BlockOasis

IOT SYSTEM FOR WATER QUALITY MANAGEMENT IN NILE RIVER - REPORT

Contents

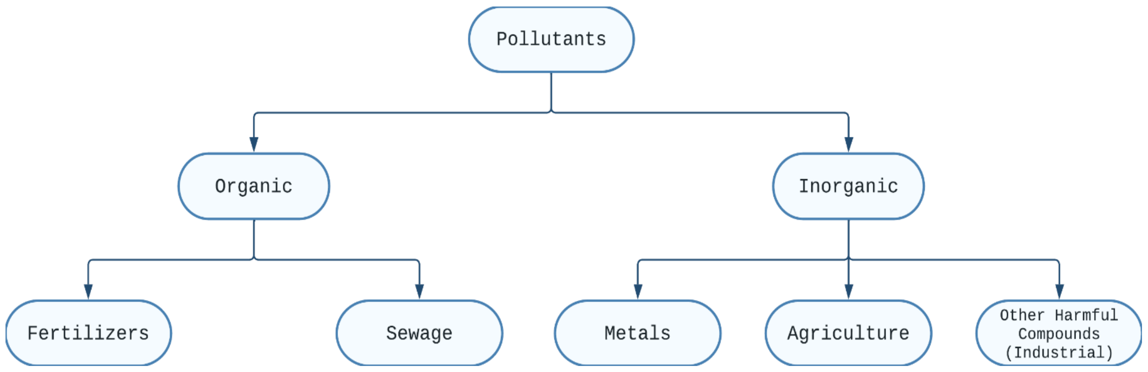
Region Of Interest – Nile:

- The river Nile is the primary source of freshwater for drinking, irrigation, and industrial purposes in the whole African continent. Thus, the water quality in this river concerns the health of inhabitants.

- The water quality variations were mainly at big cities and related to inorganic nutrients and heavy metals, where the sites affected by intensive load of urban, agricultural and industrial wastewater showed serious deterioration of water quality.

- Water quality of Nile becomes important when it comes Agriculture of countries like Egypt where agriculture is possible through Nile River only and hence directly affecting the economic development of the country.

Pollutants & their broad Classification (mainly rivers):



1. **Industrial:** The industrial sector is an important user of water and a contributor to pollution. Egypt is the most industrialized country of the Nile and faces rapidly increasing deterioration of its surface and groundwater due to increasing discharge of polluted domestic and industrial effluents. The food processing industry is responsible for more than 50% of biological oxygen demand (BOD) while the chemical industry is responsible for more

than 60% of heavy metal discharges. The high BOD load from food processing is attributed to ten sugar factories between Aswan and Cairo.

2. **Heavy Metals (Inorganic):** During its transit to Egypt, the Nile receives numerous non-point and point source discharges. Lead and cadmium were detected in the raw Nile and in the finished drinking water of four treatment plants in greater Cairo between 1993 and 1994. The levels were 29.6 µg l−1 for lead and 4.15 µg l−1 for cadmium. The problem of wastewater disposal has tremendously increased since the expansion of industrialization in the countries residing beside the Nile. The discharge of large quantities of wastewater without treatment affects the use of river water for both drinking and irrigation purposes and has a damaging effect on aquatic life. Wastewater from electroplating industries in the Helwan area (south Cairo) is characterized by a high content of Cr, Zn, Cu, Fe, and Mn. It was determined that the levels of Cu, Zn, Cd, and Pb in water, sediment, and aquatic plants from the Nile and its branches at sites characterized by heavy industrialization and dense population. The results revealed that concentrations in sediments were, in most cases, much higher than in plants. Metal content in plants downstream of wastewater discharge points was usually higher than in plants upstream of these points. In Lake Victoria too, concentrations of Pb and Cd were high at most of the sites of the lake. This is alarming since these metals are among the most hazardous to humans and the environment. Mercury contamination from the processing of gold ore on the southern shores is currently considered among the most emergent phenomena of chemical contamination.
3. **Agriculture:** Drainage water seeping from agriculture fields is a non-point source of pollution. Such non-point sources, however, collect and concentrate in agricultural drains to become point sources to the river, the northern lakes, and irrigation canals. Non-point sources may also influence groundwater quality. Major pollutants in agricultural drains are salts, nutrients, pesticide residues, pathogens, and toxic organic and inorganic pollutants. In Egypt only, the Nile from Aswan to the delta barrage receives wastewater from 124-point sources, of which 67 are agricultural drains.

4. **Organic Pollution:** Several studies assess Nile pollution by pesticides and heavy metals and pollution by organic chemicals in delta lakes and sediments in Egypt. Drainage water was highly polluted and contained more pesticide residues than canal waters. It was confirmed the presence of 16 organochlorine pesticides in most of the water samples; the percent positive samples followed the order endrin > total BHC > total DDT (1,1,1-trichloro-2,2-bis (4- chlorophenyl) ethane) > endosulfan > heptachlor epoxide > heptachlor. These pesticide residues represent different chemical classes.
5. **Domestic Sewage:** The population increase in countries along the Nile River, especially Egypt, results in more wastewater and this requires governments to increase the number of

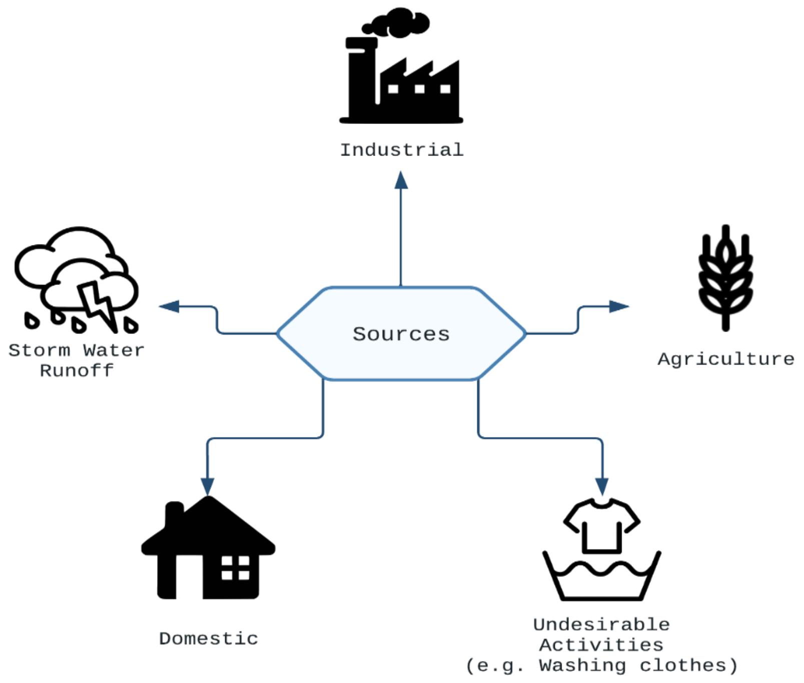
water treatment plants. The constituents of concern in domestic and municipal wastewater are pathogens, parasites, nutrients, oxygen-demanding compounds, and suspended solids. Toxic substances such as heavy metals and organic micro-pollutants also occur, due to the mixing of domestic with industrial and commercial activities. The bulk of treated and untreated domestic wastewater is discharged into agricultural drains. All drains of Upper Egypt flow back to the Nile and many irrigations canals

6. **Oil Pollution:** The Nile is used for water supply and to transport different types of cargo, including oil and oil products. In addition, many factories and oil distribution centers are located along its shores. Oil pollution may also derive from barges, tankers, and boats on rivers or canals, from industrial wastes, metallurgical industries, engineering work, garages, or any places using lubricating or fuel oils. Process and cooling water are pumped from the Nile and returned to it after use. The toxicity of oils is caused by unsaturated hydrocarbons, naphthenic acids, and other compounds containing aromatic groups and nitrogen.
7. **Miscellaneous:**
    a. **Cyanotoxins -** Cyanobacteria often show extraordinary development (blooms) in eutrophic and hypertrophic lakes and rivers. Increased eutrophication due to agricultural, municipal, and industrial runoff contributes much to the growth of such species, many of which are toxin-producers. Intoxicated blooms degrade drinking or recreational water and increase the risk of poisoning to animals and humans. In spite of discharging of different kinds of wastes into the river Nile, the water quality does not yet exhibit pollution levels high enough to create health risks except locally, where the presence of Coli bacteria indicates unsafe levels for direct use of such water in irrigation and fisheries (APRP, 2002). The high dilution factor and the high self-assimilation capacity of Nile water makes the pollution of the river Nile is only worrisome near to cities and industrial discharges.
    b. **Radioactive Pollutants -** Mining and processing of ores, Use in research, agriculture, medical and industrial activities, such as I131, P32, Co60, Ca45, S35, C14, etc. Radioactive discharge from nuclear power plants and nuclear reactors, e.g., Sr90, Cesium Cs137, Plutonium Pu248. Uses and testing of nuclear weapons. These isotopes are toxic to life forms; they accumulate in the bones and teeth and can cause serious disorders.
    c. **Thermal Pollutants -** Considerable thermal pollution results due to discharge of hot water from thermal power plants, nuclear power plants, and industries where water is used as coolant. As a result of hot water discharge, the temperature of the water body increases, which reduces the DO (dissolved oxygen) content of the water. This alters the spectrum of organisms, which can adapt to live at that

temperature and DO level. The discharge of hot water leads to the thermal stratification in the water body, where hot water will remain on the top.

