RIZAL TECHNOLOGICAL UNIVERSITY COLLEGE OF ENGINEERING

Smart Water Quality Management for Tank-Based Cream Dory Farming: Boosting Productivity and Sustainability Through Automation

User Operation Manual

Developed by: KMJ's

Table of Contents

Aquaculture System Overview

This aquaculture system uses the setup for tank-based cream dory farming, which is designed to maintain stable and healthy water conditions through a combination of controlled water flow, automated monitoring, and efficient filtration. This system includes a primary tank for the enclosure of the cream dory connected to three 26-liter stock boxes arranged in a gravity-driven flow sequence serving as the Recirculating Aquaculture System (RAS) in this setup. This allows water to move continuously through mechanical, biological, and chemical filtration stages before returning clean and oxygen-rich water back to the fish tank.

Smart Water Quality Management Overview

The Smart Water Quality Management System serves as the core technology that keeps the cream dory tank stable, productive, and easier to manage. Instead of relying on manual/traditional checking and guesswork, the system continuously monitors the critical water parameters that directly affect fish growth, such as temperature, pH, dissolved oxygen, ammonia levels, and salinity. Passing the set thresholds for the parameters triggers the system to power up the pump, filling up the top part of RAS with water from the tank. Sensors are placed at strategic points within the tank to provide accurate and precise real-time readings of the water quality. This system is equipped with ESP32 as the main microcontroller, which serves as the system's central processor, receiving and analyzing data collected from the sensors installed within the tank environment.

Materials Specifications

Material	Picture	Specification
ESP32 Wifi Module		• Dual-core Tensilica Xtensa LX6 processors • Clock speed: up to 240 MHz • 32-bit architecture • 448 KB ROM • 520 KB SRAM • Optional external flash: 4 MB to 16 MB It supports Wi-Fi (802.11 b/g/n) and Bluetooth 4.2 (Classic & BLE) for wireless connectivity. The chip

		includes up to 34 GPIO pins, ADC, DAC, I²C, SPI, UART, PWM, and built-in touch and hall sensors. It operates on 2.3–3.6 V
TDS and Temperature Sensor		• Parameters Measured: pH, TDS, Temperature • TDS Range: 0 – 5000 ppm • Temperature Sensor: NTC 10K Thermistor • Connection: 3/8” fast-connection pipe (likely for inline probe mounting) • Display: LCD (as stated “LCD pH Analyzer Probe”)

pH Sensor		● Supply Voltage: 5 V DC ● Operating Current: ~10 mA ● Analog Output: Yes (proportional to pH) ● pH Measurement Range: 0 – 14 pH ● Response Time: ~5 seconds E201-BNC Electrode Connector: BNC + ~1 m cable
Dissolved Oxygen Sensor		Operating Voltage: 3.3 V to 5.5 V Probe Type: Galvanic dissolved oxygen probe Measurement (Detection) Range: 0 to 20 mg/L DO Response Time: Up to 98% of full response within ~90

		seconds (at 25 °C) Pressure Range: 0 to 50 PSI Maintenance: ● Membrane cap replacement: every 1–2 months in muddy water; 4–5 months in clean water ● Filling solution (electrolyte) replacement: once a month
Ammonia Sensor		Sensor Type: Semiconductor (SnO_2) Target Gas: Ammonia (NH_3) Detection (Concentration) Range: ~5 – 500 ppm NH_3 Loop / Circuit Voltage (Vc): ≤ 24 V DC Heater Voltage

		(VH): ~5.0 V ± 0.1 V (AC or DC) Sensing (Gas) Resistance (Rs): ~2 kΩ – 15 kΩ (in ~50 ppm NH_3) Sensitivity: Rs(in air) / Rs(50 ppm NH_3) ≥ 2
Water Level Sensor		Switch Power (Max): 10 W Max Switching Voltage: 100 V Max Switching Current: 0.5 A Breakdown Voltage: 220 V DC max Carry Current (Max): 1 A Contact Resistance: ~100 milliohms (0.1 Ω) Switch Type / Behavior:

		• Horizontally mounted float switch • Can be configured as Normally-Open (NO) or Normally-Closed (NC) by rotating the float 180°. Internal Mechanism: Magnet in the float activates a sealed reed relay inside.
Water Pump		A3000 Power: **16 W** Voltage: **220–240 V** Frequency: **50 / 60 Hz** Maximum Flow (F.Max): **3000 L/h** Maximum Head (H.Max): **2 m**

