Implementing a Water Quality Sensoring System for Hydroponic and Aeroponic Agriculture

These are instructions on how to assemble a water quality sensor board using Atlas Scientific sensors, and a Raspberry Pi that will store data to a MongoDB database.

For this example, six Atlas Scientific sensors composed of probe and additional circuit will be used. For each sensor, a USB Serial (TTL) converter will be used to connect the Atlas Scientific sensor’s circuit to the Raspberry Pi. A USB hub will have to be used since the Raspberry Pi has only 2 USB ports.

**Parts needed:**

* Atlas Scientific sensors (Probe, Circuit and BNC connector):

-pH

-ORP (Oxidation-Reduction Potential)

-EC (Electrical Conductivity)

-Temperature

-Flow

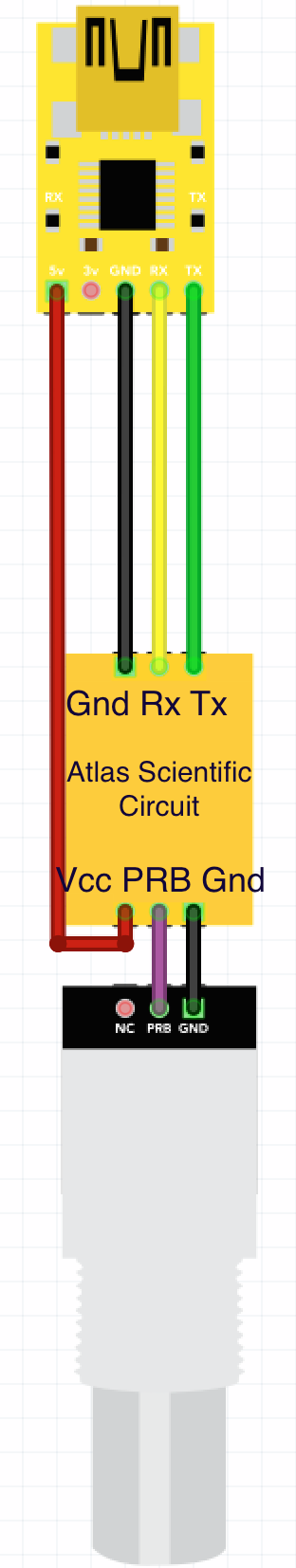
-DO (Dissolved Oxygen)

* 6 USB to Serial (TTL) converters

-A standard USB to Serial (RS232, DB9 connector) will not work, since the voltages are different than those used by TTL serial.

* 1 6-Port USB Hub
* Proto board and jumper cables

* 1 Raspberry Pi Rev. B (with Ethernet), SD card and power brick
* MongoDB server set up and running.