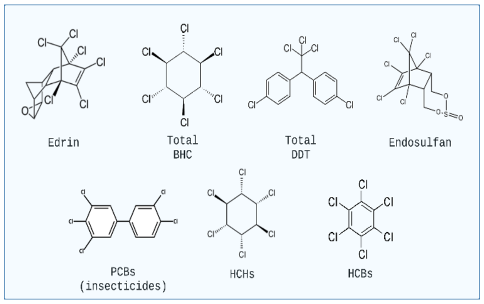
d. **Suspended solids and sediments -** Presence of suspended solids can block the sunlight penetration in the water, which is required for the photosynthesis by bottom vegetation. – Deposition of the solids in the quiescent stretches of the stream or ocean bottom can impair normal aquatic life and affect the diversity of the aquatic ecosystem.

Top 20% Pollutants cause 80% Pollution:



1. Industrial Waste (Metals):
   - Lead
   - Cadmium
   - Chromium
   - Zinc
   - Fe
   - Mn
   - Hg

2. Agricultural Waste (Nitrates, Phosphates and Organochlorines):
   - Endrin
   - Total BHC
   - Total DDT
   - Endosulfan
   - PCBs(insecticides)
   - HCHs
   - HCBs



3. Domestic Sewage:
   - Cyanotoxins
   - Eutrophication

Pollutants to be measured:

1. pH
2. Turbidity
3. Temperature
4. Level of water (abrupt changes will be alerted)
5. Dissolved Oxygen
6. TDS
7. Metals: - Cu and Fe.
8. Pesticides (Nitrates)
9. Insecticides (Organo-Chlorine)
10. Conductivity
11. ORP (Oxidation-reduction potential)
12. Residual Chlorine
13. Chlorophyll

Scope of Measuring the heavy metals: **- Heavy metals (Pb, Cd, Cr)** and Phosphate ions are measured with biosensors which are not calibrated with any microcontroller by default (calibration will stretch the timeline unnecessarily) and they are out of the budget of the project.

Microcontrollers:

Waspmote:

Waspmote hardware architecture has been specially designed to work with extremely low consumption. Digital switches allow us to turn on and off any of the sensor interfaces as well as the radio modules. Three different sleep modes make Waspmote the lowest-consumption IoT platform in the market (7 µA).

General Characteristics:

- Microcontroller: ATmega1281
- Frequency: 14.74 MHz
- SRAM: 8 kB
- EEPROM: 4 kB
- FLASH: 128 kB
- SD card: 16 GB
- Weight: 20 g
- Dimensions: 73.5 x 51 x 13 mm
- Temperature range: [-30 ºC, +70 ºC] *
- Clock: RTC (32 kHz)

Temporary extreme temperatures are supported. Regular recommended usage: -20, +60ºC

Power consumption:

- On: 17 mA
- Sleep: 30 µA
- Deep Sleep: 33 µA
- Hibernate: 7 µA

Input / Output:

- 7 analog inputs, 8 digital I/O
- 2 UARTs, 1 I2C, 1 SPI, 1 USB

Sensors embedded on board:

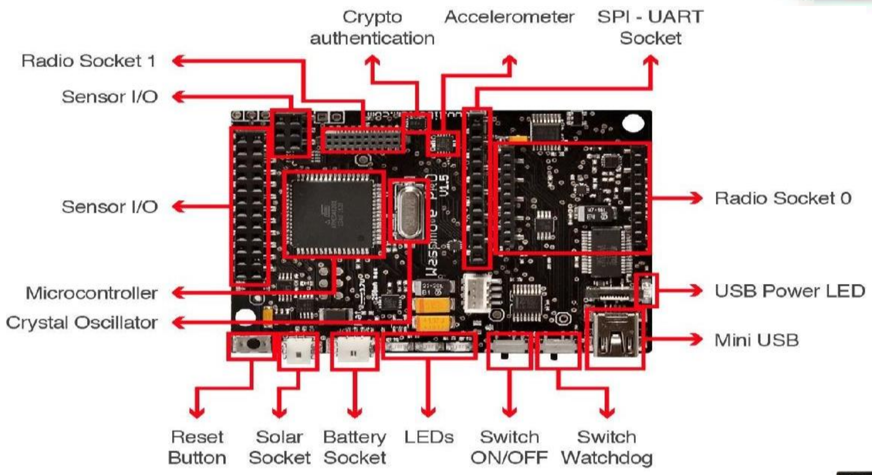
- Accelerometer: ±2g/±4g/±8g
- Low power: 0.5 / 1 / 2 / 5 / 10 Hz
- Normal mode: 50 / 100 / 400 / 1000 Hz

Electrical characteristics:

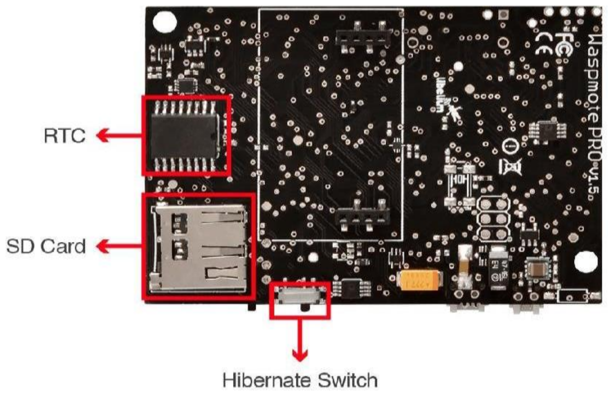
- Battery voltage: 3.3-4.2 V
- USB charging: 5 V – 100 mA
- Solar panel load: 6-12 V – 300 mA

Arduino UNO:

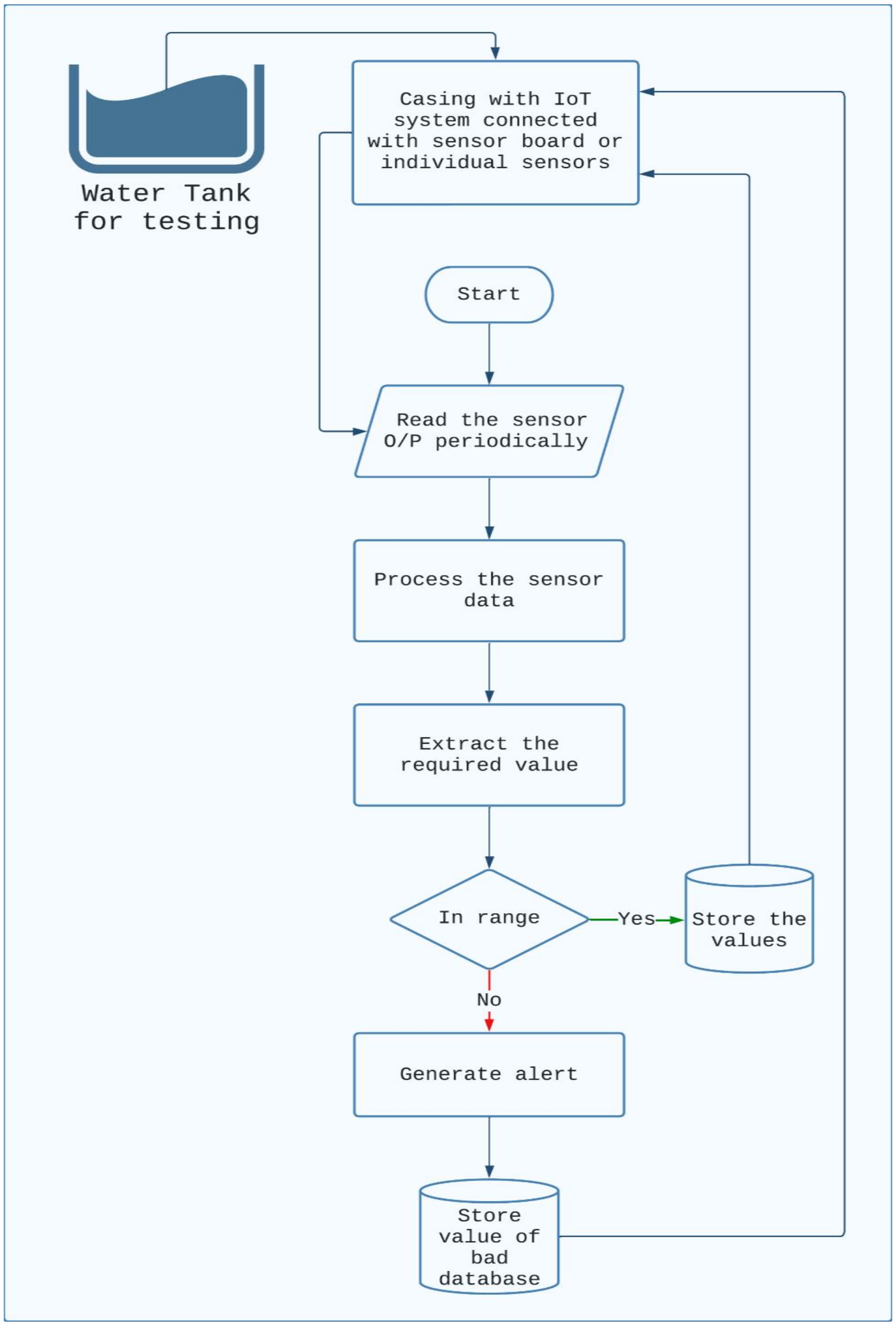
Arduino UNO is also used in the system separately for sensing water levels in the river and flow rate in the domestic pipes in homes.



