PROYECTO EN ARDUINO
MONITOREO AUTOMATIZADO DE LA CALIDAD DEL AGUA EN EL CULTIVO DE TILAPIA ROJA

PRESENTADO POR:
Wilson Darío Sánchez
Juan Fernando Jimenez
Jhon Alberth Palechor

PRESENTADO A:
Carlos Hernán Tobar Arteaga
Profesor Departamento de Telemática

Universidad del Cauca

FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
POPAYÁN
2022

INTRODUCCION

Arduino es una tarjeta electrónica digital y demás es un idioma de programación basada en C++ que es de uso exento. Su hardware está construido por un microcontrolador y asimismo es una de las tarjetas electrónicas más usadas para gestar prototipos que ayudan a allanar algunas acciones en nuestra vida cotidiana. Las instrucciones del lenguaje de Arduino son muy fáciles de conocer y aprovechar, igualmente para personas con poca noción de electrónica y programación. Arduino todavía es conocido como un instrumento de procesamiento digital, a similitud a una computadora. Como tal, tiene medios de puertas y de expectativa digital a los cuales se les puede encadenar: botones, pantallas lcd, teclados, teclados matriciales o en nuestra contingencia sensores digitales [1].

En el presente informe se pretende llevar a cabo la implementación de un sistema automatizado sobre el monitoreo de los principales parámetros establecidos que permiten determinar las condiciones de la calidad de agua en el cultivo de tilapia roja (*Orechromis sp),* en el municipio de Timbío, Departamento del Cauca. En esta primera fase del sistema se llevara a cabo mediante la ayuda del dispositivo electrónico Arduino y su simulación en software PROTEUS que mediante los datos de entrada registrados por los sensores de pH, nivel de oxígeno y temperatura se pueda seguir una serie de instrucciones programadas en el microcontrolador que cumplan con el objetivo principal, el cual consiste en la visualización de los parámetros que permitirán al operario llevar a cabo un control y/o monitoreo para la optimización en la producción de los alevinos.

PLANTEAMIENTO DEL PROBLEMA / NECESIDAD

En el Departamento del Cauca se puede encontrar la tilapia roja (*Orechromis sp)* la cual es de uso comercial para el consumo humano. Esta especie es un híbrido de la tilapia negra (*Orechromis mossambicus*), cuya descendencia es egipcia, siendo la tilapia roja capaz de adaptarse a cualquier clima. La acuicultura de esta especie presenta inconvenientes porque su crecimiento y desarrollo se pueden ver afectados por una falta de optimización en el proceso de crecimiento de los alevinos ya que no se cuenta con un sistema automático para el monitoreo de los parámetros como pH, temperatura, nivel de oxígeno del agua entre otros, lo que implica un incremento en los gastos de producción [2]. De acuerdo esto, se hace necesario llevar a cabo el diseño de un sistema electrónico que pueda suplir esta necesidad que sea de fácil uso, implementación y costo.

DISEÑO DE SOLUCION.

- **Diagrama de bloques.**

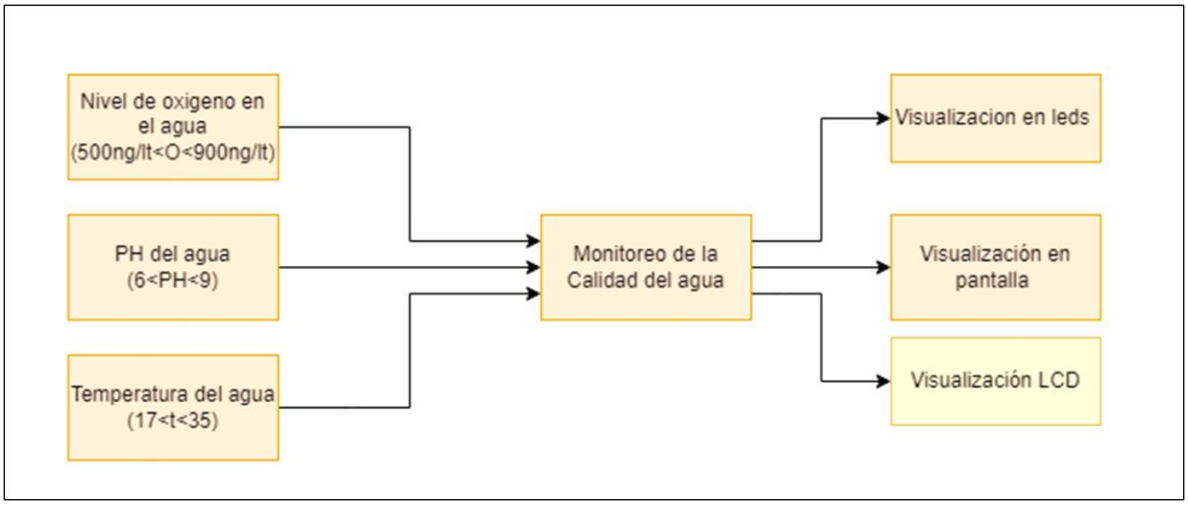


Figura 1. Diagrama de bloques

Como se puede observar anteriormente, se utilizó un diagrama de bloques para tratar de dar una solución al problema de monitoreo de la calidad del agua en la producción de alevinos, llevando a cabo la estrategia “divide y vencerás”, que nos permitió dividir por partes la solución a la problemática correspondiente al monitoreo de los 3 parámetros (pH, nivel de oxígeno y temperatura).

Además, los parámetros ya están establecidos para un rango optimo en el cual los alevinos pueden desarrollarse de manera adecuada, estos estarán aplicados mediante sensores que registren estos datos para que el microcontrolador tome decisiones mediante un diseño programado que permita la visualización del estado de calidad de agua ya sea mediante la visualización en leds, terminal virtual o visualización en una pantalla LCD, sin embargo este diseño puede ir optimizándose según los requerimientos que se vayan presentando en el transcurso de las indicaciones dadas por el docente.

- **Módulos empleados y configuraciones realizadas.**

Temperatura. De acuerdo con el diagrama de la figura 1, para el bloque de calidad de agua se utilizó el módulo de Arduino, primero llevando a cabo por separado, es decir 3 módulos para cada bloque de entrada (temperatura, pH, nivel de oxígeno) y de esta manera verificar su funcionamiento de manera dividida para cada parámetro, luego se llevará a cabo la unificación en un solo modulo que se encargue de tomar las decisiones de una manera sincronizada.

Para el diseño en PROTEUS correspondiente al parámetro de entrada de temperatura del agua se utilizaron los siguientes componentes:

- Sensor de temperatura LM35
- Arduino Uno R3 V1.0
- Leds (D1, D2) Rojo y verde
- Terminal virtual

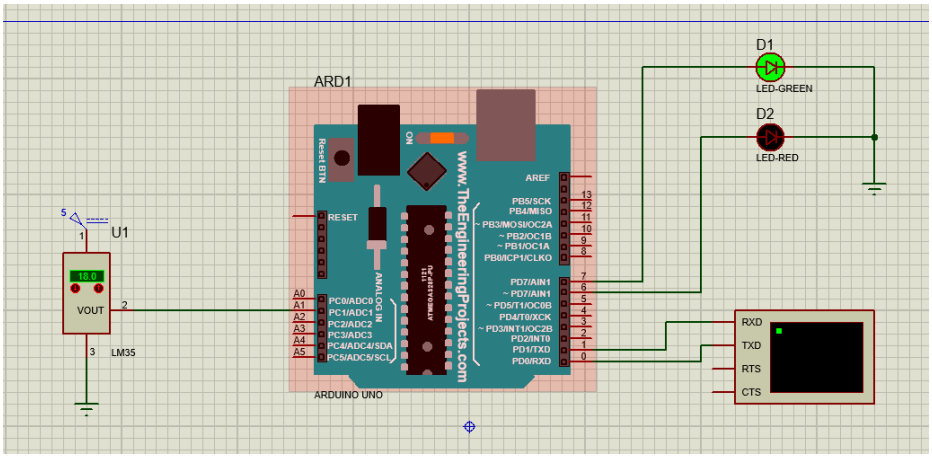
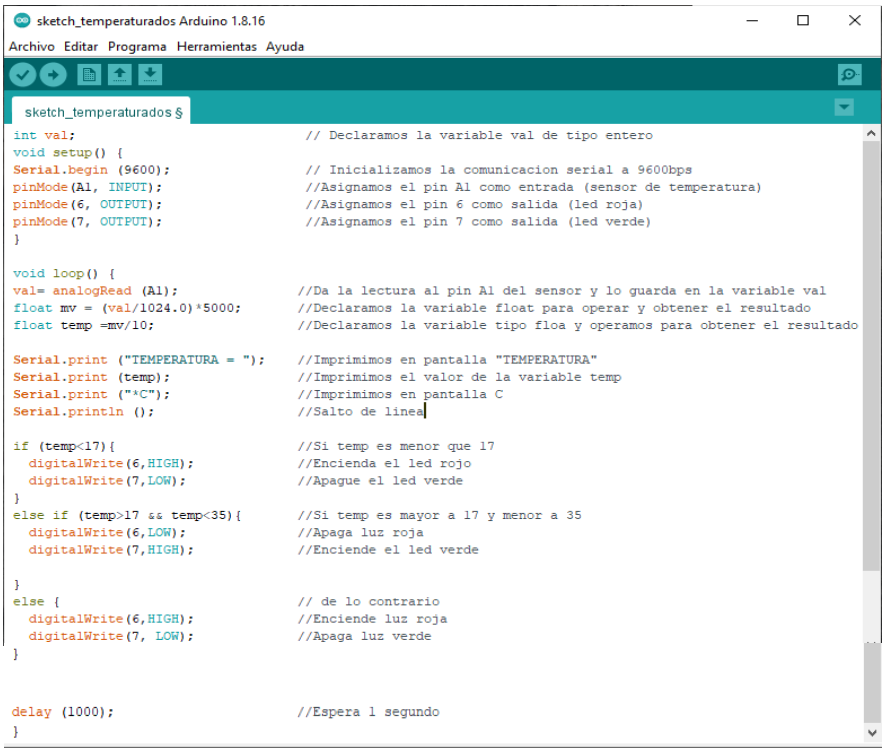


Figura 2. Componentes utilizados parámetro temperatura.