Aerator		Model: Q7-HQ-902 Power: 8 W Flow / Output: 4 × 4000 ml/min (i.e., 4 outlets) Maximum Air Volume: ~ 320 GPH according to manufacturer Pressure: 30 kPa Maximum Water Depth: < 1.2 m Noise Level: < 40 dB
Solenoid Valve		Brass Electric Solenoid Valve (2-Way, 12 V DC / 220 V AC, ½″) • Valve Type: 2-way, normally closed (N/C) — power on → valve opens; power off → valve closes • Body Material: Brass

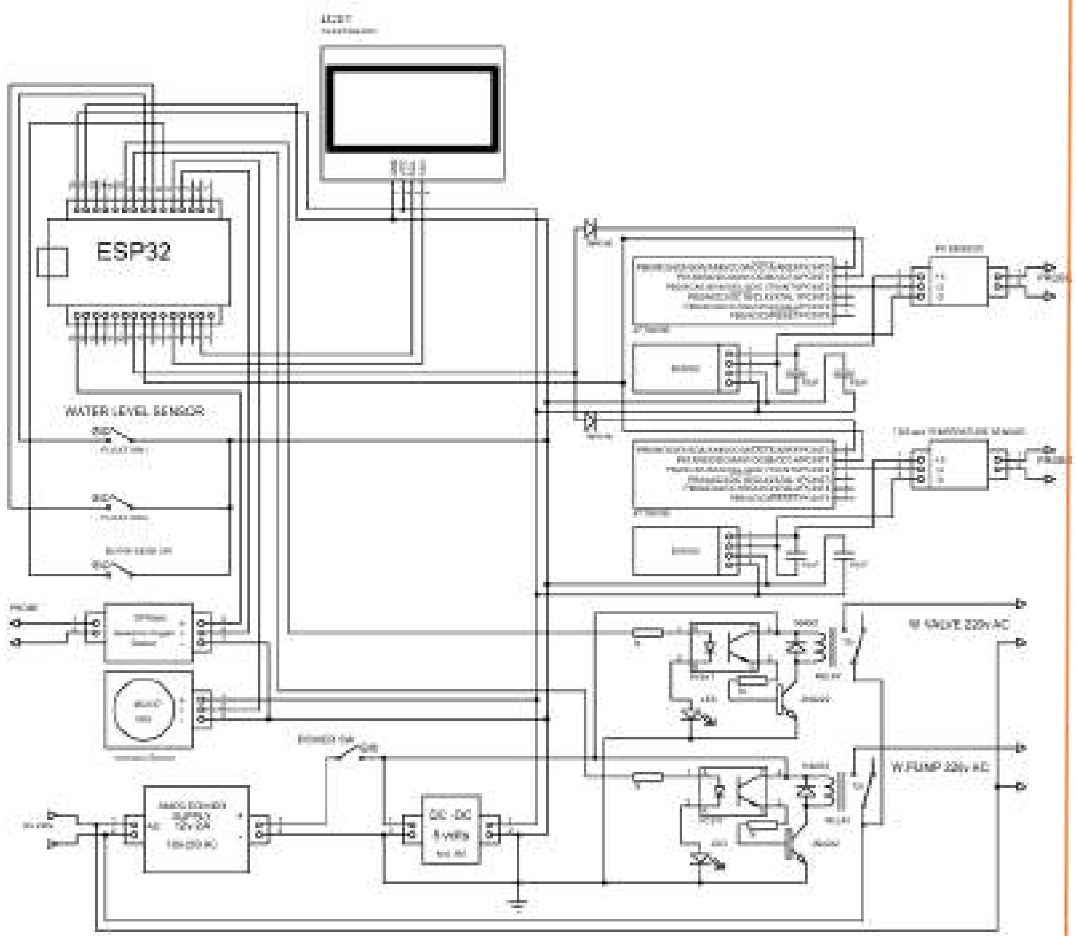
		• Seal / Plunger Material: Commonly NBR, EPDM or Viton / FKM depending on model • Media (Fluid Compatibility): Water, air, oil, gas (non-corrosive)
Stock Box 26L		Recirculating Aquaculture System