Waspmote



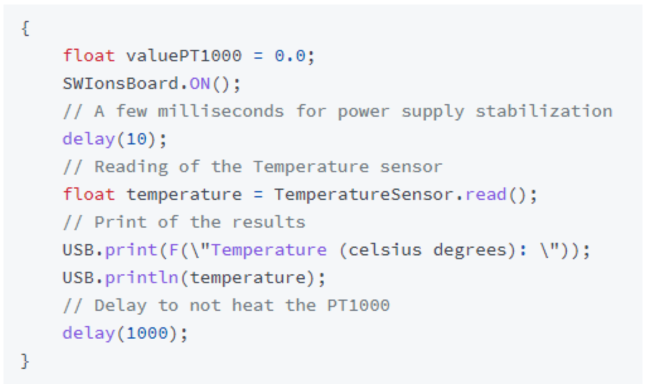
IoT System of Devices Flowchart:


Water Tank for testing
Casing with IoT system connected with sensor board or individual sensors
Start
Read the sensor O/P periodically
Process the sensor data
Extract the required value
In range
Yes
Store the values
No
Generate alert
Store value of bad database

Calibration of Sensors and Microcontroller Programming:

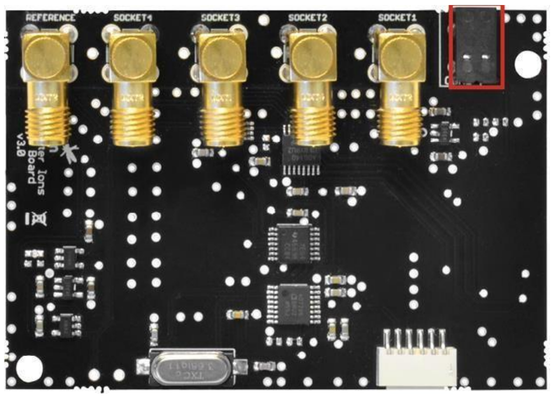
Temperature Sensor (Pt-1000):

A resistive sensor Pt-1000 powered by 5V, whose conductivity varies as a function of temperature is used. The Smart Water Ions board has been endowed with an instrumentation amplifier which allows to read the sensor placed in a voltage divider configuration along with one precision 1 kilo-ohm resistor, which leads to an operation range between 0 ºC and 100 ºC approximately. The whole reading process, from the voltage acquisition at the analog-to-digital converter to the conversion from the volts into Celsius degree, is performed by the readTemperature() function.



```
{
    float valuePT1000 = 0.0;
    SWIonsBoard.ON();
    // A few milliseconds for power supply stabilization
    delay(10);
    // Reading of the Temperature sensor
    float temperature = TemperatureSensor.read();
    // Print of the results
    USB.print(F(\"Temperature (celsius degrees): \"));
    USB.println(temperature);
    // Delay to not heat the PT1000
    delay(1000);
}
```

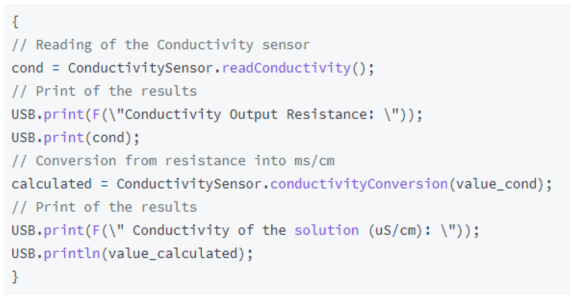
Link for complete example code of temperature sensor: https://development.libelium.com/waspmote/swi-01-temperature-sensor-reading



Socket on Smart Water Ion board for Temperature Sensor

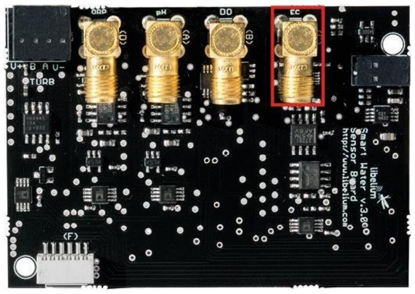
Conductivity Sensor:

The conductivity sensor is a two-pole cell whose resistance varies in function of the conductivity of the liquid it is immersed in. That conductivity will be proportional to the conductance of the sensor multiplied by the constant cell, in the case of the Libelium sensor around 1cm^{-1}, leading to a value in Siemens per centimetre (S/cm). An AC current circuit is installed to power the conductivity sensor. The `readConductivity()` function will return the resistance of the sensor in ohms. In order to convert this value into a useful conductivity unit function `conductivityConversion()` will have to be



```
{
// Reading of the Conductivity sensor
cond = ConductivitySensor.readConductivity();
// Print of the results
USB.print(F(\"Conductivity Output Resistance: \"));
USB.print(cond);
// Conversion from resistance into ms/cm
calculated = ConductivitySensor.conductivityConversion(value_cond);
// Print of the results
USB.print(F(\" Conductivity of the solution (uS/cm): \"));
USB.println(value_calculated);
}
```

Link for complete example code of conductivity sensor: [*https://development.libelium.com/waspmote/sw-05-conductivity-sensor-reading*](https://development.libelium.com/waspmote/sw-05-conductivity-sensor-reading)



Socket on Smart Water board for Conductivity Sensor

invoked with the calibration parameters of the sensor.

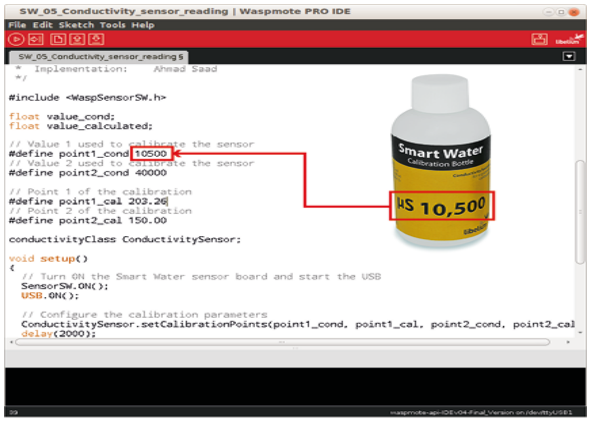
Calibration Process - There are three different Calibration kits for Conductivity: K=0.1, K=1; K=10 depending on the salinity of the water we want to measure. Each calibration kit comes with 2 solutions:

- K=0.1 - around 220 µS/cm and around 3000 µS/cm
- K=1 - around 10500 µS/cm and around 40000 µS/cm
- K=10 - around 62000 µS/cm and around 90000 µS/cm

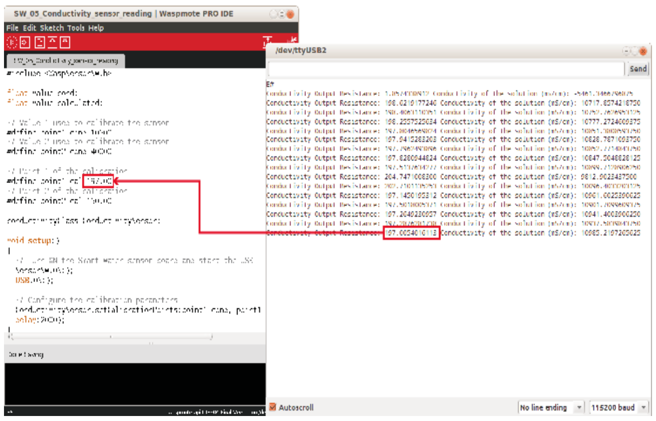
1. Turn on the Waspmote with the Smart Water sensor board and the conductivity sensor connected.
2. Upload the example "Conductivity sensor Reading for Smart Water" to the Waspmote board and make sure of receiving the data in the serial monitor.
3. Pour the conductivity solutions in two beakers.
4. Introduce the conductivity probe in the first solution and wait for a stable output. Make sure that the sensor is completely immersed in the solution and that it is not close to the beaker wall, which may affect the field between the electrodes and disturb the measurement. Once the output is steady, annotate the value of the **Conductivity Output Resistance** obtained. It

is important to give time to the output to get stable, especially the first time we use a sensor. This will take several minutes.

5. After getting the sensor from the first solution, carefully rinse it (do not dry the sensor, since the platinum black layer of the electrodes could be damaged) and repeat the process explained in step 3 with the second solution.
6. Introduce the values noted and the conductivity of the calibration solutions in your code, as shown in the next images.
7. The function `setCalibrationPoints()` is used to configure the calibration parameters.
8. Upload the code again with the new calibration values obtained from the calibration process.


SW_05_Conductivity_sensor_reading | Waspmote PRO IDE
File Edit Sketch Tools Help
SW_05_Conductivity_sensor_reading §
Smart Water
Calibration Bottle
µS 10,500
waspmote-api-IDEv04-Final_Version on /dev/ttyUSB1

```
 *  Implementation:    Ahmad Saad
 */

#include <WaspSensorSW.h>

float value_cond;
float value_calculated;

// Value 1 used to calibrate the sensor
#define point1_cond 10500
// Value 2 used to calibrate the sensor
#define point2_cond 40000

// Point 1 of the calibration
#define point1_cal 203.26
// Point 2 of the calibration
#define point2_cal 150.00

conductivityClass ConductivitySensor;

void setup()
{
  // Turn ON the Smart Water sensor board and start the USB
  SensorSW.ON();
  USB.ON();

  // Configure the calibration parameters
  ConductivitySensor.setCalibrationPoints(point1_cond, point1_cal, point2_cond, point2_cal
  delay(2000);
```


SW_05_Conductivity_sensor_reading | Waspmote PRO IDE
File Edit Sketch Tools Help
/dev/ttyUSB2
Send
Autoscroll
No line ending
115200 baud

