# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## IMPLEMENTACIÓN DE UN PROTOTIPO DE ALUMBRADO PÚBLICO INTELIGENTE

**TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES**

**KEVIN DAVID YÉPEZ ARTEAGA**

**DIRECTOR: ITALO ALEXANDER CARREÑO MENDOZA**

**DMQ, marzo 2023**

# CERTIFICACIONES

Yo, Kevin David Yépez Arteaga declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

**KEVIN DAVID YÉPEZ ARTEAGA**

**kevin.yepez01@epn.edu.ec**

**keyepez21@outlook.com**

Certifico que el presente trabajo de integración curricular fue desarrollado por Kevin David Yépez Arteaga, bajo mi supervisión.

**ITALO ALEXANDER CARREÑO MENDOZA**

**DIRECTOR**

**italo.carreno@epn.edu.ec**

# DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Kevin David Yépez Arteaga

# DEDICATORIA

Este proyecto de titulación quiero dedicar especialmente a mis padres Ximena y Mauricio y a mi hermana Fernanda, quienes fueron un pilar importante en todo este trayecto, brindándome su apoyo en cada momento y en cada paso y decisión que he tomado en el paso por la Universidad y ayudándome a levantar con cada tropiezo que he tenido en el transcurso.

*Kevin*

# AGRADECIMIENTO

Quiero expresar un sincero agradecimiento a mis padres Ximena y Mauricio por apoyarme en el transcurso de mi vida universitaria y en el proceso de realización de este proyecto de titulación, ya que sin ellos no habría podido superarme, salir adelante y crecer personal y profesionalmente.

A mi tutor de tesis Ing. Italo Carreño, el cual estuvo ayudando y respaldando continuamente en este proceso para hacer un buen trabajo.

A la Universidad, a la ESFOT, a sus docentes y en especial al Ing. Fernando Becerra quien a pesar de los conflictos que tuve en ultimas instancias, me ayudo y me apoyo en muchos aspectos para llegar a este punto.

# ÍNDICE DE CONTENIDOS

# RESUMEN

En el presente proyecto de titulación, Implementación de un Prototipo de Alumbrado Público Inteligente, el cual pretende crear un sistema de iluminación eficiente y autónomo el cual se adapte a las condiciones de luz natural existente en el área a implementarse para poder encenderse y apagarse automáticamente.

El proyecto esta divido en varias fases, entre ellas: la selección de componentes a utilizarse, tanto en software como hardware, el control de brillo de las luminarias dependiendo de la luz ambiental y el envío de los valores mediante una comunicación inalámbrica hacia una base de datos alojada en un servidor (*localhost)* para el monitoreo del brillo de las luminarias.

En la primera fase se da una introducción del proyecto, en la que se hace el planteamiento del problema para determinar los componentes que se usarán para la detección de luz ambiental, control de brillo de las luminarias y el monitoreo de la intensidad de luz de las luminarias, también se da a conocer los objetivos tanto generales como específicos los cuales ayudarán a determinar las tareas y el alcance del proyecto.

En la segunda fase, se trata sobre la metodología que será usada en el desarrollo del proyecto, esta sección se detallará cuáles son los métodos y procesos que permitirán la implementación y ejecución de los objetivos que han sido planteados para el presente proyecto.

En la tercera fase se desarrolla y se detalla los resultados que se obtuvieron en el desarrollo del proyecto, en esta sección se describen todos los requerimientos técnicos, la implementación, el diseño de todo el sistema, así como las pruebas del funcionamiento del proyecto y también se añade un video de la verificación del sistema en funcionamiento.

La cuarta y quinta sección contienen las conclusiones y recomendaciones que fueron obtenidas a partir del diseño y pruebas de funcionamiento del prototipo. Por último, en la sexta sección se detallan las referencias bibliográficas las cuales respaldan la implementación y desarrollo del sistema y del documento.

**PALABRAS CLAVE:** localhost, base de datos, comunicación inalámbrica, servidor.

# ABSTRACT

*In this degree project, Implementation of an Intelligent Public Lighting Prototype, we aim to create an efficient and autonomous lighting system that adapts to the existing natural light conditions in the area to be implemented, turning on and off automatically. The project is divided into several phases, including the selection of components to be used in both software and hardware, the control of the brightness of the luminaires depending on ambient light, and the sending of values through wireless communication to a database hosted on a server (localhost) for monitoring the brightness of the luminaires.*

*In the first phase, an introduction to the project is given, where the problem statement is presented to determine the components that will be used for ambient light detection, control of the brightness of the luminaires, and monitoring of the light intensity of the luminaires. The general and specific objectives are also presented, which will help determine the tasks and scope of the project.*

*The second phase discusses the methodology that will be used in the development of the project. This section will detail the methods and processes that will allow the implementation and execution of the objectives that have been set for this project.*

*The third phase develops and details the results obtained in the development of the project. This section describes all the technical requirements, the implementation, the design of the entire system, as well as the tests of the project's operation, and a video of the system verification in operation is also added.*

*The fourth and fifth sections contain the conclusions and recommendations that were obtained from the design and operation tests of the prototype. Finally, the sixth section details the bibliographic references that support the implementation and development of the system and the document.*



***KEYWORDS:*** *localhost, database, wireless communication, server.*

# 1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En el presente proyecto se desarrolló un sistema de alumbrado público inteligente, el cual cuenta con luminarias, cuya intensidad de luminosidad dependerá de la luz que incida sobre los sensores de igual manera, dependerá del rango predefinido en el código, para así controlar también su apagado automático. Los sensores que se utilizarán son los denominados **fotoresistores o LDR.**

El funcionamiento de los sensores se basa en la incidencia de luz sobre su superficie lo cual hace variar su resistencia para así controlar el voltaje que pase por dichos sensores.

Los valores de luminosidad se irán almacenando en una base de datos, los cuales serán mostrados en una tabla la cual contendrá todos los valores de luminosidad de cada una de las luminarias para después poder compararlos y obtener aquellos valores que hayan sido previamente configurados como muy altos o que superen el umbral configurado para que los valores puedan ser visualizados en una nueva tabla y mostrarlos como valores de alto riesgo ya que pueden dañar las luminarias por su alto nivel de luminosidad.

En la segunda tabla donde mostrarán las luminarias que superaron el umbral también se acompañaran con otros datos, como: la fecha en la que se obtuvo el umbral más alto, la luminaria que arrojo dicho valor y el valor de luminosidad.

El proyecto será desarrollado y programado con la ayuda de la plataforma IDE de Arduino, con la tarjeta de programación Arduino Uno y el módulo (shield) ESP8266 para conectar el Arduino Uno a Internet mediante WiFi y así subir los datos y crear una base de datos. Estas 2 tarjetas (Arduino Uno y ESP 8266) ayudará a desarrollar el código para el funcionamiento de los sensores conjunto con las luminarias las cuales serán simuladas mediante LEDs (Diodo Emisor de Luz).

Para subir los datos a Internet se hizo uso de la aplicación **XAMPP,** mediante la cual se puede alzar los servicios necesarios para visualizar la tabla en una página web (localhost), los servicios que se alzan son: **el servidor Apache y el protocolo *MySQL*.** Al momento de iniciar los 2 servicios en la aplicación XAMPP se puede acceder al sitio web desde cualquier buscador, dentro del sitio se puede crear varias tablas y una base de datos *(DB),* la cual contiene todas las tablas creadas.

Para acceder a las tablas y subir los datos obtenidos del Arduino se hizo uso de un código escrito en el lenguaje ***PHP*****,** en el cual se usó del protocolo ***mysql*****,** para conectarse con la base de datos así también para escoger a que tabla serán cargados los datos. Para la visualización de los datos se creó 2 páginas en código ***html***, una para la visualización de todos los datos de cada luminaria y otra en la cual serán mostrados solo los valores que cumplan con la condición de superar el umbral de luminosidad.

## 1.1 Objetivo general

Implementar un prototipo de alumbrado público inteligente.

## 1.2 Objetivos específicos

- Identificar los requerimientos para el dimensionamiento de un sistema de alumbrado público inteligente.
- Definir los componentes de *hardware* y *software* necesarios.
- Implementar el prototipo.
- Generar una base histórica de información.
- Realizar pruebas de funcionamiento del prototipo.

## 1.3 Alcance

A través de este proyecto se realizará el diseño de un sistema de alumbrado público inteligente que pueda controlar de forma autónoma la intensidad de iluminación en función de la luz solar presente. Además, se implementará el prototipo en una maqueta a pequeña escala utilizando Arduino y sensores LDR. La maqueta a implementar deberá contener al menos 5 luminarias.

En la identificación de requerimientos se analizará lo que el prototipo necesita para ser funcional. La definición de componentes abarcará una investigación acerca de las mejores opciones de *hardware* disponibles en el mercado que permitan implementar el prototipo.

La implementación del prototipo consistirá en la elaboración de la maqueta. En la generación de la base histórica de información se guardarán los datos obtenidos con los sensores en una base de datos para procesarlos y obtener aquellos que son demasiado altos, insertándolos en una tabla distinta que almacene la fecha, la luminaria y el nivel de iluminación producido. En las pruebas de funcionamiento se producirán distintos niveles de iluminación con el fin de verificar la correcta operación del prototipo, encendiendo las luminarias cuando sea necesario.

# Marco Teórico

## Arduino IDE

El Arduino *IDE (Integrate Development Environment)* es un editor de código abierto bajo la licencia pública ***GNU*** multiplataforma. Es una *app* utilizada para realizar códigos y cargarlos en tarjetas compatibles con el *IDE*, también se puede cargar a placas fabricadas por terceros debido a que se puede cargar las funciones de las placas desde las páginas oficiales de las mismas para usarlas con el mismo *IDE*. La plataforma admite 2 tipos de lenguaje: **C** y **C++,** para compilar y cargar en las placas de Arduino [1].

La plataforma cuenta con un administrador de tarjetas en el cual se puede buscar e instalar paquetes o núcleos para compilar y cargar los códigos a las placas. Así como el administrador de tarjetas también cuenta un administrador de bibliotecas que son extensiones de la aplicación que permite el control de sensores, motores o el uso de un módulo externo compatible con Arduino.

El *IDE* de Arduino tiene incorporado un monitor serial y un plotter serial**,** son herramientas que permiten visualizar o monitorear los valores que se envían desde la tarjeta de Arduino de sensores o voltajes, para la primera herramienta se hace uso del comando *Serial.print()* para la visualización de datos de forma escrita y con la segunda herramienta se pueden observar los valores de forma gráfica, con la nueva actualización se puede observar ambos al mismo tiempo debido a que el monitor serial ha sido incluido en el mismo editor [2].

## Placa Arduino Uno

Es una placa o tarjeta usada para cargar los códigos realizados en la aplicación *IDE* de Arduino.

La distribución de pines se muestra en la **Figura 1.1**. Cuenta con un conector USB para comunicarse con la computadora para cargar los programas al igual que se puede alimentar mediante esa misma conexión, la única limitante que se tiene alimentando mediante conexión USB es que simplemente puede suministrar hasta 500mA así que se usa simplemente para hacer pruebas básicas con componentes que no consuman demasiada corriente para su operación.


