INSTITUTO UNIVERSITARIO AERONÁUTICO

FACULTAD DE INGENIERÍA

TRABAJO FINAL DE GRADO

INGENIERÍA EN TELECOMUNICACIONES

“Sistema de Control y Monitoreo de Piscinas y Jardín Integrado a IoT”

AUTORES: ROTONDO, FEDERICO
VIDAL PIZARRO, IGNACIO ANTONIO

Tutor: Esp. Ing. José María Ducloux

Año 2017

Agradecimientos

En primer lugar, queremos agradecer a nuestro tutor José María Ducloux, por la ayuda y dedicación durante el desarrollo del trabajo final de grado.

También, dar las gracias a nuestros amigos y compañeros por apoyarnos durante los años de estudio.

Finalmente, agradecer a nuestras familias por el constante acompañamiento incondicional en nuestras vidas.

Resumen

Actualmente en la ciudad de Córdoba existen alrededor de 20.632 piletas en domicilios y natatorios, es decir, aproximadamente 24% más que hace cinco años. El gran número de piletas existentes y el incremento de las mismas en los últimos años son buenos indicativos para el desarrollo de una solución destinada al control y automatización del mantenimiento de las mismas utilizando nuevas tecnologías.

Como todo propietario sabe, mantener a punto el agua de la piscina, transparente y limpia exige realizar de forma periódica una serie de tratamientos físicos (limpieza y filtrado) y químicos (desinfección).

Este trabajo final de grado presenta el desarrollo de un sistema capaz de controlar y automatizar las funciones de mantenimiento e iluminación de piscinas y jardines, con la finalidad de disminuir el esfuerzo y tiempo necesarios para realizar estas tareas. El usuario mediante una aplicación móvil podrá verificar y controlar la temperatura del agua, la dosificación de los químicos, el accionamiento del motor de limpieza y la iluminación de la pileta y el patio de su hogar de manera local y remota, ya que el sistema estará conectado a Internet.

Para la realización de este proyecto se utilizan sistemas embebidos basados en microcontrolador y microprocesador, puesto que se deben adquirir los datos de los diversos sensores empleados, manejar los accesos de los usuarios de manera local y remota, y gestionar la base de datos del sistema.

Como resultado final de este trabajo, se construyó una maqueta a escala de una pileta y un patio, que a través de la aplicación móvil se pueden realizar las diversas tareas de mantenimiento e iluminación tal cual sucedería a escala real. En función de las pruebas realizas, las prestaciones del sistema son más que satisfactorias.

Palabras Claves

Android

Arduino

Domótica

Internet de las cosas (IoT)

MySQL

Piscina

Raspberry Pi

Sistemas embebidos

Contenidos

Índice de Figuras

Índice de Tablas

Índice de Códigos

1. Introducción

Desde el año 2012 se ha visto un incremento del 24% del número de piletas en la ciudad de Córdoba. Actualmente se estima que existen alrededor de 20.600 piscinas de natación en la capital cordobesa [1], [2].

En términos de mantenimiento de la pileta no sólo se trata de un tema estético, sino también de salubridad, ya que si no se mantiene limpia una piscina puede convertirse en un foco de infecciones. Para evitarlo, durante todo el año, se deben realizar algunas acciones que ayuden a conservarla limpia sin importar el uso que se le da o las condiciones climáticas según la estación.

A la hora de realizar las tareas de mantenimiento, actualmente se presentan dos principales inconvenientes. En primer lugar, realizar las tareas de limpieza uno mismo demanda tiempo y constancia por parte de los propietarios. En segundo lugar, muchas personas optan por contratar empresas que se encarguen del mantenimiento, generando un importante gasto económico al tener que abonar los servicios prestados con regularidad.

La Internet de las cosas (IoT, Internet of things) permite a los objetos físicos ver, oír, pensar y realizar tareas haciendo que “hablen” entre ellos, para compartir información y coordinar decisiones. IoT transforma estos objetos tradicionales a inteligentes, explotando su tecnología subyacente como los dispositivos embebidos, tecnologías de comunicación, redes de sensores, protocolos de Internet y aplicaciones [3].

Un sistema embebido es un sistema de computación diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real. Al contrario de lo que ocurre con las computadoras de propósito general (como por ejemplo una computadora personal o PC) que están diseñadas para cubrir un amplio rango de necesidades, los sistemas embebidos se diseñan para cubrir necesidades específicas [4]. Cuando se trabaja con IoT, se suele utilizar este tipo de sistemas ya que se

pueden programar para realizar una tarea concreta. Existen muchos sistemas embebidos aplicables a IoT que incluyen un procesador y diferentes tipos de interfaces de comunicación como Wi-Fi, LoRa, Bluetooth, Ethernet, USB, entre otras.

La alternativa que se propone para solucionar el problema del mantenimiento de piletas es combinar IoT con sistemas embebidos. Lo que se busca lograr es una pileta automatizada e inteligente, que logre llevar a cabo diversas tareas por si sola.

El resto del informe está compuesto de la siguiente manera. Se plantean primero los objetivos del proyecto, los cuales establecen el marco de trabajo. Luego se incluye un breve marco teórico de los temas más importantes que se tratan en el informe como son IoT, MySQL, Arduino, Raspberry Pi y los lenguajes de programación utilizados en el proyecto. Posteriormente, se detalla el desarrollo del trabajo propiamente dicho, en donde se presenta el diseño del sistema, su implementación, las respectivas pruebas y el montaje del mismo. Finalmente, se discuten los resultados obtenidos y se elaboran las conclusiones del trabajo final de grado.

2. Objetivos

2.1 Objetivo General

• Controlar y automatizar las funciones de mantenimiento e iluminación de piscinas y jardines, para facilitar al usuario la realización de estas tareas y brindarle nuevas medidas de seguridad.

2.2 Objetivos Específicos

• Desarrollar una aplicación para dispositivos móviles con sistema operativo Android para proveer al usuario una interfaz gráfica de control, configuración y estado del sistema.

• Controlar y dosificar los químicos de la piscina con la finalidad de ejecutar estas tareas de manera automática.

• Iluminar la pileta y jardín a ciertas horas del día de forma automática.

• Utilizar la iluminación LED de la piscina para advertir cuando no se puede ingresar a la misma por la reciente colocación de productos.

• Medir y controlar la temperatura de la piscina mediante el manejo de la calefacción de la misma.

• Controlar el motor de la piscina para la limpieza según el uso de la misma.

• Diseñar e implementar un sistema embebido para adquisición y envío de datos de los sensores a un servidor central de manera local.

• Implementar un servidor conectado a Internet que permita el acceso a los datos y realice el procesamiento de los mismos.

• Medir y controlar la humedad del suelo mediante la activación del sistema de riego de forma automática para mantener en óptimas condiciones el jardín y disminuir el consumo de agua.

3. Marco Teórico

3.1 Internet de las Cosas

3.1.1 Introducción

La IoT consiste en que las cosas tengan conexión a Internet en cualquier momento y lugar. En un sentido más técnico, consiste en la integración de sensores y dispositivos en objetos cotidianos que quedan conectados a Internet a través de redes fijas e inalámbricas. El hecho de que Internet esté presente al mismo tiempo en todas partes permite que la adopción masiva de esta tecnología sea más factible. Dado su tamaño y coste, los sensores son fácilmente integrables en hogares, entornos de trabajo y lugares públicos. De esta manera, cualquier objeto es susceptible de ser conectado en la Red. Además, el IoT implica que todo objeto puede ser una fuente de datos. Esto está empezando a transformar la forma de hacer negocios, la organización del sector público y el día a día de millones de personas.

Los objetos inteligentes junto con sus supuestas tareas constituyen aplicaciones específicas de dominio (mercados verticales), mientras que la computación ubicua y los servicios analíticos forman servicios independientes del dominio de la aplicación (mercados verticales). La Fig. 1 ilustra el concepto general de IoT en el que cada aplicación específica del dominio está interactuando con servicios independientes del dominio, mientras que en cada dominio los sensores se comunican directamente entre ellos [3], [9].

Con el tiempo, se espera que IoT tenga aplicaciones empresariales y hogareñas significativas, contribuyendo con la mejora en la calidad de vida y generando un crecimiento en la economía mundial.

Fig. 1: Esquema general de IoT[1]

[1] Imagen obtenida de: Andrea Zanella, N. B. (2014). *Internet of Things for Smart Cities.* IEEE INTERNET OF THINGS JOURNAL, VOL. 1

3.1.2 Elementos

En la siguiente sección se discuten los seis principales elementos necesarios para el funcionamiento de IoT, tal como se muestra en la Fig. 2.

A. Identificación: es crucial que IoT pueda nombrar servicios con su demanda. Abordar los objetos de IoT es fundamental para diferenciar entre ID de objeto y su dirección. ID de objeto se refiere a su nombre como “T1” para un sensor de temperatura y la dirección del objeto se refiere a su dirección en una red de comunicaciones. Los objetos dentro de la red pueden utilizar direcciones IP públicas y no privadas. Se utilizan métodos de identificación para proveer una identidad clara para cada objeto dentro de la red.

B. Detección: los medios de detección de IoT recogen datos de objetos relacionados dentro de la red y los envían a un almacén de datos o base de datos. La información recogida se analiza para realizar acciones basadas en servicios requeridos. Computadoras de placa única con sensores y funciones integradas de TCP/IP son las utilizadas normalmente para realizar productos IoT (Arduino, Yun, Raspberry Pi, etc.).

C. Comunicación: las técnicas de comunicación de IoT conectan objetos heterogéneos juntos para ofrecer servicios inteligentes específicos. Por ejemplo, algunos protocolos de comunicación utilizados son Bluetooth, Wi-Fi y LTE.

D. Computación: unidades de procesamiento (microcontroladores, microprocesadores, FPGAs) y aplicaciones de software representan el “cerebro” y la habilidad computacional del IoT. Diversas plataformas de hardware fueron desarrolladas para ejecutar aplicaciones IoT como Arduino, Galileo, Raspberry Pi, etc. Además, las plataformas de nube forman otra parte importante de la computación en IoT. Y proporcionan instalaciones para que objetos inteligentes envíen sus datos a la nube, para luego ser procesados en tiempo real.

E. Servicios: los servicios de IoT pueden ser catalogados en cuatro clases:

- *Servicios relacionados con la identidad:* son los más básicos e importantes. Cada aplicación que necesita traer objetos del mundo real al mundo virtual, debe identificar esos objetos.
- *Servicios de agregación de información:* recolectan y resumen mediciones sensoriales crudas que necesitan ser procesadas y reportadas a la aplicación IoT.
- *Servicios de conciencia colaborativa:* utilizan la información recolectada de los servicios de agregación de información para tomar decisiones y actuar en consecuencia.
- *Servicios ubicuos:* apuntan a proveer servicios de conciencia colaborativa en cualquier momento que sean necesarios y a cualquiera que los necesite en cualquier lugar.

F. Semántica: se refiere a la habilidad de extraer conocimiento inteligentemente con distintas máquinas para proporcionar los servicios requeridos. Esto incluye descubrir, utilizar recursos y modelar información. Además de reconocer y analizar los datos para proveer la decisión correcta para el servicio exacto. Así la semántica representa el cerebro de IoT enviando demandas al recurso correcto.

Fig. 2: Elementos IoT[2]

[2] Imagen obtenida de: Andrea Zanella, N. B. (2014). *Internet of Things for Smart Cities.* IEEE INTERNET OF THINGS JOURNAL, VOL. 1

3.2 MySQL

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Esta puede ser desde una simple lista de compras a una galería de pinturas o el vasto volumen de información en una red corporativa. Para agregar, acceder y procesar datos guardados en una computadora, se necesita un administrador como MySQL Server.

Dado que las computadoras son muy buenas manejando grandes cantidades de información, los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo, permitiendo velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

MySQL es software de código abierto, es decir, que es posible para cualquier persona usarlo y modificarlo. Por lo tanto, dicho código puede ser descargado y utilizado de forma gratuita. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades.

MySQL usa la licencia pública general de GNU (GNU GPL, GNU general public license) para definir qué puede hacer y qué no puede hacer con el software en diferentes situaciones.

3.3 Arduino

Arduino es una plataforma electrónica de código abierto basada en un método de uso fácil, tanto de hardware como de software, con el propósito de que cualquier persona pueda realizar proyectos interactivos.

Esta plataforma consiste en una placa que contiene en sí un microcontrolador con pines de entrada y salida pudiendo controlar motores, actuadores, luces y obtener lecturas

de sensores. Para enviarle órdenes y poder realizar diversas acciones, se puede utilizar un entorno de programación integrado propio de la marca conocido como “Arduino Software”.

Existen diversos modelos de estas placas, según la necesidad y utilidad se pueden tener en cuenta diversos factores como tamaño, cantidad de entradas/salidas, microcontrolador usado, memoria necesaria, consumo, entre otros aspectos que se pueden tener en consideración a la hora de elegir uno [5].

En este proyecto por cuestiones de consumo, número de entradas/salidas se optó por el modelo de nombre Mega 2560.

3.3.1 Arduino Mega 2560

La placa de desarrollo Arduino Mega tiene 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM), 16 entradas analógicas, 4 UARTs (puertos seriales por hardware), cristal oscilador de 16MHz, conexión USB, jack de alimentación, conector ICSP y botón de reset; tal como se ve en la Fig. 3. Arduino Mega es compatible con la mayoría de los shields diseñados para Arduino Duemilanove, Diecimila o UNO. A continuación, en la tabla 1, pueden apreciarse las especificaciones técnicas [5].

Las razones que convierten a Arduino en una plataforma interesante para desarrollar un proyecto de hardware y de software son:

- **Facilidad a la hora de programar:** el entorno de desarrollo integrado Arduino Software posee funciones preestablecidas que reducen la lógica y lectura de entradas, control de tiempos y salidas de manera semántica e intuitiva. Tiene la ventaja de no necesitar un tipo de tarjeta de programación como pasa con otros microcontroladores, sino que la placa se puede conectar con la computadora vía USB y se pueden cargar los programas sin riesgo de dañar la tarjeta debido a su protección adicional. El código es sumamente amigable y posee su propio lenguaje de alto nivel llamado Wiring, aunque esto no limita a Arduino a programarse en otros lenguajes.

- **Amplia variedad de documentación y tutoriales**: desde la misma página web de la empresa, el IDE viene con multitud de ejemplos, así como también los incontables tutoriales de YouTube.
- **Variedad de placas:** hay multitud de placas para cada necesidad del desarrollador. Entre ellas encontramos la MEGA, UNO, NANO, LEONARDO, PRO MINI, etc.
- **Gran cantidad de aplicaciones para desarrollar prácticamente todo**: se pueden mencionar como ejemplos de aplicación el envío y recepción de datos por Bluetooth, la robótica, la lectura de sensores, biomedicina y telemedicina.

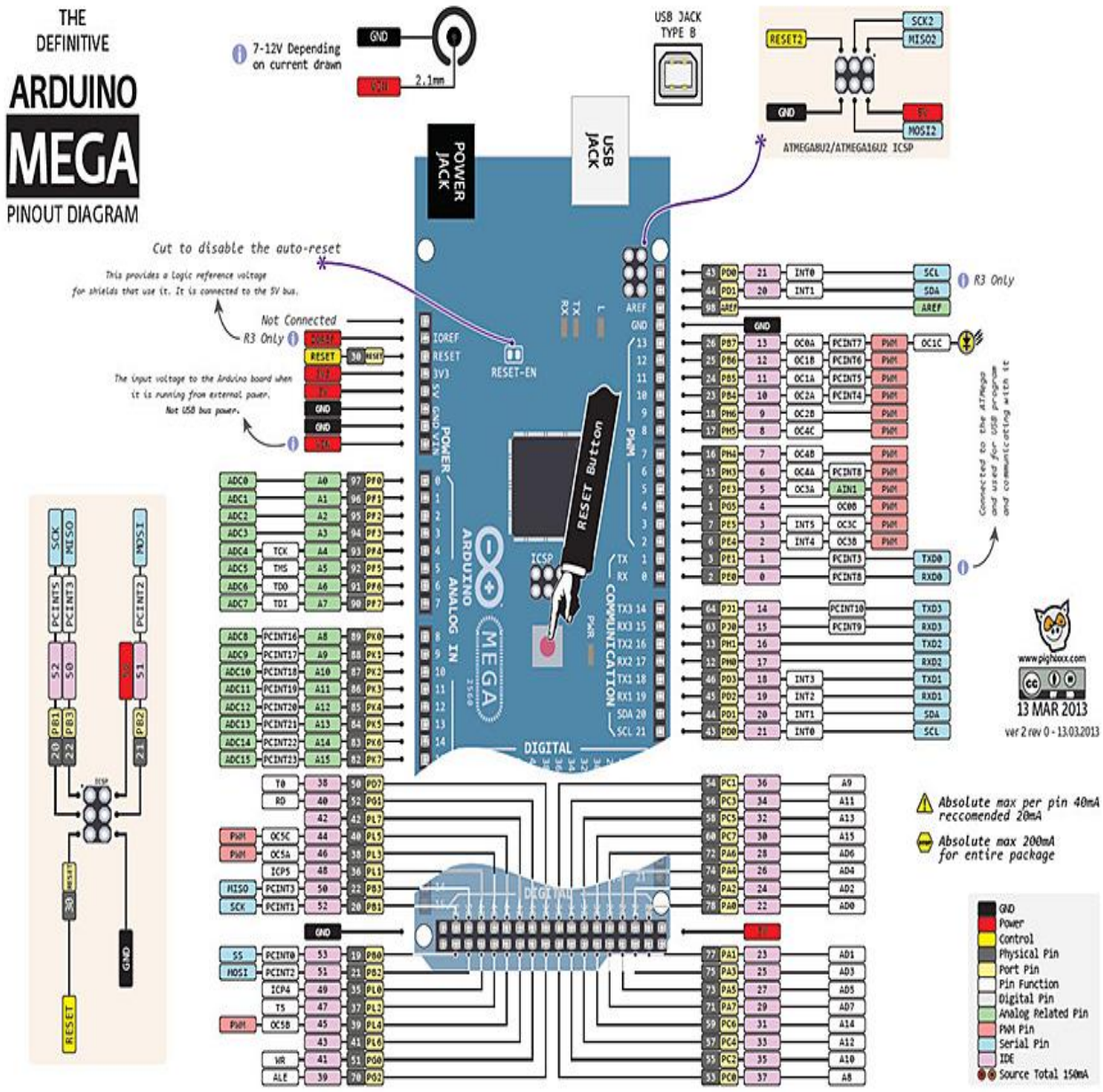


Fig. 3: Arduino Mega 2560[3]

[3] Imagen obtenida de: http://saber.patagoniatec.com/arduino-mega-2560-atmega-mega-arduino-clon-compatible-argentina-tutorial-basico-informacion-arduino-argentina-ptec/

Microcontrolador	ATmega 2560
Voltaje de alimentación de la placa	7-12V
Voltaje de operación del circuito	5V
Digital I/O Pins	54
PWM Pins	14
Analog Pins	16
Corriente DC por I/O Pin	40mA
Memoria Flash	256 kb (8 kb usados por el bootloader)
EEPROM	4 Kb
Velocidad reloj	16 Mhz

Tabla 1: Especificaciones técnicas Arduino Mega 2560

3.4 Raspberry Pi

Raspberry Pi es una computadora de placa reducida, computadora de placa única o computadora de placa simple (SBC, single board computer) de bajo costo desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Utiliza un software de código abierto, siendo su sistema operativo oficial una versión adaptada de Debian, denominada Raspbian, aunque permite usar otros sistemas operativos, incluido una versión de Windows 10. En todas sus versiones incluye un procesador Broadcom, una memoria RAM, una GPU, puertos USB, HDMI, 40 pines GPIO y un conector para cámara [6].

En este proyecto se optó por utilizar el modelo Raspberry Pi 3 por ser el modelo con mejores prestaciones y a su vez ser el único modelo que posee conectividad Wi-Fi integrada.