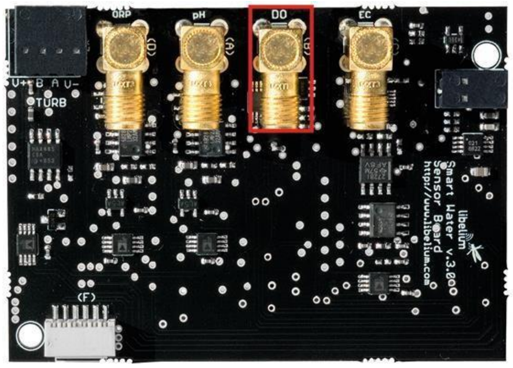
```
[illegible]
Conductivity Output Resistance: 198.6219177246 Conductivity of the solution (mS/cm): 10717.8574218750
[illegible]
Conductivity Output Resistance: 198.2557525634 Conductivity of the solution (mS/cm): 10777.2724609375
[illegible]
Conductivity Output Resistance: 197.9415283203 Conductivity of the solution (mS/cm): 10828.7871093750
[illegible]
Conductivity Output Resistance: 197.8280944824 Conductivity of the solution (mS/cm): 10847.5048828125
[illegible]
Conductivity Output Resistance: 204.7471008300 Conductivity of the solution (mS/cm): 9812.9023437500
[illegible]
Conductivity Output Resistance: 197.1450195312 Conductivity of the solution (mS/cm): 10961.6025390625
[illegible]
Conductivity Output Resistance: 197.2640230957 Conductivity of the solution (mS/cm): 10941.4003906250
[illegible]
Conductivity Output Resistance: 197.0054016113 Conductivity of the solution (mS/cm): 10985.2197265625
```

Dissolved Oxygen Sensor:

The galvanic cell provides an output voltage proportional to the concentration of dissolved oxygen in the solution under measurement without the need for a supply voltage. The value returned by the readDO() function for this sensor is expressed in volts. For a conversion into a percentage of oxygen saturation function DOConversion() will have to be used, introducing the calibration value in volt as an input.

```
{
// Reading of the DO sensor
value_do = DOSensor.readDO();
// Print of the results
USB.print(F(\"DO Output Voltage: \"));
USB.print(value_do);
// Conversion from volts into dissolved oxygen percentage
value_calculated = DOSensor.DOConversion(value_do);
// Print of the results
USB.print(F(\" DO Percentage: \"));
USB.println(value_calculated);
}
```

Link for complete example code of dissolved oxygen sensor:
https://development.libelium.com/waspmote/sw-04-dissolved-oxygen-sensor-reading



Socket on Smart Water board for Dissolved Oxygen Sensor

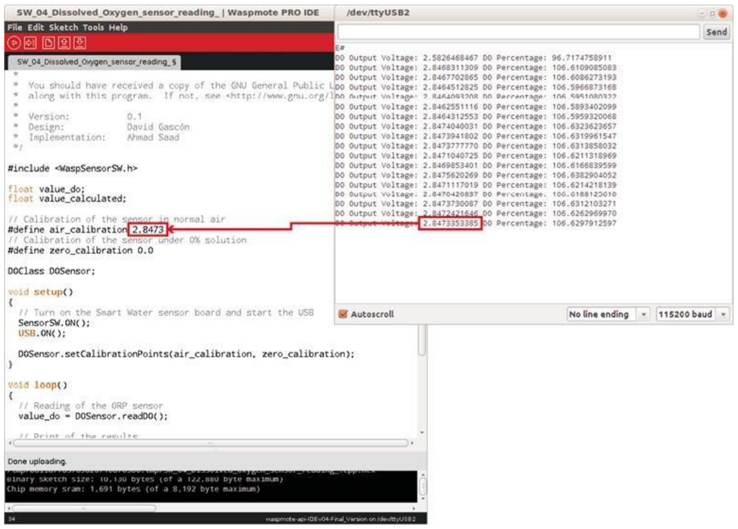
Calibration Process - The calibration process for the dissolved oxygen sensor can be divided into two parts. The first one corresponds to a single point calibration, which should be enough for most applications. In the second one, the calibration is extended to a second point, which leads to a more accurate value, although it implies a high leap in complexity.

First Point:

1. Turn on the Waspmote with the Smart Water sensor board and the dissolved oxygen sensor connected.
2. Upload the code "[Dissolved Oxygen Sensor Reading]{.underline}" and make sure the data from the sensor is being received properly in the serial monitor.
3. To get a saturated value of the sensor, just clean the sensor with distilled or de-ionized water, carefully rinse it and dry it with a paper cloth. Once in air, wait for the output stabilization. Once the measured value is steady, write it down. If the sensor has been deployed in a placement with difficult access, instead of getting it out it is possible to bubble air in the fluid until the sensor reaches saturation, though it is a less reliable method. It is really important to give time to the output to get stable, especially the first time we use a sensor. This will take several minutes.
4. This value corresponds to a saturated output (100% of dissolved oxygen). In case of a single point calibration, introduce this value in the code as shown in the image below, while

introducing a 0 for ZERO_VALUE, or add it to the conversion in the software at reception. Otherwise, go on with the second point procedure.

5. Upload the code again with the new calibration values obtained from the calibration process.


SW_04_Dissolved_Oxygen_sensor_reading_ | Waspmote PRO IDE
File Edit Sketch Tools Help
SW_04_Dissolved_Oxygen_sensor_reading_§
/dev/ttyUSB2
Send
Autoscroll
No line ending
115200 baud
Done uploading.

```
 *
 *  You should have received a copy of the GNU General Public L
 *  along with this program.  If not, see <http://www.gnu.org/l
 *
 *  Version:           0.1
 *  Design:            David Gascón
 *  Implementation:    Ahmad Saad
 */

#include <WaspSensorSW.h>

float value_do;
float value_calculated;

// Calibration of the sensor in normal air
#define air_calibration 2.8473
// Calibration of the sensor under 0% solution
#define zero_calibration 0.0

DOClass DOSensor;

void setup()
{
  // Turn on the Smart Water sensor board and start the USB
  SensorSW.ON();
  USB.ON();

  DOSensor.setCalibrationPoints(air_calibration, zero_calibration);
}

void loop()
{
  // Reading of the ORP sensor
  value_do = DOSensor.readDO();

  // Print of the results
```

```
Binary sketch size: 10.130 bytes (of a 122.880 byte maximum)
Chip memory sram: 1,691 bytes (of a 8.192 byte maximum)
```

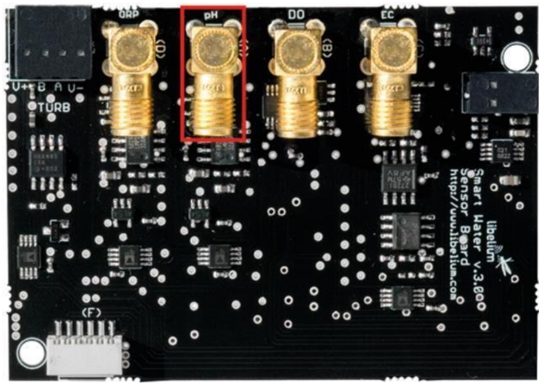
```
E#
DO Output Voltage: 2.5826468467 DO Percentage: 96.7174758911
DO Output Voltage: 2.8468311309 DO Percentage: 106.6109085083
DO Output Voltage: 2.8467702865 DO Percentage: 106.6086273193
DO Output Voltage: 2.8464512825 DO Percentage: 106.5966873168
DO Output Voltage: 2.8464093208 DO Percentage: 106.5951080322
DO Output Voltage: 2.8462551116 DO Percentage: 106.5893402099
DO Output Voltage: 2.8464312553 DO Percentage: 106.5959320068
DO Output Voltage: 2.8474040031 DO Percentage: 106.6323623657
DO Output Voltage: 2.8473941802 DO Percentage: 106.6319961547
DO Output Voltage: 2.8473777770 DO Percentage: 106.6313858032
DO Output Voltage: 2.8471040725 DO Percentage: 106.6211318969
DO Output Voltage: 2.8469853401 DO Percentage: 106.6166839599
DO Output Voltage: 2.8475620269 DO Percentage: 106.6382904052
DO Output Voltage: 2.8471117019 DO Percentage: 106.6214218139
DO Output Voltage: 2.8470420837 DO Percentage: 106.6188125610
DO Output Voltage: 2.8473730087 DO Percentage: 106.6312103271
DO Output Voltage: 2.8472421646 DO Percentage: 106.6262969970
DO Output Voltage: 2.8473353385 DO Percentage: 106.6297912597
```

pH Sensor:

It is a combination electrode that provides a voltage proportional to the pH of the solution. Corresponding the pH 7, we have the voltage reference of 2.048 V of the circuit, with an uncertainty of ±0.25 pH. it is necessary both to carry out a calibration and to compensate the output of the sensor for the temperature variation from that of the calibration moment. Once the sensor has been calibrated, these two tasks are carried out in the `pHConversion()` function.

```
{
// Read the pH sensor
value_pH = pHSensor.readpH();
// Read the temperature sensor
value_temp = temperatureSensor.readTemperature();
// Print the output values
USB.print(F(\"pH value: \"));
USB.print(value_pH);
USB.print(F(\"volts | \"));
USB.print(F(\" temperature: \"));
USB.print(value_temp);
USB.print(F(\"degrees | \"));
// Convert the value read with the information obtained in calibration
value_pH_calculated = pHSensor.pHConversion(value_pH,value_temp);
USB.print(F(\" pH Estimated: \"));
USB.println(value_pH_calculated);
}
```

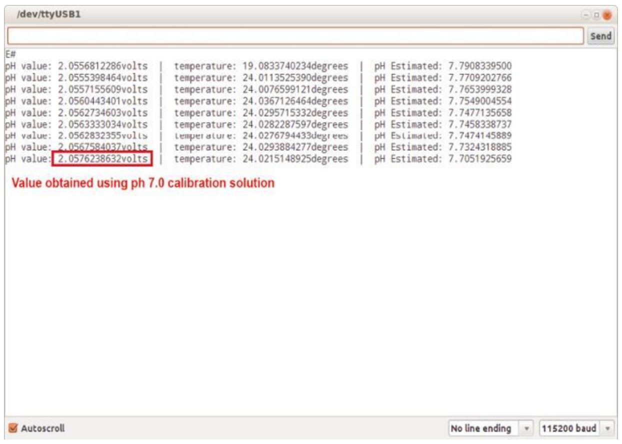
Link for complete example code of pH sensor:
https://development.libelium.com/waspmote/sw-01-ph-sensor-reading



