REPORTE FINAL DE PROYECTO

SISTEMAS EMBEBIDOS

Tema:

Piscina Camaronera

Profesor:

Ing. Christopher Vaccaro Cedillo

Integrante:

Isaías Ponce Alvarado

Paralelo práctico 104

Término Académico

I-2020

Objetivos

- Monitorear la piscina camaronera de forma remota hacia la plataforma de Ubidots, a través del protocolo mqtt.
- Simular la entrada de datos de los sensores de temperatura y pH recolectados por el Arduino hacia una raspberry aprovechando la conexión serial existente.
- Implementar el envío automático de alertas sobre límites superados de pH hacia el cliente por medio de correos eléctricos.

Descripción y explicación del proyecto

Mediante de la temperatura y sensado de pH de la piscina camaronera, se puede obtener un criterio del estado de dicha piscina para la óptima salud de los camarones, por ello se implementará un sistema monitoreo en el que el usuario pueda ejecutar un mecanismo para la respectiva corrección del pH, ya sea para incrementarlo o decrementarlo, puesto que el pH adecuado para los camarones está dentro del rango de 6.5 y 9. Para que los datos de Arduino puedan llegar a la nube, requerirá la comunicación con la Raspberry pi 3+, en este caso (Raspbian S.O.) para que envíe dichos datos en la nube (Ubidots).

Componentes que se usarían en una implementación real.

- Raspberry pi 3+ (script python utilizando el protocolo mqtt para el envío hacia la página de ubidots)
- Arduino (lenguaje C)
- Dos servomotores (Actuadores)
- Ubidots.
- 3 Sensores de ph SEN0161-V2
- 1 Sensor de temperatura DS18B20

Sensores de ph SEN0161-V2

Sensor que mide la concentración de ph, que será de gran relevancia para la piscina camaronera.

- Tipo de prueba: Grado de laboratorio
- Rango de detección de pH: 0~14
- Rango de temperatura: 5~60°C
- Response Time: 0.5 year (depending on frequency of use)
- Cable Length: 100cm
- Salida: 0 a 5v

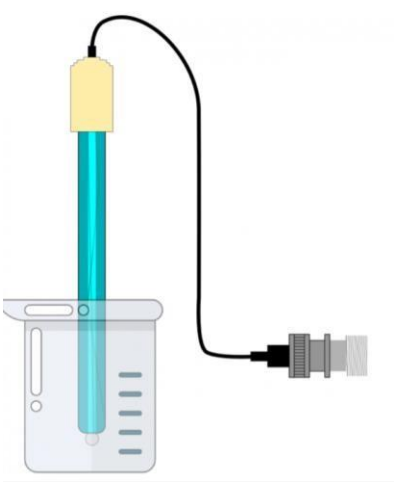


Figura #1 Sensor de pH

El Reemplazo para la simulación será la salida de dos potenciómetros de tal forma que formen un divisor de voltaje, el uso de los dos potenciómetros en vez de uno, es para aumentar la precisión de los valores de las resistencias, a fin de controlar mejor el voltaje, además para evitar que dicho voltaje se caiga, se aplica un servidor de voltaje, simulando la señal del sensor de pH pero a una escala más amplificada, de tal forma que quedaría algo similar a:

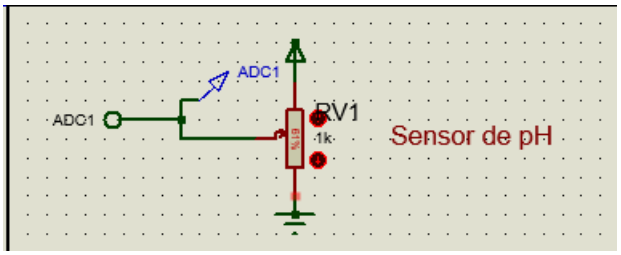


Figura #2 Sensor de pH en proteus

Para la conversión de voltaje a pH se recurrirá a la siguiente ecuación dentro de la lógica de Arduino.

$$pH = \frac{voltaje}{k}$$

Ecuación #1 Conversión V -> pH

$$k = 0.357$$

Ecuación #2 Representa los cinco voltios divido en catorce grados de pH.

Sensor de temperatura lm35

Este sensor permitirá medir la temperatura dentro del agua, sin riesgo de avería por humedad debido a cubierta protectora.

- Precisión: ±0.5°C (de -10°C a +85°C).
- Tiempo de captura inferior a 750ms.
- Alimentación: 3.0V a 5.5V.
- Identificador interno único de 64 bits.
- Múltiples sensores puede compartir el mismo pin.
- Rango de temperatura: -55 a 125°C.
- Resolución: de 9 a 12 bits (configurable).
- Interfaz 1-Wire (Puede funcionar con un solo pin).

