AIT WSN workshop

December 2014
Sebastian Büttrich

Sensor Node for Aquaculture

Table of Contents

Summary

In the AIT WSN workshop in December 2014, we deployed one

Solar powered, Arduino based sensor node currently measuring the following:

- water temperature
- pH value
- Dissolved oxygen (DO)

The nodes network connection is via a 2.4 GHz WiFI link, served by an AP at the Habitech building.

Sensor data are sent to a Thingspeak server hosted by the AIT.

Team

Chavalit Srisathapornphat	Chavalit.S@ku.th (Kasetsart University)
Tatchai	? (Aquaculture)
Jintanan	? (AIT Aquaculture)
Sethavidh Gertphol	fscisvg@ku.ac.th (Kasetsart University)
Apinun Tunpan	atunpan@ait.asia (intERLab/AIT)
Muhammad Faran Majeed	m.faran.majeed@gmail.com (intERLab/AIT)
Gabriel Ataguba	gabynotepad@yahoo.co.uk
Sebastian Büttrich	sebastian@nsrc.org

Water sensor node – Parts List and Cost

Item	**URL**	**Cost [USD]**	**Comments**
Seeduino Stalker v 2.3	http://www.seeedstudio.com/depot/seeeduino-stalker-v2b-p-727.html	39	
Arduino WiFi Shield	http://arduino.cc/en/Main/ArduinoWiFiShield	85	
Water temperature	https://www.atlas-scientific.com/product_pages/probes/env-tmp.html?	22	
Dissolved Oxygen	https://www.atlas-scientific.com/product_pages/probes/do_probe.html?	198	
pH	https://www.atlas-scientific.com/product_pages/kits/ph-kit.html?	128	
Atlas Arduino Shield	https://www.atlas-scientific.com/product_pages/components/arduino.html	24	
DC DC Power Module	http://www.dfrobot.com/index.php?route=product/product&product_id=752&search=dcdc&description=true	8.50	
Battery 7 Ah	http://www.amazon.com/ExpertPower%C2%AE-Rechargeable-12V-Sealed-Battery/dp/B003S1RQ2S/ref=sr_1_1?ie=UTF8&qid=1418978418&sr=8-1&keywords=expertpower+7AH	16.50	
Solar Panel 10 W	Sunsolar ecotech 10 W 17.5 V	15	
TOTAL		**536**	
Cables		??	
Enclosure		10	
Nuts and bolts		10	

Next steps & unfinished work

On the first node, in order of urgency:

- **Plexi plate and clamps** instead of wire and strips to hold the box. Replace the broken plexi plate.
- **Waterproofing:** Check if the lid of the enclosure closes properly..
- Analyze and solve the issue with the **pH value loop delay**. Currently readings alternate systematically between 0 and the correct value.
- **Recalibration of the DO probe** – we keep seeing DO values > 100%, which means our peak voltage (corresponding to 100%) is set too low. It is currently 28.5 mV, which in fact is lower than the probe manuals let us expect.
- A **power switch or plug connector** that allows for power on/off without opening the box,

In addition to this:

- A **second node** should be built – all components have been left at the lab.
- **Securing the data portal** (access control?)
- Integrate additional sensors / properties to measure, such Nitrate, Nitrite, Alkalinity, Ammonia, etc

Circuit

We do not document the circuit here – please see the URLs of the respective sensors for instructions.

Power consumption

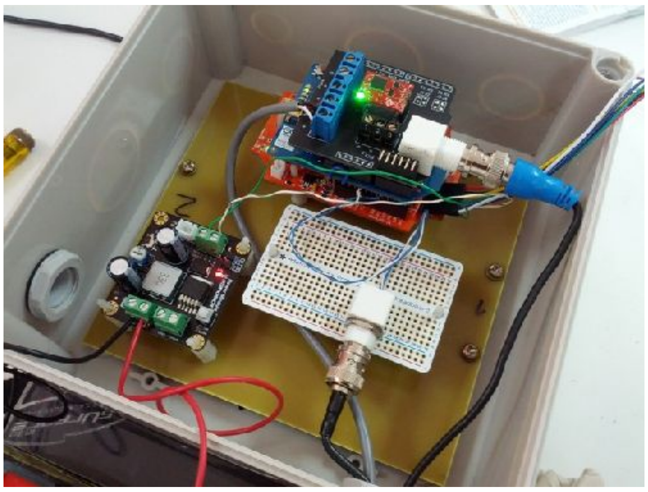
Power consumption measured via USB wattmeter: 0.15 A at 5 V ==> 0.75 W

