**Control de calidad en una camaronera**

Programación de Sistemas Telemáticos

Winter Emanuel Álava Intriago; Cesar Antonio Coloma Guzmán

Dario Javier Erreyes Ordoñez; Jancarlo André Valencia Noboa

Facultad de Ingeniería en Electricidad y Computación

Escuela Superior Politécnica del Litoral

Resumen

El presente proyecto tiene como objetivo principal el control de calidad de una camaronera, específicamente de las piscinas de cultivo, mediante el diseño de un sistema de adquisición de datos que permite la medición y comparación de los mismos, con el fin de mantener la piscina siempre entre el rango deseado. Como resultado se obtuvo un dispositivo de control con dos sensores uno de temperatura y otro de pH, que son controlados mediante un NodeMCU que se conecta a un servidor que contiene una base de datos remota, con el fin de poder acceder a dicha base de datos desde cualquier dispositivo en cualquier zona sin importar que se encuentre fuera de la red de wifi local, para esta visualización de datos se creó un aplicativo, permitiendo así que el usuario pueda observar los datos de una forma mas sencilla.

Introducción

En la industria camaronera existen varios factores que afectan la calidad del producto final, debido a que para llegar hasta el consumidor la materia prima pasa por una serie de procesos de tratamiento, no obstante, la etapa de cultivo se considera la más importante de todo el proceso, ya que el camarón se desarrolla alrededor de 4 meses dentro de piscinas controladas hasta lograr las características deseadas por el proveedor [1].

Para el control de una piscina camaronera se requiere del censado de diferentes parámetros tales como temperatura, salinidad, pH, oxígeno disuelto, alcalinidad, entre otros. A continuación, se detallan los rangos aproximados de cada parámetro utilizados por las camaroneras para el cultivo de larvas en las piscinas [2].

Salinidad. - La salinidad es la cantidad de materia solida disuelta en un Kg de agua de mar y se mide en ppm, para el agua de mar la salinidad es de 35 ppm, no obstante, para un crecimiento optimo se requiere de un rango de 15 a 25 ppm.

Turbiezas. – Se describe como el grado de opacidad que es producido por las partículas de suspensión en el agua y para el cultivo se estima un rango optimo entre 30 y 40 Secchi.

Oxígeno disuelto. – Es la variable más importante en la cría del camarón comúnmente la falta de oxígeno es la principal causante de la mortalidad de los camarones en la piscina, este valor depende de la temperatura, presión atmosférica y la salinidad. En la cría de camarones se requiere de una concentración de O2 mayor a 3 ppm, cuando el valor es menor el metabolismo de los camarones baja, afectando directamente en su supervivencia y crecimiento.

pH- El agua con un pH de 6.5 hasta 9 es considerada óptima para el cultivo de camarones.

Temperatura. - La temperatura afecta en todos los procesos químicos y biológicos que se desarrollan en la piscina, por ende, este parámetro debe de ser mantenido en su rango optimo viene a ser aproximadamente 30 °C.

Para poder mantener todos estos parámetros dentro de sus rangos óptimos se requiere de un dispositivo de control de calidad del agua, por ende, en respuesta a esta necesidad se plantea el desarrollo de un dispositivo que se encarga de censar varios de estos valores cada cierto periodo de tiempo y a su vez controle que se encuentre en el rango deseado, además de esto el dispositivo funcionara a la par con una base de datos que pueda ser visualizada desde una aplicación Android.

Procedimiento Experimental

Para el correcto desarrollo del dispositivo de control de camaronera fue necesario diseñar un diagrama de flujo, con el fin de detallar desde un principio todas las acciones que podrá tener el usuario dentro del aplicativo.

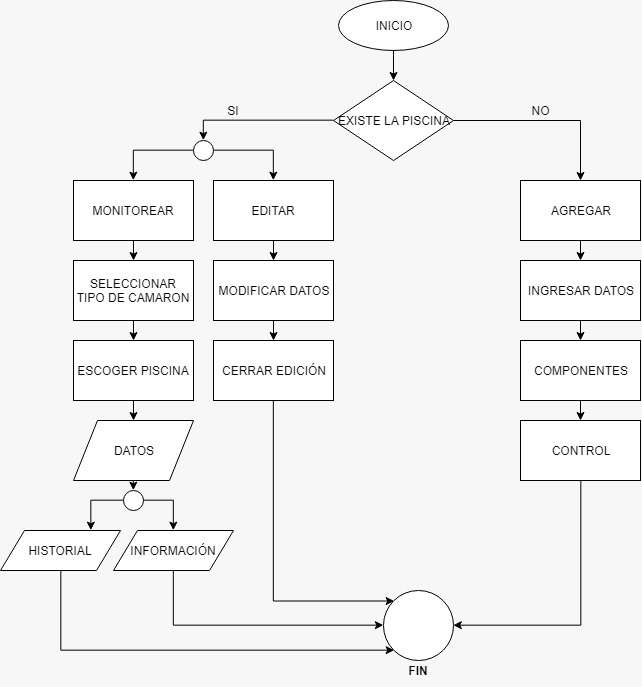


Fig. 1: Diagrama de Flujo

Fig. 2: Diagrama de Casos

A su vez se adjunta el diseño del diagrama de red, en el cual se muestran todos los componentes que forman parte de la red y su comunicación.

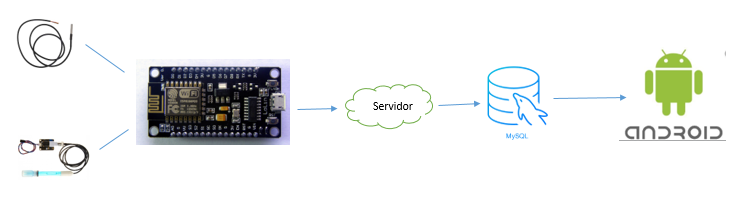


Fig. 3: Diagrama de Red

Como se puede apreciar en la figura 5 El dispositivo de control cuenta con dos sensores conectados a la placa madre, la cual mediante un microcontrolador basado en Arduino realiza la adquisición de los datos de ambos sensores, luego de esto dichos datos son enviados a un servidor que tiene una base de datos.