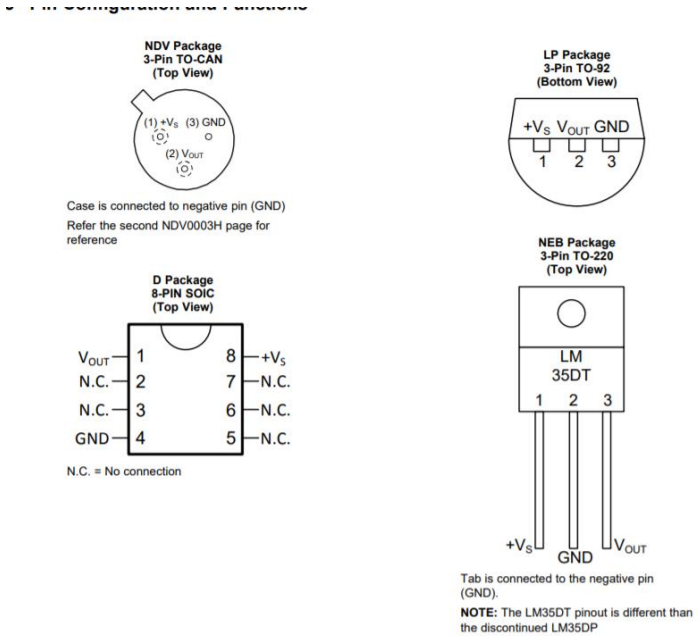


Figura #4 Sensor de temperatura

Arduino

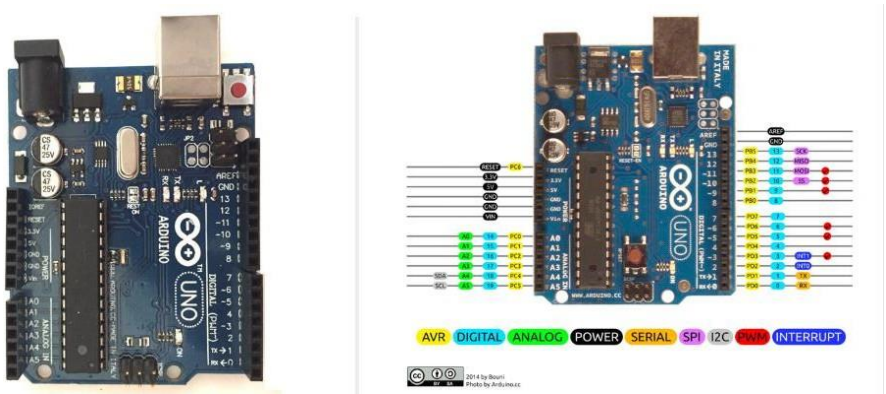


Figura #5 Arduino

- Microcontroller ATmega328P – 8 bit AVR family microcontroller
- Voltaje de operación: 5V
- Voltaje de entrada recomendado: 7-12V
- Límites de los voltajes de entrada: 6-20V
- Pines de entrada analógicos: 6 (A0 – A5)
- Corriente DC: on I/O Pins 40 mA
- Corriente DC: on 3.3V Pin 50 mA
- Flash Memory: 32 Kb
- SRAM: 2 KB
- EEPROM: 1 KB
- Frequency (Clock Speed) :16 MHz

Raspberry Pi 3+

- **PROCESADOR:** Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC
- **FRECUENCIA DE RELOJ:** 1,4 GHz
- **MEMORIA:** 1GB LPDDR2 SDRAM
- **CONECTIVIDAD INALÁMBRICA:**
  2.4GHz / 5GHz IEEE 802.11.b/g/n/ac
  Bluetooth 4.2, BLE
- **CONECTIVIDAD DE RED:** Gigabit Ethernet over
  USB 2.0 (300 Mbps de máximo teórico)
- **PUERTOS** GPIO 40
  pines HDMI
  4 x USB 2.0
  CSI (cámara Raspberry Pi)
  DSI (pantalla tácil)
  Toma auriculares / vídeo compuesto
  Micro SD
  Micro USB (alimentación)
  Power-over-Ethernet (PoE)

Figura #6 Raspberry pi 3+

El reemplazo para la simulación será el serial port, enviando datos hacia una máquina virtual con el S.O. Raspbian.

Motores.

Estos motores serán los encargados de modificar el pH, ya sea para incrementarlo como para decrementarlo, agregando distintas sustancias químicas dependiendo de la acción a realizar. Para este efecto se utilizaría un motor que trabaje máximo a 12v, el cual es un motor genérico llamado 130 motor rectangular 3-12v

- Rango de tensión: 3 V a 12 V
- Corriente: 800 mA
- Diámetro del eje: 2 mm
- Velocidad de recorrido libre @ 6 V: 8000 rpm

- Funcionamiento libre actual @ 6 V: 70 mA

- Funcionamiento con peso @ 6 V: 800 mA
- Dimensiones motor: 20 mm X 15 mm X 25 mm
- Modelo: 130

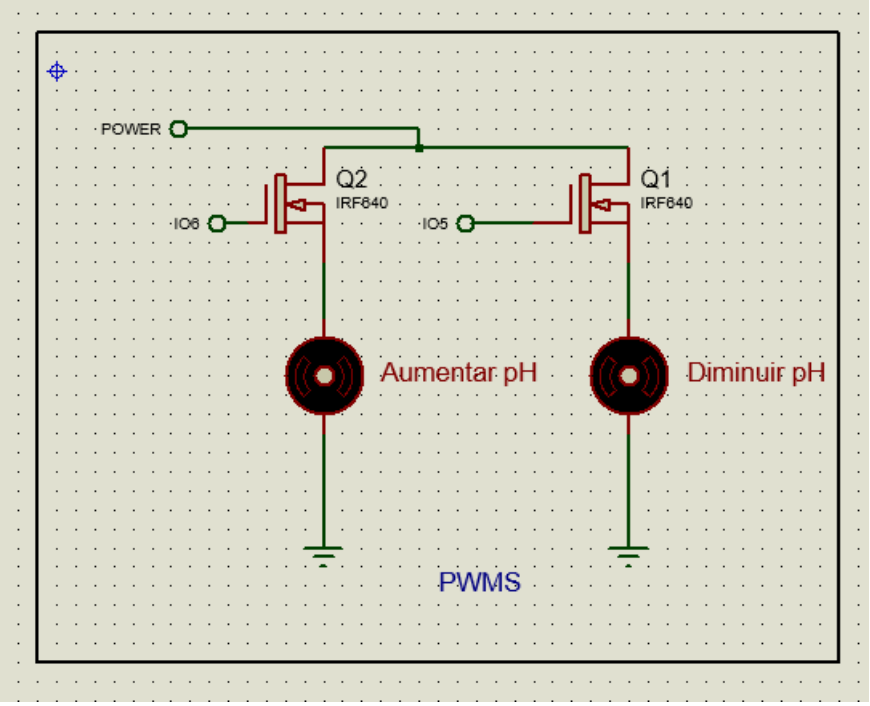


Figura #7 Actuadores para el pH

Ubidots

Plataforma especializada para conectar dispositivos IoT con el usuario, la misma en la que se monitoreará el nivel de pH, y temperatura, además de que el usuario tendría la opción de accionar un actuador para la correción de los niveles de pH en caso de necesitarlo.

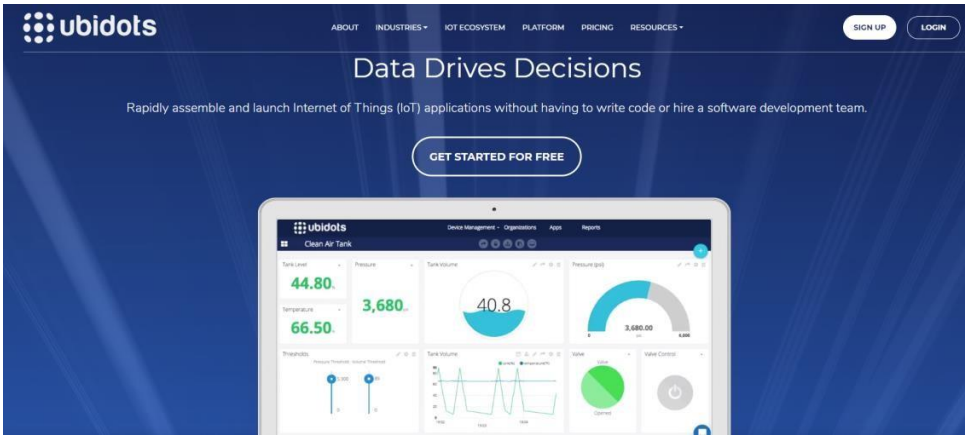


Figura #8 Ubidots para el monitoreo remoto

Protocolo ligero MQTT

MQTT (Message Queue Telemetry Transport) es un protocolo de transporte de mensajes Cliente/Servidor basado en publicaciones y subscripciones a los denominados “tópicos”. Cada vez que un mensaje es publicado será recibido por el resto de dispositivos adheridos a un tópico del protocolo.

