REPÚBLICA DEL ECUADOR
Escuela Politécnica Nacional
"E SCIENTIA HOMINIS SALUS"

La versión digital de esta tesis está protegida por la Ley de Derechos de Autor del Ecuador.

Los derechos de autor han sido entregados a la "ESCUELA POLITÉCNICA NACIONAL" bajo el libre consentimiento del (los) autor(es).

Al consultar esta tesis deberá acatar con las disposiciones de la Ley y las siguientes condiciones de uso:

- Cualquier uso que haga de estos documentos o imágenes deben ser sólo para efectos de investigación o estudio académico, y usted no puede ponerlos a disposición de otra persona.

- Usted deberá reconocer el derecho del autor a ser identificado y citado como el autor de esta tesis.

- No se podrá obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.

El Libre Acceso a la información, promueve el reconocimiento de la originalidad de las ideas de los demás, respetando las normas de presentación y de citación de autores con el fin de no incurrir en actos ilegítimos de copiar y hacer pasar como propias las creaciones de terceras personas.

Respeto hacia sí mismo y hacia los demás.

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO Y DISEÑO DE UN PROTOTIPO PARA EL MONITOREO DE ACUARIOS UTILIZANDO TECNOLOGÍA WIFI (IEEE 802.11b/g/n) ENFOCADA AL IoT (INTERNET OF THINGS) MEDIANTE UNA PLATAFORMA RASPBERRY PI Y EL SISTEMA OPERATIVO ANDROID

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

CÉSAR RAMIRO CUSI VACA
cc28cesar@hotmail.com

DARÍO FERNANDO SÁNCHEZ MOROCHO
patos17@live.com

DIRECTOR: ING. CARLOS ALFONSO HERRERA MUÑOZ, MSc.
carlos.herrera@epn.edu.ec

CODIRECTOR: ING. ROBERTO OMAR ANDRADE PAREDES, MSc.
robeto.andrade@epn.edu.ec

Quito, noviembre 2016

DECLARACIÓN

Nosotros, César Ramiro Cusi Vaca y Darío Fernando Sánchez Morocho, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____ _____

César Ramiro Cusi Vaca Darío Fernando Sánchez Morocho

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por César Ramiro Cusi Vaca y Darío Fernando Sánchez Morocho, bajo nuestra supervisión.

Ing. Carlos Herrera, MSc.	Ing. Roberto Andrade, MSc.
DIRECTOR DEL PROYECTO	**CODIRECTOR DEL PROYECTO**

AGRADECIMIENTO

Mi eterno agradecimiento a Dios por cuidarme en todo momento y lugar, por mostrarme la luz en el momento oportuno, por poner en mi camino a personas muy buenas y bondadosas, la oportunidad de vivir y regalarme una maravillosa familia.

De la misma manera agradezco a mi madre quien es una mujer muy luchadora y quien me enseñó a ganarme cada centavo, a mi padre quien con su sabiduría supo guiarme en el andar de la vida.

También mi sincero agradecimiento para cada uno de mis hermanos y hermanas quienes siempre quieren lo mejor para mí, todos tienen un lugar muy especial en mi corazón, son a quienes les debo mucho ya que cada uno contribuyeron en formarme como persona y en lo profesional. De igual manera a mis queridos sobrinos y sobrinas por el respeto y cariño que me tienen.

De manera especial agradecer al MSc. Carlos Herrera por su tiempo y colaboración para la culminación del proyecto y de la misma manera al MSc. Roberto Andrade quien fue un pilar en el proyecto de titulación.

Finalmente quiero agradecer a la sociedad ya que gracias a sus aportes con el estado ecuatoriano dan vida a las universidades públicas del país.

César Ramiro Cusi Vaca

AGRADECIMIENTO

Mi agradecimiento eterno a Dios por permitirme culminar mi carrera universitaria.

A mis padres, Enrique y Luzmila, que han sido el pilar fundamental en mi vida, porque gracias a sus consejos, enseñanzas y su apoyo incondicional, me han permitido lograr todas mis metas.

A mi hermano Leonardo, que ha sido mi ejemplo a seguir, ya que me ha enseñado a nunca rendirme y a luchar siempre para concretar cualquier objetivo en la vida.

A mis familiares, por el aliento y apoyo que de una u otra manera me han brindado a lo largo de la vida.

A la Escuela Politécnica Nacional, que por medio de sus excelentes maestros, han permitido que pueda adquirir los conocimientos necesarios para culminar mi carrera universitaria.

Finalmente agradezco de manera especial al MSc. Carlos Herrera y al MSc. Roberto Andrade por el apoyo y la gestión realizada para lograr la culminación del presente trabajo de titulación.

Darío Fernando Sánchez Morocho

DEDICATORIA

Con mucho amor y gratitud dedico el presente trabajo a mis padres, hermanos y hermanas quienes siempre creyeron en mí, quienes en cada etapa de mi vida supieron guiarme por un buen camino, por darme el coraje de seguir adelante y no desmayar en los problemas que se presentaban.

También dedico a mi querida enamorada María Elizabeth Quishpe Sacancela quien siempre estuvo pendiente de mí y cuyo amor se convirtió en el motor de mi vida, dándome las fuerzas necesarias que me permitieron llegar a este momento especial en mi vida.

César Ramiro Cusi Vaca

DEDICATORIA

El presente Proyecto de titulación está dedicado a:

Dios, por brindarme la oportunidad de poder estudiar una carrera universitaria.

Mis padres y a mi hermano, ya que siempre me apoyaron incondicionalmente para poder culminar esta etapa de mi vida.

Mis familiares, por todo el apoyo y cariño brindado a lo largo de la vida.

Mi amor Cristina, por brindarme su apoyo y cariño sincero a lo largo de estos años.

Darío Fernando Sánchez Morocho

CONTENIDO

DECLARACIÓN I
CERTIFICACIÓN II
AGRADECIMIENTO III
DEDICATORIA V
CONTENIDO VII
ÍNDICE DE FIGURAS XIII
ÍNDICE DE TABLAS XVI
ÍNDICE DE CÓDIGOS XVI
RESUMEN XIX
PRESENTACIÓN.....XX
CAPÍTULO I 1
1. INTRODUCCIÓN AL IoT……...1
1.1 GENERALIDADES DEL HÁBITAT EN UN ACUARIO 1
1.1.1 TIPOS DE ACUARIOS 2
1.1.1.1 Acuario de agua salada……2
1.1.1.2 Acuario de agua dulce……..2
1.1.1.3 Acuario de agua salobre……..2
1.1.2 PARÁMETROS PRINCIPALES PARA PRESERVAR UN ACUARIO…2
1.1.2.1 Temperatura……...3
1.1.2.2 pH……3
1.1.2.3 Conductividad eléctrica……3
1.1.2.4 Iluminación…..……4
1.1.2.5 Humedad ambiental……..4
1.2 TECNOLOGÍA WIFI (802.11 b/g/n)……..4
1.2.1 IEEE 802.11b 5

1.2.2 IEEE 802.11g……...5
1.2.3 IEEE 802.11n……...6
1.3 INTERNET DE LAS COSAS (IoT) 6
1.3.1 COXIONES EN EL IoT 8
1.3.1.1 Conexión P2P……8
1.3.1.2 Conexión M2P……8
1.3.1.3 Conexión M2M……...8
1.3.2 INCONVENIENTES PARA EL DESARROLLO DEL IoT 9
1.3.3 APLICACIONES DEL IoT 9
1.4 PLATAFORMA RASPBERRY PI 10
1.4.1 RASPBERRY PI 2 MODELO B 10
1.4.1.1 Puertos GPIO y Python……...12
1.5 PLATAFORMA ARDUINO 13
1.5.1 ARDUINO UNO 13
1.6 ARDUINO ETHERNET SHIELD 14
1.7 SENSORES 15
1.7.1 CLASIFICACIÓN DE LOS SENSORES 15
1.7.1.1 Sensor de luz BH1750……16
1.7.1.2 Sensor de humedad y temperatura DHT22……16
1.7.1.3 Sensor sumergible de temperatura DS18B20……17
1.7.1.4 Sensor de pH SKU: SEN0161……..18
1.7.1.5 Sensor de conductividad eléctrica SKU: DFR0300……18
1.8 REDIRECCIONAMIENTO DE PUERTOS 19
1.9 BASE DE DATOS....19
1.9.1 COMPONENTES DE UNA BASE DE DATOS 20
1.9.2 SISTEMA DE GESTIÓN DE BASE DE DATOS 20
1.9.2.1 MySQL……....21

1.9.2.1.1 phpMyAdmin……..21
1.10 SERVIDORES 21
1.10.1 TIPOS DE SERVIDORES.....22
1.10.1.1 Servidor Apache……..22
1.11 WEB HOSTING.....23
1.12 SISTEMAS OPERATIVOS 23
1.12.1 SISTEMA OPERATIVO ANDROID.....23
1.12.2 SISTEMA OPERTIVO LINUX 24
1.12.3 SISTEMA OPERATIVO MAC 24
1.12.4 SISTEMA OPERATIVO WINDOWS 24
1.12.5 COMPARACIÓN ENTRE SISTEMAS OPERATIVOS.....25
1.13 APP INVENTOR.....26
CAPÍTULO II 28
2. DISEÑO DEL PROTOTIPO PARA EL MONITOREO DE ACUARIOS 28
2.1 INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA OPERATIVO EN LA RASPBERRY PI 2 30
2.1.1 INSTALACIÓN DEL OS RASPBIAN JESSIE 30
2.1.1.1 Configuración de la resolución de pantalla en la Raspberry Pi 2...31
2.1.2 CONFIGURACIÓN DE INICIO EN LA RASPBERRY PI 31
2.2 INSTALACIÓN DEL SOFTWARE ARDUINO 33
2.3 DIAGRAMA DE FLUJO PARA OBTENER Y ENVIAR DATOS EN ARDUINO 34
2.4 PROGRAMACIÓN DE LOS SENSORES EN ARDUINO 35
2.4.1 PROGRAMACIÓN PARA EL SENSOR SUMERGIBLE DE TEMPERATURA DS18B20 35
2.4.2 PROGRAMACIÓN PARA EL SENSOR DE CONDUCTIVIDAD ELÉCTRICA SKU: DFR0300 36

2.4.3 PROGRAMACIÓN PARA EL SENSOR DE TEMPERATURA Y HUMEDAD AMBIENTAL DHT22 37
2.4.4 PROGRAMACIÓN PARA EL SENSOR DE LUZ BH1750 38
2.4.5 PROGRAMACIÓN PARA EL SENSOR DE PH SKU: SEN0161 39
2.5 DISEÑO DE LA BASE DE DATOS 40
2.5.1 INSTALACIÓN DEL SERVIDOR APACHE 2 40
2.5.2 INSTALACIÓN DEL PROGRAMA PHP 40
2.5.3 INSTALACIÓN DE BASE DE DATOS MySQL 41
2.5.4 INSTALACIÓN DE PHPMYADMIN 42
2.5.5 PUERTO LOGICO DE ACCESO AL SERVIDOR 44
2.5.6 CREACIÓN DE LA BASE DE DATOS EN PHPMYADMIN 44
2.6 CONEXIÓN ENTRE ARDUINO Y LA BASE DE DATOS 47
2.6.1 DIAGRAMA DE FLUJO PARA EL ENVIO DE INFORMACIÓN A LA BASE DE DATOS 47
2.6.2 PROGRAMACIÓN DE SCRIPTS EN EL SERVIDOR 48
2.6.2.1 Script de conexión con la base de datos 48
2.6.2.2 Script de envío de información a la base de datos 49
2.6.2.3 Configuración de la dirección IP estática en el servidor 50
2.6.3 PROGRAMACIÓN DEL CLIENTE PARA EL ENVÍO DE DATOS 50
2.6.3.1 Configuración de la Ethernet Shield 50
2.6.3.2 Envío de información desde Arduino a la base de datos 51
2.7 GENERACIÓN DE HISTORIAL Y REPORTES 52
2.7.1 HISTORIAL 52
2.7.2 REPORTES 53
2.8 DISEÑO DE LA APLICACIÓN PARA ANDROID 54
2.8.1 ETAPA DE AUTENTICACIÓN 56
2.8.2 ETAPA ZONAS DE SENSADO 58

2.8.3 ETAPA DE VISUALIZACIÓN DE DATOS.....59
2.8.3.1 Scripts para obtener el dato actual……...59
2.8.3.2 Pantalla de datos obtenidos……..60
2.8.3.3 Pantalla de Historial y Reportes……62
2.9 PROGRAMACIÓN DE LA RAPBERRY PI PARA CONTROLAR ALARMAS……..64
2.9.1 INSTALACIÓN DE LA LIBRERÍA RPi.GPIO.....64
2.9.2 DIAGRAMA DE FLUJO PARA EL CONTROL DE LOS PUERTOS GPIO EN PYTHON.....65
2.9.3 CONTROL DE PUERTOS GPIO CON PYTHON 65
2.10 DISEÑO DEL CIRCUITO ELÉCTRÓNICO PARA EL CONTROL DE ALARMAS 68
2.10.1 DISEÑO DE LA ETAPA DE POTENCIA DEL CIRCUITO 68
2.10.2 DIAGRAMA DE BLOQUES TOTAL DEL CIRCUITO (PLACA) 70
2.10.3 DIAGRAMA ESQUEMÁTICO TOTAL DEL CIRCUITO (PLACA) 71
2.10.4 DIAGRAMA TOTAL DEL CIRCUITO IMPRESO PARA LA PLACA ..72
2.11 CONFIGURACIÓN DEL EQUIPO WIFI 73
CAPÍTULO III 75
3.IMPLEMENTACIÓN DEL PROTOTIPO Y PRUEBAS 75
3.1 IMPLEMENTACIÓN DE LA PLACA EN BAQUELITA.....75
3.2 IMPLEMENTACIÓN DEL CABLE DE CONEXIÓN DE LOS SENSORES..77
3.3 IMPLEMENTACIÓN DE CABLES DE ALIMENTACIÓN 78
3.4 IMPLEMENTACIÓN DEL PROTOTIPO.....78
3.5 PRUEBA DE ENVÍO DE INFORMACIÓN ENTRE LOS SENSORES Y ARDUINO 80
3.6 PRUEBA DE ENVÍO DE INFORMACIÓN DESDE ARDUINO A LA BASE DE DATOS 81

3.7 PRUEBA DE FUNCIONAMIENTO DE LA APLICACIÓN PARA ANDROID….....81

3.8 PRUEBA DE FUNCIONAMIENTO DE LOS EQUIPOS ELÉCTRICOS.....83

3.8.1 PRUEBA DE FUNCIONAMIENTO DEL VENTILADOR.....83

3.8.2 PRUEBA DE FUNCIONAMIENTO DE LA LÁMPARA FLUORESCENTE 84

3.8.3 PRUEBA DE FUNCIONAMIENTO DEL CALENTADOR DE AGUA....85

3.8.4 PRUEBA DE FUNCIONAMIENTO DE LA LÁMPARA LED 85

3.8.5 PRUEBA DE FUNCIONAMIENTO DE LA ALARMA AUDITIVA 86

3.9 PRUEBA DE FUNCIONAMIENTO DEL DOMINIO Y HOSTING.....87

CAPÍTULO IV.....89

4. PRESUPUESTO REFERENCIA……89

CAPÍTULO V.....91

5. CONCLUSIONES Y RECOMENDACIONES……..91

5.1 CONCLUSIONES.....91

5.2 RECOMENDACIONES 92

REFERENCIAS BIBLIOGRÁFICAS 94

ANEXOS……ANEXOS

ANEXO A..... A1

MANUAL DE USUARIO DE LA APLICACIÓN ANDROID

ANEXO B..... B1

CÓDIGOS CREADOS EN ARDUINO Y LA RASPBERRY PI

ANEXO C……..C1

INFORMACIÓN TÉCNICA DE SENSORES Y ELEMENTOS UTILIZADOS

ÍNDICE DE FIGURAS

CAPÍTULO I

Figura 1.1 Representación de un acuario tradicional……...1
Figura 1.2 Escala de pH……..……3
Figura 1.3 Logotipo para dispositivos certificados por Wi-Fi Alliance……....5
Figura 1.4 El planeta y el IoT……7
Figura 1.5 Relación de dispositivos conectados por personas……...7
Figura 1.6 Conexiones del IoT……8
Figura 1.7 Logo Raspberry Pi……10
Figura 1.8 Raspberry Pi 2 modelo B……..11
Figura 1.9 Distribución de pines de la Raspberry Pi 2 modelo B……12
Figura 1.10 Logo Arduino……13
Figura 1.11 Arduino UNO……14
Figura 1.12 Arduino ethernet shield……15
Figura 1.13 Sensor de luz BH1750……..16
Figura 1.14 Sensor de humedad y temperatura DHT22……17
Figura 1.15 Sensor sumergible de temperatura DS18B20……17
Figura 1.16 Sensor de pH SKU: SEN0161……..18
Figura 1.17 Sensor de conductividad eléctrica SKU: DFR0300……...18
Figura 1.18 Representación de un archivo en una base de datos……..20
Figura 1.19 Esquema cliente – servidor……22
Figura 1.20 Logo App Inventor……...27

CAPÍTULO II

Figura 2.1 Diagrama esquemático referencial del prototipo……....28
Figura 2.2 Diagrama de flujo referencial de programas……29
Figura 2.3 Diagrama de bloques referencial del prototipo……30
Figura 2.4 Instalación del OS Raspbian Jessie en la micro SD con Win32 Disk Imager……31
Figura 2.5 Arranque del OS de la Raspberry Pi 2……...32
Figura 2.6 Escritorio del OS Raspbian……..32
Figura 2.7 Herramientas de configuración de la Raspberry Pi……..33
Figura 2.8 Pantalla de inicio del programa Arduino……....34

Figura 2.9 Diagrama de flujo para la programación en Arduino……34
Figura 2.10 Página por defecto del servidor Apache2……40
Figura 2.11 Página por defecto al instalar PHP……...41
Figura 2.12 Monitor de inicio de MySQL……....42
Figura 2.13 Archivo de configuración php.ini……..43
Figura 2.14 Página de inicio de phpMyAdmin……43
Figura 2.15 Creación de usuario en phpMyAdmin……44
Figura 2.16 Creación de la base de datos repsensores……...45
Figura 2.17 Creación de la tabla conductividad_electrica……45
Figura 2.18 Configuración de las columnas de la tabla conductividad_electrica...46
Figura 2.19 Base de datos repsensores y sus respectivas tablas……47
Figura 2.20 Diagrama de flujo para la obtención y envío de información a la base de datos……..48
Figura 2.21 Scripts creados en la carpeta reportsensores……50
Figura 2.22 Archivos creados para obtener el historial de los sensores……53
Figura 2.23 Archivos creados para obtener el reporte de los sensores……..54
Figura 2.24 Pantalla de inicio para programar en App Inventor……....54
Figura 2.25 Pantalla de presentación inicial de la aplicación……....55
Figura 2.26 Programación del botón LOGIN……....55
Figura 2.27 Pantalla de autenticación para el usuario……...56
Figura 2.28 Programación en bloques de la pantalla AUTENTICACIÓN…….....57
Figura 2.29 Mensaje de acceso denegado……..57
Figura 2.30 Diseño de la pantalla ZONAS……58
Figura 2.31 Programación en bloques de la pantalla ZONAS……..59
Figura 2.32 Archivos creados para obtener el último dato de los sensores……60
Figura 2.33 Pantalla TSZONA1……..61
Figura 2.34 Programación en bloques de la pantalla TSZONA1 para el sensor de CE……62
Figura 2.35 Diseño de la pantalla HISTORIAL_Y_REPORTES……...63
Figura 2.36 Programación en bloques de la pantalla HISTORIAL_Y_REPORTES para el sensor de CE……64
Figura 2.37 Diagrama de flujo para el control de los puertos GPIO de la Raspberry Pi……..65

Figura 2.38 Diagrama del circuito de potencia para un sensor……69
Figura 2.39 Simulación del circuito de potencia……..70
Figura 2.40 Diagrama de bloques total para la placa…..……..70
Figura 2.41 Diagrama esquemático total del circuito……..71
Figura 2.42 Simulación del circuito total……...72
Figura 2.43 Circuito impreso para la placa……...73
Figura 2.44 Configuración de red en el modem……..73
Figura 2.45 Forwarding de puertos para el servidor Apache……74

CAPÍTULO III

Figura 3.1 Proceso de planchado en la baquelita……...75
Figura 3.2 Limpieza y recorte de la baquelita……..75
Figura 3.3 Proceso de marcación de pistas en la baquelita……..76
Figura 3.4 Soldadura de los elementos en la placa…….....76
Figura 3.5 Placa implementada……..77
Figura 3.6 Cable de conexión de los sensores hacia la placa……..77
Figura 3.7 Cables de alimentación…….....78
Figura 3.8 Prototipo implementado……79
Figura 3.9 Prueba de obtención de datos desde los sensores hacia Arduino……80
Figura 3.10 Prueba de envío de información desde Arduino a la base de datos..81
Figura 3.11 Funcionamiento de la aplicación en diferentes dispositivos Android..82
Figura 3.12 Funcionamiento de la aplicación con todas las alarmas activadas….82
Figura 3.13 Obtención de datos para generar el historial y reportes……...83
Figura 3.14 Funcionamiento del ventilador……..84
Figura 3.15 Funcionamiento de la lámpara fluorescente……...84
Figura 3.16 Funcionamiento del calentador de agua…….85
Figura 3.17 Funcionamiento de la lámpara led.....86
Figura 3.18 Alarma auditiva……86
Figura 3.19 Prueba de conexión hacia el servidor mediante la ip pública desde internet……87
Figura 3.20 Prueba de conexión hacia el servidor mediante el dominio acuario.epn.edu.ec……...88

ANEXO A

Figura A-1 Proceso de instalación de MIT AI2 Companion……A1
Figura A-2 Instalación de la aplicación Android……A2
Figura A-3 Logo y pantalla de inicio de la Aplicación Android……A2
Figura A-4 Mensajes de autenticación fallida……A3
Figura A-5 Pantalla ZONAS para el usuario ROOT……A4
Figura A-6 Pantalla de datos e Historial y Reportes de la zona 1……A4

ANEXO C

INFORMACIÓN TÉCNICA DE SENSORES Y ELEMENTOS UTILIZADOS…....C1

ÍNDICE DE TABLAS

CAPÍTULO I

Tabla 1.1 Comparación entre los estándares 802.11b/g/n……..6
Tabla 1.2 Algunos puertos de red conocidos……...19
Tabla 1.3 Comparación entre sistemas operativos……26

CAPÍTULO II

Tabla 2.1 Nombres de tablas y columnas de la base de datos repsensores…….47
Tabla 2.2 Nombres de scripts y variables para el envío de datos……49

CAPÍTULO IV

Tabla 4.1 Presupuesto referencial del prototipo……..90

ANEXO A

Tabla A-1 Autenticación para cada zona……..A3

ÍNDICE DE CÓDIGOS

CAPÍTULO II

Código 2.1 Programación para obtener el dato de temperatura en el agua...……35
Código 2.2 Programación para obtener el dato de conductividad eléctrica……37
Código 2.3 Programación para obtener el dato de temperatura y humedad ambiental……38
Código 2.4 Programación para obtener el dato de iluminación……39

Código 2.5 Programación para obtener el nivel de pH……...40
Código 2.6 Script conexion.php……48
Código 2.7 Script repce.php……49
Código 2.8 Configuración de la ethernet shield..……51
Código 2.9 Envío de información desde Arduino a la base de datos para el sensor de CE……51
Código 2.10 Script histce.php……52
Código 2.11 Script reportce.php……53
Código 2.12 Script setrepce.php……60
Código 2.13 Programa control_gpio.py para el control de los puertos GPIO…….67

ANEXO B

Código B-1 Programa envio_sensores_arduino.ino……..B5
Código B-2 Programa control_gpio.py ……..... B7
Código B-3 Programa conexion.php……B7
Código B-4 Programa repce.php……...B7
Código B-5 Programa reph.php……B8
Código B-6 Programa rephumamb.php……..... B8
Código B-7 Programa repilum.php…….....B8
Código B-8 Programa reptempagua.php……B8
Código B-9 Programa reptemamb.php……B9
Código B-10 Programa setrepce.php…….....B9
Código B-11 Programa setreph.php ……...B9
Código B-12 Programa setrephumamb.php……B9
Código B-13 Programa setrepilum.php……..B10
Código B-14 Programa setreptempagua.php……B10
Código B-15 Programa setreptempamb.php……B10
Código B-16 Programa fsetrepce.php……B10
Código B-17 Programa histce.php……..B11
Código B-18 Programa histempagua.php…….....B11
Código B-19 Programa histempamb.php……...B12
Código B-20 Programa histhumamb.php……...B12
Código B-21 Programa histilum.php……...B13
Código B-22 Programa histph.php……..B13

Código B-23 Programa reportce.php……..B14
Código B-24 Programa reportempagua.php……..... B14
Código B-25 Programa reportempamb.php……...B15
Código B-26 Programa reporthumamb.php……...B15
Código B-27 Programa reportilum.php……...B16
Código B-28 Programa reportph.php…….....B16

RESUMEN

El presente Proyecto se encuentra enfocado al desarrollo de una aplicación para el Internet de las cosas (IoT), en base a la implementación de un prototipo para el monitoreo de acuarios por medio de la utilización de software libre y tecnologías de bajo costo, con el fin de proporcionar una solución confiable, económica y fácil de utilizar para el usuario.

Para cumplir con este propósito se utiliza una comunicación entre la plataforma Raspberry Pi y Arduino mediante el modelo cliente – servidor, para el monitoreo de los parámetros del acuario, Arduino utiliza sensores de pH, conductividad eléctrica, temperatura y humedad ambiental, luz y temperatura en el agua para obtener y enviar información a una base de datos que se encuentra almacenada en la Raspberry Pi.

En el desarrollo de esta aplicación Android, se permite visualizar por medio de Internet los parámetros y las alarmas que pudieran existir en el acuario mediante cualquier dispositivo que tenga acceso a la red.

El prototipo realiza acciones automáticamente cuando detecta parámetros fuera de rango, mediante la activación de equipos eléctricos que funcionan a 120 V permitiendo controlar y mejorar el hábitat en el acuario.

PRESENTACIÓN

Para preservar la vida dentro de un acuario se deben controlar los parámetros que influyen en sus condiciones ambientales, para ello el presente Proyecto implementa un prototipo orientado a mejorar el hábitat dentro del acuario mediante el uso de tecnologías enfocas en el IoT.

El Proyecto se desarrolla en cinco capítulos, el primer capítulo abarca la teoría relacionada con el IoT, el hábitat en acuarios, descripción de las plataformas Raspberry Pi y Arduino, sensores, base de datos y desarrollo de aplicaciones Android.

En el segundo capítulo se incluye el diseño del hardware y software de cada una de las partes que conforman el prototipo para el monitoreo de acuarios.

En el tercer capítulo se describe la implementación del prototipo y las correspondientes pruebas de funcionamiento en cada una de las etapas para la validación del prototipo implementado.

En el cuarto capítulo se incluye el presupuesto referencial para la implementación de prototipo.

En el quinto capítulo se describen las conclusiones y recomendaciones obtenidas en base al desarrollo e implementación del prototipo.

Finalmente en los anexos se adjunta el manual de usuario de la aplicación Android, los datos técnicos de los sensores y los códigos fuente de los programas desarrollados para el funcionamiento del prototipo.

CAPÍTULO I

1. INTRODUCCIÓN AL IoT

En este capítulo se realiza la descripción de los parámetros y características principales del hábitat en un acuario, con el fin de poder monitorearlo con el uso de tecnologías accesibles y de bajo costo para el usuario. De esta manera se logra la integración entre hardware y software necesarios para conseguir la interconexión de los parámetros del acuario con el Internet, facilitando al usuario el acceso al sistema en cualquier momento y lugar por medio de una red, aplicando de esta forma el concepto del IoT[1].

1.1 GENERALIDADES DEL HÁBITAT EN UN ACUARIO

Un acuario se puede definir como una estructura sólida de diferentes formas y tamaños en la que su parte frontal será de algún material trasparente, en cuyo interior contendrá agua dulce, salada o salobre, necesaria para recrear un ecosistema capaz de mantener vida en su interior y preservar especies en peligro de extinción tales como plantas, peces, reptiles, etc. El término acuario tiene su origen en los vocablos latinos “aqua” = ‘agua’ y “rium” = ‘lugar’ o ‘edificio’ y se dieron a conocer tal como en la actualidad en el siglo XVIII aproximadamente. En la figura 1.1 se representa un Acuario tradicional [1].

Figura 1.1 Representación de un acuario tradicional [1]

[1] IoT: “Internet of Things” (Internet de las Cosas).

1.1.1 TIPOS DE ACUARIOS

Los acuarios pueden clasificarse de diferente manera, en el presente Proyecto se considera la siguiente clasificación: de agua salada, agua dulce y agua salobre.

1.1.1.1 Acuario de agua salada [1]

También conocidos como acuarios marinos, son los más difíciles de preservar y los que ofrecen una mayor gama de biodiversidad en flora y fauna, siendo capaces de recrear ambientes similares a los de mares, océanos y arrecifes. La concentración salina en este tipo de acuarios varía entre 5% y 18%.

1.1.1.2 Acuario de agua dulce [1]

Conocido también como acuario tropical o de agua caliente, son más sencillos de preservar que los acuarios marinos y contienen especies muy resistentes, ya que pueden adaptarse al agua templada o caliente, este tipo de acuario simula un ambiente tropical. Las temperaturas del agua en este tipo de acuarios oscilan entre 22 ºC y 27 ºC y su concentración salina es menor al 0,5 %.

1.1.1.3 Acuario de agua salobre [2]

Este tipo de acuario son muy fáciles de mantener, ya que simulan un ambiente en el que se une un río con el mar, sus especies están acostumbrados a cambios moderados en los parámetros del agua. Los ejemplos que describen de mejor manera este tipo de ambientes son los estuarios y manglares. Su concentración salina varía entre 0,5% y 5%.

1.1.2 PARÁMETROS PRINCIPALES PARA PRESERVAR UN ACUARIO [3]

El ecosistema presente en un acuario es sumamente sensible a los cambios en las condiciones ambientales, que pueden ocasionar la extinción de ciertas especies que allí se preservan y por ello es necesario controlar parámetros físicos y químicos tales como: temperatura, pH[2], conductividad, iluminación, humedad ambiental, emisión de gases, etc., haciendo que el monitoreo de estos parámetros sean de vital importancia para preservar la vida que en ellos se contiene.

---

[2] pH: potencial Hidrógeno.

1.1.2.1 Temperatura

Uno de los principales parámetros a considerar en un acuario es la temperatura, dicho parámetro influye en el ritmo biológico de las especies que allí habitan. En el caso de los peces se tiene que a mayor temperatura, mayor es el metabolismo del pez, lo que provoca que consuma mayor energía, teniendo en cuenta que si sobrepasa los niveles permitidos de temperatura, dependiendo del acuario, incluso la especie podría llegar a morir. Para los acuarios de agua salada los rangos aproximados de temperatura oscilan entre 18ºC y 22ºC y para los de agua dulce entre 22ºC y 27ºC.

1.1.2.2 pH

En un acuario, el parámetro pH del agua puede variar según la cantidad de seres vivos y elementos que contenga. El nivel de pH está cuantificado en relación al número de iones libres de hidrógeno presentes en las moléculas de agua, su escala es logarítmica con valores de 0 a 14 considerándose el valor de 7 como pH neutro, los valores menores a 7 como ácidos y los mayores 7 como alcalinos. En la figura 1.2 se representa la escala de pH.

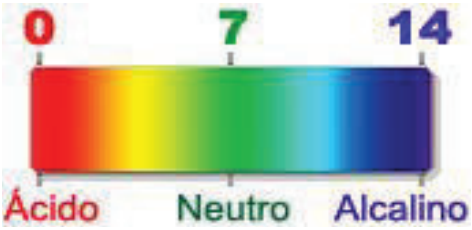


Figura 1.2 Escala de pH [3]

1.1.2.3 Conductividad eléctrica

Este parámetro permite cuantificar la capacidad que tiene el agua para conducir electricidad, que principalmente está determinada por la cantidad de sales que están disueltas en el agua.

Al disolverse en el agua las sales se separan en iones positivos y negativos que hacen posible la conducción de energía eléctrica. De esta manera, a mayor cantidad de sales presentes en el agua, se tendrá una mejor conductividad,

teniendo en cuenta que la conductividad depende también de otros factores como la temperatura y la pureza mineral del agua. La CE[3] se mide en *mS*/cm[4].

1.1.2.4 Iluminación [4]

La iluminación en un acuario permite simular un ambiente natural, brindando equilibrio al ecosistema de los seres vivos, ya que se debe tener en cuenta el ciclo natural día–noche necesario para que las plantas y animales tengan un normal desarrollo. No es recomendable utilizar fuentes de luz que emitan radiación ultravioleta, ya que es perjudicial para todo ser vivo. La unidad de medida de la iluminación es el lux (lx).

1.1.2.5 Humedad ambiental [4]

La humedad ambiental indica la cantidad de vapor de agua presente en el aire y se la puede medir en base a la humedad absoluta o humedad relativa. La humedad absoluta representa el total de vapor de agua contenido en un volumen de aire a una presión y temperatura establecidas. Mientras que la humedad relativa se la expresa en porcentaje, siendo la relación entre la humedad contenida en una masa de aire y la máxima humedad absoluta a una presión y temperatura establecidas.

1.2 TECNOLOGÍA WIFI (802.11 b/g/n) [5]

A partir de los avances tecnológicos en las telecomunicaciones, el IEEE[5] en 1997 saca a luz un nuevo estándar para definir las redes de área local inalámbricas conocido como 802.11, que en su primera versión utilizaba IR[6] para realizar la transmisión. Poco tiempo después se realizaron mejoras en el estándar que incluían métodos de transmisión por radiofrecuencia a 2.4 GHz con técnicas de transmisión FHSS[7] y DSSS[8], siendo los principales inconvenientes las bajas velocidades (1 y 2 Mbps) y el costo elevado de los equipos.

---

[3] CE: Conductividad Eléctrica.
[4] mS/cm: milisiemens por centímetro.
[5] IEEE: Institute of Electrical and Electronics Enginieers (Instituto de Ingenieros Eléctricos y Electrónicos).
[6] IR: Rayos Infrarrojos, es un tipo de radiación electromagnética y térmica que abarca frecuencias entre 300 GHz a 400 THz.
[7] FHSS: Frecuency Hopping Spread Spectrum, técnica utilizada para combatir interferencias para 1Mbps.

En 1999 se desarrollan los semiconductores con tecnología de radio a 2,4 GHz, logrando abaratar costos y permitiendo el desarrollo de nuevas versione 802.11.

Con el fin de solucionar los problemas de compatibilidad entre equipos de diferentes empresas, surge en 1999 una alianza entre las principales compañías de tecnología inalámbrica conocida como WECA[9], que sería la encargada de garantizar la interoperabilidad entre equipos que lleven la certificación Wi-Fi[10] independientemente del fabricante. En la figura 1.3 se muestra el logotipo Wi-Fi.

Figura 1.3 Logotipo para dispositivos certificados por Wi-Fi Alliance [5]

En la actualidad se han desarrollado diferentes versiones a partir del estándar 802.11, buscando diferentes mejoras y compatibilidad entre versiones anteriores. A continuación se describen las características de 802.11 b/g/n, que se consideran son los más comúnmente fabricados por su compatibilidad en las diferentes redes.

1.2.1 IEEE 802.11b

Aparece en el año 1999 con una gran aceptación entre los usuarios ya que logra transmitir a 2.4 GHz, con velocidades teóricas de 5.5 Mbps y 11Mbps a distancias aproximadas de 300m en ambientes abiertos y 50m en ambientes cerrados sin interferencias. Utiliza técnicas de transmisión DSSS y modulaciones BPSK[11], QPSK[12] Y CCK[13].

1.2.2 IEEE 802.11g [6]

Aparece en el año 2003 operando a 2.4 GHz, con velocidad teórica máxima de 54 Mbps a distancias aproximadas de 100m en ambientes sin interferencias.

---

[8] DSSS: Direct Sequence Spread Spectrum, la portadora es modulada por un código digital para 2 Mbps.
[9] WECA: Wireless Ethernet Compability Alliance.
[10] Wi-Fi: Wireless Fidelity.
[11] BPSK: Binary Phase Shift Keyed , tipo de modulación binaria por desplazamiento de fase.
[12] QPSK: Quadrature Phase Shift Keying, tipo de modulación en cuadratura por desplazamiento de fase.
[13] CCK: Complementary Code Keying, tipo de modulación por código complementario.

Utiliza técnicas de transmisión DSSS con modulaciones OFDM[14] y CCK. Este estándar es compatible con el 802.11b.

1.2.3 IEEE 802.11n [6]

Este estándar fue aprobado en el año 2009, opera en la banda de los 2.4GHz con compatibilidad para 802.11b/g y a 5GHz siendo compatible con el 802.11a. Utiliza multiplexación espacial a través de técnicas MIMO[15] y modulación OFDM, pudiendo alcanzar velocidades teóricas de hasta 600 Mbps, con un alcance de 300m para ambientes abiertos y 100 m para ambientes cerrados sin interferencia.

En la tabla 1.1 se muestra una comparación entre parámetros técnicos de los estándares presentados.

Estándar	Aprobado	Frecuencia	Modulación	Velocidad	Alcance
802.11b	1999	2.4 GHz	BPSK, QPSK y CCK	11 Mbps	300 m
802.11g	2003	2.4 GHz	OFDM y CCK	54 Mbps	100 m
802.11n	2009	2.4 y 5 GHz	OFDM	600 Mbps	300 m

Tabla 1.1 Comparación entre los estándares 802.11b/g/n

1.3 INTERNET DE LAS COSAS (IoT) [7]

En el año de 1990 John Romkey y Simon Hackett diseñaron una tostadora que se conectaba a Internet, la misma que podía ser controlada desde cualquier computador, dando origen a un futuro potencial para permitir la conexión de objetos a la red.