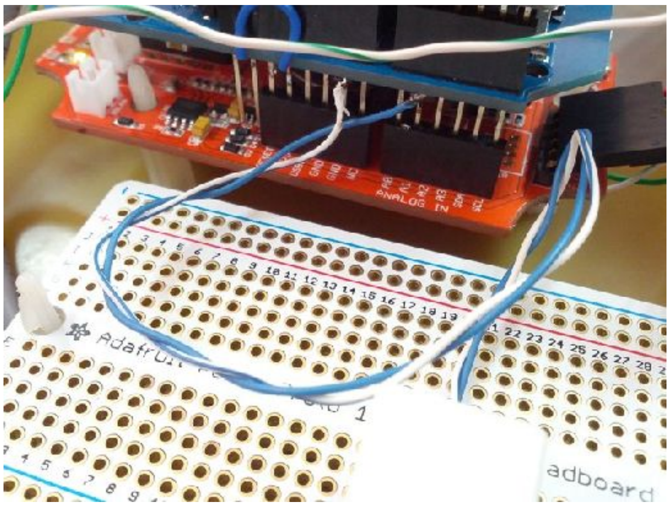
URLs

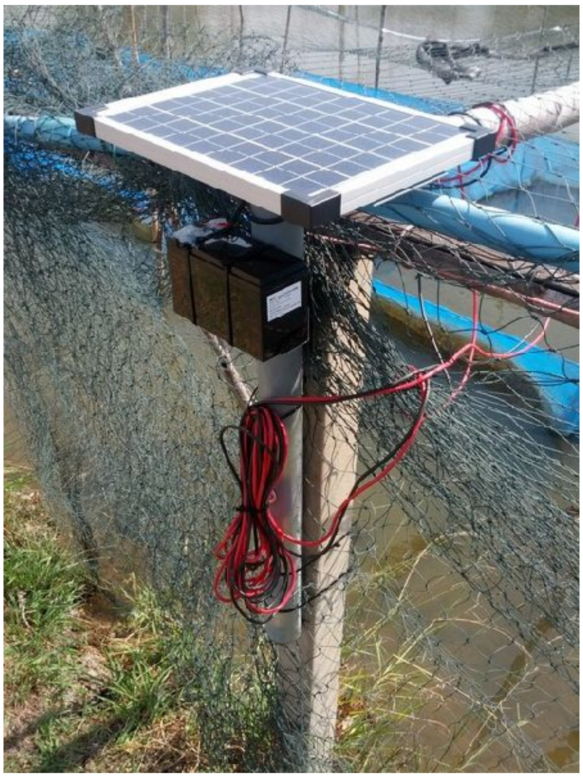
https://etherpad.mozilla.org/vYuUeuBxgp

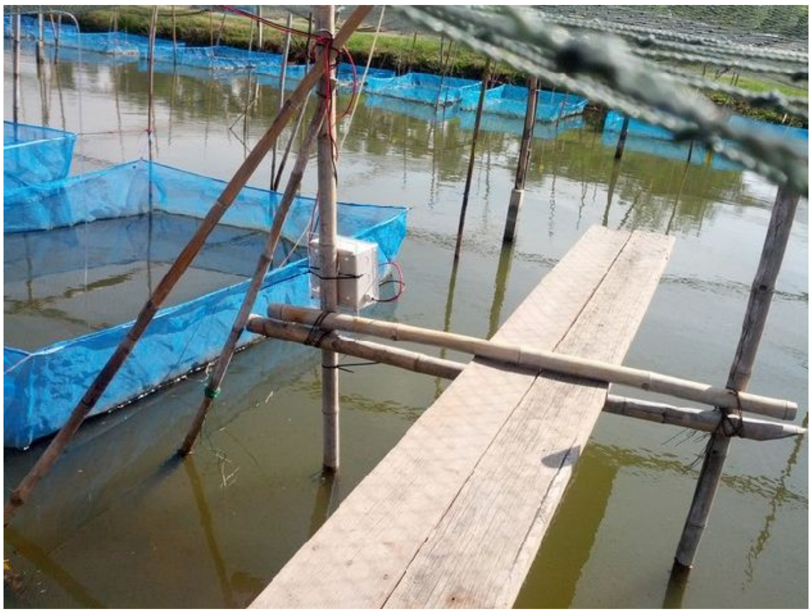
http://203.159.0.30:3000/channels/39

Pictures of deployed system

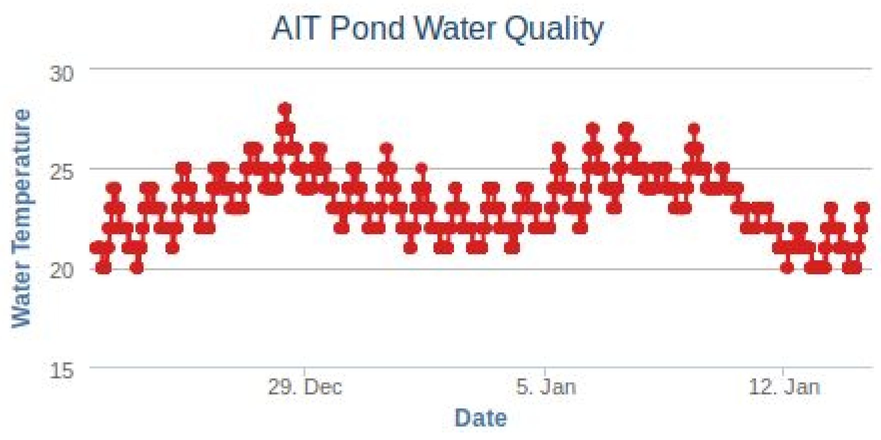


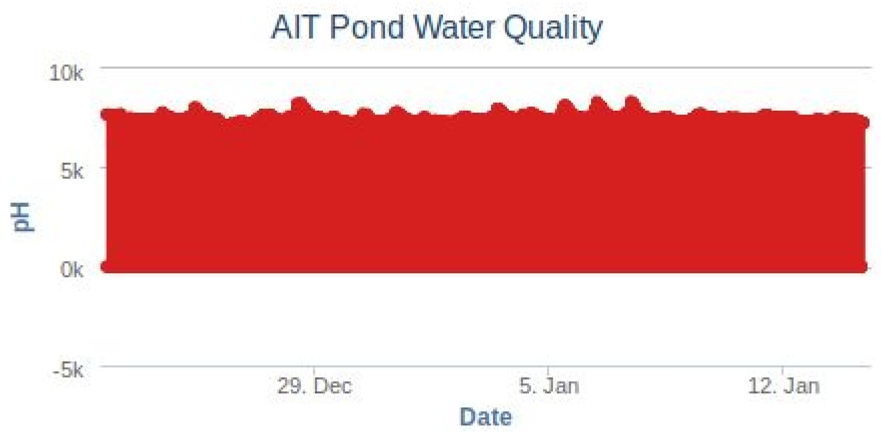


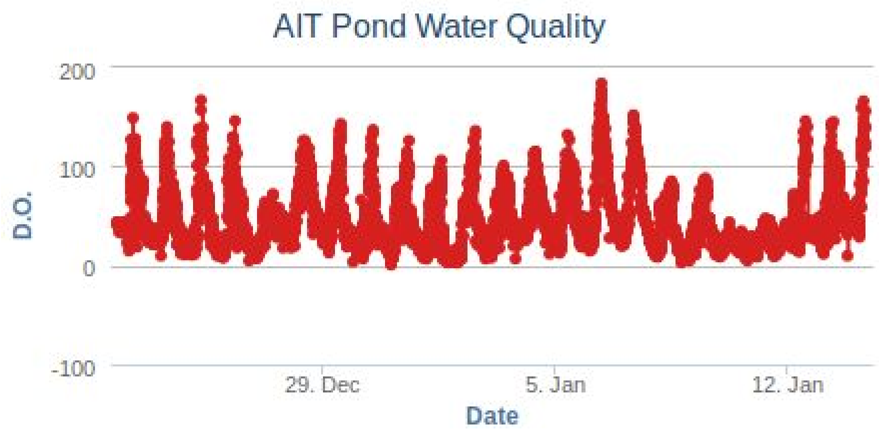


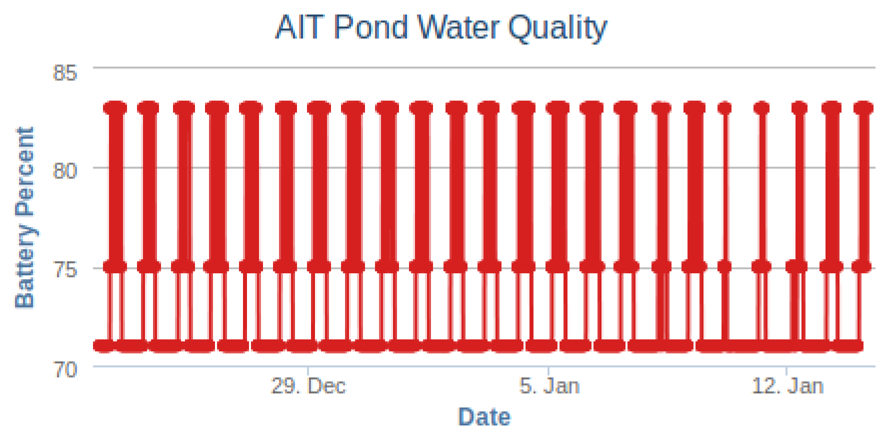


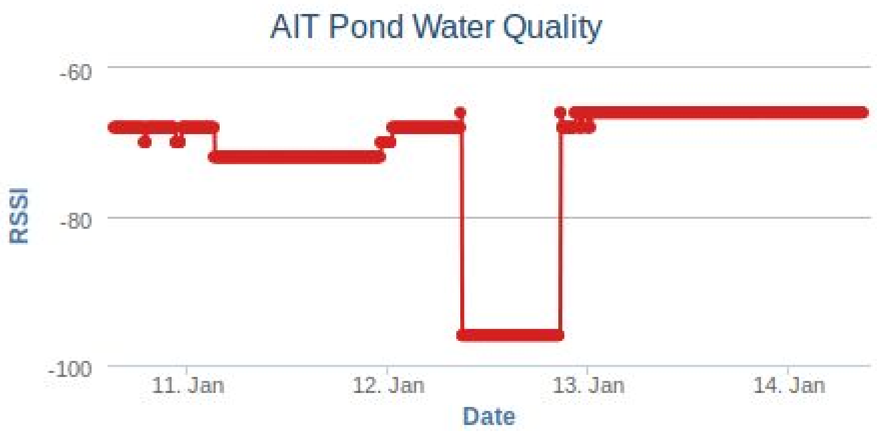
Sample sensor data December 2014 / January 2015

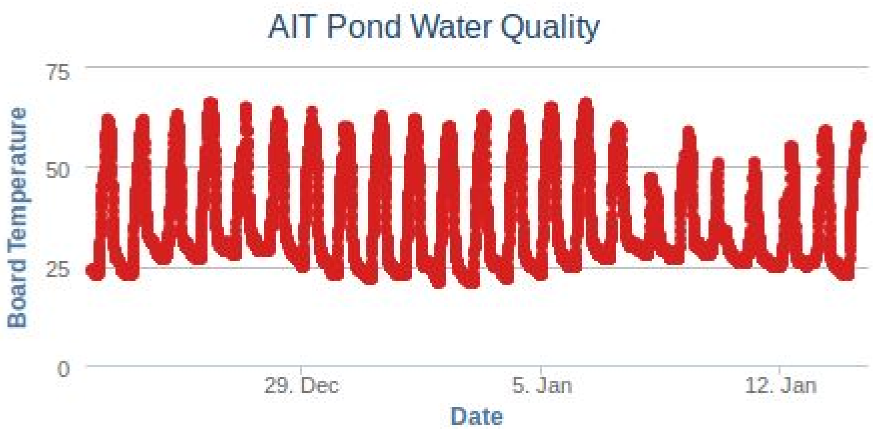







AIT Pond Water Quality
Battery Percent
85
80
75
70
29. Dec
5. Jan
12. Jan
Date


AIT Pond Water Quality
RSSI
-60
-80
-100
11. Jan
12. Jan
13. Jan
14. Jan
Date


AIT Pond Water Quality
Board Temperature
75
50
25
0
29. Dec
5. Jan
12. Jan
Date

Future ideas

- A Wireless Network for Aquaculture

  The whole area of the aquaculture research ponds may easily be covered by one central network node and set of Antennas, e.g. to be located on a lamp post or other existing elevated point.
  Such a network might be a 2.4 GHz WiFi network, but also 433 MHz or 868, or TVWS frequencies might be considered (and regulatory aspects addressed).

  The backhual connection for this central node may be established via a WiFi point to point link to this post.

  Power for this network node might come from the powerline feeding the lamp post, or, alternatively from a solar installation at the aquaculture complex – see below.