Esquemático

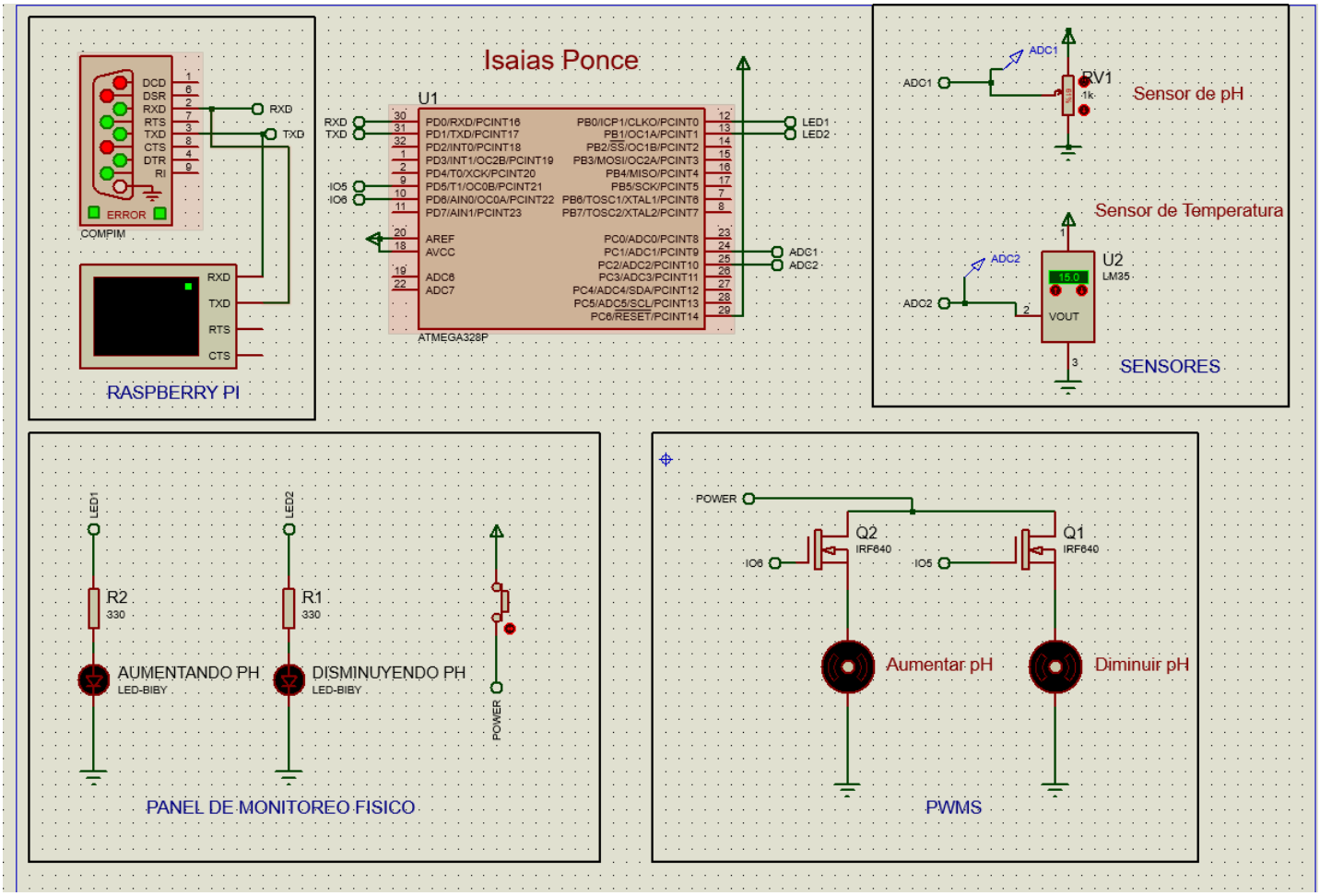


Figura #9 Esquemático en proteus

Esquemático en TinkerCad

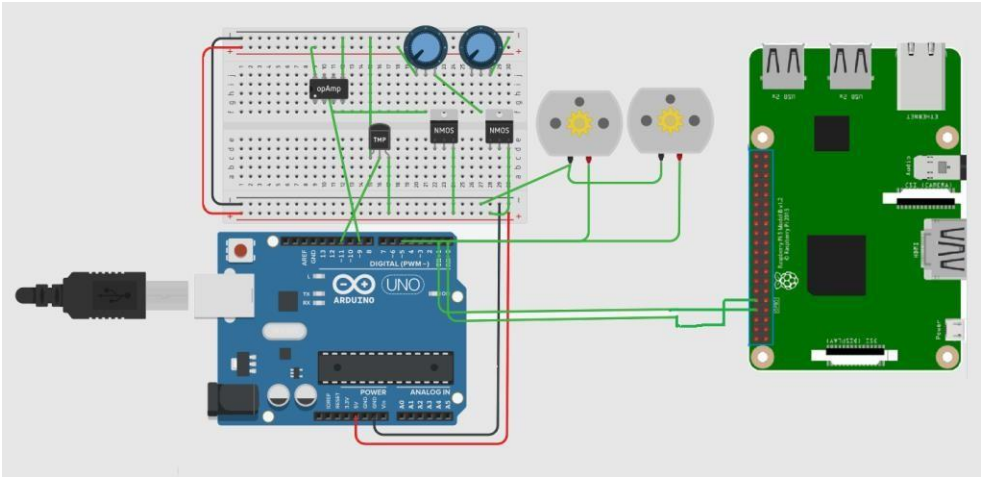


Figura #10 Esquemático en TinkerCad

Simulación