Para este módulo de Arduino se aplicaron las siguientes líneas de código:




```
int val;                          // Declaramos la variable val de tipo entero
void setup() {
Serial.begin (9600);              // Inicializamos la comunicacion serial a 9600bps
pinMode(A1, INPUT);               //Asignamos el pin A1 como entrada (sensor de temperatura)
pinMode(6, OUTPUT);               //Asignamos el pin 6 como salida (led roja)
pinMode(7, OUTPUT);               //Asignamos el pin 7 como salida (led verde)
}

void loop() {
val= analogRead (A1);            //Da la lectura al pin A1 del sensor y lo guarda en la variable val
float mv = (val/1024.0)*5000;     //Declaramos la variable float para operar y obtener el resultado
float temp =mv/10;                //Declaramos la variable tipo floa y operamos para obtener el resultado

Serial.print ("TEMPERATURA = ");  //Imprimimos en pantalla "TEMPERATURA"
Serial.print (temp);              //Imprimimos el valor de la variable temp
Serial.print ("*C");              //Imprimimos en pantalla C
Serial.println ();                //Salto de linea

if (temp<17){                     //Si temp es menor que 17
  digitalWrite(6,HIGH);           //Encienda el led rojo
  digitalWrite(7,LOW);            //Apague el led verde
}
else if (temp>17 && temp<35){     //Si temp es mayor a 17 y menor a 35
  digitalWrite(6,LOW);            //Apaga luz roja
  digitalWrite(7,HIGH);           //Enciende el led verde

}
else {                            // de lo contrario
  digitalWrite(6,HIGH);           //Enciende luz roja
  digitalWrite(7, LOW);           //Apaga luz verde
}


delay (1000);                     //Espera 1 segundo
}
```

Figura 3. Diseño del codigo para parametro de temperatura del agua.

El diseño de este codigo nos permite establecer mediante una programacion, los rangos adecuados para la temperatura y ademas los que no corresponden a un valor admisible y tal como se encuentra en los comentarios para cada linea, si la temperatura es menor que 17° y mayor que 35°, se enciende el led rojo y se apaga el verde, si por el contrario la temperatura es mayor a 17 y menor que 35 se apaga el led rojo y se enciende el verde en respuesta a los datos obtenidos por el sensor de temperatura. Entonces el led verde significa que es una temperatura adecuada y el led rojo indica una temperatura que no es adecuacada para las tilapias.

PH. Para el PH del agua fue necesario descargar primero la librería correspondiente para el sensor respectivo (pHMeterLibrary) y en total se utilizaron los siguientes componentes:

- Sensor de pH Meter
- Arduino Uno R3 V1.0
- Terminal Virtual

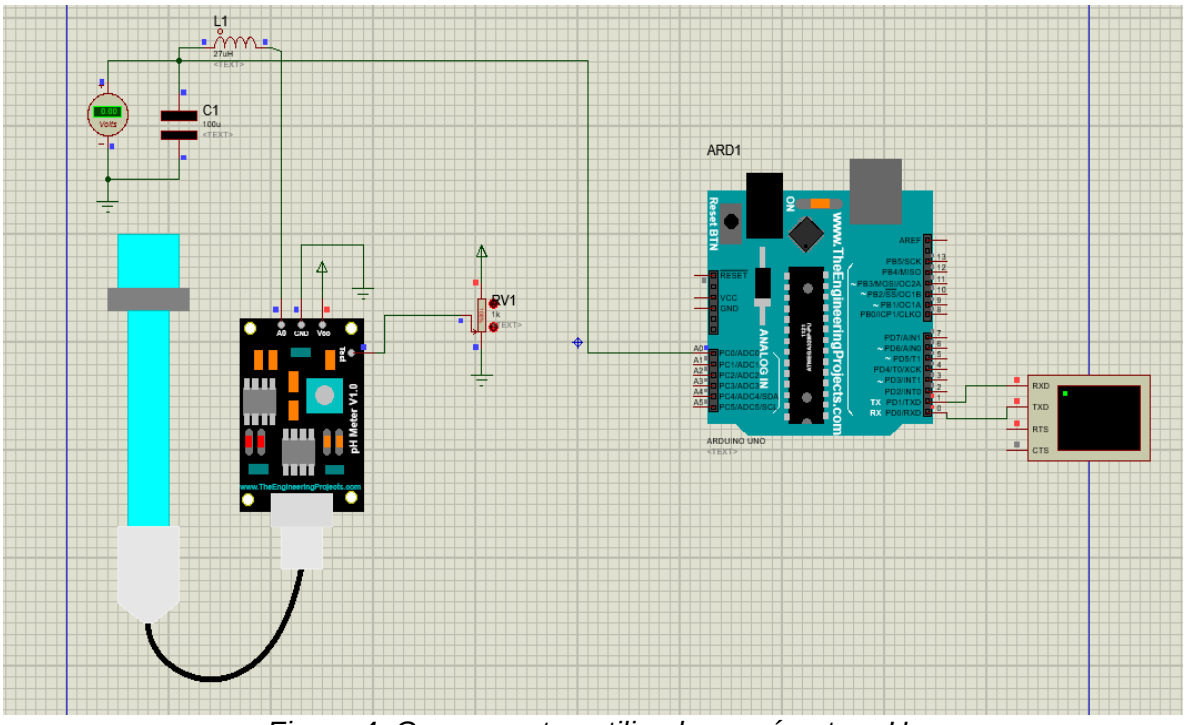


Figura 4. Componentes utilizados parámetro pH.

Para este módulo en Arduino se aplicaron las siguientes líneas de código:

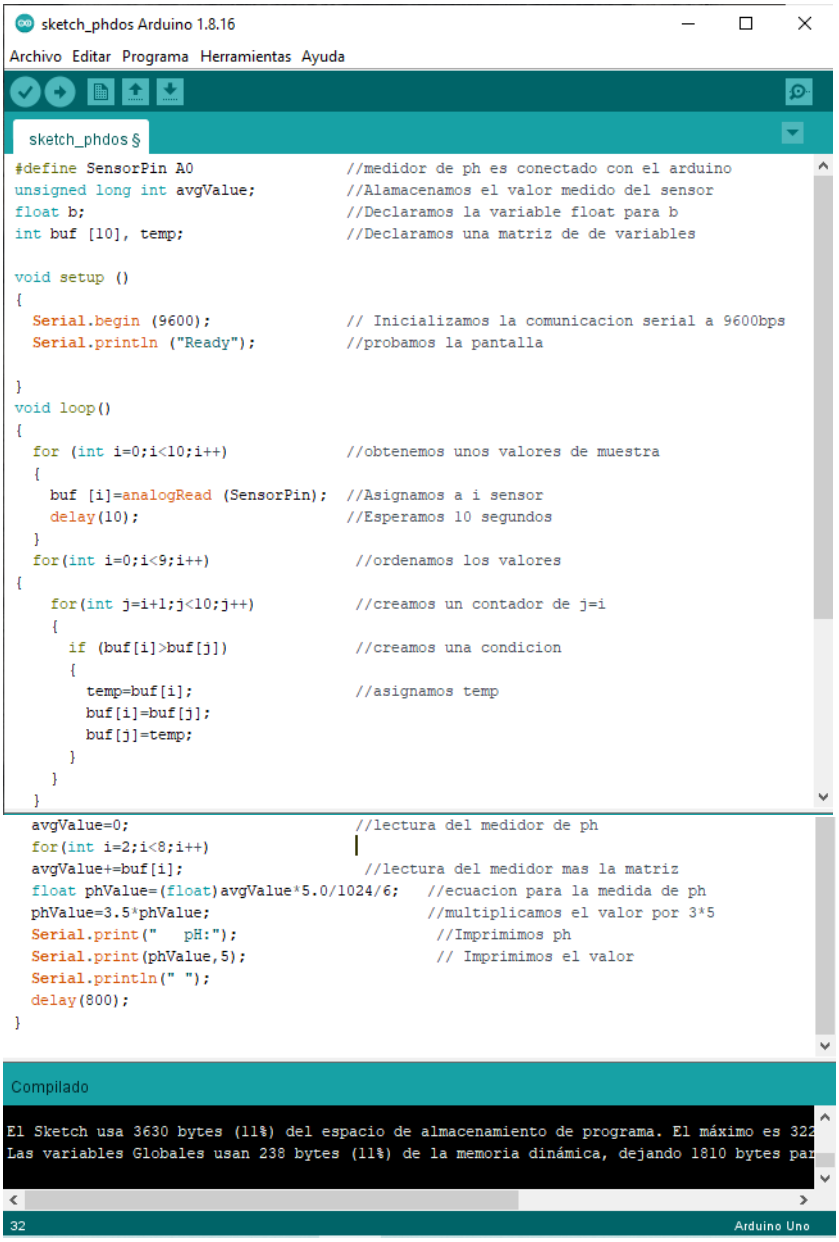


Figura 5. Diseño del codigo para parametro de pH del agua.

Para el realizar el diseño de este código, se requirió primeramente añadir la librería a PROTEUS de un medidor de pH mediante “pHMeterLibrary” ya que esta no se encuentra instalada por defecto en el programa. El potenciómetro nos ayudará a dar algunos valores de pH ya que este será tomado como testador por el medidor. En esta simulación mediante un terminal virtual, nos permitirá observar los resultados en una pantalla en Proteus determinado mediante la implementación programada de un bucle que toma diferentes valores y

llevándolos a un promedio, cada línea de código cuenta con los cometarios respectivos para un mejor entendimiento del mismo.