**Figura 1.1:** Distribución de pines [3].

El Arduino Uno se puede alimentar de otras formas, cuenta con un Jack de alimentación por el cual se puede suministrar un voltaje de **7 a 15 (VDC)** esto gracias a que incorpora un regulador para bajar la tensión a 5V que es el voltaje con el que opera la placa Arduino Uno. Otra forma de alimentar la tarjeta es mediante el pin **Vin** de la placa por la cual también se puede suministrar el mismo voltaje que por el Jack de DC.

Hay que tener en cuenta que, si se alimenta la tarjeta mediante el Jack o mediante el pin Vin, lo recomendable es alimentarlo desde los 7 a los 12 voltios, debido a que si se alimenta con menos de 7 (VDC) el regulador incorporado no suministrará los 5V con los que trabaja la tarjeta y se volverá inestable al momento de comenzar su funcionamiento. También si se suministra con más de 12 (VDC) el regulador incorporado se puede sobre calentar y dañar la placa.

La placa Arduino Uno cuenta con pines de comunicación (0-RX y 1-TX), los cuales pueden ser usados para la comunicación entre la computadora y el Arduino para la transmisión y recepción de datos al igual que se puede hacer una comunicación entre 2 arduinos haciendo una conexión entre los pines RX y TX de cada uno.

Los protocolos con los que cuenta para la comunicación son: comunicación en serie ***UART*****,** ***I2C*** y ***SPI*****.** En el presente proyecto la comunicación que se usará es la ***UART*** la cual esta explicada en la sección **Comunicación serial,** ya que se usarán simplemente los pines **0 (rx)** y **1 (tx)** para hacer la comunicación serial con el módulo ESP8266, teniendo en cuenta que las tierras (*GND*) de ambas tarjetas deben ser las mismas [4]**.**

### ESP8266

Es un módulo WiFi basado en un microcontrolador con soporte de la norma 802.11 y con los protocolos b/g/n de WiFi, soporta hasta la banda de 2,4 (GHz), cuenta con distintos modos de operación como: modo estación, punto de acceso y un modo promiscuo.

El modo de alimentación de esta placa es mediante el conector USB que viene incorporado en la placa, debido a que el voltaje suministrado al momento de conectar la placa a la PC es de 5 (V), gracias al regulador que viene incorporado en la misma placa, el voltaje baja a 3.3 (V) que es con el cual trabaja esta placa. Otra manera de alimentar al igual que el Arduino es mediante el pin de Vin**.**

En el proyecto se alimentará mediante el pin de 3.3 (V), los cuales serán suministrados de la tarjeta de Arduino del pin especifico de 3.3 (V) que esta junto al pin de 5 (V). La manera en la que se comunicará al igual que el Arduino será mediante los pines **tx** y **rx** con los que cuenta el **ESP8266,** la distribución de pines se puede observar en la **Figura 1.2.**

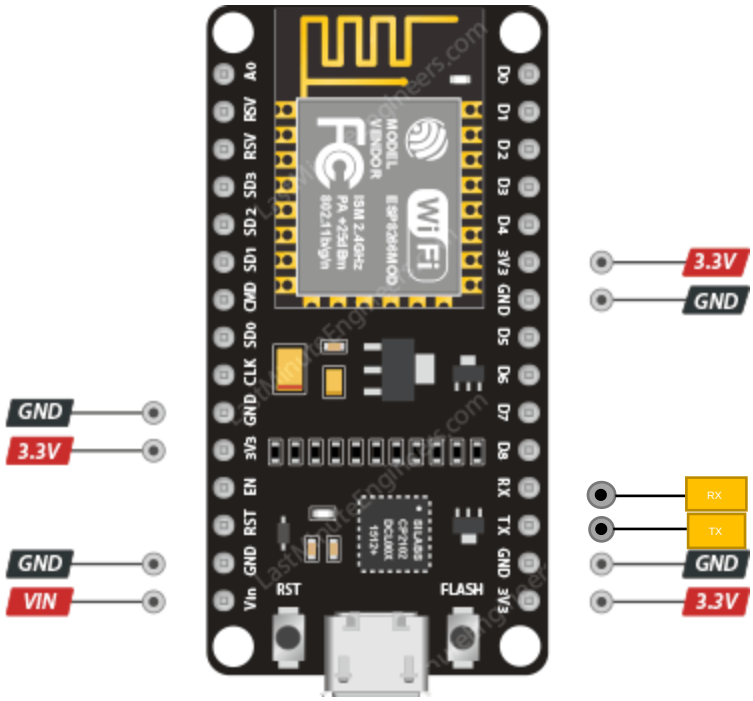



**Figura 1.2:** Distribución de pines del ESP8266 [5].

Los modos de operación del módulo WiFi son como: ***soft AP (Access Point) y Station.*** Cada modo tiene un funcionamiento distinto, en el modo *soft AP* se debe establecer un *SSID (Service Set Identifier)* y una clave para establecer una red WiFi y conectar más

*stations* al módulo. Cuando trabaja en modo *station* se debe proveer las credenciales de conexión de un *Access Point* para poder conectarse a él. En cuestión de seguridad y encriptación de la red el ESP8266 cuenta con **WPA/WPA2** y **WEP/TKIP/AES** respectivamente [6].

### LDR (Light Dependent Resistor)

Es un elemento electrónico el cual tiene la capacidad de variar el valor de su resistencia, lo cual depende de la cantidad de luz que incida sobre la superficie. El valor de su resistencia va entre los 50 y 100 (Ω) cuando la intensidad de luz sobre la superficie es alta y de varios MΩ cuando la cantidad de luz que incide sobre la superficie es baja.

Existen 2 tipos de construcciones para la superficie de la *LDR*, una son las construidas con sulfuro de cadmio como se observa en la **Figura 1.3** las cuales son sensibles a las radiaciones luminosas visibles y la cual se usó en el presente proyecto, y las segundas que están construidas con sulfuro de plomo la cuales son sensibles a las radiaciones infrarrojas.

**Figura 1.3:** LDR

Hay 2 tipos de conexiones que se pueden hacer con las LDR: la primera es, que a mayor luz, el voltaje también es mayor, para obtener este tipo de valores la conexión de la *LDR* debe ser, un pin de la *LDR* debe estar conectada a positivo de la fuente de voltaje y la segunda, que a mayor luz, el voltaje es menor, para este caso, la conexión de la *LDR* debe ser, un pin de la *LDR* debe estar conectada a negativo de la fuente de voltaje, para este proyecto se utilizará la última forma de conexión ya que con esta conexión se logró tener menos luminosidad sobre la *LDR* y la luminaria se encienda y viceversa, en ambos casos la *LDR* va junto con una resistencia en una configuración de divisor de tensión como se observa en la **Figura 1.4** [7].

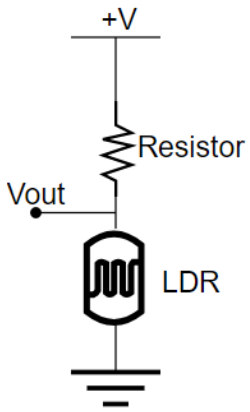



**Figura 1.4:** Configuración de LDR a negativo [8].

### MySQL

Es un sistema de base de datos de código abierto comercializada por ***MySQL AB,*** esto quiere decir que cualquiera puede usar y modificar el código según las necesidades que tenga. Una base de datos es una colección estructurada de los datos, en ella se puede almacenar un sin número de datos de forma separada y organizada, en una base de datos se puede tener múltiples tablas las cuales pueden ser unidas para establecer una relación entre las mismas y combinar sus datos [9].

Este *software* usa el lenguaje *SQL (Structured Query Languaje**)*** el cual es usado para interactuar con *MySQL* y muchas otras bases de datos para hacer peticiones, ingresar y obtener datos de las tablas [10].

*MySQL* tiene muchas ventajas en comparación a otras bases de datos, aquí se listan algunas [11]:

- Fiabilidad y rendimiento
- El *software* es de código abierto
- Es multiplataforma, puede ejecutarse en: Linux, *Windows*, etc.
- Fácil uso, ya que el *software* es muy fácil para aprender
- Tiene soporte de ***Oracle***

### *XAMPP*

Es un paquete de servidor local y gratuito de código libre. está disponible para multiplataformas (Windows, Linux y macOS). En el viene incluido un servidor ***HTTP***

***Apache,*** una base de datos (***MariaDB),*** e intérpretes de lenguajes de programación como ***PHP y Perl.***

Es muy utilizado para el desarrollo de *páginas web* y aplicaciones en un servidor local debido a que es fácil de usar, es útil ya que se puede probar y depurar de forma individual los proyectos antes de subirlos a un servidor web en vivo [12].

Al momento de iniciar la aplicación se tendrá una interfaz sencilla de usar, en la parte central como se observa en la **Figura 1.5**, se tendrá botones para interactuar con las distintas opciones que cuenta la aplicación como: iniciar ***(****Start****)*** ***o*** parar ***(****Stop****)*** los distintos servicios, así como la opción de entrar a la página de administración de cada servicio ***(****Admin****)*** y acceder a los distintos archivos de configuración de los mismos ***(****Config****)***.

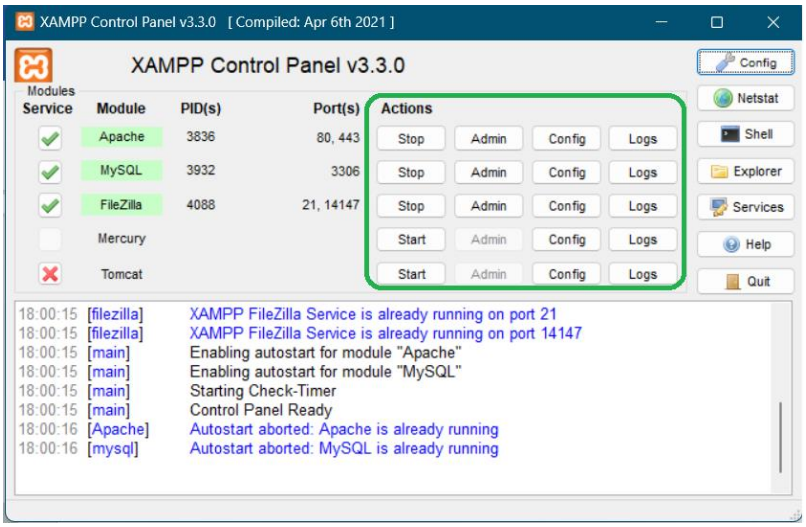



**Figura 1.5:** Botones para interactuar con los servicios.

En la parte derecha como se observa en la **Figura 1.6** existen algunos botones con opciones para acceder a herramientas del sistema como son: abrir una ventana del símbolo del sistema ***(****Shell****),*** abrir un analizador de trafico de red ***(****Netstat****),*** explorar por la carpeta de la aplicación ***(****Explorer****),*** etc.