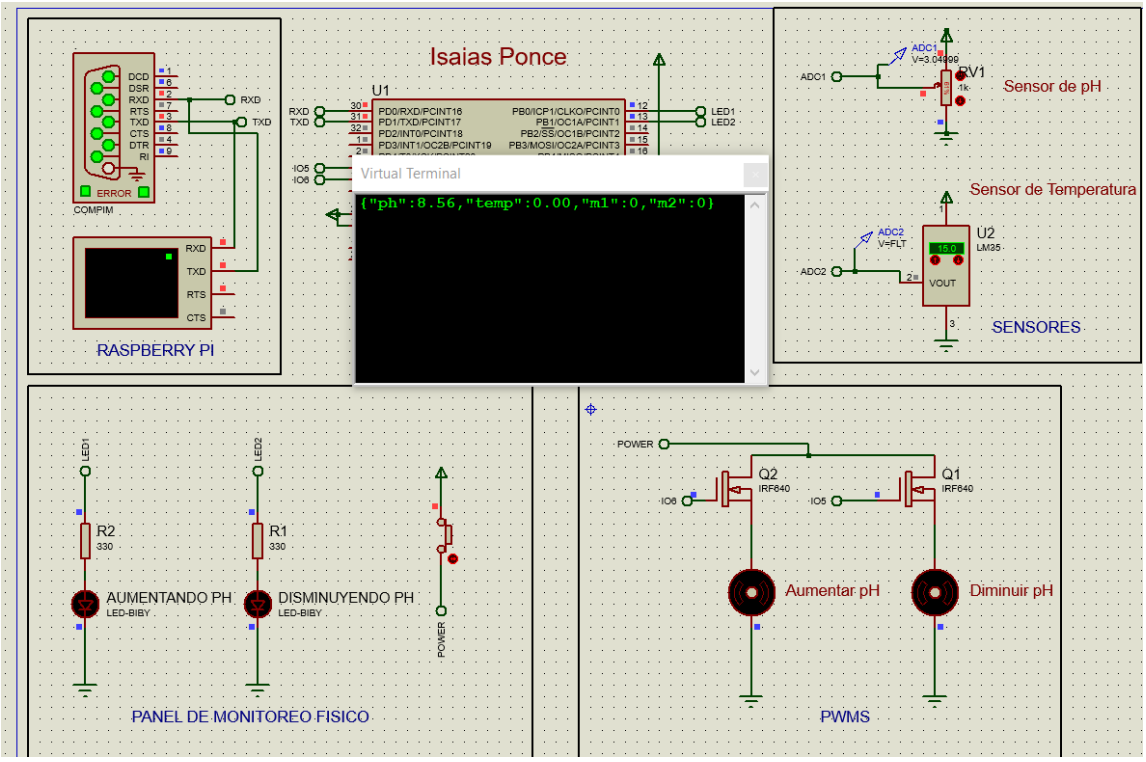


Figura #11 Nivel de 8 ph menor a 9 y mayor a 6.5, estado ok

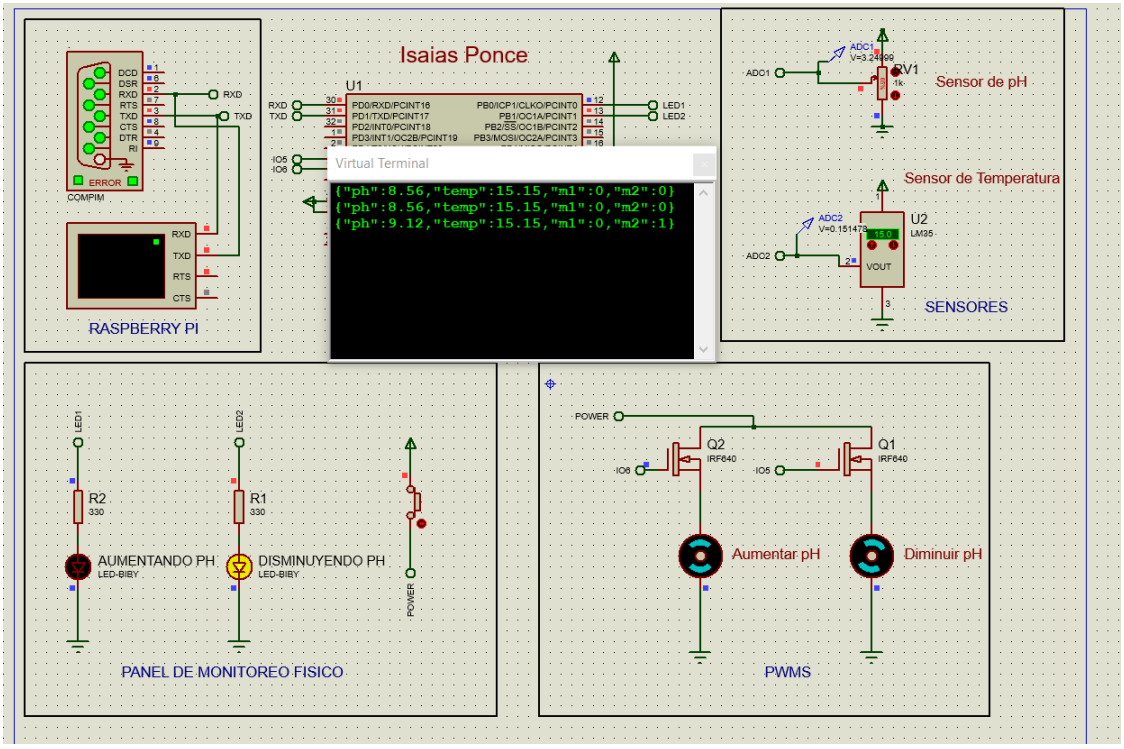


Figura #12 Activación de motor para disminuir pH, porque el pH superó la escala de 9