3.4.1 Raspberry Pi 3

Raspberry Pi 3 ensambla en su circuito un chipset *Broadcom BCM2387* con cuatro núcleos *ARM Cortex*-A53 a 1.2 GHz. La GPU encargada de los gráficos es la Broadcom VideoCore IV, una solución de dos núcleos compatible con OpenGL ES 2.0 y OpenVG que permite llegar a resoluciones Full HD. A diferencia de los anteriores modelos de Raspberry, en este modelo, se cuenta con conectividad Wi-Fi y Bluetooth integradas; como se muestra en la Fig. 4. A continuación, en la Tabla 2, se enumeran las características.

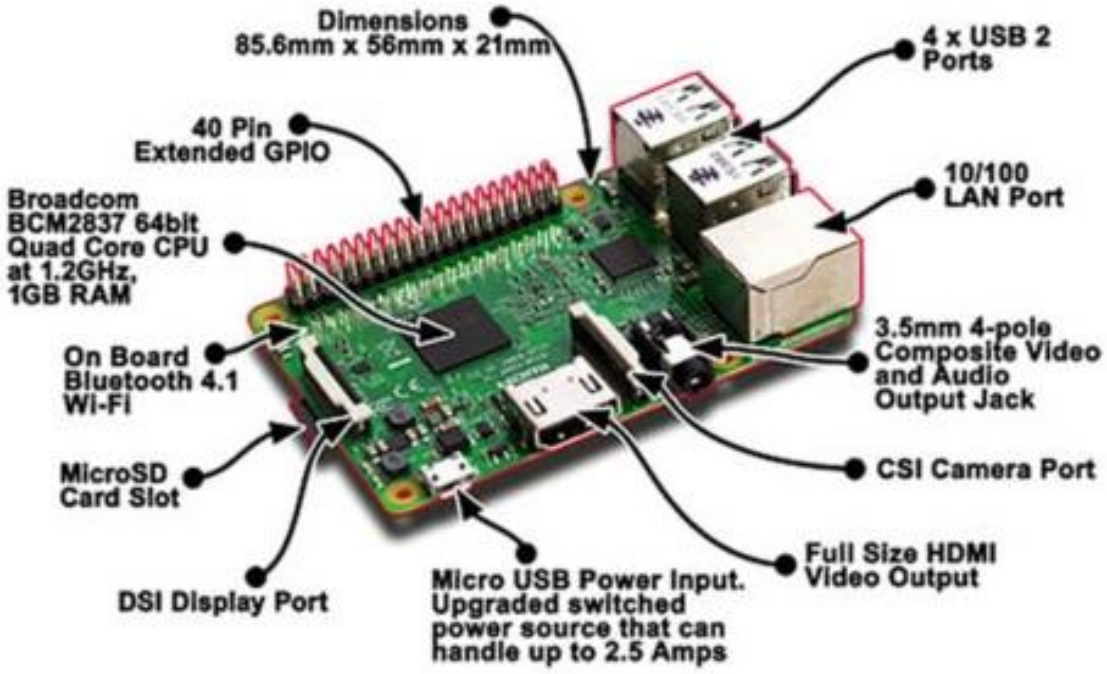


Fig. 4: Raspberry Pi 3[4]

[4] Imagen obtenida de: http://adictec.com/wp-content/uploads/2016/02/buy-raspberry-pi-3-modelB.png

Procesador	1,2 Ghz de 64 bits
RAM	1 GB
Pins GPIO	40
Salida de video	HDMI, RCA
Nucleo de Graficos	VideoCore IV 3D
Wireless LAN	802.11n
Bluetooth	4.1

Tabla 2: Especificaciones técnicas Raspberry Pi 3

3.5 Lenguajes de Programación

3.5.1 PHP

El lenguaje de preprocesdor de hipertexto PHP (PHP, PHP Hypertext Preprocessor) es un lenguaje de programación de uso general de código del lado del servidor, originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. Permite la conexión a diferentes tipos de servidores de bases de datos tanto SQL como NoSQL tales como MySQL, PostgreSQL, Oracle, Microsoft SQL Server, entre otros [7], [8].

3.5.2 Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma [7] [8].

Python ofrece mucho más chequeo de error que C, y siendo un lenguaje de muy alto nivel, tiene tipos de datos de alto nivel incorporados como arreglos de tamaño flexible y

diccionarios. Debido a sus tipos de datos más generales, Python puede aplicarse a un dominio de problemas mayor que AWK o incluso Perl, y aún así muchas cosas siguen siendo al menos igual de fácil en Python que en esos lenguajes.

3.5.3 Lenguaje C

Es un lenguaje orientado a la implementación de sistemas operativos, concretamente, Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel, ya que dispone de las estructuras típicas de los lenguajes de alto nivel, pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

4. Desarrollo

4.1 Diseño del Sistema

4.1.1 Solución Propuesta

Para llevar a cabo el sistema, se propone, integrar tres bloques principales. En primer lugar, una aplicación móvil para ejercer control del sistema (SO Android), luego, un servidor para recolectar la información y tomar decisiones sobre las acciones que deben ser llevadas a cabo (Raspberry Pi 3). Por último, un concentrador que recolecte determinada información de los sensores y accione distintos motores, según sea necesario (Arduino Mega 2560).

Los sensores (temperatura, humedad, nivel de líquidos) serán los encargados de la recolección de datos. Éstos viajarán al concentrador, el cual cumplirá las funciones de procesar y enviar los datos obtenidos vía USB al servidor, tal como se puede apreciar en la Fig. 5.

El servidor se encargará de procesar los datos permitiendo su visualización y la activación de las distintas funciones a través de una aplicación para dispositivos móviles. A su vez se implementará un control alternativo, independiente del dispositivo móvil, para poder operar el sistema, en los casos en que no funcione la red externa.

El hecho de disponer de la información en un servidor central a través de Internet para poder ser consultada representa una red de IoT. La posibilidad que ofrece IoT para que todos, personas y cosas, estén permanentemente conectados y se pueda recibir y procesar información en tiempo real, conduce a nuevos modos de toma de decisiones basados en esa disponibilidad de información. Estos cambios en los modelos de producción y consumo modifican las relaciones entre todos los agentes del sistema. Se abre así la oportunidad de diseñar y ofrecer un nuevo producto o servicios y explotar más eficientemente activos existentes, lo que crea un terreno fértil para emprender.

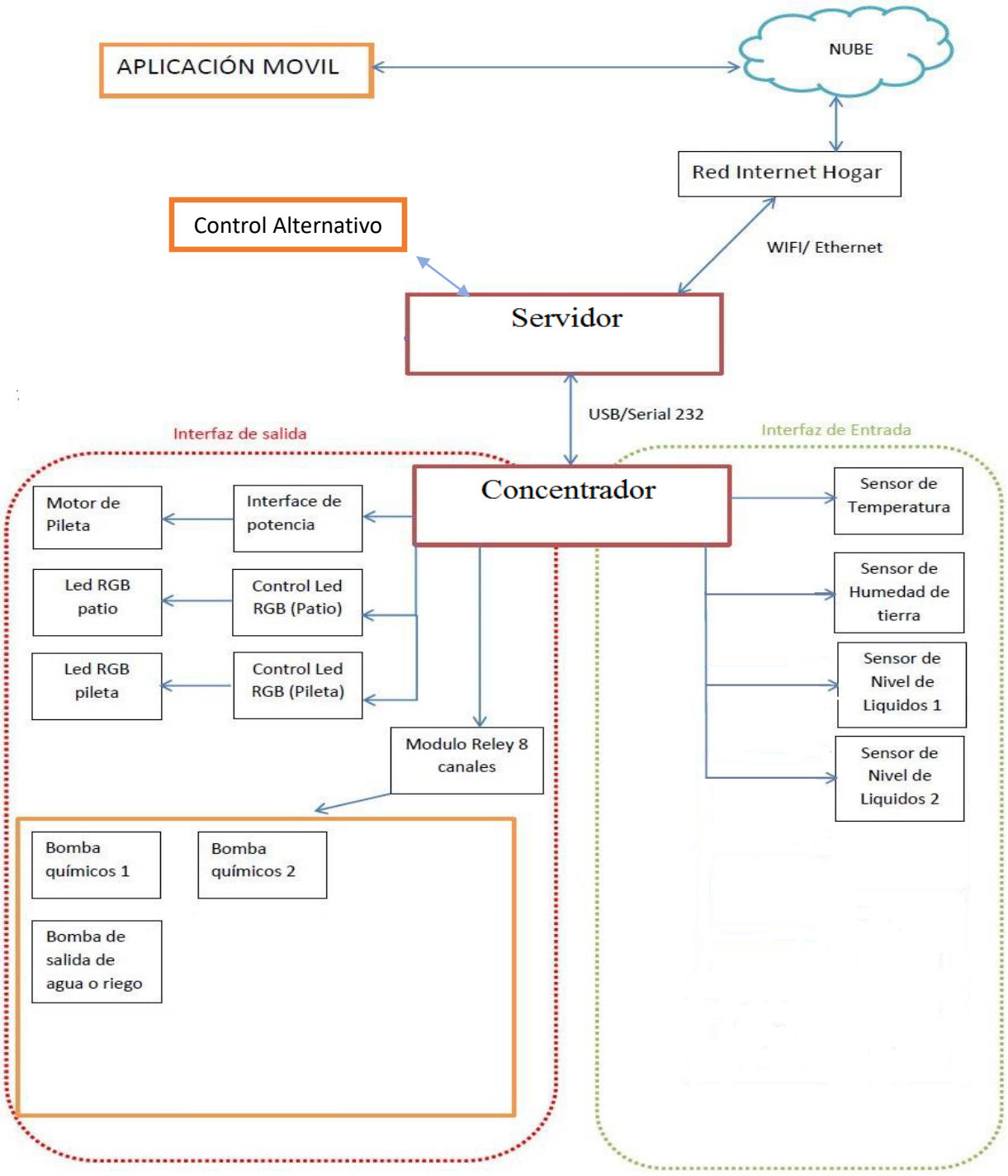


Fig. 5: Resumen técnico

4.1.2 Selección de Sensores y Periféricos

I. Sensor de Temperatura

Un Pt100 consiste en un alambre de platino que a 0 °C tiene 100 Ohms y que al aumentar la temperatura aumenta su resistencia eléctrica, es decir, existe una relación directa entre la temperatura y la resistencia eléctrica.

El incremento de la resistencia no es lineal, pero si creciente y característico del platino de tal forma que mediante tablas es posible encontrar la temperatura exacta a la que corresponde, tal como se muestra en la Fig. 6. Los Pt100 pueden fácilmente entregar precisiones de una décima de grado.

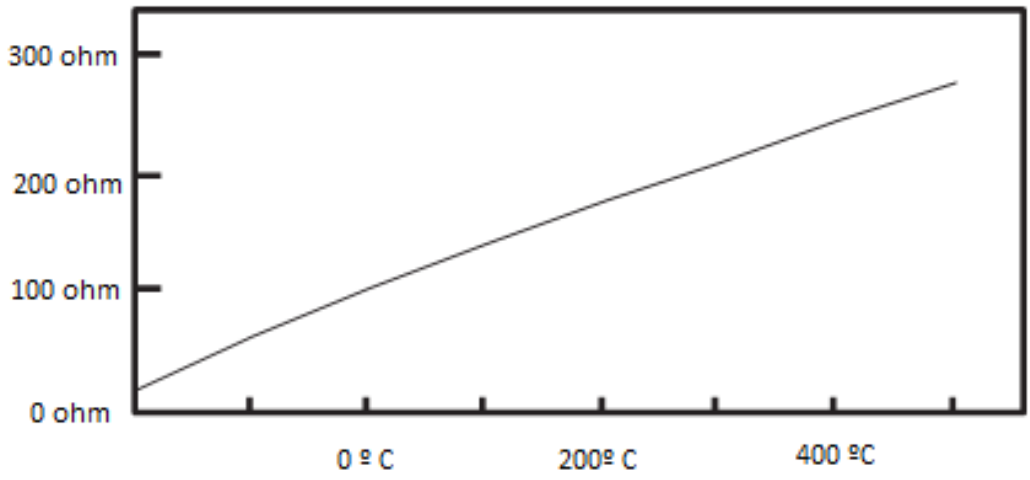


Fig. 6: Relación temperatura/resistencia del Pt100

El principal motivo por el que este termómetro fue el seleccionado es por ser una sonda sumergible, capaz de medir con precisión la temperatura del agua. A pesar de ser posible de medir un rango de temperaturas de -200 a 450 grados centígrados, esta característica no influyo en la decisión ya que se pretenden medir temperaturas de 10 a 30 grados centígrados aproximadamente.

Fig. 7: Termómetro Pt100[5]

II. Sensor de Humedad en Tierra

El módulo HL-69 es un sensor de humedad de suelo que utiliza la conductividad entre dos terminales para determinar ciertos parámetros relacionados a agua, líquidos y humedad.

Consiste en dos placas separadas entre sí por una distancia determinada. Ambas placas están recubiertas de una capa de material conductor. Si existe humedad en el suelo se creará un puente entre una punta y otra, lo que será detectado por un circuito de control con un amplificador operacional que será el encargado de transformar la conductividad registrada a un valor analógico que podrá ser leído por Arduino.

[5] Imagen obtenida de: http://www.prometec.net/sensor-temperatura/

Fig. 8: Sensor humedad HL-69[6]

El sensor de humedad posee la ventaja de ser económico, a su vez de ser simple de implementar. Estos fueron algunas de las características que llevaron a utilizarlo.

III. Sensor Ultrasónico

El sensor HC-SR04 es un módulo que incorpora un par de transductores de ultrasonido que se utilizan de manera conjunta para detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 450 cm.

La interfaz digital se logra mediante 2 pines digitales: el pin de trigger (disparo) y echo (eco).

Fig. 9: Sensor ultrasonido HC.SR04[7]

Fue seleccionado específicamente para este proyecto para medir el nivel de líquido en los tanques que contengan químicos. Al colocar este sensor en la parte superior de los tanques, se soluciona el problema de tener que sumergir algún dispositivo medidor de nivel en líquidos corrosivos.

[6] Imagen obtenida de: http://www.prometec.net/sensor-humedad/

[7] Imagen obtenida de: http://www.prometec.net/sensor-distancia/

IV. Sistema de Relés

Un relé es un interruptor que se puede activar mediante una señal eléctrica. Normalmente se utiliza cuando se requiere conmutar grandes picos de tensión o intensidad como por ejemplo arrancando motores de corriente alterna de una cierta potencia.

En este proyecto los sistemas de relé son utilizados para alimentar los motores y leds, de 12 voltios, cuando se desee que entren en funcionamiento.

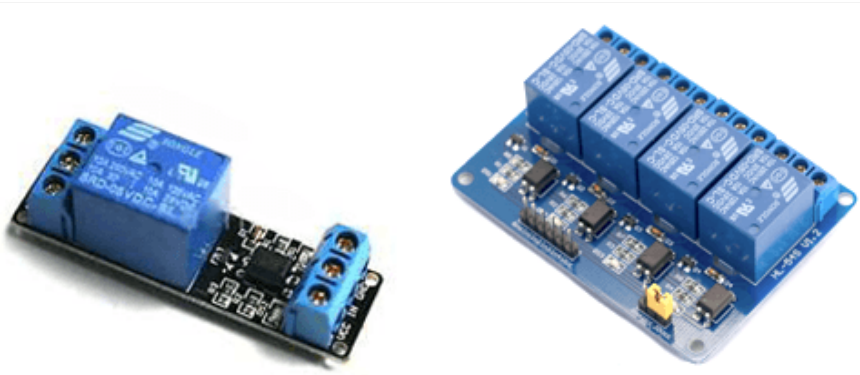


Fig. 10: Módulos de relés[8]

V. Pantalla LCD

Se utiliza una pantalla LCD para Arduino de 2 x 16 con caracteres blancos y luz de fondo azul. Cuenta con un potenciómetro para ajustar el contraste de la pantalla.

La pantalla se activará únicamente para la utilización del control alternativo a la aplicación móvil.

Fig. 11: Pantalla LCD[9]

[8] Imagen obtenida de: http://www.prometec.net/reles/

VI. Control IR

Utilizando un receptor infrarrojo y un control (emisor) se puede programar para realizar distintas acciones como controlar luces, motores, etc. Es utilizado puntualmente en el proyecto para realizar un control alternativo del sistema independiente de la aplicación móvil.

Fig. 12: Control remoto y sensor infrarrojo [10]

VII. Bomba de agua 12V

Al alimentar la bomba de agua con 12 voltios, comienza a funcionar. Se suele colocar un interruptor para poder controlar cuando se quiere que esté en funcionamiento o no.

En el caso de este trabajo final se utilizaran cuatro motores en total, dos para llevar los químicos hacia la pileta, uno para simular un filtro, y por ultimo uno para el sistema de riego del patio.

Se optó por este modelo en particular, ya que no es necesaria una alimentación de 220V, proporcionando seguridad para el operador del sistema.

[9] Imagen obtenida de: http://nosinmiarduino.blogspot.com.ar/2014/11/manejo-de-un-display

[10] Imagen obtenida de: http://saber.patagoniatec.com/control-remoto-infrarrojo-arduino-argentina-ptec/

Fig. 13: Bomba de agua 12 voltios[11]

VIII. LEDs

Son diodos emisores de luz, alimentados con 12 voltios capaces de variar el color que se demuestra, según se lo configure.

Para este trabajo final se utilizan los LEDs para iluminar la pileta, y para simular que la calefacción está encendida, al iluminar con color rojo.

Fig. 14: Tira de LEDs[12]

[11] Imagen obtenida de: http://articulo.mercadolibre.com.mx/MLM-561293844-motor-para-deposito-agua aguarisas-gm-chevy-original-_JM

[12] Imagen obtenida de: http://www.ledyluz.com/tiras-led-aluminio/1387-tira-led-rigida-luz-blanca-calida.html

4.1.3 Diseño del Concentrador de Datos

En el caso de este trabajo final de grado se utilizará la placa Mega2560 para cumplir la función de concentrador. Según el pulsador esté encendido o apagado, la placa Arduino funcionara de distinto modo. En caso de que esté apagado, funcionara en el modo automático. Tal como se ve en la Fig. 15, en este modo, el concentrador espera ordenes de recolección de datos o de ejecución de tareas de la placa Raspberry Pi. En cambio, en el caso que se encuentre seleccionado el modo manual, Arduino recibe órdenes del control IR, mostrando la información en la pantalla LCD.