Socket on Smart Water board for pH Sensor

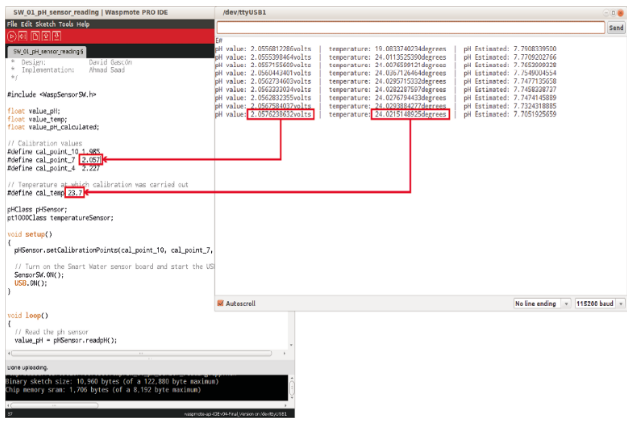
Calibration Process - Periodic calibration is highly recommended for the pH sensors. It will also be required to receive temperature compensation to update the sensitivity of the sensor to the actual conditions. For calibration, we use three pH buffer solutions, one of 7.0 pH, one of 4.0 pH and one of 10.0 pH. For a proper calibration all the buffers must be at the same temperature, being a temperature the closest possible to that of operation.

1. Turn on the Waspmote with the Smart Water sensor board and the pH sensor and the Pt-1000 connected.
2. Upload the code "[pH Sensor Reading]{.underline}" and make sure the data is being correctly received through the USB or another communication module.
3. Pour the solutions in three beakers. The 4.0 pH solution is red, the 7.0 pH solution yellow and the 10.0 pH solution blue. It is recommended that the solutions are at the temperature that will be found at the installation environment.
4. Introduce the pH sensor and the Pt-1000 in the 7.0 pH buffer solution and wait for a stable measurement, which may take a few minutes. Make sure the sensors are completely immersed in the solution. When there is a stable output for the sensors, annotate the value in volts obtained. It is really important to give time to the output to get stable, especially the first time we use a sensor. This will take several minutes.
5. Get the sensor out of the solution and rinse it gently, preferably with distilled or de-ionized water, and introduce it in the 4.0 pH solution, which will cause an increase in the output voltage, along with the Pt-1000 sensor to check that all the solutions are at the same temperature. Again, wait for the stabilization of the output values and write them down.
6. Repeat step 3 with the 10.0 pH solution, which should make the sensor output voltage fall below that for the 7.0 pH solution. Under 25 ºC the outputs expected for these solutions are 2.048 V for 7pH, 2,227 mV for 4.0 pH and 1.868 mV for 10.0 pH), with the possibility of finding a difference of a few tenths of millivolts for each value and a change in the sensitivity owing to the difference of temperature.
7. Significantly different values after the exposure of the sensor to the solutions may be caused by a bubble in the sensitive bulb, especially if it is the first calibration after shipment. Shaking the sensor downward like a clinical thermometer will remove them, solving the problem.
8. Introduce the calibration values in the measurement code as shown in the images below.
9. Upload the code again with the new calibration values obtained from the calibration process.



```
E#
pH value: 2.0556812286volts  |  temperature: 19.0833740234degrees  |  pH Estimated: 7.7908339500
pH value: 2.0555398464volts  |  temperature: 24.0113525390degrees  |  pH Estimated: 7.7709202766
pH value: 2.0557155609volts  |  temperature: 24.0076599121degrees  |  pH Estimated: 7.7653999328
pH value: 2.0560443401volts  |  temperature: 24.0367126464degrees  |  pH Estimated: 7.7549004554
pH value: 2.0562734603volts  |  temperature: 24.0295715332degrees  |  pH Estimated: 7.7477135658
pH value: 2.0563333034volts  |  temperature: 24.0282287597degrees  |  pH Estimated: 7.7458338737
pH value: 2.0562832355volts  |  temperature: 24.0276794433degrees  |  pH Estimated: 7.7474145889
pH value: 2.0567584037volts  |  temperature: 24.0293884277degrees  |  pH Estimated: 7.7324318885
pH value: 2.0576238632volts  |  temperature: 24.0215148925degrees  |  pH Estimated: 7.7051925659
```

Value obtained using ph 7.0 calibration solution



SW_01_pH_sensor_reading | Waspmote PRO IDE

```
 *  Design:          David Gascón
 *  Implementation:  Ahmad Saad
 */

#include <WaspSensorSW.h>

float value_pH;
float value_temp;
float value_pH_calculated;

// Calibration values
#define cal_point_10 1.985
#define cal_point_7  2.057
#define cal_point_4  2.227

// Temperature at which calibration was carried out
#define cal_temp 23.7

pHClass pHSensor;
pt1000Class temperatureSensor;

void setup()
{
  pHSensor.setCalibrationPoints(cal_point_10, cal_point_7,

  // Turn on the Smart Water sensor board and start the US
  SensorSW.ON();
  USB.ON();
}


void loop()
{
  // Read the ph sensor
  value_pH = pHSensor.readpH();
```

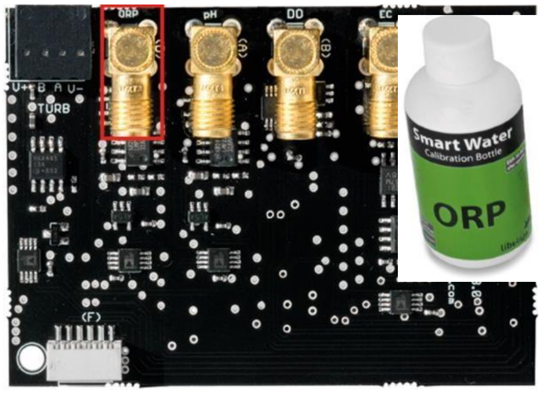
```
Done uploading.
Binary sketch size: 10,960 bytes (of a 122,880 byte maximum)
Chip memory sram: 1,706 bytes (of a 8,192 byte maximum)
```

Oxidation-reduction potential Sensor:

The ORP probe is a combination electrode whose output voltage is equivalent to the potential of the solution. The output of the circuitry to which it is connected is directly read from the analog-to-digital converter of the Smart Water sensor board, being the 2.048 V reference subtracted to obtain the actual oxidation-reduction potential in volts.

```
{
// Reading of the ORP sensor
value_orp = ORPSensor.readORP();
// Apply the calibration offset
value_calculated = value_orp - calibration_offset;
// Print of the results
USB.print(F(\" ORP Estimated: \"));
USB.print(value_calculated);
USB.println(F(\" volts\"));
}
```

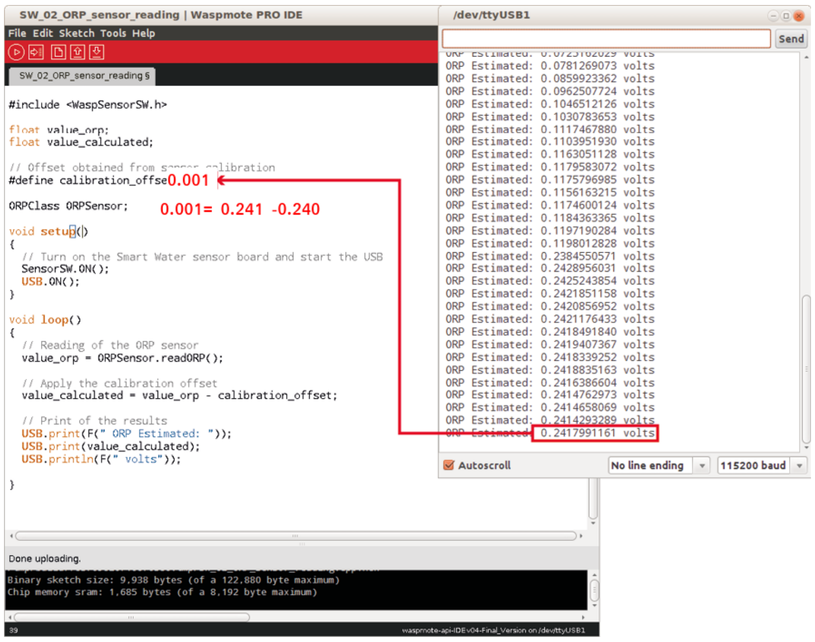
Link for complete example code of oxidation reduction potential sensor:
https://development.libelium.com/waspmote/sw-02-orp-sensor-reading



Socket on Smart Water board for ORP Sensor

Calibration Process - Since the sensor output is a straightforward voltage directly measured by the Waspmote's analog-to-digital converter there is not a conversion function. The calibration process will consist of a verification of the proper operation of the sensor with an ORP calibration standard solution.