En 1999 el Ing. Bill Joy fue quien introdujo los primeros conceptos del IoT. El investigador británico Kevin Ashton que trabajaba en el MIT[16], inventó el término Internet of Things y lo oficializó en su artículo publicado en julio del 2009 en el RFID Journal, el cual indicaba la funcionalidad que tendría el poder conectar las cosas hacia el Internet.

---

[14] OFDM: Orthogonal Frecuency Division Multiplexing, modulación multiportadora, utiliza multiples señales de portadora en diferentes frecuencias.

[15] MIMO: Multiple Inputs Multiple Outputs, técnica utilizada para transmitir streams de datos a través de diferentes antenas simultáneamente.

[16] MIT: Massachusetts Institute of Technology.

El IoT básicamente es un concepto que relaciona la interconexión de cualquier dispositivo u objeto hacia el Internet. La figura 1.4 representa un esquema del IoT.

Figura 1.4 El planeta y el IoT [7]

De acuerdo a los cálculos realizados por el Grupo de Soluciones Empresariales para Internet de CISCO (IBSG), en el 2003 existían alrededor de 500 millones de dispositivos conectados, tomando en cuenta que la población mundial aproximada era de 6300 millones para esa fecha y solo empezaban a surgir los dispositivos inteligentes, sin conocerse aún el concepto del IoT, por lo que la relación dispositivos/personas era 0,08. Para el 2010 dicha relación aumento rápidamente a 1.84, debido a que se popularizaron los dispositivos inteligentes como tablets, smartphones y laptops.

A continuación se muestra una estimación a futuro respecto a la relación entre dispositivos y personas. En la figura 1.5, se muestra la relación entre la población mundial y los dispositivos conectados a Internet.

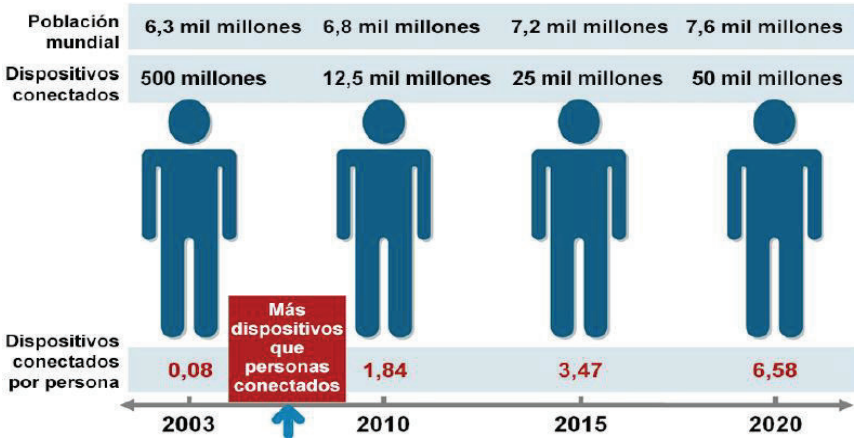


Figura 1.5 Relación de dispositivos conectados por personas [7]

1.3.1 COXIONES EN EL IoT [8]

Las principales conexiones en el IoT son: people to people (P2P), machine to people (M2P) y machine to machine (M2M).

1.3.1.1 Conexión P2P

Es la comunicación que se da entre dos personas caracterizada por aprovechar los recursos existentes en la infraestructura, dispositivos y aplicaciones. P2P también ayuda a una comunicación en línea a través de videoconferencias mediante el Internet.

1.3.1.2 Conexión M2P

Esta conexión establece una interacción entre máquinas y personas, lo cual optimiza la toma de decisiones ya que las personas pueden enviar y recibir información de un sistema.

1.3.1.3 Conexión M2M

Se refiere a la comunicación que ocurre entre diferentes máquinas que pueden estar conectadas mediante cable o de forma inalámbrica.

Se caracterizan por incluir componentes como sensores, actuadores y controladores que procesan e interpretan la información.

En la figura 1.6 se muestra los diferentes tipos de conexiones en el IoT.

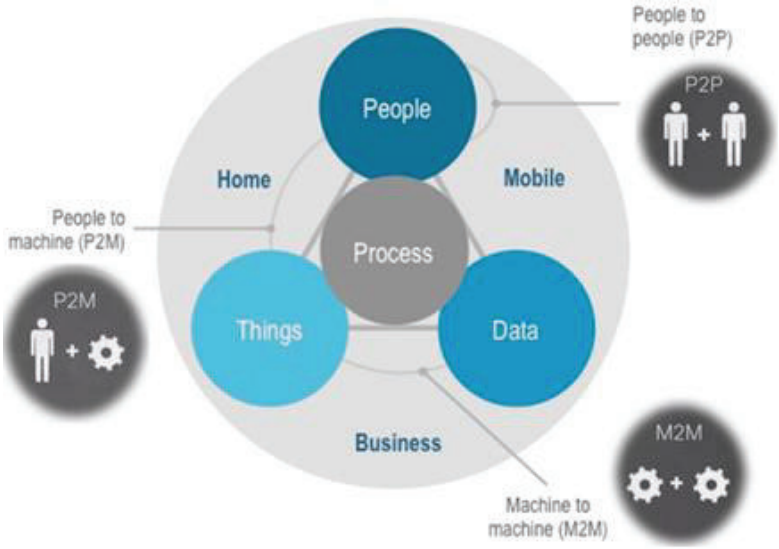


Figura 1.6 Conexiones del IoT [7]

1.3.2 INCONVENIENTES PARA EL DESARROLLO DEL IoT [8]

Existen varios inconvenientes para la consolidación del IoT, entre los principales tenemos:

- Agotamiento de direcciones IPV4[17]: debido al incremento excesivo de dispositivos conectados mediante direcciones IPV4. La solución a este inconveniente se solventará utilizando direccionamiento IPV6[18].
- Poca infraestructura compartida: debido a que la mayoría de objetos no tienen un soporte para la conexión a Internet, se requieren elementos intermedios tales como sensores, actuadores y el software necesario para permitir su acoplamiento a la infraestructura existente.
- Falta de un organismo regulador: que permita la creación de estándares que certifiquen hardware, software y los protocolos que garanticen el funcionamiento del IoT.
- Limitaciones de energía: en la actualidad la mayoría de dispositivos trabajan con baterías de duración limitada, por lo que es necesario incrementar su duración mediante de fuentes de energía renovables.
- Seguridad de la información: Al existir gran cantidad de dispositivos que emiten y reciben señales se debe garantizar que la información llegue de forma confiable a su destino.

1.3.3 APLICACIONES DEL IoT [8]

Existen un gran número de aplicaciones posibles que pueden desarrollarse con la ayuda del IoT, entre las principales se tiene:

- Casas inteligentes
- Ciudades inteligentes
- Automatización de parqueaderos
- Monitoreo de plantas y animales
- Prevención y control en la salud
- Seguridad y rastreo de vehículos

---

[17] IPV4: Protocolo de Internet versión 4.
[18] IPV6: Protocolo de Internet versión 6.

1.4 PLATAFORMA RASPBERRY PI [9]

La Raspberry PI es una placa electrónica muy potente de bajo costo, capaz de realizar funciones como cualquier computadora con la ventaja que posee una estructura física reducida. La idea inicial para construir este tipo de placa surge en el año 2006 con el fin de enseñar a programar a estudiantes de escuelas en el Reino Unido, pero tuvo su desarrollo en el año 2011 con sus versiones Alpha y Beta a cargo de la Fundación Raspberry PI de la Universidad de Cambridge, hasta que finalmente en el año 2012 se comercializa la versión Beta.

En la figura 1.7 se muestra el logo Raspberry Pi.

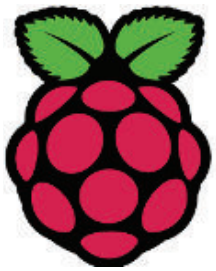


Figura 1.7 Logo Raspberry Pi [9]

Hasta la actualidad se han desarrollado algunos modelos comerciales de esta placa, entre los cuales se tiene:

- Raspberry PI A (Agosto 2011)
- Raspberry PI B (Febrero 2012)
- Raspberry PI B+ (Julio 2014)
- Raspberry PI A+ (Noviembre 2014)
- Raspberry PI 2 MODELO B (Febrero 2015)
- Raspberry PI 3 MODELO B (Febrero 2016)

1.4.1 RASPBERRY PI 2 MODELO B [10]

Esta placa al igual que sus versiones anteriores hace uso de software libre y código abierto, con una mayor capacidad de memoria y trabajando a mayor

velocidad gracias a que dispone de un procesador ARMv7[19], que permite ejecutar la mayoría de distribuciones ARM GNU/Linux[20] así como Microsoft Windows 10.

En la figura 1.8 se muestra el esquema físico de la Raspberry Pi 2 modelo B.

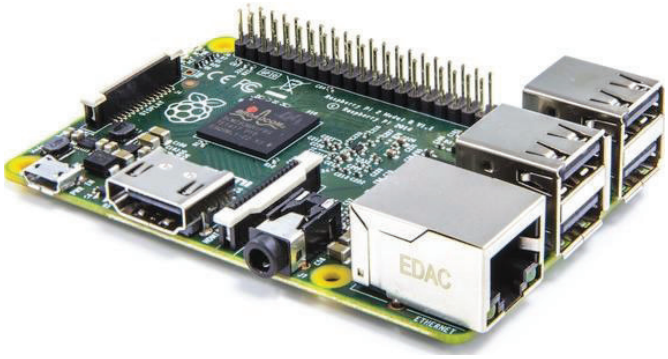


Figura 1.8 Raspberry Pi 2 modelo B [10]

A continuación se indican sus principales características:

- Chip Broadcom BCM2836, con procesador central ARMv7 Cortex-A7 de 4 núcleos a 900 MHz
- Procesador gráfico Broadcom VideoCore IV a 250 MHz
- Memoria RAM[21] de 1GB a 450MHz
- 4 puertos USB[22]
- Puerto HDMI versión 1.4
- Conector de audio y video de 3,5 mm
- Lector de tarjeta microSD
- Puerto Ethernet RJ-45 de 10/100 Mbps
- 40 pines GPIO[23]
- Interfaz de cámara CSI[24]
- Conector para alimentación microUSB
- Interfaz de display DSI[25]

---

[19] ARMv7: Arquitectura del procesador en la versión 7.
[20] GNU/Linux: Sistema operativo con núcleo Linux.
[21] RAM: Random Access Memory.
[22] USB: Universal Serial Bus.
[23] GPIO: General Purpose Input/Output.
[24] CSI: Camera Serial Interface.

- Tamaño (85.6 x 56.5 x17mm)
- Peso (45 gramos)
- Consumo de energía 5V, 2A

1.4.1.1 Puertos GPIO y Python [11]

Los puertos GPIO permiten la comunicación de la Raspberry Pi con otros dispositivos mediante mecanismos de lectura o escritura de información en sus correspondientes pines, que se realiza por medio de algún lenguaje de programación como Python o C.

Los pines de lectura/escritura de la Raspberry Pi 2 no cuentan con un circuito de protección interno frente a magnitudes de voltaje o corriente fuera de rango y en el modelo B se encuentran distribuidos de la siguiente manera:

- Pines de alimentación: dispone de pines GND y VCC limitados a 50mA, de los cuales 2 pines son de 3.3V y 2 de 5V.
- GPIO comunes: varios pines de propósito general configurables como entrada o salida.
- GPIO especiales: utilizados por ciertos buses de comunicación serial como UART[26] e I2C[27], para la comunicación entre placas como por ejemplo Arduino y Raspberry Pi.

En la figura 1.9 se muestra la distribución de pines de la Raspberry Pi 2 modelo B.

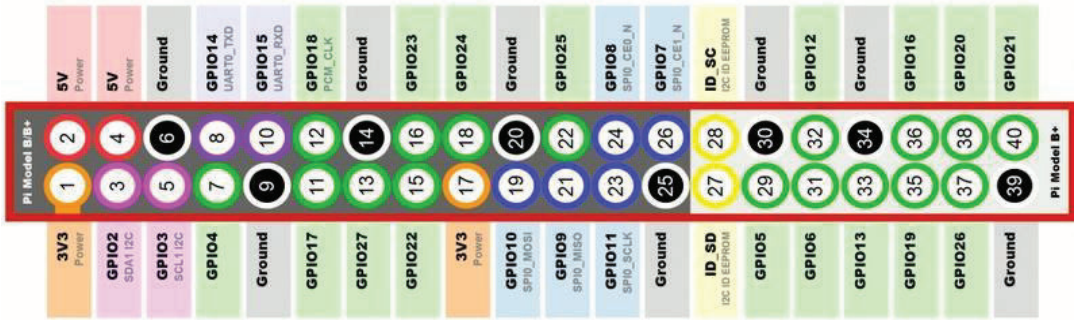


Figura 1.9 Distribución de pines de la Raspberry Pi 2 modelo B [11]

---

[25] DSI: Display Serial Interface.
[26] UART: Universal Asynchronous Receiver-Transmitter.
[27] I2C: Inter-Integrated Circuit.

Por defecto en la Raspberry Pi viene instalado lenguaje Python, siendo este uno de los más utilizados para el control de los puertos GPIO, ya que dispone de una interfaz gráfica y librerías que permiten crear scripts de forma sencilla.

1.5 PLATAFORMA ARDUINO [12]

Arduino es una placa electrónica de código abierto que dispone de hardware y software fáciles de usar, con pines de entrada y salida tanto analógicos como digitales teniendo como principal componente un microprocesador de la familia Atmega de bajo costo, por lo que resulta muy económico para el desarrollo de aplicaciones. Arduino debe su nombre al Bar Italiano di Re Arduino que era visitado por Massimo Banzi, quien en el año 2005 tuvo las primeras ideas de crear una placa electrónica con el fin de ayudar al aprendizaje de los estudiantes en el Instituto IVRAE, ya que las placas existentes en ese momento utilizaban microcontrolares bastante caros.

En la figura 1.10 se muestra en logo de Arduino.

Figura 1.10 Logo Arduino [12]

En la actualidad se han desarrollado diferentes versiones de placas Arduino, en el presente Proyecto se utiliza la plataforma Arduino UNO.

1.5.1 ARDUINO UNO [13]

Fue la primera placa que salió al mercado, sirviendo de referencia para la mayoría de versiones posteriores, está basada en el microcontrolador Atmel ATmega328 y es capaz leer datos tanto analógicos como digitales. Esta placa es compatible con cualquier Ethernet Shield disponible en el mercado.

En la figura 1.11 se muestra el esquema del Arduino UNO.

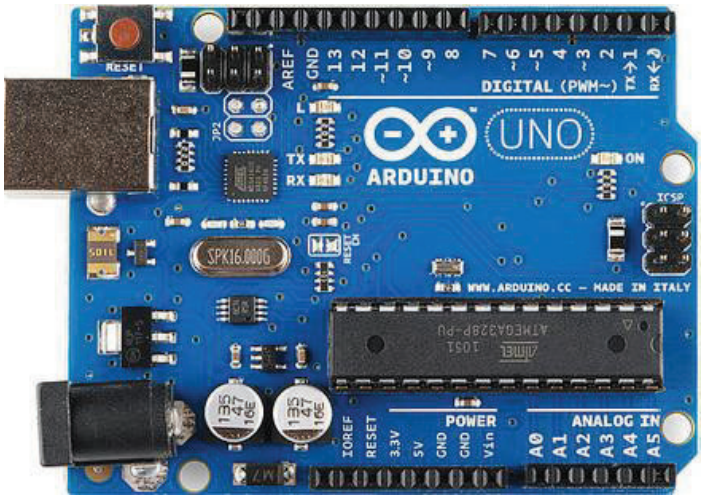


Figura 1.11 Arduino UNO [13]

Las principales características técnicas del Arduino UNO son:

- Microcontrolador Atmega328
- 14 pines digitales de entrada y salida
- 6 pines para señales analógicas
- Memoria flash 32 KB
- SRAM[28] 2KB
- EEPROM[29] 1KB
- Frecuencia de reloj 16 MHz
- Interfaz USB
- Botón para Reset
- Consumo de energía 5V, 40mA
- Tamaño (68.6 x 53.4 x12.5mm)
- Peso (25 g)

1.6 ARDUINO ETHERNET SHIELD [14]

Este módulo al acoplarse a la placa Arduino y mediante el uso de la librería *Ethernet.h* permite la conexión a Internet por medio del puerto Ethernet que posee.

---

[28] SRAM: Static Random Access Memory.
[29] EEPROM: Electrically Erasable Programmable Read-Only-Memory.

Al igual que cualquier componente Arduino, la Ethernet Shield también es de código abierto y de libre acceso. En la figura 1.12 se muestra el esquema de la Ethernet Shield para Arduino.

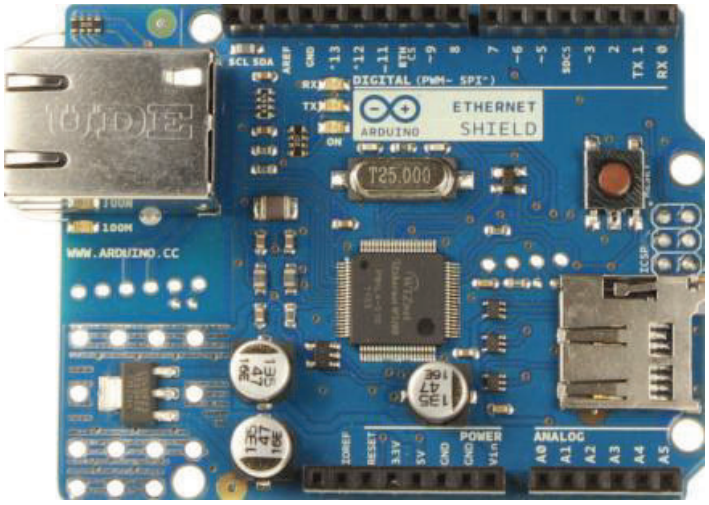


Figura 1.12 Arduino ethernet shield [14]

Arduino Ethernet Shield tiene las siguientes características:

- Controlador Ethernet W5100
- Velocidad de conexión 10/100 Mbps
- Conexión hacia la placa Arduino mediante el puerto SPI[30]
- Alimentación de 5V proporcionada por la placa Arduino
- Lector de memoria microSD

1.7 SENSORES [15]

Es un componente que tiene la capacidad de obtener información de magnitudes físicas o químicas y transformarlas normalmente en señales eléctricas, para de esta manera poder interpretar los datos obtenidos.

1.7.1 CLASIFICACIÓN DE LOS SENSORES

Los sensores pueden clasificarse de diferentes maneras, una de ellas es la siguiente:

---

[30] SPI: Serial Peripheral Interface.

- Eléctricos: capaces de medir las magnitudes relacionadas con la Ley de Ohm.
- Magnéticos: pueden medir el flujo y el campo magnético.
- Mecánicos: miden magnitudes físicas como presión, torque y fuerza.
- Térmicos: capaces de medir variaciones relacionadas con el calor, como la temperatura y el calor específico.

A continuación se describen las principales características de los sensores utilizados para el presente Proyecto.

1.7.1.1 Sensor de luz BH1750 [16]

Es un sensor digital que nos permite medir la intensidad de luz, dispone de un conversor A/D[31] de 16 bits, su unidad de medida es el lux y el rango de medida es de 0 a 65535 lx con una variación mínima de medición de ± 20%. Este sensor puede funcionar con un voltaje entre 3.3 y 5V.

En la figura 1.13 se muestra el sensor de luz BH1750.

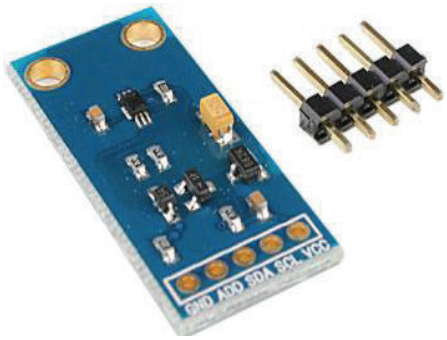


Figura 1.13 Sensor de luz BH1750 [16]

1.7.1.2 Sensor de humedad y temperatura DHT22 [17]

Es un sensor digital de gran precisión que nos permite medir la humedad y temperatura ambiente, pudiendo funcionar con un voltaje entre 3.3 y 6V con un período de sensado mínimo de 2 segundos.

[31] A/D: Análogo/Digital.

El rango de operación para la humedad está entre 0 y 100%RH[32] con una sensibilidad de 0.1%RH y para la temperatura entre -40 y 80 ºC con una sensibilidad de 0.1ºC. En la figura 1.14 se muestra el sensor de humedad y temperatura DHT22.

Figura 1.14 Sensor de humedad y temperatura DHT22 [17]

1.7.1.3 Sensor sumergible de temperatura DS18B20 [18]

Sensor digital sumergible que permite medir la temperatura de un líquido entre -25 y 125 ºC con una sensibilidad de ± 0.5 ºC. Funciona con una fuente de alimentación entre 3 y 5.5V.

En la figura 1.15 se muestra el sensor sumergible de temperatura DS118B20.

Figura 1.15 Sensor sumergible de temperatura DS18B20 [18]

[32] RH: Humedad Relativa, es la unidad de medida de la humedad.

1.7.1.4 Sensor de pH SKU: SEN0161 [19]

Sensor analógico de pH que tiene un rango de medición de 0 a 14 pH con una precisión de ± 0.1 pH a 25 ºC. Utiliza una fuente de alimentación de 5 voltios y su tiempo de respuesta de sensado es menor o igual a 1 minuto.

En la figura 1.16 se muestra el sensor de pH SKU: SEN0161.

Figura 1.16 Sensor de pH SKU: SEN0161 [19]

1.7.1.5 Sensor de conductividad eléctrica SKU: DFR0300 [20]

Permite medir la conductividad eléctrica en el agua en un rango de 1 a 20 mS/cm con una precisión de ± 10%. Es un sensor analógico que opera con una fuente de alimentación de 5 voltios.

En la figura 1.17 se muestra el sensor de conductividad eléctrica SKU: DFR0300.

Figura 1.17 Sensor de conductividad eléctrica SKU: DFR0300 [20]

1.8 REDIRECCIONAMIENTO DE PUERTOS [21]

También conocido como forwarding de puertos, es el proceso que nos permite configurar un router mediante una dirección IP y un lógico puerto de red para poder ingresar a una aplicación específica dentro un equipo determinado, ya sea de forma local o remota. Un puerto lógico de red es una interfaz virtual de comunicación necesaria para el intercambio de información entre equipos.

Un puerto es representado mediante un número pre asignado por la IANA[33] y utiliza protocolos TCP[34] y UDP[35] para la comunicación. Existen 65535 puertos, los más conocidos van del 0 al 1023, en cambio los que van del 1024 al 49151 son puertos reservados y los que van del 49152 al 65535 son puertos dinámicos o privados.

La lista completa de puertos de red se la puede encontrar en la página oficial de la IANA. En la tabla 1.2 se indican algunos de los puertos más conocidos.

# Puerto	Nombre	Descripción
20	FTP	File Transfer Protocol (Protocolo de transferencia de datos).
22	SSH	Secure Shell Protocol (Servicio de Shell seguro), protocolo de acceso remoto seguro.
23	TELNET	Telecomunication Network, protocolo de acceso remoto.
25	SMTP	Simple Mail Transfer Protocol.
53	DNS	Domain Name Service.
80	HTTP	Hypertext Transfer Protocol, usado para servicios World Wide Web (WWW).
3306	MySQL	Utilizado en sistemas de gestión de base de datos.

Tabla 1.2 Algunos puertos de red conocidos [21]

1.9 BASE DE DATOS [22]

Es un sistema que permite almacenar datos que tienen relación entre sí de forma estructurada, de manera que puedan ser utilizados en forma rápida por usuarios y

[33] IANA: Internet Assigned Numbers Authority.
[34] TCP: Transmission Control Protocol.
[35] UDP: User Datagram Protocol.

programas en un futuro. Las bases de datos surgen por la necesidad de las grandes empresas de guardar gran cantidad de información de manera confiable y que pueda ser administrada en cualquier momento y lugar sin tener que trasladarse personalmente al área de archivos.

1.9.1 COMPONENTES DE UNA BASE DE DATOS

Una base de datos está constituida esencialmente por uno o más archivos (tablas). Cada archivo se compone de filas y columnas a las cuales se les denomina registros y campos respectivamente.

En la figura 1.18 se muestran las partes de las que se compone una tabla.

Figura 1.18 Representación de un archivo en una base de datos [22]

Los componentes de una base de datos son:

- Datos: información almacenada en forma de archivos.
- Hardware: unidades de almacenamiento de información tales como discos duros y servidores.
- Software: es el programa que permite administrar la base de datos.

1.9.2 SISTEMA DE GESTIÓN DE BASE DE DATOS

Conocido también como DataBase Management System (DBMS), es un software muy específico que permite la interacción entre el usuario y la base de datos, manteniendo su confidencialidad integridad y seguridad. El DBMS utiliza lenguajes de definición, manipulación y consulta de datos. Entre los sistemas gestores más conocidos tenemos:

- Microsoft SQL Server
- PostgreSQL
- MySQL
- Oracle
- Microsoft Access

A continuación se describe el gestor de base de datos MySQL que es utilizado en el presente Proyecto.

1.9.2.1 MySQL

Es un sistema gestor de base de datos open source que utiliza el modelo relacional, es multiusuario y multiplataforma. MySQL es muy intuitivo, haciendo fácil su manejo y aprendizaje, lo que permite abaratar costos a la hora de crear y administrar una base de datos.

MySQL trabaja muy bien con el lenguaje PHP, permitiendo su integración con cualquier aplicación que utilice este tipo de lenguaje. Puede trabajar tanto en entorno de texto vía comandos como en entorno gráfico con la ayuda de la herramienta phpMyAdmin.

1.9.2.1.1 phpMyAdmin

Es una herramienta de software libre que se ejecuta bajo un servidor y que permite la administración de MySQL en un entorno gráfico a través de páginas web utilizando lenguaje PHP.

1.10 SERVIDORES [23]

Son equipos informáticos locales o remotos que proveen diferentes tipos de servicios cuando reciben una petición de otras máquinas conocidas como clientes.

La información almacenada en un servidor es entregada a los clientes en forma de páginas web, utilizando el formato HTML o PHP.

En la figura 1.19 se representa el esquema cliente - servidor.

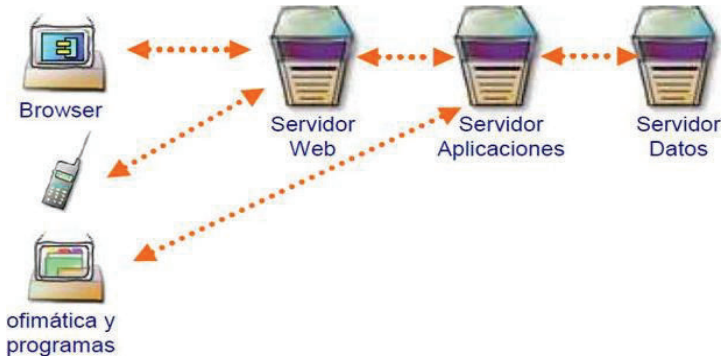


Figura 1.19 Esquema cliente - servidor [23]

Un servidor pude ser de uso dedicado cuando hace uso de todos sus recursos para responder a todas las peticiones de los clientes, o de uso compartido cuando además de responder a peticiones de clientes, se utiliza como cualquier computador de forma local.

1.10.1 TIPOS DE SERVIDORES

Los servidores pueden ser de diferente tipo, teniendo en cuenta la clase de servicio que ofrecen a los clientes y se pueden indicar entre los más importantes los siguientes:

- Servidor de Correo: se encarga de almacenar y ejecutar funciones relacionadas con correos eléctricos de sus clientes, como por ejemplo Outlook, Gmail y Yahoo.
- Servidor de base de datos: almacena y gestiona servicios de bases de datos a clientes, como por ejemplo MySQL server y PortgreSQL server.
- Servidor Web: generalmente almacena información en lenguaje HTML proveniente de un sitio de Internet, como por ejemplo Apache que será utilizado en el presente Proyecto.

1.10.1.1 Servidor Apache [24]

Es un servidor web multiplataforma muy potente que trabaja con software open source, que se caracteriza por su seguridad y rendimiento, convirtiéndolo en el más utilizado para servicios a páginas web.

1.11 WEB HOSTING [25]

También llamado como alojamiento web es el servicio que ofrece una empresa proporcionando espacio de sus servidores, para que los usuarios puedan almacenar y administrar cualquier tipo de información mediante páginas web.

Para elegir un proveedor de web Hosting se debe principalmente tener en cuenta los siguientes parámetros:

- Espacio suficiente para alojar el sitio web teniendo en cuenta un futuro crecimiento.
- Alta disponibilidad para que el sitio web permanezca activo el mayor tiempo posible.
- Soporte técnico garantizado ante una posible falla del servicio.
- Tener en cuenta la relación costo - beneficio al contratar un plan.
- En caso de necesitar un dominio para la página web, se recomienda contratarlo con el mismo proveedor de web Hosting.

Existen varias empresas que proveen el servicio con diferentes tipos de planes, para que el usuario elija el plan que mejor se ajuste a sus necesidades.

1.12 SISTEMAS OPERATIVOS [26]

Un sistema operativo (OS) es un conjunto de programas necesarios para que un equipo funcione. Este software permite controlar los procesos básicos del equipo y la interacción con sus elementos periféricos.

Cualquier sistema operativo cumple principalmente las siguientes funciones: inicializa el hardware del equipo, administra el procesador, gestiona la memoria utilizada para cada aplicación, gestiona los periféricos, controla la ejecución de aplicaciones y mantiene la integridad del sistema.

1.12.1 SISTEMA OPERATIVO ANDROID [27]

Es el sistema operativo desarrollado inicialmente en el 2003 por Android Inc., que posteriormente fue comprada en el 2005 por la compañía Google.

Este OS dispone de una plataforma abierta basada en GNU Linux para dispositivos móviles y sus aplicaciones desarrolladas están basadas en lenguaje Java, por lo que pueden ejecutarse varias de ellas a la vez.

Android cuenta con su propia plataforma de descarga llamada Google Play, la cual permite descargar diferente tipo de aplicaciones en su mayoría gratuitas.

1.12.2 SISTEMA OPERTIVO LINUX [28]

Su nombre correcto es GNU/Linux, está compuesto por un núcleo Linux conjuntamente con herramientas GNU (GNU No es Unix). Es un sistema operativo de software libre, multiusuario, multiplataforma y multitarea, compatible con UNIX[36] que fue desarrollado en 1991 por el estudiante de informática Linus Torvalds de la Universidad de Helsinki (Finlandia).

Linux dispone de varias distribuciones y gestores de ventanas para su entorno gráfico.

1.12.3 SISTEMA OPERATIVO MAC [29]

El sistema operativo Macintosh (MAC OS) fue desarrollado por la empresa multinacional Apple Inc., inicialmente para sus computadores Macintosh.

Este sistema se popularizó por ser el primero en acoplar un mouse al computador para la interacción con la interfaz gráfica permitiendo desplazarse por el escritorio, íconos, menús y ventanas a un bajo costo.

En la actualidad el MAC OS está presente en cualquier dispositivo Apple y dispone de la tienda Mac App Store que permite descargar diferente tipo aplicaciones en su mayoría pagadas.

1.12.4 SISTEMA OPERATIVO WINDOWS [30]

Fue desarrollado desde 1981 por Microsoft Corporation y en 1985 se presenta como complemento del MS-DOS[37]. En 1995 se desarrolla el Win95 que fue la

---

[36] UNIX: Sistema operativo desarrollado por los laboratorios Bell de AT&T en 1969.
[37] MS-DOS: Microsoft Disk Operating System.

primera versión totalmente independiente del MS-DOS. Se encuentra provisto de una interfaz gráfica para el usuario basada en el prototipo de ventanas.

Actualmente es el OS más utilizado y se lo encuentra tanto en computadores como en dispositivos inteligentes.

1.12.5 COMPARACIÓN ENTRE SISTEMAS OPERATIVOS

Dependiendo del área de aplicación el usuario podrá elegir el sistema operativo que mejor se adapte a sus necesidades.

En la tabla 1.3 se muestra la comparación entre sistemas operativos.

SISTEMA OPERATIVO	CARACTERÍSTICAS
[27]	✓ Código abierto ✓ Ideal para dispositivos inteligentes como celulares y tablets ✓ Multitarea y multiusuario ✓ Mayoría aplicaciones gratuitas ✓ Utiliza lenguaje JAVA en sus aplicaciones ✓ Kernel basado en Linux ✓ Equipos con este sistema son de bajo costo ✓ Multimedia y optimiza la batería ✓ Seguro y confiable frente a virus
[28]	✓ Código abierto ✓ Multitarea, multiplataforma y multiusuario ✓ Software libre y Kernel basado en Linux ✓ Existen diferentes tipos de distribuciones ✓ Multimedia ✓ Seguro y confiable frente a virus ✓ Velocidad y estabilidad en su procesamiento ✓ Administración por comandos ✓ Permite partición del disco duro

Mac [29]	✓ Código propietario ✓ Multitarea y multiusuario ✓ Mejor interfaz gráfica del mercado ✓ Muy seguro y confiable frente a virus ✓ Ideal para diseño gráfico ✓ Mejor manejo multimedia ✓ Mejor Velocidad y estabilidad de procesamiento ✓ El más costoso del mercado ✓ Optimización en la batería ✓ Soportado en todos los dispositivos Apple
Windows [30]	✓ Multitarea y multiusuario ✓ Mayor cantidad de software desarrollado ✓ Vulnerable frente a virus ✓ Multimedia ✓ Mejor Velocidad y estabilidad de procesamiento ✓ Mas popular y fácil de conseguir en el mercado ✓ Permite partición del disco duro ✓ Requiere actualizaciones constantes ✓ Existe gran cantidad de soporte ✓ Código propietario

Tabla 1.3 Comparación entre sistemas operativos

1.13 APP INVENTOR [31]

Es un entorno web creado conjuntamente por Google Labs y el MIT para el desarrollo de aplicaciones Android. Utiliza una interfaz gráfica con lenguaje de programación orientado a eventos y basado en bloques muy fácil de usar, por lo que es ideal para programadores sin experiencia, pudiendo crear desde aplicaciones muy sencillas hasta aplicaciones muy complejas.

En la figura 1.20 se muestra el logo de App Inventor.

Figura 1.20 Logo App Inventor [31]

App Inventor dispone de tres herramientas indispensables para la creación de aplicaciones:

- Gestor de Proyectos: donde se almacena las aplicaciones que se van creando.
- Diseñador: permite construir la interfaz gráfica de la aplicación que interactuará con el usuario.
- Editor de bloques: permite hacer la programación interna de cada uno de los componentes que constituyen la aplicación.

Otra herramienta que resulta útil para un usuario es el emulador, que permite probar el avance de la aplicación en desarrollo, sin la necesidad de disponer de un dispositivo Android.

CAPÍTULO II

2. DISEÑO DEL PROTOTIPO PARA EL MONITOREO DE ACUARIOS

En este capítulo se realiza el diseño del prototipo para el monitoreo de acuarios utilizando la plataforma Raspberry Pi, la plataforma Arduino, una aplicación para Android, una red LAN, el conjunto de sensores y un circuito de potencia para accionar equipos eléctricos. En la figura 2.1 se presenta un diagrama esquemático referencial del prototipo a implementarse.

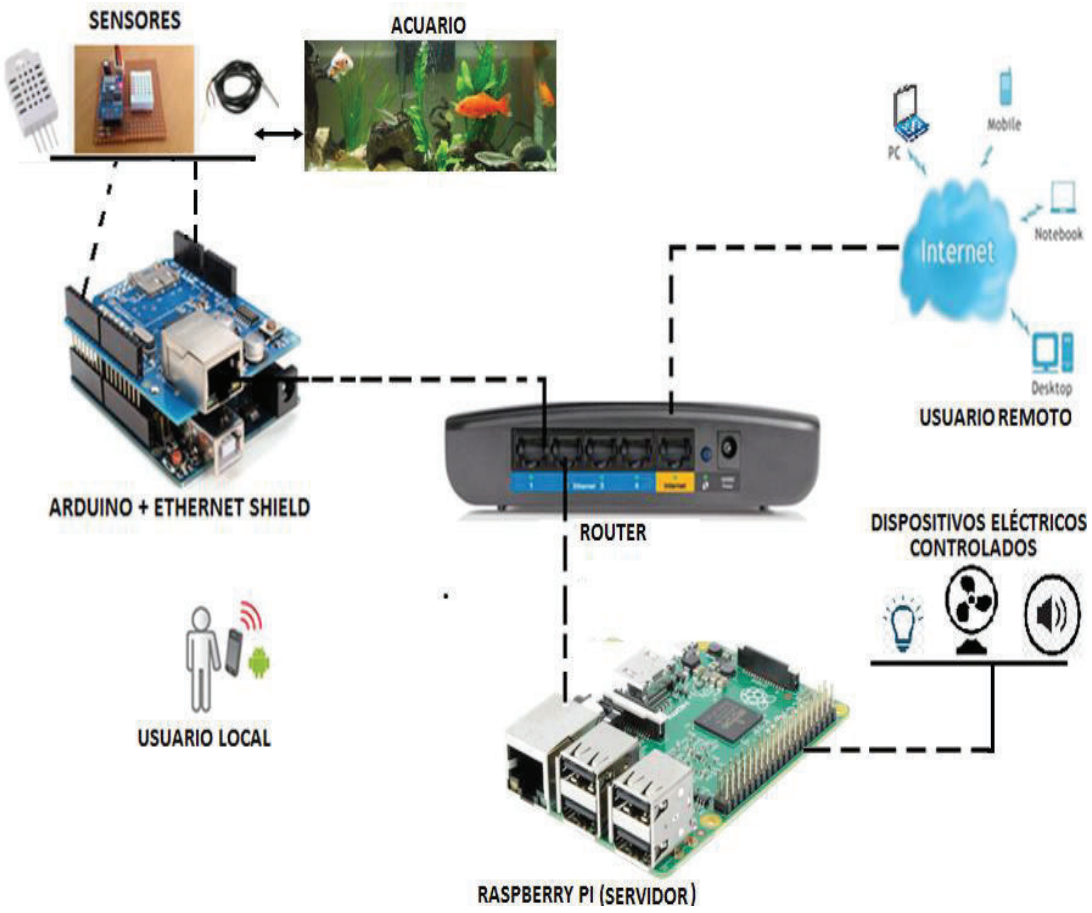


Figura 2.1 Diagrama esquemático referencial del prototipo

Inicialmente se diseña cada etapa del prototipo por separado, luego se va integrando todo el sistema para finalmente comprobar su funcionamiento.

