "MySQL_Pas
sword": "password", "mailUser": "yourusername", "mailPassword": "yourmailpasswor
d", "notifyAddress": "you@example.com", "fromAddress": "yourfromaddress@example.
com", "enableText": false, "textnotifyAddress": "yournumber@yourprovider", "enab
lePixel": false, "pixelPin": "21", "SolarMAX_Present": false, "SolarMAX_Type": "
LEAD", "BMP280_Altitude_Meters": "626.0", "Sunlight_Gain": "High", "weather": fa
lse, "USEWEATHERSTEM": false, "INTERVAL_CAM_PICS__SECONDS": "60", "STATIONKEY":
"", "WeatherUnderground_Present": false, "WeatherUnderground_StationID": "KWXXXX
X", "WeatherUnderground_StationKey": "YYYYYY", "USEBLYNK": false, "BLYNK_AUTH":
"", "AS3935_Lightning_Config": "[2,1,3,0,3,3]", "REST_Enable": false, "Camera_Ni
ght_Enable": false, "MQTT_Enable": false, "MQTT_Server_URL": "", "MQTT_Port_Numb
er": "5900", "MQTT_Send_Seconds": "500", "manual_water": true, "Tank_Pump_Level"
: "15.0", "WirelessDeviceJSON": [{"return_value": 0, "id": "8EDD", "name": "Corn
 Rows", "ipaddress": "192.168.1.38", "hardware": "esp32", "return_string": "8EDD
,1,1,1,1,024", "connected": true}], "UltrasonicLevel": "4"}

pi@SwitchDocLabs:~/SDL_Pi_SmartGardenSystem2 $ more SGSConfiguration.JSON

{"SGSConfigVersion": "001", "Valves": [{"id": "8EDD", "ValveNumber": 1, "Control
": "Timed", "MSThresholdPercent": "65", "TimerSelect": "15 Minutes", "StartTime"
: "05:00", "OnTimeInSeconds": "30", "ShowGraph": true}, {"id": "8EDD", "ValveNum
ber": 2, "Control": "Off", "MSThresholdPercent": "65", "TimerSelect": "Daily", "
StartTime": "05:00", "OnTimeInSeconds": "10", "ShowGraph": false}, {"id": "8EDD"
, "ValveNumber": 3, "Control": "Off", "MSThresholdPercent": "65", "TimerSelect":
 "Daily", "StartTime": "05:00", "OnTimeInSeconds": "10", "ShowGraph": false}, {"
id": "8EDD", "ValveNumber": 4, "Control": "Off", "MSThresholdPercent": "65", "Ti
merSelect": "Daily", "StartTime": "05:00", "OnTimeInSeconds": "10", "ShowGraph":
 false}, {"id": "8EDD", "ValveNumber": 5, "Control": "Off", "MSThresholdPercent"
: "65", "TimerSelect": "Daily", "StartTime": "05:00", "OnTimeInSeconds": "10", "
ShowGraph": false}, {"id": "8EDD", "ValveNumber": 6, "Control": "Off", "MSThresh
oldPercent": "65", "TimerSelect": "Daily", "StartTime": "05:00", "OnTimeInSecond
s": "10", "ShowGraph": false}, {"id": "8EDD", "ValveNumber": 7, "Control": "Off"
, "MSThresholdPercent": "65", "TimerSelect": "Daily", "StartTime": "05:00", "OnT
imeInSeconds": "10", "ShowGraph": false}, {"id": "8EDD", "ValveNumber": 8, "Cont
rol": "Off", "MSThresholdPercent": "65", "TimerSelect": "Daily", "StartTime": "0
5:00", "OnTimeInSeconds": "10", "ShowGraph": false}]}
```

Initial Testing of your Smart Garden System

Start your SGS2 system in a terminal window by typing:

```
sudo python3 SGS2.py
```

You will see something similar to this as it scrolls across your screen. By the way, if you turn Software debugging ON (table “DM” - Debug configuration on) in SGSConfigure you will see much more on your terminal window.

pi@SwitchDocLabs:~/SDL_Pi_SmartGardenSystem2 $ sudo python3 SGS2.py
b"
b"
##
SGS2 Version 014 - SwitchDoc Labs
##

Program Started at:2020-07-14 11:00:10

```
SGS.JSON File exists
SGSConfiguration.JSON File exists
----------
Local Devices
----------
OLED:           Not Present
BMP280:         Present
DustSensor:     Not Present
----------
Checking Wireless SGS Devices
----------
Corn Rows - 8EDD:       Present
subscribing to  SGS/8EDD


----------
Plant / Sensor Counts
----------
Wireless Unit Count: 1
Sensor Count:  4
Valve Count:  8

----------
Other Smart Garden System Expansions
----------
Weather:            Not Present
GardenCam:          Present
SunAirPlus:         Not Present
SolarMAX:           Not Present
Lightning Mode:             Not Present
MySQL Logging Mode:             Present
UseBlynk:           Not Present