Nivel de Oxígeno. Para el medidor de nivel de Oxígeno en el agua, se tuvieron inconvenientes para encontrar un componente adecuado para este parámetro, por esto se toma la decisión de que este medidor tomara los valores respectivos mediante un potenciómetro ajustable. Los componentes utilizados fueron:

- Potenciómetro ajustable (actúa como sensor de nivel de oxígeno en el agua)
- Arduino Uno-Simulino Uno V4.0
- Display LCD
- Multiplexor

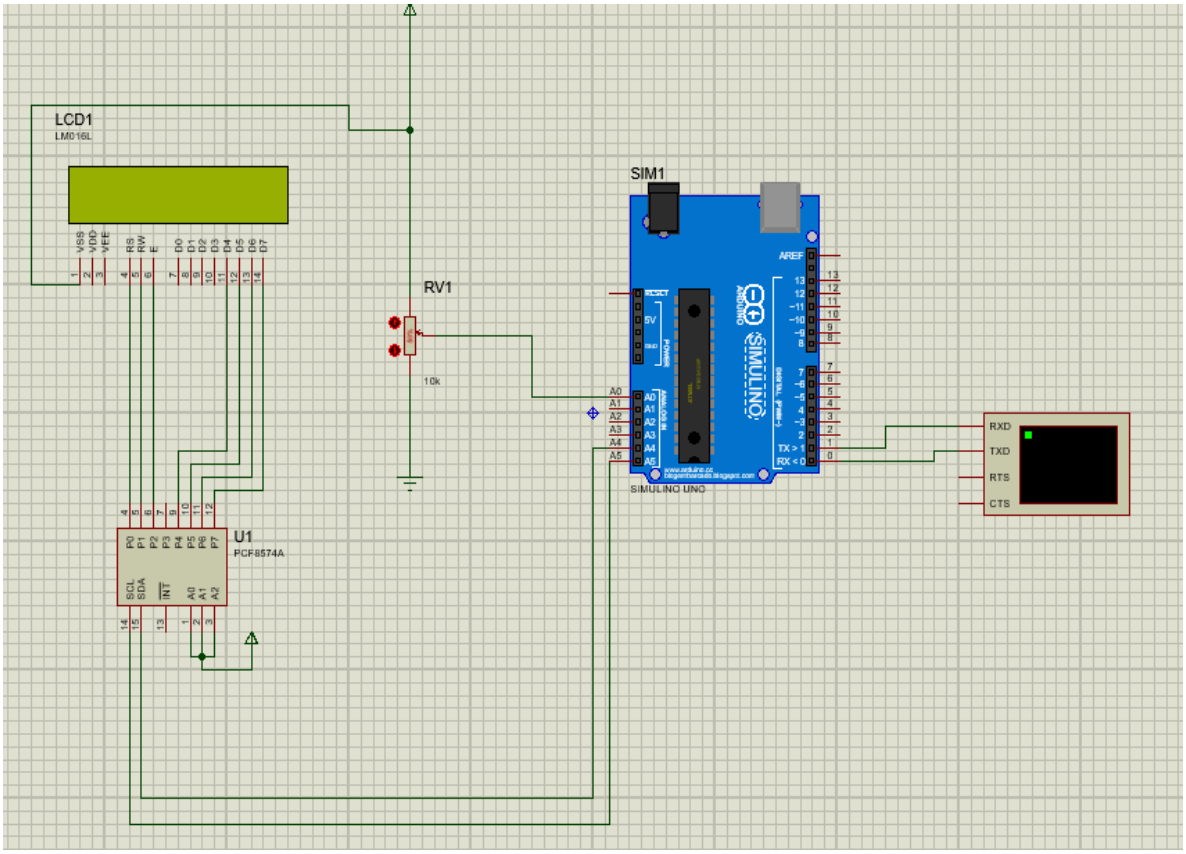


Figura 6. Componentes utilizados parámetro medidor de nivel de oxígeno del agua.

Para este módulo Arduino se aplicó las siguientes líneas de código:

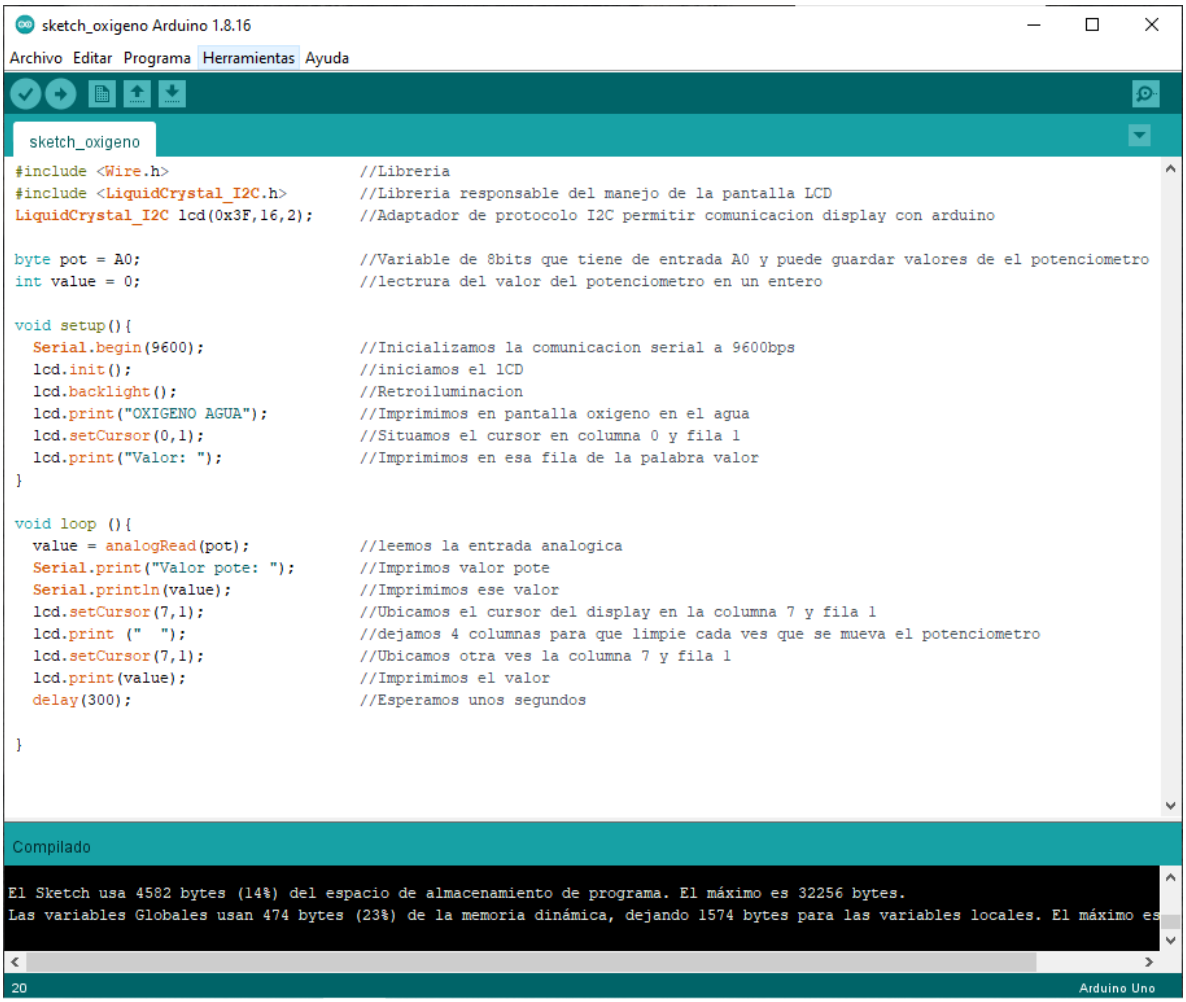


Figura 7. Diseño del codigo para parametro de nivel de oxigeno.

Parael diseño de este codigo, es necesario primero añadir la librería en arduino correspondiente al LCD “LiquidCrystal_I2C” que permite la visualizacion de los resultados para este parametro. Primero se inicializa el LCD activando la retroiluminacion imprimiento en pantalla “Oxigeno en el Agua” tan como se indica en la columna 0 y fila 1 impirmiendo la parabra “valor”. Luego mediante un bucle se lee la entrada analogica del potenciometro imprimiendo el valor de este de tal manera que el operario observar la calidad de oxigeno del agua en el display LCD.

PRUEBAS FUNCIONALES Y ANALISIS DE RESULTADOS

- Para temperatura del agua:

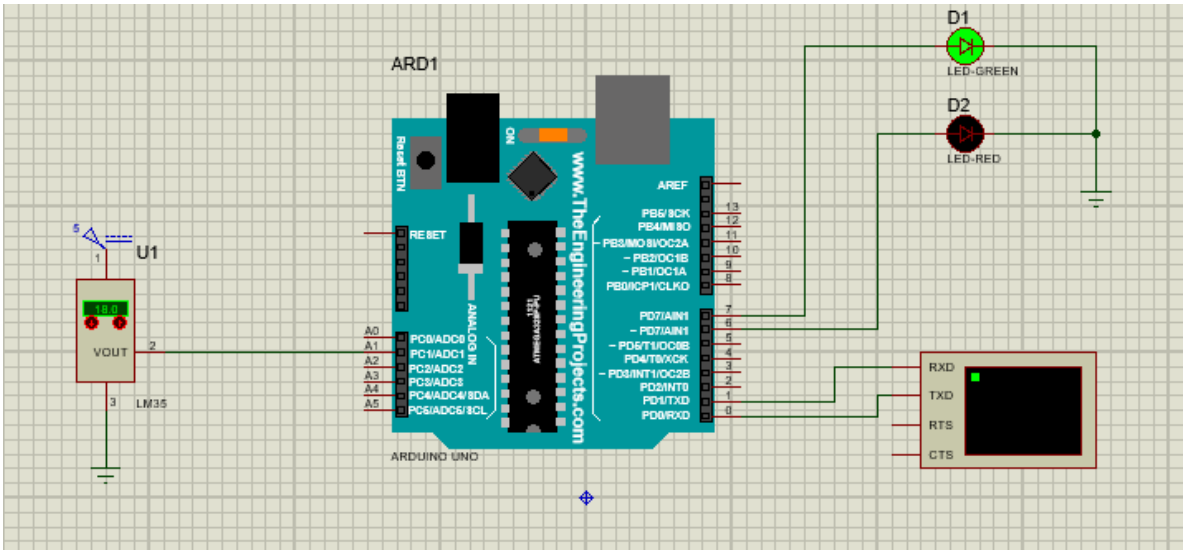


Figura 8. Prueba funcional con parámetro temperatura del agua (valor adecuado).