En la figura 2.2 se presenta un diagrama de flujo referencial de programas a desarrollarse para la implementación del prototipo.

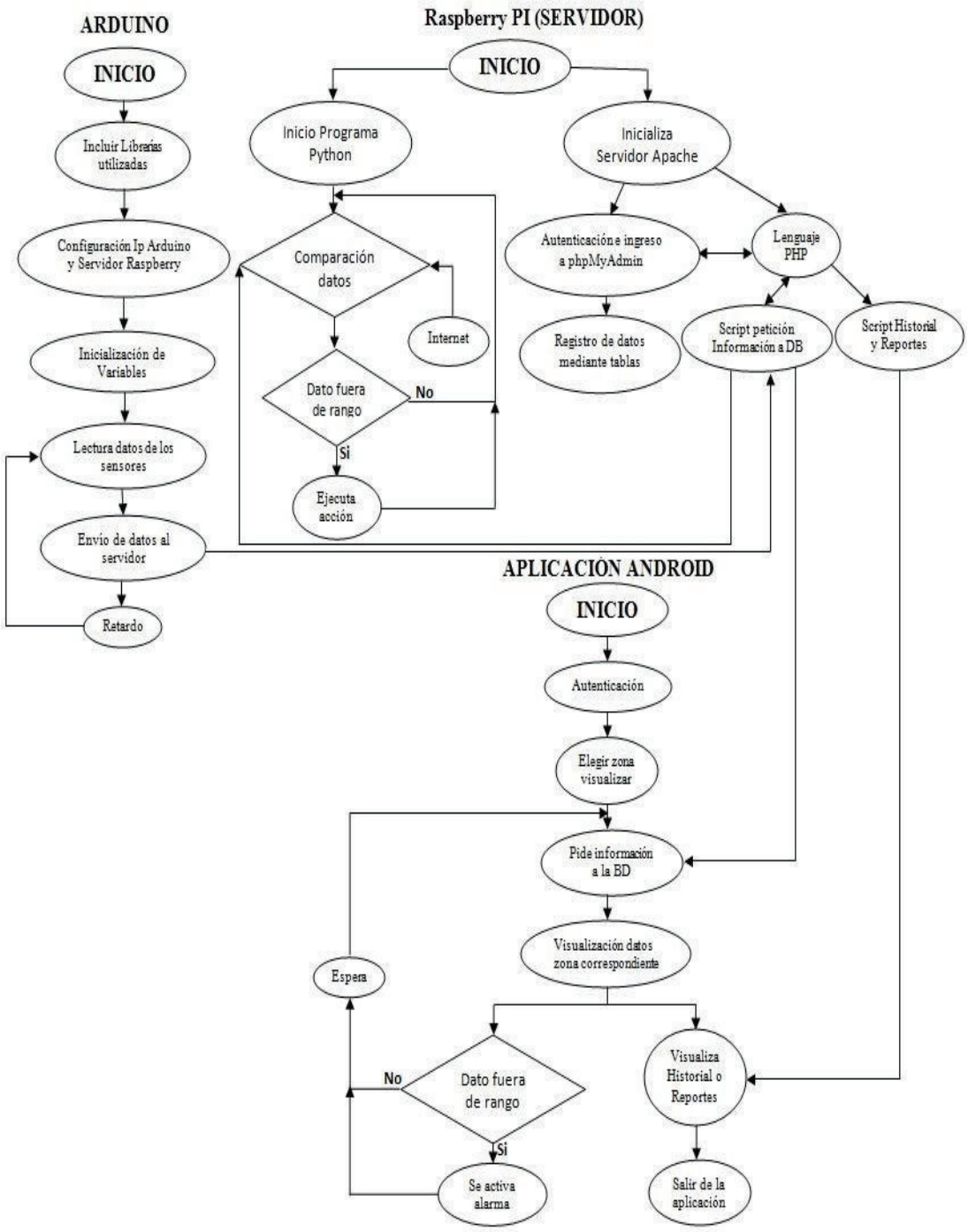


Figura 2.2 Diagrama de flujo referencial de programas

En la figura 2.3 se presenta un diagrama de bloques referencial del prototipo a implementarse.

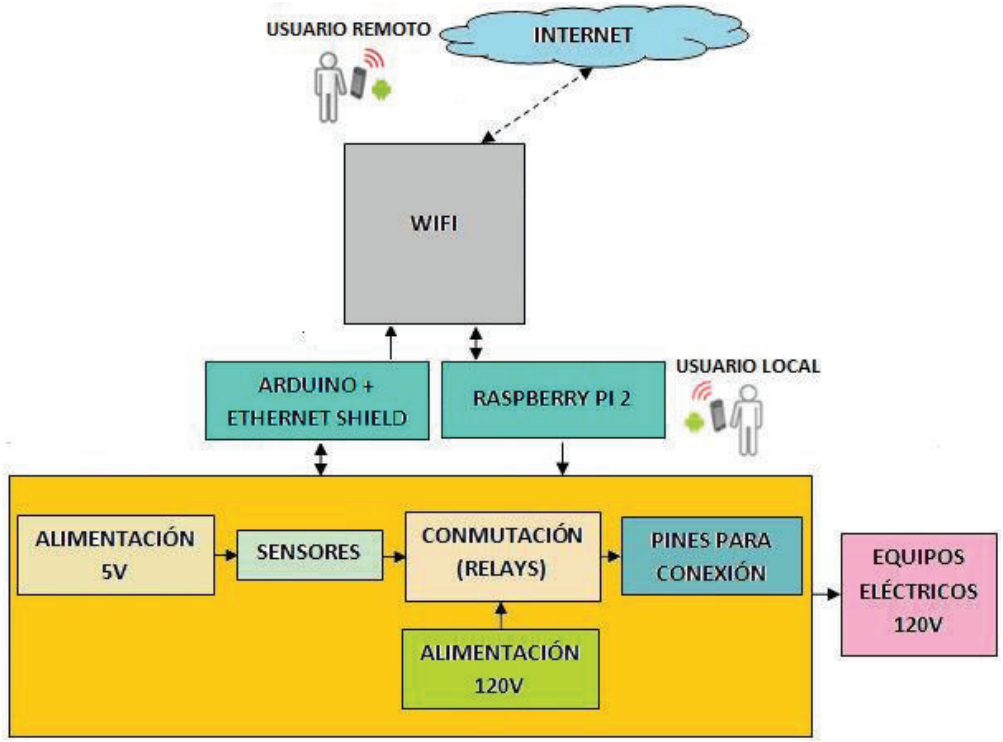


Figura 2.3 Diagrama de bloques referencial del prototipo

2.1 INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA OPERATIVO EN LA RASPBERRY PI 2 [32]

Para su funcionamiento la Raspberry Pi necesita de un OS que debe ser instalado en una tarjeta micro SD mediante algún software que permita la instalación de imágenes, la imagen puede ser descargada desde la página oficial de la fundación Raspberry “https://www.raspberrypi.org”.

2.1.1 INSTALACIÓN DEL OS RASPBIAN JESSIE

Una vez descargada la versión 4.4 de la imagen se procede a cargarla utilizando el software Win32 Disk Imager en una tarjeta micro SD de capacidad mínima de 4GB.

En la figura 2.4 se muestra la instalación de la imagen del OS Raspbian Jessie en la tarjeta micro SD.

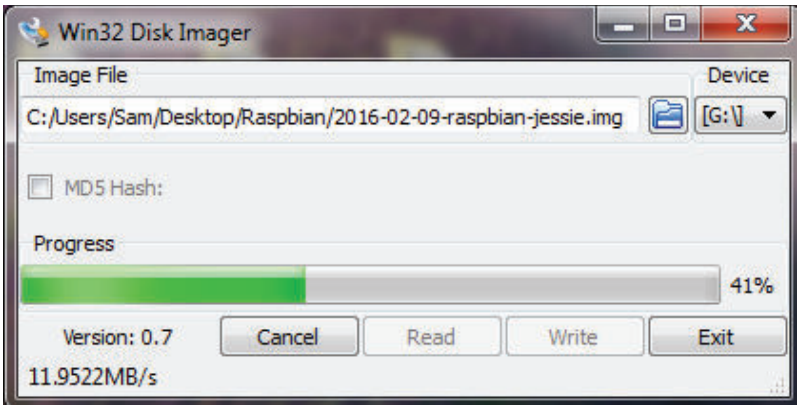


Figura 2.4 Instalación del OS Raspbian Jessie en la micro SD con Win32 Disk Imager

2.1.1.1 Configuración de la resolución de pantalla en la Raspberry Pi 2

Para poder visualizar la interfaz gráfica de la Raspberry Pi mediante una pantalla o monitor convencional, es necesario configurar ciertos parámetros en el archivo config.txt que se encuentra en la tarjeta micro SD una vez cargada la imagen.

Se debe habilitar las siguientes líneas y ajustar los parámetros de acuerdo a la pantalla que se va a utilizar.

- hdmi_force_hotplug=1: por defecto configurado en la opción 1 para forzar la detección Plug and Play en el puerto HDMI.
- hdmi_group=2: por defecto configurado en la opción 1 para pantallas de alta definición o la opción 2, que es recomendada para monitores convencionales.
- hdmi_mode=39: por defecto configurado en la opción 1, debe ser configurado según la resolución y frecuencia del monitor utilizado. En este caso la resolución y frecuencia es de 1360x768 a 60Hz.
- hdmi_drive=2: por defecto configurado en la opción 2 para permitir la salida de audio por HDMI.

2.1.2 CONFIGURACIÓN DE INICIO EN LA RASPBERRY PI

Se introduce la tarjeta micro SD en la Raspberry Pi 2, se conectan todos los periféricos necesarios (mouse, teclado, etc.) y finalmente se energiza el dispositivo con una fuente que proporcione una corriente mínima de 2A.

En la figura 2.5 se muestra la imagen de arranque del OS Raspbian Jessie.

Figura 2.5 Arranque del OS de la Raspberry Pi 2

Una vez finalizado el arranque aparece la imagen del escritorio del OS Raspbian tal como se muestra en la figura 2.6.

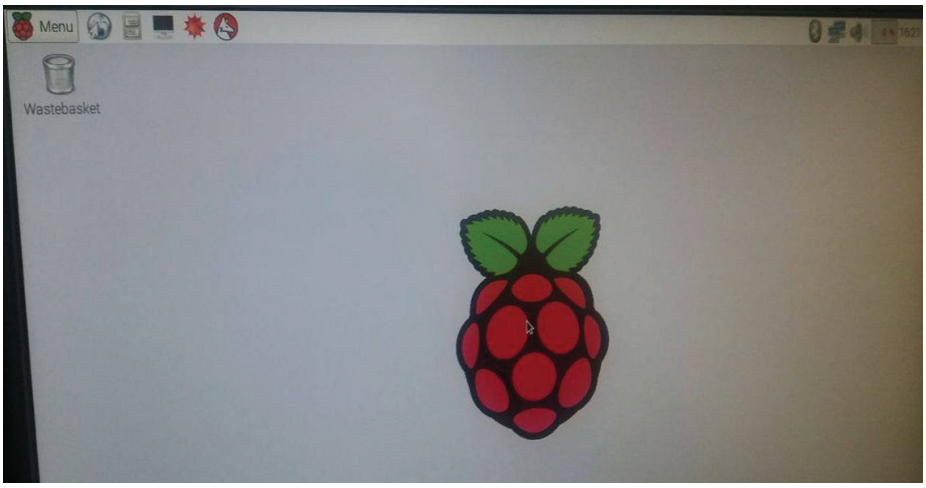


Figura 2.6 Escritorio del OS Raspbian

Mediante el comando *sudo raspi-config* ejecutado desde la terminal se pueden realizar todas las configuraciones necesarias en el sistema. La principal configuración que se debe tomar en cuenta es expandir la partición del OS para que pueda hacer uso de toda la capacidad de memoria de la tarjeta micro SD.

En la figura 2.7 se muestra las herramientas de configuración de la Raspberry Pi.

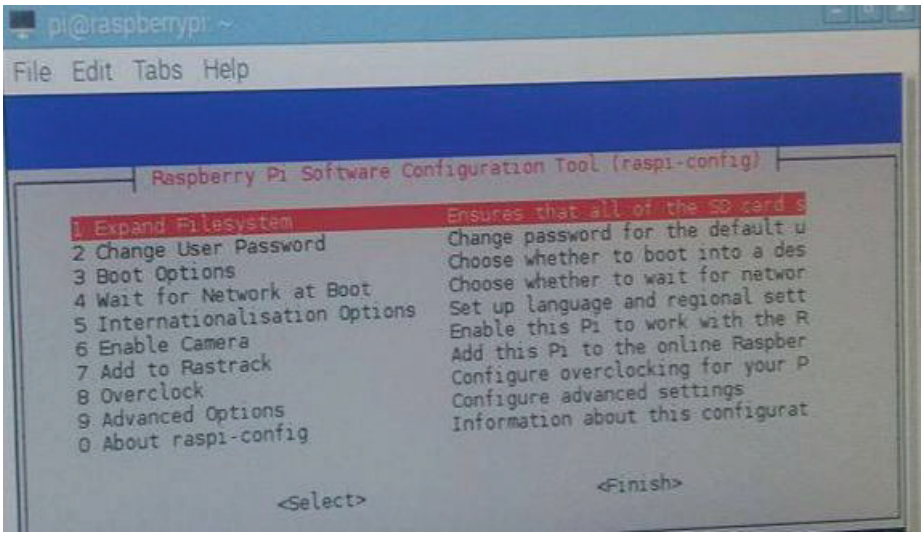


Figura 2.7 Herramientas de configuración de la Raspberry Pi

Una vez realizada todas las configuraciones necesarias se debe reiniciar el sistema.

2.2 INSTALACIÓN DEL SOFTWARE ARDUINO [33]

Antes de instalar cualquier programa en la Raspberry Pi, es necesario ejecutar los siguientes comandos desde la terminal:

- `sudo apt-get update`: para actualizar la lista de paquetes y versiones disponibles.
- `sudo apt-get upgrade`: para actualizar los paquetes disponibles en la versión más reciente.

A continuación se procede a instalar el software Arduino mediante el comando `sudo apt-get install arduino arduino-core`.

Posteriormente se conecta el cable USB desde la placa Arduino hacia la Raspberry Pi, se detectará el puerto por defecto ttyACM0 al cual se dará permisos mediante el comando `sudo chmod 666 /dev/ttyACM0` y de esta manera el programa Arduino estará listo para su uso.

En la figura 2.8 se muestra la pantalla de inicio del programa Arduino.

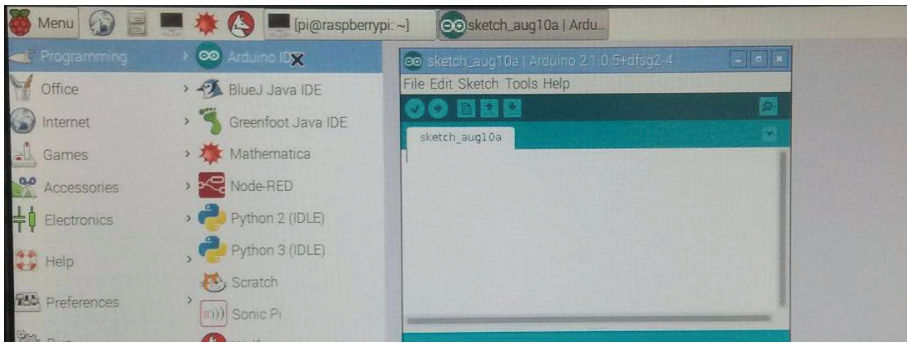


Figura 2.8 Pantalla de inicio del programa Arduino

2.3 DIAGRAMA DE FLUJO PARA OBTENER Y ENVIAR DATOS EN ARDUINO

El diagrama de flujo muestra de manera general las etapas para la obtención de datos de cada uno de los sensores y el envío de información hacia la base de datos. La figura 2.9 muestra el diagrama de flujo para la programación en Arduino.

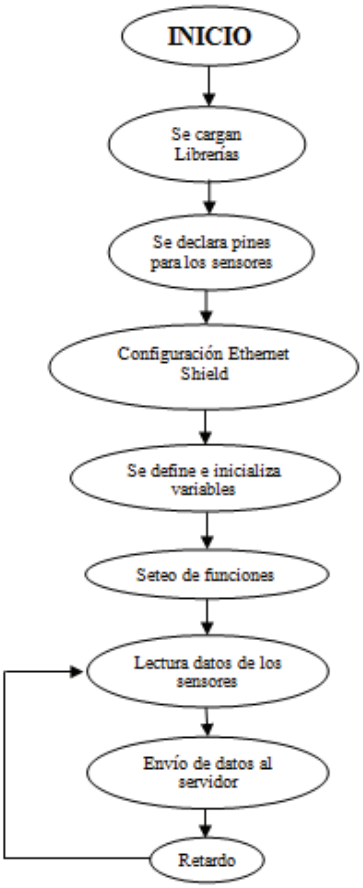


Figura 2.9 Diagrama de flujo para la programación en Arduino

2.4 PROGRAMACIÓN DE LOS SENSORES EN ARDUINO

Para la obtención de los datos de cada uno de los sensores se realiza un código individual que permita visualizar los parámetros sensados. Arduino cuenta con un conjunto de librerías para la ejecución de programas, sin embargo existen sensores que necesitan de librerías adicionales las mismas que se pueden descargar desde Internet y deben ser guardadas en la dirección */usr/share/arduino/libraries* de la Raspberry Pi.

2.4.1 PROGRAMACIÓN PARA EL SENSOR SUMERGIBLE DE TEMPERATURA DS18B20 [18]

El sensor hace uso de las librerías *OneWire.h* y *DallasTemperature.h* para obtener el dato de temperatura del líquido. A continuación en el código 2.1 se presenta la programación para obtener el dato de temperatura en grados centígrados del agua en el acuario.

```
//Se cargan las librerías necesarias para el sensor
#include <OneWire.h>
#include <DallasTemperature.h>
//Se declara el pin donde se conecta el dato del sensor
#define pin18b20 8

// Se define e inicializa las variables
OneWire conex(pin18b20);
DallasTemperature s18b20(&conex);

//Seteo de funciones
void setup()
{
  Serial.begin(9600);
  s18b20.begin();
}

//Lazo de repetición
void loop()
{
  s18b20.requestTemperatures();
  float Tagua= s18b20.getTempCByIndex(0);

// Imprimir variables
  Serial.print("Temperatura_Agua: "););
  Serial.print(Tagua);
  Serial.println(" *C");
  Serial.println("");
  delay(2000);
}
```

Código 2.1 Programación para obtener el dato de temperatura en el agua [18]

2.4.2 PROGRAMACIÓN PARA EL SENSOR DE CONDUCTIVIDAD ELÉCTRICA SKU: DFR0300 [20]

Este sensor trabaja conjuntamente con el sensor DS18B20 para obtener el dato de conductividad eléctrica, ya que el mismo depende de la temperatura a la que se encuentre el líquido.

En el código 2.2 se muestra la programación utilizada para obtener el dato de conductividad eléctrica en el acuario.

```
//Se declara el pin donde se conecta el dato del sensor
byte pinCE = A2;

//Se define e inicializa las variables
const byte nlecturas = 20;
unsigned int intervmuestra=25, intervimpr=700;
unsigned int lecturas[nlecturas];
byte i = 0;
byte j = 0;
unsigned long valorT = 0;
unsigned int valorprom = 0,voltprom=0;
unsigned long tmuestra,timpr;
float CE;
//Seteo de funciones
void setup()
{
  Serial.begin(9600);

  for (i = 0; i < nlecturas; i++)
  lecturas[i] = 0;
  tmuestra=millis();
  timpr=millis();
}
//lazo de repetición
void loop()
{
  if(millis()-tmuestra>=intervmuestra)
  {
    tmuestra=millis();
    valorT = valorT - lecturas[j];
    lecturas[j] = analogRead(pinCE);
    valorT = valorT + lecturas[j];
    j = j + 1;
    if (j >= nlecturas)
    j = 0;
    valorprom = valorT / nlecturas;
  }

  // Imprimir variables
  if(millis()-timpr>=intervimpr)
  {
```

```
    timpr=millis();
    voltprom=valorprom*(float)5000/1024;
    Serial.print("Voltage: ");
    Serial.print(voltprom);
    Serial.print("mV");
    Serial.print("     CE:");

    //Cálculo de CE a diferente temperatura:
    //coefvolt(25°C)=voltprom/(1.0+0.0185*(tempactual-25.0))
    float coeftemp=1.0+0.0185*(27-25.0);
    //El valor 27 será cambiado por "s18b20.getTempCByIndex(0)" el
    //momento en que funcione el sensor de CE en conjunto con el
    //sensor ds18b20
    float coefvolt=(float)voltprom/coeftemp;
    if(coefvolt<150)Serial.println("<1mS/cm");
    else if(coefvolt>3300)Serial.println(">20mS/cm");
    else
    {
      //1ms/cm<EC<=3ms/cm
      if(coefvolt<=448)CE=6.84*coefvolt-64.32;
      //3ms/cm<EC<=10ms/cm
      else if(coefvolt<=1457)CE=6.98*coefvolt-127;
      //10ms/cm<EC<20ms/cm
      else CE=5.3*coefvolt+2278;
      CE/=1000;
      Serial.print(CE,2);
      Serial.println("ms/cm");
    }
  }
      delay(2000);
}
```

Código 2.2 Programación para obtener el dato de conductividad eléctrica [20]

2.4.3 PROGRAMACIÓN PARA EL SENSOR DE TEMPERATURA Y HUMEDAD AMBIENTAL DHT22 [17]

El sensor DHT22 utiliza la librería *DHT.h* para detectar la temperatura y humedad ambiental. El código 2.3 muestra la programación para obtener el dato de temperatura ambiental en ºC y la humedad relativa en %RH.

```
//Se cargan las librerias necesarias para el sensor

#include <DHT.h>

//Se define el sensor y el pin donde se conecta el dato
#define pindht 9
#define num DHT22

//Se define e inicializa la variable
DHT sdht(pindht,num);

//Seteo de funciones
```

```
void setup()
{
  Serial.begin(9600);
  sdht.begin();
}

//Lazo de repetición
void loop()
{
    float H=sdht.readHumidity();
    float T=sdht.readTemperature();

// Imprimir variables

    Serial.print("Humedad: ");
    Serial.print(H);
    Serial.println(" %HR");

    Serial.print("Temperatura_Ambiente: ");
    Serial.print(T);
    Serial.println(" *C");
    Serial.println("");
    delay(2000);
 }
```

Código 2.3 Programación para obtener el dato de temperatura y humedad ambiental [17]

2.4.4 PROGRAMACIÓN PARA EL SENSOR DE LUZ BH1750 [16]

El sensor de luz BH1750 hace uso de las librerías *BH1750.h* y *Wire.h. La librería Wire.h* se instala por defecto junto con Arduino y es la que permite la comunicación I2C habilitando los pines analógicos A4 (SDA[38]) y A5 (SCL[39]).

El código 2.4 indica la programación del sensor de luz BH1750 para obtener el nivel de luminosidad en lx.

```
//Se cargan las librerías necesarias para el sensor
#include <Wire.h>
#include <BH1750.h>

//Se define e inicializa la variable
BH1750 sluz; //

//Seteo de funciones
void setup()
{
  Serial.begin(9600);
```

[38] SDA: Serial Data Line.
[39] SCL: Serial Clock Line.

```
   sluz.begin();
}

//Lazo de repetición
void loop()
{
    uint16_t lx = sluz.readLightLevel();

    // Imprimir variables
    Serial.print("Iluminacion: ");
    Serial.print(lx);
    Serial.println(" lx");
    Serial.println("");
    delay(2000);
 }
```

Código 2.4 Programación para obtener el dato de iluminación [16]

2.4.5 PROGRAMACIÓN PARA EL SENSOR DE PH SKU: SEN0161 [19]

El sensor obtiene el nivel de pH en el acuario a través del pin analógico A3 del Arduino. En el código 2.5 se indica la programación para el sensor de pH SKU: SEN0161 para obtener el nivel de pH en el acuario.

```
//Se declara el pin donde se conecta el dato del sensor
#define pinpH A3
//Se define e inicializa las variables
unsigned int spH;
int k,guardar[10];
float pH;

//Seteo de funciones
void setup()
{
  Serial.begin(9600);
}

//Lazo de repetición
void loop()
{
  for(k=0;k<10;k++)
  {
    guardar[k]=analogRead(pinpH);
    delay (10);
  }

  spH=0;
  for(k=0;k<10;k++) spH+=guardar[k];

  pH=(float)spH*5/1024/10;//Convierte el valor analógico en mv
  pH=3.5*pH;    //Convierte el valor de mv a unidades de pH

 // Imprimir variables
```

```
    Serial.print("pH: ");
    Serial.print(pH);
    Serial.println(" ");
    delay(2000); // Se espera 3 segundos para leer otra vez
}
```

Código 2.5 Programación para obtener el nivel de pH [19]

En el ANEXO C se indican los datos técnicos de cada uno de los sensores.

2.5 DISEÑO DE LA BASE DE DATOS [34] [35] [36]

La base de datos se programa en la Raspberry Pi, para lo cual se requiere instalar el servidor Apache 2, el lenguaje de programación PHP, MySQL y el software phpMyAdmin.

2.5.1 INSTALACIÓN DEL SERVIDOR APACHE 2

En la terminal de la Raspberry Pi se ejecuta el comando `sudo apt-get install apache2` para instalar el servidor Apache2. Para comprobar que se instaló correctamente, se escribe en el navegador *http://localhost y* deberá aparecer una página por defecto como la indicada en la figura 2.10.

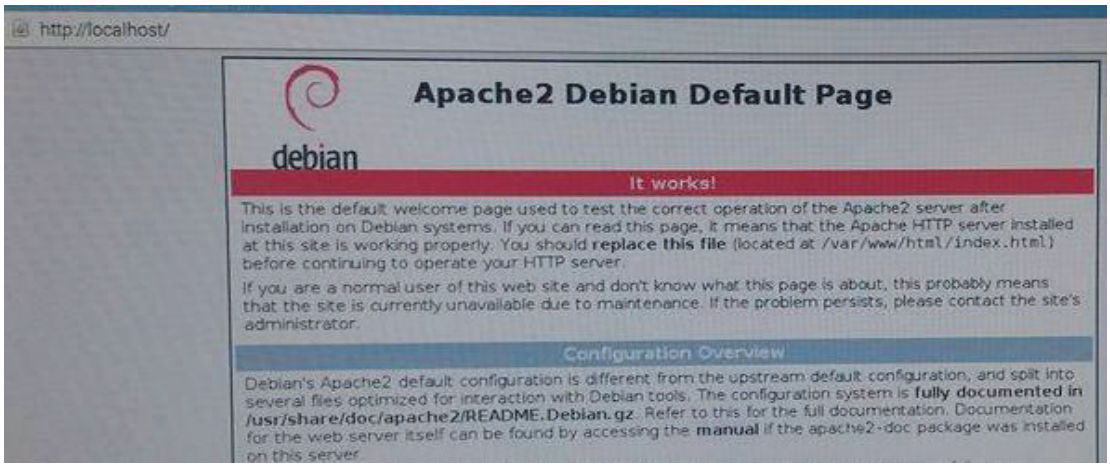


Figura 2.10 Página por defecto del servidor Apache2

2.5.2 INSTALACIÓN DEL PROGRAMA PHP

Para la instalación del programa PHP ejecutamos desde la terminal el comando `sudo apt-get install php5`, luego se instalan paquetes necesarios mediante los comandos `sudo apt-get install libapache2-mod-php5` y `sudo apt-get install libapache2-mod-perl2 php5 php5-cli php5-`

`common php5-curl php5-dev php5-gd php5-imap php5-ldap php5-mhash php5-mysql php5-odbc`.

Posteriormente en la dirección */var/www/html* se crea el archivo "info.php" mediante el comando `sudo nano info.php`, en el cuál se escribe el código `<?php phpinfo(); ?>`.

La figura 2.11 muestra la página por defecto que deberá aparecer en el navegador al escribir *http://localhost/info.php*.

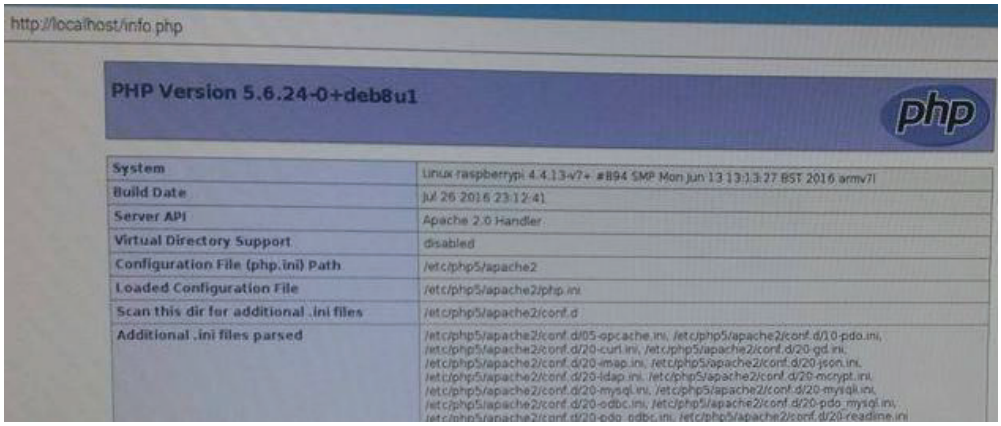


Figura 2.11 Página por defecto al instalar PHP

2.5.3 INSTALACIÓN DE BASE DE DATOS MySQL

Para instalar MySQL se debe ejecutar desde la terminal el comando `sudo apt-get install mysql-server mysql-client php5-mysql`; durante el proceso de instalación se debe configurar un usuario y contraseña de libre elección para finalmente iniciar el servicio mysql mediante el comando `sudo service mysql start`.

Para Ingresar a MySQL se escribe en la terminal el comando `mysql -uroot -hlocalhost -p` y después de ingresar la contraseña configurada durante la instalación, se deberá mostrar una pantalla como la indicada en la figura 2.12.

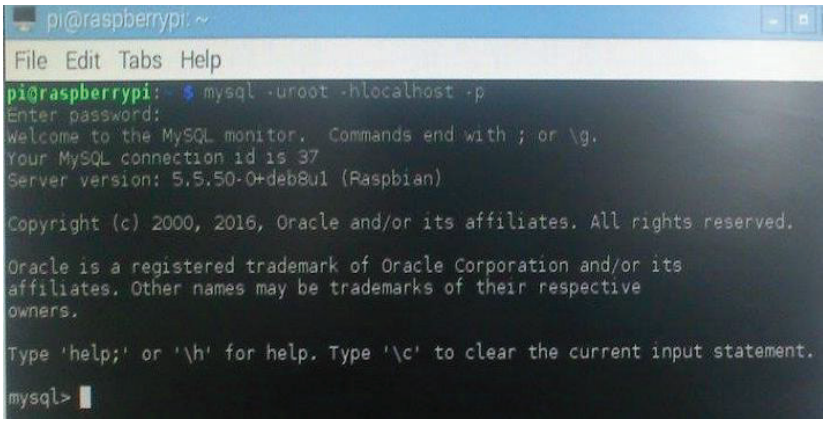


Figura 2.12 Monitor de inicio de MySQL

2.5.4 INSTALACIÓN DE PHPMYADMIN

Para la instalación de phpMyAdmin se ejecuta desde la terminal el comando `sudo apt-get phpmyadmin`; conforme avanza la instalación se debe seguir los siguientes pasos:

- Elegir el servidor web Apache2 que va a ser utilizado por el servicio, ya que es el que se instaló previamente.
- Configurar el acceso a la base de datos por medio del phpMyAdmin, para lo cual se pide primeramente la clave de ingreso a MySQL y posteriormente se debe configurar una clave para ingresar al phpMyAdmin.

Una vez que terminen de instalarse los paquetes, se deben realizar las siguientes configuraciones:

- Ingresar por medio de la terminal al archivo de configuración de Apache2 mediante la instrucción `sudo nano /etc/apache2/apache2.conf` y añadir al final del mismo la línea de configuración `Include /etc/phpmyadmin/apache.conf`.
- Reiniciar el servidor Apache2 mediante el comando `sudo /etc/init.d/apache2 restart`.
- Ingresar al fichero php.ini mediante la instrucción `sudo nano /etc/php5/apache2/php.ini` y añadir la línea `extension=mysql.so`

después de la sección *"Dynamic Extensions"*, tal como se muestra en la figura 2.13.

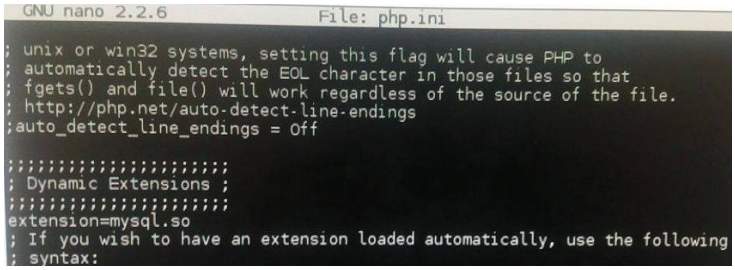


Figura 2.13 Archivo de configuración php.ini

- Se reinicia la Raspberry Pi mediante el comando `sudo reboot`.
- Se crea una carpeta con el nombre "conf.d" mediante la instrucción `sudo mkdir /etc/apache2/conf.d`.
- Se ejecuta el comando `sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf.d/phpmyadmin.conf`.
- Se actualiza el servidor Apache2 con el comando `sudo /etc/init.d/apache2 reload`.
- Finalmente al escribir en el navegador *http://localhost/phpmyadmin* se accede a la página de inicio de phpMyAdmin, tal como se muestra en la figura 2.14.

Figura 2.14 Página de inicio de phpMyAdmin

2.5.5 PUERTO LOGICO DE ACCESO AL SERVIDOR

Con el fin de brindar seguridad a la información y distinguir entre las múltiples aplicaciones que tiene la Raspberry Pi, se opta por cambiar el puerto “80” predeterminado para el servidor Apache 2 por otro número de puerto lógico.

Para cambiar el puerto de escucha en el servidor se modifican los archivos `/etc/apache2/ports.conf` y `/etc/apache2/sites-available/000-default.conf`, cambiando el puerto predeterminado para Apache por el puerto “2801”, finalmente de reinicia el servicio mediante el comando `service apache2 restart`.

2.5.6 CREACIÓN DE LA BASE DE DATOS EN PHPMYADMIN

Para ingresar a la base de datos, el usuario por defecto es “root” con la contraseña configurada al momento de la instalación. En phpMyAdmin se pueden crear usuarios, para ello se ingresa a la opción “Users”, se configura el nombre de usuario que para el presente Proyecto es “ardupi”, la contraseña “inopi”, se brinda privilegios globales y se guarda los cambios tal como se indica en la figura 2.15.

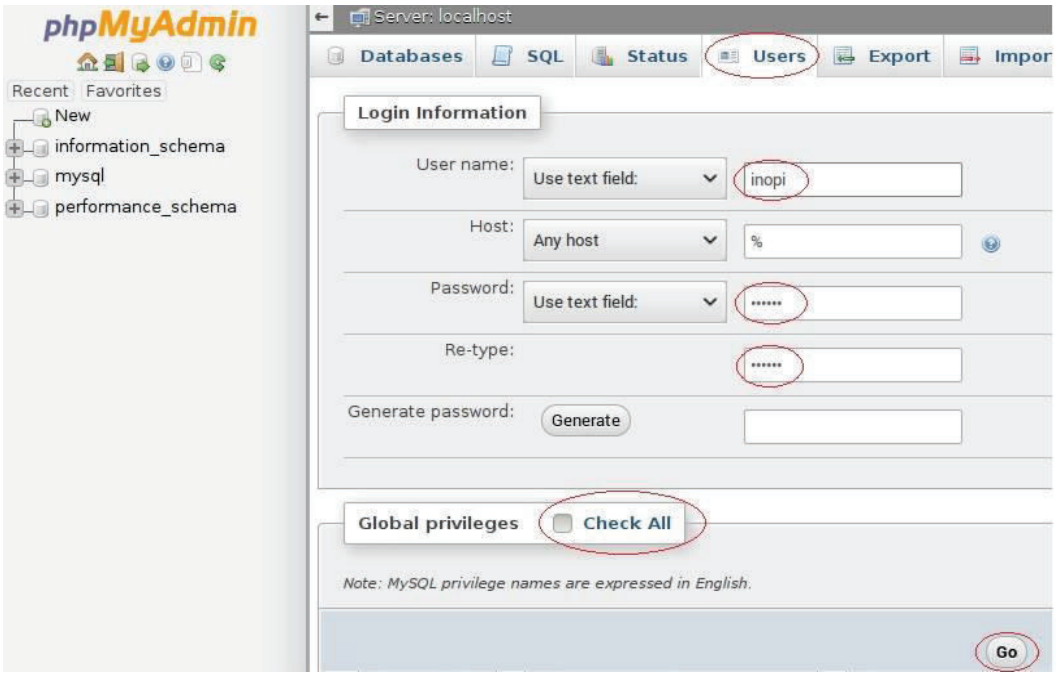


Figura 2.15 Creación de usuario en phpMyAdmin

Se crea una base de datos, la cual contiene tablas independientes que almacenan información de cada uno de los sensores. A continuación se presenta el ejemplo de creación de la base de datos llamada “repsensores” y la tabla “conductividad_electrica”.