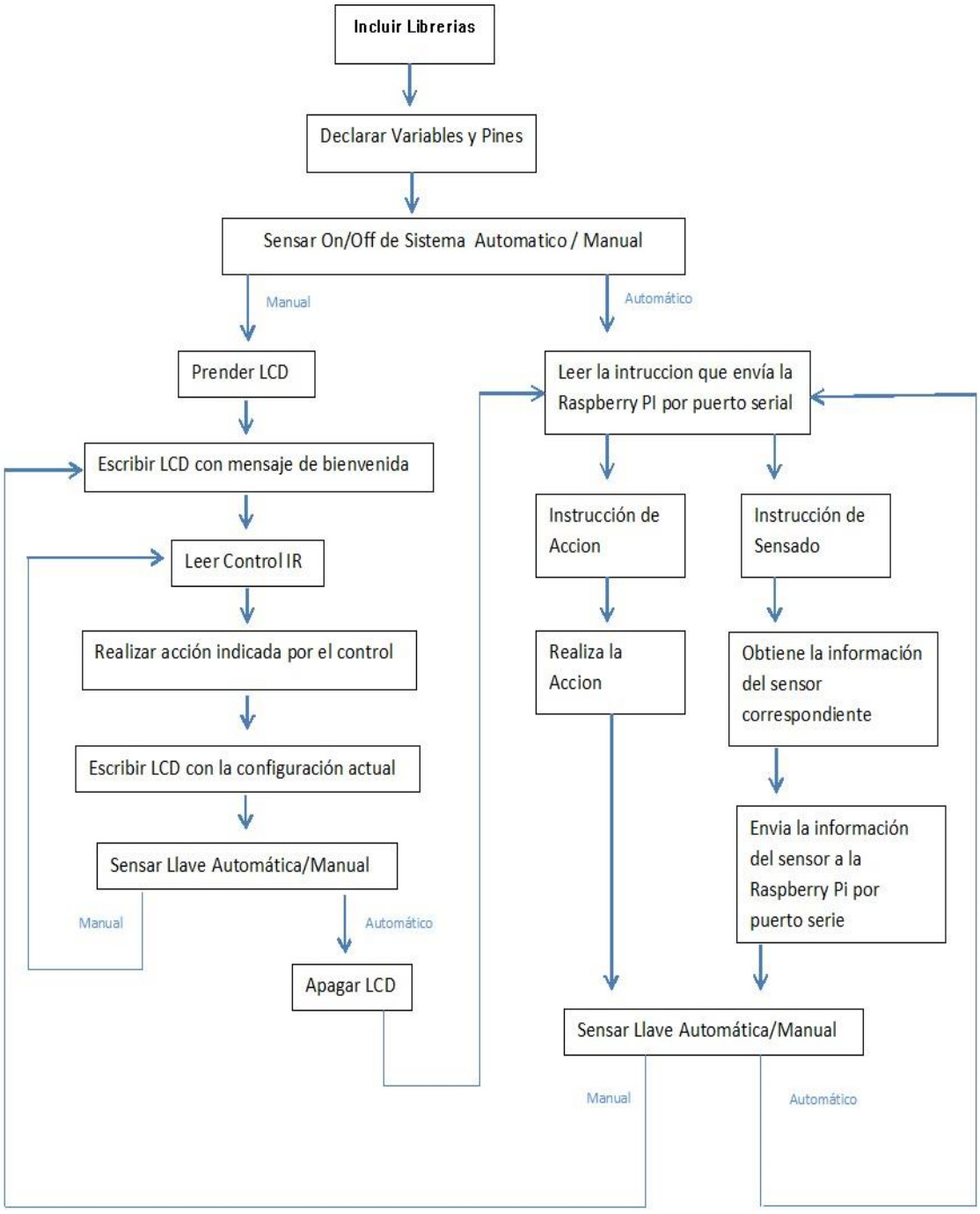


Fig 15: Diagrama de flujo Arduino

4.1.4 Diseño del Servidor

En el servidor (Raspberry Pi), que es la placa principal de todo el sistema, está alojada la base de datos MySQL, los programas PHP y el programa principal del sistema. Además, es la encargada de enviarle las instrucciones a la placa Arduino para accionarla y adquirir los datos desde los sensores.

El programa principal de todo el sistema tiene el diagrama de flujo de la Fig. 16. Luego de incluir las librerías y declarar la base de datos, ingresa en un bucle infinito en donde primero envía instrucciones vía puerto serial a la placa Arduino. Una vez adquiridos los valores actuales de los sensores, se envían estos datos a la base de datos para almacenarse en la tabla Sensores. Luego, descarga los valores de la tabla Aplicación en la base de datos, en donde se encuentra el modo seleccionado por el usuario en la aplicación y actúa según esto.

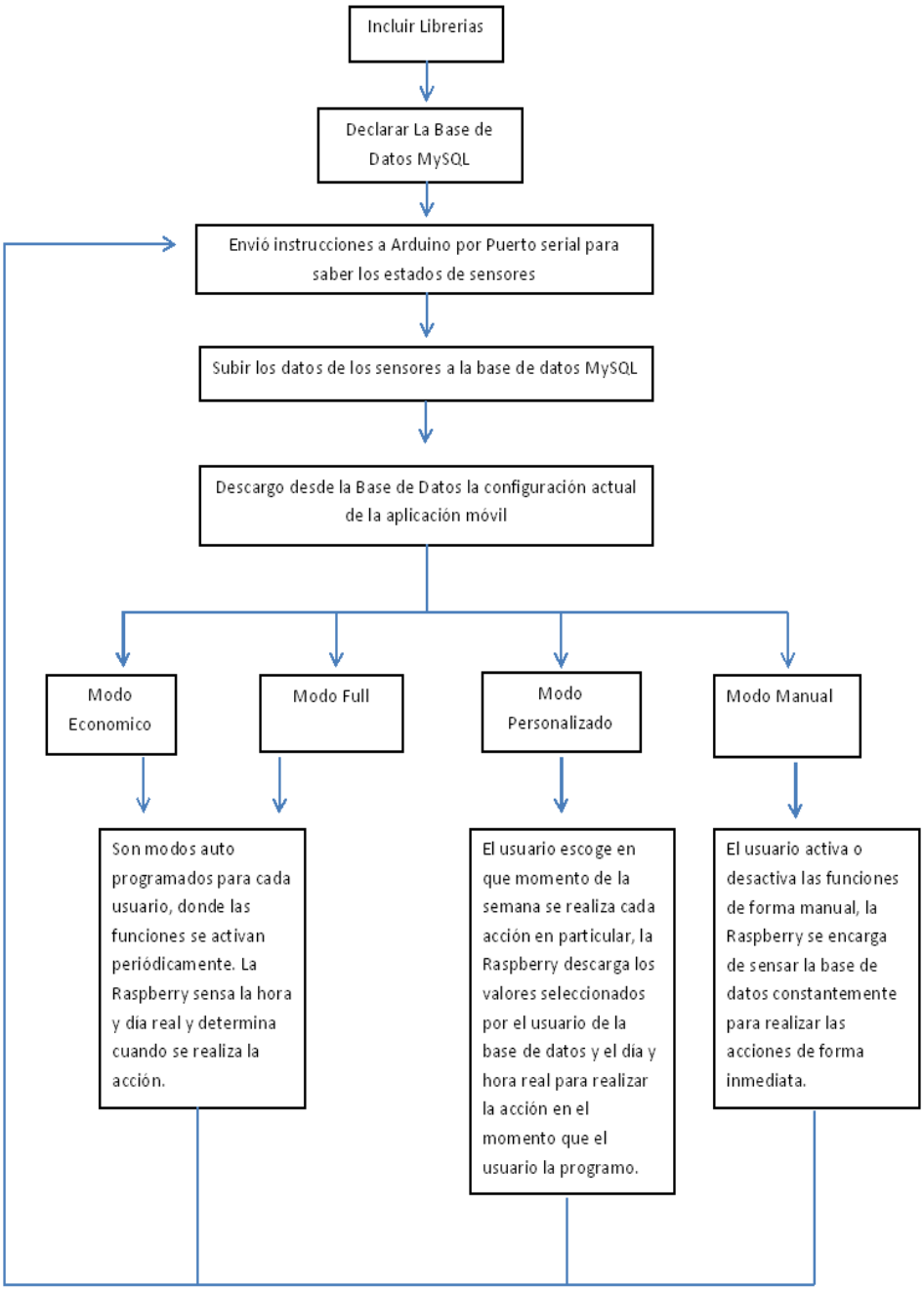


Fig. 16: Diagrama de flujo servidor

4.1.5 Diseño de la Aplicación Móvil

El objetivo del proyecto es poder controlar y administrar las funciones de la pileta desde una aplicación móvil. Esta contará con 4 modalidades de funcionamiento, de las cuales dos son pre-programadas (económico y full), una es programable (personalizada) y la última es de acción inmediata de las distintas acciones disponibles.

La función principal de dicha aplicación es enviar y recibir datos desde la base de datos MySQL ubicada en la Raspberry Pi y que, además, el usuario la pueda utilizar de manera fácil e intuitiva, ya que el proyecto está dirigido a personas no especialistas en el tema.

Existen dos modos de funcionamiento automáticos pre-programados, el modo Económico y el modo Full. El primero fue pensado para ser seleccionado en épocas que la pileta no es utilizada con demasiada frecuencia (otoño e invierno), ya que realizara las tareas de mantenimiento en periodos más espaciados y utilizando los mínimos recursos. En cambio, el modo Full, fue pensado para las épocas en la que la pileta es utilizada frecuentemente (primavera y verano), ya que se realizan tareas de mantenimiento a diario, y se emplean la mayor cantidad de recursos para que esta se encuentre en condiciones óptimas de uso.

Finalmente, seleccionando el modo Personalizado, el usuario podrá programar los días y frecuencia con los que quiere que se realicen las tareas de funcionamiento. En cambio, a través del modo Manual, se selecciona el encendido o apagado de las funciones en forma de acción inmediata.

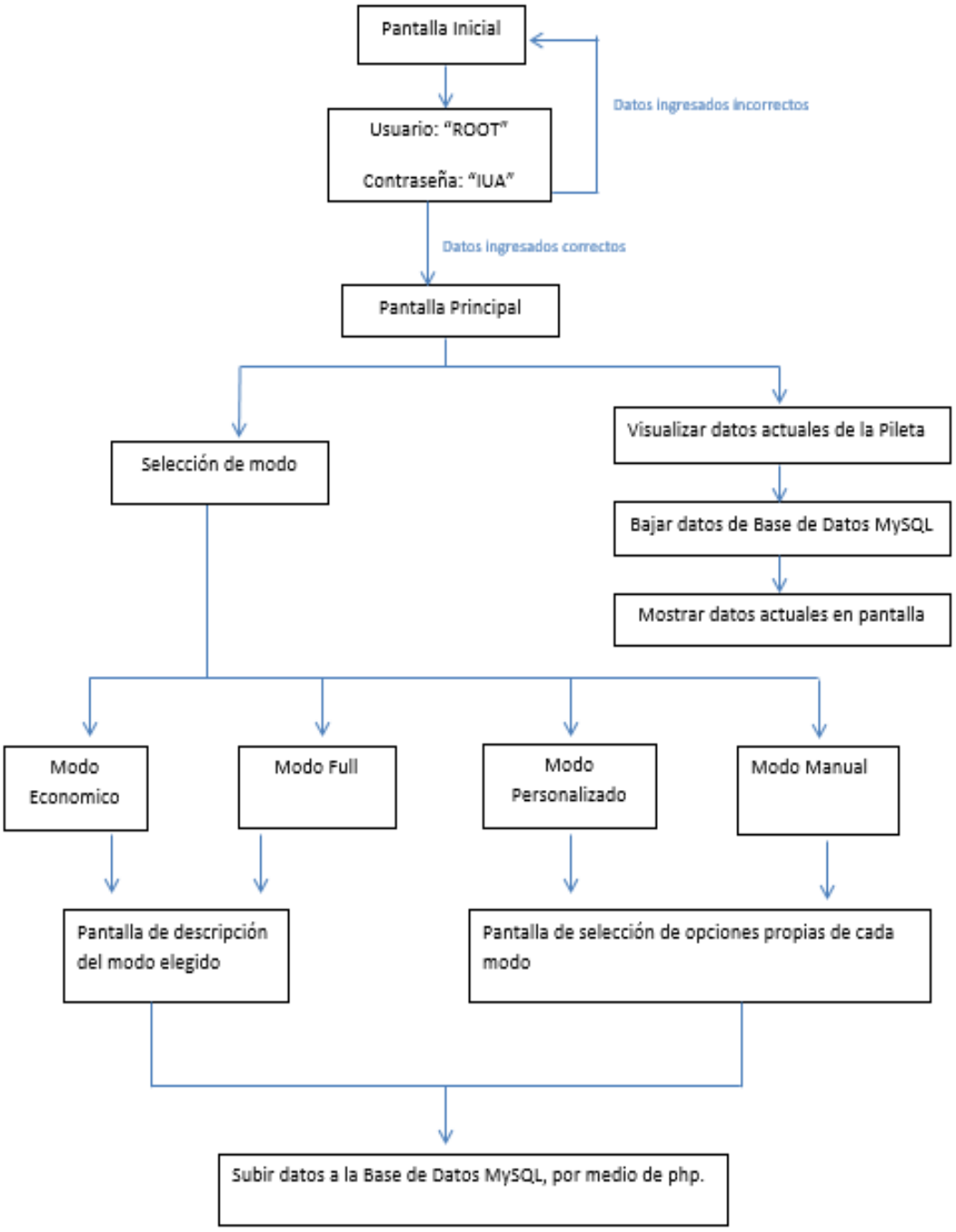


Fig. 17: Diagrama de flujo aplicación móvil

5. Implementación

5.1 Concentrador de Datos

5.1.1 Configuración del IDE de Arduino

En primer lugar, lo que se debe hacer cuando se desarrolla un programa en Arduino es indicar que placa se irá a utilizar, ya que sin esto luego no se puede compilar y cargar el código en la placa Arduino.

En el caso de este trabajo final de grado se utilizará la placa Mega2560, por lo que se realiza lo siguiente: Herramientas Placa Arduino Mega o Mega 2560.

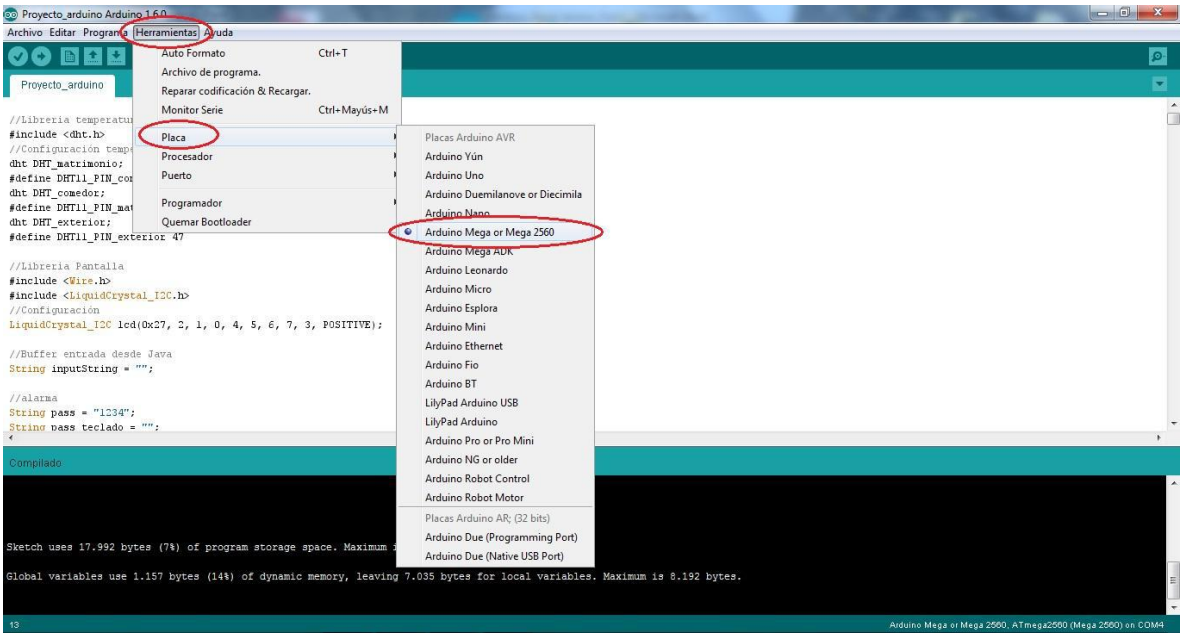


Fig. 18: Selección placa Arduino

5.1.2 Importación de clases

Como ya ocurre con otros lenguajes, en Arduino también se incluyen las librerías que se vayan a utilizar al inicio del programa.

Para importar la clase que necesaria, se debe añadir al **workspace** de Arduino donde se está trabajando, que por defecto se guarda en la siguiente ruta cuando se utiliza Windows: *C:\Users\usuario\Documents\Arduino\libraries*

Desde la interfaz de Arduino se puede añadir la librería entrando a Programa -> Importar Librería -> Añadir librería.

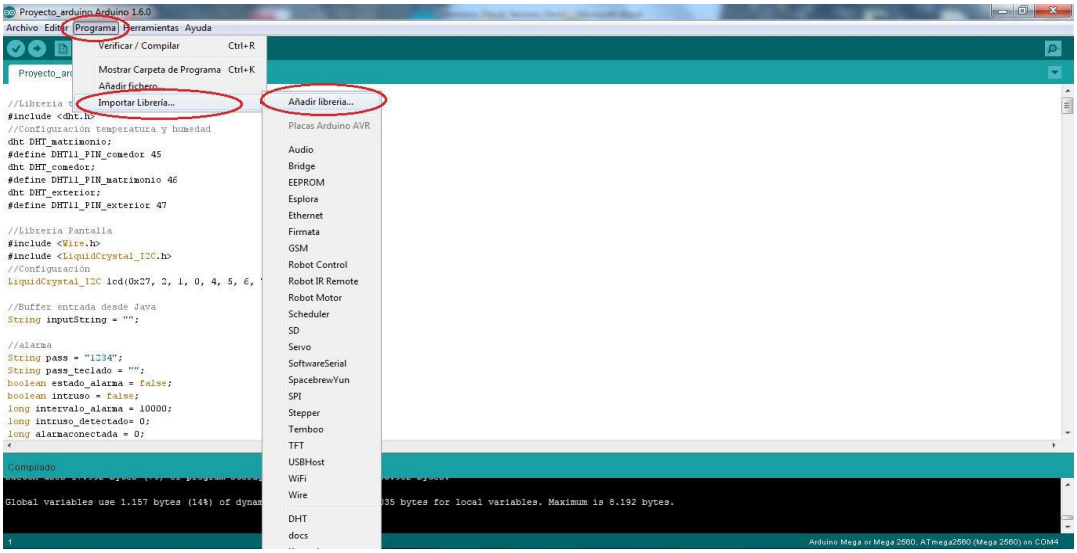


Fig. 19: Añadir librería en Arduino

Por último, en el programa solo se deben añadir los nombres de las librerías. En el caso de este programa se utilizaron las siguientes librerías:

- Pantalla LCD:

```
#include<Wire.h>
#include<LiquidCrystal_I2C.h>
```

- Control IR

```
#include <IRremote.h>
```

5.1.3 Configuración Inicial

Luego de declarar las variables que serán utilizadas, se deben inicializar las funciones respectivas de cada sensor, y configurar los pines en donde fueron conectados. A continuación, en la Tabla 3, se muestra cada sensor y el pin en el que fue conectado respectivamente.

Numero de Pin	Función del pin	Analógico/Digital	Entrada/Salida
A0	Sensor de Temperatura	Analógico	Entrada
A1	Sensor de Humedad	Analógico	Entrada
2	Sensor de Control IR	Digital	Entrada
4	On/Off Sistema Independiente	Digital	Entrada
SDA	Pin SDA LCD	Digital	Entrada
SCL	Pin SCL LCD	Digital	Entrada
46	Pin Trigger Sensor Ultrasónico 1	Digital	Entrada
47	Pin Echo Sensor Ultrasónico 1	Digital	Entrada
48	Pin Trigger Sensor Ultrasónico 2	Digital	Entrada
49	Pin Echo Sensor Ultrasónico 2	Digital	Entrada
30	On/Off Bomba de Riego	Digital	Salida
32	On/Off Bomba Limpia fondo	Digital	Salida
34	On/Off Bomba Cloro	Digital	Salida
36	On/Off Bomba Químicos	Digital	Salida
22	On/Off Luces Jardín	Digital	Salida
24	On/Off Luces Pileta	Digital	Salida
26	On/Off Calefacción	Digital	Salida

Tabla 3: Distribución pines Arduino

5.1.4 Programa Principal

El programa desarrollado para este trabajo final de grado se puede dividir en dos bloques, como ya se ha mencionado. En primer lugar, si se activa la llave, se controla todo de forma manual utilizando el control IR. En cambio, al no activar el pulsador, el programa funciona de forma automática.

Programa Manual

Cuando este modo de funcionamiento es seleccionado, utilizando el control IR, se pueden encender o apagar distintos componentes. Como la bomba de cloro, bomba de químicos, motor de filtro, calefacción, luces y riego.