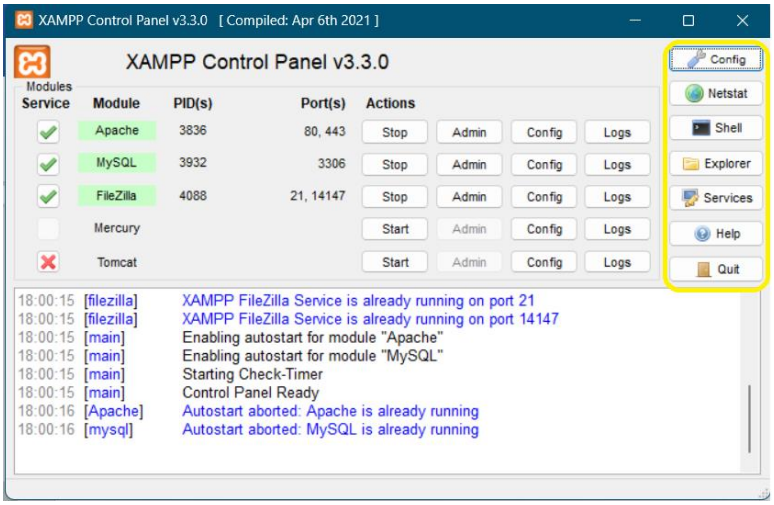



**Figura 1.6:** Acceso a herramientas del sistema.

### Comunicación serial

Es un protocolo de comunicación entre dispositivos para la transmisión y recepción de datos. Para usar este tipo de comunicación se hace uso de 3 líneas: **Tx, Rx y tierra o *GND*,** esta última tiene que ser común entre los 2 dispositivos que se va a realizar esta comunicación.

Existen 2 tipos de comunicación: comunicación síncrona y comunicación asíncrona, en el tipo de comunicación síncrona se tiene el protocolo de comunicación ***I2C*,** este protocolo hace uso de 2 líneas: ***SDA*** *(Serial Data)* y ***SCL*** *(Serial Clock)*. El otro protocolo de comunicación síncrona es el ***SPI*** *(Serial Peripheral Interface)***,** este protocolo hace uso de 4 líneas: ***MOSI*** *(Master data Output, Slave data Input)*, ***MISO*** *(Master data Input, Slave data Output)*, ***SCLK*** *(Serial Clock)* y **SS** *(Select Slave)*.

En el caso de la comunicación serial asíncrona ***UART*** (la cual se usó en el presente proyecto), este tipo de comunicación hace uso de 2 líneas: **Tx, Rx** y una tercera que es tierra **o *GND*,** esta última tiene que ser común entre los 2 dispositivos que se va a realizar esta comunicación, como se observa en la **Figura 1.7**.

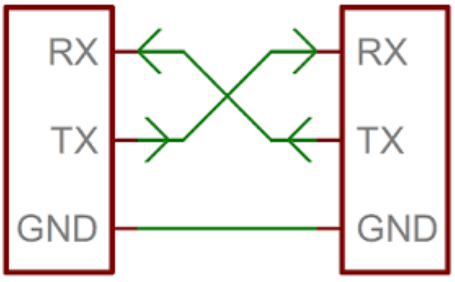



**Figura 1.7:** Conexión de pines ara comunicación UART.

Se hace uso de este tipo de comunicación en el proyecto debido a que es la forma más sencilla de implementación y también a que no se transmite una gran cantidad de datos simultáneamente. Al ser una comunicación asíncrona ninguna de los 2 dispositivos comparte un reloj en común para poder comunicarse, sin embargo, necesitan de un "reloj" configurado por *software* para hacer la transmisión y recepción de datos.

La unidad en la que se dan las velocidades de este "reloj" son los **baudios,** ambos deben tener la misma velocidad configurada para que los datos se enviados y recibidos correctamente por cada uno de los dispositivos, las velocidades en baudios más comunes o usadas son: 4800, 9600, 19200, 57600 y 115200 [13] [14].

# 2 METODOLOGÍA

Debido a que no en todos los lugares la luz natural o luz solar es igual o también pueden existir días en los que en la mañana o tarde los días son extremadamente nublados o incluso lluviosos, se ha hecho uso de los sensores *LDR* para detectar la incidencia de luz y así permitir encender la luminaria con una intensidad adecuada con respecto al clima.

Partiendo de esa información y el comprender que la luz solar no es uniforme durante el día, se realizó este proyecto, el diseño de un sistema de alumbrado público inteligente que controla de forma autónoma la intensidad de iluminación en función de la luz solar presente que incide sobre los sensores *LDR*.

Para la realización del sistema, se hizo una investigación previa para la elección de las técnicas de comunicación inalámbrica que permitirá el monitoreo y envió de información sobre la intensidad de iluminación de cada luminaria a una base de datos.

La utilización de este tipo de comunicación inalámbrica, admite la comunicación entre los dispositivos que se tiene en el sistema para tener un control de los datos de los distintos sensores y luminarias de manera remota. Los sensores que se usó permiten el monitoreo en tiempo real, debido a que el cambio de estado de estos es casi inmediato.

También, se consideró realizar el diseño de los diagramas de flujo, los cuales ayudan a dar un enfoque a las exigencias de los códigos de programación que se usó para las distintas etapas del proyecto. Antes de ser implementado de manera física se realizó una simulación mediante *software* del circuito a ser realizado posteriormente para probar

el funcionamiento correcto del código, así como del comportamiento del sistema a implementar.

Para la implementación física se la realizó en una maqueta, comprobando que todos los elementos del *hardware* se ajusten al diseño. También se consideró ciertos parámetros los cuales son primordiales como: la energización del sistema y el cableado de los distintos sensores y luminarias a implementarse en la maqueta.

Para la parte de la conexión y funcionamiento de los sensores al igual que las luminarias se usó el módulo Arduino Uno y para la parte de comunicación inalámbrica y subida de datos se usó el módulo ESP8266, el cual cuenta con un chip integrado para poder conectarse a Internet mediante WiFi. Además, en cuestión de *software* para la recolección de datos y creación de las tablas se usó el programa *Xampp.*

Con todo lo mencionado anteriormente, se desarrollaron los distintos códigos para cada etapa del proyecto partiendo de los diagramas de flujo y conjunto con el *hardware* del proyecto se logró crear el sistema de alumbrado público inteligente y el envío de datos hacia Internet.

Después de haber realizado el proyecto, se hicieron los ensayos necesarios de los sensores, luminarias y comunicación a Internet con la base de datos. Finalmente, se procedió a valorar el funcionamiento del sistema completo para identificar y corregir posibles fallos de calibración de los sensores que pudieran haberse presentado.

# 3 RESULTADOS

El proyecto está diseñado para que las luminarias se enciendan dependiendo del nivel de luz solar que incida sobra los sensores *LDR*. Una vez captada esa señal de los sensores, el Arduino es el encargado de leer esos datos para después poder tratarlos mediante funciones de programación y convertir esos datos a valores en los cuales trabaja la luminosidad de las luminarias.

Los mismos datos que se obtiene de los sensores son convertidos a valores de intensidad luminosa con la unidad de ***luxes (lx)***, esos valores son convertidos a texto y enviados mediante comunicación serial al ESP8266 en forma de *String* (cadena de texto).

En el ESP8266, esa cadena de texto es convertida nuevamente a valores enteros y separados por el valor de cada sensor para posteriormente poder enviarlos mediante Internet a la base de datos y cargarlos en la tabla principal, en la cual están todos los valores en general de cada luminaria, posteriormente, esos datos insertados en la tabla principal son filtrados mediante ciertas condiciones definidas para poder insertarlos en una segunda tabla. Estas 2 tablas son mostradas en una página web estática del servidor local ***(Xampp)*** mediante una interfaz gráfica creada en *HTML.*

## 3.1 Identificación de los requerimientos para el dimensionamiento del sistema

Para empezar, se hizo una investigación sobre la intensidad de luz solar que se tiene en un día normal para establecer qué tipo de sensor se iba a utilizar para la detección de luz solar. Uno de los principales factores que podrían afectar en la calibración y detección de luz solar es el clima (nublado, soleado o lluvioso), ya que debido a esto varía mucho la intensidad de luz solar.

Partiendo de ese punto se debería determinar valores en los cuales las iluminarias deberían empezar a encenderse con una intensidad de iluminación adecuada para así garantizar que pueda ser visible el lugar por donde se está transitando, al igual que para el apagado automático de las luminarias cuando no sea necesario o exista una buena iluminación en la zona.

Todos los componentes que se usarán, serán implementados en una maqueta, la cual será diseñada para alojar todos los componentes necesarios, tendrá compartimientos para hacer las conexiones de cada elemento a usarse en el proyecto.

Es necesario un sistema de control el cual ayudará con la función de lectura de datos de los sensores al igual que el encendido y apagado de las luminarias. Así también se debe considerar que el proyecto va a contar con luminarias que van a consumir un amperaje considerable del sistema de control debido a que serán luminarias con una gran potencia e intensidad de iluminación.

Estos sensores también serán energizados por el mismo sistema de control y debido a que están en una configuración de divisor de voltaje, el sistema de control debe proporcionar el voltaje adecuado para que dicho divisor de voltaje funcione correctamente y entregue los datos correctamente hacia el sistema de control, se debe considerar todos estos parámetros para escoger una fuente de alimentación, la cual

cumpla con todas estas demandas de corriente y voltaje para que el proyecto funcione correctamente.

Todos estos datos deben ser enviados y cargados a Internet, para esto se necesitará un módulo extra el cual ayude con la transmisión de los datos mediante comunicación inalámbrica (WiFi) y subirlos a Internet.

Teniendo ya el proyecto funcionando correctamente con todos los requerimientos necesarios, los datos deben ser visualizados y monitorizados remotamente, para esto es necesario la creación de una base de datos la cual recolectará todos los datos de cada una de las luminarias y las almacenará en una tabla para después tratar esos datos y tomar medidas dependiendo de la luminaria y el dato que se haya visualizado (sea grave o simplemente para información).

## 3.2 Componentes de *hardware* y *software* a utilizarse

Como principal componente para la detección de luz solar se usó un sensor *LDR* [7], el cual ayudará con la detección de niveles de luz solar para posteriormente procesar esos datos y encender los *LEDs* con una intensidad regulada, se escogió este sensor debido a que es uno de los más comunes, fáciles de configurar y económicos.

Este tipo de sensores son los más usados es circuitos de detección de luminosidad ya que estos varían su valor o resistencia en función de la luz que se detecte en la superficie, lo cual nos ayudará para poder tener un control de brillo de las luminarias.

Para la ayuda de la lectura de datos que envían los sensores al igual que para el encendido de los *LEDs*, se utilizó un ARDUINO UNO [3], debido a que este contaba con los pines necesarios (entradas analógicas para los *LDR* y salidas *PWM* para el control del brillo del *LED*), al igual que 2 pines para la comunicación serial.