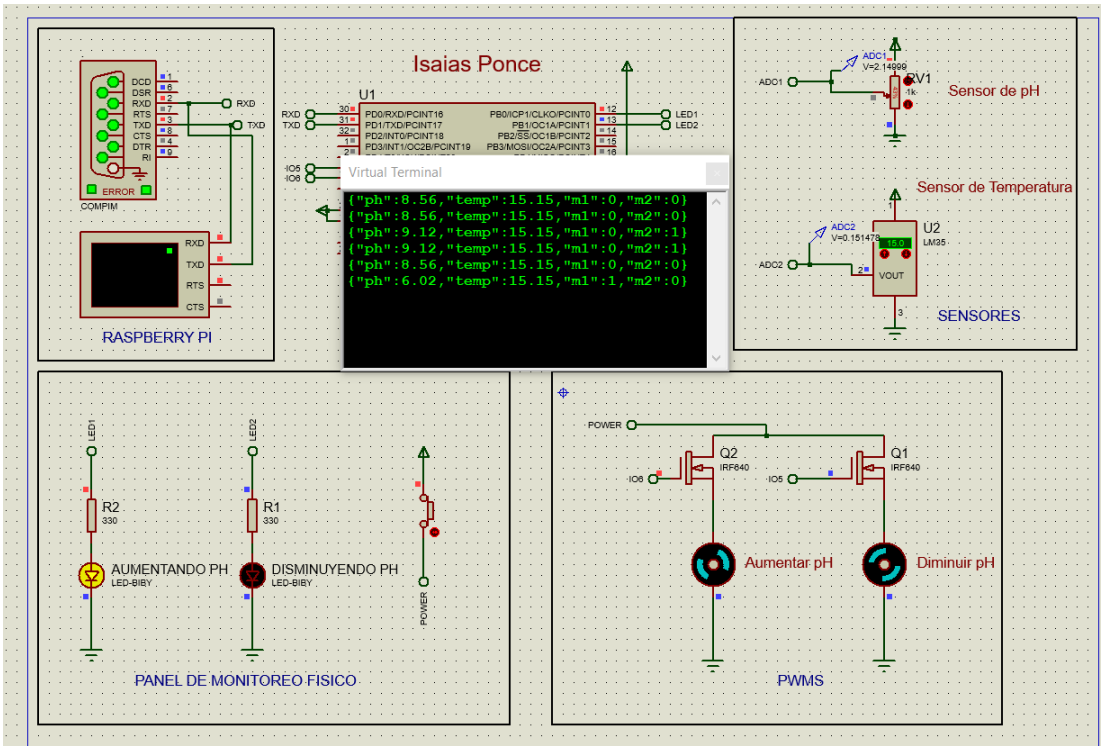


Figura #13 Activación de motor para Aumentar pH, porque el pH es menor escala de 6.5

```
while(1){
  int contador=0;
  promPh=0;
  promTemp=0;
  while(contador<10){
  leer();
    promPh=promPh+ph ;
  leer2();
     promTemp=promTemp+valor_real2;
     contador++;
  }
  promPh=promPh/contador;
  promTemp=promTemp/contador;
  if(promPh>=10){
       dtostrf(promPh, 5,2,buf);
      }else{
         dtostrf(promPh, 4,2,buf);
         }

      if(promTemp>=10){
       dtostrf(promTemp, 5,2,buf2);
      }else{
         dtostrf(promTemp, 4,2,buf2);
         }
        imprimir();
         _delay_ms(5200L);



}
 return 0;
```

Figura #14 Se realiza un promedio de datos obtenidos tanto para la temperatura como para el pH

```
*** valor aceptable del ph en una piscina camaronera es de 6.5 a 9 ***

Fuente: https://www.awamantenimiento.com/cual-es-nivel-ph-piscina/#:~:text=L

*/
  ph=volts_2_ph(valor_real);

if(ph<6.5 || ph>9){
  if(ph<6.5){//pH bajo para los camarones, entonces aumentar pH
    TCCR0A &=~(1<<5);
     TCCR0A &=~(1<<4);
      PORTD&= ~(1<<PD5);  //El de disminuir pH: OFF
      PORTB&= ~(1<<PB1);  //El de disminuir pH: OFF
   //PB1 LED DE DISMINUIR   PD5
    //PB0 LED DE AUMENTAR PD6
    motor2='0';
    TCCR0A |=(1<<6);
      TCCR0A |=(1<<7);
    PORTD|= (1<<PD6);//El de aumentar pH: ON
    motor1='1';
     _delay_ms(20);
      PORTB|= (1<<PB0);  //El de disminuir pH: OFF

    }else if(ph>9){//pH muy alto para los camarones, entonces disminuir pH
       TCCR0A |=(1<<5);
     TCCR0A |=(1<<4);
    PORTD|= (1<<PD5); //Diminuyendo pH: ON
         _delay_ms(20);
       PORTB|= (1<<PB1);  //El de disminuir pH: ON
     motor2='1';
```

Figura #15 Condiciones para el prendido y apagado de los motores en función del pH.