1. Turn on the Waspmote with the Smart Water sensor board and the ORP sensor connected.
2. Upload the code "ORP Sensor Reading" and make sure that the data from the sensor is being received through the USB or another communication module.
3. Pour the calibration solution in a beaker. Libelium provides a standard solution of 240 mV at 25 ºC.
4. Rinse the sensor with distilled or de-ionized water and softly dry it with filter paper.
5. Introduce the sensor into the calibration solution, making sure it stays completely immersed without contact with the beaker walls or bottom, and wait for the output value to stabilize. If the test is being carried out with the solution provided by Libelium at approximately 25 ºC, the output should be around 240 mV, with a 10%~15% error.
6. It is important to give time to the output to get stable, especially the first time we use a sensor. This will take several minutes.
7. A similar problem to the one mentioned for the pH sensor may appear owed to air bubbles in the sensitive bulb. If this is the case, shaking the sensor downward as stated for that sensor will also solve this problem.
8. Remove the sensor, rinse it with distilled or de-ionized water again and return it to its working place.
9. Write down the offset (the obtained value -- 240 mV) and introduce it in the Waspmote code or in the data processing in the receiver. Consider that there is no conversion function for this sensor in the Smart Water libraries.


SW_02_ORP_sensor_reading | Waspmote PRO IDE
File Edit Sketch Tools Help
SW_02_ORP_sensor_reading §
/dev/ttyUSB1
Send
0.001= 0.241 -0.240
Autoscroll
No line ending
115200 baud
39
waspmote-api-IDEv04-Final_Version on /dev/ttyUSB1

```
#include <WaspSensorSW.h>

float value_orp;
float value_calculated;

// Offset obtained from sensor calibration
#define calibration_offse0.001

ORPClass ORPSensor;

void setup()
{
  // Turn on the Smart Water sensor board and start the USB
  SensorSW.ON();
  USB.ON();
}

void loop()
{
  // Reading of the ORP sensor
  value_orp = ORPSensor.readORP();

  // Apply the calibration offset
  value_calculated = value_orp - calibration_offset;

  // Print of the results
  USB.print(F(" ORP Estimated: "));
  USB.print(value_calculated);
  USB.println(F(" volts"));

}
```

```
Done uploading.
Binary sketch size: 9,938 bytes (of a 122,880 byte maximum)
Chip memory sram: 1,685 bytes (of a 8,192 byte maximum)
```

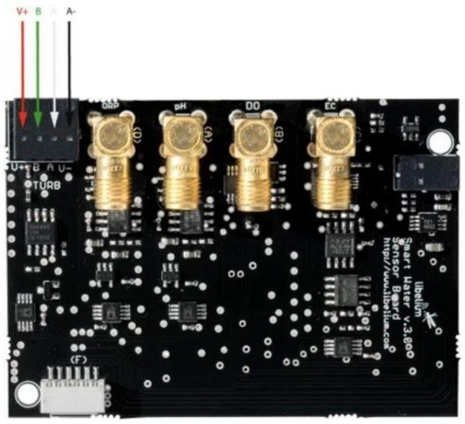
```
ORP Estimated: 0.0781269073 volts
ORP Estimated: 0.0859923362 volts
ORP Estimated: 0.0962507724 volts
ORP Estimated: 0.1046512126 volts
ORP Estimated: 0.1030783653 volts
ORP Estimated: 0.1117467880 volts
ORP Estimated: 0.1103951930 volts
ORP Estimated: 0.1163051128 volts
ORP Estimated: 0.1179583072 volts
ORP Estimated: 0.1175796985 volts
ORP Estimated: 0.1156163215 volts
ORP Estimated: 0.1174600124 volts
ORP Estimated: 0.1184363365 volts
ORP Estimated: 0.1197190284 volts
ORP Estimated: 0.1198012828 volts
ORP Estimated: 0.2384550571 volts
ORP Estimated: 0.2428956031 volts
ORP Estimated: 0.2425243854 volts
ORP Estimated: 0.2421851158 volts
ORP Estimated: 0.2420856952 volts
ORP Estimated: 0.2421176433 volts
ORP Estimated: 0.2418491840 volts
ORP Estimated: 0.2419407367 volts
ORP Estimated: 0.2418339252 volts
ORP Estimated: 0.2418835163 volts
ORP Estimated: 0.2416386604 volts
ORP Estimated: 0.2414762973 volts
ORP Estimated: 0.2414658069 volts
ORP Estimated: 0.2414293289 volts
ORP Estimated: 0.2417991161 volts
```

Turbidity Sensor:

The turbidity sensor is extremely sensitive, and the user must treat it with special care in all situations. The sensor must be installed in a solid way and protected from any impact. The turbidity sensor allows the measure of the temperature, so the Pt-1000 temperature sensor must be disconnected and cannot be used simultaneously with the turbidity sensor. The correct way to measure the turbidity using this sensor is to take samples for approximately 60-90 seconds and then make the mean between the measured values. The Turbidity sensor is calibrated in a factory and verified. Basically, the provider performs measurements with a range of normalized chemical solutions, which have a known and exact NTU value. This allows them to generate calibration data which is hard coded inside the sensor to improve the accuracy of the sensor.

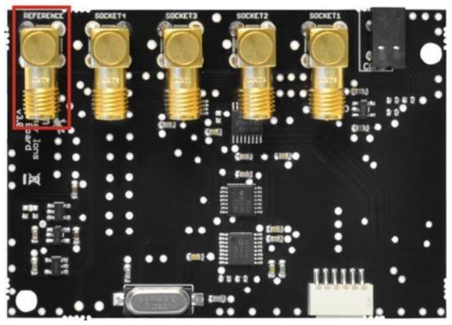
```
{
// Start a new measure
turbiditySensor.readTurbidity();
// Get the Turbidity Measure
float turbidity = turbiditySensor.getTurbidity();
}
```

Link for complete example code of turbidity sensor:
https://development.libelium.com/waspmote/sw-07-turbidity-sensor-reading



Socket on Smart Water board for Turbidity Sensor

Ion Sensors:



Reference Probe Connection

Reference Probe - A reference electrode is an electrode which has a stable and well-known electrode potential. Reference electrodes are critical to acquiring good electrochemical data. The Smart Water Ions Sensor Board has 3 different Reference Probes, depending on the ion to be measured.

The next sensors must be used with the Single Junction Reference Probe:

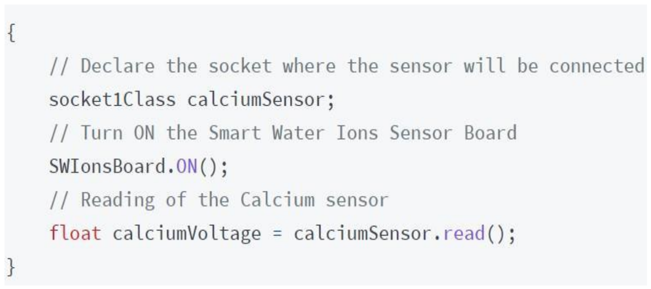
- Calcium Ion (Ca^{2+}) Sensor Probe
- Fluoride Ion (F^-) Sensor Probe
- Fluoroborate Ion (BF_4^-) Sensor Probe
- Nitrate Ion (NO_3^-) Sensor Probe

The next sensors must be used with the Double Junction Reference Probe:

- Bromide Ion (Br-) Sensor Probe
- Chloride Ion (Cl-) Sensor Probe
- Cupric Ion (Cu2+) Sensor Probe
- Iodide Ion (I-) Sensor Probe
- Silver Ion (Ag+) Sensor Probe

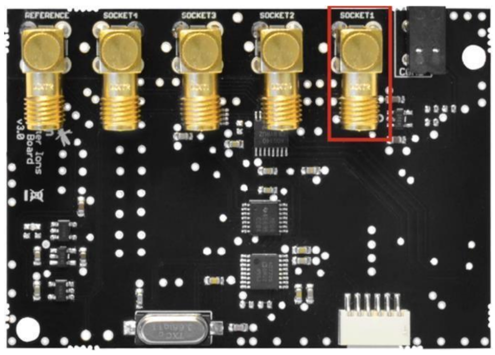
All the PRO sensors must be used with the PRO Reference Probe (including the pH [PRO] sensor). Sensor Connection – Only sensors of either single junction reference probe or double junction reference probe can be connected simultaneously and both types cannot be used simultaneously at same time.

Connect the sensor in the socket marked as SOCKET1 in the Smart Water Ions Sensor Board and connect the corresponding Reference Probe.



```
{
    // Declare the socket where the sensor will be connected
    socket1Class calciumSensor;
    // Turn ON the Smart Water Ions Sensor Board
    SWIonsBoard.ON();
    // Reading of the Calcium sensor
    float calciumVoltage = calciumSensor.read();
}
```

Link for complete example code of ion sensors:
https://development.libelium.com/waspmote/swi-03-socket1-sensor-reading/



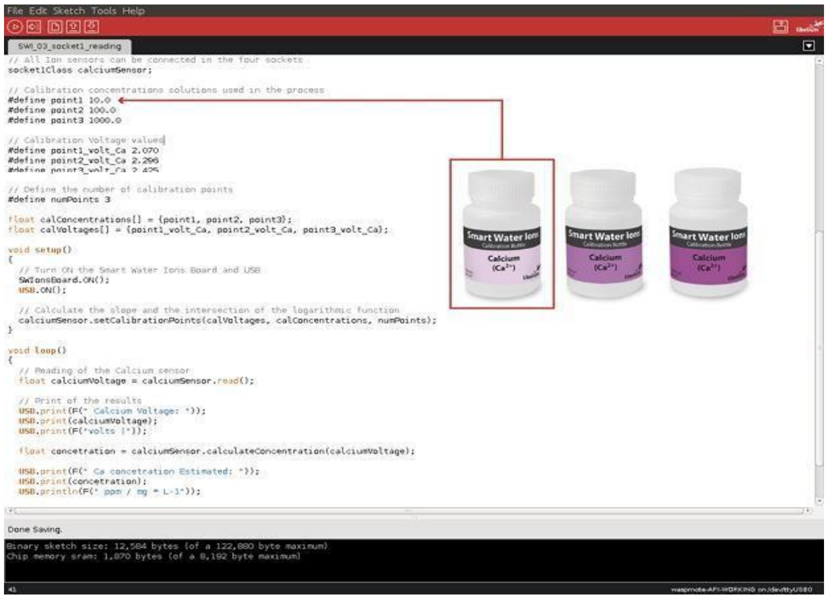
Socket 1 Connection

Repeat the same process for socket 2, socket 3 and socket 4.