LCD Display		**OLED 128x64 I2C display** **Display Type**: OLED **Resolution**: 128 × 64 pixels **Display Color**: Monochrome**Size / Diagonal**: ~0.96 inches **Controller IC**: Often SSD1306 **Input / Logic Voltage**: Typically **3.3 V**, many modules are 5 V-tolerant **Power Consumption**: Very low — often around **~0.06 W** to **~0.08 W**

Modem		**LTE 4G USB Modem with WiFi Hotspot** **Cellular connectivity**: Supports 4G LTE (often LTE Category 4) **SIM card slot**: Accepts a standard or nano SIM **Download / Upload speeds** (cellular): Typical 4G modems support up to **≈ 150 Mbps download / ≈ 50 Mbps upload** under ideal LTE Cat-4 conditions. **WiFi hotspot**: The modem acts as a WiFi access point **Power / Interface**: Powered over USB (5 V), or via a USB charger / power bank / laptop USB port; many are “plug-and-play” with no

		complex setup. **Portability / Compactness**: Small "stick" or dongle form factor **Network standard coverage**: 4G (FDD-LTE / TDD-LTE), 3G (UMTS / HSPA+ / WCDMA), and older 2G (GSM / EDGE / GPRS) depending on region and device.

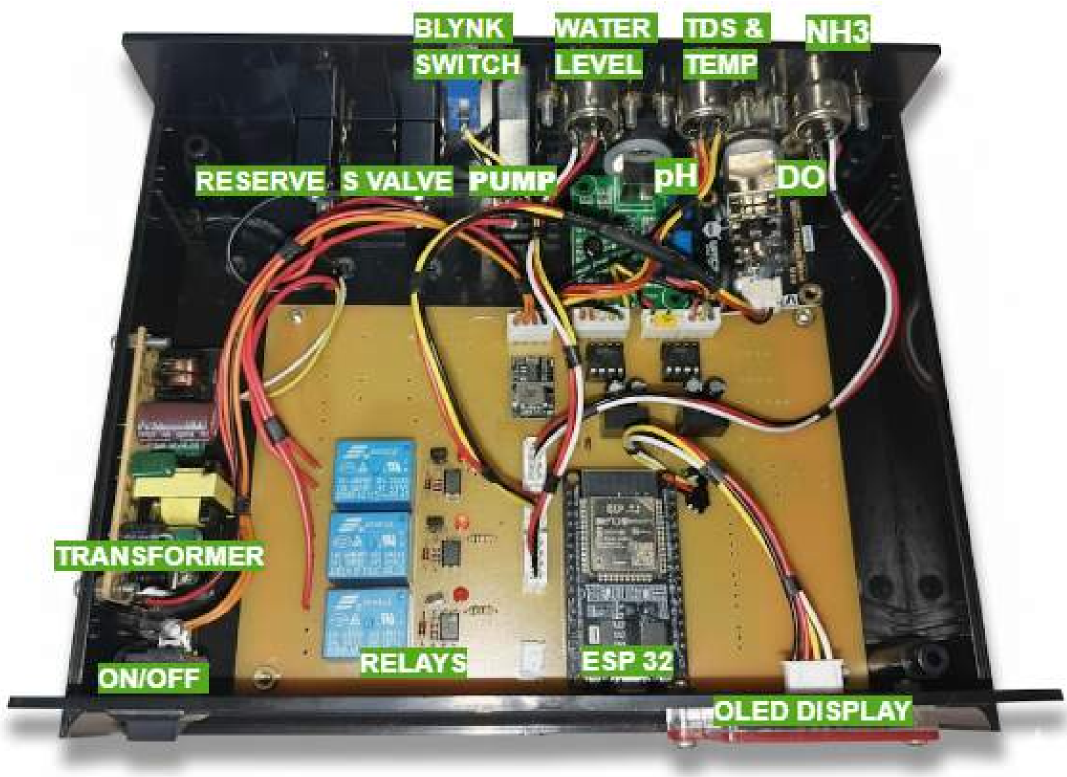
SWQM Assembly and Schematic Diagram



SWQM Schematic Diagram

Development of smart water quality management system

The development of a Smart Water Quality Management System integrates sensors for pH, dissolved oxygen (DO), temperature, salinity, ammonia, and water level to continuously monitor water conditions and automate corrective actions, ensuring optimal aquaculture performance and fish health. Alongside this development is the components layout, which illustrates the arrangement and labeling of the parts used in the system providing a clear guide for installation, maintenance, and troubleshooting.