Ubidots

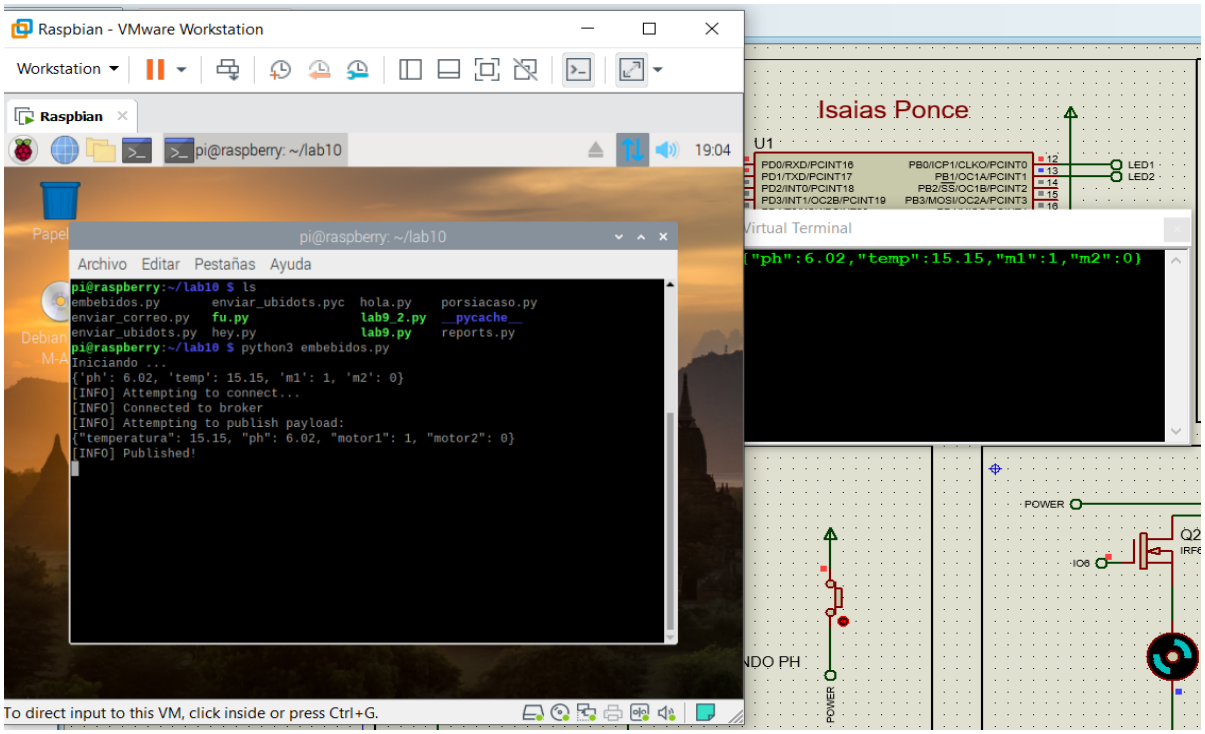


Figura #16 Envío de datos desde Arduino a la Raspberry a través del serial, y a su vez, el envío a la plataforma IoT Ubidots.

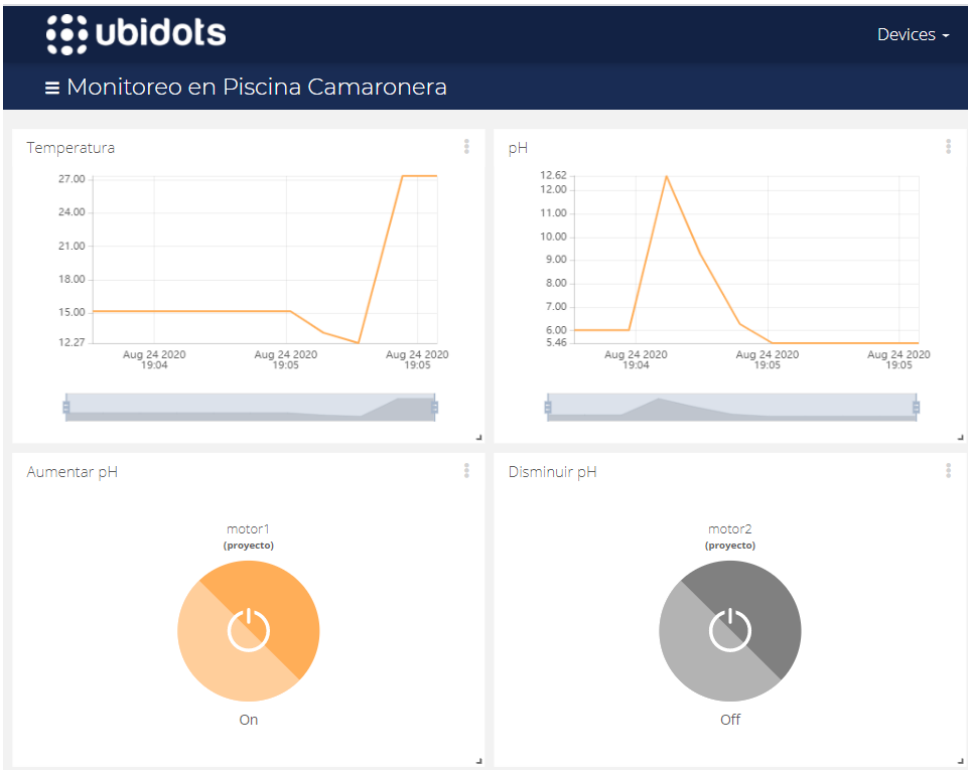
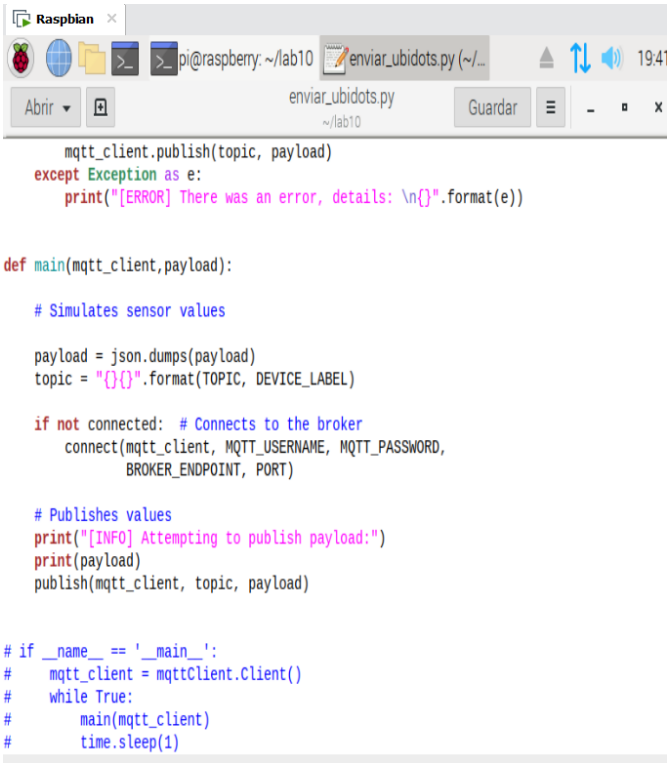


Figura #17 Envío de datos desde Arduino a la Raspberry a través del serial, y a su vez, el envío a la plataforma IoT Ubidots.




```
        mqtt_client.publish(topic, payload)
    except Exception as e:
        print("[ERROR] There was an error, details: \n{}".format(e))


def main(mqtt_client,payload):

    # Simulates sensor values

    payload = json.dumps(payload)
    topic = "{}{}".format(TOPIC, DEVICE_LABEL)

    if not connected:  # Connects to the broker
        connect(mqtt_client, MQTT_USERNAME, MQTT_PASSWORD,
                BROKER_ENDPOINT, PORT)

    # Publishes values
    print("[INFO] Attempting to publish payload:")
    print(payload)
    publish(mqtt_client, topic, payload)


# if __name__ == '__main__':
#     mqtt_client = mqttClient.Client()
#     while True:
#         main(mqtt_client)
#         time.sleep(1)
```

Figura #18 Código para el envío a la página de Ubidots

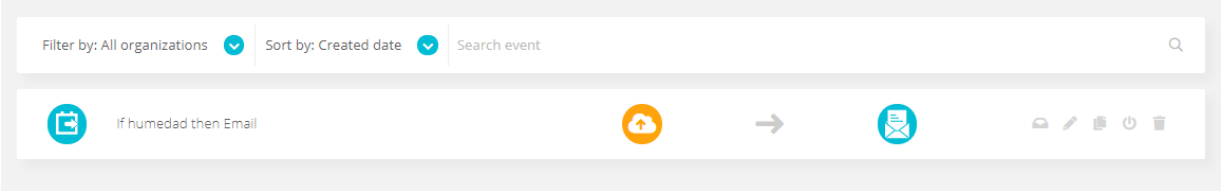


Figura #19 Automatización de envio de correos cuando la duración con ph fuera de rangos sea mayor a 3 minutos.