****

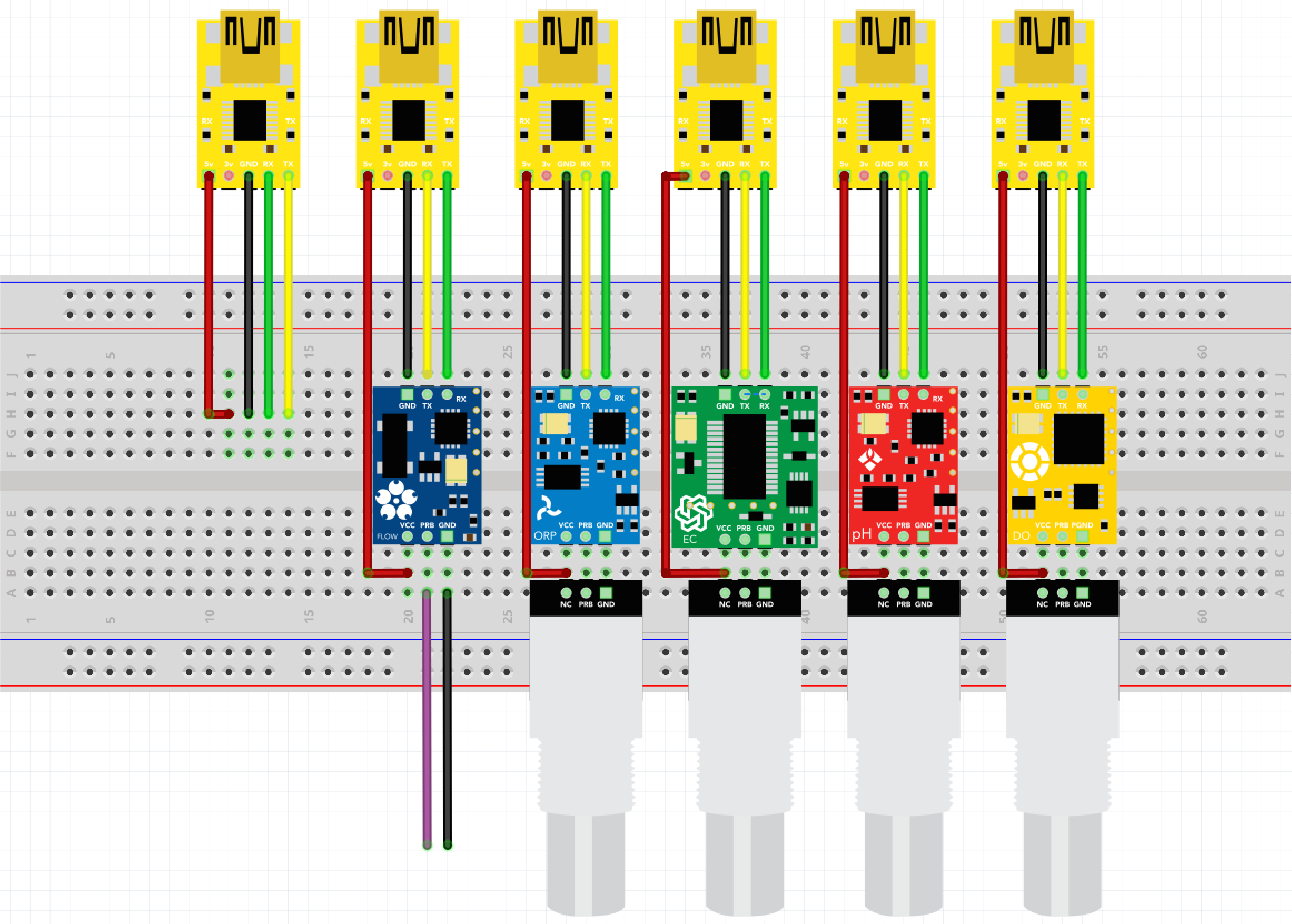
**Assembling Protoboard:**

For each sensor, a BNC connector, Atlas Scientific circuit and USB-Serial converter will be connected together as following:

The Atlas Scientific Circuit board operates from 3.3V to 5.5V; these voltages can be supplied by the USB-Serial converter, as in the diagram above. The Rx and Tx connections from the circuit are normally connected to their opposites on the USB-Serial converter, i.e. Rx on Tx, and Tx on Rx, However some USB-Serial converters are labeled differently, and these connections might have to be done Rx on Rx, and Tx on Tx. The BNC Connector, as per Atlas Scientific instructions, must be connected to the protoboard with the circuit, not with jumper cables.

All six sensors can be assembled on the protoboard like the following image:

To Flow meter



Temperature sensor connection

ORP EC pH DO

To Flow Meter

USB Serial Converters

Atlas Scientific sensor circuits

**Installing the Operating System to the Raspberry Pi**

Download the zipped Arch Linux image form the Raspberry Pi website:

<http://www.raspberrypi.org/downloads/>

Decompress the zip file to reveal the ‘.img’ file. To transfer it to an SD Card, use the following command on a Linux/Unix or Mac OS system:

dd bs=1M if=/path/to/archlinux.img of=/dev/sdcard\_path

Where *‘sdcard\_path’* is the path to the SD card. *(‘/dev/sdX’*, *‘/dev/mmcblkX’*, on Linux systems, *‘/dev/rdiskX’* on Mac OS systems)

After completing the process, remove and connect the SD card to the Raspberry Pi and plug it into a power supply. Arch Linux should boot up with a default username of ‘root’ and password ‘root’.

**Installing the required software**

To transfer the files from the demo folder, use the following command on a computer (not on the Raspberry Pi):

scp -r /path\_to\_demo\_folder/rpi/ user@ip\_address\_rpi:/root

Where “user” is the username on the Raspberry Pi (“root” is the default user for Arch Linux), “ip\_address” is the IP address or URL of the board, and ‘*path\_to\_demo\_folder’* is the path to the demo folder downloaded from GitHub, which contain the folder ‘*rpi’* inside, this will be the folder copied over to the Raspberry Pi. Now There’ll be a folder named ‘*rpi’* in the ‘*/root’* folderof the Raspberry Pi.

To configure the script (rpi\_service.py) for execution, open an SSH connection to the Raspberry Pi board, running the following command:

ssh user@ip\_address\_rpi

Proceed to install the required packages, *Python*, *PyMongo*, *PySerial*, and two additional packages, *screen* and *ntp*:

pacman –Sy  
pacman –S python2 python2-pyserial python2-pymongo screen ntp  
systemctl enable ntpd.service  
chmod +x rpi\_script.py rpi\_sh.sh  
systemctl reboot

From the additional packages, *screen* will be used to open a virtual terminal, run the script, and keep it running in background. *ntp* is the time synchronization service, used to keep the board’s date and time up to date.

The commands provided will, in order, update the package database, install the required and additional packages, and enable the time synchronization service, to maintain the date and time correct, finally the board will be rebooted, and the ssh connection will be closed; just open the ssh connection again after the board finished booting up.