Al iniciar sesión en phpMyAdmin se busca la opción “New” para crear una nueva base de datos, luego en “Database name” se indica el nombre que para el presente Proyecto es “repsensores”, en “Collation” se escoge “utf8_general_ci” y finalmente se selecciona la opción “crear”, tal como se indica en la figura 2.16.

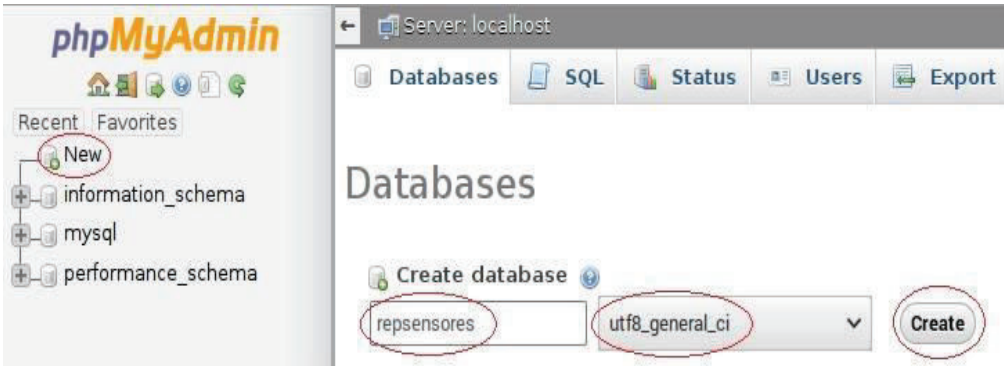


Figura 2.16 Creación de la base de datos repsensores

Se selecciona la base de datos “repsensores”, se crea una tabla de 2 columnas con el nombre “conductividad_electrica” y se selecciona la opción “Go”, tal como se indica en la figura 2.17.

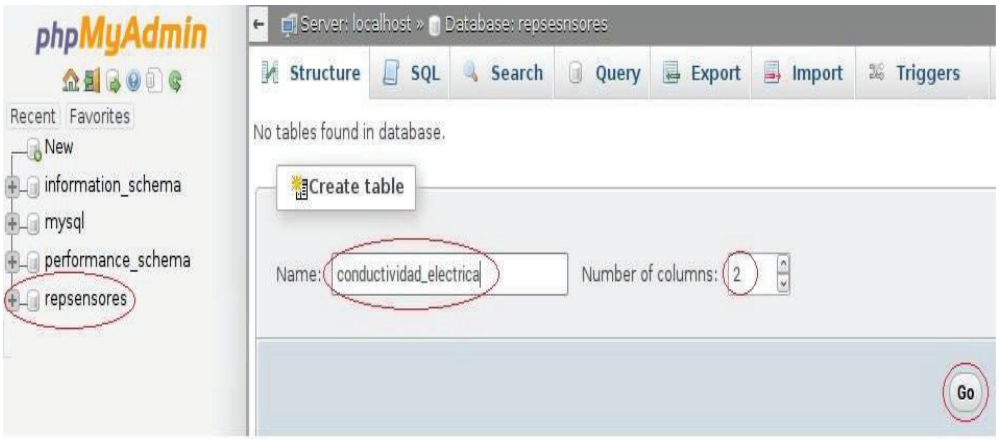


Figura 2.17 Creación de la tabla conductividad_electrica

A continuación se configuran las 2 columnas de las tablas en donde se almacena la información de los sensores.

- La primera columna nombrada “mSxcm” de tipo “FLOAT” y “NULL”, es la que guarda el valor medido por el sensor de CE.
- La segunda columna nombrada “fecha_CE” de tipo “TIMESTAMP”, “CURRENT_TIMESTAMP” con indexación “PRIMARY”, será la que registre la fecha y hora del dato ingresado. Finalmente se guardan los cambios con la opción “save”.

La configuración de las columnas se muestra en la figura 2.18.

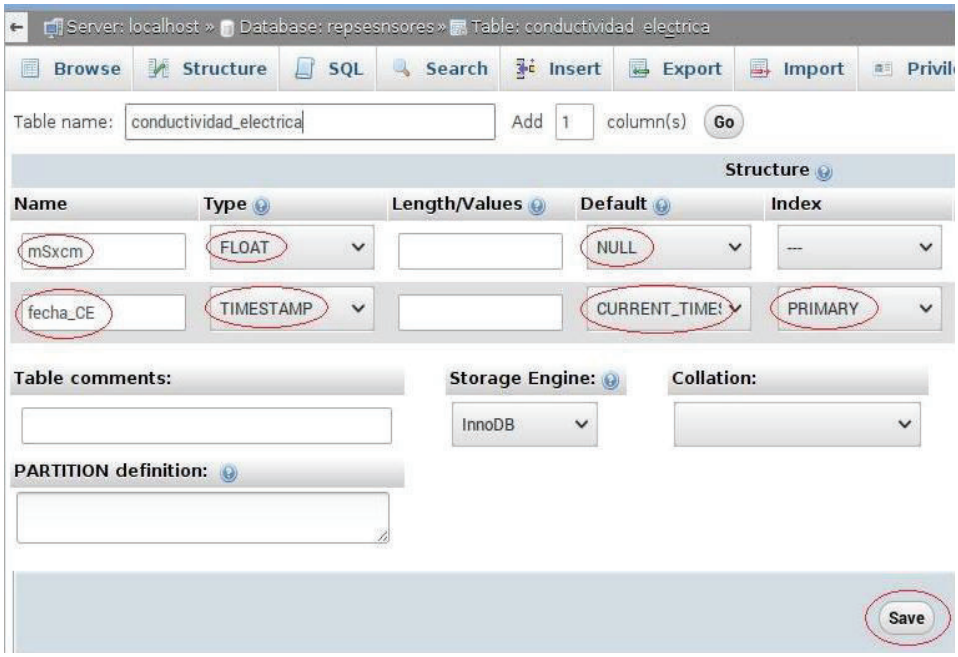


Figura 2.18 Configuración de las columnas de la tabla conductividad_electrica

De la misma manera que se creó la tabla para el sensor de conductividad eléctrica, se crean las tablas para cada uno de los parámetros de los sensores, con la diferencia en los nombres de las tablas y sus respectivas columnas tal como se muestra en la tabla 2.1.

Base de datos: repsensores		
Nombre Tabla	**Nombre Columna 1**	**Nombre Columna 2**
conductividad_electrica	mSxcm	fecha_CE

humedad_ambiental	RH	fecha_hum
iluminacion	lx	fecha_lx
nivel_pH	pH	fecha_pH
temperatura_agua	gradC	fecha_tagua
temperatura_ambiental	gradoC	fecha_tamb

Tabla 2.1 Nombres de tablas y columnas de la base de datos repsensores

En la figura 2.19, se muestra la base de datos “repsensores” con sus respectivas tablas.

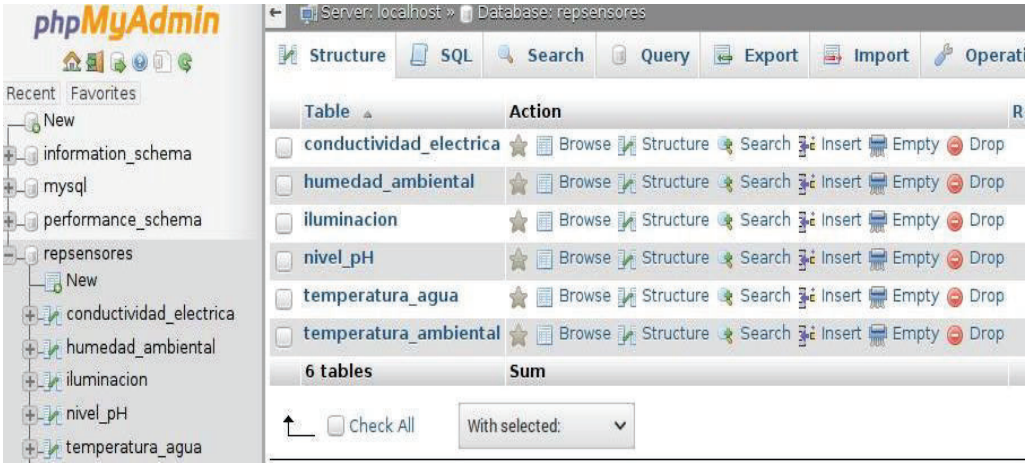


Figura 2.19 Base de datos repsensores y sus respectivas tablas

2.6 CONEXIÓN ENTRE ARDUINO Y LA BASE DE DATOS [37] [38]

Para enviar información desde el cliente (Arduino) a la base de datos que se encuentra en el servidor (Raspberry Pi), se necesita de un lenguaje de programación capaz de unir los dos entornos, para ello se utiliza el lenguaje PHP que es entendido tanto por Arduino y por la base de datos MySQL.

2.6.1 DIAGRAMA DE FLUJO PARA EL ENVIO DE INFORMACIÓN A LA BASE DE DATOS

El diagrama de flujo indicado en la figura 2.20 muestra el proceso para la obtención y envió de información hacia la base datos.

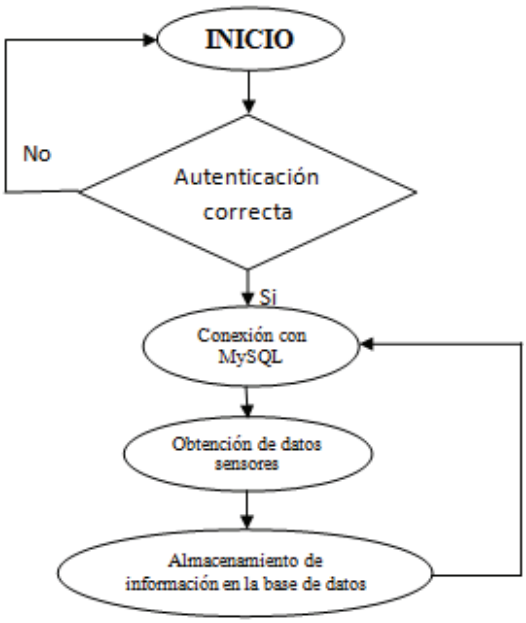


Figura 2.20 Diagrama de flujo para la obtención y envío de información a la base de datos

2.6.2 PROGRAMACIÓN DE SCRIPTS EN EL SERVIDOR

Se crea una carpeta llamada “reportsensores” en la dirección */var/www/html* de la Raspberry Pi, en la cual están guardados todos los archivos en lenguaje PHP que permiten la conexión y envió de información entre la base de datos y la plataforma Arduino.

2.6.2.1 Script de conexión con la base de datos

Se crea el script “conexion.php” que contiene información necesaria para la conexión con la base de datos, tal como se muestra el código 2.6.

```
<?php
  // Autenticación
  $equipobd = "localhost";
  $usuariobd = "ardupi";
  $clavebd = "inopi";
  $nombrebd = "repsensores";
  // Conexión con mysql
  $conex =
mysqli_connect($equipobd,$usuariobd,$clavebd,$nombrebd);
?>
```

Código 2.6 Script conexion.php

2.6.2.2 Script de envío de información a la base de datos

Se crea el script “repce.php”, que permite enviar la información recibida desde el sensor de conductividad eléctrica a la base de datos, tal como se muestra en el código 2.7.

```
<?php

    // Se hace el llamado al script conexion.php
    require("conexion.php");

    // Se lee el dato de Conductividad Eléctrica mediante GET
    $datoce = mysqli_real_escape_string($conex, $_GET['mSxcm']);

    // Se guarda el valor para insertar en la base de datos
    $insertarce = "INSERT INTO conductividad_electrica(mSxcm)
VALUES('".$datoce."')";

    // Se inserta el valor en la base de datos
    mysqli_query($conex, $insertarce);
    mysqli_close($conex);
?>
```

Código 2.7 Script repce.php

Los scripts que permitan registrar los datos de los demás sensores serán creados de manera similar al código 2.7 con sus respectivos nombres y variables, teniendo en cuenta que todos harán el llamado al script “conexion.php”.

La tabla 2.2 muestra el nombre de los scripts PHP para el registro de datos de cada uno de los sensores.

Scripts para el envío de datos creados en la carpeta reportsensores		
Nombre del script	**Variable de lectura**	**Variable de escritura**
repce.php	$datoce	$insertarce
rephumamb.php	$datohuma	$insertarhuma
repilum.php	$datoilum	$insertarilum
reph.php	$datoph	$insertarph
reptempagua.php	$datotagua	$insertartagua
reptempamb.php	$datotamb	$insertartamb

Tabla 2.2 Nombres de scripts y variables para el envío de datos

En la figura 2.21 se muestran los scripts creados en la carpeta reportsensores.

```
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano conexion.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano repce.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano reph.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano rephumamb.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano repilum.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano reptempagua.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano reptempamb.php
pi@raspberrypi:/var/www/html/reportsensores $ ls
conexion.php  reph.php       repilum.php      reptempamb.php
repce.php     rephumamb.php  reptempagua.php
```

Figura 2.21 Scripts creados en la carpeta reportsensores

2.6.2.3 Configuración de la dirección IP estática en el servidor

Para evitar cambios de direccionamiento lógico dentro de la red, se configura una IP estática en la Raspberry Pi que depende de la red a la que se encuentre conectada. Para el presente Proyecto, el servidor se encuentra en la red 172.31.36.0 con máscara 255.255.255.0, default gateway 172.31.36.1 y DNS 8.8.8.8.

Para realizar esta configuración se edita el archivo “dhcpcd.conf” ubicado en la dirección */etc* añadiendo al inicio las siguientes líneas:

```
interface eth0
static ip_address=172.31.36.37/24
static routers=172.31.36.1
static domain_name_servers=8.8.8.8
```

2.6.3 PROGRAMACIÓN DEL CLIENTE PARA EL ENVÍO DE DATOS

Para el envió de la información de los sensores desde el Arduino a la base de datos se necesita la placa Ethernet Shield.

2.6.3.1 Configuración de la Ethernet Shield

La placa Ethernet Shield utiliza las librerías *Ethernet.h* y *SPI.h* para su funcionamiento, debe ser configurada al inicio del código con la dirección MAC e IP del Arduino, la dirección IP del servidor y el modo en que trabaja la placa. Además en la sección *void setup ()*, se debe incluir la línea para inicializa la Ethernet Shield. En el código 2.8 se muestra la configuración de la Ethernet Shield.

```
//Se cargan las librerías necesarias para la Ethernet Shield
#include <Ethernet.h>
#include <SPI.h>
```

```
//Configuración de la Ethernet Shield
byte MAC[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x12}; //Dirección MAC
byte IP_CLIENTE[] = {172,31,36,34}; //Dirección IP del Arduino
byte IP_SERVIDOR[] = {172,31,36,37}; //Dirección IP del servidor
EthernetClient client;
void setup()
{
  Ethernet.begin(MAC, IP_CLIENTE);
}
```

Código 2.8 Configuración de la ethernet shield

2.6.3.2 Envío de información desde Arduino a la base de datos

Para el ingreso de información en la base de datos desde Arduino se incluye al final del *void loop()* las líneas de código que permiten establecer la conexión entre el cliente y el servidor mediante el puerto 80, mediante la instrucción GET se envía el dato por medio del script .php a la tabla correspondiente de la base de datos. El código 2.9 muestra las instrucciones necesarias para el envío de información del sensor de conductividad eléctrica desde Arduino a la base de datos.

```
 //Envió del valor sensado a la base de datos
  if (client.connect(IP_SERVIDOR, 2801)>0)
{
  client.print("GET /reportsensores/repce.php?mSxcm=");
  client.print(CE);
  client.println();
  Serial.println("Dato enviado CE");
}
else
{
  Serial.println("Sin conexion");
}
client.stop();
client.flush();
delay(2000);
```

Código 2.9 Envío de información desde Arduino a la base de datos para el sensor de CE

Para el envío de información de los demás sensores, se utilizan las mismas instrucciones del código 2.9, teniendo en cuenta el nombre del script .php, la tabla correspondiente en la base de datos y la variable a la que hace referencia.

Se unen todos los códigos de cada uno de los sensores en un solo programa en Arduino llamado “envio_sensores_arduino.ino” (ANEXO B), el cual permite obtener los datos de cada uno de los sensores y el envío de información hacia la base de datos.

2.7 GENERACIÓN DE HISTORIAL Y REPORTES [37] [39]

Toda la información de cada uno de los sensores se puede visualizar mediante un historial que indica el valor del dato ingresado con su respectiva fecha y hora. Los valores sensados que se encuentran fuera de rango serán registrados mediante reportes. En la carpeta “reportsensores”, se crean los archivos para la generación de historial y reportes.

2.7.1 HISTORIAL

Se crea el script “histce.php” que permite mostrar toda la información que se encuentra guardada en la tabla “conductividad_electrica” de la base de datos “repsensores”.

El código 2.10 muestra el contenido del script “histce.php”.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $consulta = "SELECT * FROM conductividad_electrica order by
fecha_CE desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
    echo '<b> HISTORIAL</b><br>';
    echo '<b>CONDUCTIVIDAD ELECTRICA</b>';
    echo '<tr>';
    echo '<th>mSxcm</th>';
    echo '<th>Fecha_CE</th>';
    echo '</tr>';
    while($reg=mysql_fetch_array($valor)){
        echo '<tr>';
        echo '<td>'.$reg['mSxcm'].'</td>';
        echo '<td>'.$reg['fecha_CE'].'</td>';
        echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código 2.10 Script histce.php

De manera similar se crean scripts para los demás sensores.

La figura 2.22 muestra los archivos creados para obtener el historial de cada uno de los sensores.

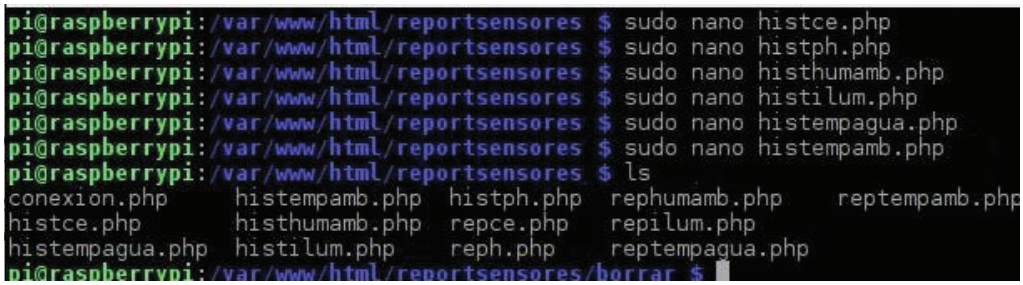


Figura 2.22 Archivos creados para obtener el historial de los sensores

2.7.2 REPORTES

Se crea el script "reportce.php" que permite mostrar la información correspondiente a valores fuera de rango que se encuentra en la tabla "conductividad_electrica" de la base de datos "repsensores". El código 2.11 muestra el contenido del script "reportce.php".

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $consulta = "SELECT * FROM conductividad_electrica WHERE mSxcm
>= 3 order by fecha_CE desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> REPORTE</b><br>';
       echo '<b>CONDUCTIVDAD ELECTRICA</b>';
       echo '<tr>';
       echo '<th>mSxcm</th>';
       echo '<th>Fecha_CE</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['mSxcm'].'</td>';
       echo '<td>'.$reg['fecha_CE'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código 2.11 Script reportce.php

De igual forma se crean scripts para los demás sensores. La figura 2.23 muestra los archivos creados para obtener el reporte de cada uno de los sensores.

```
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano reportce.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano reportph.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano reporthumamb.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano reportilum.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano reportempagua.php
pi@raspberrypi:/var/www/html/reportsensores $ sudo nano reportempamb.php
pi@raspberrypi:/var/www/html/reportsensores $ ls
conexion.php     histhumamb.php  reph.php       reportempagua.php  reportph.php
histce.php       histilum.php    rephumamb.php  reportempamb.php   reptempagua.php
histempagua.php  histph.php      repilum.php    reporthumamb.php   reptempamb.php
histempamb.php   repce.php       reportce.php   reportilum.php
pi@raspberrypi:/var/www/html/reportsensores $
```

Figura 2.23 Archivos creados para obtener el reporte de los sensores

2.8 DISEÑO DE LA APLICACIÓN PARA ANDROID [31] [40]

La aplicación para Android se diseña en el entorno web App Inventor en diferentes etapas, con el propósito de que el usuario pueda autenticarse mediante un usuario y contraseña, visualizar la información obtenida por los sensores y generar alarmas si existen valores fuera de rango.

Para utilizar App Inventor es necesario registrarse con una cuenta de Google, en la pantalla principal se crea un nuevo Proyecto, se le asigna un nombre y se desplegará una nueva ventana la cual proporciona todas las herramientas necesarias para programar la aplicación.

La figura 2.24 muestra la pantalla de inicio para programar en App Inventor.

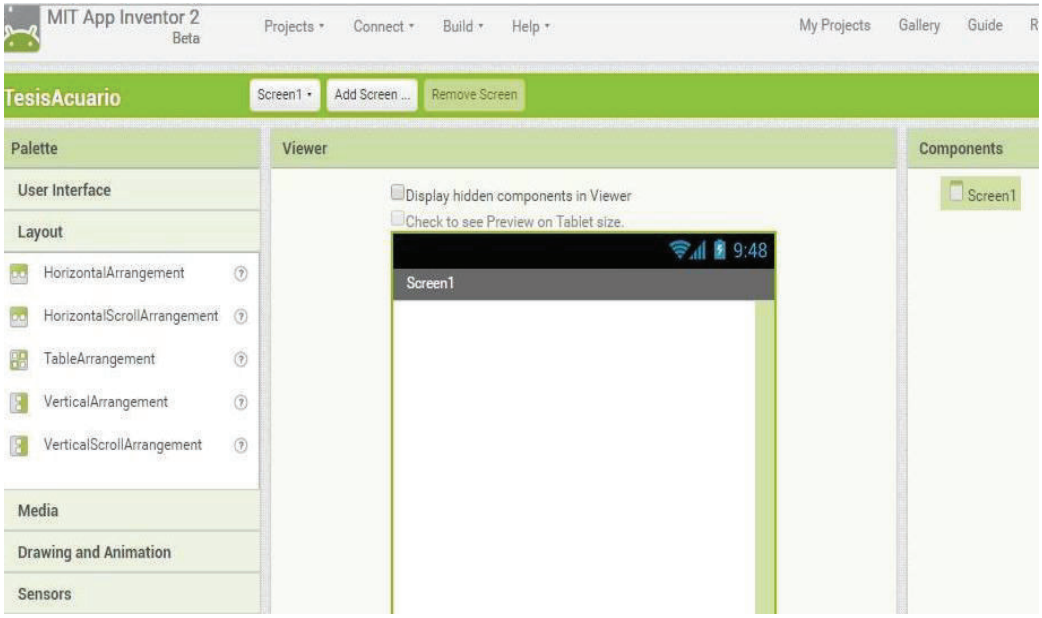


Figura 2.24 Pantalla de inicio para programar en App Inventor

Con las herramientas que proporciona la opción “Palette” y “Properties”, se crea la pantalla de presentación inicial de la aplicación, la cual permitirá que el usuario pueda dirigirse a la pantalla de autenticación al pulsar el botón “LOGIN”, tal como se muestra en la figura 2.25.

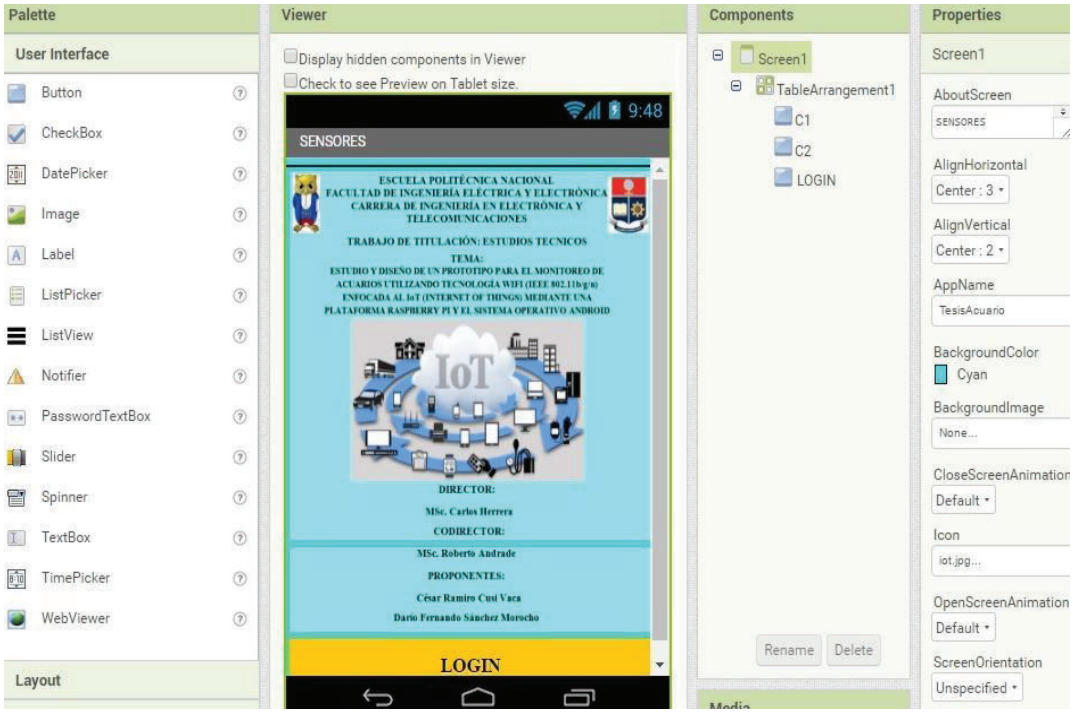


Figura 2.25 Pantalla de presentación inicial de la aplicación

Para que los botones ejecuten la acción determinada, deben ser programados mediante el editor de bloques.

La figura 2.26 muestra la programación del botón “LOGIN”.

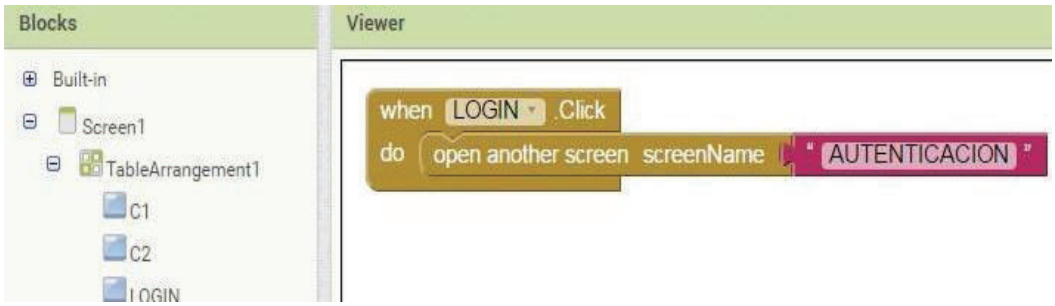


Figura 2.26 Programación del botón LOGIN

2.8.1 ETAPA DE AUTENTICACIÓN

Se crea la pantalla “AUTENTICACION”, en la cual se debe ingresar un nombre de usuario y contraseña para el acceder a la pantalla que muestra la información de los sensores, además dispone de las opciones para regresar a la pantalla de inicio o salir de la aplicación.

La figura 2.27 muestra la pantalla de autenticación para el usuario.

Figura 2.27 Pantalla de autenticación para el usuario

Una vez diseñada la pantalla de autenticación se programan cada uno de sus botones.

La figura 2.28 muestra la programación en bloques de la pantalla AUTENTICACIÓN.

Figura 2.28 Programación en bloques de la pantalla AUTENTICACIÓN

Si se digita un nombre de usuario o contraseña incorrecto por tres ocasiones, la aplicación mostrará un mensaje de acceso denegado con las opciones de poder salir de la aplicación o regresar a la pantalla de inicio, tal como se muestra en la figura 2.29.

Figura 2.29 Mensaje de acceso denegado

2.8.2 ETAPA ZONAS DE SENSADO

La pantalla “ZONAS” de la aplicación se desarrolla en base a 10 posibles zonas, en la que cada una de ellas representa un acuario, que pueden o no estar en lugares diferentes. Para el ingreso a esta pantalla el usuario debe autenticarse como “ROOT” y con la contraseña “SENSORES”, con lo cual tiene el acceso a la información de todas las zonas.

Para acceder a una determinada zona el usuario deberá autenticarse como “ZONA#” y con la contraseña “SENSORES#” en donde el “#” representa el numero de la zona que se desea visualizar. La programación en bloques de la etapa de autenticación del usuario ROOT y las 9 zonas restantes se realiza de la misma manera que se muestra en la figura 2.28. La figura 2.30 muestra el diseño de la pantalla ZONAS.

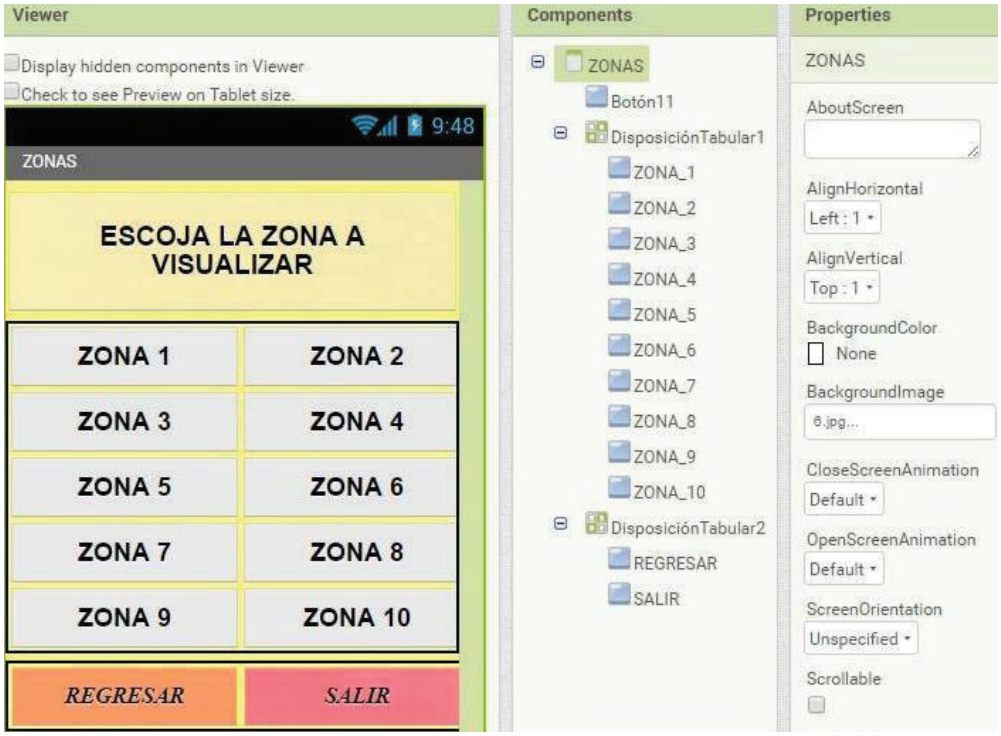


Figura 2.30 Diseño de la pantalla ZONAS

La programación en bloques de la pantalla “ZONAS” permite visualizar los datos obtenidos por los sensores según la zona que se escoja. El prototipo dispone solamente de un cliente y un servidor, por lo cual todas las zonas están redireccionadas a la información de la zona1.

La figura 2.31 muestra la programación en bloques de la pantalla ZONAS.

Figura 2.31 Programación en bloques de la pantalla ZONAS

En caso de contar con otros acuarios se debe realizar la programación en la pantalla "ZONAS", para redireccionar a la información de la zona correspondiente.

En el presente Proyecto, el diseño de la aplicación se desarrolla con referencia a un acuario, por lo cual todas las etapas están desarrolladas en base a la ZONA1.

2.8.3 ETAPA DE VISUALIZACIÓN DE DATOS

Esta etapa permitirá visualizar los valores obtenidos por los sensores, las alarmas generadas y la opción para revisar el historial y reportes.

2.8.3.1 Scripts para obtener el dato actual

Se crea el script "setrepce.php" que permite obtener el último dato del sensor de CE que se encuentra guardado en la tabla "conductividad_electrica" de la base de datos "repsensores".

El código 2.12 muestra el script "setrepce.php".

```
<?php

    $conex = mysql_connect('localhost','ardupi','inopi');
```

```
    mysql_select_db('repsensores',$conex);
    $selecciona = "SELECT * FROM conductividad_electrica order by
fecha_CE desc";
    $valorce = mysql_query($selecciona);
    $recupera=mysql_fetch_object($valorce);
    echo $recupera->mSxcm;

?>
```

Código 2.12 Script setrepce.php

De la misma manera se crean scripts para obtener el último dato de los demás sensores, para mostrar la fecha y hora del último dato registrado se crea el script "fsetrepce.php" que toma como referencia al sensor de conductividad eléctrica.

La figura 2.32 muestra los archivos creados para obtener el último dato de cada uno de los sensores.

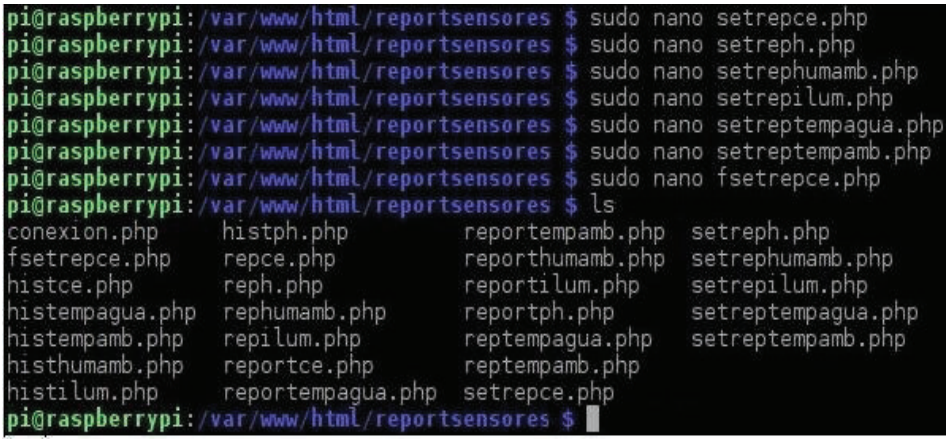


Figura 2.32 Archivos creados para obtener el último dato de los sensores

2.8.3.2 Pantalla de datos obtenidos

Se crea la pantalla "TSZONA1", la cual indica los valores obtenidos por los sensores, las alarmas visuales y auditivas que serán generadas si un parámetro medido en el acuario se encuentra fuera del nivel permitido.

La figura 2.33 muestra el diseño de la pantalla TSZONA1.

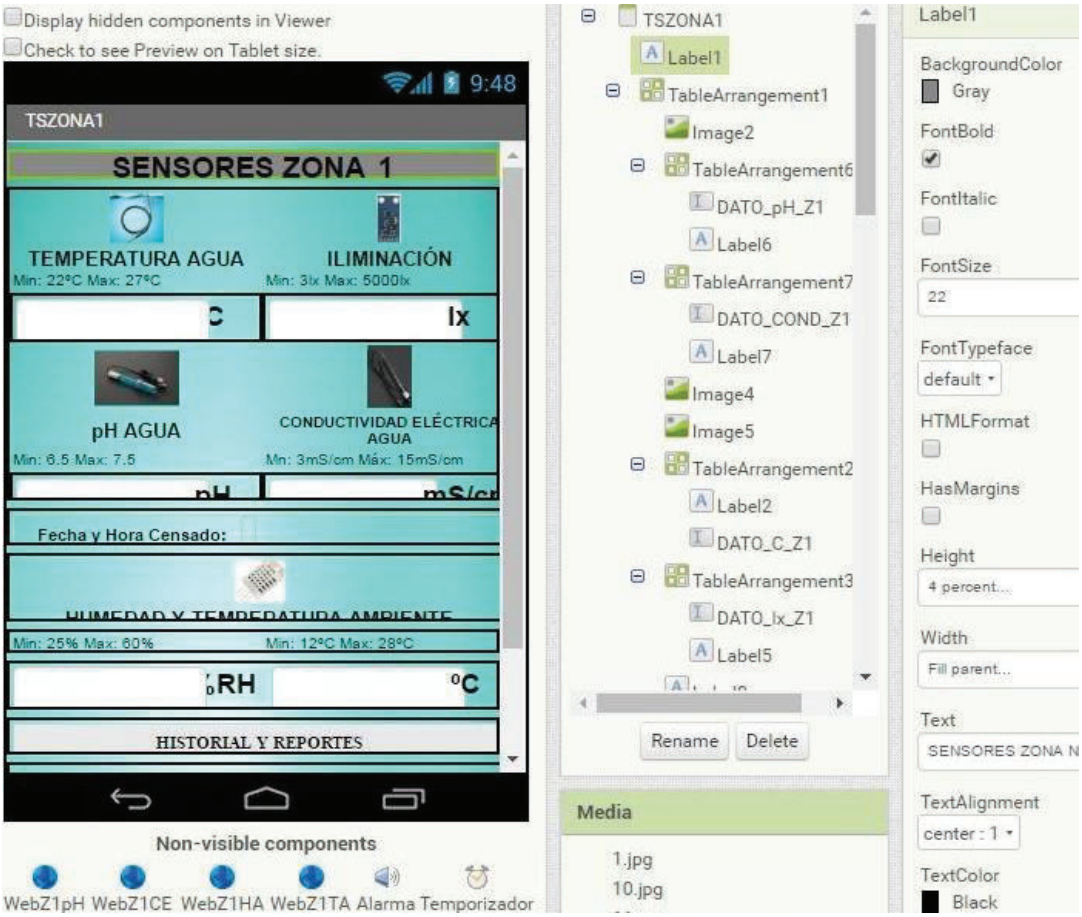


Figura 2.33 Pantalla TSZONA1

La programación en bloques para el sensor de conductividad eléctrica de la pantalla “TSZONA1” contiene:

- La variable “global VAR”, permite capturar el dato del sensor de conductividad eléctrica para mostrarlo en pantalla mediante la dirección http://172.31.36.37:2801/reportsensores/setrepce.php y el procedimiento “LLAMAR_DATO”.
- La variable “global F_H”, permite capturar la información de la fecha y hora del dato sensado para mostrarlo en la pantalla mediante la dirección http://172.31.36.37:2801/reportsensores/fsetrepce.php.
- El bloque “Temporizador”, actualiza la información cada 10 segundos.
- El bloque de “Alarma”, activará el mensaje y sonido de alarma en caso de que el dato sensado sea mayor a 3mS/cm.