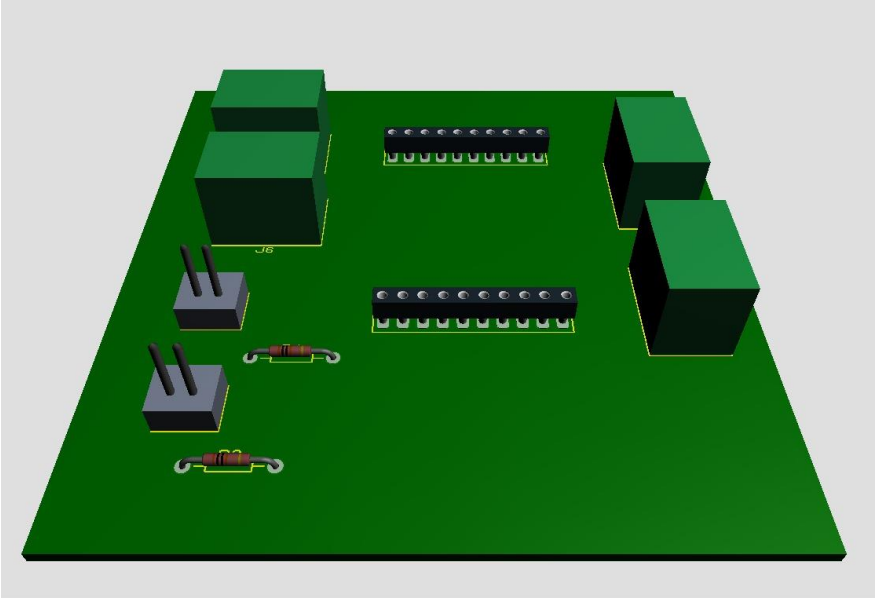


Figura #20 Diseño de la PCB, con el socket para el microcontrolador

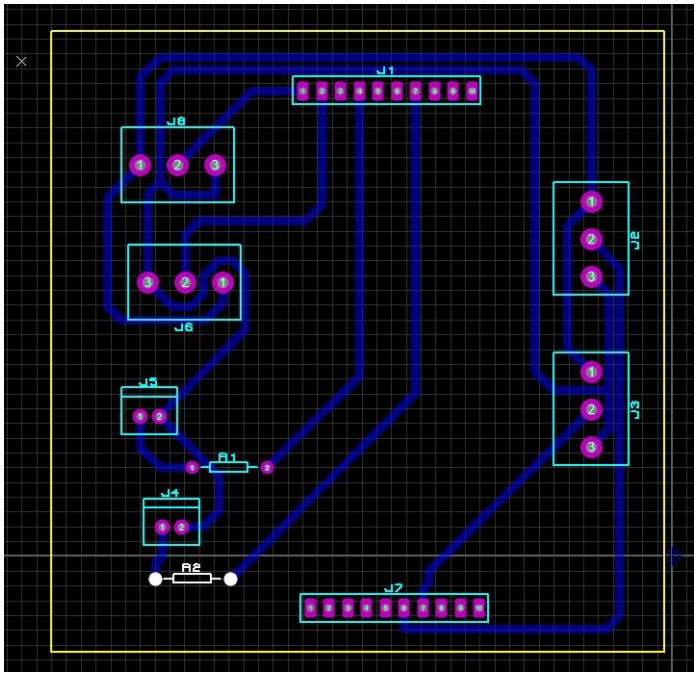


Figura #21 Pistas para la PCB.

Conclusiones y Recomendaciones:

- Se logró monitorear la piscina camaronera de forma remota a través de la plataforma Ubidots en conjunto con los datos obtenidos de los sensores con Arduino y raspberry pi, además de recibir alertas por correo sobre el tiempo.
- Implementación exitosa del sistema de autocorrección de los niveles de pH, en los que de acuerdo a la necesidad, el sistema sabrá aumentar o disminuir el pH para mantener la salud de los camarones.
- Utilizar el protocolo mqtt para el envío de datos, puesto que es un protocolo ligero y es altamente recomendado para sistemas embebidos de bajo consumo.
- Como buena práctica es importante realizar un promedio periódico de los datos recolectados por los sensores, puesto que puede tengan una gran variabilidad, sin embargo esto se elimina al realizar el promedio de n cantidad de números dependiendo de la variabilidad.

Anexos

main.c uart.c

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/eeprom.h>
#include "uart.h"
#include <stdio.h>
#include <stdlib.h>
#define CONSTANTE 0.357

void imprimir();
float volts_2_ph(float voltaje);
volatile float ph=0.0;
unsigned int voltaje1=0;
volatile float valor_real=0.0;
char buf[6]={0};
void leer();
char intToChar(int numero);
void avanzar2() ;
void avanzar1() ;

char buf2[6]={0};
char motor1='0';
char motor2='0';

float promPh=0;
float promTemp=0;

unsigned int voltaje2=0;
volatile float valor_real2=0.0;
void leer2();


int main()

{
   serial_begin();


 DDRD |= 0b01100000; //Salidas para los pwms
 DDRB |=0b00000011;  //Salidas para los leds
TCCR0A = 0b11110011 ;
TCCR0B = 0b00000011;
OCR0A = 0; //valor inicial de pwm para el pin OC0A
OCR0B = 0; //valor inicial de pwm para el pin OC0B
```

Figura #21 Declaración e inicialización de variables

main.c uart.c