Como se puede observar la figura 8, se tiene una temperatura de 18°C la cual es ideal para las tilapias, entonces de acuerdo con la respuesta del microcontrolador este enciende el led de color verde indicando que la temperatura del agua es óptima.

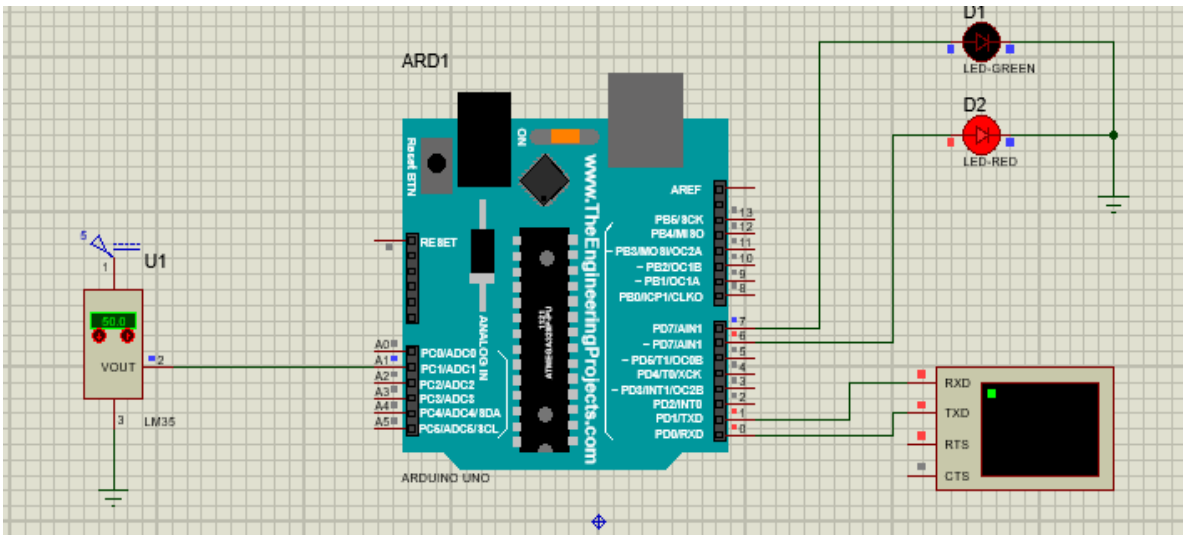


Figura 9. Prueba funcional con parámetro temperatura del agua (valor no adecuado).

Como se puede observar en la figura 9, por el contrario, en este caso se tiene una temperatura de 50°C la cual es no ideal para las tilapias, por ello debe encenderse

el led rojo que indique al operario una advertencia en el monitoreo de este parámetro para su posterior corrección.

- **Para el pH del agua:**

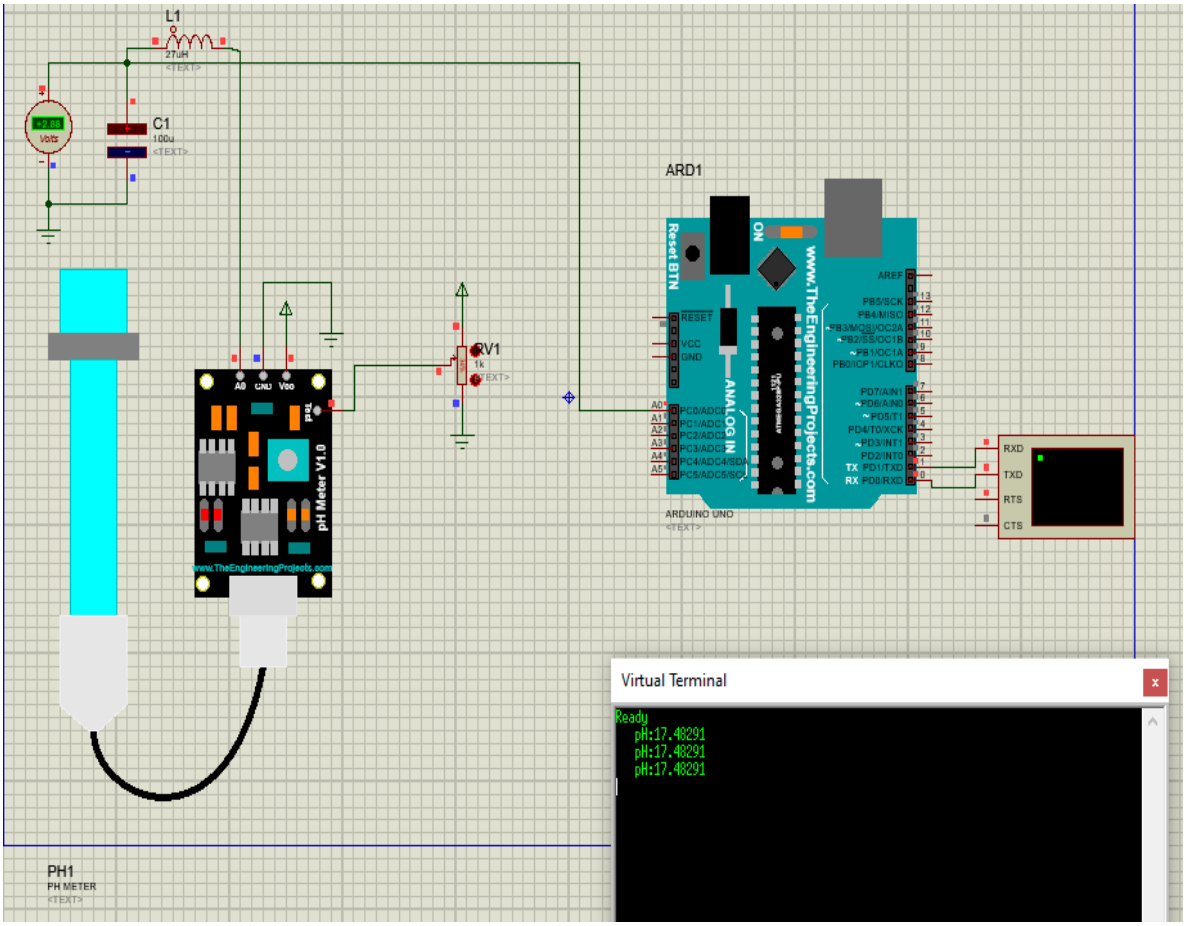


Figura 10. Prueba funcional con parámetro pH del agua.

Como se puede observar en la figura 10, se encuentra el medidor de pH con el potenciómetro variable, el cual, al ser llevado a la simulación, da un resultado de 17.48 el cual se visualiza en el terminal virtual. Este valor se encontraría dentro del rango adecuado.

- **Para nivel de oxígeno en el agua:**

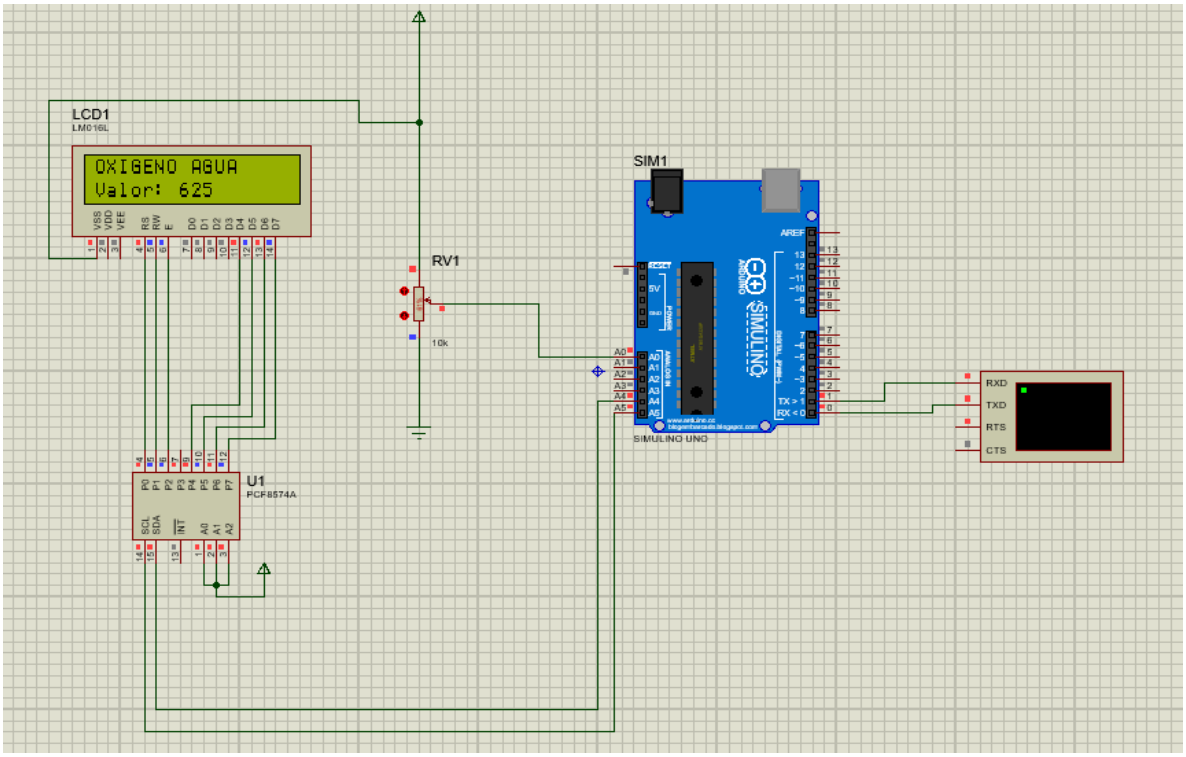


Figura 11. Prueba funcional con parámetro nivel de oxígeno del agua.