Se usó este Arduino, debido a que tiene los pines necesarios para las entradas analógicas y las salidas *PWM,* en comparación al Arduino Nano, el cual solo cuenta con 4 salidas *PWM* y no se usó el Arduino Mega debido a su tamaño y también a su precio, el cual es más caro que el Arduino Uno.

Para establecer la comunicación inalámbrica, el envío de valores hacia la base de datos y la comunicación serial con el Arduino, se utilizó el módulo ESP8266 [5], el cual cuenta con la característica para la conexión a Internet mediante WiFi. Se usó este módulo debido a que cuenta con una gran ventaja en comparación a tecnologías inalámbricas como *Bluetooth y ZigBee,* ya que el alcance para la comunicación entre *hardware*

*(Sensores, ESP8266, Arduino UNO) y software (Xampp)* es uno de los factores más importantes, también, admite el envío de datos sin la necesidad de una red cableada.

Se eligió esta placa ya que como se mencionó, cuenta con la característica principal de poder conectarnos a Internet mediante comunicación inalámbrica (WiFi) en comparación con el módulo ESP32, se escogió el ESP8266 por ser más económico y debido a que simplemente se necesita la característica de conexión a Internet.

Para la recolección y visualización de los datos enviados por el ESP8266, se hizo uso de un programa de PC llamado *XAMPP* [12], el cual ayuda a la creación de un servidor local con la facilidad para administrar una base de datos en la cual se puede crear distintas tablas las cuales contendrán los valores de cada luminaria.

Este programa es muy útil debido a que se puede crear una base de datos sin la necesidad de saber programación gracias a que cuenta con las herramientas necesarias como un servidor Apache y un servidor *MySQL* para el levantamiento de un *localhost* y también se puede hacer intercambio de datos entre distintas tablas.

## 3.3 Implementación del prototipo

Una vez ya seleccionados todos los componentes, tanto en *software* como *hardware*, se procedió con la implementación del prototipo del sistema de alumbrado público inteligente.

Para la detección de luz solar, se hizo una calibración en el sensor *LDR*, el cual detectará el cambio de luz solar y así encenderá las luminarias con un control de su intensidad de brillo según los valores establecidos en el código de Arduino, debido a que los valores que entrega este sensor va en un rango de 0 a 1023, se limitó estos valores en el código a un rango que empieza en 280 y termina en 1023, como se muestra en la **Figura 3.1** ya que se realizó pruebas con el sensor con diferentes tipos de incidencia de luz solar y se determinó que al valor de 280 el sistema funcionaría correctamente ya que en un rango inferior el apagado de las luces se haría aún cuando no exista mucha luz ambiental y dificultaría la visibilidad del entorno en el cual está la luminaria.

```
23   //Variables para restringir la intensidad
24   int max = 1023;
25   int min = 280;
```

**Figura 3.1:** Valores restringidos de los sensores.

### Método de petición HTTP GET

La función *GET* es utilizada para obtener valores que se envían mediante una petición *HTTP GET.* La petición *HTTP GET* es una solicitud que se la realiza a servidores *web* para obtener datos, valores, etc, y estos parámetros se pasan mediante la *URL* de la solicitud [15].

Mediante el uso de este método, se realizó el código *PHP,* el cual fue desarrollado en la aplicación *Sublime Text*, el código empieza con la función *error_reporting()* como se observa en la **Figura 3.2***,* la cual es utilizada para configurar el nivel de informe de errores del archivo *(script),* para este caso se estableció para informar todos los errores excepto las **advertencias.**

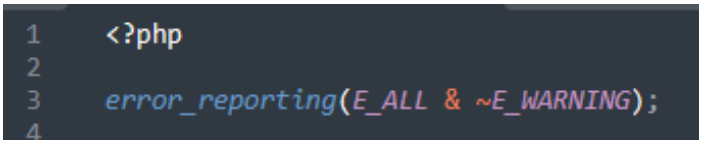



**Figura 3.2:** Nivel de informe errores del script.

Las siguientes 6 líneas que se observan en la **Figura 3.3**, son usadas para asignar los valores de los parámetros *ldr1 a ldr6,* dichos valores corresponden a los sensores *LDR,* los cuales son enviados mediante la petición *HTTP GET* a las variables *$ldr1 a $ldr6.*

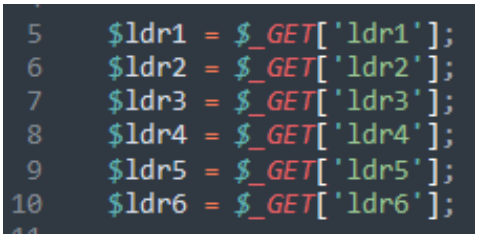



**Figura 3.3:** Asignación de los valores de los LDR.

Después, se establece conexión a la base de datos en *MySQL* mediante 4 variables *($usuario, $contraseña, $servidor y $basededatos)*, para el proyecto se usó el usuario y contraseña que trae por defecto la base de datos, las cuales se puede muestra en la **Figura 3.4**. Estas variables son usadas más adelante para establecer conexión a la base de datos y realizar algunas operaciones con ellas, como la de escoger la base de datos a la cual se conectará para subir los valores.

```
$usuario = "root";
$contraseña = "";
$servidor = "localhost";
$basededatos = "esp8266ldr";
```

**Figura 3.4:** Usuario, contraseña y nombre de la base de datos.

Posteriormente se realiza la conexión a la base de datos llamada *"esp8266ldr",* en caso que no se haya podido establecer la conexión con dicha base de datos se mostrará el siguiente mensaje **"No se ha podido seleccionar la Base de Datos",** como se observa en la **Figura 3.5.**

```
$conexion = mysqli_connect ( $servidor, $usuario, "") or die ("No se ha podido conectar con el
    servidor");

$db = mysqli_select_db ($conexion, $basededatos) or die ("No se ha podido seleccionar la Base de Datos
    ");
```

**Figura 3.5:** Conexión con la base de datos.

A continuación, se utiliza la función *INSERT INTO* como se observa en la **Figura 3.6**, para insertar los datos o lecturas de las *LDR* en una tabla llamada **"historicoldr"**, al igual que se añade una variable más que mostrará la fecha y hora en la que se insertó dicho valor en la tabla. Por último, se realiza una consulta a la base de datos con la función *mysqli_query()* con 2 parámetros: la conexión a la base de datos ($conexion) y la consulta *SQL* o la inserción de los datos a la tabla **(**$consulta**).**

```
$consulta = "INSERT INTO `historicoldr` (`id`, `ldr1`, `ldr2`, `ldr3`, `ldr4`, `ldr5`, `ldr6`, `Fecha`
    ) VALUES (NULL, '$ldr1', '$ldr2', '$ldr3', '$ldr4', '$ldr5', '$ldr6', CURRENT_TIMESTAMP)";

$resultado = mysqli_query( $conexion, $consulta );
```

**Figura 3.6:** Inserción de los datos en la tabla principal.

Como resultado de esta función se tiene los valores que se insertarán en la tabla, o en caso de que la resolución de esta función falle dará un valor de *false*, que significa que la consulta falló y no se insertaron los datos correctamente. El código completo de la conexión con la base de datos mediante petición *HTTP GET,* se observa en la **Figura 3.7.**

```php
<?php

error_reporting(E_ALL & ~E_WARNING);

$ldr1 = $_GET['ldr1'];
$ldr2 = $_GET['ldr2'];
$ldr3 = $_GET['ldr3'];
$ldr4 = $_GET['ldr4'];
$ldr5 = $_GET['ldr5'];
$ldr6 = $_GET['ldr6'];


$usuario = "root";
$contraseña = "";
$servidor = "localhost";
$basededatos = "esp8266ldr";

$conexion = mysqli_connect ( $servidor, $usuario, "") or die ("No se ha podido conectar con el
    servidor");

$db = mysqli_select_db ($conexion, $basededatos) or die ("No se ha podido seleccionar la Base de Datos
    ");

$consulta = "INSERT INTO `historicoldr` (`id`, `ldr1`, `ldr2`, `ldr3`, `ldr4`, `ldr5`, `ldr6`, `Fecha`
    ) VALUES (NULL, '$ldr1', '$ldr2', '$ldr3', '$ldr4', '$ldr5', '$ldr6', CURRENT_TIMESTAMP)";

$resultado = mysqli_query( $conexion, $consulta );

echo "Datos enviados correctamente. ";


?>
```

**Figura 3.7:** Código mediante la petición HTTP GET

### Flujograma del código de la placa Arduino UNO

Para empezar, nos basamos en el flujograma que se muestra en la **Figura 3.8**, se empieza definiendo y declarando las variables necesarias para los sensores, las luminarias y el almacenamiento de todos los valores de los sensores, tanto los valores crudos (sin procesar), variables para la conversión a luxes, así como los valores que se enviarán al módulo ESP8266, también se cuenta con una variables para hacer de contador las cuales se resetearan cada vez que la función *loop()* cumpla con todas las condiciones que se han establecido.

En la función del *setup(),* se inicializa la comunicación serial definiendo a qué velocidad (baudios) se van a transmitir los datos y se llama a las variables anteriormente definidas para declararlas como entradas o salidas respectivamente.

En la función *loop(),* se empieza llamando a la variable **“**seg**”** la cual es la encargada de hacer de contador para posteriormente ser comparada mediante un condicional y enviar los datos.

Después se llama a las funciones (*ldr_1()* hasta *ldr_6()*) las cuales son las encargadas de leer los valores de los sensores, restringir los valores a un mínimo y máximo como se explicó en el párrafo anterior, se mapea o se convierte los valores leídos y restringidos a valores en los cuales se encenderán las luminarias, se hace una operación matemática para que los valores leídos de los sensores puedan ser convertidos a luxes dependiendo de la luminosidad de las luminarias, por último, estos valores en luxes también son restringidos a un rango de 0 a 100 luxes.

Después se hace uso de un condicional, el cual permite encender las luminarias dependiendo si los valores leídos de los sensores están dentro del rango establecido en el condicional y así encender las luminarias con un control de brillo haciendo uso de los pines *PWM* y, si está fuera de ese rango, que las luminarias permanezcan apagadas.

Por último, se hace uso de otra función condicional la cual espera por la variable que se inicializó al inicio de la función *loop()* (seg()), y como esta empieza en un número preestablecido y va cuenta atrás, se compara si esa variable ha llegado a cero, antes de enviar los valores, debido a que estos están declarados como enteros *(int),* se hace uso de una función llamada *String(),* la cual permite convertir los números a una cadena de caracteres y son separados mediante una coma y almacenados en otra variable definida como *String()* para enviar eso valores convertidos a luxes mediante comunicación serial hacia el ESP8266.