Components layout

Construction and Assembly Procedures

1. Install sensors into the system according to the ESP32 pins.
2. Install actuators such as the submersible pump, air pump, and solenoid valves according to the system's outlet.
3. Microcontroller and wiring: Place the ESP32 and relay module in waterproof enclosures above the water level. Ensure secure wiring connections to all sensors and actuators.
4. Deploy the RAS properly with its filtration system.
5. Connect the pump to the first box of the RAS.
6. Place the sensor inside the tank.
7. Set the aerator hose to cover the area of the tank
8. Plug the modem into your outlet
9. Power up the device

Main Source Code

```
// GMAIL:tankbasedcreamdoryfarming@gmail.com
// GMAIL PW:rtuececreamdory
// BLYNK:tankbasedcreamdoryfarming@gmail.com
// BLYNK PW:Oct092025
// Date Oct.12 2025 8:12PM

#define BLYNK_TEMPLATE_ID "TMPL6UjcH9omd"
#define BLYNK_TEMPLATE_NAME "ESP32 Tank Based Farming"
```

```
#define BLYNK_AUTH_TOKEN "dsDoc5gd17Ij5cy0bNYoEr9PRZXJAPuQ"
#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = BLYNK_AUTH_TOKEN;

char ssid[] = "4G-UFI-542";
char pass[] = "1234567890";

BlynkTimer timer;
WidgetTerminal terminal(V0);


#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH1106.h>
//#include <EEPROM.h>
#include <SoftwareSerial.h>
//#include <RTClib.h>

HardwareSerial Hserial2(2);

#define OLED_SDA 21
#define OLED_SCL 22
Adafruit_SH1106 display(21, 22);
#if (SH1106_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SH1106.h!");
#endif
```

```
/////////////////// DO
#define DO_PIN 35
#define VREF 3300    //VREF (mv)
#define ADC_RES 4096 //ADC Resolution
#define READ_TEMP (25) //Current water temperature ℃, Or temperature
sensor function
//Single point calibration needs to be filled CAL1_V and CAL1_T
#define CAL1_V (750) //mv 1600
#define CAL1_T (25)   //℃

const uint16_t DO_Table[41] = {
    14460, 14220, 13820, 13440, 13090, 12740, 12420, 12110, 11810, 11530,
    11260, 11010, 10770, 10530, 10300, 10080, 9860, 9660, 9460, 9270,
    9080, 8900, 8730, 8570, 8410, 8250, 8110, 7960, 7820, 7690,
    7560, 7430, 7300, 7180, 7070, 6950, 6840, 6730, 6630, 6530, 6410};

  uint8_t Temperaturet;
  uint16_t ADC_Raw;
  uint16_t ADC_Voltage;
  uint16_t DO;

  int16_t readDO(uint32_t voltage_mv, uint8_t temperature_c){
   uint16_t V_saturation = (uint32_t)CAL1_V + (uint32_t)35 * temperature_c -
(uint32_t)CAL1_T * 35;
  return (voltage_mv * DO_Table[temperature_c] / V_saturation);
  }
/////////////////// DO



/// SENSORS OUTPUT
float   ph_val    = 0;
float   temp_val  = 0;
```

```
int     tds_val   = 0;
int     ntu_val   = 0; ///// not used
int     do_val    = 0; ///// not used

int     mq3_val = 0;
int     nh3ppm  = 0;
int     data_in = 0;
int     map_min = 0;

const int led          = 2;   // led

int waterlevel         = 0;  // water level selector
const int flt1Pin      = 25; //
const int flt2Pin      = 33; //
const int blynkPin     = 32; // pin send mute
const int mq3_input    = 34; //

int       mq_timer    = 0;

boolean    Sauto       = false;
boolean    Snotif      = false;

const int rly1       = 13; // water pump