**Connecting sensors to the Raspberry Pi**

Each USB-Serial converter must be connected to the USB hub, which in turn, will be connected to the Raspberry Pi. For the Atlas Scientific sensors, connect the probes to the BNC connectors on the protoboard.

**Configuring the software**

The script can now be configured, and data can start being collected. The following commands will be used to run the script:

cd /root/RPi (Where the files have been copied to)  
screen (A new virtual terminal will open, press  
 enter to continue)  
python2 rpi\_service.py

Since it's the first time the script is executed, it'll ask to create a configuration file in the folder ‘*/root/rpi/’* . This can be changed by altering the ‘*base\_path’* variable in the script. More information can be found in the documentation.

File not found, or nonexistent, press enter to create a new settings file (Enter)

Press Enter, and enter in the requested information:

Enter board name: (Will be used to create a collection at the specified  
 database, each board name must be unique)  
  
Enter database name: (Database being used to store sensor data)  
  
Enter server address or IP: (Server address. Can be an IP address or URL)  
  
Enter sensor reading frequency (seconds): (Interval in which the board will   
 send readings to the database)

The script will try to connect to the database, and will find the available serial ports. When done, the following message will appear:

Wait to use predefined sensors, press ctrl-c to manually enter sensors (will delete old sensors).

Since there are no sensors on file yet, press the *ctrl^c* combination to enter manual sensor configuration. The available serial ports will be shown:

Available serial ports: (Available ports will be shown, as the example below)  
 (0) /dev/ttyACM0  
 (1) /dev/ttyUSB0  
 (2) /dev/ttyUSB1  
 (3) /dev/ttyUSB2

Four settings will be asked for each sensor. Just enter the information (port number, sensor/measurements names, sensor/measurements units and time required for measurement):

Sensor #1  
Type a port number:  
Type the sensor's measurement name:   
Type the sensor's units  
Type the time this sensor takes to return a measurement (in milliseconds):

Sensor#2  
Type a port number:

The port number is the index of the list shown, so, in this example, ‘*/dev/ttyUSB0’* would be ‘*1’.* Typing 1 and pressing enter will move o to the next settings.

The sensor’s measurement name and units can range from a single name/unit or comma separated (CSV) names and units, to accommodate for sensors that report more than one measurement, e.g.:

Measurement name: ‘Temperature’

Units: ‘C’

Measurement name: ‘Temperature,Humidity,CO2’

Units: ‘C,RH,ppm’

The last entry is the amount of time in milliseconds the sensor takes to return a valid and complete measurement. This information is usually provided on the sensor’s datasheet.

After all sensors have been entered, just press enter on the “Type a port number:” dialogue, this will save the sensor settings to se config file, and continue executing the script.

The continuous reading loop will start, and JSON formatted strings will be sent to the database, and shown, like the example below:

Reading Started:

1  
{"Temperature":{"value": 21.98, "units": "C"}, "Humidity":{"value": 38.4, "units": "RH"}, "CO2":{"value": 469, "units": "ppm"}}

2  
{"Temperature":{"value": 21.86, "units": "C"}, "Humidity":{"value": 38.3, "units": "RH"}, "CO2":{"value": 467, "units": "ppm"}}

**Running script on the background**

To run the script on the background, we can detach the virtual terminal that’s running the script. Pressing *‘ctrl^a’* will enable screen to take commands, then pressing ‘d’ will detach the screen and return to the original terminal. The script will be now running on background. This can be checked by running the following command:

ps –aux | grep py

This will show all the running processes with ‘py’ in its name.

To reattach the virtual terminal, just run the following command:

screen -r

**Using script as a daemon**

To run the script as a daemon, that is, so it can be executed automatically upon boot, while on the ‘rpi’ folder, use the following commands:

cp rpi\_service.service /etc/systemd/system  
systemctl enable rpi\_service.service

The next time the Raspberry Pi boots up, the script will be started, and data will be reported. To check the script is running, use the ‘ps’ command shown above.

To start and stop the service, use:

systemctl start rpi\_service.service   
systemctl stop rpi\_service.service

**Log**

A logging system is used, and is useful for checking the status of scripts running on background. The log file is located on the *‘base\_path’* folder, in this example*, ‘/root/rpi’*. The log filename is *‘log.log’.* To view the log, use the following command:

cat /root/rpi/log.log

**Calibrating the sensors**

To calibrate the sensors, the “sensor\_terminal.py” script can be used. This script provides access to the sensor in a terminal like manner, where commands can be entered (Carriage Return is added automatically) and strings sent from the sensor can be viewed. This terminal can be used to send calibration commands to the sensor. Follow the sensor’s instructions on how to execute the calibration.

No scripts using the sensor in question should be running.

Run the script:

python2 sensor\_terminal.py

Select the sensors to be accessed. When done, press enter on the “Type a port number: ” dialogue.

Select a sensor from the list, and enter the necessary commands. When done, press *ctrl^c* to go back to the sensor list. Another sensor can be chosen, or pressing enter will exit the script.