----------
----------
Scheduled Jobs
----------
Jobstore default:
   blinkLED (trigger: interval[0:00:05], next run at: 2020-07-14 11:00:19 PDT)
   checkForButtons (trigger: interval[0:00:10], next run at: 2020-07-14 11:00:24 PDT)
   statusLEDs (trigger: interval[0:00:15], next run at: 2020-07-14 11:00:29 PDT)
   checkForAlarms (trigger: interval[0:00:15], next run at: 2020-07-14 11:00:29 PDT)
   manualCheck (trigger: interval[0:00:15], next run at: 2020-07-14 11:00:33 PDT)
   valveCheck (trigger: interval[0:01:00], next run at: 2020-07-14 11:01:18 PDT)
   tick (trigger: interval[0:05:00], next run at: 2020-07-14 11:05:14 PDT)
   readWiredSensors (trigger: interval[0:08:20], next run at: 2020-07-14 11:08:34 PDT)
   updateDeviceStatus (trigger: interval[0:12:00], next run at: 2020-07-14 11:12:14 PDT)
----------
```

Wireless MQTT Message received: b'{"id": "8EDD", "messagetype": "1", "timestamp": "07/14/2020 18:02:18", "valvestate": "V10000000"}'
Wireless MQTT Message received: b'{"id": "8EDD", "messagetype": "1", "timestamp": "07/14/2020 18:02:48", "valvestate": "V00000000"}'

When 15 minutes have passed (after the USB Light Stick has turned on and off) minutes, hit “ctrl-c’ to quit.

With software debugging ON, you would have seen this:

```
pi@SwitchDocLabs:~/SDL_Pi_SmartGardenSystem2 $ sudo python3 SGS2.py
b''
b''
###############################################
SGS2 Version 014  - SwitchDoc Labs
###############################################

Program Started at:2020-07-14 10:24:57

SGS.JSON File exists
SGSConfiguration.JSON File exists
----------------------
Local Devices
----------------------
OLED:               Not Present
BMP280:             Present
DustSensor:               Not Present
----------------------
Checking Wireless SGS Devices
----------------------
myURL= http://192.168.1.38/setValves?params=admin,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
return= {'return_value': 0, 'id': '8EDD', 'name': 'Corn Rows', 'ipaddress': '192.168.1.38',
'hardware': 'esp32', 'return_string': '', 'connected': True}
Corn Rows - 8EDD:                 Present
MQTT:  Sending CONNECT (u0, p0, wr0, wq0, wf0, c1, k60) client_id=b'SGS2'
MQTT:  Received CONNACK (0, 0)
subscribing to  SGS/8EDD
MQTT:  Sending SUBSCRIBE (d0, m1) [(b'SGS/8EDD', 0)]
MQTT:  Received SUBACK


----------------------
Plant / Sensor Counts
----------------------
Wireless Unit Count: 1
Sensor Count:  4
Valve Count:  8

----------------------
Other Smart Garden System Expansions
----------------------
Weather:            Not Present
GardenCam:          Present
SunAirPlus:               Not Present
SolarMAX:           Not Present
Lightning Mode:           Not Present
MySQL Logging Mode:               Present
UseBlynk:           Not Present

----------------------
myURL= http://192.168.1.38/enableMoistureSensors?params=admin,1,1,1,1
myURL= http://192.168.1.38/readMoistureSensors?params=admin
-----------------
MoistureSensorStates
[{'id': '8EDD', 'sensorType': 'C1', 'sensorNumber': '1', 'sensorValue': '77.85', 'timestamp':
'2020-07-14 10:25:05'}, {'id': '8EDD', 'sensorType': 'C1', 'sensorNumber': '2', 'sensorValue':
```