- Power network for Aquaculture

  Solar panels on the roof of the hut at the ponds to power all the pond complex – a first rough estimate of the roof surface indicates that 4-5 kW peak power should be possible.

Appendix: Arduino code

The following is the code at time of deployment, 20141219.

```
### Note: The first 3 pH readings are 0 and this is normal. However, right now
the subsequent readings alternate between 0 and actual value. This may be caused
by the loop delay that is too high (greater than 30 seconds.) May have to try
multiple readings in a loop.

 // Water Quality measurement code

//  for Seeeduino + WiFi + pH ADC shield + Temp and DO probes

//  part of IoT training at AIT 2014

//  by Sethavidh Gertphol and Chavalit Srisathapornphat

//  participants from Kasetsart University

#include <SoftwareSerial.h>      //we have to include the SoftwareSerial
library, or else we can't use it.

#include <Wire.h>

#include <Battery.h>

#include "DS3231.h"

#include <WiFi.h>

#define sensorReadingInterval 600000 //define delay between readings in
milliseconds

#define numSecWait            10  //define number of seconds waiting for reply
from ThingSpeak before break

#define numReconnect           5  //define number of trying to reconnect WiFi if
lost

#define rx 2                      //define what pin rx is going to be.

#define tx 3                      //define what pin Tx is going to be.

SoftwareSerial myserial(rx, tx); //define how the soft serial port is going to
work.

//Pin setting

#define TEMP_PROBE_READ_PIN 0

#define DO_PROBE_READ_PIN   1

//DO Probe setting

#define DO_DELAY_READ       100 //in ms

#define DO_NUM_READ         89  //number DO read before avg

//pH probe setting

char ph_data[20];                  //we make a 20 byte character array to hold
incoming data from the pH.

//Battery setting

Battery battery;

int chcnt = 0;
```

```
//Temp/Time setting
DS3231 rtc;
//WiFi setting
#define WiFiWait       15  // seconds to wait for WiFi connection
char ssid[] = "WSN";     //  your network SSID (name)
char pass[] = "sensor2014";  // your network password
//char ssid[] = "WSN";     //  your network SSID (name)
//char pass[] = "sensor2014";  // your network password
int status = WL_IDLE_STATUS;     // the Wifi radio's status
WiFiClient client;
long    lastConnectionTime = 0;
boolean lastConnected = false;
int     failedCounter = 0;
// ThingSpeak Settings
char thingSpeakAddress[] = "203.159.0.30";  //TS
String writeAPIKey = "XY1DTMWT59FTXNX6";    //TS
const int updateThingSpeakInterval = 16 * 1000;      // Time interval in
milliseconds to update ThingSpeak (number of seconds * 1000 = interval)
String timestamp;
int numWait = 0;
int reconnectCount = 0;
void setup(){
  Serial.begin(9600);          //enable the hardware serial port
  myserial.begin(9600);        //enable the software serial port
  Wire.begin();
  rtc.begin();

  //start-up pH probe
  myserial.print("c,0\r");   //take the pH Circuit out of continues mode.
  delay(50);                 //on start up sometimes the first command is
missed.
  myserial.print("c,0\r");   //so, let’s send it twice.
  delay(50);                 //a short delay after the pH Circuit was taken out
of continues mode is used to make sure we don’t over load it with commands.
  connectWiFi();
}

void loop(){
  float water_temp;
  float pH;
  float DO;
  int batt_lvl;
  float board_temp;
  int rssi;
  String updateString;

  Serial.println("====Starting sensors reading.====");
  DO = read_DO();
  delay(5000);
  pH = read_pH();
  water_temp = read_temp();    //call the function “read_temp” and return the
temperature in C°
  board_temp = read_board_temp();
  batt_lvl = read_batt();
  //get_timestamp();
  //Serial.println(timestamp);

  //marshalling update message
  char s[100];
  updateString = "1=";
  updateString += int(water_temp);
  updateString += "&2=";
  /*pH=4.567;
  if (pH >= 10.0) {
    dtostrf(pH, 6,3,s);
  }
  else {
    dtostrf(pH,5,3,s);
  }*/
  /*
  char pHString[6];
  int pH100 = int(pH*100);
  pHString[0] = int(pH100/1000)+48;
  pH100 = pH100 % 1000;
  pHString[1] = int(pH100/100)+48;
  pH100 = pH100 % 100;
  pHString[2] = '.';
  pHString[3] = int(pH100/10)+48;
  pH100 = pH100 % 10;
  pHString[4] = pH100+48;
  pHString[5] = '\0';
  updateString += pHString;
  */
  updateString += int(pH*1000);
```

```
 updateString += "&3=";
 updateString += int(DO);
 updateString += "&4=";
 updateString += batt_lvl;
 updateString +="&5=";
 updateString += WiFi.RSSI();
 updateString +="&6=";
 updateString += int(board_temp);
 /*
 updateString +="&7=";
 updateString += timestamp;
*/
 Serial.println(updateString);
 //sendToThingSpeak(updateString);

 //check and reconnect WiFi if necessary
 while ( WiFi.status() != WL_CONNECTED && reconnectCount < numReconnect) {
   Serial.print("Attempting to re-connect to WPA SSID: ");
   Serial.println(ssid);
   // Connect to WPA/WPA2 network:
   status = WiFi.begin(ssid, pass);
   reconnectCount += 1;
   // wait 15 seconds for connection:
   delay(WiFiWait * 1000);
 }

 reconnectCount = 0;
 //sending to ThingSpeak
 Serial.println("Sending to ThingSpeak.");
 updateThingSpeak(updateString);
 while (!client.available()  && numWait < numSecWait) {
   Serial.println("Waiting...");
   numWait += 1;
   delay(1000);
 }
 numWait = 0;
 while (client.available())
 {
   char c = client.read();
   Serial.print(c);
 }
```

```
  Serial.println("...disconnected");
  Serial.println();
  delay(sensorReadingInterval);

  client.flush();
  Serial.println("Client flushed");
  client.stop();
  Serial.println("Client stopped");
}
void sendToThingSpeak(String & s){
  Serial.println(s);
  // Print Update Response to Serial Monitor
  while (client.available())
  {
    char c = client.read();
    Serial.print(c);
  }

  // Disconnect from ThingSpeak
  if (!client.connected() && lastConnected)
  {
    Serial.println("...disconnected");
    Serial.println();

    client.stop();
    client.flush();
  }

  //connect to ThingSpeak
  if(!client.connected() && (millis() - lastConnectionTime >
updateThingSpeakInterval))
  {
     Serial.println("Sending to ThingSpeak.");
     updateThingSpeak(s);
  }

  lastConnected = client.connected();
}
void get_timestamp(){
    DateTime now = rtc.now(); //get the current date-time
    timestamp = String(now.year())+"/"+String(now.month())
+"/"+String(now.date());
    timestamp = timestamp+"-"+String(now.hour())+":"+String(now.minute())
+":"+String(now.second());
    //Serial.println(timestamp);
    //return timestamp;
}