const int rly2       = 26; // water valve
const int rly3       = 27; // reserve

boolean    Srly1    = false; // wpump
boolean    Srly2    = false; // water valve
boolean    Srly3    = false; // reserve



// Sensor REF
```

```
int      RH_ph       = 8  ;      // greater
int      RL_ph       = 5.5;      // less
int      RH_nh3      = 10;       // greater
int      RH_tds      = 400;      // greater
int      RL_do       = 0;        // less monitor only

// sensor TRIGGERED

boolean    Sph    = false;
boolean    Snh3   = false;
boolean    Stds   = false;


String Rblynkstring;
int    Rblynkint   = 0;

String Srxdata;
String Srx1;
String Srx2;
uint16_t Tdly1 = 0; ///

/// SENSOR COUNTER
int      Cnc1 = 0;
int      Cnc2 = 0;
int      Cnc3 = 0;
int      Cnc4 = 0;
int      Cnc5 = 0;


int      Acount = 70;


unsigned long time_now = 0; /// time delay millis
```

```
void setup(){
   Serial.begin(115200);   // Start the serial terminal
   Serial.println("chancom electronics");
 //   EEPROM.begin(512);  //
   Hserial2.begin(9600, SERIAL_8N1, 16,17 );
  //  Hserial2.println("chancom electronics");
   delay (100);
   pinMode(led, OUTPUT);
   pinMode(rly1, OUTPUT);
   pinMode(rly2, OUTPUT);
   pinMode(rly3, OUTPUT);
   pinMode(flt1Pin,INPUT_PULLUP);
   pinMode(flt2Pin,INPUT_PULLUP);
   pinMode(blynkPin,INPUT_PULLUP);
   delay (1000);
   Initialize_LCD();
   delay (100);
   Blynk.begin(auth, ssid, pass);
   timer.setInterval(10000L, Blynk_send); //

   mq_timer  = 60;  //
   delay (1000);
   }



void loop() {
   if(millis() > time_now + 1000){
   time_now = millis();
   digitalWrite(led, !digitalRead(led));  ///// test
   pulse();
   get_wlevel(); /// float sensor
   get_mq137();  /// ammonia
```

```
    get_ph();     /// ph & ntu
    get_tds();    /// tds & temp
    get_do();
    disp_all();   /// get time
    test_auto();
    }
    test_Sout();
    Blynk.run();
    timer.run(); // Initiates SimpleTimer
    }

void pulse(){
    if (Acount !=0){
    Acount --;
    }
    if (Acount == 10){
    Sauto = true;
    }
    }

void test_Sout(){

      if (Sauto == true){  //////////////////////////////////
      if ((Snh3 == true)||(Sph == true)||(Stds == true)){
      digitalWrite(rly1,HIGH);
      Srly1 = true;
      }else{
      digitalWrite(rly1,LOW);
      Srly1 = false;
      }
      }                  ////////////////////////////////////
      }
```

```
void Initialize_LCD(){
   // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
  display.begin(SH1106_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3D (for the 128x64)
  // init done
  display.clearDisplay();
  // text display tests
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(30,5);
  display.print("Development");
  display.setCursor(55,20);
  display.print("of");
  display.setCursor(10,35);
  display.print("Tank-Based Cream");
  display.setCursor(20,50);
  display.print("Dory Farming");
  display.display();
  delay(5000);
  }

void disp_all(){
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0,0);
    ///
    display.setCursor(0,0);
    display.print("NH3 :");
    if (mq_timer != 0){
    display.print("Calibrating..");
```