- El bloque “HISTORIAL_REPORTES”, abre una nueva pantalla en la cual se encontrarán el historial y los reportes de los datos sensados.

La figura 2.34 muestra la programación en bloques de la pantalla TSZONA1 para el sensor de CE.

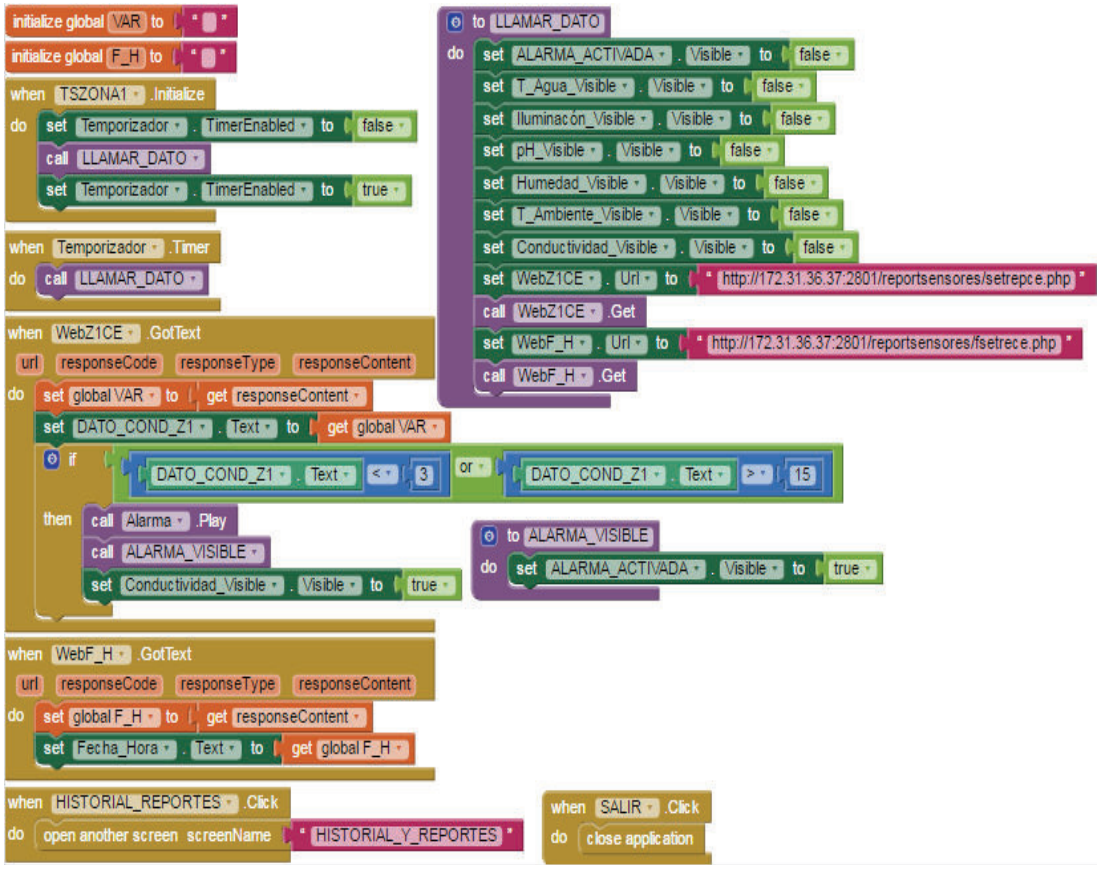


Figura 2.34 Programación en bloques de la pantalla TSZONA1 para el sensor de CE

La programación en bloques de los demás sensores en la pantalla TSZONA1, se realiza de manera similar al sensor conductividad eléctrica.

2.8.3.3 Pantalla de Historial y Reportes

Se crea la pantalla “HISTORIAL_Y_REPORTES”, la cual muestra un menú que contiene el historial y reportes de cada uno de los sensores.

La figura 2.35 muestra el diseño de la pantalla HISTORIAL_Y_REPORTES.

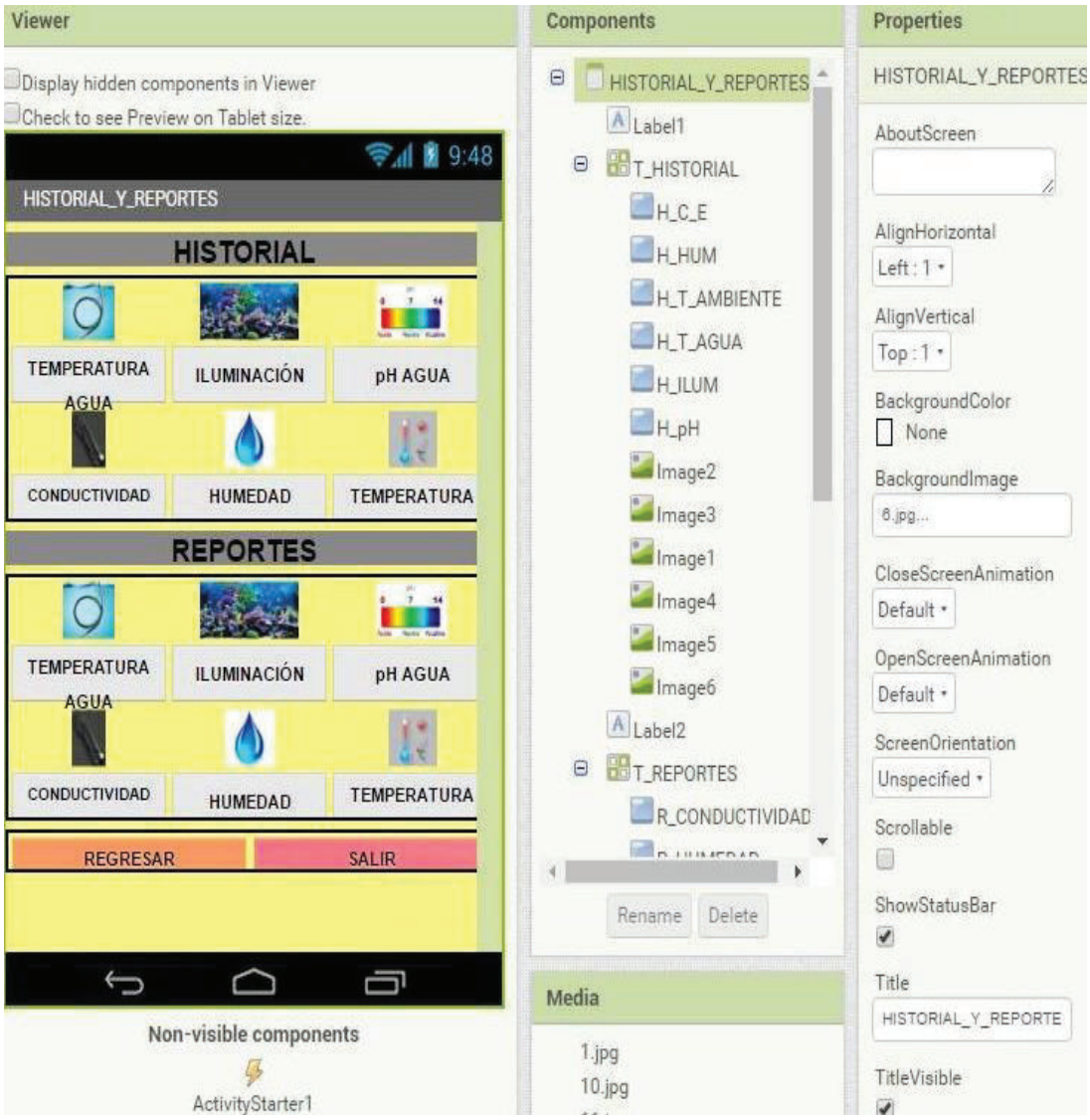


Figura 2.35 Diseño de la pantalla HISTORIAL_Y_REPORTES

La programación en bloques para el historial y reporte del sensor de CE contiene:

- El bloque “H_C_E”, permite mostrar los datos del historial en la dirección web http://172.31.36.37:2801/reportsensores/histce.php.
- El bloque “R_CONDUCTIVIDAD”, permite mostrar los datos del historial en la dirección web http://172.31.36.37: 2801/reportsensores/reportce.php.

La figura 2.36 muestra la programación en bloques de la pantalla HISTORIAL_Y_REPORTES del sensor de conductividad eléctrica.

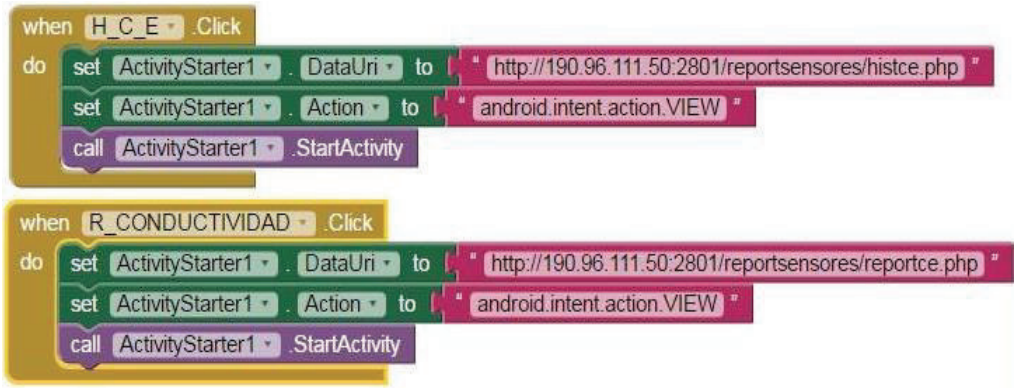


Figura 2.36 Programación en bloques de la pantalla HISTORIAL_Y_REPORTES para el sensor de CE

La programación en bloques de los demás sensores en la pantalla HISTORIAL_Y_REPORTES se realiza de manera similar al sensor conductividad eléctrica.

2.9 PROGRAMACIÓN DE LA RAPBERRY PI PARA CONTROLAR ALARMAS [41]

La programación para realizar una determinada acción cuando se detecten alarmas dentro del acuario se realiza en base al control de los puertos GPIO de la Raspberry Pi mediante el lenguaje Python.

2.9.1 INSTALACIÓN DE LA LIBRERÍA RPi.GPIO

Para poder utilizar los puertos GPIO en la Raspberry Pi con Python es necesario instalar la librería *RPi.GPIO*, de la siguiente manera:

- Descargar y descomprimir la librería GPIO para Python desde el repositorio "SourceForce" mediante los comandos: `sudo wget 'http://downloads.sourceforge.net/project/raspberry-gpio-python/RPi.GPIO-0.6.2.tar.gz'` y `sudo tar zxvf RPi.GPIO-0.6.2.tar.gz`.
- Ingresar al directorio *RPi.GPIO-0.6.2/* e instalar el paquete "python-dev" con el comando: `sudo apt-get install python-dev`.
- Finalmente instalar la librería con el comando `sudo python setup.py install`.

Una vez finalizada la instalación se procede a programar los puertos GPIO en base a las condiciones de cada sensor.

2.9.2 DIAGRAMA DE FLUJO PARA EL CONTROL DE LOS PUERTOS GPIO EN PYTHON

El diagrama de flujo de la figura 2.37 muestra el procedimiento empleado en Python para el control de los puertos GPIO de la Raspberry Pi.

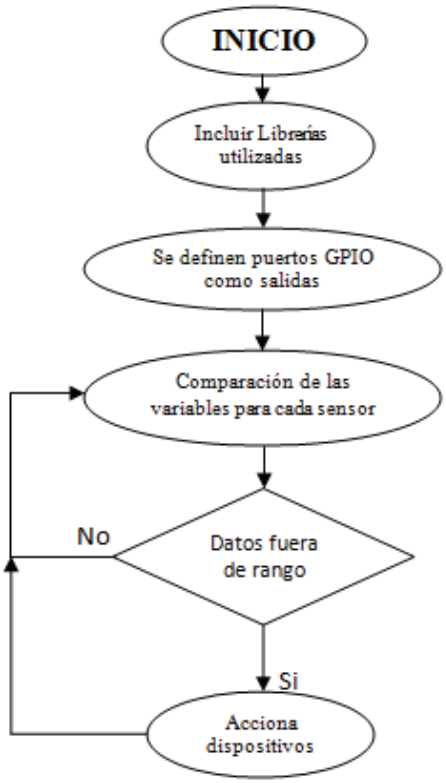


Figura 2.37 Diagrama de flujo para el control de los puertos GPIO de la Raspberry Pi

2.9.3 CONTROL DE PUERTOS GPIO CON PYTHON

Se crea un programa que permita activar o desactivar la salida de un puerto en la Raspberry Pi, el control de los puertos estará determinado por el rango de medición normal establecido para cada sensor, la comparación entre datos de los sensores que tienen cierta relación y datos obtenidos desde servidores de Internet.

El código 2.13 muestra el programa “control_gpio.py” para el control de los puertos GPIO.

```
# Programa para el control de puertos gpio de la Raspberry Pi en
Python
#!/usr/bin/python  Indica la dirección desde donde se ejecuta
Python

#Se cargan los módulos y librerías necesarias para el programa
import RPi.GPIO as GPIO
from urllib2 import urlopen
from datetime import datetime, time

GPIO.setmode(GPIO.BCM)  #Define el modo de numeración de los pines
GPIO.setwarnings(False) #desactiva las alertas en caso de que el
canal gpio este en uso

#Define los puertos GPIO como salidas
GPIO.setup(4, GPIO.OUT)    #Sensor DHT22
GPIO.setup(18, GPIO.OUT)   #Sensor BH1750
GPIO.setup(23, GPIO.OUT)   #Sensor PH
GPIO.setup(24, GPIO.OUT)   #Sensor CE
GPIO.setup(27, GPIO.OUT)   #Sensor DS18b20

#Lazo de repetición
i=1
while i<3:

    #Control para el sensor de humedad y temperatura DHT22
    dtempwww =
urlopen('http://www.accuweather.com/es/ec/quito/129846/weather-
forecast/129846')
    Tempwww = ""

    for line in dtempwww:
        if '<span class="local-temp">' in line:
            Tempwww=line.replace('<span class="local-temp">','')
            Tempwww=Tempwww.replace('&deg;C</span>','').strip()
            break

    dTempwww = float(Tempwww)
    dhtdato=
urlopen('http://172.31.36.37:2801/reportsensores/setreptempamb.php
')
    dht22= float(dhtdato.read())
    if ( dht22 >= 28 and dTempwww >= 24 ) :
        GPIO.output(4, True)
    else:
        GPIO.output(4, False)

    #Control para el sensor de luz BH1750
    datbh1750 =
urlopen('http://172.31.36.37:2801/reportsensores/setrepilum.php')
    bh1750 = float(datbh1750.read())
    horabh = datetime.now()  # Obtiene fecha y hora actual
```

```
    if ( bh1750 < 30 and  ( horabh.hour >= 18 or horabh.hour <= 6
) ) :
        GPIO.output(18, True)
    else:
        GPIO.output(18, False)

#Control para el sensor sumergible de temperatura del agua DS18b20
    dat18b20 =
urlopen('http://172.31.36.37:2801/reportsensores/setreptempagua.ph
p')
    ds18b20 = float(dat18b20.read())

    if ( ds18b20 < 22 and dht22 < 24 ) :
        GPIO.output(27, True)
    else:
        GPIO.output(27, False)

    #Control para el sensor de pH y conductividad eléctrica
    datph =
urlopen('http://172.31.36.37:2801/reportsensores/setreph.php')
    dph = float(datph.read())
    datce =
urlopen('http://172.31.36.37:2801/reportsensores/setrepce.php')
    dce = float(datce.read())

    if ( (dph < 6) or (dph > 7.5 and dce <= 3)) :
        GPIO.output(23, False)
    else:
        GPIO.output(23, True)

    if ( dce > 3 and ds18b20 <= 22 ) :
        GPIO.output(24, True)
    else:
        GPIO.output(24, False)
```

Código 2.13 Programa control_gpio.py para el control de los puertos GPIO

El código 2.13 consta de las siguientes partes:

- Al inicio se cargan los módulos y librerías necesarias para que el código se ejecute, donde la librería RPI.GPIO permite controlar los puertos GPIO, el módulo urllib2 permite interactuar con páginas web y el módulo datetime para trabajar con información de la fecha y hora.
- Se define el modo de numeración de los puertos GPIO y se asigna un determinado puerto a una alarma de cada sensor.
- Para que el programa sea repetitivo se utiliza la instrucción while.
- La función *urlopen* permite obtener el último dato almacenado por cada uno de los sensores en la Raspberry Pi o un dato desde la web para

realizar comparaciones y en base a ello ejecutar una acción correspondiente.

- Se realiza la comparación entre el dato de temperatura ambiental registrado por el sensor DHT22 y el dato del clima que proporciona la página web “AccuWeather” para la ciudad de Quito. Si el dato medido por el sensor es mayor o igual a 28ºC y el dato de la página web es mayor o igual a 24ºC se activará el puerto GPIO 4, caso contrario el puerto permanecerá desactivado.
- La comparación entre el dato registrado por el sensor de luz y el dato actual del tiempo proporcionado por el módulo *datetime* servirán para controlar el puerto GPIO 18. Si la luminosidad es menor que 30lx y la hora es mayor o igual que 18:00 o menor o igual a 6 se activará el puerto, caso contrario el puerto permanecerá desactivado.
- Para controlar el puerto GPIO 23 se realiza la comparación entre el dato registrado por el sensor de pH y el sensor de CE. El puerto solamente se activará si el dato de pH es menor a 6, o cuando el dato de pH sea mayor a 7.5 y el dato de CE sea menor o igual que 3 mS/cm.
- Si la comparación entre el dato de CE es mayor a 3 mS/cm y el dato de temperatura del agua es menor o igual que 22ºC, se activará el puerto GPIO 24, caso contrario el puerto permanecerá desactivado.
- Para la activación del puerto GPIO 27 el dato del sensor de temperatura en el agua debe ser menor que 22ºC y el dato de temperatura ambiental debe ser menor que 24ºC, caso contrario el puerto quedará desactivado.

2.10 DISEÑO DEL CIRCUITO ELÉCTRÓNICO PARA EL CONTROL DE ALARMAS

Con el propósito de integrar todos los dispositivos que intervienen en el prototipo y controlar las alarmas con aparatos eléctricos que funcionan a 120V, se realiza el diseño de una placa de hardware en base al programa ISIS Proteus.

2.10.1 DISEÑO DE LA ETAPA DE POTENCIA DEL CIRCUITO

Para controlar dispositivos eléctricos que funcionan a 120V en base a una señal de 5V que será generada en los puertos GPIO de la Raspberry Pi, se crea un

circuito que actúe en forma de un interruptor digital cuando exista un cambio de estado lógico en el puerto GPIO.

El circuito de potencia consta de los siguientes elementos:

- Un relay SRD-05VDC-SL-C que sirve como interruptor para el circuito eléctrico que funciona a 120V cuando se induce un voltaje de 5V en su bobina.
- Un transistor 2N3904 para controlar la bobina del relé en función del flujo de corriente, trabajando en la zona de corte o en la zona saturación como un interruptor digital.
- Una resistencia que limita el flujo de corriente que circula por el transistor.

La figura 2.38, indica el diagrama del circuito de potencia para un sensor.

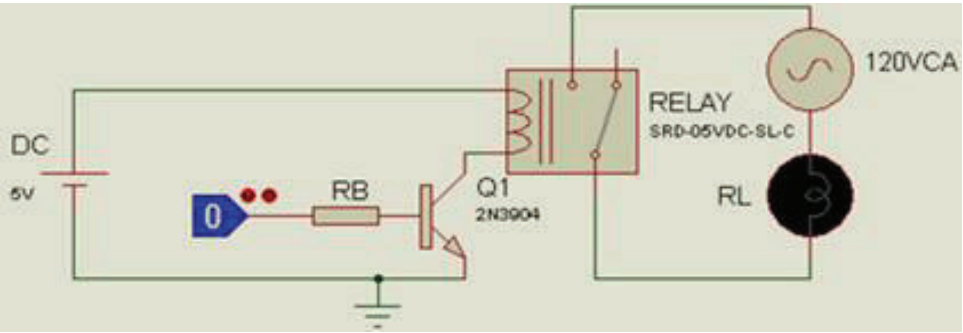


Figura 2.38 Diagrama del circuito de potencia para un sensor

La resistencia de base (R_B) se calcula en función de la corriente de colector mínima (I_C) que debe circular por la bobina para que funcioné el relé, el voltaje de entrada (V_{in}) del circuito generado por el puerto GPIO de la Raspberry Pi y el β del transistor. Algunos datos son obtenidos de los datasheets de los elementos.

A continuación se muestra el cálculo de la resistencia R_B.

Datos: $V_{in} = 5V, \beta = 100, I_{bobina_{min}} = 70mA, V_{CE} = 1V, V_E = 0$

$$R_B = \frac{V_{RB}}{I_{B_{min}}}$$

$$I_{C_{min}} = \beta I_{B_{min}},$$

$$V_{in} = V_{RB} + V_{CE} + V_E$$

$$R_B = \frac{V_{in} - V_{CE} - V_E}{\frac{I_{C_{min}}}{\beta}}$$

Donde $I_{C_{min}} = I_{bobina_{min}}$

$$R_B = \frac{5V - 1V - 0}{\frac{70mA}{100}} = 5.71K\Omega$$

Para asegurar un flujo de corriente optimo en la base del transistor, se escoge una resistencia $R_B = 4.7K\Omega$.

La figura 2.39 muestra la simulación del circuito de potencia.

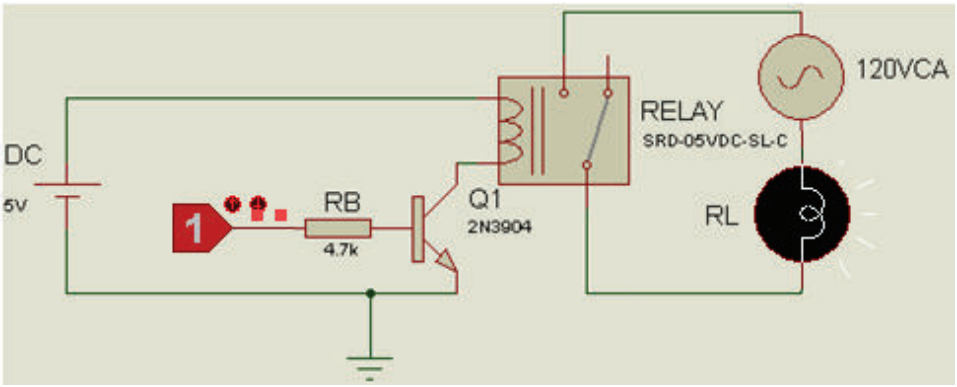


Figura 2.39 Simulación del circuito de potencia

2.10.2 DIAGRAMA DE BLOQUES TOTAL DEL CIRCUITO (PLACA)

En la figura 2.40 se muestra el diagrama de bloques total para la placa.

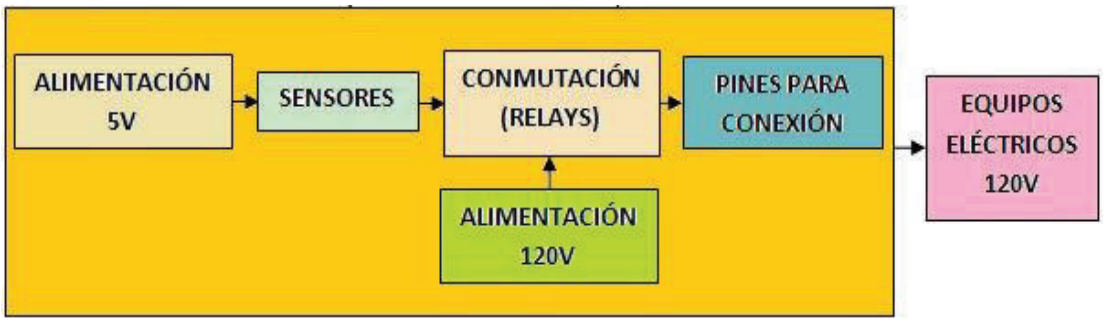


Figura 2.40 Diagrama de bloques total para la placa

2.10.3 DIAGRAMA ESQUEMÁTICO TOTAL DEL CIRCUITO (PLACA)

El diagrama total para el control de alarmas integrará circuitos de potencia para cada sensor como la indicada en la figura 2.34, terminales de polarización AC/DC y pines de entrada/salida para conectar los sensores, el Arduino y la Raspberry Pi.

La figura 2.41 muestra el diagrama esquemático total del circuito.

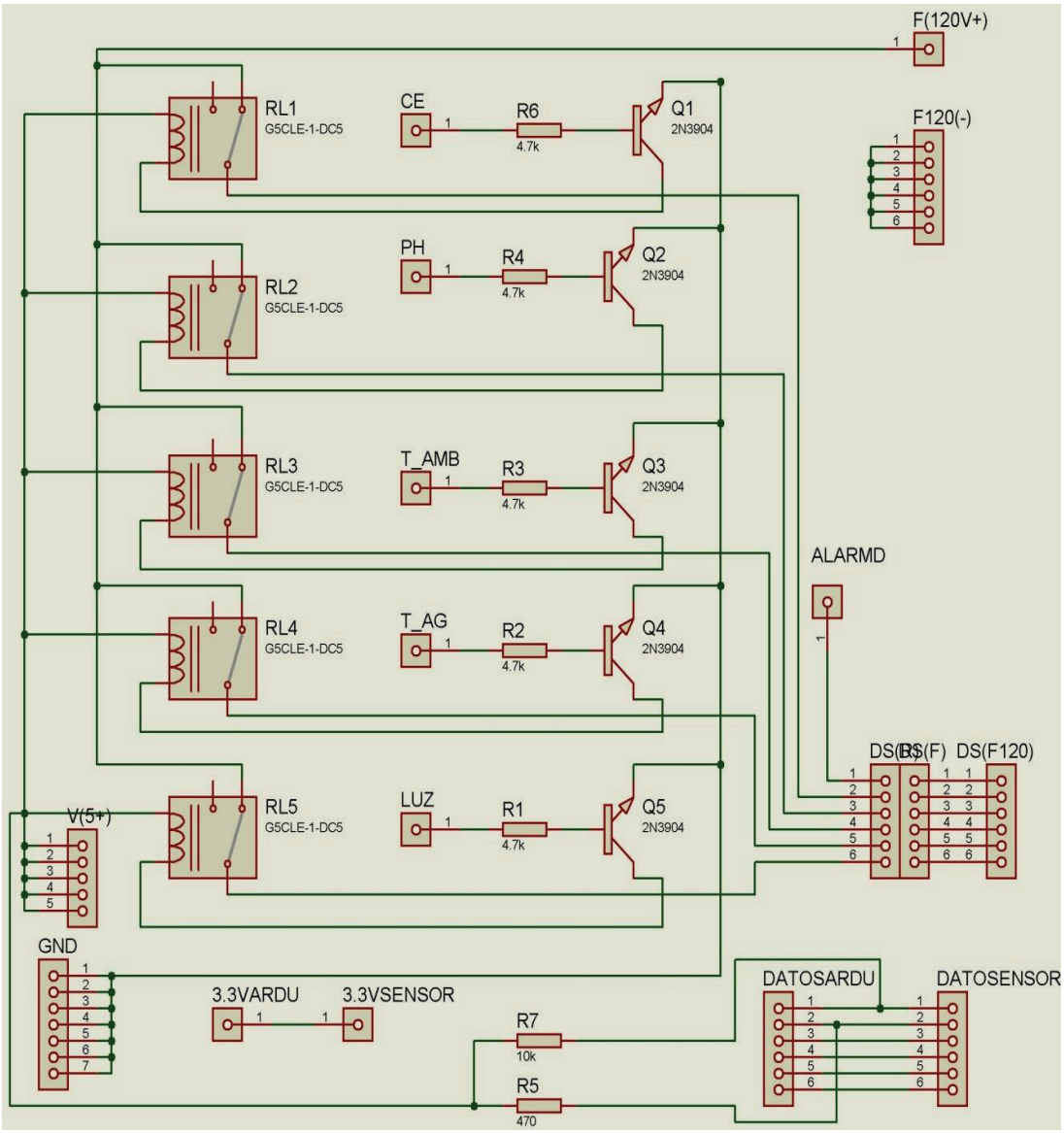


Figura 2.41 Diagrama esquemático total del circuito

La figura 2.42 muestra la simulación del circuito total en el programa ISIS Proteus.

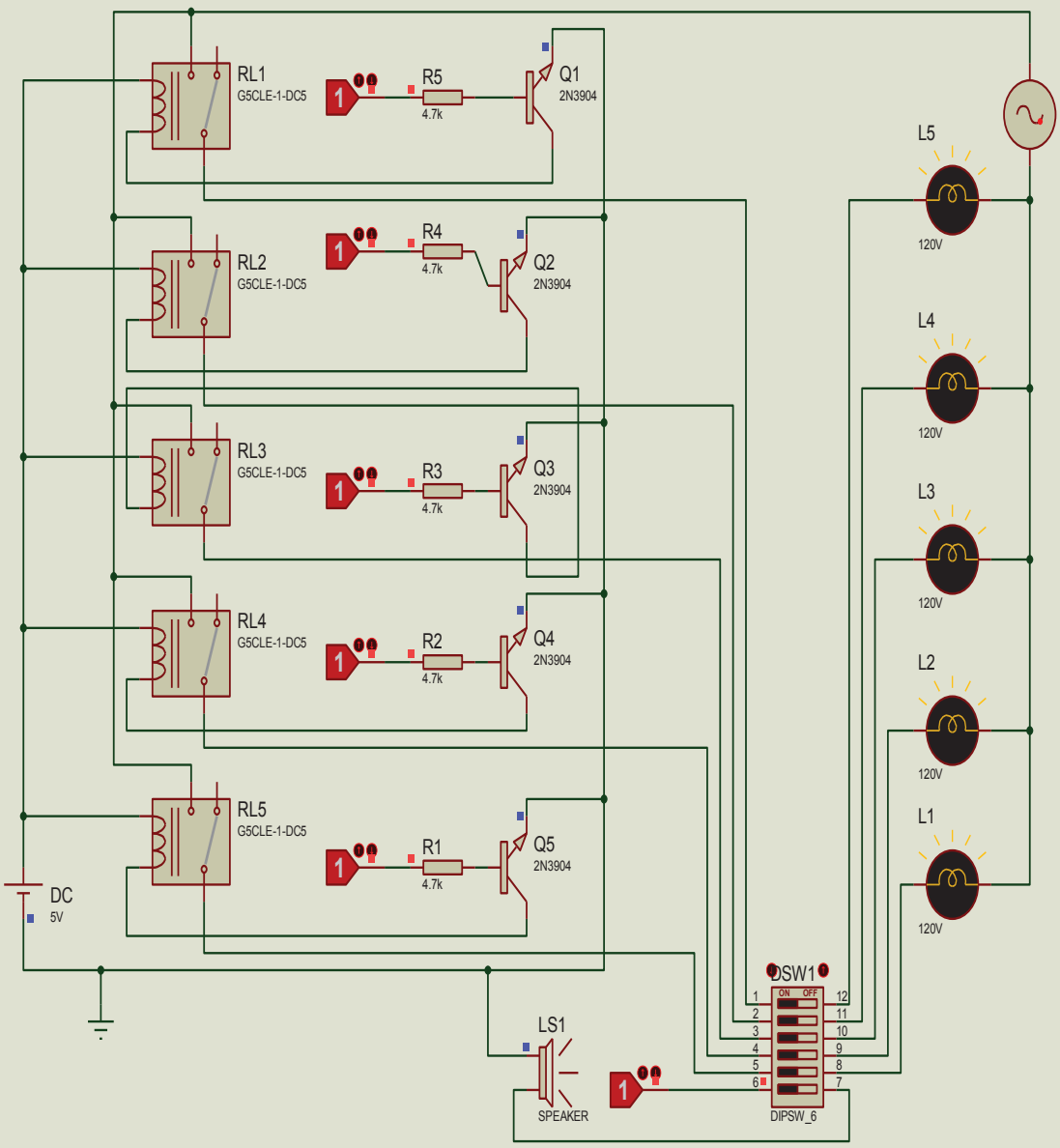


Figura 2.42 Simulación del circuito total en el programa ISIS Proteus

2.10.4 DIAGRAMA TOTAL DEL CIRCUITO IMPRESO PARA LA PLACA

Una vez realizado el diseño con la ayuda de la herramienta ARES, se importa el circuito para realizar las pistas de la placa del circuito impreso.

La figura 2.43 muestra el circuito impreso para la placa.

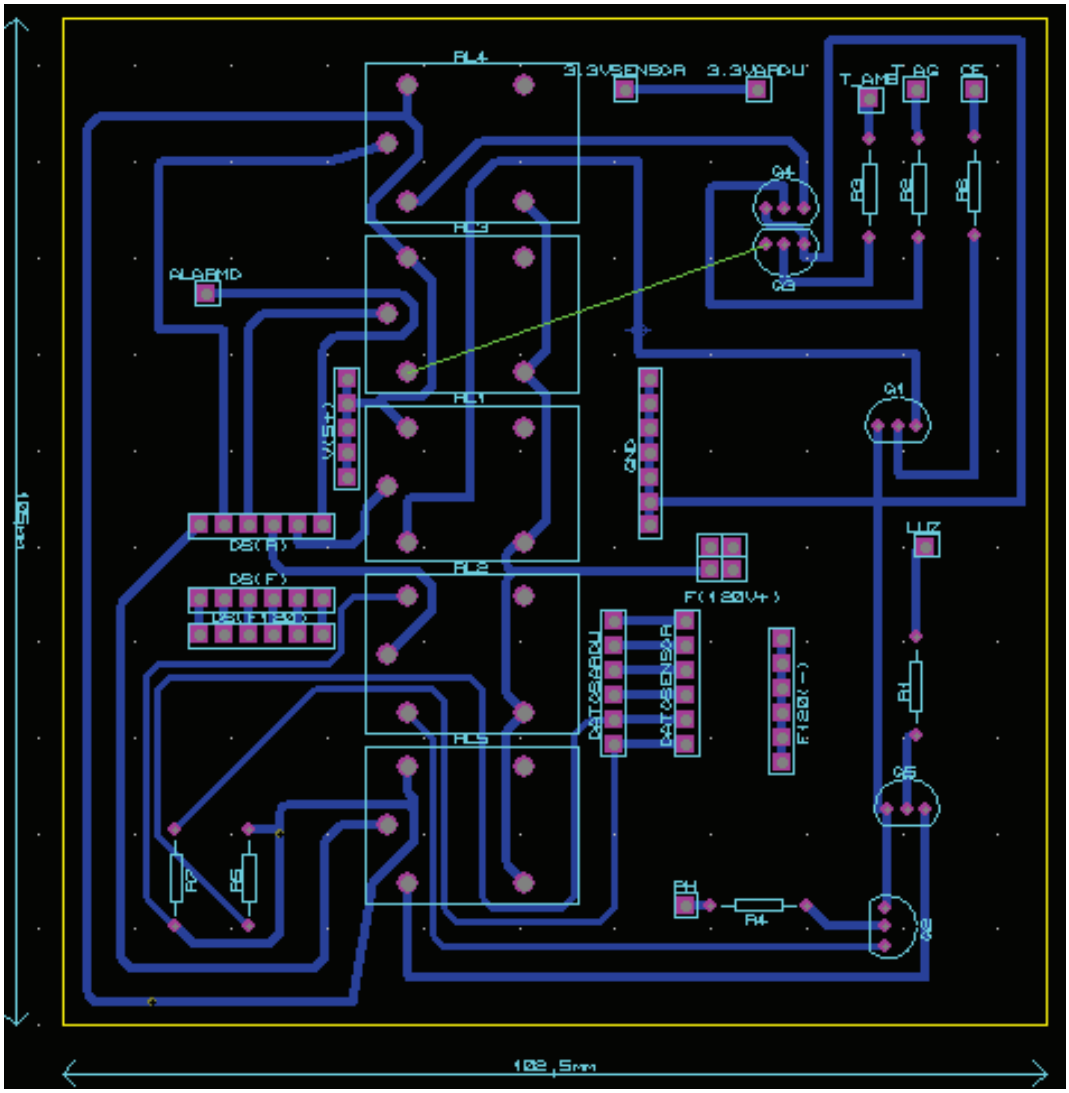


Figura 2.43 Circuito impreso para la placa

2.11 CONFIGURACIÓN DEL EQUIPO WIFI

En el modem Wi-Fi se busca la opción para configurar la red, al equipo se le asigna la primera dirección IP válida de la red privada 172.31.36.0/24. En la figura 2.44 se muestra la configuración de red en el módem.