float read_board_temp(){

  float temp;
  rtc.convertTemperature();
  temp = rtc.getTemperature();
  Serial.print("Board temperature = ");
  Serial.print(temp);
  Serial.println(" C.");
  return temp;

}
int read_batt(){
  battery.update();
  float voltage = battery.getVoltage();
  int percentage = battery.getPercentage();
  String percentageString = String(percentage);
  char* CS = battery.getChStatus();

  bool ch = battery.isCharging();
  if(ch) chcnt++;

  Serial.print("battery: ");
  Serial.print(voltage);
  Serial.print("V  -> ");
  Serial.print(percentage);
  Serial.print("%     Charge Status: ");
  Serial.print(CS);
  Serial.print("     charging counter: ");
  Serial.println(chcnt);
  return percentage;
}
float read_DO(){
  int i=0;
  float v_sum=0;
  float v_avg=0;
  float sat_avg=0;
  String DO;
  String temperature;
  analogReference(INTERNAL);  // INTERNAL = 1.1v
  Serial.println("Switching voltage to INTERNAL (1.1v)");
  delay(1000);

  do
  {
    delay(DO_DELAY_READ);          // wait for sensors to stabilize
    int sensorValue = analogRead(DO_PROBE_READ_PIN);
    float voltage = sensorValue * (1100.0/1023.0);
    //Serial.println(voltage);
    v_sum=v_sum+voltage;
    i++;
  } while (i < DO_NUM_READ);
  //delay(5000);
  v_avg=v_sum/DO_NUM_READ;
  sat_avg=v_avg/28.5*100;
  DO = String((int)(sat_avg),DEC);
  Serial.println("Voltage / Saturation average");
  Serial.print(v_avg);
  Serial.print("mV / ");
  Serial.print(DO);
  Serial.println("% ");

  analogReference(DEFAULT);  // DEFAULT = 3.3v
  Serial.println("Switching voltage to DEFAULT (3.3v)");
  delay(1000);

  return sat_avg;
}
float read_pH(){
  float ph;
  byte received_from_sensor=0;       //we need to know how many characters have
been received.
  byte string_received=0;            //used to identify when we have received a
string from the pH circuit.

  if(myserial.available() > 0){        //if we see that the pH Circuit has sent
a character.
     received_from_sensor=myserial.readBytesUntil(13,ph_data,20); //we read the
data sent from pH Circuit until we see a <CR>. We also count how many character
```

```
have been received.
     ph_data[received_from_sensor]=0;  //we add a 0 to the spot in the array
just after the last character we received. This will stop us from transmitting
incorrect data that may have been left in the buffer.
     string_received=1;               //a flag used when the Arduino is
controlling the pH Circuit to let us know that a complete string has been
received.
     Serial.print("pH = ");
     Serial.println(ph_data);          //lets transmit that data received from
the pH Circuit to the serial monitor.
     }
   myserial.print("R\r");             //send it the command to take a single
reading.
   if(string_received==1){            //did we get data back from the ph
Circuit?
     ph=atof(ph_data);                //many people ask us "how do I convert a
string into a float?" This is how...
     string_received=0;              //reset the string received flag.
   }
   return ph;
}
float read_temp(void){   //the read temperature function
  float v_out;             //voltage output from temp sensor
  float temp;              //the final temperature is stored here
  String temperature;
  digitalWrite(TEMP_PROBE_READ_PIN, LOW);   //set pull-up on analog pin
  delay(2);                //wait 2 ms for temp to stabilize
  v_out = analogRead(TEMP_PROBE_READ_PIN);   //read the input pin
  v_out*= 0.0048;            //convert ADC points to volts (we are using .0048
because this device is running at 5 volts)
  v_out*=1000;             //convert volts to millivolts
  temp= 0.0512 * v_out -20.5128; //the equation from millivolts to temperature
  Serial.print("Voltage read =");
  Serial.println(v_out);
  Serial.println(temp);  //print the temperature data
  temperature = String(int(temp),DEC);
  Serial.print(temperature);
  Serial.write(186);
  Serial.println("C");
  return temp;             //send back the temp
}
void connectWiFi(){
    // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue:
    while(true);
  }

  // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network:
    status = WiFi.begin(ssid, pass);
    // wait 10 seconds for connection:
    delay(WiFiWait * 1000);
  }

  // you're connected now, so print out the data:
  Serial.print("You're connected to the network");
  printCurrentNet();
  printWifiData();
}
void printWifiData() {
  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  Serial.println(ip);

  // print your MAC address:
  byte mac[6];
  WiFi.macAddress(mac);
  Serial.print("MAC address: ");
  Serial.print(mac[5],HEX);
  Serial.print(":");
  Serial.print(mac[4],HEX);
  Serial.print(":");
  Serial.print(mac[3],HEX);
  Serial.print(":");
  Serial.print(mac[2],HEX);
  Serial.print(":");
  Serial.print(mac[1],HEX);
  Serial.print(":");
  Serial.println(mac[0],HEX);

}
void printCurrentNet() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  // print the MAC address of the router you're attached to:
  byte bssid[6];
  WiFi.BSSID(bssid);
  Serial.print("BSSID: ");
  Serial.print(bssid[5],HEX);
  Serial.print(":");
  Serial.print(bssid[4],HEX);
  Serial.print(":");
  Serial.print(bssid[3],HEX);
  Serial.print(":");
  Serial.print(bssid[2],HEX);
  Serial.print(":");
  Serial.print(bssid[1],HEX);
  Serial.print(":");
  Serial.println(bssid[0],HEX);
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.println(rssi);
  // print the encryption type:
  byte encryption = WiFi.encryptionType();
  Serial.print("Encryption Type:");
  Serial.println(encryption,HEX);
  Serial.println();
}
void updateThingSpeak(String &tsData)
{
  Serial.println(tsData);
  Serial.println(tsData.length());

  if (client.connect(thingSpeakAddress, 3000))
  {
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak:3000\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+writeAPIKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(tsData.length());
    client.print("\n\n");
    client.print(tsData);

    lastConnectionTime = millis();

    if (client.connected())
    {
      Serial.println("Connecting to ThingSpeak...");
      Serial.println();

      failedCounter = 0;
    }
    else
    {
      failedCounter++;

      Serial.println("Connection to ThingSpeak failed ("+String(failedCounter,
DEC)+")");
      Serial.println();
    }

  }
  else
  {
    failedCounter++;

    Serial.println("Connection to ThingSpeak Failed ("+String(failedCounter,
DEC)+")");
    Serial.println();

    lastConnectionTime = millis();
  }
}
```