```
  }else{
  display.print(mq3_val);  /////////
  display.print("mv / ");
  display.print(nh3ppm);  /////////
  display.print(" ppm");
  }
  ///
  display.setCursor(0,10);
  display.print("PH  :");
  display.print(ph_val,1);  /////////
  ///
  display.setCursor(0,20);
  display.print("TEMP:");
  display.print(temp_val,1);
  display.print((char)247); // degree symbol
  display.print("C");
  ///
  display.setCursor(0,30);
  display.print("TDS :");
  display.print(tds_val);
  display.print(" ppm");
  ///
  display.setCursor(0,40);
  display.print("DO  :");
  display.print(do_val);
  display.print(" mg/L");
//  display.print("NTU :");
//  display.print(ntu_val);
  ///
  display.setCursor(0,50);
  display.print("WL  :");
  display.print(waterlevel);
  ///
```

```
    display.setCursor(55,55);
    if (Sauto == true){
    display.print("AUTO");
    }
    display.setCursor(85,55);
    if (Srly1 == true){
    display.print("PUMP");
    }
    display.display();// dont forget this line at the end
    }

void get_wlevel(){
    int flt1_ = digitalRead(flt1Pin);
    int flt2_ = digitalRead(flt2Pin);
    if ((flt1_ == LOW)&&(flt2_ == LOW)){
    waterlevel = 0; ///
    }
    else if ((flt1_ == HIGH)&&(flt2_ == LOW)||(flt1_ == LOW)&&(flt2_ == HIGH)){
    waterlevel = 50; ///
    }
    else if ((flt1_ == HIGH)&&(flt2_ == HIGH)){
    waterlevel = 100; ///
    }
    }

void get_mq137(){
    get_input();
    mq_isready();
    mq_nowready();
    }

void get_input(){
  uint32_t data_in_ = 0;
```

```
  for (int i=0; i < 800; i++) {
  data_in = analogRead(mq3_input);
  data_in_ = data_in_ + data_in;
  }
  data_in = data_in_ / 800;
 // Serial.print("Data ");
 // Serial.println(data_in);
  }

void mq_isready(){
  if (mq_timer != 0){
  mq_timer --;
  map_min = data_in;  //// save calibrated data
  }
  }
void mq_nowready(){
  if (mq_timer != 0){
  Serial.println("calibrating....");
  return;
  }

  mq3_val =  data_in - map_min;
  if (mq3_val < 1 ){
  mq3_val = 0;
  }
  Serial.print("millivolt ");
  Serial.println(mq3_val);
  nh3ppm = map(mq3_val, 0,1242 , 0,50);

  Serial.print("ppm ");
  Serial.println(nh3ppm);
  }
```

```
void get_ph(){
 Hserial2.println('B');
 delay (10);
 if (Hserial2.available()) {
 Srxdata = Hserial2.readStringUntil('\r'); //n
// Serial.println(Srxdata);
 sel_string();

   ntu_val  = Srx1.toInt();
   if (ntu_val < 0){
   ntu_val = 0;
   }
   ph_val   = Srx2.toFloat();
   }
   }

void get_tds(){
 Hserial2.println('A');
 delay (10);
 if (Hserial2.available()) {
 Srxdata = Hserial2.readStringUntil('\r'); // r
// Serial.println( Srxdata );
 sel_string();
 tds_val  = Srx1.toInt();
 temp_val = Srx2.toFloat();
 }
 }

void get_do(){
  Temperaturet = (uint8_t)READ_TEMP;
  ADC_Raw = analogRead(DO_PIN);
  ADC_Voltage = uint32_t(VREF) * ADC_Raw / ADC_RES;
  do_val = ((readDO(ADC_Voltage, Temperaturet))/1000);
```

```
//  Serial.print (do_val);
//  Serial.println (" mg/L");
  }

void sel_string(){
       int  stdex;
       int  stdexe;
       Srx1  =  Srxdata.substring(stdex,stdexe);
       stdex++;
       Srx2   =  Srxdata.substring(stdex,stdexe);
       Srxdata = "";
     //  Serial.println(Srx1);