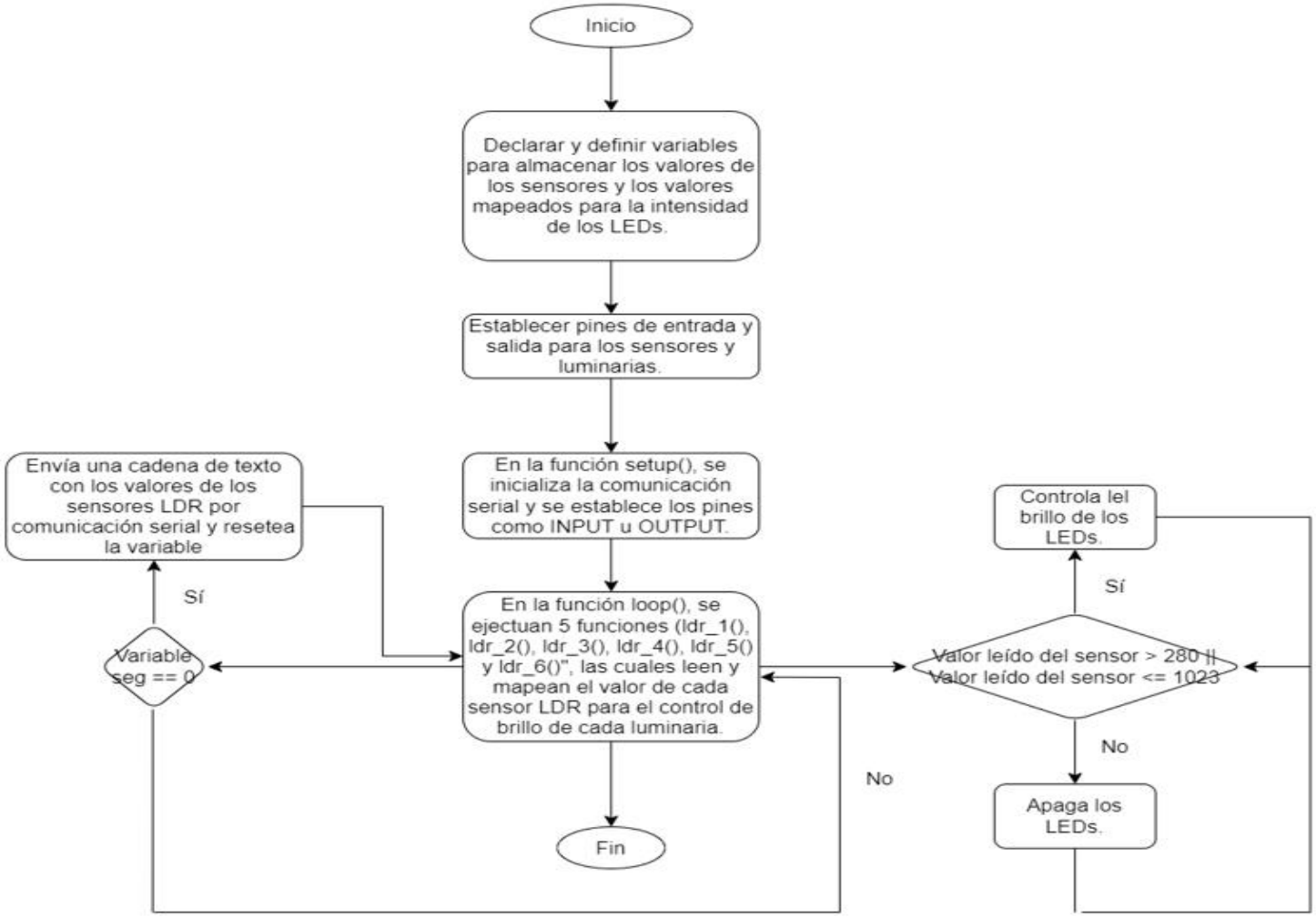



**Figura 3.8:** Flujograma del código de Arduino.

### Distribución de pines de la placa Arduino UNO

Los pines que fueron usados en la tarjeta Arduino UNO, se muestran en la **Figura 3.9**. Para el presente proyecto los pines que se usarán serán:

- Los 6 pines analógicos con los que cuenta para la conexión de las *LDR* (pin 14 al 19).
- Los 6 pines con la función de *PWM* para controlar la intensidad de la luminaria (pin 3 – 5 – 6 – 9 al 11).
- Y los 2 pines de comunicación *UART* (tx “pin 1”– rx “pin 0”).

Como se puede observar, se utilizaron los 6 pines analógicos con los que cuenta la placa, en los cuales fueron conectados todos los sensores *LDR.* Para las luminarias se utilizó también los 6 pines digitales con salida *PWM*, debido a que con estos pines se puede controlar la intensidad de brillo de las luminarias.

Para la conexión entre el Arduino UNO y el ESP8266, se usa los 2 pines de comunicación serial *(rx/tx)* que corresponden a los pines 0 y 1 del Arduino respectivamente. Para energizar los sensores y luminarias se utilizó la salida de 5 (V), con la que cuenta el Arduino al igual que tierra o *GND,* también se utilizó el pin de *GND* de Arduino.

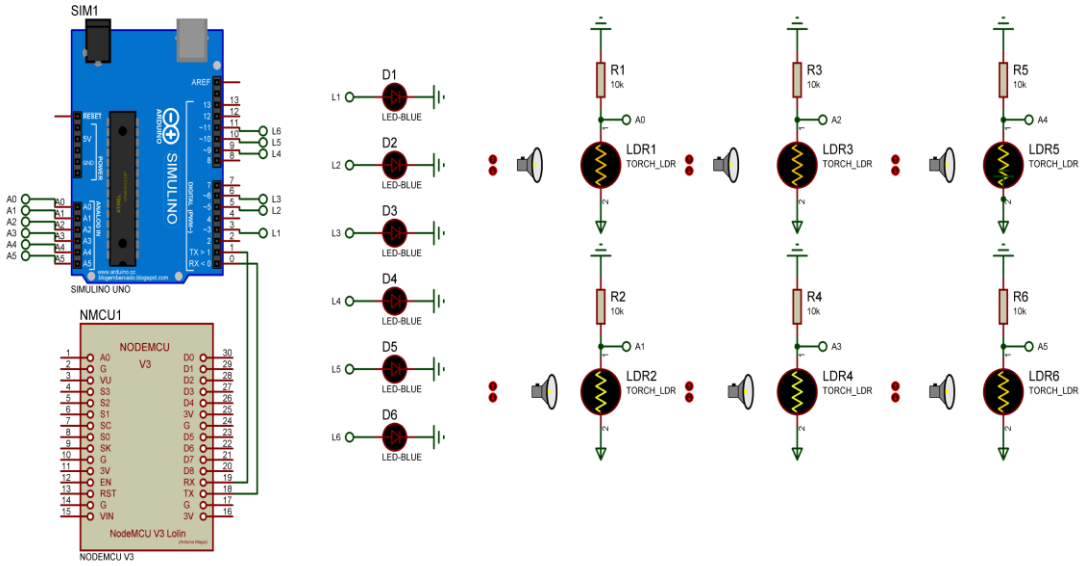



**Figura 3.9:** Distribución de pines de la placa Arduino UNO

### Distribución de pines del módulo ESP8266

El módulo ESP v.3 cuenta con pines para ser alimentando externamente, de los cuales se tomaron 2 pines: el de **3.3** (V) y *GND.* Este será energizado desde el pin de 3.3 (V) que cuenta el Arduino UNO y la tierra o *GND* será la misma con la que cuenta el Arduino debido a que como se explicó en la sección de **Comunicación serial,** la tierra o *GND* debe ser el mismo para establecer correctamente la comunicación asíncrona. Al igual se utilizó los pines de *rx/tx* para establecer la comunicación con el Arduino, como se muestra en la **Figura 3.10**.

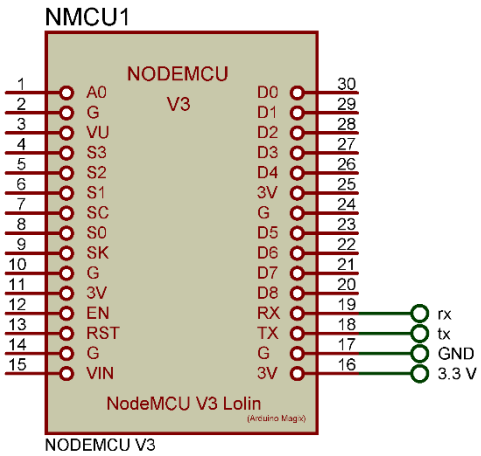



**Figura 3.10:** Pines usados en el ES8266.

### Flujograma del código del módulo ESP8266

El módulo ESP8266 ayudará con la comunicación inalámbrica y envío de datos hacia la base de datos. El código del módulo inicia incluyendo las bibliotecas necesarias para la conexión mediante *WiFi* del módulo y para la conexión con el servidor. Después, se declara algunas variables necesarias, tanto para la conexión con el *router* y para poder conectarse con la base de datos. También se declararon variables necesarias para recibir el *String* enviado desde la placa Arduino y variables para almacenar y posteriormente enviar cada valor de las *LDR* hacia la base de datos.

En la parte de la función *setup(),* primero se inicializa la velocidad a la cual estará trabajando la comunicación serial, al igual que en la placa Arduino, se la inicializa a la misma velocidad para sincronizarlos correctamente, después se inicializa la librería que permitirá la conexión a Internet mediante *WiFi y se* muestra un mensaje mediante monitor serial mencionando que el módulo se ha conectado a la red.

En la función *loop(),* se inicia con un condicional*,* el cual comprueba si hay datos disponibles en el puerto serial. La siguiente línea, lee los datos recibidos en el puerto serial, hasta encontrarse con un salto de línea. Se separa cada valor que existe el cual está separado por una **coma.** Este proceso se repite 6 veces, debido a que son 6 los valores que deben ser separados del *String* general.