Para crear una base de datos que se pueda acceder desde cualquier dispositivo se implementó un servidor en una raspberry pi 3 que almacenaba una base de datos MySQL, dicho servidor permite el envío de datos de manera remota.

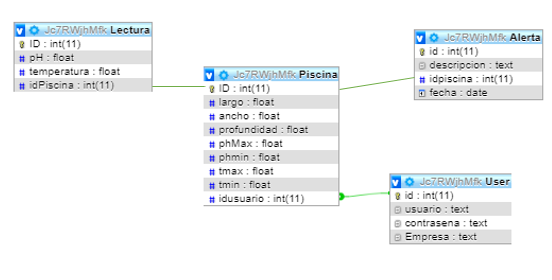


Fig. 4: Diagrama de Entidad-Relación.

Para el diseño de la base de datos remota es necesario definir todas las entidades que participan dentro de la misma mediante un diagrama Entidad-Relación, con el fin de llevar un orden en la de los datos con base en las reglas de la normalización.

Posterior a la etapa del diseño de la base de datos, se realiza el código de NodeMCU que se encarga de la obtención de los valores censados por los diferentes módulos de temperatura y pH, dichos valores son los que van a controlar la calidad del agua de las piscinas. Con la misma placa base se realiza el envío de los datos a la anteriormente creada base de datos, ya que se basa en el módulo WIFI ESP8266 de arduino no es necesario incluir un módulo para la conexión a internet, luego de esto el servidor obtiene los datos y los almacena.

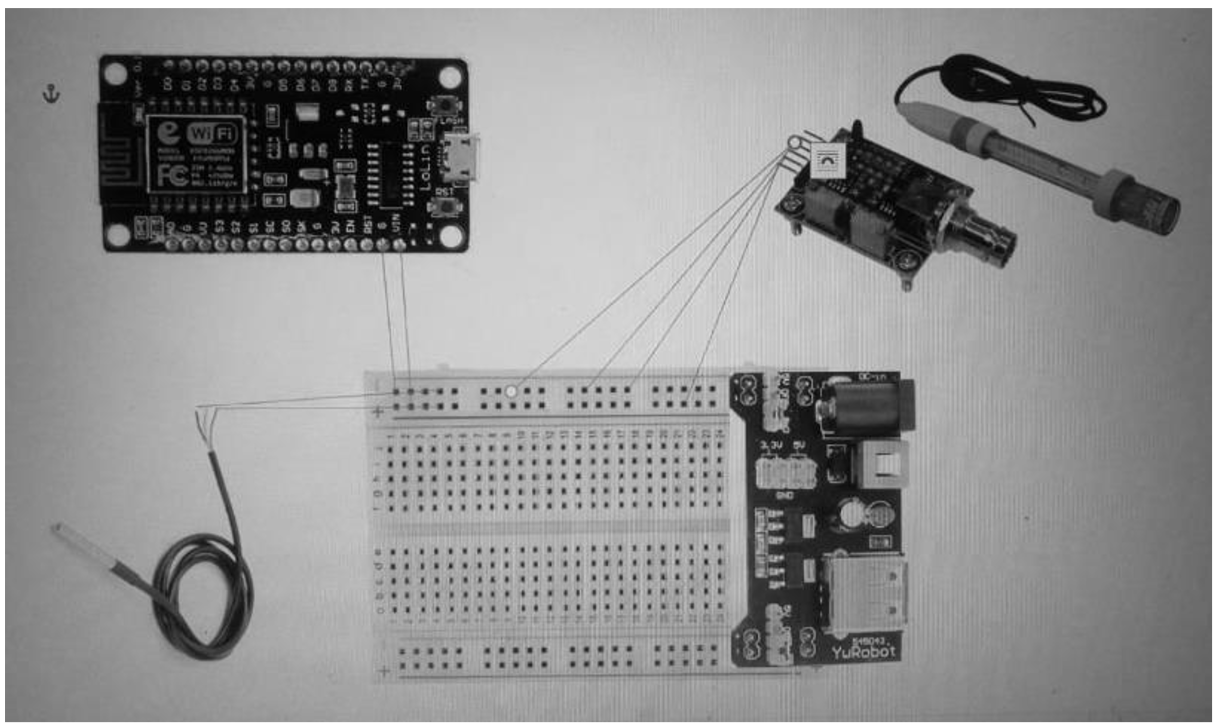


Fig. 5: Diagrama de circuito

Para que el usuario visualice los datos se desarrolló una aplicación que cuenta con un sistema de inicio de sesión haciendo que solo las personas que tienen cuenta registrada en la base de datos puedan ingresar al monitoreo de las piscinas, por ende, se realizaron validaciones en el código de la aplicación para esta sección.

Luego de esto se encuentra el menú del aplicativo que cuenta con dos botones el de monitoreo, que permite visualizar las piscinas y sus datos a detalle y el de agregar piscina que permite agregar una piscina a la base de datos, debido a que implementamos un sistema que crea los botones con base en la cantidad de piscinas que se encuentren en la base, si agregamos una nueva piscina inmediatamente se va a crear el botón correspondiente a la misma dentro de la zona de monitoreo.

CÓDIGO DE ARDUINO

El código usado para el modulo permite leer los valores del sensor de pH y el de temperatura, además se conecta con el servidor que envía una orden para que ingrese los valores del sensados en la base de datos del proyecto.