Appendix 2: Group etherpad, Status 20150112

Water quality Pad

Team:

Sebastian +4560434784 / +66 929458732

Chavalit +66-840003750 (Kasetsart University)

Tatchai +66894801258 (Aquaculture)

Jintanan +66870196292 (Aquaculture)

Sethavidh +66846468663 (Kasetsart University)

Apinun +66 84 771 32232 (intERLab/AIT)

Faran: +66 806 163230 (intERLab/AIT)

Gabriel +66955763349

What are we measuring exactly? จะวัดอะไร?

===

Temperature https://www.atlas-scientific.com/product_pages/probes/env-tmp.html?

pH https://www.atlas-scientific.com/product_pages/probes/ph_probe.html?

Dissolved oxygen https://www.atlas-scientific.com/product_pages/probes/do_probe.html?

all analog sensors

Calibration!!!

===

how do we calibrate? D.O. - at the water lab, comparing to a trusted machine

The water will then be de-oxygenated using sodium trioxosulphate V (Na_2SO_3) solution using cobalt chloride ($CoCl_2.6H_2O$) as a catalyst (APHA 1980).

The initial DO concentration of the water will be used to determine the quantity of Na_2SO_3 to be added. Concentration of de-oxygenation chemicals will be added as indicated by Lawson (2002). Na_2SO_3 will be made into solution first at a concentration of 7.88mg/L of water. This is the amount required to remove1.0mg/L of DO. Hence calculation will be carried out to determine the required concentration according to the amount of initial DO in the test water. The cobalt chloride solution will be added at a concentration of 0.25mg/L. Cobalt salt acts as a catalyst for Na_2SO_3 and will be added to the water first. Before adding the salt to the test tank it will be dissolved in water to prevent big particles (ASCE, 2007). The Na_2SO_3 will also be dissolved in water before adding it evenly to the test tank. Na_2SO_3 deoxygenates the water by taking up oxygen molecules (Larson et al., 2007)

Reference table for saturated concentration of dissolved Oxygen in water at different temperatures

http://www.fao.org/docrep/field/003/ac183e/ac183e04.htm

Physical installation, enclosure, pole

===

Pole for solar panel & battery: Jon made drawing (APINUN)

Cable ties/iron bolts

Thermopol

Enclosure for board

Probes should be 60 cm below surface

IDEA: build a platform (GABRIEL, deadline: wednesday)

Physical security
==

There might be kids playing there? Open battery?

How often do we need to measure? จะวัดบ่อยแค่ไหน?
===

every 15 minutes

How many nodes do we need? จะใช้โหนดทั้งหมดกี่โหนด?

Do we have electricity? มีไฟฟ้าหรือไม่?

Nope! sure=?

TIMEPLAN
==

Wednesday

8.30 Lab
8.40 Site survery at pond
9.30 back at lab
split into teams
Team 1/ Calibration (Tatchai, Jintanan, Sebastian)
Team 2/ Code (Chavalit, Sethavidh)
Team 3/ Build the enclosure (Gabriel)
Team 4/ Network (Far@n)

What sensors do we have available in the lab? Add data sheet here!

What means of communication do we have? WiFi? How strong is the signal?
WiFi is confirmed. Tested during the morning of 17 Dec.
SSID: WSN
Key: not shown here for security reasons.

Where are we measuring? Please add GPS position.

Can you find relevant scientific papers/documentation?

Add the link to the thingspeak graphs here!

=============================

Coding issues:

- Retry Wifi associate if disconnected
- What to do with data collected during network disconnection

Fields:

1: Water Temperature

2: pH

3: D.O.

4: Battery

5: RSSI

6: Timestamp (in case connection to ThingSpeak is lost.)

===

Issue running DO and temp probes on one board

===

the DO output is about 0 .. 40 mV,

so in order to get reasonable accuracy with a

10-bit ADC,

we have to put the analog ref voltage to 1.1 V -

that gives us about 1 point / mV resolution,

i.e. about +- 2.5 % accuracy for the DO.