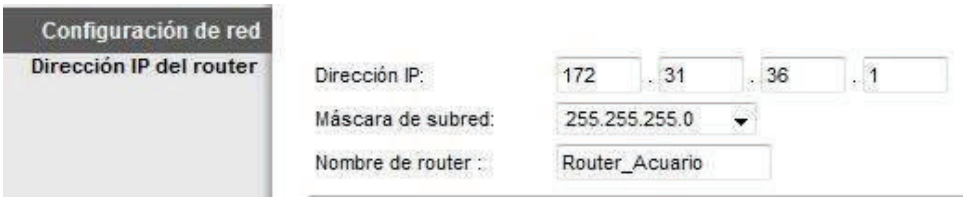


Figura 2.44 Configuración de red en el modem

Para el forwarding de puertos, se busca la opción correspondiente y se configura el puerto interno y externo 2801 en la dirección 172.31.36.37 correspondiente al servidor Apache. La figura 2.45 muestra el forwarding de puertos para el servidor Apache.

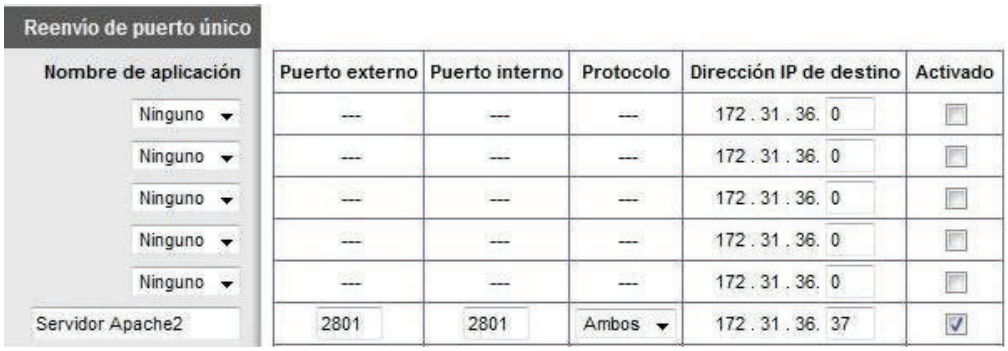


Figura 2.45 Forwarding de puertos para el servidor Apache

CAPÍTULO III

3. IMPLEMENTACIÓN DEL PROTOTIPO Y PRUEBAS

En el presente capítulo se describe la integración del hardware y software diseñado en el capítulo anterior, a fin de realizar las pruebas necesarias para validar el prototipo implementado.

3.1 IMPLEMENTACIÓN DE LA PLACA EN BAQUELITA

Para la implementación de la placa primeramente se imprime a láser en papel fotográfico el circuito total diseñado que muestra la figura 2.38, se sujeta firmemente el circuito impreso a la baquelita y se procede a realizar el planchado del circuito para obtener el diagrama impreso en la baquelita. La figura 3.1 muestra el proceso de planchado en la baquelita.

Figura 3.1 Proceso de planchado en la baquelita

Luego de realizar el planchado del circuito por varios minutos se procede a retirar el papel de la baquelita y recortar el área del circuito, tal como se muestra la figura 3.2.

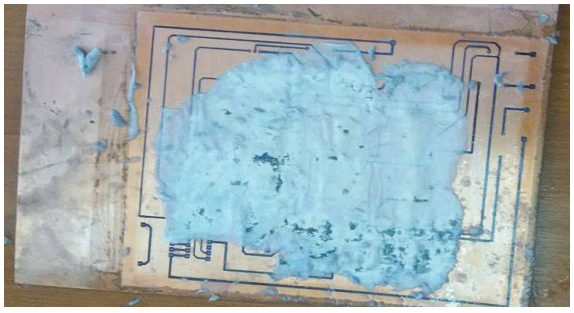


Figura 3.2 Limpieza y recorte de la baquelita

Para obtener las pistas del circuito en la baquelita se utiliza un recipiente en el cual se disuelve cloruro férrico en agua y se introduce la placa en la solución. La figura 3.3 muestra el proceso de marcación de pistas en la baquelita.

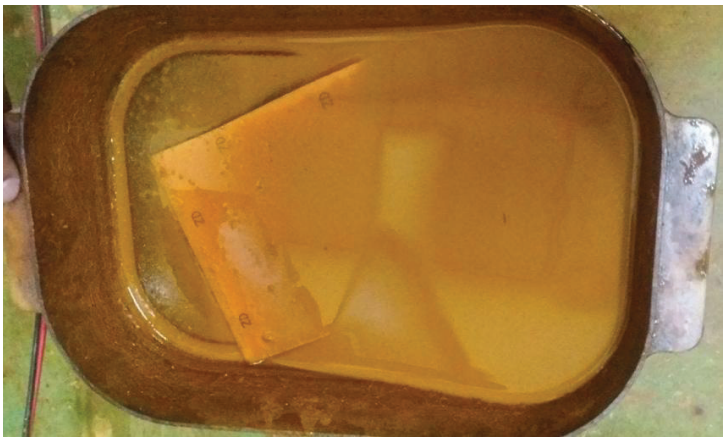


Figura 3.3 Proceso de marcación de pistas en la baquelita

Una vez marcadas las pistas en la baquelita se limpian las impurezas, se perfora la placa y se procede a soldar cada uno de los elementos. En la figura 3.4 se muestra la soldadura de los elementos en la placa.

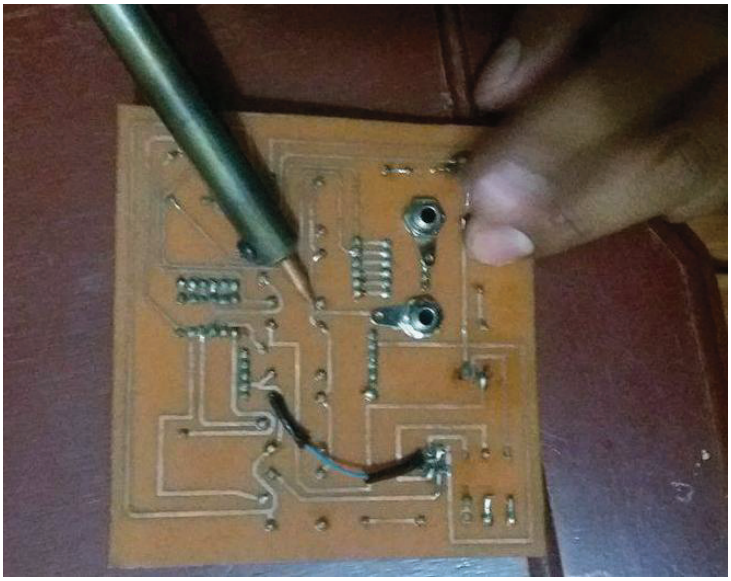


Figura 3.4 Soldadura de los elementos en la placa

Luego de soldar todos los elementos finalmente se procede a etiquetarlos para identificarlos fácilmente.

La figura 3.5 muestra la placa implementada.

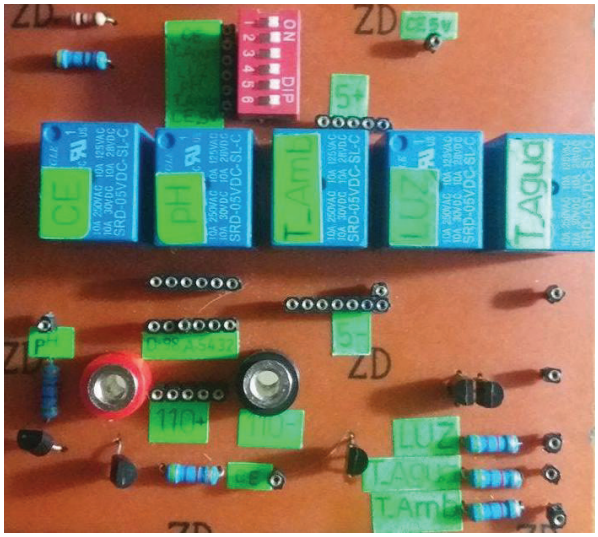


Figura 3.5 Placa implementada

3.2 IMPLEMENTACIÓN DEL CABLE DE CONEXIÓN DE LOS SENSORES

Con el fin de tener todos los sensores agrupados, se sueldan los terminales de cada sensor a un extremo de un cable UTP de 4 pares y el otro extremo del cable es utilizado para la conexión en la placa implementada. Para facilitar la conexión de cada uno de los sensores con la placa se procede a identificar los terminales del cable mediante etiquetas. La figura 3.6 muestra el cable de conexión de los sensores hacia la placa.

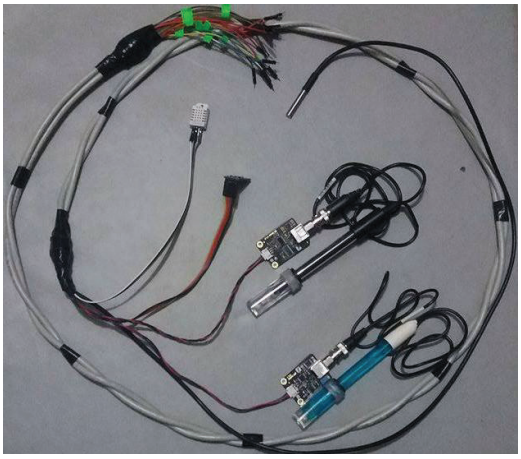


Figura 3.6 Cable de conexión de los sensores hacia la placa

3.3 IMPLEMENTACIÓN DE CABLES DE ALIMENTACIÓN

Los cables de alimentación se construyen con la finalidad de acoplar los equipos eléctricos que funcionan a 120V con la placa implementada.

La figura 3.7 muestra los cables de alimentación.

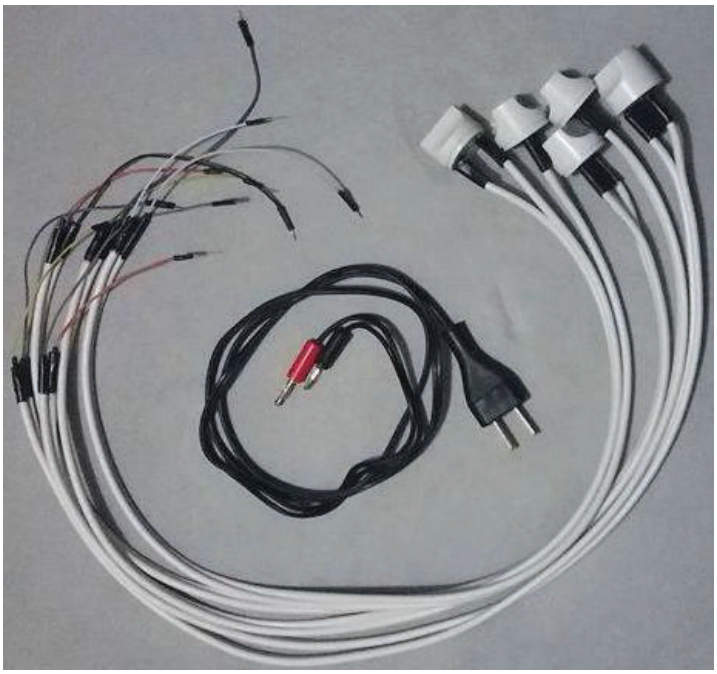


Figura 3.7 Cables de alimentación

3.4 IMPLEMENTACIÓN DEL PROTOTIPO

Para implementar el prototipo del Proyecto planteado se conectan los terminales de los sensores, los cables de alimentación, la placa Arduino y la Raspberry Pi en la placa desarrollada. Además se conectan mediante cables de red el Arduino y la Raspberry Pi al equipo Wi-Fi.

Finalmente se colocan los sensores y equipos eléctricos en el acuario para posteriormente realizar las pruebas de funcionamiento.

En la figura 3.8 se observa el prototipo implementado.

Figura 3.8 Prototipo implementado

Los equipos eléctricos se accionan mediante la detección de parámetros fuera del rango permitido, estos equipos son los siguientes:

- Un ventilador que se acciona dependiendo del estado del puerto GPIO 4.
- Una lámpara fluorescente que se acciona mediante el puerto GPIO 18.

- Una lámpara led que se acciona dependiendo del estado del puerto GPIO 23.
- Una alarma auditiva que se activa mediante el puerto GPIO 24.
- Un calentador de agua que se activa dependiendo del estado del puerto GPIO 27.

3.5 PRUEBA DE ENVÍO DE INFORMACIÓN ENTRE LOS SENSORES Y ARDUINO

Se ejecuta el programa llamado “envio_sensores_arduino.ino”, el mismo que permite observar el dato obtenido por cada sensor mediante el monitor serial de Arduino.

La figura 3.9 muestra la prueba de obtención de datos desde los sensores hacia Arduino.

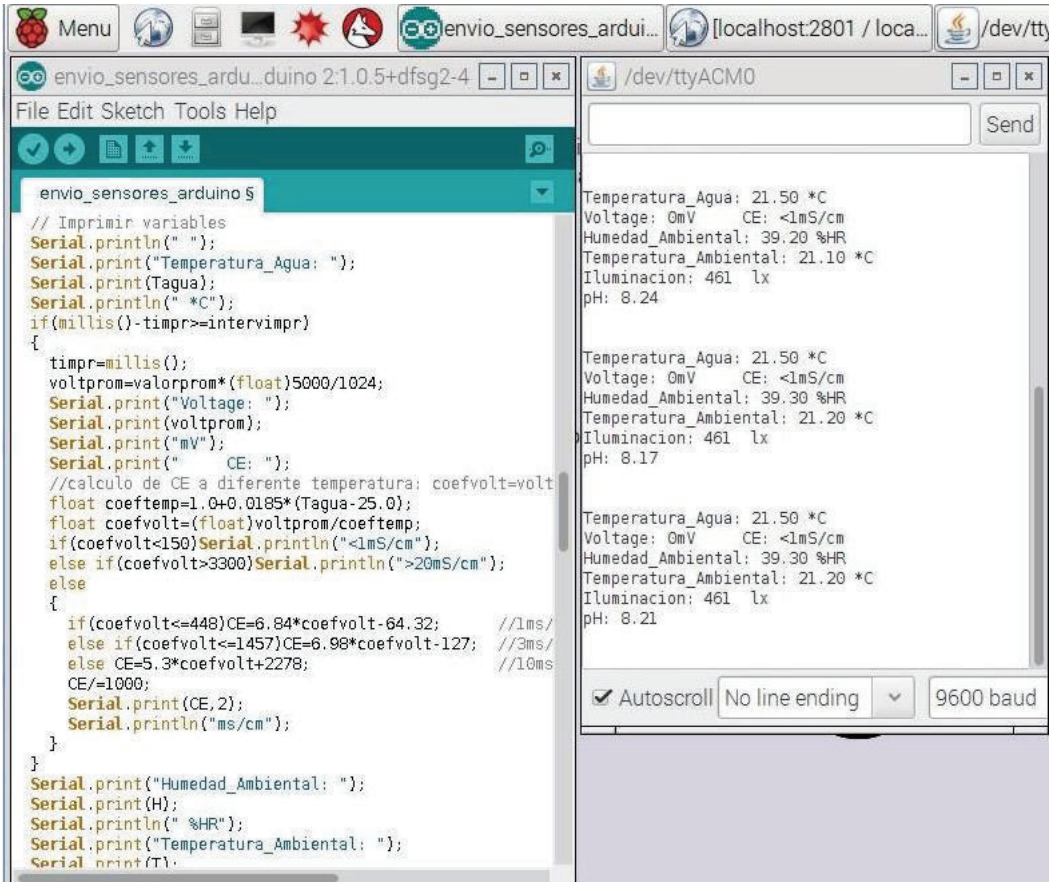


Figura 3.9 Prueba de obtención de datos desde los sensores hacia Arduino

3.6 PRUEBA DE ENVÍO DE INFORMACIÓN DESDE ARDUINO A LA BASE DE DATOS

En el archivo “envio_sensores_arduino” se agrega el código correspondiente para enviar información al servidor. Con la Raspberry Pi y el Arduino conectados a la red por medio del módem, se realiza la prueba de envío de información comprobando que los datos enviados desde Arduino lleguen a la base de datos.

La figura 3.10 muestra la prueba para el envío de información desde Arduino a la base de datos.

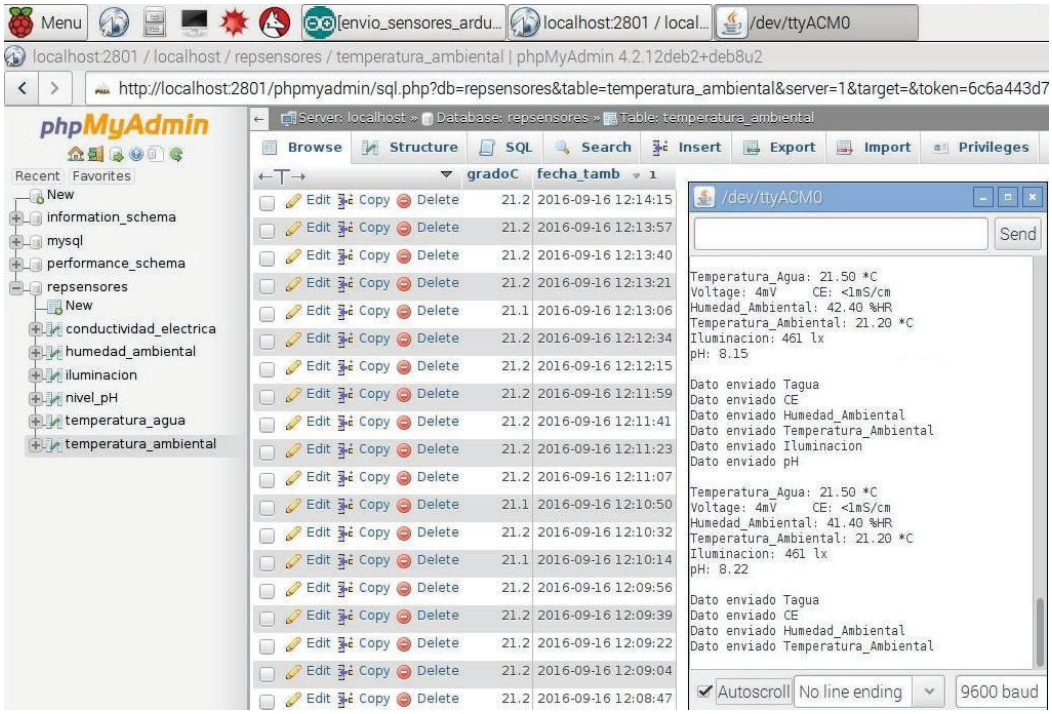


Figura 3.10 Prueba de envío de información desde Arduino a la base de datos

3.7 PRUEBA DE FUNCIONAMIENTO DE LA APLICACIÓN PARA ANDROID

Para realizar esta prueba se debe instalar la aplicación desarrollada en un dispositivo Android, luego del proceso de autenticación, la aplicación muestra la información de cada uno de los sensores.

La figura 3.11 indica los datos obtenidos en la aplicación Android en diferentes dispositivos.

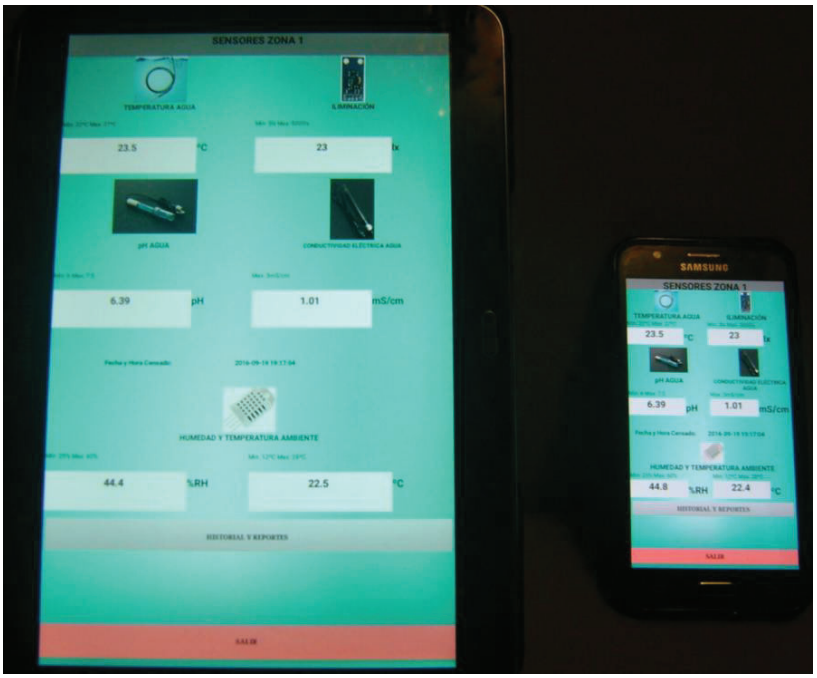


Figura 3.11 Funcionamiento de la aplicación en diferentes dispositivos Android

La figura 3.12 muestra el funcionamiento de la aplicación con todas las alarmas activadas.

Figura 3.12 Funcionamiento de la aplicación con todas las alarmas activadas

También se comprueba la obtención de datos del sensor de pH, para generar el historial y reportes, tal como se indica en la figura 3.13.

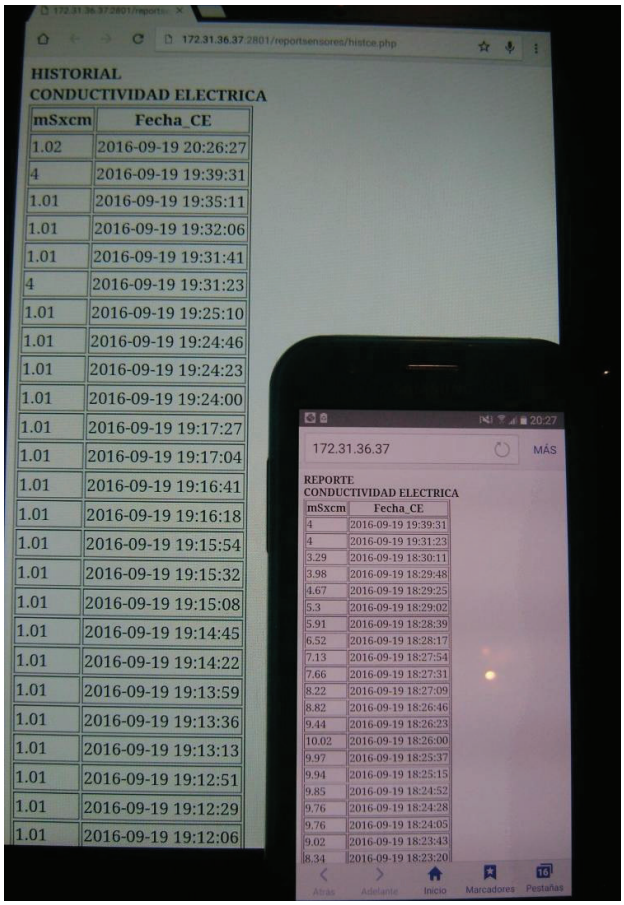


Figura 3.13 Obtención de datos para generar el historial y reportes

3.8 PRUEBA DE FUNCIONAMIENTO DE LOS EQUIPOS ELÉCTRICOS

Se comprueba el funcionamiento del programa “control_gpio.py”, para la activación de los equipos eléctricos de cada uno de los sensores cuando existen parámetros fuera de rango.

3.8.1 PRUEBA DE FUNCIONAMIENTO DEL VENTILADOR

Este dispositivo eléctrico se acciona cuando los datos de la temperatura ambiente del sensor y el obtenido desde Internet se encuentran fuera de rango.

La figura 3.14 muestra el funcionamiento del ventilador.

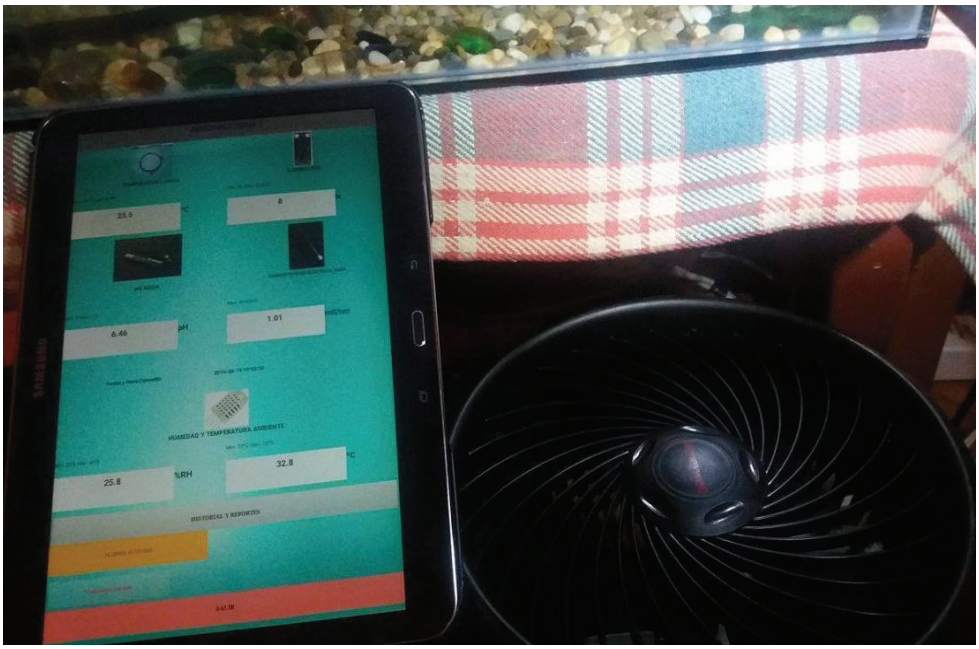


Figura 3.14 Funcionamiento del ventilador

3.8.2 PRUEBA DE FUNCIONAMIENTO DE LA LÁMPARA FLUORESCENTE

La lámpara se activa dependiendo de la hora y el nivel de luz que se detecte en el acuario.

La figura 3.15 muestra el funcionamiento de la lámpara fluorescente.

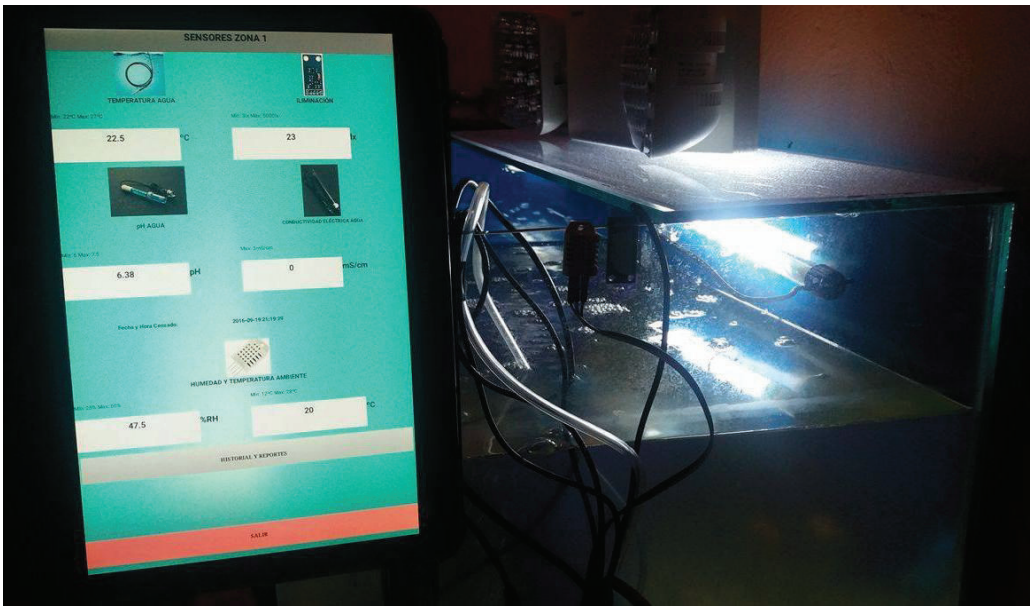


Figura 3.15 Funcionamiento de la lámpara fluorescente

3.8.3 PRUEBA DE FUNCIONAMIENTO DEL CALENTADOR DE AGUA

El calentador inicia su funcionamiento cuando detecta un nivel bajo de temperatura en el acuario y en el ambiente.

La figura 3.16 muestra el funcionamiento del calentador de agua.

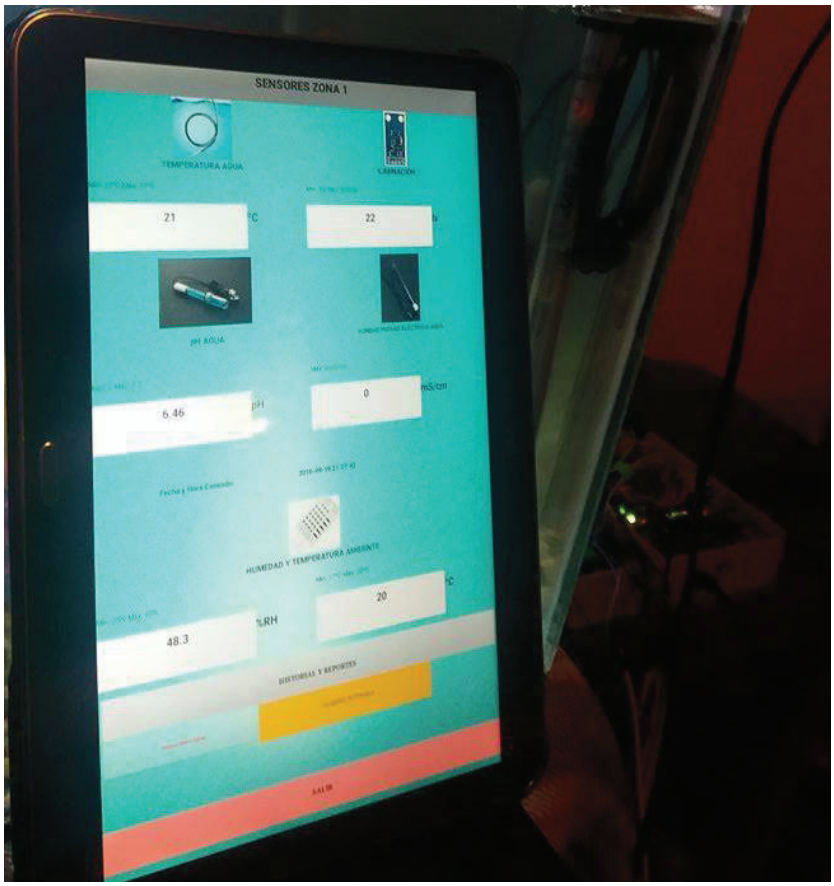


Figura 3.16 Funcionamiento del calentador de agua

3.8.4 PRUEBA DE FUNCIONAMIENTO DE LA LÁMPARA LED

La lámpara led sirve como una señal de alarma cuando el valor del pH se encuentre fuera del rango normal.

La figura 3.17 muestra el funcionamiento de la lámpara led.

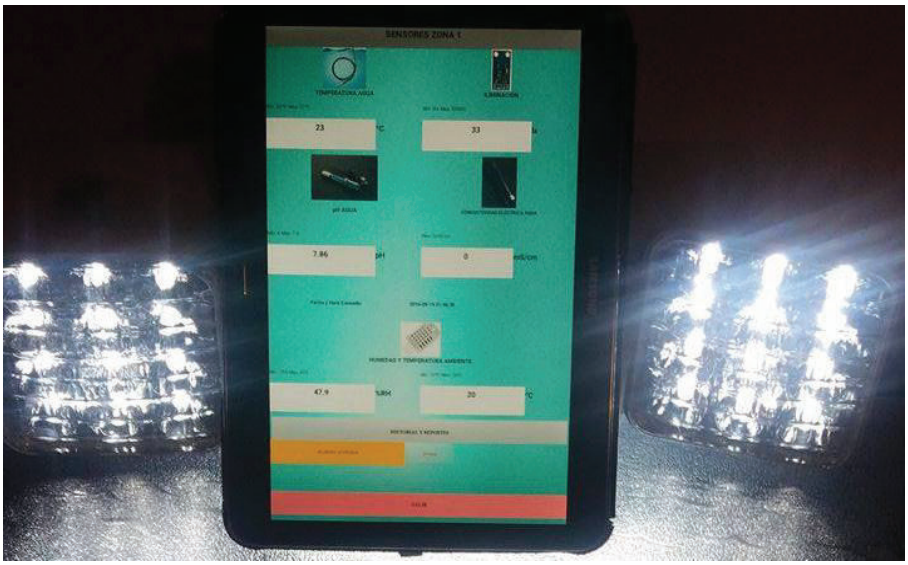


Figura 3.17 Funcionamiento de la lámpara led

3.8.5 PRUEBA DE FUNCIONAMIENTO DE LA ALARMA AUDITIVA

Cuando la conductividad eléctrica sea mayor al rango establecido se activa una alarma auditiva para alertar acerca de este parámetro.

La figura 3.18 muestra la alarma auditiva.

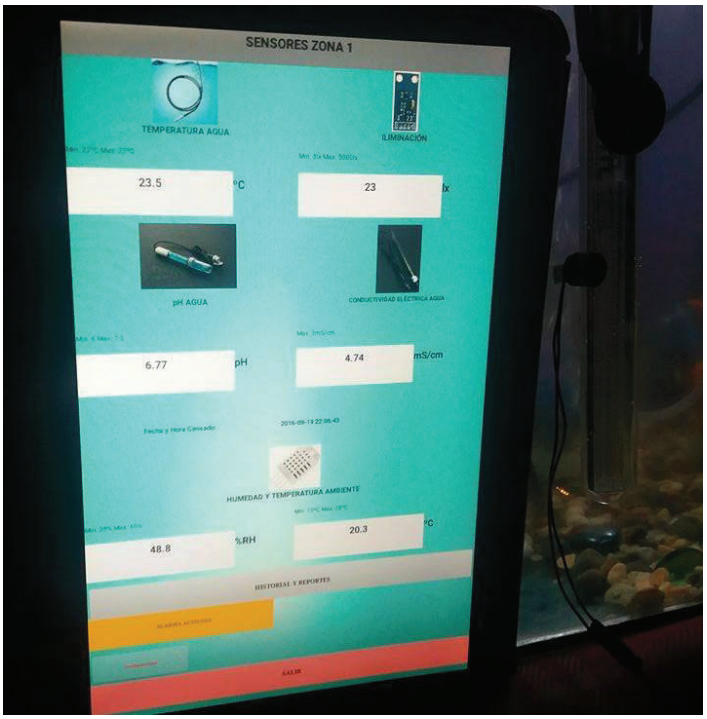


Figura 3.18 Alarma auditiva

Los equipos eléctricos se activan al existir alarmas en los sensores de luz, temperatura del agua y temperatura ambiental, permitiendo corregir los parámetros fuera de rango de manera automática.

En el caso de las alarmas correspondientes a los sensores de pH y CE, los equipos eléctricos permiten únicamente alertar al usuario de forma visual y auditiva, ya que no realizan una acción para corregir el parámetro de forma automática.

3.9 PRUEBA DE FUNCIONAMIENTO DEL DOMINIO Y HOSTING

Para ingresar a la aplicación Android desde Internet se debe contratar una dirección IP pública que se enlazará a la IP privada del servidor, para ello la Escuela Politécnica Nacional facilitó la IP pública 190.96.111.50, con lo cual se realizaron las pruebas de conexión hacia el servidor.

La figura 3.19 muestra la conexión hacia el servidor desde Internet utilizando una IP pública.

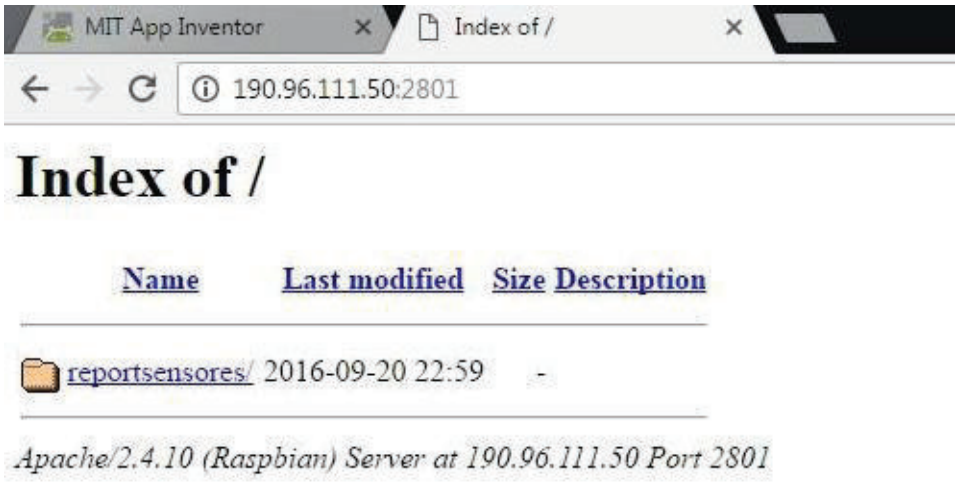


Figura 3.19 Pruebas de conexión hacia el servidor mediante la IP pública desde Internet

La Escuela Politécnica Nacional proporcionó el domino acuario.epn.edu.ec y el alojamiento Hosting anclado a la IP pública, con lo cual se realizan las pruebas de conexión hacia el servidor mediante el domino indicado.

La figura 3.20 muestra la prueba de conexión hacia el servidor, mediante el domino acuario.epn.edu.ec.