// Librerias para el modulo de wifi

#include <ESP8266WiFi.h>

// Librerias para el sensor DS18B20

#include <DallasTemperature.h>

#include <OneWire.h>

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Definicion de variables y constantes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// CREDENCIALES DE ACCESO A INTERNET

const char\* ssid = "POCOPHONE";

const char\* password = "winteralava";

const char\* host = "watercontrol.ddns.net";

// DEFINICION DE SENSOR DS18B20

const int pinDatosDQ = 4;

OneWire oneWireObjeto(pinDatosDQ);

DallasTemperature sensorDS18B20(&oneWireObjeto);

// DEFINICION DE SENSOR DE pH

const int analogInPin = A0;

int sensorValue = 0;

unsigned long int avgValue;

float b;

int buf[10], temp1;

void setup() {

Serial.begin(9600);

// SETUP CONEXION WIFI

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

// Se inicializa el wifi

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

// SETUP SENSOR DS18B20

sensorDS18B20.begin();

}

void loop() {

// LEYENDO SENSOR DS18B20

Serial.println("Mandando comandos a los sensores");

sensorDS18B20.requestTemperatures();

Serial.print("Temperatura sensor DS18B20: ");

Serial.print(sensorDS18B20.getTempCByIndex(0));

float temp = sensorDS18B20.getTempCByIndex(0);

// LEYENDO SENSOR DE pH

for (int i = 0; i < 10; i++)

{

buf[i] = analogRead(analogInPin);

delay(10);

}

for (int i = 0; i < 9; i++)

{

for (int j = i + 1; j < 10; j++)

{

if (buf[i] > buf[j])

{

temp1 = buf[i];

buf[i] = buf[j];

buf[j] = temp1;

}

}

}

avgValue = 0;

for (int i = 2; i < 8; i++)

avgValue += buf[i];

float pHVol = (float)avgValue \* 5.0 / 1024 / 6;

float phValue = -5.70 \* pHVol + 21.34;

Serial.print("sensor = ");

Serial.println(phValue);

// REALIZANDO CONEXION A INTERNET

Serial.print("connecting to ");

Serial.println(host);

WiFiClient client;

const int httpPort = 80;

if (!client.connect(host, httpPort)) {

Serial.println("connection failed");

return;

}

String url = "/enviodedatos.php";

String key = "?pass=1234";

String dato1 = "&temperatura=";

String dato2 = "&ph=";

Serial.print("Requesting URL: ");

Serial.println(url);

// Se envia el paquete de datos por medio de internet, para ejecutar el archio php en el servidor

// Para que el servidor ejecute los querys en la base de datos

client.print(String("GET ") + url + key + dato1 + temp + dato2 + phValue + " HTTP/1.1\r\n" +

"Host: " + host + "\r\n" +

"Connection: close\r\n\r\n");

unsigned long timeout = millis();

while (client.available() == 0) {

if (millis() - timeout > 5000) {

Serial.println(">>> Client Timeout !");

client.stop();

return;

}

}

while (client.available()) {

String line = client.readStringUntil('\r');

Serial.print(line);

}

Serial.println();

Serial.println("closing connection");

delay(15000);

}

CÓDIGO DEL APLICATIVO MÓVIL

SplashScreen.java

**public class** SplashScreen **extends** AppCompatActivity {  
  
 */\*\*  
 \* Permite generar una pantalla de presentacion de la app por un tiempo asigando y luego nos dirige a la siguiente ventana.  
 \** ***@param savedInstanceState*** *\*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_splash\_screen***);  
  
 **new** Handler().postDelayed(**new** Runnable() {  
 @Override  
 **public void** run() {  
 Intent i=**new** Intent (SplashScreen.**this**, MainActivity.**class**);  
 startActivity(i);  
 finish();  
 }  
 },1500);  
 }  
}

MainActivity.java

**public class** MainActivity **extends** AppCompatActivity {  
 **private static final** String ***STRING\_PREFERENCES***=**"usuario"**;  
 **private static final** String ***USUARIO***=**"usuario.sesion"**;  
 **private static final** String ***CONTRASEÑA***=**"usuario.contraseña"**;  
 **private** EditText **user**,**password**;  
 **private** String **u**=**"ccoloma"**, **c**=**"1234"**;  
  
 */\*\*  
 \* En este método sobreescrito se inicializan las variables a usar y que se encuentran definidas en el archivo XML  
 \* Ademas nos permite revisar en la memoria de la app si aulgun usuario a iniciado sesion antes para iniciar con la siguiente ventana, caso contrario pide que ingrese su usuario y cotnraseña  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **if** (ObtenercontraseñaGuardada().equals(**c**) && ObtenerUsuarioGuardado().equals(**u**)){  
 Intent i = **new** Intent(**this**, InicioActivity.**class** );  
 startActivity(i);  
 Notificaciones.*guardarEstado*(MainActivity.**this**,**true**);  
 startService(**new** Intent (**this**,Notificaciones.**class**));  
 finish();  
 }  
  
 **user**= (EditText) findViewById(R.id.***user***);  
 **password**= (EditText) findViewById(R.id.***password***);  
 }  
  
 */\*\*  
 \* Este metodo valida el usuario y contraseña ingresado, de ser correctos los dirige a la siguiente ventana, caso contrario indica que los datos ingresados son incorrectos.  
 \*/* **public void** Ingresar (View v) {  
 String usuario=**user**.getText().toString().trim();  
 String contrasena=**password**.getText().toString().trim();  
  
 **if**(usuario.equals(**u**) && contrasena.equals(**c**)){  
 Intent i = **new** Intent(**this**, InicioActivity.**class** );  
 startActivity(i);  
 guardarInicio();  
 Notificaciones.*guardarEstado*(MainActivity.**this**,**true**);  
 startService(**new** Intent (**this**,Notificaciones.**class**));  
 finish();  
 }  
 **else** {  
 Toast.*makeText*(**this**, **"El usuario o contraseña ingresados son incorrectos!"**,Toast.***LENGTH\_SHORT***).show();  
 **user**.setText(**""**);  
 **password**.setText(**""**);  
 }  
  
 }  
  
 */\*\*  
 \* Este metodo permite guardar la informacion del usuario con el fin de que este no tenga q iniciar sesion si llegara a cerrar la app  
 \*/* **public void** guardarInicio(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putString(***USUARIO***,**user**.getText().toString().trim()).apply();  
 preferences.edit().putString(***CONTRASEÑA***,**password**.getText().toString().trim()).apply();  
 }  
  
 */\*\*  
 \* Este metodo elimina los datos del usuario cuando este cierra sesion en el aplicativo.  
  