     //  Serial.println(Srx2);
       }
void test_auto(){
  if (Sauto == false){
   return;
   }
   test_waterlevel();
   test_nh3();
   test_ph();
   test_tds();
 //  test_do();   // monitor only
   }

void test_waterlevel(){
    if (waterlevel == 0 ){
    Cnc1 ++;
    if (Cnc1 > 4){
    Cnc1 = 0;
    notification();
    Serial.println("WATERLEVEL TRIGGER");
    delay (1000);
```

```
    Blynk_send();
    digitalWrite(rly2,HIGH);  //// solinoid valve on
    }
    }else{
    Cnc1 = 0;
    digitalWrite(rly2,LOW);  //// solinoid valve on
    }
    Serial.println(Cnc1);
    }

void test_nh3(){
    if (nh3ppm > RH_nh3 ){
    Cnc2 ++;
    if (Cnc2 > 4){
    Cnc2 = 0;
    Serial.println("NH3 TRIGGER");
    notification();
    Snh3 = true;
    delay (1000);
    Blynk_send();
    }
    }else{
    Cnc2 = 0;
    Snh3 = false;
    }
    Serial.println(Cnc2);
    }

void test_ph(){
    if ((ph_val > RH_ph )||(ph_val < RL_ph )){
    Cnc3 ++;
    if (Cnc3 > 4){
    Cnc3 = 0;
```

```
    Serial.println("PH TRIGGER");
    notification();
    Sph = true;
    delay (1000);
    Blynk_send();
    }
    }else{
    Cnc3 = 0;
    Sph = false;
    }
    Serial.println(Cnc3);
    }

void test_tds(){
    if (tds_val > RH_tds ){
    Cnc4 ++;
    if (Cnc4 > 4){
    Cnc4 = 0;
    Serial.println("TDS TRIGGER");
    notification();
    Stds = true;
    delay (1000);
    Blynk_send();
    }
    }else{
    Cnc4 = 0;
    Stds = false;
    }
    Serial.println(Cnc4);
    }

void test_do(){
    if (do_val < RL_do){
```

```
    Cnc5 ++;
    if (Cnc5 > 4){
    Cnc5 = 0;
    notification();
    Serial.println("DO TRIGGER");
    delay (1000);
    Blynk_send();
    }
    }else{
    Cnc5 = 0;
    }
    Serial.println(Cnc5);
    }

void notification(){
    if (Snotif == true){
    return;
    }
    Serial.println("notify");
        Serial.println ("notification","Please take action to maintain optimal conditions");
    Snotif = true;
    }

///////////// BLYNK
   BLYNK_WRITE(V7){            // auto
   Rblynkint = param.asInt();
   Serial.println(Rblynkint);
   if (Rblynkint != 1){
   return;
   }
   if (Sauto == false){
   Sauto  = true; /// auto on
```

```
  }else{
  Sauto = false;
  }
  Snotif = false;
  Srly1  = false;
  Srly2  = false;
  digitalWrite(rly1,LOW);
  digitalWrite(rly2,LOW);
  Blynk_send();
  }

BLYNK_WRITE(V8){          // wpump
 Rblynkint = param.asInt();
 Serial.println(Rblynkint);
 if (Rblynkint != 1){
 return;
 }
 if (Srly1 == false){
 Srly1 = true;
 digitalWrite(rly1,HIGH);
 }else{
 Srly1 = false;
 digitalWrite(rly1,LOW);
 }
 Sauto = false;
 Blynk_send();
 }


void Blynk_send(){
  int blynk_ = digitalRead(blynkPin);
  if (blynk_ == HIGH){
  return;
```

```
}
Serial.println("blynk send");
Blynk.virtualWrite(V1,nh3ppm);      /////
Blynk.virtualWrite(V2,ph_val);
Blynk.virtualWrite(V3,temp_val);
Blynk.virtualWrite(V4,tds_val);
Blynk.virtualWrite(V5,do_val);
Blynk.virtualWrite(V6,waterlevel);
if (Sauto == true){
Blynk.virtualWrite(V9,255);   // int
}else{
Blynk.virtualWrite(V9,0);   // int
}
if (Srly1 == true){
Blynk.virtualWrite(V10,255);   // int
}else{
Blynk.virtualWrite(V10,0);   // int
}
}
```