Los componentes son programados simplemente para que actúen, cuando se los comanda, en un estado HIGH (se encienden) o en estado LOW (se los apaga). Cuando se genera un cambio de estado, en la pantalla LCD se imprime el cambio seleccionado. Por ejemplo, para el caso de las luces se utiliza lo mostrado en el Cód. 1.

```
1         if (luces == 0)
2            {
3              lcd.setCursor(32, 0);
4              lcd.print("LUCES OFF");
5            }
6         if (luces == 1)
7            {
8              lcd.setCursor(32, 0);
9              lcd.print("LUCES ON ");
10            }
11        if (irrecv.decode(&results)) {
12             IR = results.value, HEX;
13             switch (IR)
14           case 16769055:
15              Serial.println("LUCES ON ");
16              digitalWrite(lucespin, HIGH);
17              luces = 1;
18              break;
19            case 16754775:
20               Serial.println("LUCES OFF");
21               luces = 0;
22               digitalWrite(lucespin, LOW);
23              break;
```

Cód. 1: Sección del programa que controla el estado de las luces

Tal como se muestra en el ejemplo anterior, fueron configurados el resto de los componentes activos. En un principio, luego que se activa la llave, se chequea el estado actual en el que se encuentra cada componente respectivo y se imprime en la pantalla LCD.

Finalmente, en un segundo momento, se configura que tecla específica se debe accionar para encender o apagar el elemento.

Programa automático

Cuando este modo de funcionamiento es seleccionado, la placa Arduino espera instrucciones, que son enviadas vía puerto serial desde la placa Raspberry Pi 3. Estas órdenes dependen de la configuración que se elija desde la aplicación móvil.

La única diferencia con respecto al modo manual, es que los sensores (temperatura, ultrasonido y humedad) recolectan y envían información a la placa Raspberry Pi 3. Pero al igual que el modo anterior, espera a que le comanden si enciende o apaga los componentes como los motores o las luces.

5.2 Base de Datos

phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. En esta se puede crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, y exportar datos en varios formatos.

Para acceder a configurar la base de datos hay que escribir en el navegador la siguiente dirección*: http://ipdispositivo/phpmyadmin*, donde se encontrará la interfaz gráfica que se muestra en la Fig. 18.

Fig. 20: Interfaz web de phpMyAdmin

Esta interfaz presenta como ventaja ser muy intuitiva a la hora de crear base de datos y tablas. En el caso de este trabajo final de grado, se creó una base de datos llamada TESIS, tal como se muestra en la Fig. 21. Dentro de ésta, existen 4 tablas distintas (modo, personalizada, manual, sensores) para poder cargar los datos de forma más ordenada.

Fig. 21: Base de datos TESIS

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	ID	int(11)			No	Ninguna	AUTO_INCREMENT	Cambiar Eliminar Primaria Único Más
2	Motor	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
3	Cloro	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
4	Quimicos	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
5	Alarma	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
6	Luces	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
7	Calefaccion	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
8	Calefaccion01	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más

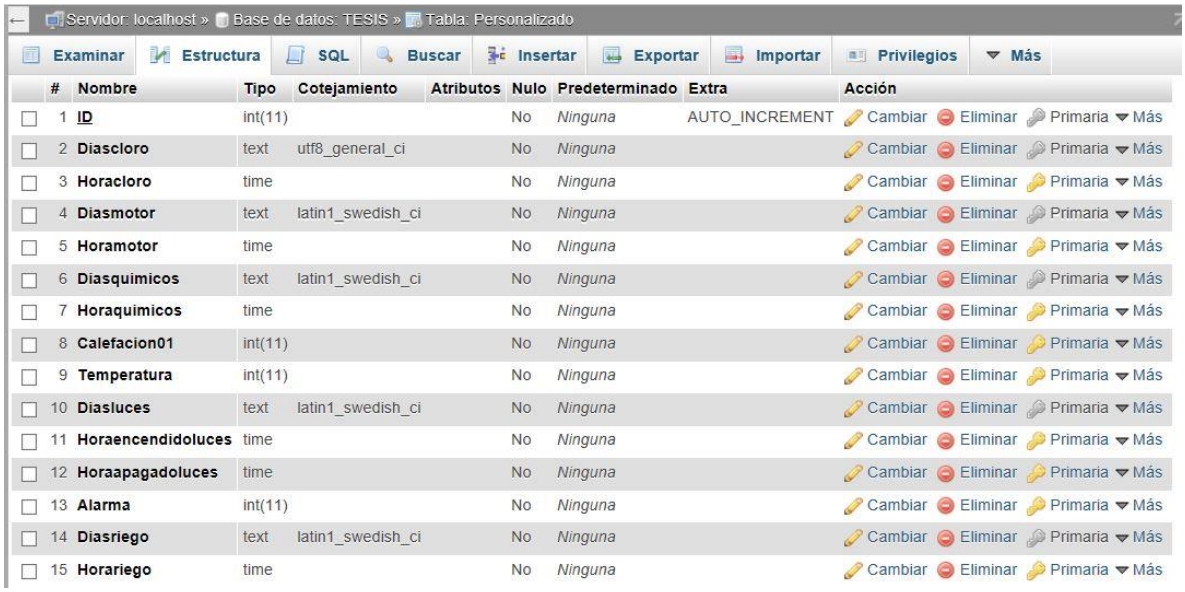
Fig. 22: Tabla manual

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	ID	int(11)			No	Ninguna	AUTO_INCREMENT	Cambiar Eliminar Primaria Único Índice Más
2	Modo	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Índice Más

Fig. 23: Tabla aplicación

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	ID	int(11)			No	Ninguna	AUTO_INCREMENT	Cambiar Eliminar Primaria Único Más
2	Temperatura	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
3	Humedad	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
4	Liquido1	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
5	Liquido2	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más
6	Alarma	int(11)			No	Ninguna		Cambiar Eliminar Primaria Único Más

Fig. 24: Tabla sensores



Servidor: localhost » Base de datos: TESIS » Tabla: Personalizado

Examinar | Estructura | SQL | Buscar | Insertar | Exportar | Importar | Privilegios | Más

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
1	ID	int(11)			No	*Ninguna*	AUTO_INCREMENT	Cambiar Eliminar Primaria Más
2	Diascloro	text	utf8_general_ci		No	*Ninguna*		Cambiar Eliminar Primaria Más
3	Horacloro	time			No	*Ninguna*		Cambiar Eliminar Primaria Más
4	Diasmotor	text	latin1_swedish_ci		No	*Ninguna*		Cambiar Eliminar Primaria Más
5	Horamotor	time			No	*Ninguna*		Cambiar Eliminar Primaria Más
6	Diasquimicos	text	latin1_swedish_ci		No	*Ninguna*		Cambiar Eliminar Primaria Más
7	Horaquimicos	time			No	*Ninguna*		Cambiar Eliminar Primaria Más
8	Calefacion01	int(11)			No	*Ninguna*		Cambiar Eliminar Primaria Más
9	Temperatura	int(11)			No	*Ninguna*		Cambiar Eliminar Primaria Más
10	Diasluces	text	latin1_swedish_ci		No	*Ninguna*		Cambiar Eliminar Primaria Más
11	Horaencendidoluces	time			No	*Ninguna*		Cambiar Eliminar Primaria Más
12	Horaapagadoluces	time			No	*Ninguna*		Cambiar Eliminar Primaria Más
13	Alarma	int(11)			No	*Ninguna*		Cambiar Eliminar Primaria Más
14	Diasriego	text	latin1_swedish_ci		No	*Ninguna*		Cambiar Eliminar Primaria Más
15	Horariego	time			No	*Ninguna*		Cambiar Eliminar Primaria Más

Fig. 25: Tabla personalizado

5.3 Aplicación Móvil

Para la implementación de la aplicación móvil se utilizo MIT App Inventor. App inventor es un framework creado inicialmente por el MIT (Instituto Tecnológico de *Massachusetts)* y adquirido por Google para que cualquier persona con interés pueda crearse su propia aplicación móvil, ya sea para su empresa, para su casa o por otros intereses.

Fig. 26: Logotipo de App Invetor

La principal ventaja por la cual se eligió App Inventor fue por su facilidad a la hora de programar, ya que no es necesario conocer un lenguaje de programación específico como en la mayoría de los casos.

La programación en esta plataforma se realiza por bloques, donde cada uno de ellos cuenta con una función preestablecida y son encastrados de forma tal que adquieren una estructura consistente con lo que se desea obtener. Los bloques presentan formas y colores distintivos para poder facilitar su utilización y no conectar funciones que no tienen una lógica establecida.

Como desventaja se puede mencionar que la programación en bloque representa un problema cuando la aplicación comienza a tomar un tamaño considerable, los bloques se empiezan a volver voluminosos, difíciles de manejar y muchas veces difíciles de encontrar.

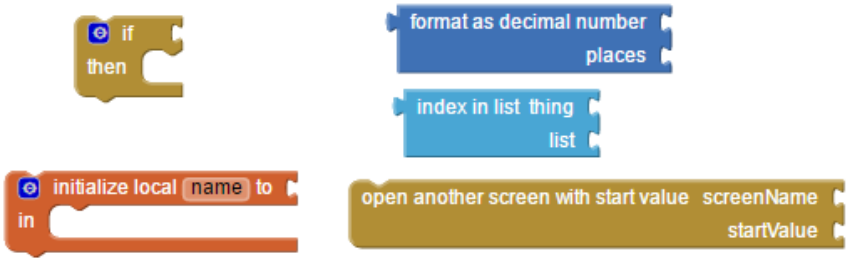


Fig. 27: Bloques de App Inventor

En este entorno se tienen dos partes, la primera es la pantalla de diseño, en donde todos los componentes para hacer que aplicación luzca como interesa. En la segunda el editor de bloques, que básicamente es la encargada de darle una función a cada una de las partes de la aplicación y hacerla funcional.

Como se observa en la Fig. 26 ver la interfaz de diseño, en ella se encuentran diferentes partes cada una con una función:

1. Paleta. En ella todas las funciones que puede tener nuestra aplicación ya sean botones, deslizadores, archivos de texto, conectividad, etc.

2. Visor. En el una pantalla similar a la de un celular. Es en donde iremos ingresando los elementos para darle forma a la aplicación.

3. Componentes. En esta lista se detallan todos los elementos con los que la aplicación cuenta.

4. Media. Todos los archivos del tipo media (imágenes, audios, videos) que estén cargados en la aplicación aparecerán en esta lista.

5. Propiedades. Esta sección permite editar las características de los elementos utilizados con la finalidad de personalizar según las necesidades. [10]

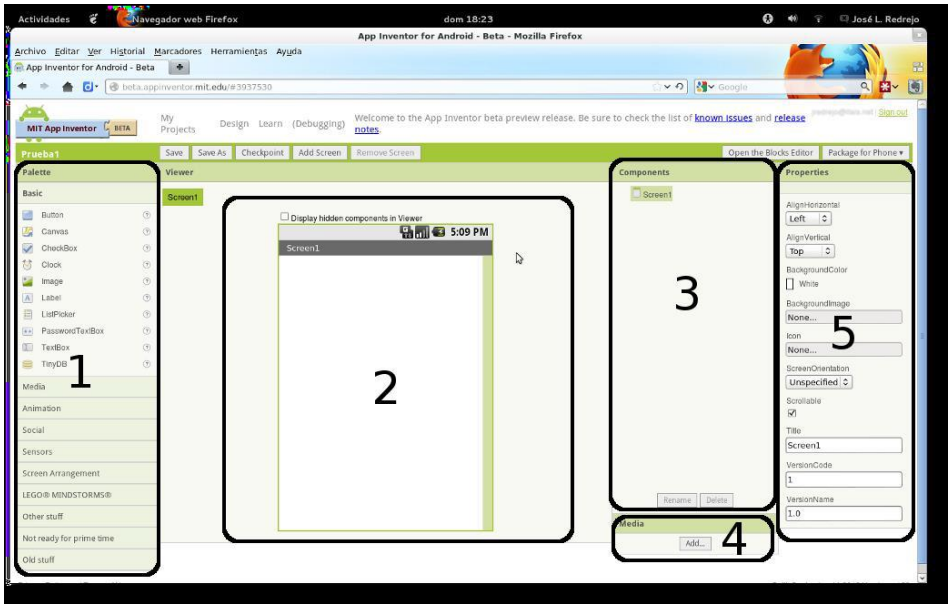


Fig. 28: Interfaz de diseño App Inventor

Esta aplicación cuenta con una pantalla para que el usuario ingrese a la configuración en donde se pide un usuario y contraseña y así la aplicación autentifica al cliente y su ingreso a la base de datos Fig 29.

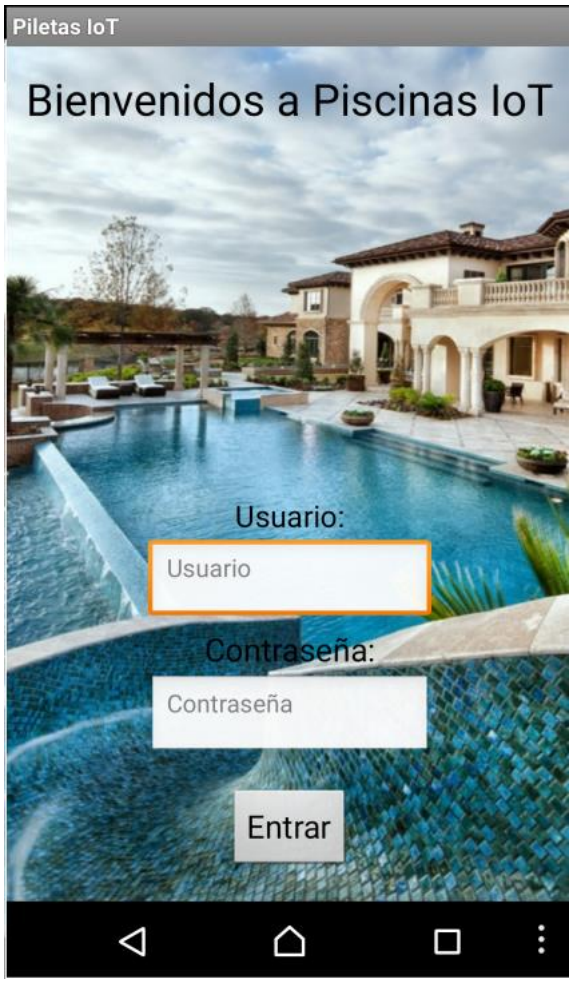


Fig. 29: Pantalla de ingreso

Luego el usuario se encuentra con una pantalla en donde puede intuitivamente seleccionar el modo en el que quiere que su pileta funcione, y además acceder a cada modo de forma individual y puede ingresar a ver los valores actuales de la pileta como se muestra en la Fig 30.

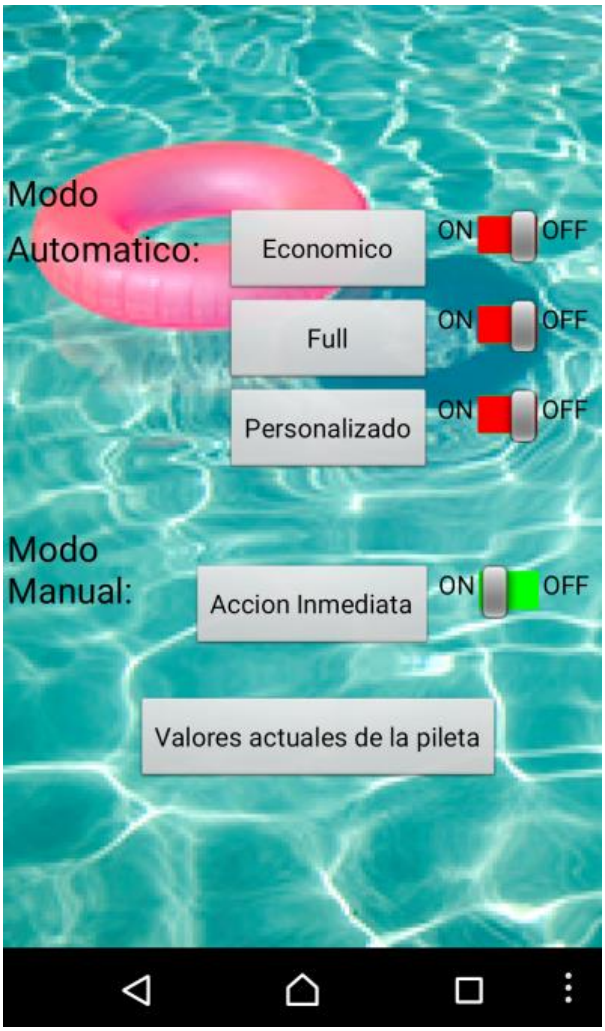


Fig. 30: Pantalla de inicio

Cumpliendo con el diseño propuesto previamente, en la Fig. 31, se pueden observar los modos Económico y Full pre-programados.

Fig. 31*: Características modo económico*

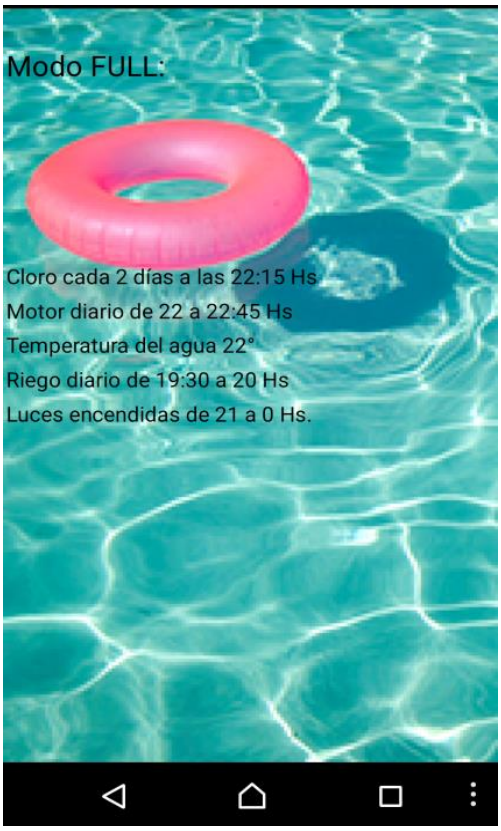


Fig. 32: Características modo full

En cambio, en el modo Personalizado el usuario podrá elegir qué día y a qué hora va a realizar la acción deseada, como se demuestra en las Fig. 33 y 34.