 \*/* **public static void** CerrarSesion(Context c,String users, String passwords){  
 SharedPreferences preferences=c.getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putString(***USUARIO***,users).apply();  
 preferences.edit().putString(***CONTRASEÑA***,passwords).apply();  
 }  
  
 */\*\*  
 \* Obtiene al nombre del usurio que inicia sesion desde la memoria del aplicativo  
 \*/* **public** String ObtenerUsuarioGuardado(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 **return** preferences.getString(***USUARIO***,**""**);  
 }  
  
 */\*\*  
 \* Retorna la contraseña del usurio que inicia sesion guardada en la memoria del aplicativo  
 \*/* **public** String ObtenercontraseñaGuardada(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 **return** preferences.getString(***CONTRASEÑA***,**""**);  
 }  
}

InicioActivity.java

**public class** InicioActivity **extends** AppCompatActivity {  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_inicio***);  
 }  
  
 */\*\*  
 \* metodo onClick del boton monitorear donde nos lleva a la vantana de monitoreo donde se muestras las piscinas ingresadas por el usuario  
 \*/* **public void** Monitorear (View v){  
 Intent i = **new** Intent(**this**, MonitoreoActivity.**class** );  
 startActivity(i);  
 }  
  
 */\*\*  
 \* metodo onClick del boton agregar donde nos lleva a la vantana agregar piscina  
 \*/* **public void** Agregar(View v) {  
 Intent i = **new** Intent(**this**, AgregarActivity.**class** );  
 startActivity(i);  
 }  
  
 */\*\*  
 \* metodo onClick del boton salir que cierra sesion del usuario llevandonos a la ventana de ingreso  
 \*/* **public void** Salir(View v){  
 MainActivity.*CerrarSesion*(InicioActivity.**this**, **""**,**""**);  
 Intent i = **new** Intent(**this**, MainActivity.**class** );  
 stopService(**new** Intent (**this**,Notificaciones.**class**));  
 startActivity(i);  
 finish();  
 }  
  
}

MonitoreoActivity.java

**public class** MonitoreoActivity **extends** AppCompatActivity {  
 **private** LinearLayout **contenedor**;  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
  
  
 */\*\*  
 \* Este metodo permite ingresar a la actividad y tambien agrega botones en esta de manera dinamica dependiendo de la cantidad de piscinas ingresadas en la base de datos  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_monitoreo***);  
 **contenedor**= (LinearLayout) findViewById(R.id.***contenedor***);  
 ArrayList<Boton> botones= **new** ArrayList<Boton>();  
 **for** (**int** i=1; i<=(cantPiscinas()); i++){  
 botones.add(**new** Boton(**"Piscina "**+String.*valueOf*(i),i));  
 }  
 **for** (**final** Boton bt:botones){  
 Button btn = **new** Button(**this**);  
 btn.setText(bt.**texto**);  
 btn.setId(bt.**ID**);  
 btn.setTextSize(18);  
 btn.setBackgroundColor(Color.*parseColor*(**"#FFE794"**));  
 btn.setTextColor(Color.***BLACK***);  
 **contenedor**.addView(btn);  
 **contenedor**.addView(**new** TextView(**this**));  
 btn.setOnClickListener(**new** View.OnClickListener() {  
 **public void** onClick(View v) {  
 Intent i = **new** Intent(MonitoreoActivity.**this**,EstadoActualActivity.**class** );  
 i.putExtra(**"ID"**,String.*valueOf*(bt.**ID**));  
 startActivity(i);  
 }  
 });  
 }  
 }  
  
  
 */\*\*  
 \* Retorna la cantidad de piscinas ingresadas en la base de datos tomando en cuenta al usuario  
 \*/* **public int** cantPiscinas(){  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Piscina;"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(MonitoreoActivity.**this**,**"Conexión Establecida"**, Toast.***LENGTH\_LONG***).show();  
  
 String resultadoConsulta = resultadoSQL[0];  
 String numFilas = resultadoSQL[1];  
 String numColumnas = resultadoSQL[2];  
 **return** Integer.*parseInt*(numFilas);  
 }**catch**(Exception ex)  
 {  
 Toast.*makeText*(**this**, **"Error al obtener resultados de la consulta Transact-SQL: "** + ex.getMessage(), Toast.***LENGTH\_LONG***).show();  
 **return** 0;  
 }  
 }  
  
 **class** Boton{  
 String **texto**;  
 **int ID**;  
  
 **public** Boton(String texto, **int** ID) {  
 **this**.**texto** = texto;  
 **this**.**ID** = ID;  
 }  
 }  
}

EstadoActualActivity.java

**public class** EstadoActualActivity **extends** AppCompatActivity {  
 **private** String **piscina**,**ID**;  
 **private** TextView **name**;  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
 **private** TextView **consulta**;  
  
 */\*\*  
 \* En este método sobreescrito se inicializan las variables a usar y que se encuentran definidas en el archivo XML  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_estado\_actual***);  
 Bundle bundle = getIntent().getExtras();  
 **ID**= bundle.getString(**"ID"**);  
 **piscina**=**"Piscina #"** + **ID**;  
 **name**=(TextView) findViewById(R.id.***title***);  
 **consulta**=(TextView) findViewById(R.id.***historial***);  
 **name**.setText(**piscina**);  
 MostrarDatos();  
 }  
  
 */\*\*  
 \* Este genera un cuadro de dialogo de seleccion que permite que el usuario elija la opcion de historial que desea ver  
 \*/* **public void** Historial(View v){  
 **final** String[] items = {**"General"**, **"Alertas"**};  
 AlertDialog.Builder builder =**new** AlertDialog.Builder(**this**);  
 builder.setTitle(**"HISTORIAL"**).setItems(items, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** item) {  
 **if** (items[item].equals(**"General"**)){  
 Intent i = **new** Intent(EstadoActualActivity.**this**, HistorialActivity.**class** );  
 i.putExtra(**"IDpiscina"**,String.*valueOf*(**ID**));  
 startActivity(i);  
 }  
 **else**{  
 Intent i = **new** Intent(EstadoActualActivity.**this**, AlertasActivity.**class** );  
 i.putExtra(**"IDpiscina"**,String.*valueOf*(**ID**));  
 startActivity(i);  
 }  
 }  
 });  
  
 builder.create().show();  
 }  
  
 */\*\*  
 \* Este metodo muestra los datos de la piscina que desea monitorear  
 \*/* **public void** MostrarDatos(){  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Piscina WHERE ID="** +**ID**+**";"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(EstadoActualActivity.**this**,**"Conexión Establecida"**, Toast.***LENGTH\_LONG***).show();  
  