Figura 3.20 Prueba de conexión hacia el servidor mediante el dominio acuario.epn.edu.ec

Las pruebas realizadas anteriormente reflejan el funcionamiento de cada una de las etapas que conforman el prototipo, con lo cual se valida el Proyecto implementado.

CAPÍTULO IV

4. PRESUPUESTO REFERENCIAL

Los elementos eléctricos y electrónicos necesarios para la implementación del prototipo fueron adquiridos en diversas tiendas electrónicas sin ningún tipo de inconveniente, salvo el sensor de conductividad eléctrica que fue importado, por lo fue necesario esperar un determinado tiempo para poder adquirirlo.

En el presupuesto referencial para la implementación del prototipo, se detalla el costo incluido IVA de cada uno de los elementos presentes en el mismo. En la tabla 4.1 se indica el presupuesto referencial.

DETALLE	Cantidad	Valor Unidad ($)	Valor Total ($)
PLACAS ELECTRÓNICAS			
Plataforma Raspberry Pi Modelo B	1	73	73
Memoria micro SD 8GB	1	8	8
Cargador Raspberry Pi	1	6	6
Plataforma Arduino Uno R3	1	15	15
Placa Arduino Ethernet Shield W5100	1	15	15
SENSORES			
Sensor de conductividad eléctrica SKU:DFR0300	1	105	105
Sensor de pH SKU:SEN0161	1	65	65
Sensor de humedad y temperatura DHT22	1	12	12
Sensor de Luz BH1750	1	8	8
Sensor sumergible de temperatura DS18B20	1	6	6
EQUIPOS ELÉCTRICOS			
Termostato para acuario HT-875 75W	1	12	12
Lámpara sumergible para acuario Gk140 6W	1	12	12
COMPONENTES ELÉCTRICOS Y ELECTRÓNICOS			
Baquelita 20cmx30cm	1	3,50	3,50
Papel termotransferible	1	1,10	1,10

DETALLE	Cantidad	Valor Unidad ($)	Valor Total ($)
Ácido para baquelita	1	0,60	0,60
Brocas para baquelita	3	0,40	1,20
Alarma auditiva 5V	1	1,50	1,50
Relay SRD-05VDC-SL-C	5	0,75	3,75
Transistor 2N3904	5	0,10	0,50
Resistencia 4.7kΩ	5	0,02	0,12
Resistencia 10kΩ	1	0,02	0,02
Resistencia 470Ω	1	0,02	0,02
Dip-Switch 6 pines	1	1,25	1,25
Zócalo 6 pines	1	0,10	0,10
Espadin macho 40 pines	2	1	2
Espadin hembra 40 pines	2	1,20	2,40
Conector banana hembra	2	0,15	0,30
Conector banana macho	2	0,15	0,30
1 metro cable de luz gemelo 2x16 blanco	4	0,50	2
1 metro cable de luz gemelo 2x16 negro	1	0,50	0,50
1 metro Cable de red UTP categoría 6	4	0,80	3,20
Enchufe hembra 120 voltios	5	0,65	3,25
Enchufe macho 120 voltios	5	0,70	3,50
Kit 40 Conectores macho-macho 20 cm	1	3,25	3,25
Kit 10 Conectores macho-hembra 20 cm	1	1,20	1,20
Kit 10 Conectores macho-hembra 30 cm	1	1,80	1,80
1 metro estaño	3	0,60	1,80
Rollo taipe pequeño	1	0,90	0,90
Cinta para etiqueta	1	0,50	0,50
Patch core 1 metro	2	2	4
COSTO TOTAL DEL PROTOTIPO			**$381,56**

Tabla 4.1 Presupuesto referencial del prototipo

El costo de implementación del prototipo desarrollado resulta más económico en comparación a soluciones similares presentes en el mercado, teniendo en cuenta que la solución desarrollada es escalable, de fácil uso y que si se implementan un mayor número de prototipos, su costo reduciría aún más.

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

Las conclusiones y recomendaciones detalladas a continuación se basan en la experiencia adquirida durante el proceso de diseño, desarrollo e implementación del prototipo, de manera que sirvan como una guía para mejorar el prototipo implementado o sean la base de nuevos Proyectos.

5.1 CONCLUSIONES

- Para el desarrollo de este Proyecto se utilizan sensores analógicos y digitales, para lo cual se usa la placa Arduino UNO que mediante sus pines permite el control de los sensores.
- En el Proyecto desarrollado Arduino trabaja en conjunto con la Raspberry Pi mediante el modelo cliente servidor, lo cual brinda mayor seguridad al prototipo en caso de existir algún tipo de ataque desde Internet.
- La aplicación Android desarrollada en este proyecto permite visualizar desde Internet los parámetros medidos en el acuario independientemente del momento y del lugar en el que se encuentre el usuario.
- En el desarrollo del Proyecto se comprobó que en un acuario de agua dulce conforme transcurre el tiempo el nivel de pH disminuye, esto sucede debido a que el acuario tiende a contaminarse en mayor o menor grado dependiendo del tipo de flora y fauna que habitan en él.
- Mediante las pruebas realizadas en el acuario implementado se concluye que, si la temperatura del agua sobrepasa el rango establecido, los peces tienden a volverse agresivos y en ocasiones pueden morir, por otro lado, si la temperatura del agua baja del rango establecido los peces se debilitan y mueren.
- Para el envío de información obtenida por los sensores a la base de datos en el prototipo implementado, se integró la tarjeta Ethernet Shield a la placa Arduino Uno, ya que Arduino Uno por sí solo no puede conectarse a Internet.

- Mediante las pruebas realizadas se comprobó que, en el caso de que Arduino quede sin conexión a Internet la aplicación Android mostrará los últimos valores registrados en la base de datos, mientras que si la Raspberry Pi no tiene acceso a Internet, la aplicación solamente mostrará un error de conexión hacia el servidor.
- En el prototipo implementado se utiliza una lámpara y un calentador de agua, ya que la luz y la temperatura dentro del acuario es fundamental para el desarrollo biológico y preservación de las especies que habitan en él.
- Para el desarrollo de este Proyecto se configuró el puerto lógico 2801, con lo cual se brinda acceso y seguridad al servidor Apache 2 que se encuentra instalado en la Raspberry Pi 2.
- Para el desarrollo de la aplicación Android del presente proyecto se utilizó el entorno web App Inventor, que permite desarrollar de manera sencilla la aplicación “TesisAcuario” mediante una interfaz gráfica amigable y fácil de interpretar.
- Para el control de los equipos eléctricos utilizados en el presente proyecto se utilizó el programa “control_gpio.py”, desarrollado en lenguaje de programación Python que es propio de la Raspberry Pi y que se caracteriza por respetar estructuras de identación (espacios) para la generación y ejecución del programa.
- La Raspberry Pi representa aproximadamente el 25% del costo total en la implementación del prototipo desarrollado, ya que es la parte central donde se almacena y procesa toda la información necesaria para la comunicación y funcionamiento de los diferentes elementos del prototipo.

5.2 RECOMENDACIONES

- En caso de no poder visualizar el entorno gráfico de la Raspberry Pi mediante un monitor, se recomienda ingresar en la tarjeta micro SD al archivo config.txt y buscar la opción “hdmi_mode=1”, para editar el valor asignado según la resolución y frecuencia del monitor utilizado.
- Para el correcto funcionamiento del prototipo implementado, es recomendable utilizar una fuente de poder para la Raspberry Pi que proporcione una corriente mínima de 2A.

- Se recomienda instalar en la Raspberry Pi una aplicación de escritorio remoto para acceder a la plataforma desde cualquier computador.
- Para la instalación del sistema operativo en la Raspberry Pi, es recomendable utilizar una tarjeta micro SD de al menos 8GB, para no tener inconvenientes de falta de memoria en la instalación de programas.
- Es recomendable verificar el correcto funcionamiento de todos los sensores para garantizar que los datos obtenidos por el prototipo sean confiables.
- Si en el prototipo implementado existen alarmas en las mediciones de pH o CE, se recomienda al usuario corregir personalmente estas variaciones ya que son parámetros difíciles de manipular automáticamente.
- En caso de que no llegue información a la base de datos del prototipo implementado luego de generarse un apagón eléctrico, se recomienda desconectar y volver a conectar el cable USB del Arduino, ya que el equipo pudo haber quedado inhibido.
- Si al compilar en Arduino el programa “envio_sensores_arduino.ino” existen errores referentes a librerías, se recomienda dar permisos generales a cada una de las librerías mediante el terminal de la Raspberry Pi.
- Para el desarrollo de este Proyecto es recomendable configurar una IP fija en la Raspberry Pi, para evitar inconvenientes de direccionamiento e identificar al equipo siempre bajo una misma dirección IP.

REFERENCIAS BIBLIOGRÁFICAS

[1] “Acuario (Recipiente) - EcuRed”. [Online]. Recuperado de: http://www.ecured.cu/Acuario_%28Recipiente%29. (Consultado el 15 de Febrero del 2016).

[2] Arboleda, Duván A.(Agosto 2005). "REDVET - Calidad del agua y mantenimiento de acuarios". [Online]. Recuperado de: http://www.veterinaria.org/revistas/redvet/n080805/080513.pdf. (Consultado el 16 de Febrero del 2016).

[3] "Peces y Acuarios - artemisa.unicauca.edu". [Online]. Recuperado de: http://artemisa.unicauca.edu.co/~almunoz/peces/peces-acuarios.html#inicio. (Consultado el 16 de Febrero del 2016).

[4] “Química del Agua I - elacuarista”. [Online]. Recuperado de: http://www.elacuarista.com/secciones/quimica1_agua.htm. (Consultado el 17 de Febrero del 2016).

[5] Carballar José. "Wi-Fi Cómo construir una red inalámbrica". 2ª edición (2006). Col. Del Valle, México: Alfaomega Grupo Editor. (Consultado el 18 de Febrero del 2016).

[6] Neil Reid, Ron Seide. "MANUAL de redes inalámbricas". 1ª edición (2004). México: McGraw-Hill. (Consultado el 18 de Febrero del 2016).

[7] Evans, Dave.(Abril 2011). “Internet de las cosas - CISCO”. [Online]. Recuperado de: http://www.cisco.com/c/dam/global/es_mx/solutions/executive/assets/pdf/internet-of-things-iot-ibsg.pdf. (Consultado el 19 de Febrero del 2016).

[8] Estévez, Ricardo.(Septiembre 2013). “6 claves que debes conocer sobre Internet de las Cosas - ECOinteligencia”. [Online]. Recuperado de: http://www.ecointeligencia.com/2013/09/6-claves-internet-de-las-cosas-iot/. (Consultado el 19 de Febrero del 2016).

[9] “RASPBERRY PI - Blog Historia de la Informática”. (Diciembre 2013) [Online]. Recuperado de: http://histinf.blogs.upv.es/2013/12/18/raspberry-pi/.

(Consultado el 20 de Febrero del 2016).

[10] Pascual, Juan.(Febrero 2015). “Raspberry Pi: ¿Qué modelo me compro? - ComputerHoy.com”. [Online]. Recuperado de: http://computerhoy.com/noticias/hardware/raspberry-pi-que-modelo-me-compro-23811. (Consultado el 20 de Febrero del 2016).

[11] Isaac, P.(Mayo 2015). “Todo sobre los GPIO Raspberry Pi - comohacer.eu”. [Online]. Recuperado de: http://comohacer.eu/gpio-raspberry-pi/. (Consultado el 20 de Febrero del 2016).

[12] “Historia - ARDUINO: TECNOLOGÍA PARA TODOS”. [Online]. Recuperado de: http://arduinodhtics.weebly.com/historia.html. (Consultado el 22 de Febrero del 2016).

[13] " ARDUINO UNO R3 - ARDUINO.cl". [Online]. Recuperado de: http://arduino.cl/arduino-uno/. (Consultado el 22 de Febrero del 2016).

[14] "Arduino Ethernet Shield - ARDUINO". [Online]. Recuperado de: https://www.arduino.cc/en/Main/ArduinoEthernetShield. (Consultado el 23 de Febrero del 2016).

[15] Olivia, Jesús, Emilio.(Dicembre 2010). “CLASIFICACIÓN DE LOS SENSORES - SENSORES”. [Online]. Recuperado de: http://thelastlabproject.blogspot.com/2010/12/clasificacion-de-los-sensores.html. (Consultado el 23 de Febrero del 2016).

[16] Demay, Vicent.(Junio 2014). "Measure Light with arduino and BH1750 module - Homautomaion". [Online]. Recuperado de: http://www.homautomation.org/2014/06/15/measure-light-with-arduino-and-bh1750-module. (Consultado el 23 de Febrero del 2016).

[17] García, Antony.(Febrero 2014). "DHT22: Sensor de humedad/temperatura de precisión para Arduino - Panama Hitek". [Online]. Recuperado de: http://panamahitek.com/dht22-sensor-de-humedadtemperatura-de-precision-para-arduino. (Consultado el 23 de Febrero del 2016).

[18] García, Antony.(Enero 2014). "Aprendiendo a utilizar el sensor de temperatura DS18B20 - Panama Hitek". [Online]. Recuperado de: http://panamahitek.com/aprendiendo-utilizar-el-sensor-de-temperatura-ds18b20. (Consultado el 23 de Febrero del 2016).

[19] "PH meter (SKU: SEN0161) - openhacks". [Online]. Recuperado de: https://www.openhacks.com/uploadsproductos/ph_meter_sku__sen0161__-_robot_wiki.pdf. (Consultado el 23 de Febrero del 2016).

[20] "Analog EC Meter SKU: DFR0300 - dfrobot". [Online]. Recuperado de: http://www.dfrobot.com/wiki/index.php/Analog_EC_Meter_SKU:DFR0300. (Consultado el 23 de Febrero del 2016).

[21] "Service Name and Transport Protocol Port Number Registry - IANA". [Online]. Recuperado de: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. (Consultado el 24 de Febrero del 2016).

[22] eSandra senior consultant.(Febrero 2012). “Curso de MySQL (I): Introducción a las Bases de Datos Relacionales - eSandra senior consultant”. [Online]. Recuperado de: http://www.esandra.com/mysql-i-introduccion-a-las-bases-de-datos-relacionales/. (Consultado el 24 de Febrero del 2016).

[23] Cortés, Gregorio. “EVOLUCIÓN DE LOS DIFERENTES TIPOS DE ARQUITECTURAS DE SISTEMAS – IEDGE”. [Online]. Recuperado de: http://www.iedge.eu/gregorio-cortes-evolucion-de-los-diferentes-tipos-de-arquitecturas-de-sistemas. (Consultado el 25 de Febrero del 2016).

[24] "¿Qué es Apache? - Culturacion". [Online]. Recuperado de: http://culturacion.com/que-es-apache/. (Consultado el 25 de Febrero del 2016).

[25] "5 consejos para elegir proveedor de hosting – VIRTUAL WEB ESTUDIO". [Online]. Recuperado de: http://www.virtualwebestudio.com/consejos-elegir-proveedor-de-hosting/. (Consultado el 25 de Febrero del 2016).

[26] "Que es el sistema operativo? - INFORMATICAHOY". [Online]. Recuperado de: http://www.informatica-hoy.com.ar/aprender-informatica/Que-es-el-sistema-operativo.php. (Consultado el 26 de Febrero del 2016).

[27] "Android: principales caracteristicas del sistema operativo de Google - Culturacion". [Online]. Recuperado de: http://culturacion.com/android-principales-caracteristicas-del-sistema-operativo-de-google/. (Consultado el 26 de Febrero del 2016).

[28] Orozco, David.(Mayo 2011). “Definición de Linux - CONCEPTODEFINICION.DE”. [Online]. Recuperado de: http://conceptodefinicion.de/linux/. (Consultado el 26 de Febrero del 2016).

[29] "Mac OS - EcuRed". [Online]. Recuperado de: http://www.ecured.cu/Mac_OS. (Consultado el 26 de Febrero del 2016).

[30] "Microsoft Windows - EcuRed". [Online]. Recuperado de: http://www.ecured.cu/Microsoft_Windows. (Consultado el 26 de Febrero del 2016).

[31] Posada, Prieto.(2015). “Diseños de apps con MIT App Inventor 2 – Tabletas en el aula”. [Online]. Recuperado de: http://canaltic.com/tablet/pdf/0506_disenoAppsAI2.pdf. (Consultado el 27 de Febrero del 2016).

[32] “DOWNLOADS - RASPBERRY PI FOUNDATION”. [Online]. Recuperado de: https://www.raspberrypi.org/. (Consultado el 1 de Marzo del 2016).

[33] Zagur.(Junio 2015). “Instalar el IDE de Arduino en GNU/LINUX – Portal Linux”. [Online]. Recuperado de: http://portallinux.es/instalar-el-ide-de-arduino-en-gnulinux/. (Consultado el 2 de Marzo del 2016).

[34] Esquiva, Alejandro.(Agosto 2013). “Tutorial Raspberry Pi - 15. Instalación de Apache + MySQL + PHP – Geeky Theory”. [Online]. Recuperado de: https://geekytheory.com/tutorial-raspberry-pi-15-instalacion-de-apache-mysql-php/. (Consultado el 2 de Mayo del 2016).

[35] “Tutorial - Install PhpMyAdmin on your Raspberry Pi – Ste Wright”. [Online]. Recuperado de: https://www.stewright.me/2012/09/tutorial-install-phpmyadmin-on-your-raspberry-pi/. (Consultado el 4 de Mayo del 2016).

[36] SerInformaticos.(Noviembre 2013). “Como cambiar puertos en Apache – SerInformaticos Servicios Informáticos Especializados”. [Online]. Recuperado de: http://www.serinformaticos.es/como-cambiar-puertos-en-apache/. (Consultado 5 de Mayo del 2016).

[37] Pérez, Mario.(Mayo 2014). “INTERNET DE LAS COSAS [PARTE 2] – SUBIR LOS DATOS A UNA BASE DE DATOS - Geeky Theory”. [Online]. Recuperado de: https://geekytheory.com/internet-de-las-cosas-parte-2-subir-los-datos-a-una-base-de-datos/. (Consultado el 10 de Mayo del 2016).

[38] SiZiOUS.(Agosto 2015). “Cacharreando con la Raspberry Pi (5/8): Poniendo una ip fja - delinux.com”. [Online]. Recuperado de: http://delinux.com/cacharreando-con-la-raspberry-pi-5x-poniendo-una-ip-fija/. (Consultado el 18 de Mayo del 2016).

[39] Cowburn, Peter.(Diciembre 2009). “Manual de PHP – php.net”. [Online]. Recuperado de: http://php.net/manual/. (Consultado el 27 de Mayo del 2016).

[40] “Tutorials for App Inventor - MIT App Inventor”. [Online]. Recuperado de: http://appinventor.mit.edu/explore/ai2/tutorials.html. (Consultado el 6 de Junio del 2016).

[41] Pérez, Mario.(Febrero 2014). “TUTORIAL RASPBERRY PI – GPIO [PARTE 2]: CONTROL DE LEDs CON PYTHON - Geeky Theory”. [Online]. Recuperado de: https://geekytheory.com/tutorial-raspberry-pi-gpio-parte-2-control-de-leds-con-python/. (Consultado el 15 de Junio del 2016).

ANEXOS

ANEXO A

MANUAL DE USUARIO DE LA APLICACIÓN ANDROID

INSTALACIÓN DE MIT AI2 COMPANION

MIT AI2 Companion es una aplicación que permite probar el desarrollo de la aplicación e instalar el archivo .apk que se genera desde la plataforma MIT App Inventor 2. Para instalar esta aplicación se la busca desde cualquier dispositivo Android en Play Store de Google, se descarga y se instala, tal como indica la figura A-1.

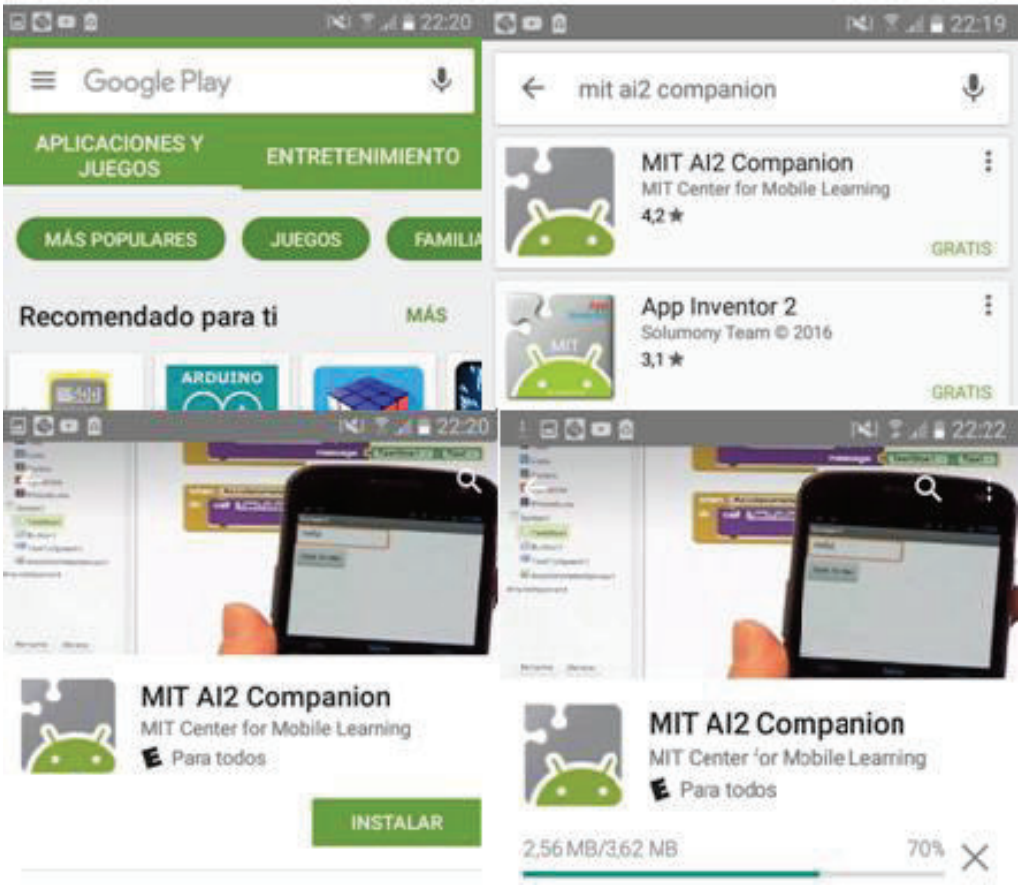


Figura A-1 Proceso de instalación de MIT AI2 Companion

INSTALACIÓN DE LA APLICACIÓN ANDROID

Se genera el archivo .apk de la aplicación en la plataforma MIT App Inventor 2 mediante la opción “App (provide QR code for .apk)” y se instala en el dispositivo Android mediante la aplicación MIT AI2 Companion, tal como se indica en la figura A-2.

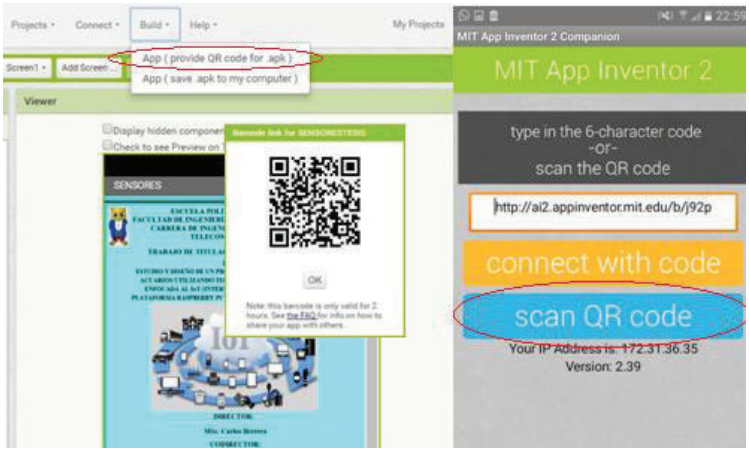


Figura A-2 Instalación de la aplicación Android

USO DE LA APLICACIÓN EN CUALQUIER DISPOSITIVO ANDROID

Una vez instalada la aplicación, debe aparecer el logo de la misma y al ejecutarla se presenta la pantalla de inicio, que permite ingresar a la pantalla de autenticación mediante el botón LOGIN, tal como se muestra en la figura A-3.

Figura A-3 Logo y pantalla de inicio de la aplicación Android

Al ingresar a la pantalla de autenticación se debe escribir un nombre de usuario y contraseña válido para poder acceder a la información de una determinada zona, en caso de ingresar una contraseña o usuario no válido la aplicación negará el acceso y pedirá que se realice nuevamente el proceso de autenticación hasta un máximo de tres intentos, caso contrario mostrará un mensaje de conexión fallida ya que se excedió el número de intentos permitidos.

La figura A-4 muestra los mensajes de autenticación fallida.

Figura A-4 Mensajes de autenticación fallida

En la tabla A-1 se muestra la autenticación válida para cada una de las zonas.

USUARIO	CONTRASEÑA	ACCESO
ROOT	SENSORES	Todas las zonas
ZONA1	SENSORES1	ZONA 1
ZONA2	SENSORES2	ZONA 2
ZONA3	SENSORES3	ZONA 3
ZONA4	SENSORES4	ZONA 4
ZONA5	SENSORES5	ZONA 5
ZONA6	SENSORES6	ZONA 6
ZONA7	SENSORES7	ZONA 7
ZONA8	SENSORES8	ZONA 8
ZONA9	SENSORES9	ZONA 9
ZONA10	SENSORES10	ZONA 10

Tabla A-1 Autenticación para cada zona

La figura A-5 muestra la pantalla ZONAS que se obtiene al autenticarse como usuario ROOT.

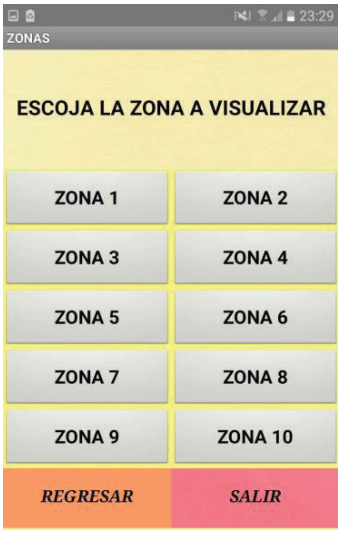


Figura A-5 Pantalla ZONAS para el usuario ROOT

Una vez que el usuario ingresa a una zona determinada observará la información de cada todos los sensores. Además tendrá la opción de revisar el historial y reporte de cualquiera de los sensores mediante el botón HISTORIAL Y REPORTES, tal como se indica en la figura A-6.

Figura A-6 Pantalla de datos e Historial y Reportes de la zona 1

ANEXO B

CÓDIGOS CREADOS EN ARDUINO Y LA RASPBERRY PI

CÓDIGO DE PROGRAMACIÓN PARA ARDUINO

El código B-1 muestra el programa envio_sensores_arduino.ino desarrollado en Arduino.

```
//Se cargan las librerías necesarias para el sensor

#include <Ethernet.h>
#include <SPI.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <DHT.h>
#include <BH1750.h>
#include <Wire.h>

//Se declara el pin donde se conecta el dato del sensor
#define pin18b20 8
#define pindht 9
#define num DHT22
#define pinpH A3
byte pinCE = A2;
BH1750 sluz;

// Configuracion del Ethernet Shield
byte MAC[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x12}; //Dirección MAC
byte IP_CLIENTE[] = {172,31,36,34}; //Dirección IP del Arduino
byte IP_SERVIDOR[] = {172,31,36,37}; //Dirección IP del servidor
EthernetClient client;

//Se define e inicializa las variables
OneWire conex(pin18b20);
DallasTemperature s18b20(&conex);

const byte nlecturas = 20;
unsigned int intervmuestra=25,intervimpr=700;
unsigned int lecturas[nlecturas];
byte i = 0;
byte j = 0;
unsigned long valorT = 0;
unsigned int  valorprom= 0,voltprom=0;
unsigned long tmuesta,timpr;
float CE;

DHT sdht(pindht,num);

unsigned int spH;
int k,guardar[10];
```

```
float pH;

//Seteo de funciones
void setup() {
  Ethernet.begin(MAC, IP_CLIENTE);
  Serial.begin(9600);
  s18b20.begin();

  for (i = 0; i < nlecturas; i++)
    lecturas[i] = 0;
  tmuesta=millis();
  timpr=millis();

  sdht.begin();

  sluz.begin();
}

//Lazo de repetición
void loop() {
  s18b20.requestTemperatures();
  float Tagua=s18b20.getTempCByIndex(0);

  if(millis()-tmuesta>=intervmuestra)
  {
    tmuesta=millis();
    valorT = valorT - lecturas[j];
    lecturas[j] = analogRead(pinCE);
    valorT = valorT + lecturas[j];
    j = j + 1;
    if (j >= nlecturas)
    j = 0;
    valorprom = valorT / nlecturas;
  }

  float H=sdht.readHumidity();
  float T=sdht.readTemperature();

  uint16_t lx = sluz.readLightLevel();

  for(k=0;k<10;k++)
  {
    guardar[k]=analogRead(pinpH);
    delay(10);
  }
  spH=0;
  for(k=0;k<10;k++)spH+=guardar[k];
  pH=(float)spH*5/1024/10;//convierte el valor analógico en
milivoltios
  pH=3.5*pH;//Convierte el valor de mv a unidades de pH

  // Imprimir variables
  Serial.println(" ");
  Serial.print("Temperatura_Agua: ");
  Serial.print(Tagua);
```

```
  Serial.println(" *C");

  if(millis()-timpr>=intervimpr)
  {
    timpr=millis();
    voltprom=valorprom*(float)5000/1024;
    Serial.print("Voltage: ");
    Serial.print(voltprom);
    Serial.print("mV");
    Serial.print("     CE: ");
    //cálculo de CE a diferente temperatura:
coefvolt=voltprom/(1.0+0.0185*(tempactual-25.0))
    float coeftemp=1.0+0.0185*(Tagua-25.0);
    float coefvolt=(float)voltprom/coeftemp;
    if(coefvolt<150)Serial.println("<1mS/cm");
    else if(coefvolt>3300)Serial.println(">20mS/cm");
    else
    {
      if(coefvolt<=448)CE=6.84*coefvolt-64.32;
//1ms/cm<EC<=3ms/cm
      else if(coefvolt<=1457)CE=6.98*coefvolt-127;
//3ms/cm<EC<=10ms/cm
      else CE=5.3*coefvolt+2278;
//10ms/cm<EC<20ms/cm
      CE/=1000;
      Serial.print(CE,2);
      Serial.println("ms/cm");
    }
  }

  Serial.print("Humedad_Ambiental: ");
  Serial.print(H);
  Serial.println(" %HR");
  Serial.print("Temperatura_Ambiental: ");
  Serial.print(T);
  Serial.println(" *C");

  Serial.print("Iluminacion: ");
  Serial.print(lx);
  Serial.println(" lx");

  Serial.print("pH: ");
  Serial.print(pH);
  Serial.println(" ");
  Serial.println(" ");

  delay(2000);


  // Proceso de envio de muestras al servidor
  if (client.connect(IP_SERVIDOR, 2801)>0)
  {
    client.print("GET /reportsensores/reptempagua.php?gradC=");
    client.print(Tagua);
     client.println();
```

```
  Serial.println("Dato enviado Tagua");
}
else
{
  Serial.println("Sin conexion");
}
client.stop();
client.flush();
delay(2000);


if (client.connect(IP_SERVIDOR, 2801)>0)
{
  client.print("GET /reportsensores/repce.php?mSxcm=");
  client.print(CE);
  client.println();
  Serial.println("Dato enviado CE");
}
else
{
  Serial.println("Sin conexion");
}
client.stop();
client.flush();
delay(2000);


if (client.connect(IP_SERVIDOR, 2801)>0)
{
  client.print("GET /reportsensores/rephumamb.php?RH=");
  client.print(H);
  client.println();
  Serial.println("Dato enviado Humedad_Ambiental");
}
else
{
  Serial.println("Sin conexion");
}
client.stop();
client.flush();
delay(2000);


if (client.connect(IP_SERVIDOR, 2801)>0)
{
  client.print("GET /reportsensores/reptempamb.php?gradoC=");
  client.print(T);
  client.println();
  Serial.println("Dato enviado Temperatura_Ambiental");
}
else
{
  Serial.println("Sin conexion");
}
client.stop();
```

```
  client.flush();
  delay(2000);


  if (client.connect(IP_SERVIDOR, 2801)>0)
  {
    client.print("GET /reportsensores/repilum.php?lx=");
    client.print(lx);
    client.println();
    Serial.println("Dato enviado Iluminacion");
  }
  else
  {
    Serial.println("Sin conexion");
  }
  client.stop();
  client.flush();
  delay(2000);


  if (client.connect(IP_SERVIDOR, 2801)>0)
  {
    client.print("GET /reportsensores/reph.php?pH=");
    client.print(pH);
    client.println();
    Serial.println("Dato enviado pH");
  }
  else
  {
    Serial.println("Sin conexion");
  }
  client.stop();
  client.flush();
  delay(2000);
}
```

Código B-1 Programa envio_sensores_arduino.ino

CÓDIGO DE PROGRAMACIÓN EN PYTHON

El código B-2 muestra el programa control_gpio.py desarrollado en Python.

```
# Programa para el control de puertos gpio de la Raspberry Pi en
Python

#!/usr/bin/python  Indica la dirección desde donde se ejecuta
Python

#Se cargan los módulos y librerías necesarias para el programa
import RPi.GPIO as GPIO
from urllib2 import urlopen
from datetime import datetime, time

GPIO.setmode(GPIO.BCM)     #Define el modo de numeración de los
pines
```

```
GPIO.setwarnings(False)    #desactiva las alertas en caso de que
el canal gpio este en uso

#Define los puertos GPIO como salidas
GPIO.setup(4, GPIO.OUT)   #Sensor DHT22
GPIO.setup(18, GPIO.OUT)   #Sensor BH1750
GPIO.setup(23, GPIO.OUT)   #Sensor PH
GPIO.setup(24, GPIO.OUT)   #Sensor CE
GPIO.setup(27, GPIO.OUT)   #Sensor DS18b20

#Lazo de repetición
i=1
while i<3:

    #Control para el sensor de humedad y temperatura DHT22
    dtempwww =
urlopen('http://www.accuweather.com/es/ec/quito/129846/weather-
forecast/129846')
    Tempwww = ""
    for line in dtempwww:
        if '<span class="local-temp">' in line:
            Tempwww=line.replace('<span class="local-temp">','')
            Tempwww=Tempwww.replace('&deg;C</span>','').strip()
            break
    dTempwww = float(Tempwww)
    dhtdato=
urlopen('http://172.31.36.37:2801/reportsensores/setreptempamb.php
')
    dht22= float(dhtdato.read())
    if ( dht22 >= 28 and dTempwww >= 24 ) :
        GPIO.output(4, True)
    else:
        GPIO.output(4, False)


    #Control para el sensor de luz BH1750
    datbh1750 =
urlopen('http://172.31.36.37:2801/reportsensores/setrepilum.php')
    bh1750 = float(datbh1750.read())
    horabh = datetime.now()  # Obtiene fecha y hora actual
    if ( bh1750 < 30 and ( horabh.hour >= 18 or horabh.hour <= 6 )
) :
        GPIO.output(18, True)
    else:
        GPIO.output(18, False)


    #Control para el sensor sumergible de temperatura del agua
DS18b20
    dat18b20 =
urlopen('http://172.31.36.37:2801/reportsensores/setreptempagua.ph
p')
    ds18b20 = float(dat18b20.read())
    if ( ds18b20 < 22 and dht22 < 24 ) :
         GPIO.output(27, True)
```

```
    else:
        GPIO.output(27, False)

    #Control para el sensor de pH y conductividad eléctrica
    datph =
urlopen('http://172.31.36.37:2801/reportsensores/setreph.php')
    dph = float(datph.read())
    datce =
urlopen('http://172.31.36.37:2801/reportsensores/setrepce.php')
    dce = float(datce.read())
    if ( (dph < 6) or ( dph > 7.5 and dce <= 3 )) :
        GPIO.output(23, False)
    else:
        GPIO.output(23, True)

    if ( dce > 3 and ds18b20 <= 22 ) :
        GPIO.output(24, True)
    else:
        GPIO.output(24, False)
```

Código B-2 Programa control_gpio.py

CÓDIGOS PHP GENERADOS EN LA RASPBERRY PI

El código B-3 muestra el programa conexión.php.

```
<?php

   //Autenticación
   $equipobd = "localhost";
   $usuariobd = "ardupi";
   $clavebd = "inopi";
   $nombrebd = "repsensores";
   //Conexión con mysql
    $conex = mysqli_connect($equipobd, $usuariobd, $clavebd,
$nombrebd);
?>
```

Código B-3 Programa conexion.php

El código B-4 muestra el programa repce.php.

```
<?php
    //Se hace el llamado al script conexion.php
    require("conexion.php");
    //Se lee el dato de Conductividad Eléctrica mediante GET
    $datoce = mysqli_real_escape_string($conex, $_GET['mSxcm']);
    //Se guarda el valor para insertar en la base de datos
    $insertarce = "INSERT INTO conductividad_electrica(mSxcm)
VALUES('".$datoce."')";
    //Se inserta el valor en la base de datos
    mysqli_query($conex, $insertarce);
    mysqli_close($conex);
?>
```

Código B-4 Programa repce.php

El código B-5 muestra el programa reph.php.

```
<?php
    require("conexion.php");
    $datoph = mysqli_real_escape_string($conex, $_GET['pH']);
    $insertarph = "INSERT INTO nivel_pH(pH)
VALUES('".$datoph."')";
    mysqli_query($conex, $insertarph);
    mysqli_close($conex);
?>
```