Calibration Process - The calibration process is very sensitive, and it must be carried out carefully and following the recommendations of this technical guide. Three solutions are available for calibration – 10, 100, 1000 concentrations.

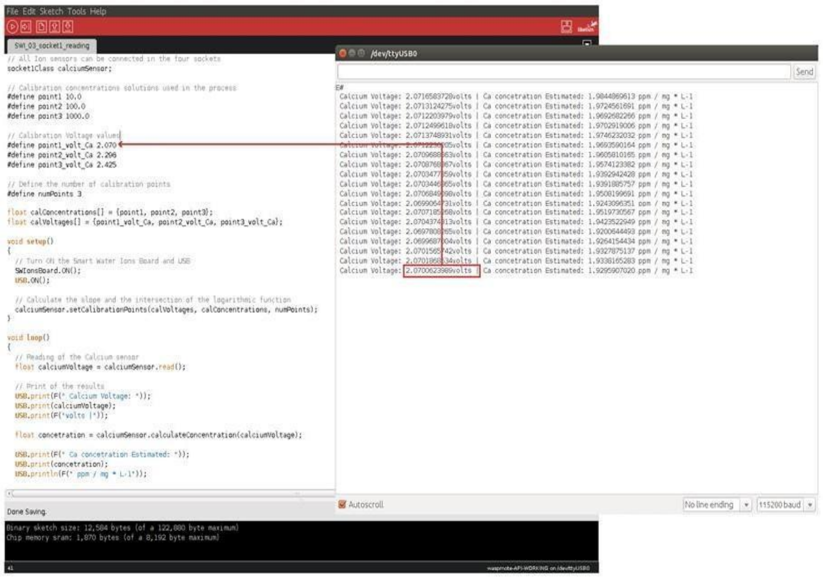
In this example we are going to calibrate the Calcium Ion (Ca2+) Sensor Probe in socket1 of the Smart Water Ions Sensor Board.

1. Turn on the Waspmote with the Smart Water Ions Sensor Board and the Calcium Ion (Ca2+) Sensor Probe connected.
2. Upload the example "Socket1 reading for Smart Water Ions" to the Waspmote board and make sure of receiving the data in the serial monitor.
3. Pour one of the solutions (about 50 ml should be enough) in a clean beaker. In this case we are going to use 10 mg/L solution of Calcium.
4. Immerse the probe in pure de-ionized water for about 30 seconds to ensure that the membranes were always in the same state at the beginning of each measurement.
5. Introduce the Calcium Ion (Ca2+) Sensor Probe in the solution and wait for a stable output. Make sure that the sensor is completely immersed in the solution and that it is not close to the beaker wall, which may affect the field between the electrodes and disturb the measurement. Once the output is steady, annotate the value in volts obtained.
6. After getting the sensor from the first solution, carefully rinse it and repeat the process explained in step 3 with the next solution.
7. Introduce the values noted and the concentration of the calibration solutions in your code, as shown in the next images.



```
// All Ion sensors can be connected in the four sockets
socket1Class calciumSensor;

// Calibration concentrations solutions used in the process
#define point1 10.0
#define point2 100.0
#define point3 1000.0

// Calibration Voltage values
#define point1_volt_Ca 2.070
#define point2_volt_Ca 2.296
#define point3_volt_Ca 2.425

// Define the number of calibration points
#define numPoints 3

float calConcentrations[] = {point1, point2, point3};
float calVoltages[] = {point1_volt_Ca, point2_volt_Ca, point3_volt_Ca};

void setup()
{
  // Turn ON the Smart Water Ions Board and USB
  SWIonsBoard.ON();
  USB.ON();

  // Calculate the slope and the intersection of the logarithmic function
  calciumSensor.setCalibrationPoints(calVoltages, calConcentrations, numPoints);
}

void loop()
{
  // Reading of the Calcium sensor
  float calciumVoltage = calciumSensor.read();

  // Print of the results
  USB.print(F(" Calcium Voltage: "));
  USB.print(calciumVoltage);
  USB.print(F("volts |"));

  float concetration = calciumSensor.calculateConcentration(calciumVoltage);

  USB.print(F(" Ca concetration Estimated: "));
  USB.print(concetration);
  USB.println(F(" ppm / mg * L-1"));
```

```
Done Saving.
Binary sketch size: 12,584 bytes (of a 122,880 byte maximum)
Chip memory sram: 1,870 bytes (of a 8,192 byte maximum)
41                                              waspmote-API-WORKING on /dev/ttyUSB0
```



```
// All Ion sensors can be connected in the four sockets
socket1Class calciumSensor;

// Calibration concentrations solutions used in the process
#define point1 10.0
#define point2 100.0
#define point3 1000.0

// Calibration Voltage values
#define point1_volt_Ca 2.070
#define point2_volt_Ca 2.296
#define point3_volt_Ca 2.425

// Define the number of calibration points
#define numPoints 3

float calConcentrations[] = {point1, point2, point3};
float calVoltages[] = {point1_volt_Ca, point2_volt_Ca, point3_volt_Ca};

void setup()
{
  // Turn ON the Smart Water Ions Board and USB
  SWIonsBoard.ON();
  USB.ON();

  // Calculate the slope and the intersection of the logarithmic function
  calciumSensor.setCalibrationPoints(calVoltages, calConcentrations, numPoints);
}

void loop()
{
  // Reading of the Calcium sensor
  float calciumVoltage = calciumSensor.read();

  // Print of the results
  USB.print(F(" Calcium Voltage: "));
  USB.print(calciumVoltage);
  USB.print(F("volts |"));

  float concetration = calciumSensor.calculateConcentration(calciumVoltage);

  USB.print(F(" Ca concetration Estimated: "));
  USB.print(concetration);
  USB.println(F(" ppm / mg * L-1"));
```

```
E#
 Calcium Voltage: 2.0716583728volts | Ca concetration Estimated: 1.9844869613 ppm / mg * L-1
 Calcium Voltage: 2.0713124275volts | Ca concetration Estimated: 1.9724561691 ppm / mg * L-1
 Calcium Voltage: 2.0712203979volts | Ca concetration Estimated: 1.9692682266 ppm / mg * L-1
 Calcium Voltage: 2.0712499618volts | Ca concetration Estimated: 1.9702919006 ppm / mg * L-1
 Calcium Voltage: 2.0713748931volts | Ca concetration Estimated: 1.9746232032 ppm / mg * L-1
 Calcium Voltage: 2.07122[illegible]05volts | Ca concetration Estimated: 1.9693590164 ppm / mg * L-1
 Calcium Voltage: 2.0709688663volts | Ca concetration Estimated: 1.9605810165 ppm / mg * L-1
 Calcium Voltage: 2.0708768967volts | Ca concetration Estimated: 1.9574123382 ppm / mg * L-1
 Calcium Voltage: 2.0703477859volts | Ca concetration Estimated: 1.9392942428 ppm / mg * L-1
 Calcium Voltage: 2.0703446865volts | Ca concetration Estimated: 1.9391885757 ppm / mg * L-1
 Calcium Voltage: 2.0706849098volts | Ca concetration Estimated: 1.9508199691 ppm / mg * L-1
 Calcium Voltage: 2.0699064731volts | Ca concetration Estimated: 1.9243096351 ppm / mg * L-1
 Calcium Voltage: 2.0707185268volts | Ca concetration Estimated: 1.9519730567 ppm / mg * L-1
 Calcium Voltage: 2.0704374813volts | Ca concetration Estimated: 1.9423522949 ppm / mg * L-1
 Calcium Voltage: 2.0697808265volts | Ca concetration Estimated: 1.9200644493 ppm / mg * L-1
 Calcium Voltage: 2.0699687004volts | Ca concetration Estimated: 1.9264154434 ppm / mg * L-1
 Calcium Voltage: 2.0701565742volts | Ca concetration Estimated: 1.9327875137 ppm / mg * L-1
 Calcium Voltage: 2.0701868634volts | Ca concetration Estimated: 1.9338165283 ppm / mg * L-1
 Calcium Voltage: 2.0700623989volts | Ca concetration Estimated: 1.9295907020 ppm / mg * L-1
```

```
Done Saving.
Binary sketch size: 12,584 bytes (of a 122,880 byte maximum)
Chip memory sram: 1,870 bytes (of a 8,192 byte maximum)
41                                              waspmote-API-WORKING on /dev/ttyUSB0
```

8. The function `setCalibrationPoints()` is used to configure the calibration parameters. The pH Sensor Probe (for Smart Water Ions), has its own calibration function called `setpHCalibrationPoints()`.
9. During the process, the temperature of the solution must be measured and annotated.
10. Upload the code again with the new calibration values obtained from the calibration process.

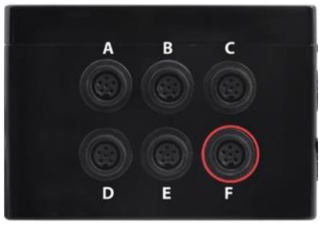
Fluorometer Chlorophyll Sensor:

Fluorometric sensors emit light at a certain wavelength and look for a very specific wavelength in return. The magnitude of the return light is relatable to the amount of the analysed parameter. They require non-trivial calibration.