 String[] resultadoConsulta = resultadoSQL[0].split(**","**);  
 **consulta**.setText(**"Datos:\n"**+**"Largo= "**+resultadoConsulta[1] + **"\nAncho= "** +resultadoConsulta[2] + **"\nProfundidad= "** +  
 resultadoConsulta[3] + **"\n\nDatos de Control:\npH maximo= "** +resultadoConsulta[4] + **"\npH minimo= "** +resultadoConsulta[5]  
 + **"\nTemperatura Maxima= "** +resultadoConsulta[6] + **" ºC\nTemperatura minima= "** +resultadoConsulta[7]+**" ºC"**);  
 }**catch**(Exception ex)  
 {  
 Toast.*makeText*(**this**, **"Error al obtener resultados de la consulta Transact-SQL: "** + ex.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
  
 }  
}

HistorialActivity.java

**public class** HistorialActivity **extends** AppCompatActivity {  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
 **private** String **ID**;  
 **private** TableLayout **table**;  
  
 */\*\*  
 \* En este método sobreescrito se inicializan las variables a usar y que se encuentran definidas en el archivo XML  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_historial***);  
 Bundle bundle = getIntent().getExtras();  
 **ID**= bundle.getString(**"IDpiscina"**);  
 **table** = (TableLayout) findViewById(R.id.***tablelayout***);  
 mostrarResultados();  
  
 }  
  
 */\*\*  
 \* Este metodo permite mostrar los datos de las lecturas tomadas por los sensores ubicadas en la base de datos  
 \*/* **public void** mostrarResultados()  
 {  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Lectura WHERE idPiscina="**+**ID**+**";"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(HistorialActivity.**this**,**"Conexión Establecida"**, Toast.***LENGTH\_LONG***).show();  
 String[] cabecera={**"\n ,Fecha y Hora,Valor de pH,Temperatura, "**,**"1"**,**"5"**};  
 Tablas(cabecera);  
 Tablas(resultadoSQL);  
 *// String resultadoConsulta = resultadoSQL[0];  
 //filasconsulta.setText("ID, pH, Temperatura, #Piscina\n"+resultadoConsulta );* }**catch**(Exception ex)  
 {  
 Toast.*makeText*(**this**, **"Error al obtener resultados de la consulta Transact-SQL: "** + ex.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
 }  
  
 */\*\*  
 \* Este metodo nos permite colocar los datos obtenidos de la base de datos y colocarlos dentro de un Table Layout, con el fin de mejorar la presentacion de dicha infromación  
 \*/* **public void** Tablas(String[] resultado) {  
  
 String resultadoConsulta = resultado[0];  
 **int** NUM\_ROW = Integer.*parseInt*(resultado[1]);  
 **int** NUM\_COLS = Integer.*parseInt*(resultado[2]);  
 String[] filas = resultadoConsulta.split(**"\n"**);  
 **for** (**int** i = 1; i < (NUM\_ROW+1); i++) {  
 TableRow tableRow = **new** TableRow(HistorialActivity.**this**);  
 **table**.addView(tableRow);  
 tableRow.setBackgroundColor(Color.***GRAY***);  
 String[] columnas = filas[i].split(**","**);  
  
 **for** (**int** j = 1; j < (NUM\_COLS-1); j++) {  
 TextView valor = **new** TextView(getApplicationContext());  
 valor.setText(columnas[j] + **" "**);  
 valor.setTextColor(Color.***BLACK***);  
 tableRow.addView(valor);  
 }  
 }  
 }  
}

AlertasActivity.java

**public class** AlertasActivity **extends** AppCompatActivity {  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
 **private** String **ID**;  
 **private** TableLayout **table**;  
  
 */\*\*  
 \* En este método sobreescrito se inicializan las variables a usar y que se encuentran definidas en el archivo XML, o gruadados en extras  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_alertas***);  
 Bundle bundle = getIntent().getExtras();  
 **ID**= bundle.getString(**"IDpiscina"**);  
 **table** = (TableLayout) findViewById(R.id.***tablealertas***);  
 mostrarResultados();  
 }  
  
 */\*\*  
 \* Este metodo permite mostrar los resultados de las alertas registradas en la bases de datos.  
 \*/* **public void** mostrarResultados()  
 {  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Alerta WHERE idpiscina="**+**ID**+**";"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(AlertasActivity.**this**,**"Conexión Establecida"**, Toast.***LENGTH\_LONG***).show();  
 String[] cabecera={**"\n ,Fecha y Hora,Descripcion, "**,**"1"**,**"4"**};  
 Tablas(cabecera);  
 Tablas(resultadoSQL);  
 }**catch**(Exception ex)  
 {  
 Toast.*makeText*(**this**, **"Error al obtener resultados de la consulta Transact-SQL: "** + ex.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
 }  
  
 */\*\*  
 \* Este metodo nos permite colocar los datos obtenidos de la base de datos y colocarlos dentro de un Table Layout, con el fin de mejorar la presentacion de dicha infromación  
 \** ***@param resultado*** *\*/* **public void** Tablas(String[] resultado) {  
  
 String resultadoConsulta = resultado[0];  
 **int** NUM\_ROW = Integer.*parseInt*(resultado[1]);  
 **int** NUM\_COLS = Integer.*parseInt*(resultado[2]);  
 String[] filas = resultadoConsulta.split(**"\n"**);  
 **for** (**int** i = 1; i < (NUM\_ROW+1); i++) {  
 TableRow tableRow = **new** TableRow(**this**);  
 **table**.addView(tableRow);  
 tableRow.setBackgroundColor(Color.***LTGRAY***);  
 String[] columnas = filas[i].split(**","**);  
  
 **for** (**int** j = 1; j < (NUM\_COLS-1); j++) {  
 TextView valor = **new** TextView(getApplicationContext());  
 valor.setText(columnas[j] + **" "**);  
 valor.setTextColor(Color.***BLACK***);  
 tableRow.addView(valor);  
 }  
 }  
 }  
}