Como se puede podemos observar en la simulación mostrada en la figura 11, al ajustar un valor mediante el potenciómetro variable se visualiza mediante la ayuda del multiplexor que permite codificar la entrada y es llevada al LCD arrojando el valor correspondiente al nivel de oxígeno en el agua. En este caso se tomó un valor de 625g/litro lo cual indica un valor adecuado.

- **Integración de todos los sensores controlados por un solo modulo Arduino.**

Posteriormente se procede a realizar la integración de todos los módulos para conformar uno que realice todas las funciones de los sensores utilizando un microcontrolador SIMULINO UNO como se observa a continuación.

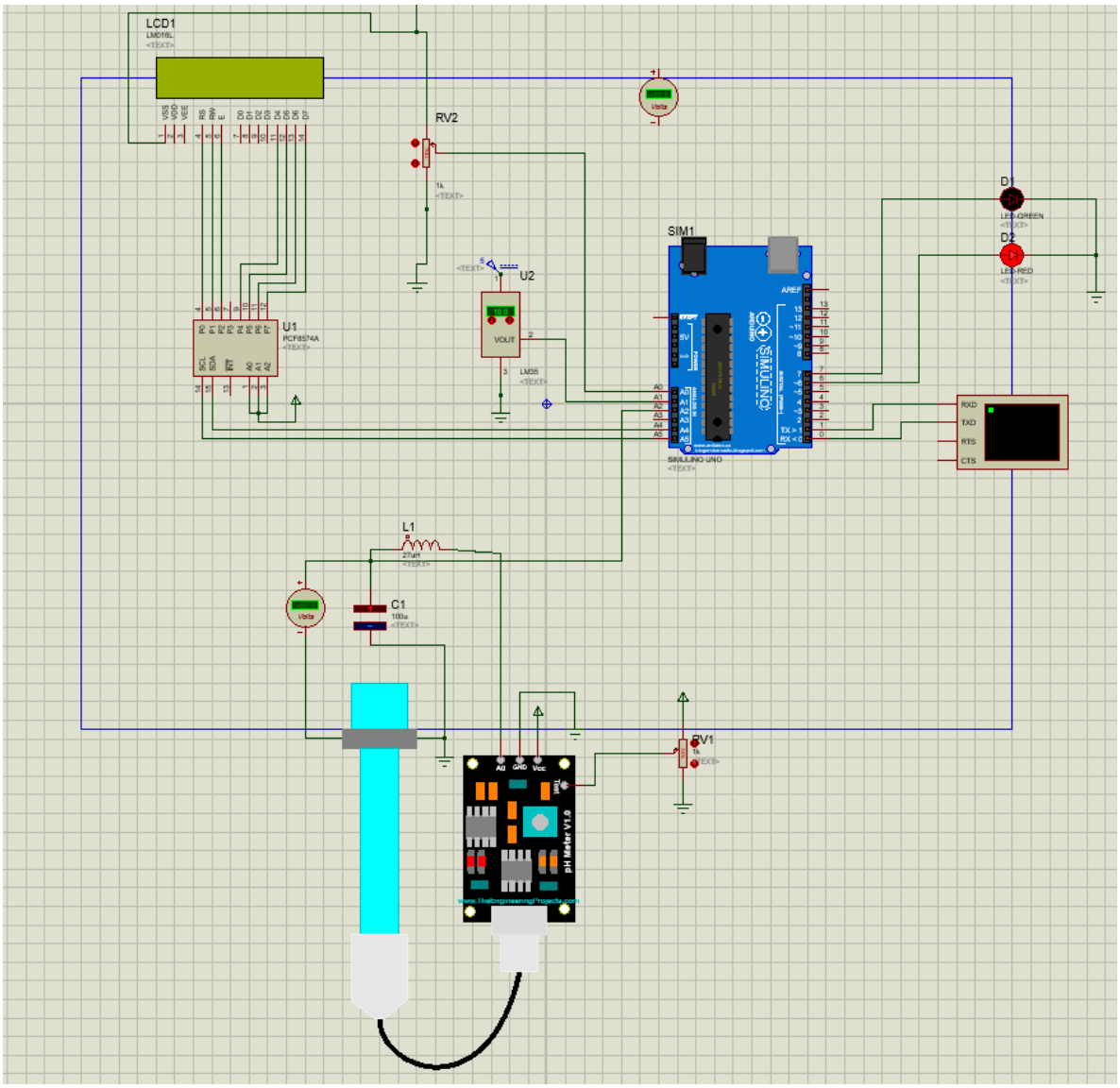


Figura 12. Integración de todos los sensores controlados por un solo modulo Arduino.

Como se observa en la figura anterior, sen tomaron como entradas para el ingreso de datos de los sensores la entrada A0 para el potenciómetro que permite seleccionar el valor del nivel de oxígeno, A1 para el sensor de temperatura, A2 sensor de pH, A4 y A5 como salida de datos dirigidos para ser codificados y mostrados en el LCD, salida 6 y 7 para los leds verde y rojo y salida 1(TX) y 0(RX) para el terminal virtual.

El módulo Arduino se programó, primero tomando el código ya realizado anteriormente por separado y luego realizando la unión de todas las líneas de código necesarias, se organizó de tal manera que cada respuesta de los sensores

fuera llamada mediante funciones que nos permitieron tener un orden claro para la visualización de los datos, tal como se observa a continuación.

sketch_completo §

```
//Oxigeno
#include <Wire.h>                       //Libreria
#include <LiquidCrystal_I2C.h>          //Libreria responsable del manejo de la pantalla LCD
LiquidCrystal_I2C lcd(0x3F,16,2);       //Adaptador de protocolo I2C permitir comunicacion display con arduino
  int pot = A0;                          //Variable de 8bits que tiene de entrada A0 y puede guardar valores de el potenciometro
  int value = 0;                         //lectrura del valor del potenciometro en un entero

//PH
#define SensorPin A2                   //medidor de ph es conectado con el arduino
  unsigned long int avgValue;           //Alamacenamos el valor medido del sensor
  float b;                              //Declaramos la variable float para b
  int buf [10], temp;                   //Declaramos una matriz de de variables

//Temperatura
  int val;                              // Declaramos la variable val de tipo entero

  void setup() {
  //Oxigeno

  pinMode(A4, OUTPUT);
  pinMode(A5, OUTPUT);
  Serial.begin(9600);                   //Inicializamos la comunicacion serial a 9600bps
  lcd.init();                           //iniciamos el lCD
  lcd.backlight();                      //Retroiluminacion
  lcd.print("OXIGENO AGUA");            //Imprimimos en pantalla oxigeno en el agua
  lcd.setCursor(0,1);                   //Situamos el cursor en columna 0 y fila 1
  lcd.print("Valor: ");                 //Imprimimos en esa fila de la palabra valor

//PH
  Serial.println ("Ready");           //probamos la pantalla

 //Temperatura

 pinMode(A1, INPUT);                        //Asignamos el pin A1 como entrada (sensor de temperatura)
 pinMode(6, OUTPUT);                        //Asignamos el pin 6 como salida (led roja)
 pinMode(7, OUTPUT);                        //Asignamos el pin 7 como salida (led verde)

 }
 void loop() {
   //Oxigeno
   medirOxigeno();
   delay(300);

   //PH
   medirPh();
   delay(300);

   //Temperatura
   medirTemperatura();
   delay (300);
 }

 //Funcion que mide el oxigeno
 void medirOxigeno(){
   value = analogRead(pot);             //leemos la entrada analogica
   delay(100);
   Serial.print("Valor pot: ");        //Imprimos valor pote
   Serial.println(value);               //Imprimimos ese valor
   lcd.setCursor(7,1);                  //Ubicamos el cursor del display en la columna 7 y fila 1
   lcd.print ("  ");                    //dejamos 4 columnas para que limpie cada ves que se mueva el potenciometro
   lcd.setCursor(7,1);                  //Ubicamos otra ves la columna 7 y fila 1
   lcd.print(value);                    //Imprimimos el valor
 }//Fin funcion medir oxigeno
```

```
//Funcion que mide el Ph
void medirPh(){
   for (int i=0;i<10;i++)             //obtenemos unos valores de muestra
    {
      buf [i]=analogRead (SensorPin);  //Asignamos a i sensor
      delay(10);                       //Esperamos 10 segundos
    }
    for(int i=0;i<9;i++)                //ordenamos los valores
    {
      for(int j=i+1;j<10;j++)           //creamos un contador de j=i
      {
        if (buf[i]>buf[j])              //creamos una condicion
        {
          temp=buf[i];                  //asignamos temp
          buf[i]=buf[j];
          buf[j]=temp;
        }
      }
    }
    avgValue=0;                         //lectura del medidor de ph
    for(int i=2;i<8;i++)
      avgValue+=buf[i];                    //lectura del medidor mas la matriz

    float phValue=(float)avgValue*5.0/1024/6;   //ecuacion para la medida de ph
    phValue=3.5*phValue;                        //multiplicamos el valor por 3*5
    Serial.print("   pH:");                    //Imprimimos ph
    Serial.print(phValue,5);                   // Imprimimos el valor
    Serial.println(" ");
}//Fin funcion medir Ph

//Funcion que mide la Temperatura

void medirTemperatura(){
  val= analogRead (A1);               //Da la lectura al pin A1 del sensor y lo guarda en la variable val
  float mv = (val/1024.0)*5000;       //Declaramos la variable float para operar y obtener el resultado
  float temp =mv/10;                  //Declaramos la variable tipo floa y operamos para obtener el resultado

  Serial.print ("TEMPERATURA = ");    //Imprimimos en pantalla "TEMPERATURA"
  Serial.print (temp);                //Imprimimos el valor de la variable temp
  Serial.print ("*C");                //Imprimimos en pantalla C
  Serial.println ();                  //Salto de linea

  if (temp<17){                       //Si temp es menor que 17
    digitalWrite(6,HIGH);             //Encienda el led rojo
    digitalWrite(7,LOW);              //Apague el led verde
  }
  else if (temp>17 && temp<35){       //Si temp es mayor a 17 y menor a 35
    digitalWrite(6,LOW);              //Apaga luz roja
    digitalWrite(7,HIGH);             //Enciende el led verde
  }
  else {                              // de lo contrario
    digitalWrite(6,HIGH);             //Enciende luz roja
    digitalWrite(7, LOW);             //Apaga luz verde
  }
}//Fin funcion medir Temperatura
```

Compilado

Figura 13. Líneas de código control mediante un solo modulo Arduino.