```
37
38    DDRD |= 0b01100000; //Salidas para los pwms
39    DDRB |=0b00000011;  //Salidas para los leds
40   TCCR0A = 0b11110011 ;
41   TCCR0B = 0b00000011;
42   OCR0A = 0; //valor inicial de pwm para el pin OC0A
43   OCR0B = 0; //valor inicial de pwm para el pin OC0B
44
45
46    DDRB |= ((1<<0)|(1<<1));
47   ADC_init();
48   TCCR0A  &=~(1<<6);
49   TCCR0A  &=~(1<<7);
50   TCCR0A  &=~(1<<5);
51   TCCR0A  &=~(1<<4);
52
53   //serial_print_str("Iniciando, espere ...");
54   //serial_print_str("\r\n");
55   _delay_ms(2000L);
56
57 while(1){
58      int contador=0;
59      promPh=0;
60      promTemp=0;
61    while(contador<10){
62      leer();
63        promPh=promPh+ph ;
64      leer2();
65         promTemp=promTemp+valor_real2;
66         contador++;
67      }
68      promPh=promPh/contador;
69      promTemp=promTemp/contador;
70    if(promPh>=10){
71           dtostrf(promPh, 5,2,buf);
72          }else{
73             dtostrf(promPh, 4,2,buf);
74             }
75
76         if(promTemp>=10){
77          dtostrf(promTemp, 5,2,buf2);
78         }else{
79            dtostrf(promTemp, 4,2,buf2);
```

Figura #22 Inicialización de los timers para los pwms y promedio de los valores de los sensores

main.c uart.c

```
84
85
86  }
87    return 0;
88   }
89 void leer(){
90      voltaje1=ADCGet(1);//ph
91
92      valor_real=voltaje1*5.0/1023;
93
94    /*
95
96    *** valor aceptable del ph en una piscina camaronera es de 6.5 a 9 ***
97
98    Fuente: https://www.awamantenimiento.com/cual-es-nivel-ph-piscina/#:~:text=El%20pH%2
99
100   */
101     ph=volts_2_ph(valor_real);
102
103 if(ph<6.5 || ph>9){
104   if(ph<6.5){//pH bajo para los camarones, entonces aumentar pH
105      TCCR0A  &=~(1<<5);
106       TCCR0A  &=~(1<<4);
107        PORTD&= ~(1<<PD5);  //El de disminuir pH: OFF
108        PORTB&= ~(1<<PB1);  //El de disminuir pH: OFF
109     //PB1 LED DE DISMINUIR   PD5
110      //PB0 LED DE AUMENTAR PD6
111     motor2='0';
112     TCCR0A  |=(1<<6);
113        TCCR0A  |=(1<<7);
114     PORTD |= (1<<PD6);//El de aumentar pH: ON
115     motor1='1';
116      _delay_ms(20);
117        PORTB|= (1<<PB0);  //El de disminuir pH: OFF
118
119     }else if(ph>9){//pH muy alto para los camarones, entonces disminuir pH
120          TCCR0A  |=(1<<5);
121        TCCR0A  |=(1<<4);
122      PORTD |= (1<<PD5); //Diminuyendo pH: ON
123           _delay_ms(20);
124         PORTB|= (1<<PB1);  //El de disminuir pH: ON
125       motor2='1';
126
```

Figura #23 Aplicando las condiciones

```
main.c   uart.c
126
127          TCCR0A  &=~(1<<6);
128         TCCR0A  &=~(1<<7);
129          PORTD&= ~(1<<PD6);  //El de aumentar pH: OFF
130
131          motor1='0';
132         //    PORTB&= ~(1<<PB1);  //El de disminuir pH: OFF
133            PORTB&= ~(1<<PB0);
134          }
135
136
137       }else{
138            TCCR0A  &=~(1<<6);
139          TCCR0A  &=~(1<<7);
140           PORTD&= ~(1<<PD6);  //El de aumentar pH: OFF
141               TCCR0A  &=~(1<<5);
142          TCCR0A  &=~(1<<4);
143           PORTD&= ~(1<<PD5);  //El de disminuir pH: OFF
144
145          motor1='0';
146          motor2='0';
147             PORTB&= ~(1<<PB1);  //El de disminuir pH: OFF
148             PORTB&= ~(1<<PB0);  //El de disminuir pH: OFF
149        }
150
151     }
152 void leer2(){
153
154
155     voltaje2=ADCGet(2);
156     valor_real2=voltaje2*5.0*100/1023;
157
158     }
159
160 float volts_2_ph(float voltaje){
161
162     float ph=voltaje/CONSTANTE;
163     return ph;
164     }
```

Figura #24 Lectura de la temperatura.

```
165 void imprimir(){
166
167       serial_print_char('{');
168         serial_print_char(34);
169         serial_print_str("ph");
170         serial_print_char(34);
171          serial_print_char(':');
172         serial_print_str(buf);
173
174         serial_print_char(',');
175         serial_print_char(34);
176          serial_print_str("temp");
177         serial_print_char(34);
178         serial_print_char(':');
179         serial_print_str(buf2);
180
181         serial_print_char(',');
182         serial_print_char(34);
183         serial_print_str("m1");
184         serial_print_char(34);
185         serial_print_char(':');
186         serial_print_char(motor1);
187
188         serial_print_char(',');
189         serial_print_char(34);
190         serial_print_str("m2");
191         serial_print_char(34);
192         serial_print_char(':');
193         serial_print_char(motor2);
194         serial_print_char('}');
195         serial_print_str("\r\n");
196
197       }
```

Figura #25 Impresión en formato json

Bibliografias

(DF ROBOT) ***Gravity: Analog pH Sensor/Meter Kit V2*** *(2017) Obtenido en:* https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0161-V2_Web.pdf

(Xataca, Pastor J.) ***Raspberry pi 3 model b+*** *(2018) Obtenido en:* https://www.xataka.com/ordenadores/raspberry-pi-3-model-b-analisis-mas-potencia- y-mejor-wifi-para-un-minipc-que-sigue-asombrando

(Maxim-Integrated) ***Programmable Resolution 1-Wire Digital Thermometer*** *(2019) Obtenido en:* https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

(Components 101) ***Arduino Uno*** *(2018) Obtenido en:*
https://components101.com/microcontrollers/arduino-uno

(Texas Instruments) ***LM35 Precision Centigrade Temperature Sensors*** (1997)
https://www.ti.com/lit/ds/symlink/lm35.pdf

(Macías F.***)Consultoría en Camarones*** (2017)
http://www.fao.org/3/ac397s/AC397S05.htm#:~:text=Agua%20con%20PH%20de%206,oxidaci%C3%B3n%20del%20sedimento%20con%20sulfides.

(UBIDOTS) ***Ubidots api docs*** (2019) https://ubidots.com/docs/hw/?language=Python#send-data

(Empresa TST) ***Protocolo MQTT*** (2018) http://www.tst-sistemas.es/mqtt/#:~:text=MQTT%20(Message%20Queue%20Telemetry%20Transport,a%20un%20t%C3%B3pico%20del%20protocolo.