AgregarActivity.java

**public class** AgregarActivity **extends** AppCompatActivity {  
 **private** EditText **alto**, **ancho**, **largo**, **pHmax**, **pHmin**, **TempMax**, **TempMin**;  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
  
 */\*\*  
 \* En este método sobreescrito se inicializan las variables a usar y que se encuentran definidas en el archivo XML  
 \*/* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_agregar***);  
 **alto**=(EditText) findViewById(R.id.***alto***);  
 **ancho**=(EditText) findViewById(R.id.***ancho***);  
 **largo**=(EditText) findViewById(R.id.***largo***);  
 **pHmax**=(EditText) findViewById(R.id.***phmax***);  
 **pHmin**=(EditText) findViewById(R.id.***phmin***);  
 **TempMax**=(EditText) findViewById(R.id.***tempmax***);  
 **TempMin**=(EditText) findViewById(R.id.***tempmin***);  
 }  
  
 */\*\*  
 \* Este metodo permite agregar en la base de datos una nueva piscina de la camaronera, enviando los datos ingresados en la activity  
 \* a la base de datos del proyecto, validando todos los datos ingresados.  
 \*  
 \*/* **public void** agregar(View v){  
 **if** (validarDatos()){  
 AlertDialog.Builder builder =  
 **new** AlertDialog.Builder(**this**);  
  
 builder.setMessage(**"¿Desea Registrar la piscina?"**).setTitle(**"Mensaje de confirmación"**)  
 .setPositiveButton(**"YES"**, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** id) {  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"INSERT INTO Piscina (ID, largo, ancho, profundidad, phMax, phmin, tmax, tmin, idusuario) VALUES (NULL,"**+  
 **largo**.getText().toString()+**","**+**ancho**.getText().toString() +**","**+**alto**.getText().toString()+**","**+**pHmax**.getText().toString()  
 +**","**+**pHmin**.getText().toString()+**","**+**TempMax**.getText().toString()+**","**+**TempMin**.getText().toString()+**", 1);"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 Toast.*makeText*(AgregarActivity.**this**, **"Se ha registrado la piscina con exito!"**,Toast.***LENGTH\_SHORT***).show();  
 **alto**.setText(**""**); **ancho**.setText(**""**); **largo**.setText(**""**); **pHmax**.setText(**""**); **pHmin**.setText(**""**); **TempMax**.setText(**""**); **TempMin**.setText(**""**);  
  
 }**catch**(Exception ex)  
 {  
 Toast.*makeText*(AgregarActivity.**this**, **"Error al agregar la piscina."**,Toast.***LENGTH\_SHORT***).show();  
 }  
  
 }  
 }).setNegativeButton(**"NO"**,**new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** id) {  
 dialog.cancel();  
 }  
 });  
 builder.create().show();  
 }  
  
 **else**{  
 AlertDialog.Builder builder =  
 **new** AlertDialog.Builder(**this**);  
  
 builder.setMessage(**"La información ingresada no es valida."**)  
 .setTitle(**"Error"**)  
 .setPositiveButton(**"OK"**, **new** DialogInterface.OnClickListener() {  
 **public void** onClick(DialogInterface dialog, **int** id) {  
 dialog.cancel();  
 }  
 });  
 builder.create().show();  
 }  
 }  
  
 */\*\*  
 \* Esta funcion se encarga de validar que todos los datos sean de tipo float, ademas que el valor maximo de algun parametro sea siempre mayor que el minimo,  
 \* usando la funcion de try--catch donde si logra transformarlo a float todos los valores inresados, entonces retorna true, y de caso contrario retorna false.  
 \*  
 \** ***@return*** *\*/* **public** Boolean validarDatos(){  
 **try**{  
 Float.*parseFloat*(**alto**.getText().toString().trim());  
 Float.*parseFloat*(**ancho**.getText().toString().trim());  
 Float.*parseFloat*(**largo**.getText().toString().trim());  
 Float phM=Float.*parseFloat*(**pHmax**.getText().toString().trim());  
 Float phm=Float.*parseFloat*(**pHmin**.getText().toString().trim());  
 Float TM=Float.*parseFloat*(**TempMax**.getText().toString().trim());  
 Float Tm=Float.*parseFloat*(**TempMin**.getText().toString().trim());  
 **if** ((phm<phM)&&(phm>0)&&(phM<14)&&(TM>Tm)){  
 **return true**;  
 }  
 **return false**;  
 }**catch**(Exception e){  
 **return false**;  
 }  
 }  
}

AsyncQuery.java

**public class** AsyncQuery **extends** AsyncTask<String[],Void,String[]> {  
  
 **private** Connection **conexionMySQL**;  
 **private** Statement **st** = **null**;  
 **private** ResultSet **rs** = **null**;  
  
 */\*\*  
 \* Este metodo nos permite ingresar a la base de datos obtener y enviar datos hacia ella.  
  
 \*/* **protected** String[] doInBackground(String[]... datos) {  
 String sql = datos[0][5];  
 String resultadoSQL = **""**;  
 String[] totalResultadoSQL = **null**;  
 **int** numColumnas = 0;  
 **int** numFilas = 0;  
 String SERVIDOR = datos[0][0];  
 String PUERTO = datos[0][1];  
 String BD = datos[0][2];  
 String USUARIO = datos[0][3];  
 String PASSWORD = datos[0][4];  
  
 **try**{  
 **conexionMySQL** = DriverManager.*getConnection*(**"jdbc:mysql://"** + SERVIDOR + **":"** + PUERTO + **"/"** + BD,  
 USUARIO,PASSWORD);  
  