Al llevar a cabo la simulación, se observa la pantalla del terminar virtual como este comienza a generar la visualización de los datos suministrados por cada sensor, por ejemplo, para el sensor de nivel de oxígeno, el potenciómetro se encarga de fijar un valor en este caso de 799g/lt, lo cual indicaría que este es un valor adecuado. En cuanto al sensor de temperatura como ejemplo se encuentra fijado en 10°C, el led rojo se enciende, indicando que no es una temperatura adecuada en el agua para

los alevinos y finalmente para el sensor de pH se encuentra variando en valores mayores a 9, indicando que no es un valor adecuado en el agua y debe ser revisado por el operario encargado.

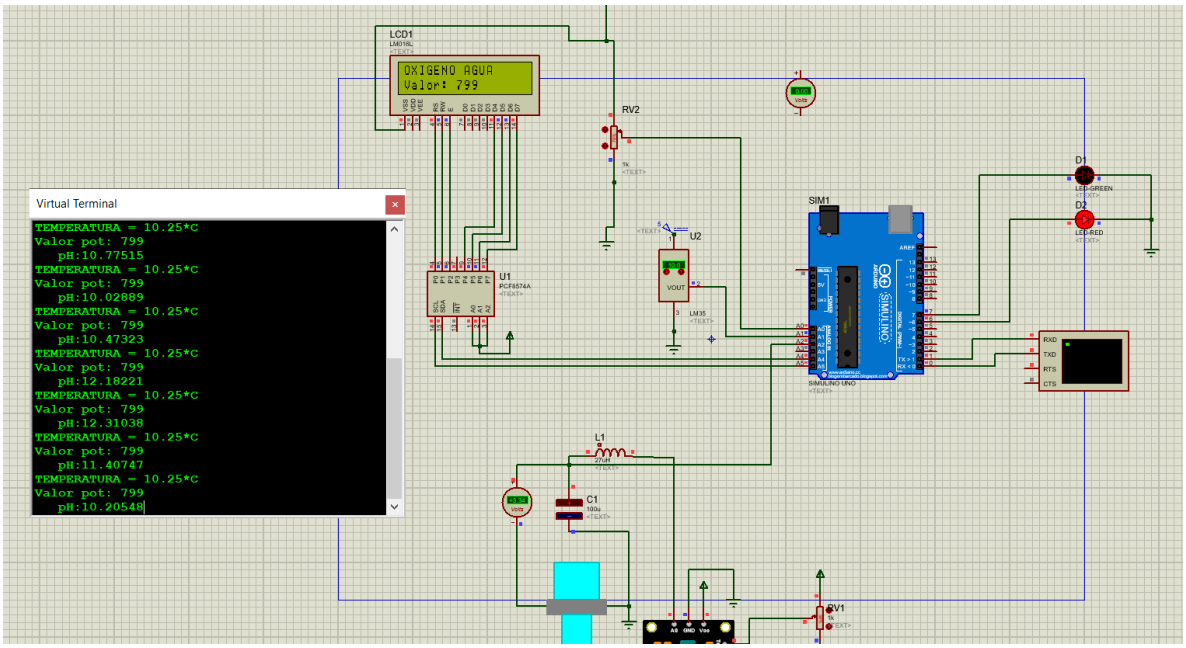


Figura 14. Simulación del circuito y resultados obtenidos.

- **Aplicación del modulo Bluetooth e interfaz móvil.**

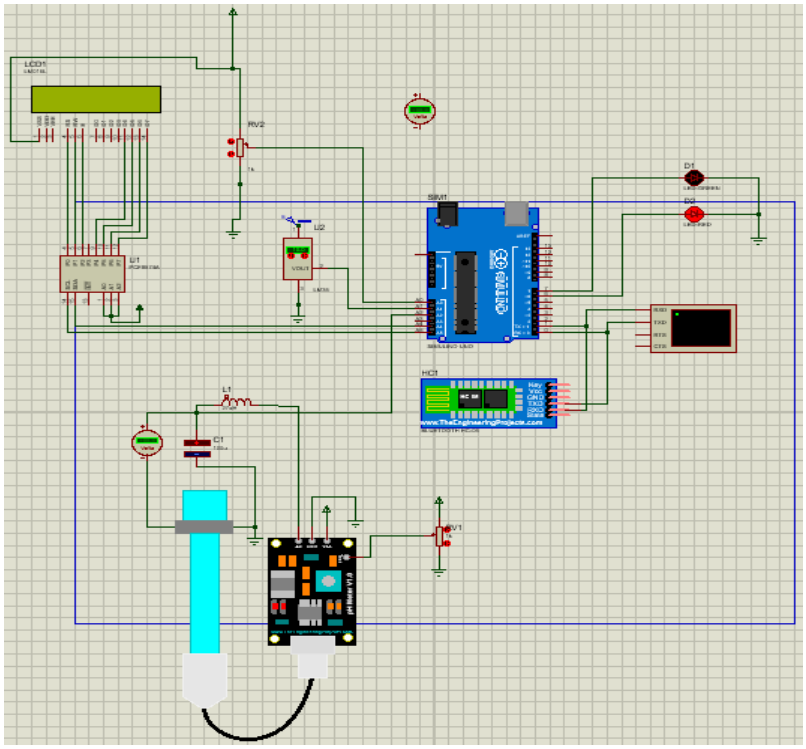


Figura 15. Implementación de modulo Bluetooth.

En Proteus podemos observar los componentes que nos permiten mediante la simulación los registros de los parámetros de control de la calidad de agua. Para la conexión Bluetooth se incorpora al circuito un elemento electrónico (BLUETOOTH HC-05) que nos permite enlazar el dispositivo móvil al equipo PC y de esta forma se transmite la información de los datos recopilados de los sensores.

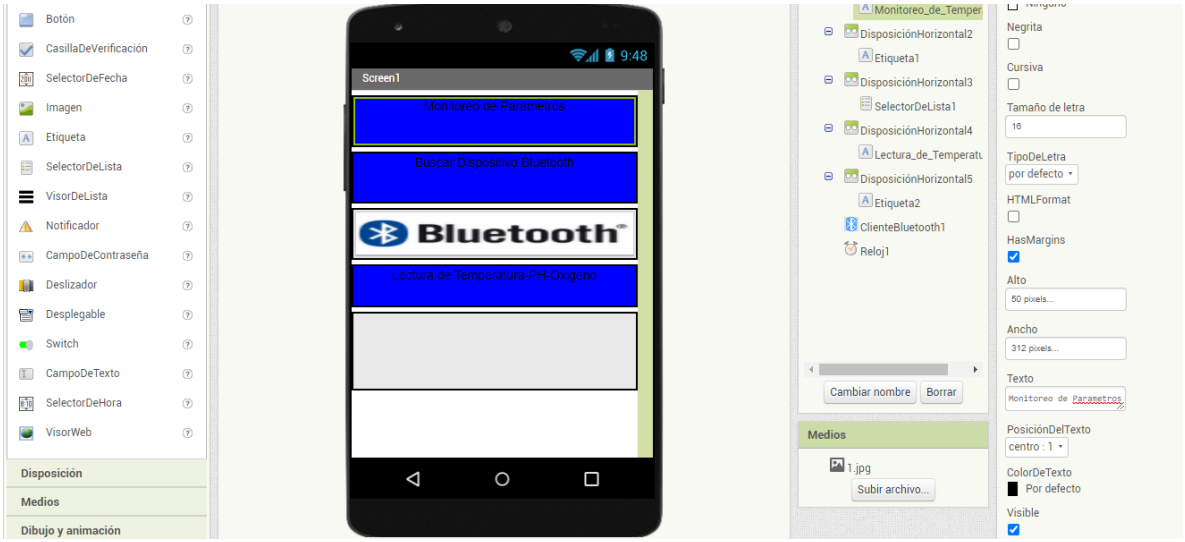


Figura 16. Desarrollo de aplicación móvil mediante MIT App Inventor.

Para el desarrollo de una aplicación móvil que le permita al usuario una visualización práctica de los parámetros de Temperatura, PH y oxígeno, se utiliza **MIT App Inventor,** que es un entorno de desarrollo de software creado por Google Labs para la elaboración de aplicaciones destinadas al sistema operativo Android.

En la figura se puede observar una interfaz que le permite al usuario mediante la conectividad vía Bluetooth acceder a los parámetros para el control de la calidad de agua.