The temp probe outputs a maximum of 3000 mV.

If we run this on 1.1 V, we go into saturation (1023 points) at 33 celsius.

So we can not run the DO and the temp probe on one ref voltage.

We need two different analogReference voltages for thw two probes.

Possible solutions:

1) optimally, add a 12-bit ADC for the DO probe

2) for now, run on two different boards.

Atlas EZO pH Sensor

## Full Sample Code Here: https://www.atlas-scientific.com/_files/code/ino_files/EZO_ph_sample_code.zip

1.) On power-up will not show the correct pH for up to ten readings

2.) Turning off the sensor by sinking VCC to ground will cause incorrect readings for up to 5 minutes

3.) To properly shut off pH circuit, for hibernation, cut the ground line.

4.) The EZO pH sensor is accessed via Software Serial.

5.) On power-up, the first command sent to the pH circuit will result in an error

6.) Send a blank character to clear the serial buffer

When sending commands to the pH sensor:

1.) C = Enable or disable continuous readings (enabled by default)

2.) R = Return a single reading

3.) Response = Enable or disable response codes (enabled by default)

4.) T = Set or query the temperature compensation (set to 25C by default) (T,N<CR> where N=temp reading)

5.) X = Factory Reset

==============================

Code Samples:

```
#include <SoftwareSerial.h>      //we have to include the SoftwareSerial library, or else we can't
use it.

#define rx 2                     //define what pin rx is going to be.

#define tx 3                     //define what pin Tx is going to be.

SoftwareSerial myserial(rx, tx); //define how the soft serial port is going to work.

char ph_data[20];                  //we make a 20 byte character array to hold incoming data from
the pH.

char computerdata[20];             //we make a 20 byte character array to hold incoming data from a
pc/mac/other.

byte received_from_computer=0;     //we need to know how many characters have been received.

byte received_from_sensor=0;       //we need to know how many characters have been received.

byte arduino_only=0;               //if you would like to operate the pH Circuit with the Arduino
only and not use the serial monitor to send it commands set this to 1. The data will still come out
on the serial monitor, so you can see it working.

byte startup=0;                    //used to make sure the Arduino takes over control of the pH
Circuit properly.

float ph=0;                        //used to hold a floating point number that is the pH.

byte string_received=0;            //used to identify when we have received a string from the pH
circuit.


 void setup(){

     Serial.begin(9600);          //enable the hardware serial port

     myserial.begin(9600);        //enable the software serial port

      }


 void serialEvent(){               //this interrupt will trigger when the data coming from the
serial monitor(pc/mac/other) is received.

        if(arduino_only!=1){       //if Arduino_only does not equal 1 this function will be
bypassed.

           received_from_computer=Serial.readBytesUntil(13,computerdata,20); //we read the data sent
from the serial monitor(pc/mac/other) until we see a <CR>. We also count how many characters have
been received.

           computerdata[received_from_computer]=0; //we add a 0 to the spot in the array just after
the last character we received.. This will stop us from transmitting incorrect data that may have
been left in the buffer.

           myserial.print(computerdata);           //we transmit the data received from the serial
monitor(pc/mac/other) through the soft serial port to the pH Circuit.

           myserial.print('\r');                   //all data sent to the pH Circuit must end with a
<CR>.

          }

        }


 void loop(){


  if(myserial.available() > 0){        //if we see that the pH Circuit has sent a character.

     received_from_sensor=myserial.readBytesUntil(13,ph_data,20); //we read the data sent from pH
Circuit until we see a <CR>. We also count how many character have been received.

     ph_data[received_from_sensor]=0;  //we add a 0 to the spot in the array just after the last
character we received. This will stop us from transmitting incorrect data that may have been left in
the buffer.

     string_received=1;                //a flag used when the Arduino is controlling the pH Circuit
to let us know that a complete string has been received.

     Serial.println(ph_data);          //lets transmit that data received from the pH Circuit to the
serial monitor.

     }
  if(arduino_only==1){Arduino_Control();} //If the var arduino_only is set to one we will call this
function. Letting the Arduino take over control of the pH Circuit


 }



void Arduino_Control(){


      if(startup==0){                //if the Arduino just booted up, we need to set some things up
first.
          myserial.print("c,0\r");   //take the pH Circuit out of continues mode.
          delay(50);                 //on start up sometimes the first command is missed.
          myserial.print("c,0\r");   //so, let’s send it twice.
          delay(50);                 //a short delay after the pH Circuit was taken out of continues
mode is used to make sure we don’t over load it with commands.
          startup=1;                 //startup is completed, let's not do this again during normal
operation.
      }




  delay(800);                         //we will take a reading ever 800ms. You can make this much
longer or shorter if you like.
   myserial.print("R\r");             //send it the command to take a single reading.
   if(string_received==1){            //did we get data back from the ph Circuit?
     ph=atof(ph_data);                //many people ask us "how do I convert a string into a float?"
This is how...
     if(ph>=7.5){Serial.println("high\r");} //This is the proof that it has been converted into a
float.
     if(ph<7.5){Serial.println("low\r");}   //This is the proof that it has been converted into a
float.
     string_received=0;}             //reset the string received flag.



}
===============================================================
This is the code for DO and temperature
===============================================================
/*
Sebastian 20141217
Read DO and temp and send to Thingspeak
NOT TESTED YET - esp check use of the serial
 */