 **st** = **conexionMySQL**.createStatement();  
 **if**(sql.contains(**"INSERT"**)){  
 Log.*d*(**"Query: "**,sql);  
 **st**.executeUpdate(sql);  
 }  
 **else** {  
 **rs** = **st**.executeQuery(sql);  
 **rs**.last();  
 numFilas = **rs**.getRow();  
 **if**(numFilas == 0)  
 {  
 resultadoSQL = **"No se ha producido ningún resultado. Revise la consulta realizada.\n"**;  
 }**else** {  
 **for**(**int** i=1;i<=numColumnas;i++){  
 **if**(i>1) resultadoSQL += **","**;  
 resultadoSQL += **rs**.getMetaData().getColumnName(i);  
 }  
 resultadoSQL += **"\n"**;  
 **rs**.beforeFirst();  
 **while** (**rs**.next())  
 {  
 numColumnas = **rs**.getMetaData().getColumnCount();  
 **for**(**int** i=1;i<=numColumnas;i++){  
 **if**(i>1) resultadoSQL += **","**;  
 resultadoSQL += **rs**.getString(i);  
 }  
 resultadoSQL += **"\n"**;  
 }  
 }  
 totalResultadoSQL = **new** String[]{ resultadoSQL,String.*valueOf*(numFilas),String.*valueOf*(numColumnas)};  
 }  
  
 }**catch**(SQLException ex)  
 {  
 Log.*d*(**"Error de conexion"**, ex.getMessage());  
 }  
 **finally** {  
 **try** {  
 **if**(**rs** != **null**)  
 {  
 **rs**.close();  
 }  
 **st**.close();  
 **conexionMySQL**.close();  
 } **catch** (SQLException e)  
 {  
 e.printStackTrace();  
 }  
 }  
 **return** totalResultadoSQL;  
 }  
  
}

Notificaciones.java

**public class** Notificaciones **extends** Service {  
 **private final static** String ***CHANNEL\_ID***=**"NOTIFICACION"**;  
 **private final static int *NOTIFICACION\_ID***=1;  
 **private static final** String ***STRING\_PREFERENCES***=**"alertas"**;  
 **private static final** String ***num***=**"alerta.cantidad"**;  
  
 **private** String **serverIP** = **"remotemysql.com"**;  
 **private** String **port** = **"3306"**;  
 **private** String **userMySQL** = **"Jc7RWjhMfk"**;  
 **private** String **pwdMySQL** = **"BPY0HGFDSZ"**;  
 **private** String **database** = **"Jc7RWjhMfk"**;  
 **private** String[] **datosConexion** = **null**;  
  
 @Override  
 **public void** onCreate(){  
  
 }  
  
 */\*\*  
 \*Este metodo sobreescrito permite iniciar un servicio en segundo plano con el fin de retornar una notificacion si una alerta se produce en las piscinas  
 \*/* @Override  
 **public int** onStartCommand(Intent intent, **int** flag, **int** idProcess){  
 **if** (ObtenerEstado()) {  
 mostrarResultados();  
 startService(**new** Intent(**new** Intent(**this**, Notificaciones.**class**)));  
 }  
 **return *START\_STICKY***;  
  
 }  
  
 */\*\*  
 \* Este metodo permite detener el servicio iniciado, deteniendose en el momento que el usuario cierra sesion en el aplicativo.  
 \*/* @Override  
 **public void** onDestroy(){  
 SharedPreferences preferences= (SharedPreferences) getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putBoolean(**"servicio"**,**false**).apply();  
 }  
  
 @Override  
 **public** IBinder onBind(Intent intent) {  
 **return null**;  
 }  
  
 */\*\*  
 \* Este metodo permite generar una notificacion del aplicativo  
 \*/* **public void** generarNotificaicon(){  
 **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***O***){  
 CharSequence name =**"Alerta"**;  
 NotificationChannel notificacionChanel= **new** NotificationChannel(***CHANNEL\_ID***,name, NotificationManager.***IMPORTANCE\_DEFAULT***);  
 NotificationManager notification =(NotificationManager) getSystemService(***NOTIFICATION\_SERVICE***);  
 notification.createNotificationChannel(notificacionChanel);  
 }  
  
 NotificationCompat.Builder builder=**new** NotificationCompat.Builder(getApplicationContext(),***CHANNEL\_ID***);  
 builder.setSmallIcon(R.drawable.***alert***);  
 builder.setContentTitle(**"Alerta"**);  
 builder.setContentText(**"Ha ocurrido un cambio de parametros en el agua."**);  
 builder.setColor(Color.***BLUE***);  
 builder.setPriority(NotificationCompat.***PRIORITY\_DEFAULT***);  
 builder.setLights(Color.***MAGENTA***,1000,1000);  
 builder.setVibrate(**new long**[]{1000,1000,1000,1000});  
 builder.setDefaults(Notification.***DEFAULT\_SOUND***);  
 NotificationManagerCompat not=NotificationManagerCompat.*from*(getApplicationContext());  
 not.notify(***NOTIFICACION\_ID***,builder.build());  
 }  
  
 */\*\*  
 \* Este metodo permite determinar si ha sido agragada un nuevo dato en la tabla de alertas de la base de datos, generando una notificacion sobre esto.  
 \*/* **public void** mostrarResultados()  
 {  
 String[] resultadoSQL = **null**;  
 **try**{  
 **datosConexion** = **new** String[]{  
 **serverIP**,  
 **port**,  
 **database**,  
 **userMySQL**,  
 **pwdMySQL**,  
 **"SELECT \* FROM Alerta;"** };  
 String driver = **"com.mysql.jdbc.Driver"**;  
 Class.*forName*(driver).newInstance();  
 resultadoSQL = **new** AsyncQuery().execute(**datosConexion**).get();  
 String filas=resultadoSQL[1];  
 **int** filasanteriores=Integer.*parseInt*(ObtenerAlertas());  
 **if**(Integer.*parseInt*(filas)>filasanteriores){  
 guardarAlerta(filas);  
 generarNotificaicon();  
 }  
  
 }**catch**(Exception ex) { }  
 }  
  
 */\*\*  
 \* Este metodo permite guardar la ultima cantidad de datos guardados en la tabla de alertas  
 \** ***@param s*** *\*/* **public void** guardarAlerta(String s){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putString(***num***,s).apply();  
 }  
  
 */\*\*  
 \*Permite guardar el estado de inicio de sesion, siendo true si ha iniciado algun usuario o false si no lo ha hecho o ha cerrado sesion  
 \*/* **public static void** guardarEstado(Context c, **boolean** s){  
 SharedPreferences preferences=c.getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 preferences.edit().putBoolean(**"servicio"**,s).apply();  
 }  
  
 */\*\*  
 \* Permtie obtener el dato de la cantidad de los ultimos datos guardados en la tabla de alertas  
 \** ***@return*** *\*/* **public** String ObtenerAlertas(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 **return** preferences.getString(***num***,**"0"**);  
 }  
  
 */\*\*  
 \* Permite Obtener los datos del estado de la app  
 \** ***@return*** *\*/* **public boolean** ObtenerEstado(){  
 SharedPreferences preferences=getSharedPreferences(***STRING\_PREFERENCES***,***MODE\_PRIVATE***);  
 **return** preferences.getBoolean(**"servicio"**,**false**);  
 }  
  
}

Resultados y Análisis.