Finalmente, se establece conexión con el servidor, se construye la *URL* que se utilizará para enviar los datos**.** Por último, se lee la respuesta del servidor si los datos fueron enviados correctamente y se muestra en el monitor serial. El flujograma del código explicado se muestra en la **Figura 3.11**

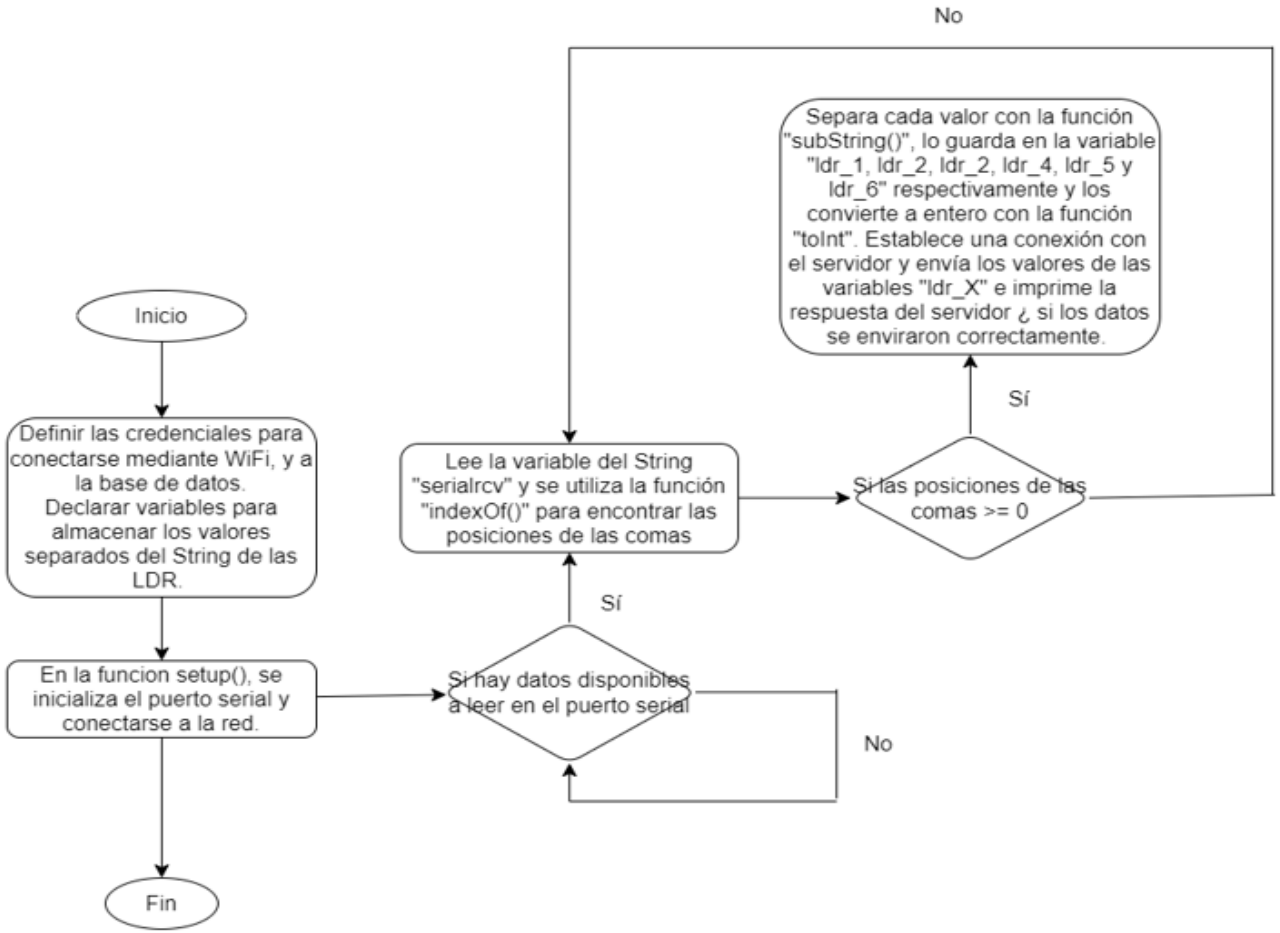



**Figura 3.11:** Flujograma del código del módulo ESP8266.

### Flujograma del circuito completo

En esta sección se explicará de manera más detallada el código de cada tarjeta utilizada (Arduino y ESP8266), así como el código para la conexión con la base de datos, la creación de las tablas usadas y la inserción de los datos en cada tabla. Para empezar a realizar los diagramas de flujo, como se menciona en la sección de **Componentes de *hardware* y *software* a utilizarse,** fue necesario la identificación de los componentes a utilizarse, como son: Arduino Uno, *LDR,* módulo ESP8266 y el programa que ayudará con el levantamiento de la base de datos mediante un *localhost.* El flujograma completo se puede observar en la **Figura 3.32.**

#### Explicación del código de Arduino

Se empieza definiendo variables para un contador, el cual será el encargado de dar cierto tiempo para que los datos sean enviados hacia el módulo ESP8266, variables para:

- Guardar las lecturas de los sensores *LDR* y para definir los pines en los que serán conectados.
- Valores mapeados para la intensidad de las luminarias.
- Pines en los cuales serán conectadas las luminarias.
- Valores para convertir la lectura de los sensores *LDR* a *luxes.*
- Guardar los valores convertidos en *luxes.*
- Valores a enviar mediante comunicación serial.

Cabe aclarar que la lectura de los sensores *LDR* son mapeados debido a que, estos van en un rango de 0 a 1023 y las luminarias trabajan en un rango de 0 a 255 para controlar su intensidad, esto se debe a que la resolución que se puede enviar mediante la función *analogWrite()* es de **8 *bits*** lo cual da un rango de 0 a 255, por otro lado, los valores que pueden leer los pines analógicos tienen una resolución de **10 *bits,*** lo cual da un rango de 0 a 1023. La declaración de variables del código de Arduino se puede observar en las **Figura 3.12 y Figura 3.13.**

```
1   //Variables reseteo de funcion millis
2   extern volatile unsigned long timer0_millis;
3   unsigned long new_value = 15;
4   int seg = 0;
5
6   //Variables para guardar los valores del LDR
7   int LDR = 0;
8   volatile int valorldr1 = 0;
9   volatile int valorldr2 = 0;
10  volatile int valorldr3 = 0;
11  volatile int valorldr4 = 0;
12  volatile int valorldr5 = 0;
13  volatile int valorldr6 = 0;
14
15  //Variables mapeadas para la intensidad de los LEDs
16  volatile int l1 = 0;
17  volatile int l2 = 0;
18  volatile int l3 = 0;
19  volatile int l4 = 0;
20  volatile int l5 = 0;
21  volatile int l6 = 0;
22
23  //Variables para restringir la intensidad
24  int max = 1023;
25  int min = 280;
26
27  //Variables de las iluminarias
28  const int Lum1 = 3;
29  const int Lum2 = 5;
30  const int Lum3 = 6;
31  const int Lum4 = 9;
32  const int Lum5 = 10;
33  const int Lum6 = 11;
```

**Figura 3.12:** Declaración de variables en Arduino.

```
35  //Variables para transformar a luxes
36  float m = 0.13; //Pendiente de la recta
37  volatile float b = -37.31; //Intercepto
38  int maxled = 100;
39  int minled = 0;
40  volatile int lux1 = 0;
41  volatile int lux2 = 0;
42  volatile int lux3 = 0;
43  volatile int lux4 = 0;
44  volatile int lux5 = 0;
45  volatile int lux6 = 0;
46
47  //Variable a transmitir
48  String valores;
```

**Figura 3.13:** Declaración de variables Arduino. Parte 2

En la función del *void setup(),* se inicializa la comunicación serial con una velocidad de 9600 baudios (la cual tiene que ser la misma velocidad de transmisión y recepción en

las tarjetas involucradas en este tipo de comunicación) y se definen como entradas y salidas los pines de los sensores y las luminarias, respectivamente como se observa en la **Figura 3.14**.

```
50  void setup() {
51    Serial.begin(9600);
52    for (LDR = 14; LDR <=19; LDR++){
53      pinMode(LDR, INPUT);
54    }
55    pinMode(Lum1, OUTPUT);
56    pinMode(Lum2, OUTPUT);
57    pinMode(Lum3, OUTPUT);
58    pinMode(Lum4, OUTPUT);
59    pinMode(Lum5, OUTPUT);
60    pinMode(Lum6, OUTPUT);
61
62  }
```

**Figura 3.14:** Función void setup().

En la función *void loop(),* primero se inicia el contador y se ejecutan 6 funciones, las cuales están fuera de la función *void loop(),* esto es debido a que estas funciones pueden ser utilizadas en cualquier parte del programa, también para optimizar la memoria del Arduino y evitar que el programa corra lento y para que el código sea más organizado y entendible.

Cada una de estas funciones leen los sensores conectados en los pines analógicos, restringen el valor de lectura para empezar desde **280** como se explicó en la sección **Implementación del prototipo,** mapean o escalan la lectura de los sensores en el rango de 0 a 255 para las luminarias como se explicó anteriormente, convierten la lectura de los sensores a *luxes* y ese valor es restringido a valores en los cuales las luminarias trabajan que es de 0 a 100 para luminarias *LED.*

Cada función está acompañada con un condicional el cual si se cumple ejecuta lo que está dentro de los paréntesis, en este caso, el condicional sirve para encender y apagar automáticamente las luminarias y lo que se ejecuta dentro del paréntesis es para encender las luminarias con un control de su brillo.

Por último, se compara si el contador llegó a **0** para construir un *String* o cadena de datos con los valores convertidos a luxes de cada luminaria los cuales están separados cada uno por una coma y enviarlos hacia el ESP8266, como se muestra en las **Figura 3.15 y Figura 3.16.**

```
64  void loop() {
65    seg = new_value - (millis() / 1000);
66
67    ldr_1();
68    if (valorldr1 > 280 || valorldr1 <=1023) {
69    | analogWrite (Lum1, l1);
70    }
71    else {
72    | digitalWrite (Lum1, LOW);
73    }
74
75    ldr_2();
76    | if (valorldr2 > 280 || valorldr2 <= 1023) {
77    | analogWrite (Lum2, l2);
78    }
79    | else {
80    | digitalWrite (Lum2, LOW);
81    }
82
83    ldr_3();
84    if (valorldr3 > 280 || valorldr3 <= 1023) {
85    | analogWrite (Lum3, l3);
86    }
87    else {
88    | digitalWrite (Lum3, LOW);
89    }
90
91    ldr_4();
92    if (valorldr4 > 280 || valorldr4 <= 1023) {
93    | analogWrite (Lum4, l4);
94    }
95    else {
96    | digitalWrite (Lum4, LOW);
97    }
```

**Figura 3.15:** Función void loop().

```
99   ldr_5();
100  | if (valorldr5 > 280 || valorldr5 <= 1023) {
101  | analogWrite (Lum5, l5);
102  }
103  else {
104  | digitalWrite (Lum5, LOW);
105  }
106
107  ldr_6();
108  | if (valorldr6 > 280 || valorldr6 <= 1023) {
109  | analogWrite (Lum6, l6);
110  }
111  else {
112  | digitalWrite (Lum6, LOW);
113  }
114
115  //Valores a transmitir y reseteo del contador
116  if (seg == 0){
117  | valores = String(lux1) + "," + String(lux2) + "," + String(lux3) + "," + String(lux4) + "," + String(lux5) + "," + String(lux6);
118  | Serial.println(valores);
119  | setMillis(new_value);
120  }
121 }
```

**Figura 3.16:** Función void loop() parte 2.

### Explicación del código del módulo ESP8266

Se empieza definiendo algunas de las librerías como se observa en la **Figura 3.17** que son necesarias para poder conectarse a una red WiFi configurada en una lista y otra librería que permitirá la conexión con la base de datos a través una petición *HTTP GET*, que fue descrita en la sección **Método de petición HTTP GET.**

```
1   #include <ESP8266HTTPClient.h>
2   #include <ESP8266WiFiMulti.h>
```

**Figura 3.17:** Librerías del módulo ESP8266.

A continuación, se definen las credenciales como el *SSID* y la *password* del *router* al cual se conectará el ESP8266 para acceder a Internet, al igual que algunos campos para la conexión con el servidor como el *host* y el puerto en el cual trabaja el *localhost.* También hay variables para:

- Recibir el *String* del Arduino
- Guardar la posición en la que se encuentra el carácter de **coma.**
- Guardar el *substring* resultante de la separación del *String* principal.
- Guardar cada valor separado del *substring.*

Cada una de estas variables se puede observar en la **Figura 3.18.**

```
4   // defino credenciales red
5   static const char* ssid = "INTERNET_CNT YEPEZ";
6   static const char* password = "Titokeluckyfer12345";
7
8   //Variables para conexión con el servidor
9   const char* host = "192.168.1.12";
10  const int port = 80;
11
12  //Variables a enviar
13  String serialrcv;
14  String sub_str;
15  int ldr_1 = 0;
16  int ldr_2 = 0;
17  int ldr_3 = 0;
18  int ldr_4 = 0;
19  int ldr_5 = 0;
20  int ldr_6 = 0;
21
22  //Variables para separar el String
23  int coma1 = 0;
24  int coma2 = 0;
25  int coma3 = 0;
26  int coma4 = 0;
27  int coma5 = 0;
```

**Figura 3.18:** Variables a usarse en el ESP8266.

En la función del *void setup(),* al igual que en el Arduino se empieza definiendo la velocidad de recepción de los datos, la cual debe ser igual para que los datos sean recibidos correctamente, que este caso se usó 9600 *baudios.* Se inicializa una de las instancias de la librería #***include <ESP8266WiFiMulti.h>*****,** la cual permitirá realizar la conexión con nuestro *AP (Access Point)* o *router,* para acceder a Internet, teniendo conexión a Internet se mostrará un mensaje en el monitor serial del nombre de la red a la cual se conectó al igual que la *IP* que se le asignó al dispositivo, como se observa en la **Figura 3.19.**

```
29  void setup() {
30    Serial.begin(9600);
31    while(!Serial){;
32    }
33
34    WiFi.begin(ssid, password);
35    Serial.print("Conectando...");
36    while (WiFi.status()!= WL_CONNECTED) {
37      delay(500);
38      Serial.print(".");
39    }
40    Serial.println();
41    Serial.print("Conectado a: ");
42    Serial.println(ssid);
43    Serial.print("IP Local: ");
44    Serial.println(WiFi.localIP());
45    Serial.println();
46  }
```

**Figura 3.19:** Función void setup() ESP8266.

En la función *loop(),* se inicia con un condicional *(if()),* el cual comprueba si hay datos disponibles en el puerto serial, si no existen datos la condición no se cumple y las líneas siguientes no son ejecutadas. La siguiente línea, lee los datos recibidos en el puerto serial, hasta que se encuentra con el carácter de nueva línea **(\n)** y estos datos son almacenados en la variable *serialrcv.*

Lo siguiente es procesar el *String*, para separar cada valor que existe el cual está separado por una **coma,** para estos se hizo uso de la función *indexOf() y substring(),* las cuales ayudan a encontrar la posición de la existencia de una **coma** en el *String* para separarlo y almacenarlos en una variable respectivamente, esta es convertida a un número entero y almacenada en una nueva variable la cual será enviada posteriormente.

Este proceso se repite 6 veces, debido a que son 6 los valores que deben ser separados del *String* general.

Finalmente, se declara una instancia del objeto *WiFiClient* de la librería *HTTPClient,* el cual será usado para establecer conexión con el servidor, después se construye la *URL* que se utilizará para enviar los datos mediante la solicitud *GET HTTP,* la cual fue explicada en la sección **Método de petición HTTP GET.**

Por último, se lee la respuesta del servidor si los datos fueron enviados correctamente y se muestra en el monitor serial. Esta parte del código se puede observar en las **Figura 3.20, Figura 3.21 y Figura 3.22**.

```
48  void loop() {
49    if (Serial.available()){
50      serialrcv = Serial.readStringUntil ('\n'); //creamos y llenamos buffer hasta un "enter"
51      Serial.println (serialrcv);
52
53      coma1 = serialrcv.indexOf(",");
54
55      if (coma1 >= 0){
56        sub_str = serialrcv.substring(0, coma1);
57        ldr_1 = sub_str.toInt();
58        Serial.print("LDR 1: ");
59        Serial.println(ldr_1);
60
61        coma2 = serialrcv.indexOf(",", coma1 + 1);
62
63        if (coma2 >= 0){
64          sub_str = serialrcv.substring(coma1 + 1, coma2);
65          ldr_2 = sub_str.toInt();
66          Serial.print("LDR 2: ");
67          Serial.println(ldr_2);
68
69          coma3 = serialrcv.indexOf(",", coma2 + 1);
70
71          if (coma3 >= 0){
72            sub_str = serialrcv.substring(coma2 + 1, coma3);
73            ldr_3 = sub_str.toInt();
74            Serial.print("LDR 3: ");
75            Serial.println(ldr_3);
76
77            coma4 = serialrcv.indexOf(",", coma3 + 1);
78
79            if (coma4 >= 0){
80              sub_str = serialrcv.substring(coma3 + 1, coma4);
81              ldr_4 = sub_str.toInt();
82              Serial.print("LDR 4: ");
```

**Figura 3.20:** Función void loop() ESP8266.

```
83                  Serial.println(ldr_4);
84
85                  coma5 = serialrcv.indexOf(",", coma4 + 1);
86
87                  if (coma5 >= 0){
88                    sub_str = serialrcv.substring(coma4 + 1, coma5);
89                    ldr_5 = sub_str.toInt();
90                    Serial.print("LDR 5: ");
91                    Serial.println(ldr_5);
92
93                    sub_str = serialrcv.substring(coma5 + 1);
94                    ldr_6 = sub_str.toInt();
95                    Serial.print("LDR 6: ");
96                    Serial.println(ldr_6);
97
98
99                    WiFiClient client;
100
101                   if (!client.connect(host, port)){
102                     Serial.println("No se pude conectar al servidor");
103                     return;
104                     }
105                   String url = "/ESP8266ldr/enviodatos.php?ldr1=";
106                   url += ldr_1;
107                   url += "&ldr2=";
108                   url += ldr_2;
109                   url += "&ldr3=";
110                   url += ldr_3;
111                   url += "&ldr4=";
112                   url += ldr_4;
113                   url += "&ldr5=";
114                   url += ldr_5;
115                   url += "&ldr6=";
116                   url += ldr_6;
```

**Figura 3.21:** Función void loop() ESP8266. Parte 2

```
118                   //Envio de petición al servidor
119                   client.print(String("GET ") + url + " HTTP/1.1\r\n" +
120                                "Host: " + host + "\r\n" +
121                                "Conexion: cerrada \r\n\r\n");
122                   unsigned long timeout = millis();
123                   while (client.available() == 0) {
124                     if (millis() - timeout > 10000) {
125                       Serial.println(">>> Client Timeout !");
126                       client.stop();
127                       return;
128                       }
129                     }
130
131                   //Respuesta del servidor
132                   while (client.available()){
133                     String line = client.readStringUntil('\r');
134                     Serial.println(line);
135                   }
136                 }
137               }
138             }
139           }
140       }
141       delay(1);
142     }
143   }
```

**Figura 3.22:** Función void loop() ESP8266. Parte 3.

### Explicación del código de la conexión con la base de datos y la creación de las tablas

Para poder conectarse y enviar los datos hacia la base de datos, se necesita de un archivo principal el cual debe contener el nombre, usuario y contraseña de la base de datos a conectarse. Partiendo de esas necesidades, se creó el script principal el cual empieza con la función *error_reporting,* la cual establece el nivel de error de reporte de *PHP*, en este caso se estableció que se indique todos los niveles de error excepto las **advertencias**, esto se hizo debido a que mostraba una advertencia de que algunos *arrays* no han sido definidas como se muestran en la **Figura 3.23.**

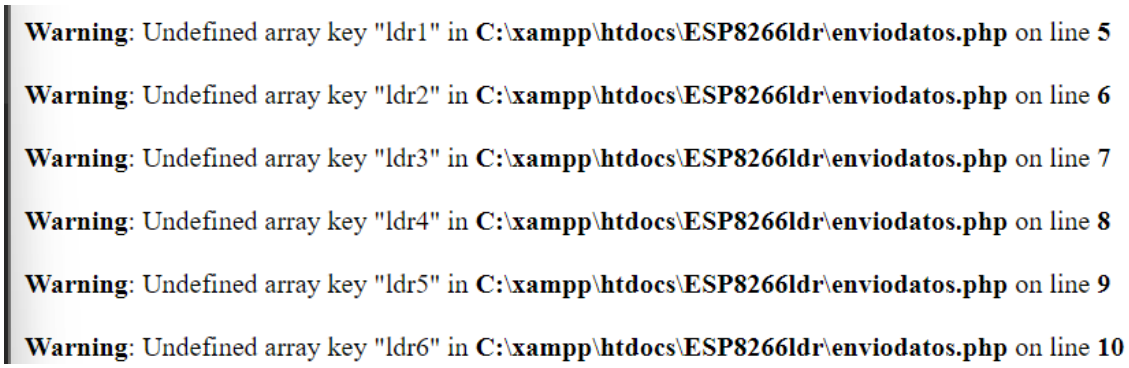



Figura 3.23: Nivel de error "advertencia".

Se crea 6 variables que son las cuales van a recibir los valores de los sensores *LDR* a través de la solicitud *HTTP GET.* Posteriormente, se establece conexión con la base de datos definiendo los valores de conexión como: nombre del servidos, usuario, contraseña y nombre de la base de datos como se muestra en la **Figura 3.24**

```
5   $ldr1 = $_GET['ldr1'];
6   $ldr2 = $_GET['ldr2'];
7   $ldr3 = $_GET['ldr3'];
8   $ldr4 = $_GET['ldr4'];
9   $ldr5 = $_GET['ldr5'];
10  $ldr6 = $_GET['ldr6'];
11
12
13  $usuario = "root";
14  $contraseña = "";
15  $servidor = "localhost";
16  $basededatos = "esp8266ldr";
17
18  $conexion = mysqli_connect ( $servidor, $usuario, "") or die ("No se ha podido conectar con el
        servidor");
19
20  $db = mysqli_select_db ($conexion, $basededatos) or die ("No se ha podido seleccionar la Base de
        Datos");
```

**Figura 3.24:** Conexión con la base de datos.

Una vez conectado a la base de datos se realiza una inserción de los valores de los sensores en la tabla principal llamada **"**historicldr**"**, la cual se llenará con los valores de todos los sensores *LDR,* también existirá una columna con la hora y fecha en la cual fue ingresado el valor, esto se realizará con la ayuda de la función *INSERT INTO* y finalmente se mostrará un mensaje de que los datos se han ingresado correctamente como se muestra en la **Figura 3.25.**

```
21
22    $consulta = "INSERT INTO `historicoldr` (`id`, `ldr1`, `ldr2`, `ldr3`, `ldr4`, `ldr5`, `ldr6`, `
          Fecha`) VALUES (NULL, '$ldr1', '$ldr2', '$ldr3', '$ldr4', '$ldr5', '$ldr6', CURRENT_TIMESTAMP)
          ";
23
24    $resultado = mysqli_query( $conexion, $consulta );
25
26    echo "Datos enviados correctamente. ";
```

**Figura 3.25:** Inserción de los valores en la tabla principal.

Para la parte de la creación de la tabla principal, se empieza añadiendo el *script* creado anteriormente para la conexión con la base de datos y también se incluye el comando para mostrar todos los errores excepto las **advertencias**.