```
'100.00', 'timestamp': '2020-07-14 10:25:05'}, {'id': '8EDD', 'sensorType': 'C1', 'sensorNumber':
'3', 'sensorValue': '100.00', 'timestamp': '2020-07-14 10:25:05'}, {'id': '8EDD', 'sensorType':
'C1', 'sensorNumber': '4', 'sensorValue': '100.00', 'timestamp': '2020-07-14 10:25:05'}]
-----------------
-----------------
Scheduled Jobs
-----------------
Jobstore default:
    blinkLED (trigger: interval[0:00:05], next run at: 2020-07-14 10:25:06 PDT)
    checkForButtons (trigger: interval[0:00:10], next run at: 2020-07-14 10:25:11 PDT)
    statusLEDs (trigger: interval[0:00:15], next run at: 2020-07-14 10:25:16 PDT)
    checkForAlarms (trigger: interval[0:00:15], next run at: 2020-07-14 10:25:16 PDT)
    manualCheck (trigger: interval[0:00:15], next run at: 2020-07-14 10:25:20 PDT)
    valveCheck (trigger: interval[0:01:00], next run at: 2020-07-14 10:26:05 PDT)
    tick (trigger: interval[0:05:00], next run at: 2020-07-14 10:30:01 PDT)
    readWiredSensors (trigger: interval[0:08:20], next run at: 2020-07-14 10:33:21 PDT)
    updateDeviceStatus (trigger: interval[0:12:00], next run at: 2020-07-14 10:37:01 PDT)
-----------------
MQTT:  Received PUBLISH (d0, q0, r0, m0), 'SGS/8EDD', ...  (175 bytes)
Wireless MQTT Message received:  b'{"id": "8EDD", "messagetype": "4", "timestamp": "07/14/2020
17:25:26", "enableSensors": "1,1,1,1,", "sensorValues": "77.85,100.00,100.00,100.00,",
"sensorType": "C1,C1,C1,C1"}'
Sensor Message Recieved
-----------------
Processing MQTT Sensor Message
-----------------
MoistureSensorStates
[{'id': '8EDD', 'sensorType': 'C1', 'sensorNumber': '1', 'sensorValue': '77.85', 'timestamp':
'2020-07-14 10:25:27'}, {'id': '8EDD', 'sensorType': 'C1', 'sensorNumber': '2', 'sensorValue':
'100.00', 'timestamp': '2020-07-14 10:25:27'}, {'id': '8EDD', 'sensorType': 'C1', 'sensorNumber':
'3', 'sensorValue': '100.00', 'timestamp': '2020-07-14 10:25:27'}, {'id': '8EDD', 'sensorType':
'C1', 'sensorNumber': '4', 'sensorValue': '100.00', 'timestamp': '2020-07-14 10:25:27'}]
-----------------
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
>>>>>>Valve Check<<<<<<
nextMoistureValveSensorCheck =  2020-07-14 10:15:00
nextMoistureValveSensorCheck =  2020-07-14 10:30:00
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
>>>>>>Valve Check<<<<<<
MQTT:  Sending PUBLISH (d0, q0, r0, m2), 'b'SGS/8EDD/Valves'', ... (113 bytes)
MQTT:  Received PUBLISH (d0, q0, r0, m0), 'SGS/8EDD', ...  (97 bytes)
Wireless MQTT Message received:  b'{"id": "8EDD", "messagetype": "1", "timestamp": "07/14/2020
17:27:05", "valvestate": "V10000000"}'
Valve Change Received
Timer Fired!  Next Fire= 2020-07-14 10:30:00
MQTT:  Received PUBLISH (d0, q0, r0, m0), 'SGS/8EDD', ...  (97 bytes)
Wireless MQTT Message received:  b'{"id": "8EDD", "messagetype": "1", "timestamp": "07/14/2020
17:27:35", "valvestate": "V00000000"}'
Valve Change Received
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
The time is: 2020-07-14 10:30:01.355894
>>>>>>Valve Check<<<<<<
MQTT:  Sending PUBLISH (d0, q0, r0, m3), 'b'SGS/8EDD/Valves'', ... (113 bytes)
MQTT:  Received PUBLISH (d0, q0, r0, m0), 'SGS/8EDD', ...  (97 bytes)
Wireless MQTT Message received:  b'{"id": "8EDD", "messagetype": "1", "timestamp": "07/14/2020
17:30:05", "valvestate": "V10000000"}'
Valve Change Received
Timer Fired!  Next Fire= 2020-07-14 10:45:00
nextMoistureValveSensorCheck =  2020-07-14 10:45:00
MQTT:  Received PUBLISH (d0, q0, r0, m0), 'SGS/8EDD', ...  (97 bytes)
```

```
Wireless MQTT Message received:  b'{"id": "8EDD", "messagetype": "1", "timestamp": "07/14/2020
17:30:35", "valvestate": "V00000000"}'
Valve Change Received
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
The time is: 2020-07-14 10:35:01.355834
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
MQTT:  Received PUBLISH (d0, q0, r0, m0), 'SGS/8EDD', ...  (175 bytes)
Wireless MQTT Message received:  b'{"id": "8EDD", "messagetype": "4", "timestamp": "07/14/2020
17:35:27", "enableSensors": "1,1,1,1,", "sensorValues": "77.85,100.00,100.00,100.00,",
"sensorType": "C1,C1,C1,C1"}'
Sensor Message Recieved
-----------------
Processing MQTT Sensor Message
-----------------
MoistureSensorStates
[{'id': '8EDD', 'sensorType': 'C1', 'sensorNumber': '1', 'sensorValue': '77.85', 'timestamp':
'2020-07-14 10:35:27'}, {'id': '8EDD', 'sensorType': 'C1', 'sensorNumber': '2', 'sensorValue':
'100.00', 'timestamp': '2020-07-14 10:35:27'}, {'id': '8EDD', 'sensorType': 'C1', 'sensorNumber':
'3', 'sensorValue': '100.00', 'timestamp': '2020-07-14 10:35:27'}, {'id': '8EDD', 'sensorType':
'C1', 'sensorNumber': '4', 'sensorValue': '100.00', 'timestamp': '2020-07-14 10:35:27'}]
-----------------
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
>>>>>>Valve Check<<<<<<
MQTT:  Sending PINGREQ
MQTT:  Received PINGRESP
The time is: 2020-07-14 10:40:01.355954
```

Disclaimer

SwitchDoc Labs, LLC takes no responsibility for any physical injuries and possession loss caused by those reasons which are not related to product quality, such as operating without following the operating manual and cautions, natural disasters or force majeure.
SwitchDoc Labs, LLC has compiled and published this manual which covers the latest product description and specification. The contents of this manual are subject to change without notice.