En la figura se puede apreciar el prototipo final del dispositivo de control que cuenta con los dos sensores y el NodeMCU dentro de un contenedor que se puede desplazar a lo largo de la maqueta que simula una piscina camaronera. Como idea inicial se plantaba realizar un contenedor que pueda flotar sobre la piscina, no obstante, en ese caso requeriría de una fuente de alimentación que pueda desplazarse de igual forma, por ende, se optó por esta alternativa.

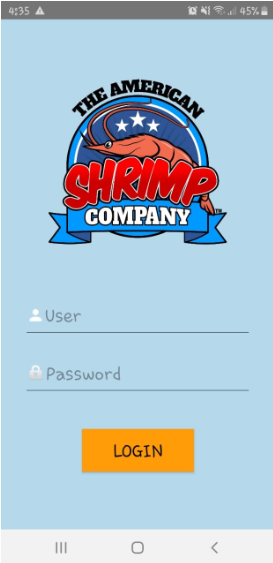


Fig. 6: Sistema de inicio de sesión

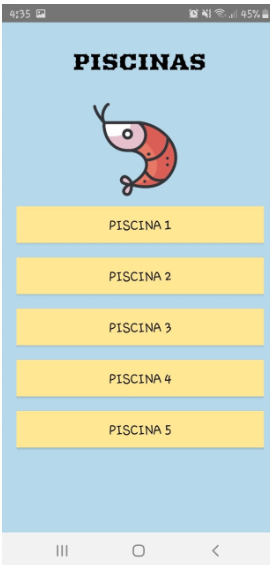


Fig. 7: Piscinas por monitorear

La aplicación recibe los datos censados de la base de datos y los muestra dependiendo de la piscina, además de esto cada piscina tiene un botón de historial que se encarga de guardar los datos censados, cabe recalcar que el periodo entre cada censado es de 1 hora y es ajustable dependiendo de lo que solicite el usuario.

Como podemos observar los parámetros de la piscina 1 se encuentran en los rangos correctos tanto en temperatura como en pH, asegurando así el cultivo óptimo de las larvas, en caso de que uno de estos parámetros se salga del rango preestablecido al usuario le llega una notificación de tipo alerta, indicando que piscina tiene un parámetro fuera de rango.

Conclusiones

Se concluye que realizado el proyecto se puede establecer efectivamente una conexión entre los datos censados por un Arduino y un servidor montado en una Raspberry Pi. Esta gestión de datos se la desarrolló mediante MySQL. Por otro lado, el desarrollo de una app móvil que pueda obtener estos datos resulto complicado debido a que se necesitaba obtener permisos de poder manejar los puertos de la conexión a internet en la que el servidor estaba conectado.

Los datos censados como el pH y la temperatura ofrecieron una visión básica pero fundamental para el control de una piscina camaronera. Además, se logró desarrollar un prototipo del montaje de los soportes que se podrían implementar en una camaronera real.

Finalmente, la actualización de la base de datos por medio de los sensores que recepta el módulo de wifi ejecuta archivos get y post de la red local.

Recomendaciones

Se puede desarrollar una mejor gestión de los puertos utilizados para el acceso a la base de datos directa utilizando un conector de la app móvil a MySQL. De no ser posible, probar con una conexión de la app en Android con archivos php del servidor implementado para que de esta manera no sea necesario acceder directamente a la base de datos.

También se sugiere utilizar más sensores como los de oxígeno disuelto y turbulencia para poder tener un mayor control del estado de una piscina. A su vez, poder gestionar la aplicación desarrollada para que se pueda crear esta app para varias empresas, en otras palabras, ofrecer la opción de tener varios usuarios.

Referencias

|  |  |
| --- | --- |
| [1] | D. P. Chanratchakool, «Recomendaciones Tecnicas y principios sobre manejo de estanques,» Bangkok, 2002. |
| [2] | «CONSULTORIA EN CULTIVO DE CAMARON,» Francia, 1989. |
| [3] | R. M. A. Gabriela, «Evaluacion y determinacion de la calidad del agua en las piscinas de la camaronera Boca Salima, parael mejoramiento de la produccion de camaron,» Loja, 2015. |
| [4] | B. F. S. V., «Zoetecnocampo,» 2015. [En línea]. Available: http://www.zoetecnocampo.com/Documentos/manejo\_camaron.htm. |

Anexos



Fig.8: Dispositivo de control

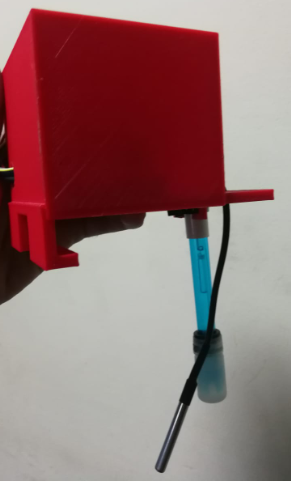


Fig.9 Dispositivo de control (2)

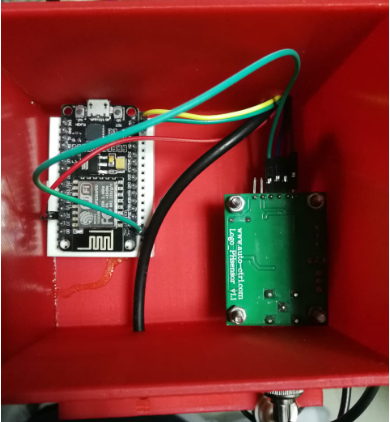


Fig.10 Vista interior del circuito sin fuente de alimentación