 #include <WiFi.h>
 #include <SPI.h>


 // ThingSpeak Settings
char thingSpeakAddress[] = "203.159.0.30";  //TS
String writeAPIKey = "YL684THYMQLPNITA";    //TS
const int updateThingSpeakInterval = 60 * 1000;      // Time interval in milliseconds to update
ThingSpeak (number of seconds * 1000 = interval)
```

```
char ssid[] = "WSN_1";     //  your network SSID (name)
char pass[] = "wsn@2014";  // your network password
int status = WL_IDLE_STATUS;     // the Wifi radio's status
// initialize the library instance:
WiFiClient client;
// Variable Setup
long lastConnectionTime = 0;
boolean lastConnected = false;
int failedCounter = 0;
int a;
int B=3975;
float ctemp;
float res;
// Sensor stuff
float temp;                //where the final temperature data is stored
////////////////////////////////////////////////////////////////////////
// setup //////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
void setup() {
  Serial.begin(9600);
// set voltage reference to 1.1 volts
  analogReference(INTERNAL);
//OUTPUT pin f. temp sensor
pinMode(2, OUTPUT);        //set pin 2 as an output
// WiFI stuff

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi shield not present");
    // don't continue:
    while(true);
  }

 // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network:
    status = WiFi.begin(ssid, pass);
    // wait 10 seconds for connection:
    delay(10000);
  }

  // you're connected now, so print out the data:
  Serial.print("You're connected to the network");
  printCurrentNet();
  printWifiData();
}
```

```
///////////////////////////////////////////////////////////////////////
// loop //////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////
void loop() {
int i=0;
float v_sum=0;
float v_avg=0;
float sat_avg=0;
String DO;
String temperature;
do
{
  delay(100);          // wait for sensors to stabilize
  int sensorValue = analogRead(A1);
  float voltage = sensorValue * (1100.0/1023.0);
  v_sum=v_sum+voltage;
i++;
} while (i < 100);
delay(5000);
v_avg=v_sum/100;
sat_avg=v_avg/28.5*100;
DO = String((int)(sat_avg),DEC);
  Serial.println("Voltage / Saturation average");
  Serial.print(v_avg);Serial.print("mV / ");Serial.print(DO);Serial.println("% ");
temp = read_temp();       //call the function “read_temp” and return the temperature in C°
Serial.println(temp);     //print the temperature data
temperature = String((int)(temp),DEC);
// WiFI & Thingspeak
  // check the network connection once every 10 seconds:
  a=analogRead(0);
  res=(float)(1023-a)*10000/a;
  ctemp=1/(log(res/10000)/B+1/298.15)-273.15;
  String analogPin0 = String((int)(ctemp*100), DEC);
 // Print Update Response to Serial Monitor
  if (client.available())
  {
    char c = client.read();
    Serial.print(c);
  }
  // Disconnect from ThingSpeak
  if (!client.connected() && lastConnected)
  {
    Serial.println("...disconnected");
    Serial.println();

    client.stop();
  }
  // Update ThingSpeak
  if(!client.connected() && (millis() - lastConnectionTime > updateThingSpeakInterval))
  {
    updateThingSpeak("field1="+temperature);
     Serial.print(analogPin0);
     Serial.write(186);
     Serial.println("C");
     printCurrentNet();
    updateThingSpeak("field2="+DO);
     Serial.print(analogPin0);
     Serial.write(186);
     Serial.println("C");
     printCurrentNet();
  }

  lastConnected = client.connected();

}
/////////////////////////////////////////////////////////////////////////
// Here be Functions ////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////
float read_temp(void){   //the read temperature function
float v_out;             //voltage output from temp sensor
float temp;              //the final temperature is stored here
digitalWrite(A0, LOW);   //set pull-up on analog pin
digitalWrite(2, HIGH);   //set pin 2 high, this will turn on temp sensor
delay(2);                //wait 2 ms for temp to stabilize
v_out = analogRead(0);   //read the input pin
digitalWrite(2, LOW);    //set pin 2 low, this will turn off temp sensor
v_out*=.0011;            //convert ADC points to volts (we are using .0048 because this device is
running at 5 volts)
v_out*=1000;             //convert volts to millivolts
temp= 0.0512 * v_out -20.5128; //the equation from millivolts to temperature
return temp;             //send back the temp
}
void updateThingSpeak(String tsData)
{
  if (client.connect(thingSpeakAddress, 3000))
  {
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak:3000\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+writeAPIKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(tsData.length());
    client.print("\n\n");
    client.print(tsData);
    lastConnectionTime = millis();

    if (client.connected())
    {
      Serial.println("Connecting to ThingSpeak...");
      Serial.println();

      failedCounter = 0;
    }
    else
    {
      failedCounter++;

      Serial.println("Connection to ThingSpeak failed ("+String(failedCounter, DEC)+")");
      Serial.println();
    }

  }
  else
  {
    failedCounter++;

    Serial.println("Connection to ThingSpeak Failed ("+String(failedCounter, DEC)+")");
    Serial.println();

    lastConnectionTime = millis();
  }
}
void printWifiData() {
  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  Serial.println(ip);

  // print your MAC address:
  byte mac[6];
  WiFi.macAddress(mac);
  Serial.print("MAC address: ");
  Serial.print(mac[5],HEX);
  Serial.print(":");
  Serial.print(mac[4],HEX);
  Serial.print(":");
  Serial.print(mac[3],HEX);
  Serial.print(":");
  Serial.print(mac[2],HEX);
  Serial.print(":");
  Serial.print(mac[1],HEX);
  Serial.print(":");
  Serial.println(mac[0],HEX);

}
void printCurrentNet() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  // print the MAC address of the router you're attached to:
  byte bssid[6];
  WiFi.BSSID(bssid);
  Serial.print("BSSID: ");
  Serial.print(bssid[5],HEX);
  Serial.print(":");
  Serial.print(bssid[4],HEX);
  Serial.print(":");
  Serial.print(bssid[3],HEX);
  Serial.print(":");
  Serial.print(bssid[2],HEX);
  Serial.print(":");
  Serial.print(bssid[1],HEX);
  Serial.print(":");
  Serial.println(bssid[0],HEX);
  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.println(rssi);
  // print the encryption type:
  byte encryption = WiFi.encryptionType();
  Serial.print("Encryption Type:");
  Serial.println(encryption,HEX);
  Serial.println();
}
```