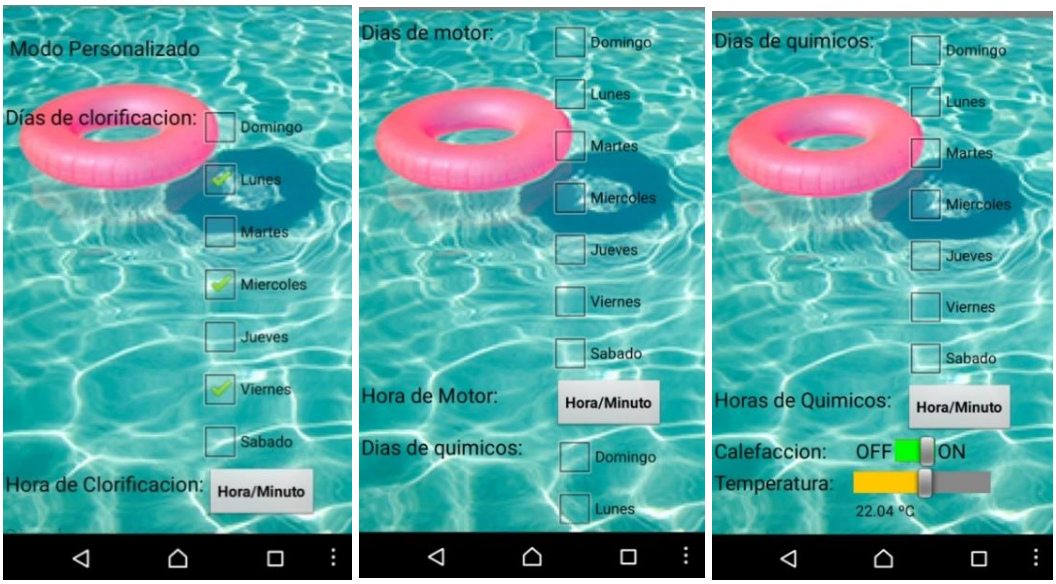


Fig. 33: Pantalla modo personalizado

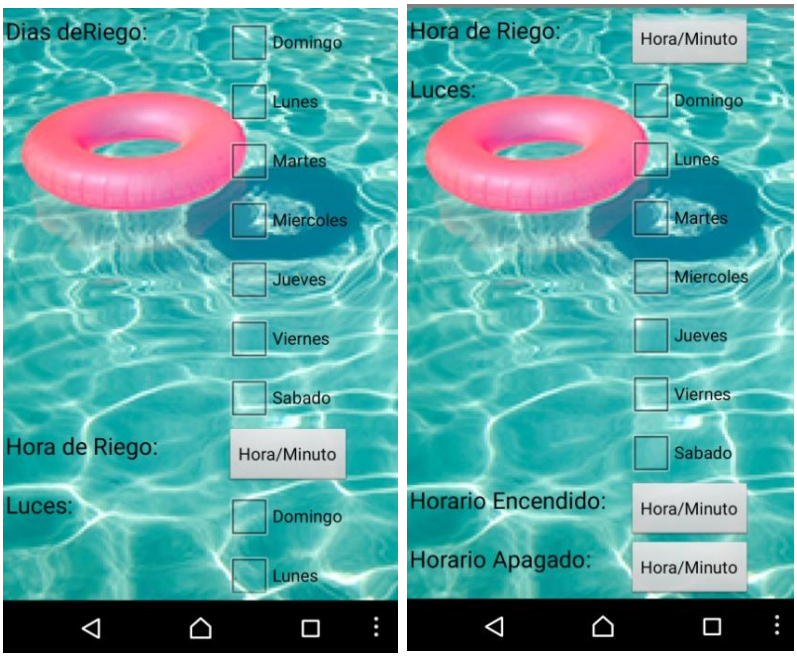


Fig. 34: Pantalla modo personalizado

Por otro lado, un modo manual, de acción inmediata que sin ninguna configuración previa se pueden accionar los sistemas como se muestra en la Fig 35.

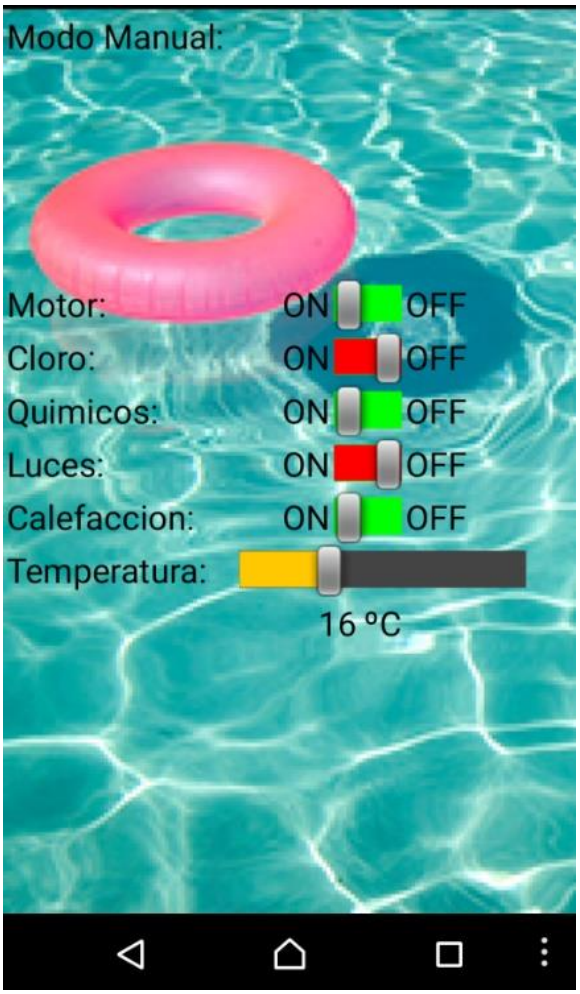


Fig. 35: Pantalla modo manual

Y por último existe una pantalla en donde se pueden observar los valores actuales de la pileta como se muestra en la Fig 36.

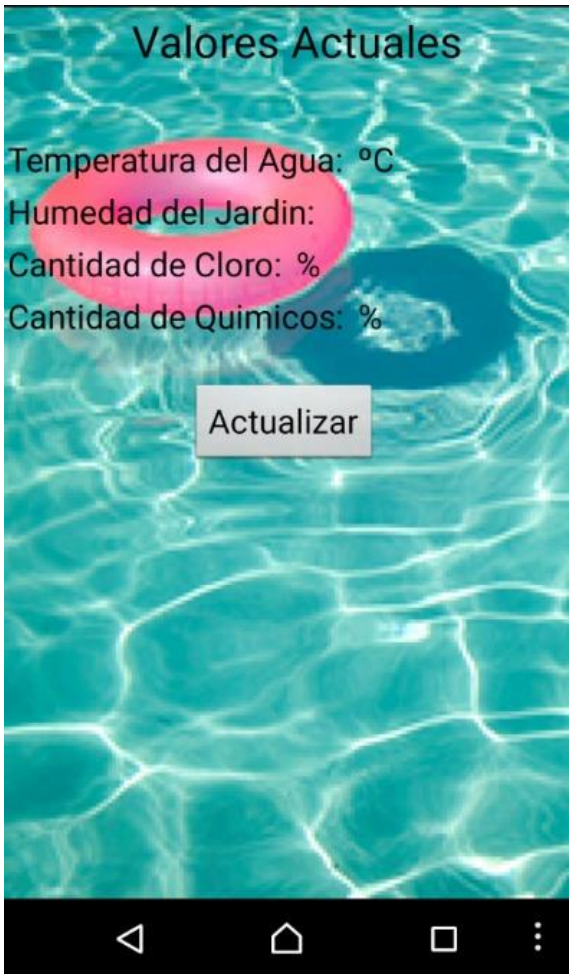


Fig. 36: Pantalla valores actuales

5.4 Servidor PHP + Base de Datos

Se convierte a la Raspbery Pi en un servidor, que soporte PHP, para lograr crear un panel de control para el usuario.

En la Fig. 37 se puede ver un diagrama del funcionamiento del servidor web.

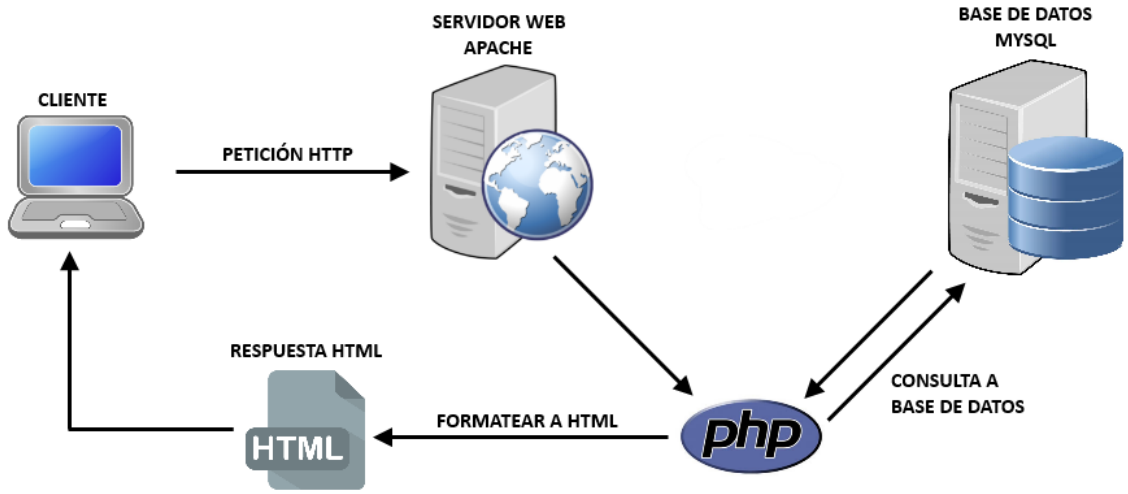


Fig. 37: Funcionamiento servidor web[13]

En el caso de este trabajo final de grado, se crearon distintos programas PHP, con el propósito de leer los valores de la base de datos desde la aplicación móvil y viceversa, es decir, guardar los valores seleccionados desde la aplicación en la base de datos.

Carga Valores a Base de Datos

Fueron creados tres programas distintos para cargar los datos que se envían de la aplicación móvil a las distintas tablas que requieran parámetros modificables por el usuario (modo, personalización, manual).

A continuación, en el Cód. 2, se muestra el código que carga los datos a la tabla "modo".

```
1        <?php
2        $host= "localhost";
3        $user= "root";
4        $pass= "iua";
5        $base= "TESIS";
6        $con= mysqli_connect($host, $user, $pass, $base);
7        if (!$con) {
```

[13] Imagen obtenida de: http://blog.segu-info.com.ar/2015/07/cracking-rc4-en-wpa-tkip-y-tls-en-52.html

```
8       die("conexion fallida" . mysqli_connect_error());
9       }
10      echo "conexion exitosa";
11      $mod = $_GET['Modo'];
12      mysqli_query($con, "INSERT INTO Aplicacion (Modo)VALUES('$mod')");
13      echo "datos insertados";
14      ?>
```

Cód. 2: Cargador de valores modo a base de datos

Como se puede ver en el Cód. 2, una vez que se realiza la conexión exitosa con la base de datos, visto en la línea 10, se inserta el valor “Modo”, que se recolecta de la aplicación móvil, a la tabla llamada “Aplicación”, tal como se ve en la línea 12.

Lectura de Valores en Base de Datos

Se desarrollaron cuatro programas PHP distintos, para leer los datos actuales de los sensores que recolectan información de la placa Arduino (temperatura, nivel de líquidos 1, nivel de líquidos 2 y humedad). Estos valores son interpretados por la placa Raspberry Pi 3 y mostrados en la aplicación móvil.

A continuación, en el Cód. 3, se muestra el código que lee los datos del sensor de temperatura.

```
1       <?php
2       $host= "localhost";
3       $user= "root";
4       $pass= "iua";
5       $base= "TESIS";
6       $con= mysqli_connect($host, $user, $pass, $base);
7       if (!$con) {
8          die("conexion fallida" . mysqli_connect_error());
9       }
10      $datos= "select * from Sensores ORDER BY ID DESC LIMIT 1";
11      $resultados= mysqli_query($con, $datos);
12      while ($resultados1=mysqli_fetch_array($resultados)){
13      //echo "Temperatura=";
14      echo $resultados1 ['Temperatura'];
15      }
16      ?>
```

Cód. 3: lectura valores sensor de temperatura

5.5 Servicio NO-IP

Se utilizará la Raspberry Pi como servidor para acceder remotamente desde cualquier red, es más práctico y seguro crear un subdominio para librarse de usar la dirección IP estatica que fue asignada al dispositivo. Además, como es posible que la IP del router sea dinámica, es decir, puede variar cada vez que éste se reinicie, con este servicio de hostname virtual (NO-IP) se soluciona el problema de que cambie la IP.

En primer lugar, se debe crear una cuenta gratuita en la página web http://www.noip.com y configurar un hostname. Una vez que ya se obtiene un usuario y contraseña se debe instalar el paquete NO-IP en la Raspberry Pi. Finalmente, para utilizar este servicio, se necesita abrir el puerto 80 del router (forwarding), ya que se usa el protocolo HTTP, al que esté conectada la Raspberry Pi para así poder utilizar el servicio NO-IP.

5.6 Servidor

A continuación, en el Cód. 4, se detalla el programa general explicado en la sección de diseño.

```
1 import serial
2 import time
3 import MySQLdb
4 import modo3
5 import modo4
6 import modo2
7 import modo1
8 import programafinalsensoresycargadedatos0
9 db_host = 'localhost'
10 usuario = 'root'
11 clave = 'iua'
12 base_de_datos = 'TESIS'
13 verdad= 1
14 while verdad < 2 :
15 programafinalsensoresycargadedatos0.cargadatos()
16 db = MySQLdb.connect(host=db_host, user=usuario, passwd=clave,
17 db=base_de_datos)
18 cursor = db.cursor()
19 mi_query = "SELECT Modo FROM Aplicacion ORDER BY ID DESC"
20 cursor.execute(mi_query)
21 modo = cursor.fetchone()
22 print(modo)
23 # comparamos los distintos modos de operar
24 if modo== (4L,):
```

```
25 modo4.moda4()
26 elif modo== (3L,):
27 modo3.modo()
28 elif modo== (2L,):
29 modo2.moda2()
30 else:
31 modo1.moda1()
```

Cód. 4: Programa General Servidor

Si el modo seleccionado es Económico o Full (línea 29 y 31 del Cód. 4), el programa carga automáticamente una configuración preestablecida por el programador, y la Raspberry Pi compara los días seleccionados en el programa con los días reales y le envía la orden de actuar a la placa Arduino en el día y hora establecidos (línea 18 del Cód. 5).

A continuación, en el Cód. 5, se muestra el programa del modo Económico que es muy similar al Modo Full.

```
1# MODO ECONOMICO
2#cloro los viernes a las 22:15
3#motor lunes y viernes de 22 a 22:30
4#calef off
5#luces off
6#riego cada martes y sabado dias de 19:30 a 20
7#quimicos jueves 22:15
8def moda1():
9 import time
10 import serial
11 arduino = serial.Serial('/dev/ttyACM0', 9600)
12 time.sleep(0.2)
13 diareal=time.strftime("%A")
14 horareal=time.localtime(time.time())[3]
15 minutoreal=time.localtime(time.time())[4]
16 segundoreal=time.localtime(time.time())[5]
17  #cloro
18 if diareal=="Friday" and horareal==22 and minutoreal==15 and
19    segundoreal<10:
20    arduino.write('m')
21    time.sleep(15)
22    arduino.write('n')
23  #motor
24 if diareal=="Monday" and horareal==22 and minutoreal==00 and
25    segundoreal<10:
26    arduino.write('z')
27 if diareal=="Monday" and horareal==22 and minutoreal==30 and
28    segundoreal<10:
29        arduino.write('y')
30 if diareal=="Friday" and horareal==22 and minutoreal==00 and
31     segundoreal<10:
32     arduino.write('z')
33 if diareal=="Friday" and horareal==22 and minutoreal==30 and
```

```
34     segundoreal<10:
35     arduino.write('y')
36   #quimicos
37 if diareal=="Thursday" and horareal==22 and minutoreal==15 and
38     segundoreal<10:
39     arduino.write('i')
40     time.sleep(15)
41     arduino.write('f')
42     print("activado")
43   #riego
44 if diareal=="Tuesday" and horareal==19 and minutoreal==30 and
45     segundoreal<10:
46     arduino.write('v')
47 if diareal=="Tuesday" and horareal==20 and minutoreal==00 and
48     segundoreal<10:
49     arduino.write('u')
50 if diareal=="Saturday" and horareal==19 and minutoreal==30 and
51     segundoreal<10:
52     arduino.write('v')
53 if diareal=="Saturday" and horareal==20 and minutoreal==00 and
54     segundoreal<10:
55     arduino.write('u')
56 arduino.close()
```

Cód. 5: Modo Económico

Por otra parte si el modo seleccionado es el Personalizado, la placa descarga los datos de la tabla Personalizado ubicada en la base de datos, que el usuario previamente programo por la aplicación. Y luego compara los días y horas seleccionadas con los días y horas reales y acciona los sistemas por medio de la placa Arduino en el momento establecido. El código del programa es similar al Cód. 5 excepto que se agrega la descarga de datos de la tabla antes mencionada.

Por último, si el modo seleccionado es el modo manual de acción inmediata, la placa Raspberry descarga los datos desde la tabla Manual. E inmediatamente acciona los sistemas que el usuario enciende o apaga desde la aplicación móvil.

6. Test del Sistema

6.1 Test del Subsistema de Temperatura

El sensor de temperatura utilizado es una sonda sumergible que al aumentar la temperatura aumenta su resistencia. Por lo tanto, el circuito utilizado para poder detectar estos cambios de señal en el concentrador de datos Arduino Mega 2560 fue un circuito divisor resistivo. El circuito se conectó como se muestra en la Fig. 38.

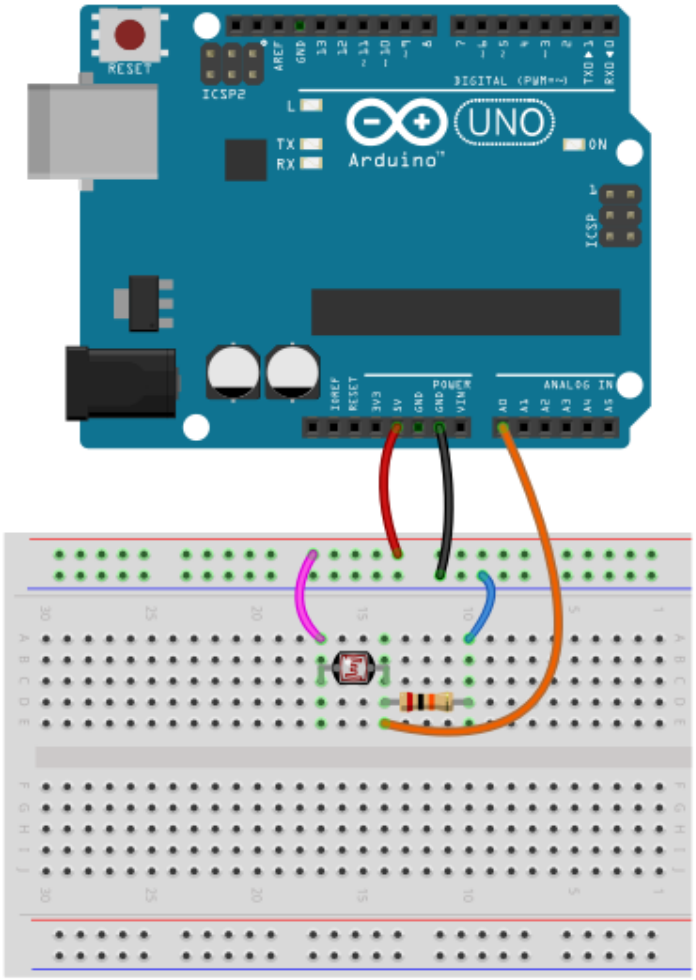


Fig. 38: Circuito sensor temperatura.

En el Cód. 6, a continuación, se observa como fue configurada la sonda.

```
1 float temperatura;
2 float voltaje;
3 int Vin = A0;
4 int resistencia1 = 220;
5 float resistenciapt;
6 int Calibrar= 0.38
7
8 void setup() {
9  Serial.begin(9600);
10 }
11
12 void loop() {
13   voltaje = (analogRead(Vin) / 1023.0) * 5.0;
14   resistenciapt = ((resistencia1 * 5.0) / voltaje) - resistencia1;
15   temperatura = (resistenciapt - 100) / Calibrar;
16   Serial.println(temperatura);
17   delay(200);
18 }
```

Cód. 6: Sensor Temperatura

Para calibrar este sensor y elegir la resistencia utilizada en el divisor resistivo, se sometió el circuito de la Fig. 40 a una prueba en tres recipientes con líquido a temperaturas conocidas, agua a 0ºC., agua a 100ºC. y agua a 22ºC.

Finalmente, como resultado de estas pruebas se utilizó una resistencia de 220 Omhs por ser la que mejor se acoplaba al circuito, y se calibro el programa con un factor "Calibrar" de 0.38.

6.2 Test del Subsistema de Humedad

El sensor de humedad de suelo se conectó al concentrador Arduino por medio de una placa amplificadora de señal como se observa en la Fig. 39.