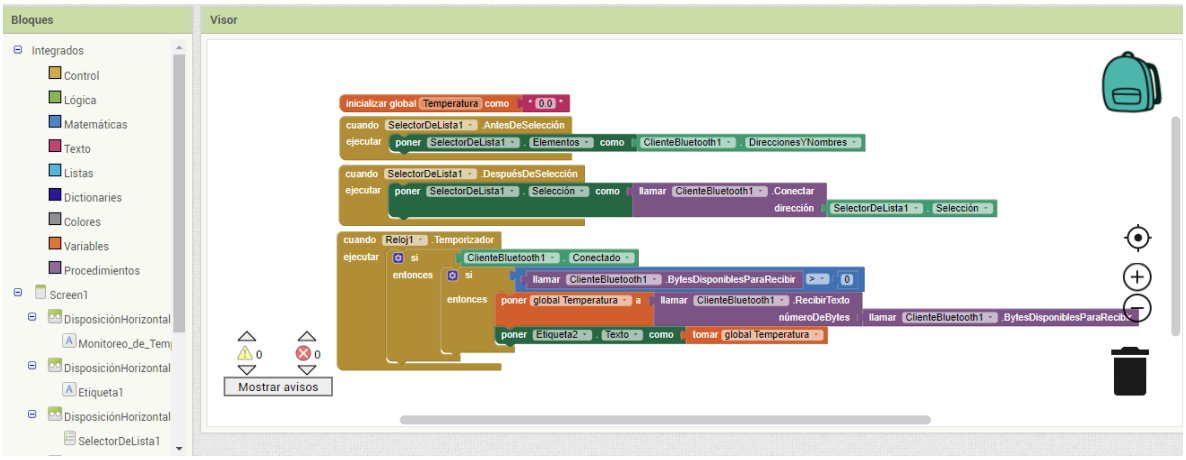


Figura 17. Programación por bloques.

App inventor cuenta con una herramienta importante de programación por bloques. Estos bloques están hechos con elementos comunes a la mayoría de los lenguajes de programación existentes. Se colocan bloques para construir bucles, condiciones, variables, etc. que permiten pensar lógicamente y solucionar los problemas de forma metódica.

La estructura de la programación en el primer bloque empieza con la inicialización global de una variable con un valor inicial de “0.0”, seguidamente en la segunda parte la integración de bloques nos permite la localización de los dispositivos Bluetooth cercanos, en la tercera parte de bloques después de la localización de los dispositivos cercanos nos posibilita seleccionar el dispositivo al cual se va a realizar la conexión, y por último en la cuarta parte de adhesión de bloques tras la vinculación del dispositivo vía Bluetooth, mediante una estructura condicional si registran a la entrada una serie de Bytes disponibles mayores a cero, se asigna en forma de texto los valores de Temperatura, pH y oxígeno en la etiqueta de la interfaz gráfica del usuario.

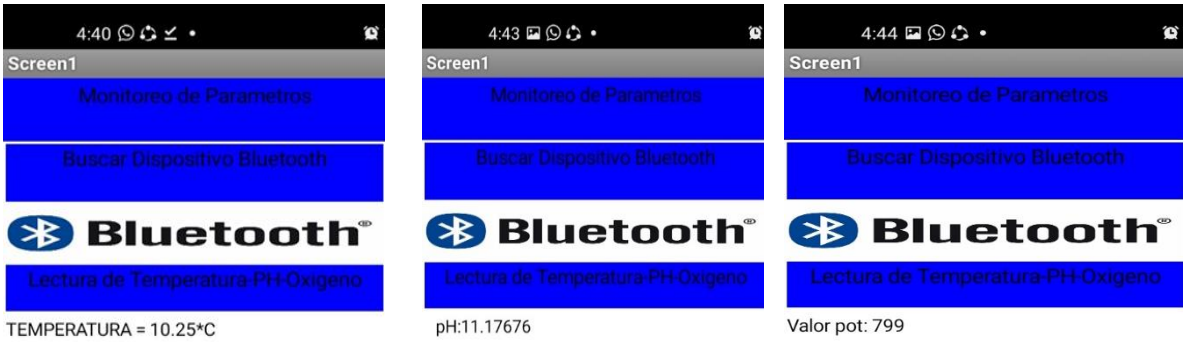


Figura 18. Visualización en App para el Monitoreo de parámetro de Calidad de Agua.

Después de haber descargado la app en el dispositivo móvil, se realiza la descarga del archivo mediante el escaneo de un código QR, que al abrirlo nos permite obtener acceso a la interfaz de usuario, como se puede apreciar en la figura.

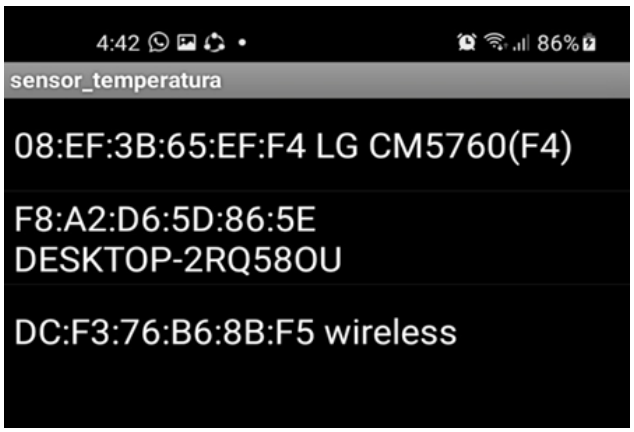


Figura 19. Lista Dispositivos Bluetooth

Al accionar la opción de bluetooth nos despliega la cantidad de dispositivos cercanos al cual se puede vincular, de esta manera elegimos la correspondiente al equipo donde se encuentra la simulación en Proteus.

De esta manera se finaliza esta fase de implementación del sistema que permitirá monitorizar automáticamente los parámetros del agua en el cultivo de tilapia roja, proporcionando datos importantes al momento de llevar a cabo esta actividad ya que, de estos parámetros de calidad, se podrá obtener un mejor nivel de producción. Es importante aclarar que este no será el modelo definitivo, ya que se planea ir optimizando el funcionamiento del sistema electrónico, como por ejemplo mejorar la interfaz de la app o también ingresar un módulo wifi para una mejor conexión que permita llevar a cabo este monitoreo de manera remota por internet y esto pueda ofrecer al usuario obtener estos datos de una manera más sencilla y práctica.

REPOSITORIO EN GITHUB

Primero es necesario ubicar los archivos correspondientes a simulación individual para cada uno de los sensores y el circuito completo en Proteus, sketchs de Arduino de cada uno de los sensores, y sketchs del código unificado, esto organizado por carpetas y realizado mediante Git (administrador de repositorios).

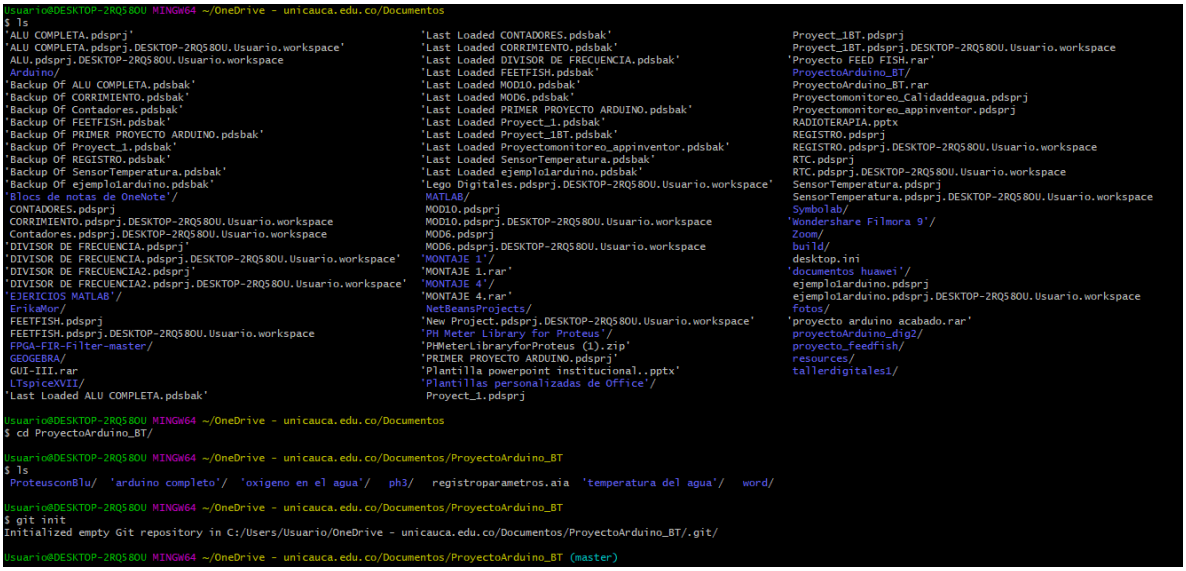


Figura 20. Administrador de repositorios

Luego se debe crear el repositorio local y para ello se ingresa en la consola “Git init”, en el cual aparece en el equipo junto a las carpetas, la carpeta oculta “Git”, indicando la inicialización del repositorio local.

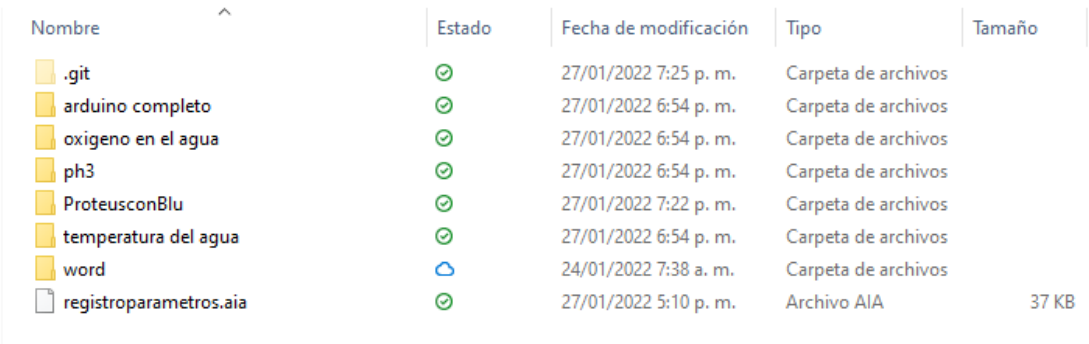


Figura 21. Inicialización del repositorio local

Luego en la consola se ingresa “Git Status” que muestra los archivos subrayados en rojo que me indican que se deben inicializar en el repositorio local y además me sugiere que no hay “Commit”.