Operating Procedures

1. Plug in the SWQM system into the outlet and then power up.
2. Plug in the 4G LTE Modem.
3. Powering up the system it will show its starting display. Flip the switch for the sending of notifications in the app.
4. Once it is connected to the modem it will now show the parameters(NH3, pH, Temp, TDS, DO, and Water Level)

5. Connect the water pump, solenoid valve to the systems' outlet. Plug in the air pump.
6. Fill the tank with water and arrange the RAS. Connect the water pump to the RAS
7. Upon measuring the parameters for water quality, it will now display its reading on the lcd display and in the mobile controller app (Blynk).
8. When the system detects a breach in the preset thresholds value for water quality, the microcontroller will automatically start the water pump initiating the RAS process. It will also send a notification to the app.
9. Once the water quality is now stable the SWQM power off the pump completing the process.
10. For manually powering up the water pump, you can press the W PUMP button in the app.
11. For water level monitoring, arrange the position of the sensor in the tank that matches the water level.
12. For a full tank it will display 100%, once it reaches 0% the SWQM automatically starts/powers up the solenoid valve filling up the tank. Upon reaching 100% or 50% it will automatically power off the solenoid valve.

Maintenance Procedure

Keeping the Smart Water Quality Management System in good condition is important to make sure it works smoothly and provides accurate readings for your tank. Regular maintenance helps avoid problems and keeps your cream dory healthier. Below are the essential maintenance practices that help keep the system operating smoothly and consistently.

1. **Visual Inspection**
    - Regularly inspect all sensors (pH, DO, temperature, ammonia, salinity, water level) for dirt, algae, or debris buildup.
    - Make sure sensor readings displayed on the OLED screen and dashboard appear normal and consistent with tank conditions.
    - Check pumps and aerators to ensure they are running smoothly.
2. **Sensor Care and Cleaning**
    - Clean pH, DO, and temperature probes gently using clean water and a soft cloth to remove biofilm and dirt.
    - Avoid using harsh chemicals on sensors unless recommended by the manufacturer.
    - Recalibrate sensors as needed, especially when readings become unstable or inaccurate.
3. **RAS and Tank Component Maintenance**
    - Inspect mechanical filters (Japanese mat, foam pads, fine filters) and biological media for clogging or buildup.

- Rinse mechanical filters using tank water—not tap water—to protect beneficial bacteria.
- Ensure the sludge collector, lava rocks, and K1 media are functioning properly and not obstructed.

4. **System Diagnostics and Connectivity**
    - Check that the LTE 4G USB modem and WiFi hotspot are connected and transmitting data properly.
    - Verify that the control board, power supply, and backup systems are operating without interruption.
    - Confirm that there are no loose connections, exposed wires, or signs of corrosion around the electronics.
5. **Monitoring Water Quality Trends**
    - Review system logs, alarms, or app notifications for unusual fluctuations in pH, temperature, DO, ammonia, or salinity.
    - Investigate any sudden changes immediately to prevent stress or mortality in fish.
    - Manually cross-check water parameters when needed to validate sensor accuracy.
6. **Replacement of Consumable Components**
    - Replace activated charcoal or chemical media once they are exhausted or no longer effective.

- Replace sensors reaching the end of their operating lifespan to maintain precise measurements adhering to the devices specifications.

7. **Technical Assistance from Researchers**

- The research team is available to provide technical support for troubleshooting, system adjustments, or performance evaluation.
- Users may contact the researchers for assistance with calibration, software updates, sensor replacements, and interpretation of system data.
- In cases of unusual sensor behavior, persistent water quality issues, or suspected system malfunctions, the research team can conduct onsite or remote diagnostics to help restore system functionality.
- Researchers may also assist in training new operators, ensuring proper system handling, and providing recommendations to optimize water quality and fish growth.

Safety Guidelines

To ensure safe and efficient operation of the Smart Water Quality Management System, users must follow these safety practices at all times.

These guidelines help protect the operator, the equipment, and the cream dory stock from harm or system failure.

1. **Electrical Safety**
   - Always ensure hands are dry before handling electrical components.
   - Avoid placing controllers, sensors, and modems near areas prone to splashing.
   - Use waterproof enclosures for all electronics.
   - Immediately replace damaged wires or exposed connectors.
2. **Sensor and Chemical Handling Safety**
   - Handle calibration solutions (pH buffers, ammonia test solutions) with gloves.
   - Rinse sensors only with clean water—avoid harsh cleaners.
3. **Mechanical Safety**
   - Turn off pumps and aerators before performing maintenance.
   - Avoid sticking hands near moving impellers or aeration devices.
   - Ensure proper lifting techniques when handling heavy filter media or equipment.
4. **Water Quality Safety**
   - Replace water gradually to avoid sudden parameter changes that can shock fish.
   - Check temperature, pH, and salinity before adding new water.

- Never rely solely on the automated system—always verify manually when abnormalities occur.

5. **General Farm Safety**

- Wear gloves and boots when handling fish or tank components.
- Ensure proper ventilation in enclosed tank areas.
- Do not allow untrained individuals to handle the system.

6. **Emergency Procedures**

- In case of system failure, switch to manual monitoring of pH, DO, temperature, and ammonia.
- Keep backup aerators and spare sensors readily available.
- Have emergency contact numbers listed (technician, supplier, farm manager).
- For electrical failure, disconnect main power and inspect only when safe.