In the case of chlorophyll, two fluorometers are available with the ranges shown below:

Chlorophyll a -- blue: 0 to 500 g/l

Chlorophyll a -- red: > 500 g/l



Socket on Smart Water Xtream for Chlorophyll Sensor

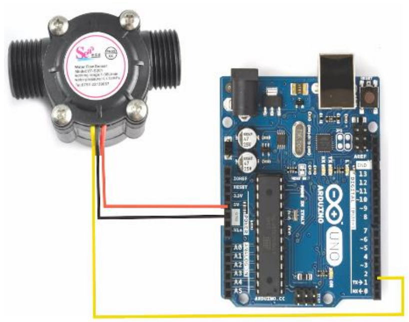
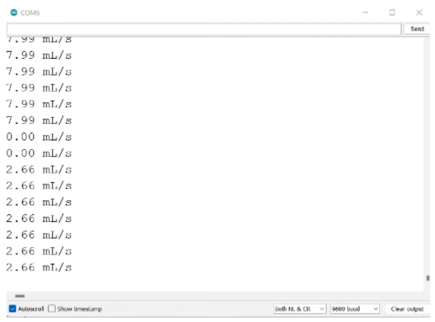
Aqualabo C4E TDS Sensor:

This sensor can measure specific conductance, salinity and total dissolved solids (TDS). The specific conductance is compensated for temperature. It includes four easy-to-clean graphite electrodes, and an optional sensor provides ±0.5% of reading accuracy to 100 mS/cm. The salinity is calculated from specific conductance. PSS = Practical Salinity Scale which is roughly equivalent to ppt. Total dissolved solids (TDS) are calculated from specific conductance.

```
{
    // 1. Declare an object for the sensor
    eureka mySensor;
    // 2. Turn ON the sensor
    mySensor.ON();
    // 3. Read the sensor. Values stored in class variables
    // Check complete code example for details
    mySensor.read();
    // 4. Turn off the sensor
    mySensor.OFF();
}
```

Water Flow Sensor

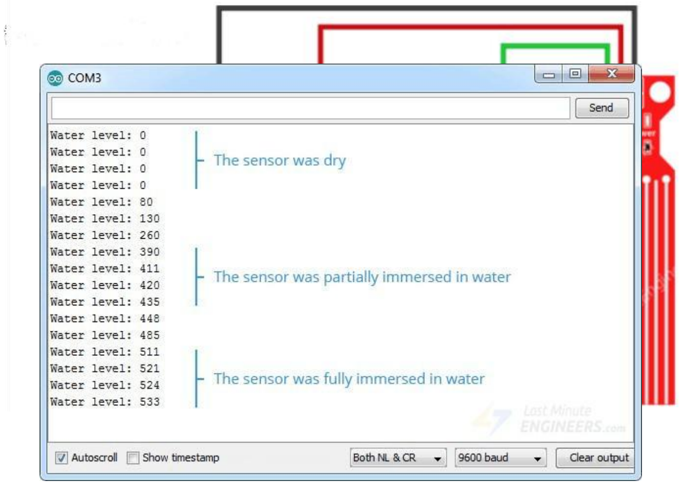
This calculates the amount of water used and flow of water in pipes.

```
7.99 mL/s
7.99 mL/s
7.99 mL/s
7.99 mL/s
7.99 mL/s
7.99 mL/s
0.00 mL/s
0.00 mL/s
2.66 mL/s
2.66 mL/s
2.66 mL/s
2.66 mL/s
2.66 mL/s
2.66 mL/s
2.66 mL/s
```

Water Level Sensor:

This sensor is able to measure the water level, monitor a sump pit, detect rainfall, and detect leaks. The sensor has ten exposed copper traces, five of which are power traces and the remaining five are sense traces. These traces are interlaced so that there is one sense trace between every two power traces. The resistance of the sensor varies inversely with the depth of immersion of the sensor in water. The sensor generates an output voltage proportional to the resistance; by measuring this voltage, the water level can be determined. It is controlled using an Arduino UNO microcontroller.



```
Water level: 0
Water level: 0
Water level: 0
Water level: 0
Water level: 80
Water level: 130
Water level: 260
Water level: 390
Water level: 411
Water level: 420
Water level: 435
Water level: 448
Water level: 485
Water level: 511
Water level: 521
Water level: 524
Water level: 533
```

The sensor was dry

The sensor was partially immersed in water

The sensor was fully immersed in water

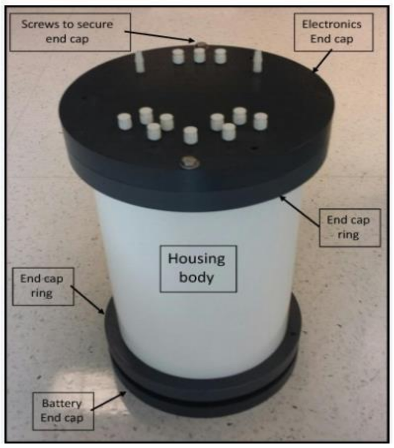
Casing of the System

The IoT system needs to be closured and kept protected from any potential damages due to natural or other causes in order to provide the best possible performance and accurate measurements, especially when the application of the system is in a harsh and uneasy terrain and in close contact with moisture as the system alone may come short in handling such exposures. Hence a casing for the system, which is water resistant, rouged, transparent and long lasting in natural conditions is essential. All these factors point towards the use of weather-resistant plastics for casing of the device.

Weather-resistant plastics have superior durability suited for extensive outdoor applications. Their wide variety of material properties including heat, corrosion and impact resistance, ease of molding and manufacture, bio-deterioration proof and cost effectiveness make them exceptionally adaptive for above mentioned application. Plastic materials including PDCPD, Phenolic resins, Polycarbonate or Polyurethane can be used.

We propose a methodology using sampling. A sample amount of water is taken in from the region of interest into a testing environment, where all the tests mentioned are being conducted with the help of sensors. This procedure is done on a regular basis of trusted time interval so as to get the idea regarding the entire population of water in the respective water body. The system will be enclosed in the whether resistant plastic casing consisting of holding mechanism for the sensor probes on the lid and only the tip of the sensor probes will be exposed into the material to be examined. The transparency of the casing allows for the inspection of proper alignment of sensors for producing accurate data. Plastic makes the casing lightweight, durable, and easy to relocate. The whole system can be installed on riverbanks, near water bodies or any place of convenience.

Casing of IoT System



Autonomous Water Sampler

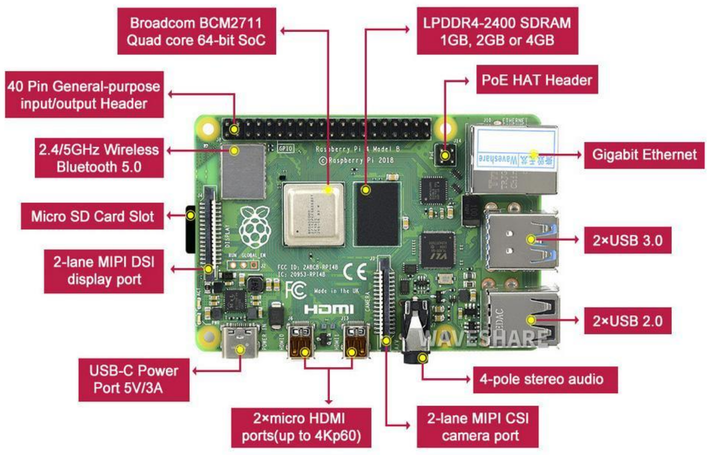
WQM parameters and testing in artificially created environment:

Parameter	**Permissible range/min required quantity**
Ph	6.5-9.5
Turbidity	<5 NTU
Temperature	6 deg Celsius to 20 deg Celsius
Dissolved Oxygen	6.5-8 mg/L or 80-120%
TDS	50-150
Cu	< 1300 ug/L
Fe	< 0.3 mg/L
Nitrates	< 10 mg/L
Cl-	< 4 mg/L
Conductivity	< 1000 mcg/cm
ORP	300-500 mV
Chlorophyll	< 50 mg/m^2

With these parameters we can create artificial polluted Nile water in the lab and debug the system for better functionality and accurate results.

ALTERNATIVE:

We can measure the general quantities (pH, Turbidity, Temperature, Level of water (abrupt changes will be alerted), Dissolved Oxygen and conductivity) along with one important pollutant (here it is nitrate N, indicating agricultural pollution).



RASPBERRY PI 4

SENSOR AND EXPERIMENTAL SETUP FOR CHARACTERIZATION OF THE SENSOR.

Planar type interdigital sensors can be used for the detection of nitrate-N in water samples. Characterization of the sensor is required for accurately measuring the quantity and a stabilization and interfacing circuit is required for connecting it to a microcontroller.

ensor

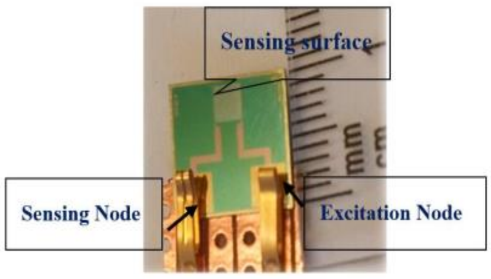


Fig. 2. Interdigital sensor with Parylene coating

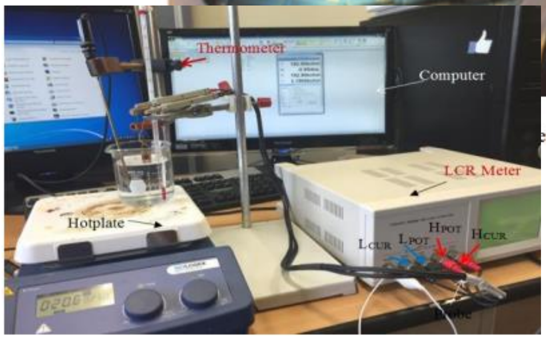


Fig. 3. Experimental arrangement for temperature measurement