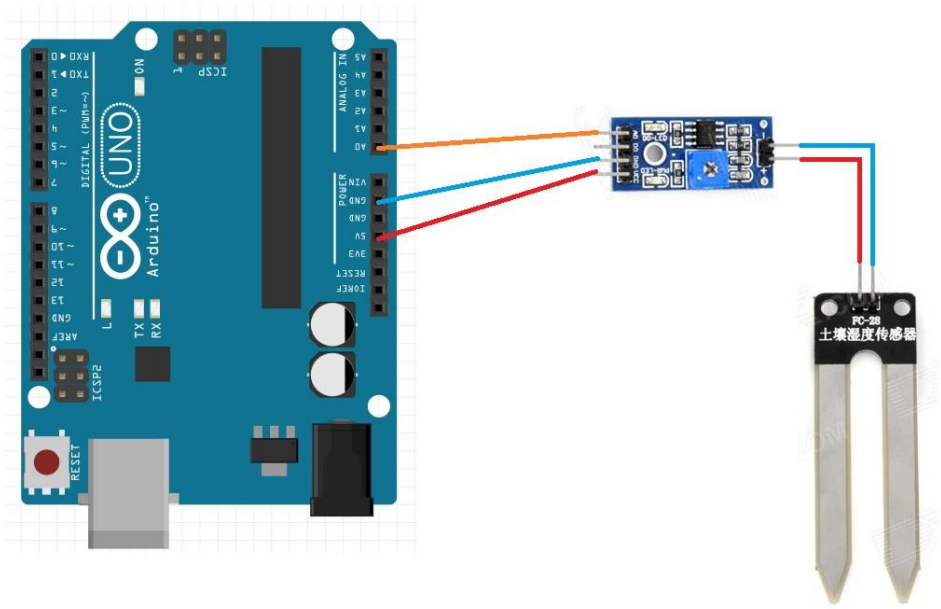


Fig. 39: Circuito sensor humedad.

Luego de realizar diversas pruebas en suelos con distinta humedad, se optó por aplicarle el Cód. 7 como programa de configuración dividiendo el rango del sensor analógico en cuatro posibles estados: Muy seco, Seco, Húmedo y Muy húmedo. Y a su vez se calibro cada estado en correspondencia un distinto valor de humedad de suelo.

```
float voltajehumedad;
int VinHumedad = A5;
void setup() {
 Serial.begin(9600);
}
void loop() {
  voltajehumedad = (analogRead(VinHumedad) / 1023.0) * 5.0;

  if (voltajehumedad <= 2.5) {
    Serial.println("muy humedo");
    delay(100);
  } else if (voltajehumedad < 3.5) {
    Serial.println("humedo");
    delay(100);
  } else if (voltajehumedad < 4 ) {
    Serial.println("seco");
    delay(100);
  } else {
    Serial.println("muy seco");
    delay(100);
```

Cód. 7: Sensor Humedad

6.3 Test del Subsistema IR

El sistema IR, consiste en un receptor IR y un control emisor IR. Al concentrador Arduino se conetecta el receptor de la señal IR por el siguiente circuito que se muestra en la Fig. 40.

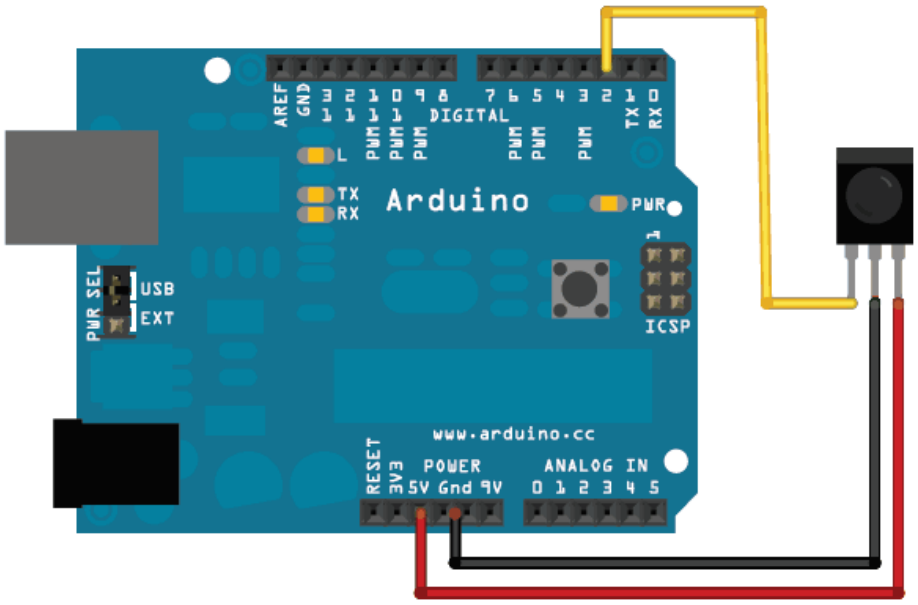


Fig. 40: Circuito sistema IR.

Luego de conectar el receptor se realizó una prueba para poder conocer el código de cada botón del control IR, obteniendo así la tabla 4.

Tecla del Control IR	Código numérico correspondiente
Motor ON	16753245
Motor OFF	16736925
Cloro	16720605
Quimicos	16712445
Luces ON	16769055
Luces OFF	16754775
Riego ON	16738455
Riego OFF	16750695
Temperatura ON	16724175
Temperatura OFF	16718055

Tabla 4: Códigos numéricos control IR.

6.4 Test del Subsistema de Pantalla LCD

Para la prueba del sistema de la pantalla LCD se realizó un circuito con el siguiente esquema de conexión que se muestra en la Fig. 41, cabe destacar que el sistema LCD se conectó por medio de un módulo I2C para poder utilizar menos pines en el concentrador Arduino.

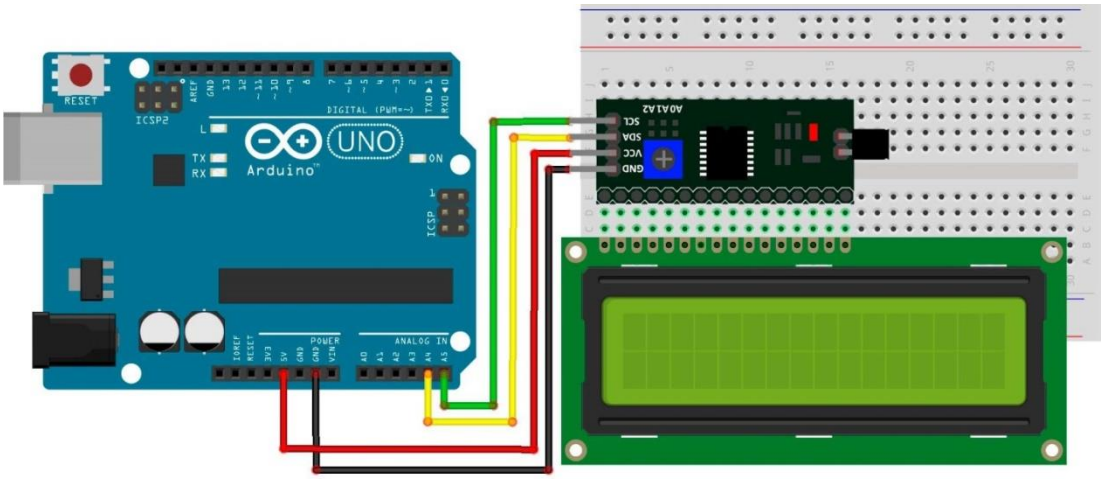


Fig. 41: Circuito pantalla LCD.

Luego de realizar esta conexión se cargó el Cód. 8 en la placa Arduino para probar el módulo I2C, la pantalla LCD y calibrar el brillo y la nitidez de la misma.

```
#include<Wire.h>
#include<LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3f,16,2);

void setup()
{
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
}

void loop() {
  lcd.setCursor(0, 0);
  lcd.print("  BIENVENIDOS ");
  lcd.setCursor(0,1);
  lcd.print("  A PILETA IoT");
}
```

Cód. 8: Calibración Pantalla LCD

6.5 Test del Subsistema de Relé

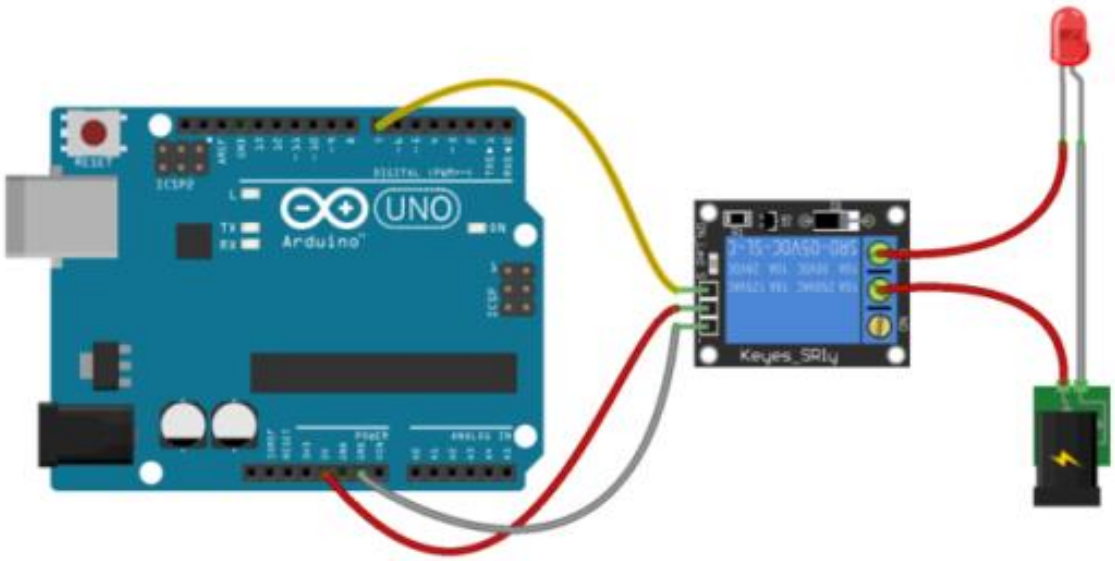


Fig. 42: Circuito sistema de relé.

Esta prueba consistió en encender un LED por medio de un relé conectado al concentrador Arduino para poder observar el correcto funcionamiento del mismo, el esquema del circuito utilizado fue el que se muestra en la Fig. 42.

6.6 Test del Subsistema de Medición de Distancia

Para este sistema se utilizaron sensores ultrasónicos para poder medir el nivel de agua correspondiente a cada recipiente de químicos. Por lo tanto, se realizaron dos pruebas distintas de calibración y medición. La primera prueba fue por medio del esquema de conexión que se observa en la Fig. 43.

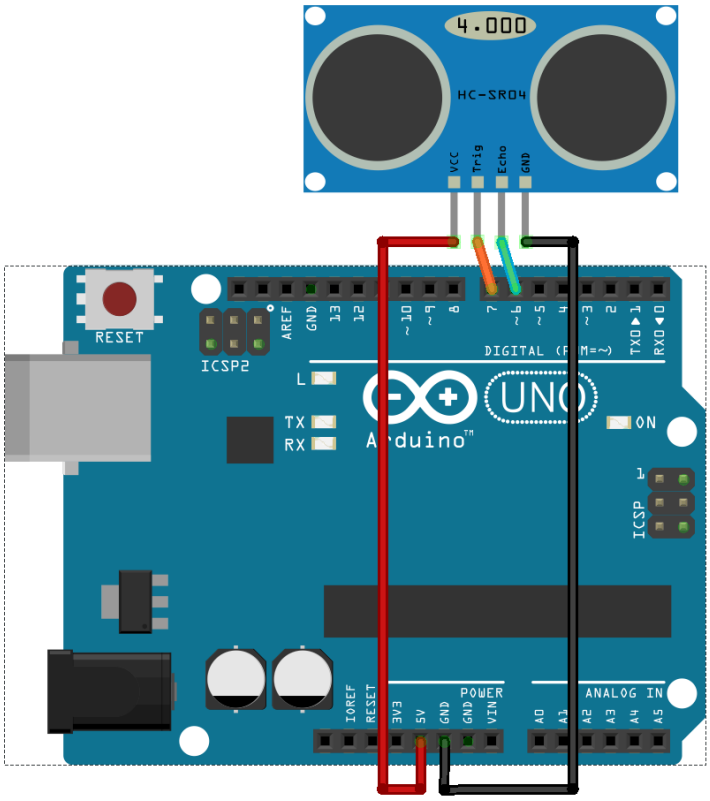


Fig. 43: Circuito medición de distancia

La primera prueba fue ubicando el sensor ultrasónico de forma paralela a una cinta de medir, y moviendo una superficie plana al frente del mismo para observar, por medio de la placa Arduino, si el valor de la distancia medida por la placa y visto por la cinta métrica eran iguales. Por otro lado, se realizó una prueba en donde se ubicó al sensor en la tapa del recipiente donde se colocaría en el sistema, y se calibro la altura mínima y máxima a donde pueden llegar los químicos y se escribió el código de programación Cód. 9 donde se observan estos datos.

```
1 const int PinTrig1 = 48;
2 const int PinEcho1 = 53;
3 const float VelSon = 34000.0;
4 float distancia1;
5 float maximotanque1 = 50;
6 float cantidaddeliquido1;
7
8 void setup() {
9
10  Serial.begin(9600);
```

```
11  pinMode(PinTrig1, OUTPUT);
12  pinMode(PinEcho1, INPUT);
13}
14
15 void loop() {
16  //calcula distancia 1
17   iniciarTrigger1();
18   unsigned long tiempo1 = pulseIn(PinEcho1, HIGH);
19   distancia1 = tiempo1 * 0.000001 * VelSon / 2.0;
20     cantidaddeliquido1 = 100 - (((cm1 - minimotanque1) * 100) /
21(maximotanque1 - minimotanque1));
22  if (cantidaddeliquido1 < 99.99) {
23    Serial.print(cantidaddeliquido1);
24  }
25  else {
26    Serial.print(99.99);
27  }
28  delay(200);
29}
30
31//Método que inicia la secuencia del Trigger1 para comenzar a medir
32 void iniciarTrigger1()
33 {
34  digitalWrite(PinTrig1, LOW);
35  delayMicroseconds(2);
36  digitalWrite(PinTrig1, HIGH);
37  delayMicroseconds(10);
38  digitalWrite(PinTrig1, LOW);
39 }
```

Cód. 9: Calibración Sensores de Distancia

6.7 Test Sistema de Base de Datos

Luego de crear la base de dados MySQL se le realizaron pruebas subiendo diferentes datos desde la aplicación Android y desde la Raspberry Pi para corroborar su correcto funcionamiento. En estas pruebas se detectaron algunas fallas en el formato que recibía cuando se le enviaba un dato que representaba horas y minutos, que luego se pudo corregir para que el sistema funcione correctamente.

6.8 Test Aplicación Android

Luego de generar la aplicación, se invitó a 20 posibles usuarios del sistema de piletas integrada a IoT para que prueben la aplicación y realizaran una breve encuesta. Los resultados obtenidos son los mostrados en las Fig. 44-46.

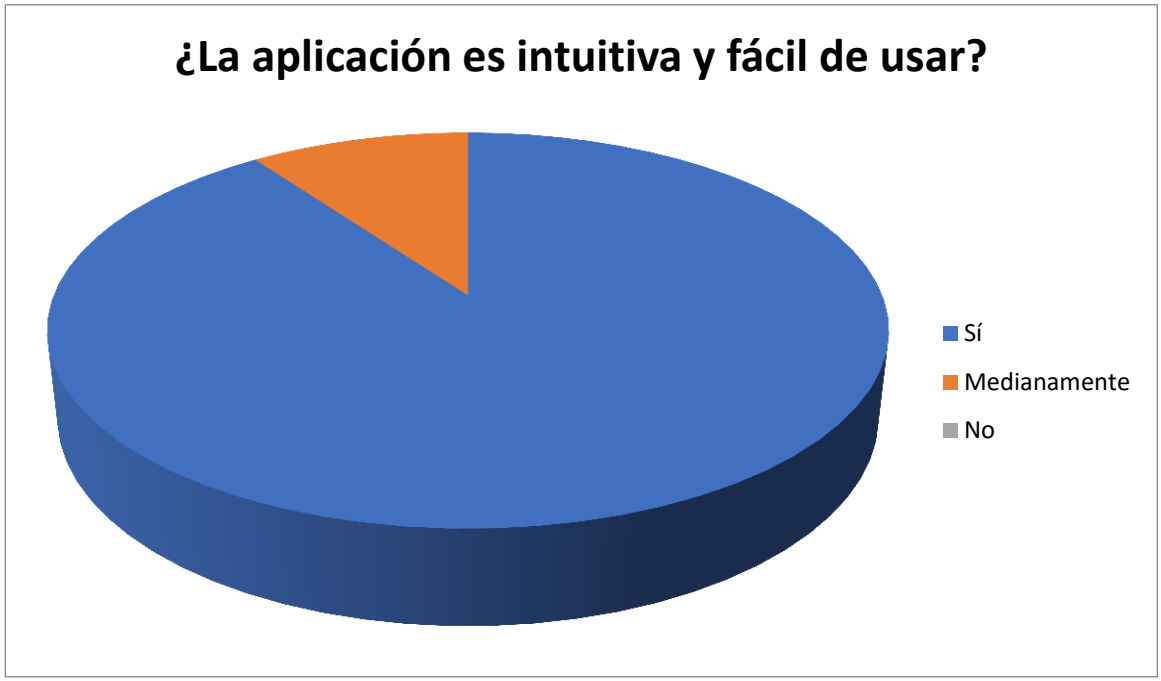


Fig. 44: Encuesta 1

Fig. 45: Encuesta 2

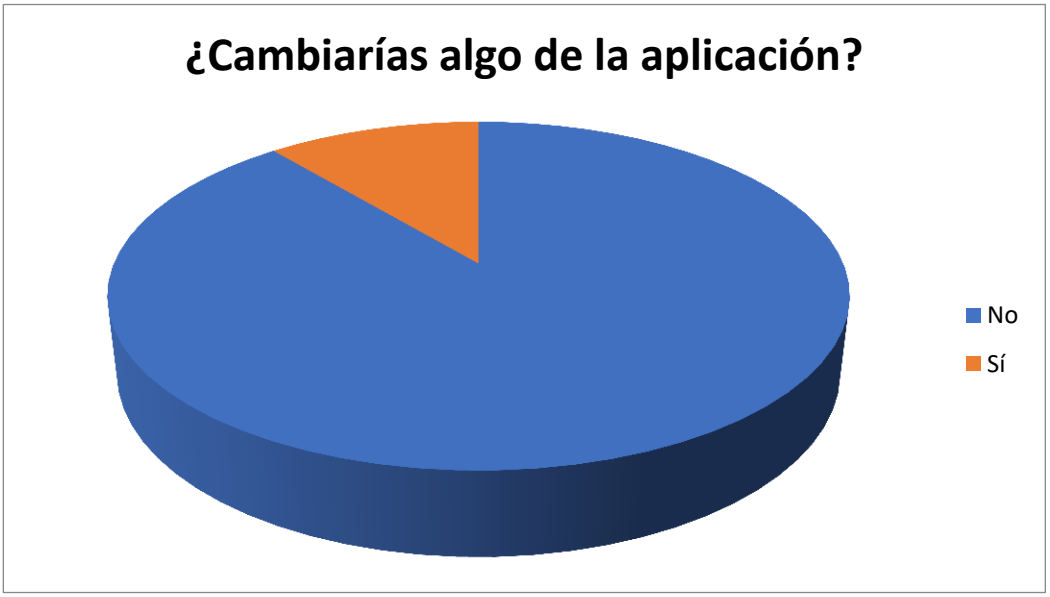


Fig. 46: Encuesta 3

Luego de esta prueba y breve encuesta, se observó que la aplicación era intuitiva y fácil de usar. Utilizando los consejos de los encuestados, se optó por realizar un cambio en

el tamaño de la letra para que sea más clara y se vea mejor. Una vez realizados los cambios sugeridos, se obtuvo la aplicación móvil detallada en capítulos anteriores donde se adaptaron los sensores, motores e iluminación para simular de la mejor manera posible todos los casos reales.