Es necesario agregar estos archivos ingresando “Git add .” al repositorio local. Al presionar enter los archivos quedan asignados.

```
Usuario@DESKTOP-2RQ58OU MINGW64 ~/OneDrive - unicauca.edu.co/Documentos/ProyectoArduino_BT
$ git init
Initialized empty Git repository in C:/Users/Usuario/OneDrive - unicauca.edu.co/Documentos/ProyectoArduino_BT/.git/

Usuario@DESKTOP-2RQ58OU MINGW64 ~/OneDrive - unicauca.edu.co/Documentos/ProyectoArduino_BT (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ProteusconBlu/
        arduino completo/
        oxigeno en el agua/
        ph3/
        registroparametros.aia
        temperatura del agua/
        word/

nothing added to commit but untracked files present (use "git add" to track)

Usuario@DESKTOP-2RQ58OU MINGW64 ~/OneDrive - unicauca.edu.co/Documentos/ProyectoArduino_BT (master)
$ git add .
warning: LF will be replaced by CRLF in arduino completo/sketch_completo/sketch_completo.ino.with_bootloader.standard.hex.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in oxigeno en el agua/ARDUINO/sketch_oxigeno/sketch_oxigeno.ino.with_bootloader.standard.hex.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in ph3/sketch_phdos/sketch_phdos.ino.with_bootloader.standard.hex.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in temperatura del agua/ARDUINO/sketch_temperaturados/sketch_temperaturados.ino.with_bootloader.standard.hex.
The file will have its original line endings in your working directory

Usuario@DESKTOP-2RQ58OU MINGW64 ~/OneDrive - unicauca.edu.co/Documentos/ProyectoArduino_BT (master)
```

Figura 22. Adición de archivos al repositorio local.

Posteriormente, se debe iniciar el “Commit” junto con un mensaje, ingresando la siguiente línea “Git commit -m “Commit inicial””

```
Usuario@DESKTOP-2RQ58OU MINGW64 ~/OneDrive - unicauca.edu.co/Documentos/ProyectoArduino_BT (master)
$ git commit -m "Commit inicial"
[master (root-commit) fce12af] Commit inicial
 30 files changed, 3305 insertions(+)
 create mode 100644 ProteusconBlu/Last Loaded Proyectomonitoreo_Calidaddeagua.pdsbak
 create mode 100644 ProteusconBlu/Proyectomonitoreo_Calidaddeagua.pdsprj
 create mode 100644 ProteusconBlu/Proyectomonitoreo_Calidaddeagua.pdsprj.DESKTOP-2RQ58OU.Usuario.workspace
 create mode 100644 arduino completo/sketch_completo/sketch_completo.ino
 create mode 100644 arduino completo/sketch_completo/sketch_completo.ino.standard.hex
 create mode 100644 arduino completo/sketch_completo/sketch_completo.ino.with_bootloader.standard.hex
 create mode 100644 oxigeno en el agua/ARDUINO/sketch_oxigeno.rar
 create mode 100644 oxigeno en el agua/ARDUINO/sketch_oxigeno/sketch_oxigeno.ino
 create mode 100644 oxigeno en el agua/ARDUINO/sketch_oxigeno/sketch_oxigeno.ino.standard.hex
 create mode 100644 oxigeno en el agua/ARDUINO/sketch_oxigeno/sketch_oxigeno.ino.with_bootloader.standard.hex
 create mode 100644 oxigeno en el agua/PROTEUS/Backup Of Nuevo proyecto.pdsbak
 create mode 100644 oxigeno en el agua/PROTEUS/Last Loaded Nuevo proyecto.pdsbak
 create mode 100644 oxigeno en el agua/PROTEUS/Nuevo proyecto.pdsprj
 create mode 100644 oxigeno en el agua/PROTEUS/Nuevo proyecto.pdsprj.DESKTOP-B7JNHU9.jhon9.workspace
 create mode 100644 ph3/Last Loaded Proteus Simulation.pdsbak
 create mode 100644 ph3/Proteus Simulation.pdsprj
 create mode 100644 ph3/Proteus Simulation.pdsprj.DESKTOP-B7JNHU9.jhon9.workspace
 create mode 100644 ph3/sketch_phdos/sketch_phdos.ino
 create mode 100644 ph3/sketch_phdos/sketch_phdos.ino.standard.hex
 create mode 100644 ph3/sketch_phdos/sketch_phdos.ino.with_bootloader.standard.hex
 create mode 100644 registroparametros.aia
 create mode 100644 temperatura del agua/ARDUINO/sketch_temperaturados/sketch_temperaturados.ino
 create mode 100644 temperatura del agua/ARDUINO/sketch_temperaturados/sketch_temperaturados.ino.standard.hex
 create mode 100644 temperatura del agua/ARDUINO/sketch_temperaturados/sketch_temperaturados.ino.with_bootloader.standard.hex
 create mode 100644 temperatura del agua/PROTEUS/Backup Of temperatura.pdsbak
 create mode 100644 temperatura del agua/PROTEUS/Last Loaded temperatura.pdsbak
 create mode 100644 temperatura del agua/PROTEUS/temperatura.pdsprj
 create mode 100644 temperatura del agua/PROTEUS/temperatura.pdsprj.DESKTOP-2RQ58OU.Usuario.workspace
 create mode 100644 temperatura del agua/PROTEUS/temperatura.pdsprj.DESKTOP-B7JNHU9.jhon9.workspace
 create mode 100644 word/Proyecto Arduino/Proyecto Arduino .docx

Usuario@DESKTOP-2RQ58OU MINGW64 ~/OneDrive - unicauca.edu.co/Documentos/ProyectoArduino_BT (master)
$
```

Figura 23. Iniciar Commit.

Seguidamente se inicializa el repositorio remoto mediante las instrucciones que muestra en el GitHub y se hace su correspondiente actualización para poder visualizar los archivos del proyecto.

Figura 24. Inicialización del repositorio remoto

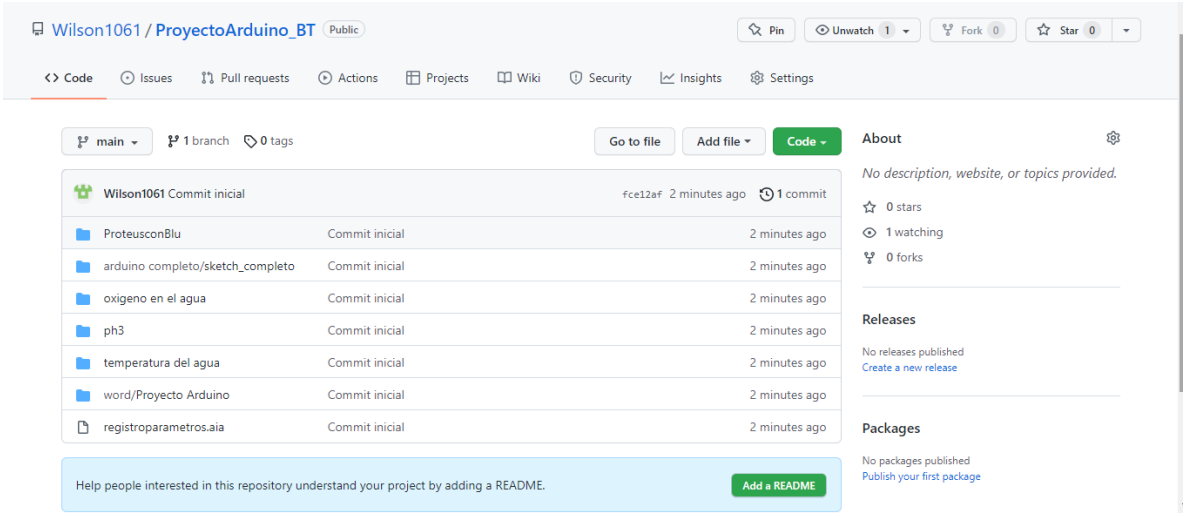


Figura 25. Archivos del proyecto por carpetas en GitHub.

Enlace del Proyecto: https://github.com/Wilson1061/ProyectoArduino_BT

CONCLUSIONES

A pesar de que partimos de la implementación de tres módulos diferentes de Arduino para cada uno de los sensores, estos se pueden llevar a la aplicación de un solo modulo Arduino, restructurando las líneas de código ya dispuestas anteriormente y haciendo el llamado mediante funciones, dando una optimización no solo del código, sino también de la parte física, ya que no necesitaremos los tres módulos Arduino, y esto nos podría ayudar en la disminución de costos si se desea implementar de manera práctica.

La práctica de simulación en PROTEUS en conjunto con Arduino ayuda a facilitar prácticas que se piensan que solo deben ser presenciales, dando esto así una gran facilidad de aplicación no solo para estudiantes universitarios, si no también distintos campos de aplicación ya que su implementación puede estar dada desde cualquier lugar remoto en el que se encuentren y como parámetro final puede ser llevado a una implementación real.

Como podemos observar mediante elementos de control, se puede llevar a cabo la la parametrización de indicadores que representen un monitoreo de la calidad del agua que este aplicada a una actividad económica como lo es la piscicultura presente en el departamento del Cauca, la cual es un pilar fundamental en esta zona agrícola, debido a su gran diversidad de climas, y apoyada por abundantes fuentes hídricas que posibilitan implementar este tipo de actividades. De acuerdo con esto, el presente proyecto tiene como finalidad el poder se implementado en este sector, que permita potenciar el nivel de producción de estas actividades económicas.

REFERENCIAS

[1] HETPRO, «Qué es Arduino?,» [En línea]. Available: https://hetpro-store.com/TUTORIALES/que-es-arduino/. [Último acceso: 24 enero 2022].

[2] F. U. D. POPAYÁN, «ESTUDIO DE FACTIBILIDAD PARA LA PRODUCCIÓN Y COMERCIALIZACIÓN DE TILAPIA ROJA,» 2020. [En línea]. Available: http://univida fup.edu.co/repositorio/files/original/c08d372dd115b833075d8e357ce36d46.pdf. [Último acceso: 26 enero 2022].