Código B-5 Programa reph.php

El código B-6 muestra el programa rephumamb.php.

```
<?php
    require("conexion.php");
    $datohuma = mysqli_real_escape_string($conex, $_GET['RH']);
    $insertarhuma = "INSERT INTO humedad_ambiental(RH)
VALUES('".$datohuma."')";
    mysqli_query($conex, $insertarhuma);
    mysqli_close($conex);
?>
```

Código B-6 Programa rephumamb.php

El código B-7 muestra el programa repilum.php.

```
<?php
    require("conexion.php");
    $datoilum = mysqli_real_escape_string($conex, $_GET['lx']);
    $insertarilum = "INSERT INTO iluminacion(lx)
VALUES('".$datoilum."')";
    mysqli_query($conex, $insertarilum);
    mysqli_close($conex);
?>
```

Código B-7 Programa repilum.php

El código B-8 muestra el programa reptempagua.php.

```
<?php

    require("conexion.php");
    $datotagua = mysqli_real_escape_string($conex,
$_GET['gradC']);
    $insertartagua = "INSERT INTO temperatura_agua(gradC)
VALUES('".$datotagua."')";
    mysqli_query($conex, $insertartagua);
    mysqli_close($conex);
?>
```

Código B-8 Programa reptempagua.php

El código B-9 muestra el programa reptempamb.php.

```
<?php
    require("conexion.php");
    $datotamb = mysqli_real_escape_string($conex,
$_GET['gradoC']);
    $insertartamb = "INSERT INTO temperatura_ambiental(gradoC)
VALUES('".$datotamb."')";
    mysqli_query($conex, $insertartamb);
    mysqli_close($conex);
?>
```

Código B-9 Programa reptemamb.php

El código B-10 muestra el programa setrepce.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $selecciona = "SELECT * FROM conductividad_electrica order by
fecha_CE desc";
    $valorce = mysql_query($selecciona);
    $recupera =mysql_fetch_object($valorce);
    echo $recupera->mSxcm;
?>
```

Código B-10 Programa setrepce.php

El código B-11 muestra el programa setreph.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $selecciona = "SELECT * FROM nivel_pH order by fecha_pH desc";
    $valorph = mysql_query($selecciona);
    $recupera=mysql_fetch_object($valorph);
    echo $recupera->pH;
?>
```

Código B-11 Programa setreph.php

El código B-12 muestra el programa setrephumamb.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $selecciona = "SELECT * FROM humedad_ambiental order by
fecha_hum desc";
    $valorhamb = mysql_query($selecciona);
    $recupera=mysql_fetch_object($valorhamb);
    echo $recupera->RH;
?>
```

Código B-12 Programa setrephumamb.php

El código B-13 muestra el programa setrepilum.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $selecciona = "SELECT * FROM iluminacion order by fecha_lx
desc";
    $valorlx = mysql_query($selecciona);
    $recupera=mysql_fetch_object($valorlx);
    echo $recupera->lx;
?>
```

Código B-13 Programa setrepilum.php

El código B-14 muestra el programa setreptempagua.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $selecciona = "SELECT * FROM temperatura_agua order by
fecha_tagua desc";
    $valortagua = mysql_query($selecciona);
    $recupera=mysql_fetch_object($valortagua);
    echo $recupera->gradC;
?>
```

Código B-14 Programa setreptempagua.php

El código B-15 muestra el programa setreptempamb.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $selecciona = "SELECT * FROM temperatura_ambiental order by
fecha_tamb desc";
    $valortamb = mysql_query($selecciona);
    $recupera=mysql_fetch_object($valortamb);
    echo $recupera->gradoC;
?>
```

Código B-15 Programa setreptempamb.php

El código B-16 muestra el programa fsetrepce.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $sql = "SELECT * FROM conductividad_electrica order by
fecha_CE desc";
    $valor = mysql_query($sql);
    $reg=mysql_fetch_object($valor);
    echo $reg->fecha_CE;
?>
```

Código B-16 Programa fsetrepce.php

El código B-17 muestra el programa histce.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $consulta = "SELECT * FROM conductividad_electrica order by
fecha_CE desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> HISTORIAL</b><br>';
       echo '<b>CONDUCTIVIDAD ELECTRICA</b>';
       echo '<tr>';
       echo '<th>mSxcm</th>';
       echo '<th>Fecha_CE</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['mSxcm'].'</td>';
       echo '<td>'.$reg['fecha_CE'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
    echo '</table>';
?>
```

Código B-17 Programa histce.php

El código B-18 muestra el programa histempagua.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $consulta = "SELECT * FROM temperatura_agua order by
fecha_tagua desc";
    $valor = mysql_query($consulta);
    echo '<table border=>';
       echo '<b> HISTORIAL</b><br>';
       echo '<b>TEMPERATURA DEL AGUA</b>';
       echo '<tr>';
       echo '<th>°C</th>';
       echo '<th>Fecha_tempagua</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['gradC'].'</td>';
       echo '<td>'.$reg['fecha_tagua'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-18 Programa histempagua.php

El código B-19 muestra el programa histempamb.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $consulta = "SELECT * FROM temperatura_ambiental order by
fecha_tamb desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> HISTORIAL</b><br>';
       echo '<b>TEMPERATURA AMBIENTAL</b>';
       echo '<tr>';
       echo '<th>°C</th>';
       echo '<th>Fecha_tempamb</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['gradoC'].'</td>';
       echo '<td>'.$reg['fecha_tamb'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-19 Programa histempamb.php

El código B-20 muestra el programa histhumamb.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $consulta = "SELECT * FROM humedad_ambiental order by
fecha_hum desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> HISTORIAL</b><br>';
       echo '<b>HUMEDAD AMBIENTAL</b>';
       echo '<tr>';
       echo '<th>RH</th>';
       echo '<th>Fecha_humedad</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['RH'].'</td>';
       echo '<td>'.$reg['fecha_hum'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-20 Programa histhumamb.php

El código B-21 muestra el programa histilum.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $consulta = "SELECT * FROM iluminacion order by fecha_lx
desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> HISTORIAL</b><br>';
       echo '<b> ILUMINACION</b>';
       echo '<tr>';
       echo '<th>lx</th>';
       echo '<th>Fecha_lx</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['lx'].'</td>';
       echo '<td>'.$reg['fecha_lx'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-21 Programa histilum.php

El código B-22 muestra el programa hisph.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $consulta = "SELECT * FROM nivel_pH order by fecha_pH desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> HISTORIAL</b><br>';
       echo '<b>NIVEL PH</b>';
       echo '<tr>';
       echo '<th>pH</th>';
       echo '<th>Fecha_pH</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['pH'].'</td>';
       echo '<td>'.$reg['fecha_pH'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-22 Programa histph.php

El código B-23 muestra el programa reportce.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
    $consulta = "SELECT * FROM conductividad_electrica WHERE mSxcm
>= 3 order by fecha_CE desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> REPORTE</b><br>';
       echo '<b>CONDUCTIVIDAD ELECTRICA</b>';
       echo '<tr>';
       echo '<th>mSxcm</th>';
       echo '<th>Fecha_CE</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['mSxcm'].'</td>';
       echo '<td>'.$reg['fecha_CE'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-23 Programa reportce.php

El código B-24 muestra el programa reportempagua.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
   $consulta = "SELECT * FROM temperatura_agua WHERE gradC not
BETWEEN 22 and 27 order by fecha_tagua desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> REPORTE</b><br>';
       echo '<b>TEMPERATURA DEL AGUA<b>';
       echo '<tr>';
       echo '<th>°C</th>';
       echo '<th>Fecha_TempAgua</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['gradC'].'</td>';
       echo '<td>'.$reg['fecha_tagua'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-24 Programa reportempagua.php

El código B-25 muestra el programa reportempamb.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
   $consulta = "SELECT * FROM temperatura_ambiental WHERE gradoC
not BETWEEN 12 and 28 order by fecha_tamb desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> REPORTE</b><br>';
       echo '<b>TEMPERATURA AMBIENTAL</b>';
       echo '<tr>';
       echo '<th>°C</th>';
       echo '<th>Fecha_TempAmb</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['gradoC'].'</td>';
       echo '<td>'.$reg['fecha_tamb'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-25 Programa reportempamb.php

El código B-26 muestra el programa reporthumamb.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
   $consulta = "SELECT * FROM humedad_ambiental WHERE RH not
BETWEEN 25 and 60 order by fecha_hum desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> REPORTE</b><br>';
       echo '<b>HUMEDAD RELATIVA</b>';
       echo '<tr>';
       echo '<th>%RH</th>';
       echo '<th>Fecha_HumRelativa</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['RH'].'</td>';
       echo '<td>'.$reg['fecha_hum'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-26 Programa reporthumamb.php

El código B-27 muestra el programa reportilum.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
   $consulta = "SELECT * FROM iluminacion WHERE lx not BETWEEN 3
and 5000 order by fecha_lx desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> REPORTE</b><br>';
       echo '<b> ILUMINACION</b>';
       echo '<tr>';
       echo '<th>lx</th>';
       echo '<th>Fecha_ilum</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['lx'].'</td>';
       echo '<td>'.$reg['fecha_lx'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-27 Programa reportilum.php

El código B-28 muestra el programa reportph.php.

```
<?php
    $conex = mysql_connect('localhost','ardupi','inopi');
    mysql_select_db('repsensores',$conex);
   $consulta = "SELECT * FROM nivel_pH WHERE pH not BETWEEN 6 and
7.5 order by fecha_pH desc";
    $valor = mysql_query($consulta);
    echo '<table border=1>';
       echo '<b> REPORTE</b><br>';
       echo '<b>NIVEL PH</b>';
       echo '<tr>';
       echo '<th>pH</th>';
       echo '<th>Fecha_pH</th>';
       echo '</tr>';
       while($reg=mysql_fetch_array($valor)){
       echo '<tr>';
       echo '<td>'.$reg['pH'].'</td>';
       echo '<td>'.$reg['fecha_pH'].'</td>';
       echo '</tr>';
   }
    mysqli_close($conex);
   echo '</table>';
?>
```

Código B-28 Programa reportph.php

ANEXO C

INFORMACIÓN TÉCNICA DE SENSORES Y ELEMENTOS UTILIZADOS

SENSOR DE LUZ BH1750

ROHM SEMICONDUCTOR — Technical Note

Ambient Light Sensor IC Series

Digital 16bit Serial Output Type Ambient Light Sensor IC

BH1750FVI

No.11046EDT01

●**Descriptions**

BH1750FVI is an digital Ambient Light Sensor IC for I²C bus interface. This IC is the most suitable to obtain the ambient light data for adjusting LCD and Keypad backlight power of Mobile phone. It is possible to detect wide range at High resolution. (1 - 65535 lx).

●**Features**

1) I²C bus Interface (f / s Mode Support)
2) Spectral responsibility is approximately human eye response
3) Illuminance to Digital Converter
4) Wide range and High resolution. (1 - 65535 lx)
5) Low Current by power down function
6) 50Hz / 60Hz Light noise reject-function
7) 1.8V Logic input interface
8) No need any external parts
9) Light source dependency is little. (ex. Incandescent Lamp. Fluorescent Lamp. Halogen Lamp. White LED. Sun Light)
10) It is possible to select 2 type of I²C slave-address.
11) Adjustable measurement result for influence of optical window
(It is possible to detect min. 0.11 lx, max. 100000 lx by using this function.)
12) Small measurement variation (+/- 20%)
13) The influence of infrared is very small.

●**Applications**

Mobile phone, LCD TV, NOTE PC, Portable game machine, Digital camera, Digital video camera, PDA, LCD display

●**Absolute Maximum Ratings**

Parameter	Symbol	Ratings	Units
Supply Voltage	Vmax	4.5	V
Operating Temperature	Topr	-40～85	℃
Storage Temperature	Tstg	-40～100	℃
SDA Sink Current	Imax	7	mA
Power Dissipation	Pd	260※	mW

※ 70mm × 70mm × 1.6mm glass epoxy board. Derating in done at 3.47mW/℃ for operating above Ta=25℃.

●**Operating Conditions**

Parameter	Symbol	Ratings			Units
		Min.	Typ.	Max.	
Vcc Voltage	Vcc	2.4	3.0	3.6	V
I²C Reference Voltage	VDVI	1.65	-	Vcc	V

●Electrical Characteristics (Vcc = 3.0V, DVI = 3.0V, Ta = 25℃, unless otherwise noted)

Parameter	Symbol	Limits			Units	Conditions
		Min.	Typ.	Max.		
Supply Current	Icc1	–	120	190	μA	Ev = 100 lx ※1
Powerdown Current	Icc2	–	0.01	1.0	μA	No input Light
Peak Wave Length	λp	–	560	–	nm	
Measurement Accuracy	S/A	0.96	1.2	1.44	times	Sensor out / Actual lx EV = 1000 lx ※1, ※2
Dark (0 lx) Sensor out	S0	0	0	3	count	H-Resolution Mode ※3
H-Resolution Mode Resolution	rHR	–	1	–	lx	
L-Resolution Mode Resolution	rLR	–	4	–	lx	
H-Resolution Mode Measurement Time	tHR	–	120	180	ms	
L-Resolution Mode Measurement Time	tLR	–	16	24	ms	
Incandescent / Fluorescent Sensor out ratio	rIF	–	1	–	times	EV = 1000 lx
ADDR Input 'H' Voltage	VAH	0.7 * VCC	–	–	V	
ADDR Input 'L' Voltage	VAL	–	–	0.3 * VCC	V	
DVI Input 'L' Voltage	VDVL	–	–	0.4	V	
SCL, SDA Input 'H' Voltage 1	VIH1	0.7 * DVI	–	–	V	DVI ≧ 1.8V
SCL, SDA Input 'H' Voltage 2	VIH2	1.26	–	–	V	1.65V ≦ DVI <1.8V
SCL, SDA Input 'L' Voltage 1	VIL1	–	–	0.3 * DVI	V	DVI ≧ 1.8V
SCL, SDA Input 'L' Voltage 2	VIL2	–	–	DVI – 1.26	V	1.65V ≦ DVI < 1.8V
SCL, SDA, ADDR Input 'H' Current	IIH	–	–	10	μA	
SCL, SDA, ADDR Input 'L' Current	IIL	–	–	10	μA	
I^2C SCL Clock Frequency	fSCL	–	–	400	kHz	
I^2C Bus Free Time	tBUF	1.3	–	–	μs	
I^2C Hold Time (repeated) START Condition	tHDSTA	0.6	–	–	μs	
I^2C Set up time for a Repeated START Condition	tSUSTA	0.6	–	–	μs	
I^2C Set up time for a Repeated STOP Condition	tSUSTD	0.6	–	–	μs	
I^2C Data Hold Time	tHDDAT	0	–	0.9	μs	
I^2C Data Setup Time	tSUDAT	100	–	–	ns	
I^2C 'L' Period of the SCL Clock	tLOW	1.3	–	–	μs	
I^2C 'H' Period of the SCL Clock	tHIGH	0.6	–	–	μs	
I^2C SDA Output 'L' Voltage	VOL	0	–	0.4	V	IOL = 3 mA

※1 White LED is used as optical source.
※2 Measurement Accuracy typical value is possible to change '1' by "Measurement result adjustment function".
※3 Use H-resolution mode or H-resolution mode2 if dark data (less than 10 lx) is need.

SENSOR DE HUMEDAD Y TEMPERATURA DHT22

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors

Digital-output relative humidity & temperature sensor/module

DHT22 (DHT22 also named as AM2302)

1. Feature & Application:

* Full range temperature compensated * Relative humidity and temperature measurement
* Calibrated digital signal *Outstanding long-term stability *Extra components not needed
* Long transmission distance * Low power consumption *4 pins packaged and fully interchangeable

2. Description:

DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

Small size & low consumption & long transmission distance(20m) enable DHT22 to be suited in all kinds of harsh application occasions.

3. Technical Specification:

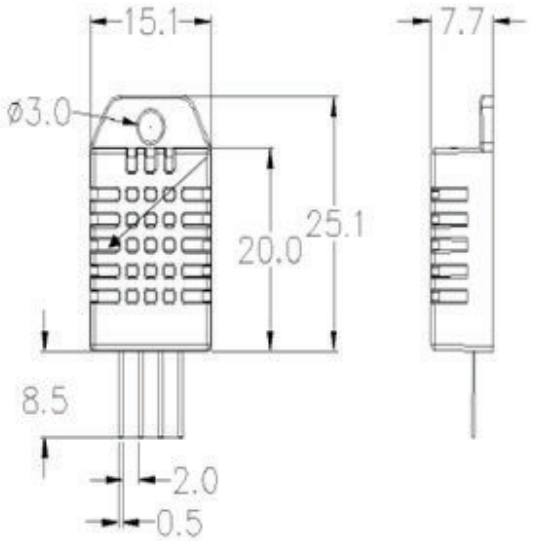
Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors

4. Dimensions: (unit----mm)

1) Small size dimensions: (unit----mm)



Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD----power supply
2	DATA--signal
3	NULL
4	GND

If signal from DHT22 is always high-voltage-level, it means DHT22 is not working properly, please check the electrical connection status.

Electrical Characteristics:

Item	Condition	Min	Typical	Max	Unit
Power supply	DC	3.3	5	6	V
Current supply	Measuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting period	Second		2		Second

*Collecting period should be : >2 second.

SENSOR SUMERGIBLE DE TEMPERATURA DS18B20

DS18B20

Programmable Resolution 1-Wire Digital Thermometer

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

Applications

- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

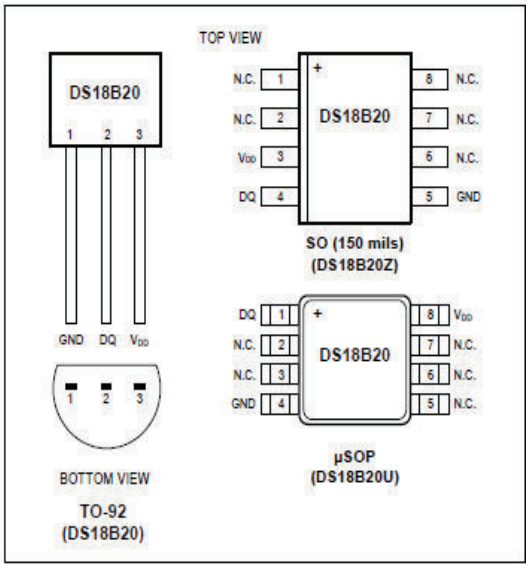
Ordering Information appears at end of data sheet.

1-Wire is a registered trademark of Maxim Integrated Products, Inc.

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations



Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground.....-0.5V to +6.0V
Operating Temperature Range..... -55°C to +125°C
Storage Temperature Range..... -55°C to +125°C
Solder Temperature.....Refer to the IPC/JEDEC J-STD-020 Specification.

These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

DC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Supply Voltage	V_{DD}	Local power (Note 1)		+3.0		+5.5	V
Pullup Supply Voltage	V_{PU}	Parasite power	(Notes 1, 2)	+3.0		+5.5	V
		Local power		+3.0		V_{DD}	
Thermometer Error	t_{ERR}	-10°C to +85°C	(Note 3)			±0.5	°C
		-55°C to +125°C				±2	
Input Logic-Low	V_{IL}	(Notes 1, 4, 5)		-0.3		+0.8	V
Input Logic-High	V_{IH}	Local power	(Notes 1,6)	+2.2		The lower of 5.5 or V_{DD} + 0.3	V
		Parasite power		+3.0			
Sink Current	I_L	$V_{I/O}$ = 0.4V		4.0			mA
Standby Current	I_{DDS}	(Notes 7, 8)			750	1000	nA
Active Current	I_{DD}	V_{DD} = 5V (Note 9)			1	1.5	mA
DQ Input Current	I_{DQ}	(Note 10)			5		µA
Drift		(Note 11)			±0.2		°C

Note 1: All voltages are referenced to ground.
Note 2: The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18B20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU_ACTUAL} = V_{PU_IDEAL} + V_{TRANSISTOR}$.
Note 3: See typical performance curve in Figure 1.
Note 4: Logic-low voltages are specified at a sink current of 4mA.
Note 5: To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
Note 6: Logic-high voltages are specified at a source current of 1mA.
Note 7: Standby current specified up to +70°C. Standby current typically is 3µA at +125°C.
Note 8: To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
Note 9: Active current refers to supply current during active temperature conversions or EEPROM writes.
Note 10: DQ line is high ("high-Z" state).
Note 11: Drift data is based on a 1000-hour stress test at +125°C with V_{DD} = 5.5V.

AC Electrical Characteristics–NV Memory

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
NV Write Cycle Time	t_{WR}			2	10	ms
EEPROM Writes	N_{EEWR}	-55°C to +55°C	50k			writes
EEPROM Data Retention	t_{EEDR}	-55°C to +55°C	10			years

AC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Temperature Conversion Time	t_{CONV}	9-bit resolution	(Note 12)			93.75	ms
		10-bit resolution				187.5	
		11-bit resolution				375	
		12-bit resolution				750	
Time to Strong Pullup On	t_{SPON}	Start convert T command issued				10	µs
Time Slot	t_{SLOT}	(Note 12)		60		120	µs
Recovery Time	t_{REC}	(Note 12)		1			µs
Write 0 Low Time	t_{LOW0}	(Note 12)		60		120	µs
Write 1 Low Time	t_{LOW1}	(Note 12)		1		15	µs
Read Data Valid	t_{RDV}	(Note 12)				15	µs
Reset Time High	t_{RSTH}	(Note 12)		480			µs
Reset Time Low	t_{RSTL}	(Notes 12, 13)		480			µs
Presence-Detect High	t_{PDHIGH}	(Note 12)		15		60	µs
Presence-Detect Low	t_{PDLOW}	(Note 12)		60		240	µs
Capacitance	$C_{IN/OUT}$					25	pF

Note 12: See the timing diagrams in Figure 2.
Note 13: Under parasite power, if t_{RSTL} > 960µs, a power-on reset can occur.

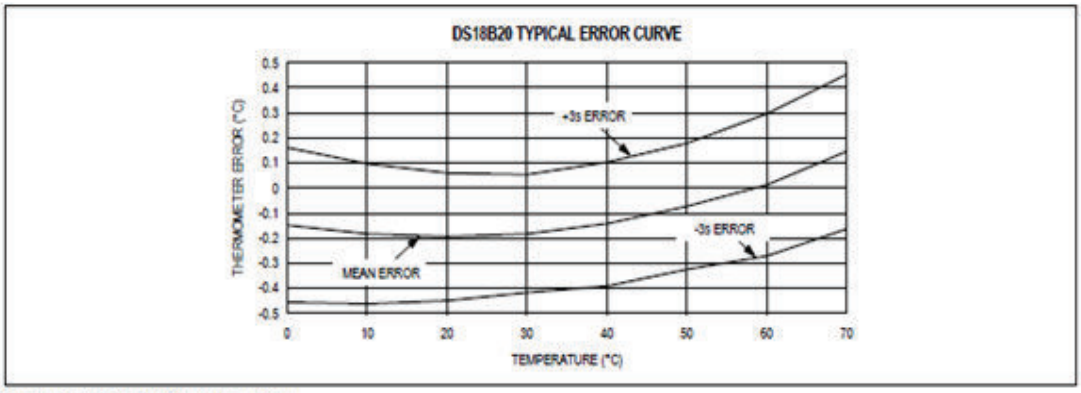


Figure 1. Typical Performance Curve

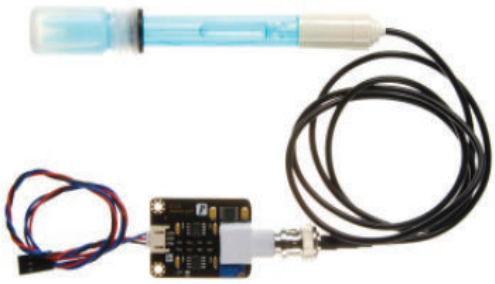
SENSOR DE PH meter (SKU: SEN0161)

PH meter(SKU: SEN0161)

From Robot Wiki

Contents

- 1 Introduction
- 2 Applications
- 3 Specification
- 4 pH Electrode Size
- 5 pH Electrode Characteristics
- 6 Use the pH Meter
  - 6.1 Connecting Diagram
  - 6.2 Step to Use the pH Meter
  - 6.3 Sample Code
- 7 Precautions
- 8 Documents



Analog pH Meter Kit

Introduction

Need to measure water quality and other parameters but haven't got any low cost pH meter? Find it difficult to use with Arduino?
Here comes an analog pH meter, specially designed for Arduino controllers and has built-in simple, convenient and practical connection and features. It has an LED which works as the Power Indicator, a BNC connector and PH2.0 sensor interface. To use it, just connect the pH sensor with BNC connector, and plug the PH2.0 interface into the analog input port of any Arduino controller. If pre-programmed, you will get the pH value easily. Comes in compact plastic box with foams for better mobile storage.
Attention:In order to ensure the accuracy of the pH probe, you need to use the standard solution to calibrate it regularly.Generally, the period is about half a year. If you meaure the dirty aqueous solution, you need to increase the frequency of calibration.

Applications

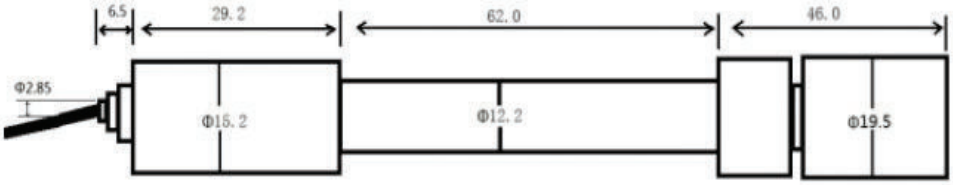
- Water quality testing
- Aquaculture

Specification

- Module Power : 5.00V
- Module Size : 43mm×32mm
- Measuring Range:0-14PH
- Measuring Temperature :0-60 ℃

- Accuracy : ± 0.1pH (25 ℃)
- Response Time : ≤ 1min
- pH Sensor with BNC Connector
- PH2.0 Interface (3 foot patch)
- Gain Adjustment Potentiometer
- Power Indicator LED
- Cable Length from sensor to BNC connector:660mm

pH Electrode Size



pH Electrode Characteristics

The output of pH electrode is Millivolts,and the pH value of the relationship is shown as follows (25 ℃):

VOLTAGE (mV)	pH value	VOLTAGE (mV)	pH value
414.12	0.00	-414.12	14.00
354.96	1.00	-354.96	13.00
295.80	2.00	-295.80	12.00
236.64	3.00	-236.64	11.00
177.48	4.00	-177.48	10.00
118.32	5.00	-118.32	9.00
59.16	6.00	-59.16	8.00
0.00	7.00	0.00	7.00

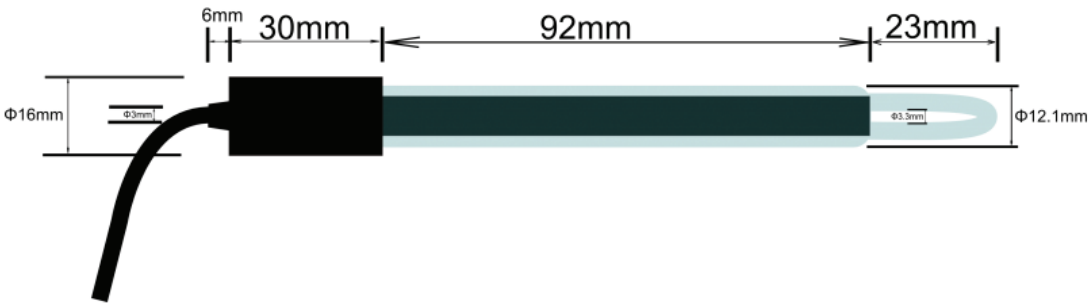
SENSOR DE CONDUCTIVIDAD ELÉCTRICA Meter SKU: DFR0300

Analog EC Meter SKU:DFR0300

Specification

- Operating Voltage: +5.00 V
- PCB Size: 45mm × 32mm
- Measuring Range: 1ms/cm--20ms/cm
- Operating Temperature :5-40 ℃
- Accuracy: <±10% F.S (specific accuracy depends on the accuracy of your calibration solution)
- PH2.0 Interface (3-pin SMD)
- Conductivity Electrode (Electrode Constant K = 1,BNC connector)
- Cable Length of the Electrode: about 60cm
- DS18B20 Temperature Sensor(Waterproof)
- Power Indicator

Electrode Size



Conductivity Electrode User Manual

1. **Application**

   This conductivity electrode is the measuring element of the conductivity meter. It used to measure the conductivity value of the aqueous solution or as a conductance titration.

2. **Model and Main Technical Parameters**

Model	DJS-0.1	260	DJS-1	DJS-10
Vessel Constant	0.1±0.02	0.6±0.2	1±0.2	10±2

3. **Use and Maintenance**

   1. There are two kinds of conductivity electrode, shiny electrode and platinum black electrode. platinum-plated black aims to increase the effective area of the electrode sheet and relieve being polarized. So in the measurement of large conductivity solutions, using a platinum black electrode is more appropriate.
   2. For the platinum black electrode, before use, it can be immersed in deionized water to prevent the platinum black inerting.

3. When platinum black coating off or fade, then you must re-plating platinum black to ensure the accuracy of the measurement readings.
4. The cell constant marked on the electrode is just for reference. If you want to test the cell constant is accurate or not, you can lookup the corresponding conductivity value K according to the concentration and temperature of the tested solution. Then the cell constant Q is calculated based on the measured resistance R .
   The formula is $Q = K \times R$.

See table below for details:

Model	Solution Concentration	Solution Temperature(℃)	Conductivity Value
DJS-0.1	0.001mol/L KCL	25	1.468×10^{-4}
		30	1.64×10^{-4}
260、DJS-1	0.01mol/L KCL	25	1.413×10^{-3}
		30	1.522×10^{-3}
DJS-10	0.1mol/L KCL	25	1.28×10^{-2}
		30	1.414×10^{-2}

4. Quality Warranty

1. The quality assurance of the electrode is the period for stored. It is about one year.
2. During the warranty period, if you find some problems caused by factories and not working, the manufacturer should be responsible for the repair or return.

TRANSISTOR 2N3904

2N3904

SMALL SIGNAL NPN TRANSISTOR

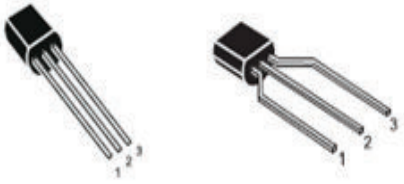
PRELIMINARY DATA

Ordering Code	Marking	Package / Shipment
2N3904	2N3904	TO-92 / Bulk
2N3904-AP	2N3904	TO-92 / Ammopack

- SILICON EPITAXIAL PLANAR NPN TRANSISTOR
- TO-92 PACKAGE SUITABLE FOR THROUGH-HOLE PCB ASSEMBLY
- THE PNP COMPLEMENTARY TYPE IS 2N3906

APPLICATIONS

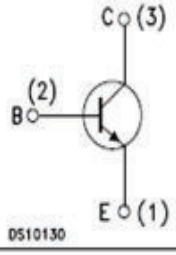
- WELL SUITABLE FOR TV AND HOME APPLIANCE EQUIPMENT
- SMALL LOAD SWITCH TRANSISTOR WITH HIGH GAIN AND LOW SATURATION VOLTAGE



TO-92 Bulk

TO-92 Ammopack

INTERNAL SCHEMATIC DIAGRAM



DS10130

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CBO}	Collector-Base Voltage (I_E = 0)	60	V
V_{CEO}	Collector-Emitter Voltage (I_B = 0)	40	V
V_{EBO}	Emitter-Base Voltage (I_C = 0)	6	V
I_C	Collector Current	200	mA
P_{tot}	Total Dissipation at T_C = 25 °C	625	mW
T_{stg}	Storage Temperature	-65 to 150	°C
T_j	Max. Operating Junction Temperature	150	°C

2N3904

THERMAL DATA

$R_{th\,j\text{-}amb}$ •	Thermal Resistance Junction-Ambient	Max	200	°C/W
$R_{th\,j\text{-}case}$ •	Thermal Resistance Junction-Case	Max	83.3	°C/W

ELECTRICAL CHARACTERISTICS (T_{case} = 25 °C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
I_{CEX}	Collector Cut-off Current (V_{BE} = -3 V)	V_{CE} = 30 V			50	nA
I_{BEX}	Base Cut-off Current (V_{BE} = -3 V)	V_{CE} = 30 V			50	nA
$V_{(BR)CEO}$*	Collector-Emitter Breakdown Voltage (I_B = 0)	I_C = 1 mA	40			V
$V_{(BR)CBO}$	Collector-Base Breakdown Voltage (I_E = 0)	I_C = 10 µA	60			V
$V_{(BR)EBO}$	Emitter-Base Breakdown Voltage (I_C = 0)	I_E = 10 µA	6			V
$V_{CE(sat)}$*	Collector-Emitter Saturation Voltage	I_C = 10 mA I_B = 1 mA I_C = 50 mA I_B = 5 mA			0.2 0.2	V V
$V_{BE(sat)}$*	Base-Emitter Saturation Voltage	I_C = 10 mA I_B = 1 mA I_C = 50 mA I_B = 5 mA	 0.65		0.85 0.95	V V
h_{FE}*	DC Current Gain	I_C = 0.1 mA V_{CE} = 1 V I_C = 1 mA V_{CE} = 1 V I_C = 10 mA V_{CE} = 1 V I_C = 50 mA V_{CE} = 1 V I_C = 100 mA V_{CE} = 1 V	60 80 100 60 30		 300 	
f_T	Transition Frequency	I_C = 10 mA V_{CE} = 20 V f = 100 MHz	250	270		MHz
C_{CBO}	Collector-Base Capacitance	I_E = 0 V_{CB} = 10 V f = 1 MHz		4		pF
C_{EBO}	Emitter-Base Capacitance	I_C = 0 V_{EB} = 0.5 V f = 1 MHz		18		pF
NF	Noise Figure	V_{CE} = 5 V I_C = 0.1 mA f = 10 Hz to 15.7 KHz R_G = 1 KΩ		5		dB
t_d t_r	Delay Time Rise Time	I_C = 10 mA I_B = 1 mA V_{CC} = 30 V			35 35	ns ns
t_s t_f	Storage Time Fall Time	I_C = 10 mA I_{B1} = -I_{B2} = 1 mA V_{CC} = 30 V			200 50	ns ns

* Pulsed: Pulse duration = 300 µs, duty cycle ≤ 2 %

RELAY SRD-05VDC-SL-C

SONGLE RELAY

松乐继电器 ® SONGLE RELAY	RELAY ISO9002	SRD

1. MAIN FEATURES

- Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.
- UL,CUL,TUV recognized.
- Selection of plastic material for high temperature and better chemical solution performance.
- Sealed types available.
- Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

- Domestic appliance, office machine, audio, equipment, automobile, etc.
  (Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

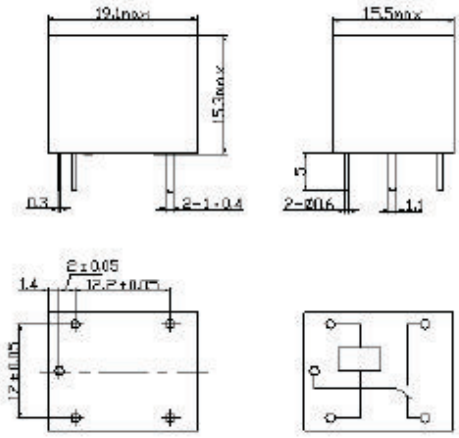
3. ORDERING INFORMATION

SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil sensitivity	Contact form
SRD	03、05、06、09、12、24、48VDC	S:Sealed type	L:0.36W	A:1 form A
				B:1 form B
		F:Flux free type	D:0.45W	C:1 form C

4. RATING

CCC	FILE NUMBER:CH0052885-2000	7A/240VDC
CCC	FILE NUMBER:CH0036746-99	10A/250VDC
UL/CUL	FILE NUMBER: E167996	10A/125VAC 28VDC
TUV	FILE NUMBER: R9933789	10A/240VAC 28VDC

5. DIMENSION(unit:mm) DRILLING(unit:mm) WIRING DIAGRAM



6. COIL DATA CHART (AT20°C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ±10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	120%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

7. CONTACT RATING

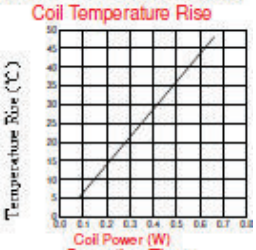
Item \ Type	SRD FORM C	SRD FORM A
Contact Capacity Resistive Load (cosΦ=1)	7A 28VDC 10A 125VAC 7A 240VAC	10A 28VDC 10A 240VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact Material	AgCdO	AgCdO

8. PERFORMANCE (at initial value)

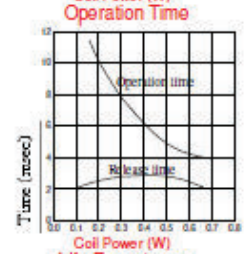
Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1 minute) 1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 300 operation/min 30 operation/min
Ambient Temperature	-25°C to +70°C
Operating Humidity	45 to 85% RH
Vibration Endurance Error Operation	 10 to 55Hz Double Amplitude 1.5mm 10 to 55Hz Double Amplitude 1.5mm
Shock Endurance Error Operation	 100G Min. 10G Min.
Life Expectancy Mechanically Electrically	 10^7 operations. Min. (no load) 10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

9.REFERENCE DATA

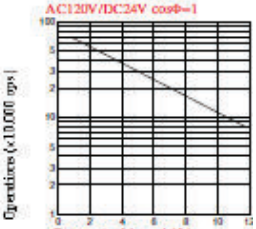
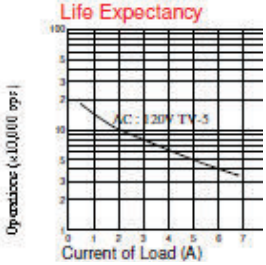
Coil Temperature Rise



Operation Time



Life Expectancy



Life Expectancy