7. Montaje

Para la realización de la maqueta se utilizó madera MDF, una pecera de 6 L, una caja de madera para representar el sistema de motores de una pileta real y dos recipientes de plástico para representar los compartimentos reales del cloro y químicos. Además de los sensores y motores mencionados con anterioridad, tal como se puede observar en la Fig. 47-49.

Fig. 47: Modelo a escala pileta y patio

Fig. 48: Modelo a escala pileta y patio

Fig. 49: Modelo a escala pileta y patio

Dentro de la caja blanca ubicada del lado izquierdo de la Fig. 47, se encuentran los motores, conectados a su respectivo relé, que son alimentados por una fuente de 12 voltios. A su vez, los relé mencionados se encuentran conectados a los pines correspondientes de la placa Arduino. En la Fig. 50 se logra apreciar el interior de la caja.

Fig. 50: Caja de motores

Si se quisiera implementar el sistema en una pileta y patio en tamaño real, se tendrían que adaptar los distintos motores y las fuentes de alimentación.

8. Discusión y Análisis de Resultados

Una vez realizado el montaje de la maqueta y finalizado el desarrollo del sistema podemos observar que los resultados fueron positivos ya que se logró automatizar de forma eficiente las tareas de mantenimiento de la pileta y del jardín mencionadas en capítulos anteriores. Luego de realizar encuestas con un grupo de personas, se observa que la aplicación móvil creada contará con la aprobación de potenciales clientes.

La principal complicación fue que no se pudo implementar el sistema a escala real debido a los problemas logísticos que esto implicaba. A su vez, muchos de los sensores de los que se habían planeado utilizar en un principio tuvieron que ser descartados por su elevado costo, por ejemplo, el sensor que mide el nivel de pH en el agua de la pileta. Finalmente se observaron dificultades para poder utilizar el sistema remotamente (desde otra red) en ciertos casos, debido a que no todos los proveedores de Internet permiten realizar forwarding en los puertos de sus routers.

9. Conclusiones

Luego de finalizar el trabajo final de grado, se concluye que el desarrollo del mismo nos permitió adquirir la capacidad de aprender diferentes temas de los que no teníamos conocimientos previos. La investigación nos favoreció mucho a nivel profesional, ya que pudimos aplicar e integrar conocimientos que hasta el momento aparentaban no tener relación entre sí.

Dentro de los objetivos alcanzados podemos mencionar haber desarrollado una aplicación móvil para poder administrar el sistema de forma eficiente. Se logró implementar el sistema a una maqueta a escala, automatizado los diversos procesos y acciones que se llevan a cabo en la administración de una pileta y jardín. Además, se logró desarrollar un sistema embebido capaz de adquirir datos de forma automática por medio de los sensores y enviarlos a una base de datos alojada en un servidor para poder acceder a esta información de forma local y remota. Se logró implementar un servidor conectado a Internet capaz de permitir el acceso a los datos y capaz de realizar el procesamiento de los mismos. Por último, se creó un sistema que de forma eficiente es capaz de realizar las tareas de mantención y cuidado de la pileta y el jardín por medio de una aplicación móvil desde cualquier lugar del planeta con acceso a internet.

Por otra parte en un futuro se podría crear un sistema para alimentar el servidor y concentrador por completo mediante baterías y/o paneles solares, para poder lograr autonomía con respecto a alguna fuente de alimentación dependiente de la red eléctrica. Además, se podría agregar un sistema de cámaras de seguridad para observar el jardín de forma remota por medio de la aplicación móvil. Como una ampliación del sistema, se podría integrar un detector de presencia de niños en las cercanías a la pileta, el cual, llegado el caso, activaría una señal de alarma que sería enviada al o los usuarios del sistema.

Bibliografía y Referencias

[1] Dirección General de Catastro de la provincia de Córdoba. Obtenido de http://dgc.cba.gov.ar/

[2] *En Córdoba hay 20.600 piletas de natación.* (15 de Junio de 2016). Obtenido de Dia a Dia: http://diaadia.viapais.com.ar/cordoba/en-cordoba-hay-20600-piletas-de-natacion

[3] Zanella, A. (2015). Internet of Things for Smart Cities. *IEEE Internet of Things Journal, VOL. 1, NO. 1*.

[4] Simposio Argentino de Sistemas Embebidos (SASE). Obtenido de http://sase.com.ar/

[5] Arduino. (18 de Junio de 2017). Obtenido de http://arduino.cl/arduino-mega-2560/

[6] Raspberry. (20 de Junio de 2017). Obtenido de https://raspberrypi.org/

[7] Donat, W. (2014). *Learn Raspberry Pi Programming with Python*

[8] Upton. (2016). *Learning Computer Architecture with Raspberry Pi*

[9] Ala Al-Fuqaha, M. G. (2015). *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications.* IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 17.

[10] Guglielmone Nicolas, N. F. (2016). *Prototipo de plantilla para prevención de úlceras plantares en personas con diabetes.* Universidad Nacional de Cordoba.

Códigos de Computación

Programa de la placa Arduino completo.

```
#include<Wire.h>
#include<LiquidCrystal_I2C.h>
#include <IRremote.h>

LiquidCrystal_I2C lcd(0x3f, 16, 2);


decode_results results;

float temperatura;
float voltajehumedad;
float voltajetemperatura;
int VinTemperatura = A7;
int VinHumedad = A0;
int resistencia1 = 220;
int hume;
int ONOFF1 = 0;
int minimotanque1=5;
int minimotanque2=5;
float resistenciapt;
const int TriggerPin1 = 11;
const int EchoPin1 = 10;
const int EchoPin2 = 43;
const int TriggerPin2 = 38;
const float VelSon = 34000.0;
float distancia1;
float distancia2;
float maximotanque1 = 16;
float maximotanque2 = 16;
float cantidaddeliquido1;
float cantidaddeliquido2;

int RECV_PIN = 2;
uint32_t IR = 0;
int motor = 0;
int luces = 0;
int calefaccion = 0;
int riego = 0;
const int motorpin = 24;
const int calefaccionpin = 44 ;
const int lucespin = 28;
const int cloropin = 32;
const int quimicospin = 34;
const int riegopin = 36;
const int LLAVE = 4;
int ONOFF = digitalRead(LLAVE);
IRrecv irrecv(RECV_PIN);




void setup() {
```

```
  Serial.begin(9600);
  // Ponemos el pin Trig en modo salida
  pinMode(TriggerPin1, OUTPUT);
  pinMode(TriggerPin2, OUTPUT);
  // Ponemos el pin Echo en modo entrada
  pinMode(EchoPin1, INPUT);
  pinMode(EchoPin2, INPUT);
  pinMode(24, OUTPUT);
  pinMode(28, OUTPUT);
  pinMode(32, OUTPUT);
  pinMode(34, OUTPUT);
  pinMode(36, OUTPUT);
  pinMode(44, OUTPUT);
  pinMode(motorpin, OUTPUT);
  pinMode(calefaccionpin, OUTPUT);
  pinMode(lucespin, OUTPUT);
  pinMode(cloropin, OUTPUT);
  pinMode(quimicospin, OUTPUT);
  pinMode(riegopin, OUTPUT);
  pinMode(LLAVE, INPUT);
  lcd.init();
  irrecv.enableIRIn();

}



void cloromanual ()
{
digitalWrite(cloropin, HIGH);
lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
lcd.print(" Lanzando Cloro ");
delay(4000);
lcd.setCursor(0, 0);
lcd.print("                 ");
lcd.init();
digitalWrite(cloropin, LOW);


}


void quimicosmanual ()
{
digitalWrite(quimicospin, HIGH);
lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
lcd.print("Lanzando Quimicos ");
delay(4000);
lcd.setCursor(0, 0);
lcd.print("                 ");
lcd.init();
digitalWrite(quimicospin, LOW);

}
```

```
void loop() {

  char c = Serial.read();

  // PARTE MANUAL


  int ONOFF = digitalRead(LLAVE);
  if (ONOFF ==  HIGH)
  {
    lcd.backlight();
    lcd.setCursor(0, 0);
    lcd.print("  BIENVENIDOS ");
    lcd.setCursor(0, 1);
    lcd.print("  A PILETA IoT");

    if (motor == 0)
    {
      lcd.setCursor(16, 0);
      lcd.print("MOTOR OFF");
    }
    if (motor == 1)
    {
      lcd.setCursor(16, 0);
      lcd.print("MOTOR ON");
    }
    if (calefaccion == 0)
    {
      lcd.setCursor(16, 1);
      lcd.print("CALEFACCION OFF");
    }
    if (calefaccion == 1)
    {
      lcd.setCursor(16, 1);
      lcd.print("CALEFACCION ON ");
    }
    if (luces == 0)
    {
      lcd.setCursor(32, 0);
      lcd.print("LUCES OFF");
    }
    if (luces == 1)
    {
      lcd.setCursor(32, 0);
      lcd.print("LUCES ON ");
    }
    if (riego == 0)
    {
      lcd.setCursor(32, 1);
      lcd.print("RIEGO OFF");
    }
    if (riego == 1)
    {
      lcd.setCursor(32, 1);
      lcd.print("RIEGO ON ");
    }

    lcd.scrollDisplayLeft();

    if (irrecv.decode(&results)) {
```

```
    IR = results.value, HEX;
    // Serial.println(IR);
    switch (IR)
    { case 16753245:
        Serial.println("MOTOR ON ");
        motor = 1;
        digitalWrite(motorpin, HIGH);
        break;
      case 16736925:
        Serial.println("MOTOR OFF");
        motor = 0;
        digitalWrite(motorpin, LOW);
        break;
      case 16720605:
        Serial.println("CLORO ");
        cloromanual();
        break;
      case 16712445:
        Serial.println("QUIMICOS");
        quimicosmanual();
        break;
      case 16769055:
        Serial.println("LUCES ON ");
        digitalWrite(lucespin, HIGH);
        luces = 1;
        break;
      case 16754775:
        Serial.println("LUCES OFF");
        luces = 0;
        digitalWrite(lucespin, LOW);
        break;
      case 16738455:
        Serial.println("RIEGO ON ");
        riego = 1;
        digitalWrite(riegopin, HIGH);
        break;
      case 16750695:
        Serial.println("RIEGO OFF");
        riego = 0;
        digitalWrite(riegopin, LOW);
        break;
      case 16724175:
        Serial.println("CALEFACCION ON ");
        digitalWrite(calefaccionpin, HIGH);
        calefaccion = 1;
        break;
      case 16718055:
        Serial.println("CALEFACCION OFF");
        digitalWrite(calefaccionpin, LOW);
        calefaccion = 0;
        break;

    }
    irrecv.resume();
  }
  //lcd.init();
  //lcd.noBacklight();
  delay(500);
}
```

```
      // PARTE AUTOMATICA Y APLICACION
      else {
        lcd.noBacklight();
        if (c == 't') {


          //calcula temperatura
          voltajetemperatura = (analogRead(VinTemperatura) / 1023.0) * 5.0;
          resistenciapt  =  ((resistencia1  *  5.0)  /  voltajetemperatura)  -
resistencia1;
          temperatura = (resistenciapt - 100) / 0.38;
          delay(100);
          Serial.print(temperatura);

        }

        else if (c == 'h') {

          //calcula humeda
          voltajehumedad = (analogRead(VinHumedad) / 1023.0) * 5.0;

          if (voltajehumedad <= 2.5) {
            hume = 1;
            delay(100);
            Serial.print(hume);
          } else if (voltajehumedad < 3.5) {
            hume = 2;
            delay(100);
            Serial.print(hume);
          } else if (voltajehumedad < 4 ) {
            hume = 3;
            delay(100);
            Serial.print(hume);
          } else {
            hume = 4;
            delay(100);
            Serial.print(hume);
          }
        }

        //calcula distancia 1
        else if (c == 'a') {
          float cm1 = ping1(TriggerPin1, EchoPin1);
          cantidaddeliquido1 = 100-(((cm1-minimotanque1) * 100) / (maximotanque1-
minimotanque1));

          if (cantidaddeliquido1 < 99.99) {
            Serial.print(cantidaddeliquido1);
          }
          else {
            Serial.print(99.99);
          }
       }


        //calcula distancia2
        else if (c == 'b') {
      float cm2 = ping2(TriggerPin2, EchoPin2);
      float  cantidaddeliquido2  =  100-(((cm2-minimotanque2)*100)/  (maximotanque2-
minimotanque2));
```

```
    if (cantidaddeliquido2 < 99.99 && cantidaddeliquido2> 0.00) {
      Serial.println(cantidaddeliquido2);
    }
    else {
      Serial.print(0.0);
    }
delay(1000);
  }

 //Poner salidas en 1 para prender motores

  else if (c == 'z') {
    digitalWrite(24, HIGH);

  }
  else if (c == 'y') {
    digitalWrite(24, LOW);

  }
  else if (c == 'x') {
    digitalWrite(28, HIGH);

  }
  else if (c == 'w') {
    digitalWrite(28, LOW);

  }
  else if (c == 'v') {
    digitalWrite(36, HIGH);

  }
  else if (c == 'u') {
    digitalWrite(36, LOW);

  }

  else if (c == 'j') {
    digitalWrite(44, HIGH);

  }
  else if (c == 's') {
    digitalWrite(44, LOW);

  }

  else if (c == 'r') {
    digitalWrite(48, HIGH);

  }
  else if (c == 'm') {
    digitalWrite(32, HIGH);

}
  else if (c == 'n') {
    digitalWrite(32, LOW);

}  else if (c == 'i') {
    digitalWrite(34, HIGH);
}
  else if (c == 'f') {
```

```
            digitalWrite(34, LOW);
        }
        }
        }

        float ping1(int TriggerPin1, int EchoPin1) {
        float duration1, distanceCm1;
        digitalWrite(TriggerPin1, LOW);//para generar un pulso limpio ponemos a LOW
4us
        delayMicroseconds(4);
        digitalWrite(TriggerPin1, HIGH);//generamos Trigger (disparo) de 10us
        delayMicroseconds(10);
        digitalWrite(TriggerPin1, LOW);
        duration1 = pulseIn(EchoPin1, HIGH);//medimos el tiempo entre pulsos, en
microsegundos
        distanceCm1 = duration1 * 10 / 292/ 2;
        return distanceCm1;
        }
        float ping2(int TriggerPin2, int EchoPin2) {
        float duration2, distanceCm2;
        digitalWrite(TriggerPin2, LOW);//para generar un pulso limpio ponemos a LOW
4us
        delayMicroseconds(4);
        digitalWrite(TriggerPin2, HIGH);//generamos Trigger (disparo) de 10us
        delayMicroseconds(10);
        digitalWrite(TriggerPin2, LOW);
        duration2 = pulseIn(EchoPin2, HIGH);//medimos el tiempo entre pulsos, en
microsegundos
        distanceCm2 = duration2 * 10 / 292/ 2;
        return distanceCm2;
        }
```

Programas de Raspberry Pi completos.

Programa principal.

```
import serial
import time
import MySQLdb
import modo3
import modo4
import modo2
import modo1
import programafinalsensoresycargadedatos0

db_host = 'localhost'
usuario = 'root'
clave = 'iua'
base_de_datos = 'TESIS'

verdad= 1

while verdad < 2 :

   programafinalsensoresycargadedatos0.cargadatos()

   db = MySQLdb.connect(host=db_host, user=usuario, passwd=clave,
db=base_de_datos)
```

```
   cursor = db.cursor()

   mi_query = "SELECT Modo FROM Aplicacion ORDER BY ID DESC"

   cursor.execute(mi_query)

   modo = cursor.fetchone()
   print(modo)

   # comparamos los distintos modos de operar
   if modo== (4L,):
      modo4.moda4()

   elif modo== (3L,):
       modo3.modo()

   elif modo== (2L,):
       modo2.moda2()

   else:
      modo1.moda1()
```

Programa carga datos a base de datos MySQL:

```
def moda4():

   import time
   import serial
   import MySQLdb

        arduino = serial.Serial('/dev/ttyACM0', 9600)

        time.sleep(2)

        db_host = 'localhost'
        usuario = 'root'
        clave = 'iua'
        base_de_datos = 'TESIS'

        db = MySQLdb.connect(host=db_host, user=usuario, passwd=clave,
db=base_de_datos)

        cursor = db.cursor()
        time.sleep(1)

#MOTOR

   mi_query = "SELECT Motor FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

   motor = cursor.fetchone()[0]

   if motor==1:
      arduino.write('z')
      print("on motor")
   elif motor==0:
      arduino.write('y')
      print("offmotor")
```

```
#cloro

   mi_query = "SELECT Cloro FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        cloro = cursor.fetchone()[0]

        if cloro==1:
      print("oncl")
                arduino.write('m')
        elif cloro==0:
      print("offl")
                arduino.write('n')

#quimicos

   mi_query = "SELECT Quimicos FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        quimicos = cursor.fetchone()[0]

        if quimicos==1:
                arduino.write('i')
      print("onqu")
        elif quimicos==0:
                arduino.write('f')
      print("offqu")

#luces

        mi_query = "SELECT Luces FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        luces = cursor.fetchone()[0]

        if luces==1:
      print("onluces")
                arduino.write('x')
        elif luces==0:
      print("offluces")
                arduino.write('w')

#Calefaccion

        mi_query = "SELECT Calefaccion01 FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        calefac = cursor.fetchone()[0]

   mi_query = "SELECT Calefaccion FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        grados = cursor.fetchone()[0]
```

```
    mi_query = "SELECT Temperatura FROM Sensores  ORDER BY ID DESC"

         cursor.execute(mi_query)

         gradospil = cursor.fetchone()[0]



    if calefac==1 and grados>gradospil:
                arduino.write('j')
       print("caleon")
         else :
                arduino.write('s')
       print("caleoff")

def cargadatos():

    import serial
    import time
    import MySQLdb

    db_host = 'localhost'
    usuario = 'root'
    clave = 'iua'
    base_de_datos = 'TESIS'

    db = MySQLdb.connect(host=db_host, user=usuario,
passwd=clave,db=base_de_datos)

    humedad=''
    temperatura=''
    liquido1=''
    liquido2=''
    txt=''

    arduino = serial.Serial('/dev/ttyACM0', 9600)

    time.sleep(2)

#    PARTE DE LA TEMPERATURA

    arduino.write('t')
    time.sleep(0.5)
    temperatura =arduino.readline(5)
    print(temperatura)

#    PARTE DE LA humedad

    arduino.write('h')
    time.sleep(0.5)
    humedad =arduino.readline(1)
    print(humedad)


#    PARTE DEl Liquido1

    arduino.write('a')
    time.sleep(1)
    liquido1 +=arduino.readline(5)
```

```
   print(liquido1)


#   PARTE DEl Liquido2

   arduino.write('b')
   time.sleep(1)
   liquido2 +=arduino.readline(5)
   print(liquido2)

   cursor = db.cursor()

   mi_query = "INSERT INTO Sensores (Temperatura, Humedad, Liquido1,Liquido2,
Alarma) VALUES (%s,%s,%s,%s,0)" %(temperatura,humedad,liquido1,liquido2)

   cursor.execute(mi_query)
   db.commit()




   arduino.close()
```

Programa modo Economico:

```
# MODO ECONOMICO
#cloro los viernes a las 22:15
#motor lunes y viernes de 22 a 22:30
#calef off
#luces off
#riego cada martes y sabado dias de 19:30 a 20
#quimicos jueves 22:15

def moda1():

   import time
   import serial

   arduino = serial.Serial('/dev/ttyACM0', 9600)
   time.sleep(0.2)
   diareal=time.strftime("%A")
   horareal=time.localtime(time.time())[3]
   minutoreal=time.localtime(time.time())[4]
   segundoreal=time.localtime(time.time())[5]

#cloro
   if diareal=="Friday" and horareal==22 and minutoreal==15 and segundoreal<10:

      arduino.write('m')
      time.sleep(15)
      arduino.write('n')

#motor

   if diareal=="Monday" and horareal==22 and minutoreal==00 and segundoreal<10:

           arduino.write('z')

   if diareal=="Monday" and horareal==22 and minutoreal==30 and segundoreal<10:

           arduino.write('y')
```

```
   if diareal=="Friday" and horareal==22 and minutoreal==00 and segundoreal<10:

            arduino.write('z')

   if diareal=="Friday" and horareal==22 and minutoreal==30 and segundoreal<10:

            arduino.write('y')

#quimicos
   if diareal=="Thursday" and horareal==22 and minutoreal==15 and segundoreal<10:

            arduino.write('i')
            time.sleep(15)
            arduino.write('f')
            print("activado")

#riego
   if diareal=="Tuesday" and horareal==19 and minutoreal==30 and segundoreal<10:

            arduino.write('v')

   if diareal=="Tuesday" and horareal==20 and minutoreal==00 and segundoreal<10:

            arduino.write('u')


   if diareal=="Saturday" and horareal==19 and minutoreal==30 and segundoreal<10:

            arduino.write('v')

   if diareal=="Saturday" and horareal==20 and minutoreal==00 and segundoreal<10:

            arduino.write('u')

   arduino.close()
```

Programa modo Full:

```
# MODO FULL
#cloro los lunes miercoles viernes a las 22:15
#motor diario de 22 a 22:30
#calef on 22 grados
#luces diarias de 22 a 00
#riego diario de 19:30 a 20
#quimicos martes jueves sabado 22:15


def moda2():
   import time
   import serial

   arduino = serial.Serial('/dev/ttyACM0', 9600)

   time.sleep(0.2)

   diareal=time.strftime("%A")
   horareal=time.localtime(time.time())[3]
   minutoreal=time.localtime(time.time())[4]
   segundoreal=time.localtime(time.time())[5]

#cloro
```

```
   if diareal=="Monday" and horareal==22 and minutoreal==15 and segundoreal<10:

      arduino.write('m')
      time.sleep(15)
      arduino.write('n')
      print("activado")

   if diareal=="Wendesday" and horareal==22 and minutoreal==15 and
segundoreal<10:

      arduino.write('m')
      time.sleep(15)
      arduino.write('n')
      print("activado")

   if diareal=="Friday" and horareal==22 and minutoreal==15 and segundoreal<10:

      arduino.write('m')
      time.sleep(15)
      arduino.write('n')
      print("activado")

#motor

   if horareal==22 and minutoreal==00 and segundoreal<10:

            arduino.write('z')

   if horareal==22 and minutoreal==30 and segundoreal<10:

            arduino.write('y')



#quimicos
   if diareal=="Tuesday" and horareal==22 and minutoreal==15 and segundoreal<10:

            arduino.write('i')
            time.sleep(15)
            arduino.write('f')
            print("activado")

   if diareal=="Thursday" and horareal==22 and minutoreal==15 and segundoreal<10:

            arduino.write('i')
            time.sleep(15)
            arduino.write('f')
            print("activado")

   if diareal=="Saturday" and horareal==22 and minutoreal==15 and segundoreal<10:

            arduino.write('i')
            time.sleep(15)
            arduino.write('f')
            print("activado")


#riego
   if horareal==19 and minutoreal==30 and segundoreal<10:
```

```
            arduino.write('v')

   if horareal==20 and minutoreal==00 and segundoreal<10:

            arduino.write('u')


#luces
   if horareal==22 and minutoreal==00 and segundoreal<10:

            arduino.write('x')

   if horareal==00 and minutoreal==00 and segundoreal<10:

            arduino.write('w')


#calefaccion

   arduino.write('t')
   time.sleep(0.1)
   temperaturareal = arduino.readline(5)

   if temperaturareal< 22.00:
          arduino.write('j')


   if temperaturareal > 22.00:
          arduino.write('s')

   arduino.close()
```

Programa modo Personalizado:

```
# MODO PERSONALIIZADO
#cloro personalizado
#motor  personalizado
#calef  personalizado
#luces personalizado
#riego personalizado
#quimicos personalizado

def modo():

   import time
   import serial
   import MySQLdb

        arduino = serial.Serial('/dev/ttyACM0', 9600)

        time.sleep(2)

        diareal=time.strftime("%A")
        horareal=str(time.localtime(time.time())[3])
        minutoreal=str(time.localtime(time.time())[4])
        segundoreal=str(time.localtime(time.time())[5])

        db_host = 'localhost'
        usuario = 'root'
```

```
        clave = 'iua'
        base_de_datos = 'TESIS'

        db = MySQLdb.connect(host=db_host, user=usuario, passwd=clave,
db=base_de_datos)
        cursor = db.cursor()


#cloro

        mi_query = "SELECT Diascloro FROM Personalizado ORDER BY ID DESC"

        cursor.execute(mi_query)

        diasc = cursor.fetchone()[0]
        dia1c=diasc[0]
        dia2c=diasc[1]
        dia3c=diasc[2]
        dia4c=diasc[3]
        dia5c=diasc[4]
        dia6c=diasc[5]
        dia7c=diasc[6]

        mi_query = "SELECT Horacloro FROM Personalizado ORDER BY ID DESC"

        cursor.execute(mi_query)

        horac1 = str(cursor.fetchone()[0])
        horac=horac1[2]+horac1[3]

        minutosc=horac1[5]+horac1[6]

       if dia1c=="D" and diareal=="Sunday" and horareal==horac and
minutoreal==minutosc and segundoreal==01:

                arduino.write('m')
               time.sleep(15)
                arduino.write('n')
                print("activado")

        if dia2c=="L" and diareal=="Monday" and horareal==horac and
minutoreal==minutosc and segundoreal==01:

                arduino.write('m')
                time.sleep(15)
                arduino.write('n')
                print("activado")

        if dia3c=="M"  and diareal=="Tuesday" and minutosc==minutoreal and
horareal==horac and segundoreal< str(10):
              arduino.write('m')
              time.sleep(15)
              arduino.write('n')
                   print("activado")


        if dia4c=="X" and diareal=="Wednesday" and horareal==horac and
minutoreal==minutosc and segundoreal==01:

                arduino.write('m')
```

```
                time.sleep(15)
                arduino.write('n')
                print("activado")

        if dia5c=="J" and diareal=="Thursday" and horareal==horac and
minutoreal==minutosc and segundoreal==01:

                arduino.write('m')
                time.sleep(15)
                arduino.write('n')
                print("activado")

        if dia6c=="V" and diareal=="Friday" and horareal==horac and
minutoreal==minutosc and segundoreal==01:

                arduino.write('m')
                time.sleep(15)
                arduino.write('n')
                print("activado")

        if dia7c=="s" and diareal=="Saturday" and horareal==horac and
minutoreal==minutosc and segundoreal==01:

                arduino.write('m')
                time.sleep(15)
                arduino.write('n')
                print("activado")


#calefaccion

   arduino.write('t')
   time.sleep(1)
   temperaturareal = arduino.readline(5)

   mi_query = "SELECT Temperatura FROM Personalizado ORDER BY ID DESC"

        cursor.execute(mi_query)

        temperatura = cursor.fetchone()[0]

   mi_query = "SELECT Calefacion01 FROM Personalizado ORDER BY ID DESC"

        cursor.execute(mi_query)

        calef = cursor.fetchone()[0]


   if temperaturareal< temperatura and calef== 1:
           arduino.write('j')


   if temperaturareal > temperatura:
          arduino.write('s')


#motor

        mi_query = "SELECT Diasmotor FROM Personalizado ORDER BY ID DESC"
```

```
        cursor.execute(mi_query)

        diasm = cursor.fetchone()[0]
        dia1m=diasc[0]
        dia2m=diasc[1]
        dia3m=diasc[2]
        dia4m=diasc[3]
        dia5m=diasc[4]
        dia6m=diasc[5]
        dia7m=diasc[6]

        mi_query = "SELECT Horamotor FROM Personalizado ORDER BY ID DESC"

        cursor.execute(mi_query)

        horam1 = str(cursor.fetchone()[0])

        horam=horam1[0]+ horam1[1]

        minutosm=horam1[2]+horam1[3]

        if dia1m=="D" and diareal=="Sunday" and horareal==horam and
minutoreal==minutosm and segundoreal==01:

                arduino.write('z')
               time.sleep(20)
                arduino.write('y')
                print("activado")

        if dia2m=="L" and diareal=="Monday" and horareal==horam and
minutoreal==minutosm and segundoreal==01:

                arduino.write('z')
                time.sleep(20)
                arduino.write('y')
                print("activado")

        if dia3m=="M"  and diareal=="Tuesday" and minutosm==minutoreal and
horareal==horam and segundoreal< str(10):
              arduino.write('z')
              time.sleep(15)
              arduino.write('y')
                print("activado")


        if dia4m=="X" and diareal=="Wednesday" and horareal==horam and
minutoreal==minutosm and segundoreal==01:

                arduino.write('z')
                time.sleep(15)
                arduino.write('y')
                print("activado")

        if dia5m=="J" and diareal=="Thursday" and horareal==horam and
minutoreal==minutosm and segundoreal==01:


                arduino.write('z')
                time.sleep(15)
                arduino.write('y')
```

```
                print("activado")

        if dia6m=="V" and diareal=="Friday" and horareal==horam and
minutoreal==minutosm and segundoreal==01:

                arduino.write('z')
                time.sleep(15)
                arduino.write('y')
                print("activado")

        if dia7m=="s" and diareal=="Saturday" and horareal==horam and
minutoreal==minutosm and segundoreal==01:

                arduino.write('z')
                time.sleep(15)
                arduino.write('y')
                print("activado")



#quimicos

        mi_query = "SELECT Diasquimicos FROM Personalizado ORDER BY ID DESC"

        cursor.execute(mi_query)

        diasq = cursor.fetchone()[0]
        dia1q=diasc[0]
        dia2q=diasc[1]
        dia3q=diasc[2]
        dia4q=diasc[3]
        dia5q=diasc[4]
        dia6q=diasc[5]
        dia7q=diasc[6]

        mi_query = "SELECT Horaquimicos FROM Personalizado ORDER BY ID DESC"

        cursor.execute(mi_query)

        horaq1 = str(cursor.fetchone()[0])

        horaq=horaq1[0]+ horaq1[1]

        minutosq=horaq1[2]+horaq1[3]


        if dia1q=="D" and diareal=="Sunday" and horareal==horaq and
minutoreal==minutosq and segundoreal==01:

                arduino.write('i')
               time.sleep(15)
                arduino.write('f')
                print("activado")

        if dia2q=="L" and diareal=="Monday" and horareal==horaq and
minutoreal==minutosq and segundoreal==01:

                arduino.write('i')
                time.sleep(15)
                arduino.write('f')
```

```
                print("activado")

        if dia3q=="M"  and diareal=="Tuesday" and minutosq==minutoreal and
horareal==horaq and segundoreal< str(10):
              arduino.write('i')
              time.sleep(15)
              arduino.write('f')
                print("activado")


        if dia4q=="X" and diareal=="Wednesday" and horareal==horaq and
minutoreal==minutosq and segundoreal==01:

                arduino.write('i')
                time.sleep(15)
                arduino.write('f')
                print("activado")

        if dia5q=="J" and diareal=="Thursday" and horareal==horaq and
minutoreal==minutosq and segundoreal==01:


                arduino.write('i')
                time.sleep(15)
                arduino.write('f')
                print("activado")

        if dia6q=="V" and diareal=="Friday" and horareal==horaq and
minutoreal==minutosq and segundoreal==01:

                arduino.write('i')
                time.sleep(15)
                arduino.write('f')
                print("activado")

        if dia7q=="s" and diareal=="Saturday" and horareal==horaq and
minutoreal==minutosq and segundoreal==01:

                arduino.write('i')
                time.sleep(15)
                arduino.write('f')
                print("activado")

#Riego


        mi_query = "SELECT Diasriego FROM Personalizado ORDER BY ID DESC"

        cursor.execute(mi_query)

        diasr = cursor.fetchone()[0]
        dia1r=diasc[0]
        dia2r=diasc[1]
        dia3r=diasc[2]
        dia4r=diasc[3]
        dia5r=diasc[4]
        dia6r=diasc[5]
        dia7r=diasc[6]

        mi_query = "SELECT Horariego FROM Personalizado ORDER BY ID DESC"
```

```
        cursor.execute(mi_query)

        horar1 = str(cursor.fetchone()[0])

        horar=horar1[0]+ horar1[1]

        minutosr=horar1[2]+horar1[3]

        if dia1r=="D" and diareal=="Sunday" and horareal==horar and
minutoreal==minutosr and segundoreal==01:

                arduino.write('v')
               time.sleep(15)
                arduino.write('u')
                print("activado")

        if dia2r=="L" and diareal=="Monday" and horareal==horar and
minutoreal==minutosr and segundoreal==01:

                arduino.write('v')
                time.sleep(15)
                arduino.write('u')
                print("activado")

        if dia3r=="M"  and diareal=="Tuesday" and minutosr==minutoreal and
horareal==horar and segundoreal< str(10):
              arduino.write('v')
              time.sleep(15)
              arduino.write('u')
                print("activado")


        if dia4r=="X" and diareal=="Wednesday" and horareal==horar and
minutoreal==minutosr and segundoreal==01:

                arduino.write('v')
                time.sleep(15)
                arduino.write('u')
                print("activado")

        if dia5r=="J" and diareal=="Thursday" and horareal==horar and
minutoreal==minutosr and segundoreal==01:


                arduino.write('v')
                time.sleep(15)
                arduino.write('u')
                print("activado")

        if dia6r=="V" and diareal=="Friday" and horareal==horar and
minutoreal==minutosr and segundoreal==01:

                arduino.write('v')
                time.sleep(15)
                arduino.write('u')
                print("activado")

        if dia7r=="s" and diareal=="Saturday" and horareal==horar and
minutoreal==minutosr and segundoreal==01:
```

```
                arduino.write('v')
                time.sleep(15)
                arduino.write('u')
                print("activado")

    arduino.close()
```

Programa modo Manual:

```
def moda4():

   import time
   import serial
   import MySQLdb

        arduino = serial.Serial('/dev/ttyACM0', 9600)

        time.sleep(2)

        db_host = 'localhost'
        usuario = 'root'
        clave = 'iua'
        base_de_datos = 'TESIS'

        db = MySQLdb.connect(host=db_host, user=usuario, passwd=clave,
db=base_de_datos)

        cursor = db.cursor()
        time.sleep(1)


#MOTOR

   mi_query = "SELECT Motor FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

   motor = cursor.fetchone()[0]

   if motor==1:
      arduino.write('z')

   elif motor==0:
      arduino.write('y')


#cloro

   mi_query = "SELECT Cloro FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        cloro = cursor.fetchone()[0]

        if cloro==1:
```

```
                arduino.write('m')
        elif cloro==0:

                arduino.write('n')

#quimicos

   mi_query = "SELECT Quimicos FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        quimicos = cursor.fetchone()[0]

        if quimicos==1:
                arduino.write('i')

        elif quimicos==0:
                arduino.write('f')


#luces

        mi_query = "SELECT Luces FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        luces = cursor.fetchone()[0]

        if luces==1:

                arduino.write('x')
        elif luces==0:

                arduino.write('w')

#Calefaccion

        mi_query = "SELECT Calefaccion01 FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        calefac = cursor.fetchone()[0]

   mi_query = "SELECT Calefaccion FROM Manual ORDER BY ID DESC"

        cursor.execute(mi_query)

        grados = cursor.fetchone()[0]


   mi_query = "SELECT Temperatura FROM Sensores  ORDER BY ID DESC"

        cursor.execute(mi_query)

        gradospil = cursor.fetchone()[0]


   if calefac==1 and grados>gradospil:
                arduino.write('j')
```

```
        else :
                arduino.write('s')

Arduino.close()
```

Programas PHP completos.

Programa de lectura de datos de temperatura.

```
<?php

$host= "localhost";
$user= "root";
$pass= "iua";
$base= "TESIS";

$con= mysqli_connect($host, $user, $pass, $base);

if (!$con) {
   die("conexion fallida" . mysqli_connect_error());
}

$datos= "select * from Sensores ORDER BY ID DESC LIMIT 1";


$resultados= mysqli_query($con, $datos);

while ($resultados1=mysqli_fetch_array($resultados)){

//echo "Temperatura=";
echo $resultados1 ['Temperatura'];
}



?>
```

Programa de lectura de datos de humedad.

```
<?php

$host= "localhost";
$user= "root";
$pass= "iua";
$base= "TESIS";

$con= mysqli_connect($host, $user, $pass, $base);

if (!$con) {
   die("conexion fallida" . mysqli_connect_error());
}

$datos= "select * from Sensores ORDER BY ID DESC LIMIT 1";
```

```
$resultados= mysqli_query($con, $datos);

while ($resultados1=mysqli_fetch_array($resultados)){

echo $resultados1 ['Humedad'];

}

?>
```

Programa lectura de cantidad de liquido:

```
<?php

$host= "localhost";
$user= "root";
$pass= "iua";
$base= "TESIS";

$con= mysqli_connect($host, $user, $pass, $base);

if (!$con) {
   die("conexion fallida" . mysqli_connect_error());
}

$datos= "select * from Sensores ORDER BY ID DESC LIMIT 1";


$resultados= mysqli_query($con, $datos);

while ($resultados1=mysqli_fetch_array($resultados)){

echo $resultados1 ['Liquido1'];

}

?>
```

Programa de carga de datos de modos de aplicacion.

```
<?php

$host= "localhost";
$user= "root";
$pass= "iua";
$base= "TESIS";

$con= mysqli_connect($host, $user, $pass, $base);
```

```
if (!$con) {
   die("conexion fallida" . mysqli_connect_error());
}
echo "conexion exitosa";


$mod = $_GET['Modo'];

mysqli_query($con, "INSERT INTO Aplicacion (Modo) VALUES('$mod')");
echo "datos insertados";


?>
```

Programa de carga de datos de modo manual de aplicacion.

```
      <?php

$host="localhost";
$user="root";
$pass="iua";
$base="TESIS";
$con=mysqli_connect($host,$user,$pass,$base);
if (!$con) {
   die("conexion fallida" . mysqli_connect_error());
}
echo "conexionetosa";

$mot = $_GET['Motor'];

$clo = $_GET['Cloro'];

$cale = $_GET['Calefaccion01'];

$grad = $_GET['Calefaccion'];

$alarm = $_GET['Alarma'];

$quim = $_GET['Quimicos'];

$luz = $_GET['Luces'];

mysqli_query($con, "INSERT INTO Manual (Motor, Cloro, Quimicos, Alarma, Luces,
Calefaccion, Calefaccion01) VALUES('$mot', '$clo', '$quim', '$alarm', '$luz',
'$grad', '$cale')");
echo "datos insertados";


?>
```

Programa de carga de datos de modo personalizado de aplicación.

```
<?php

$host= "localhost";
```

```
$user= "root";
$pass= "iua";
$base= "TESIS";

$con= mysqli_connect($host, $user, $pass, $base);

if (!$con) {
   die("conexion fallida" . mysqli_connect_error());
}
echo "conexionetosa";

$a = $_GET['Diascloro'];

$b = $_GET['Horacloro'];

$c = $_GET['Diasmotor'];

$d = $_GET['Horamotor'];

$e = $_GET['Diasquimicos'];

$f = $_GET['Horasquimicos'];

$g = $_GET['Calefaccion01'];

$h = $_GET['Temperatura'];

$i = $_GET['Diasluces'];

$j = $_GET['Horaencendidoluces'];

$k = $_GET['Horaapagadoluces'];

$l = $_GET['Alarma'];

$m = $_GET['Diasriego'];

$n = $_GET['Horariego'];

mysqli_query($con, "INSERT INTO Personalizado (Diascloro, Horacloro, Diasmotor,
Horamotor, Diasquimicos, Horaquimicos, Calefacion01, Temperatura, Diasluces,
Horaencendidoluces, Horaapagadoluces, Alarma, Diasriego, Horariego) VALUES('$a',
'$b', '$c', '$d', '$e', '$f', '$g','$h', '$i', '$j', '$k', '$l', '$m', '$n')");
echo "datos insertados";
echo $a;

?>
```