Se añade una consulta mediante el comando *DELETE FROM*, dicha consulta es almacenada en la variable *$delete_sql***,** la cual borra la fila completa la cual contenga valores iguales a **0,** y después se hace otra consulta con el comando *SELECT,* la cual inserte los valores restantes en la tabla y se chequea por errores de conexión con la tabla como se muestra en la **Figura 3.26.**

```
1   <?php
2
3   include "enviodatos.php";
4   error_reporting(E_ALL & ~E_WARNING);
5
6
7   // Borra las filas que contengan valores igual a 0
8   $delete_sql = "DELETE FROM `historicoldr` WHERE ldr1 = 0 AND ldr2 = 0 AND ldr3 = 0 AND ldr4 = 0
      AND ldr5 = 0 AND ldr6 = 0";
9   $delete_query = mysqli_query($conexion, $delete_sql);
10
11  // Chequea por errores en la consulta DELETE
12  if (!$delete_query) {
13      die("Error en la consulta de eliminación: " . mysqli_error($conexion));
14  }
15
16
17  // Define la consulta SELECT para recibir los valores de la base de datos
18  $sql = "SELECT id, ldr1, ldr2, ldr3, ldr4, ldr5, ldr6, Fecha FROM `historicoldr`";
19
20  // Ejecuta la consulta
21  $query = mysqli_query($conexion, $sql);
22
23  // Chequea errores
24  if (!$query) {
25      die("Error en la consulta: " . mysqli_error($conexion));
26  }
27
```

**Figura 3.26:** Inserción de los valores restantes en la tabla principal.

Posteriormente se hace una verificación de si hay alguna fila de datos devuelta por la última consulta, si existen datos se realiza una nueva consulta con la función *INSERT INTO,* para insertar datos en una segunda tabla llamada **"**filtroldr**"**, los datos a insertarse en esta nueva tabla deben cumplir ciertas condiciones:

- Que el **ID** no exista en la segunda tabla.
- Que los valores sean mayores o iguales a **70.**

El resultado de esta consulta es guardado en **$***result***,** en la cual se comprueba se la consulta se ha realizado o ha fallado, en ambos casos se muestra un mensaje diferente para observar el estado de la consulta, el código se puede observar en la **Figura 3.27.**

```
29  // Verifica si hay por lo menos una fila de datos devuelta por la consulta antes realizada
30  if (mysqli_num_rows($query) > 0) {
31
32      // Define la consulta SQL INSERT INTO para poder insertar los datos en la tabla "filtroldr",
        dependiendo de las condiciones
33      $tabla2 = "INSERT INTO `filtroldr` (id, ldr1, ldr2, ldr3, ldr4, ldr5, ldr6, Fecha) SELECT id,
           ldr1, ldr2, ldr3, ldr4, ldr5, ldr6, Fecha FROM `historicoldr` WHERE id NOT IN (SELECT id
         FROM `filtroldr`) AND ldr1 >= 70 OR ldr2 >= 70 OR ldr3 >= 70 OR ldr4 >= 70 OR ldr5 >= 70 OR
          ldr6 >= 70 ON DUPLICATE KEY UPDATE ldr1 = VALUES(ldr1), ldr2 = VALUES(ldr2), ldr3 = VALUES
         (ldr3), ldr4 = VALUES(ldr4), ldr5 = VALUES(ldr5), ldr6 = VALUES(ldr6), Fecha = VALUES(
         Fecha)";
34
35
36      // Ejecuta la nueva consulta
37      $result = mysqli_query($conexion, $tabla2);
38
39      // Chequea por errores
40      if (!$result) {
41          die("Error en la consulta: " . mysqli_error($conexion));
42      }
43
44      // Muestra un mensaje diciendo que los datos han sido insertados
45   echo "Los datos han sido insertados en la tabla de valores filtrados";
46  }
47
48
49  else {
50      // Muestra un mensaje indicando que no hay datos que coincidan con la consulta
51      echo "No hay datos que cumplan con los criterios especificados";
52  }
53
54
55  ?>
```

**Figura 3.27:** Inserción de valores filtrados en la segunda tabla.

La parte final del código, es la creación de una página en *HTML* en la cual se incluye un encabezado con el título de la página y el estilo de una tabla, también se añaden 2 imágenes y se le pone un título al documento. Para finalizar, en el cuerpo del documento se crea un botón con un link para acceder a la tabla con los valores filtrados y se genera una tabla en la cual se irán mostrando los valores que se obtiene de la tabla de base de

datos general. El código *HTML* se lo puede observar en las **Figura 3.28, Figura 3.29 y Figura 3.30**.

```
<!---Formación del documento--->
<!DOCTYPE html>
<html>
  <!---Encabezado del documento--->
  <head>
    <!---Título de la página web--->
    <title>Inicio - Tabla general</title>

    <!---Estilo de la tabla--->
    <style type="text/css">
    table{
      border-collapse: collapse;
      width: 70%;
      border-style: solid;
      border-width: 3px;
      border-color: #000000;
      font-family: Sans-serif;
      font-size: 15px;
      text-align: center;
    }

    th{
      background-color: #28BEC1;
      color: white;
    }
    tr:nth-child(even) {
      background-color: #28BEC1;
    }

    </style>
  </head>

  <!---Cuerpo del documento--->
  <body>
    <!--Formación del encabezado--->
    <div style="border: 1px solid black">
      <img src="https://univercimas.com/wp-content/uploads/2021/05/
      Logo-de-la-Escuela-Politecnica-Nacional-EPN-300x300.png" alt="EPN" style=" float:left;
        width: 150px; height: 170px;"/>
      <img src="https://static.esmartcity.es/media/2017/12/
      cci-3-figura-1-solucion-stelaria-elt-iluminacion-inteligente.png" alt="Iluminacion" style="
         float:right; width: 150px; height: 170px;"/>
      <div style="border: 1px solid black; overflow:auto;">
        <h1 style= "text-align: center;">Escuela Politecnica Nacional </h1>
        <h2 style= "text-align: center;">Trabajo Previo a la Obtención del Título en:</h2>
        <h2 style="text-align: center;">Tecnología Superior en Redes y Telecomunicaciones</h2>
        <h2 style= "text-align: center;"></h2>
      </div>
    </div>
    <?php echo "<br>"; ?>

    <!---Botones de acceso--->
    <center>
      <a href="http://localhost/ESP8266ldr/datos_filtrados.php" class="button button-pill
```

**Figura 3.28:** Creación de la tabla en una página HTML.

```
        button-primary" target='_blank'>Tabla con los valores filtrados</a>
107     </center>
108     <br>
109
110     <table align= "center" width= "70%" cellpadding= "5" cellspacing= "5" border= "1">
111
112         <caption style="color: orange; font-size: 36px"><b>Tabla general Luminarias</b></caption>
113
114
115   <tr>
116     <tr><th style="color:red;">ID</th><th style="color:red;">LDR 1 (lx)</th><th style="color:red;
      ">LDR 2 (lx)</th><th style="color:red;">LDR 3 (lx)</th><th style="color:red;">LDR 4 (lx)</th>
      <th style="color:red;">LDR 5 (lx)</th><th style="color:red;">LDR 6 (lx)</th><th style="color:
        red;">Fecha - Hora</th>
117   </tr>
118
119   <?php while ($row = mysqli_fetch_array($query)){ ?>
120   <tr>
121     <?php if ($query->num_rows > 0): ?>
122     <!-- Mostrar la tabla aquí -->
123     <td style="color:red;"><b><?php if (!empty($row['id'])) {
124           echo $row['id'];
125         } else {
126           echo "-";
127         } ?></b></td>
128     <td style="color:blue;"><?php if (!empty($row['ldr1'])) {
129           echo $row['ldr1'];
130         } else {
131           echo "-";
132         } ?></td>
133     <td style="color:blue;"><?php if (!empty($row['ldr2'])) {
134           echo $row['ldr2'];
135         } else {
136           echo "-";
137         } ?></td>
138     <td style="color:blue;"><?php if (!empty($row['ldr3'])) {
139           echo $row['ldr3'];
140         } else {
141           echo "-";
142         } ?></td>
143     <td style="color:blue;"><?php if (!empty($row['ldr4'])) {
144           echo $row['ldr4'];
145         } else {
146           echo "-";
147         } ?></td>
148     <td style="color:blue;"><?php if (!empty($row['ldr5'])) {
149           echo $row['ldr5'];
150         } else {
151           echo "-";
152         } ?></td>
153     <td style="color:blue;"><?php if (!empty($row['ldr6'])) {
154           echo $row['ldr6'];
155         }  else {
156           echo "-";
```

**Figura 3.29:** Creación de la tabla en una página HTML. Parte 2.

```
157             } ?></td>
158         <td style="color:brown;"><?php if (!empty($row['Fecha'])) {
159                 echo $row['Fecha'];
160             } ?></td>
161 
162         <?php endif; ?>
163 
164     </tr>
165     <?php
166 
167     } ?>
168 
169     </table>
170 
171     <!---Apéndice--->
172       <footer>
173         <?php echo "<br>" ;?>
174         <div STYLE="border: 1px solid black; overflow: auto">
175           <h3 style="float: left;">Elaborado por: Kevin David Yépez Arteaga</h3>
176           <h3 style="float: right;">TSRT</h3>
177         </div>
178       </footer>
179     </html>
```

**Figura 3.30:** Creación de la tabla en una página HTML. Parte final.

Para la inserción de los datos en la tabla con los valores filtrados, simplemente se efectúa una consulta en la base de datos en la tabla **"filtroldr"** y mediante la función *SELECT*, se selecciona todos los campos para poder insertarlos y mostrarlos en la tabla, como se observa en la **Figura 3.31**, al igual que la primera, también fue creada en *HTML* con un encabezado, título de página, el cuerpo y la creación e inserción de los valores en la tabla.

```
1   <?php
2 
3   include "enviodatos.php";
4   error_reporting(E_ALL & ~E_WARNING);
5 
6 
7       // Define the SELECT query to insert the filtered data into the "alertldr" table
8   $sql = "SELECT id, ldr1, ldr2, ldr3, ldr4, ldr5, ldr6, Fecha FROM `filtroldr`";
9 
10  $query = mysqli_query($conexion, $sql);
11 
12  if (!$query) {
13      die("Error en la consulta: " . mysqli_error($conexion));
14  }
15 
```

**Figura 3.31:** Inserción de los valores filtrados en la tabla secundaria.

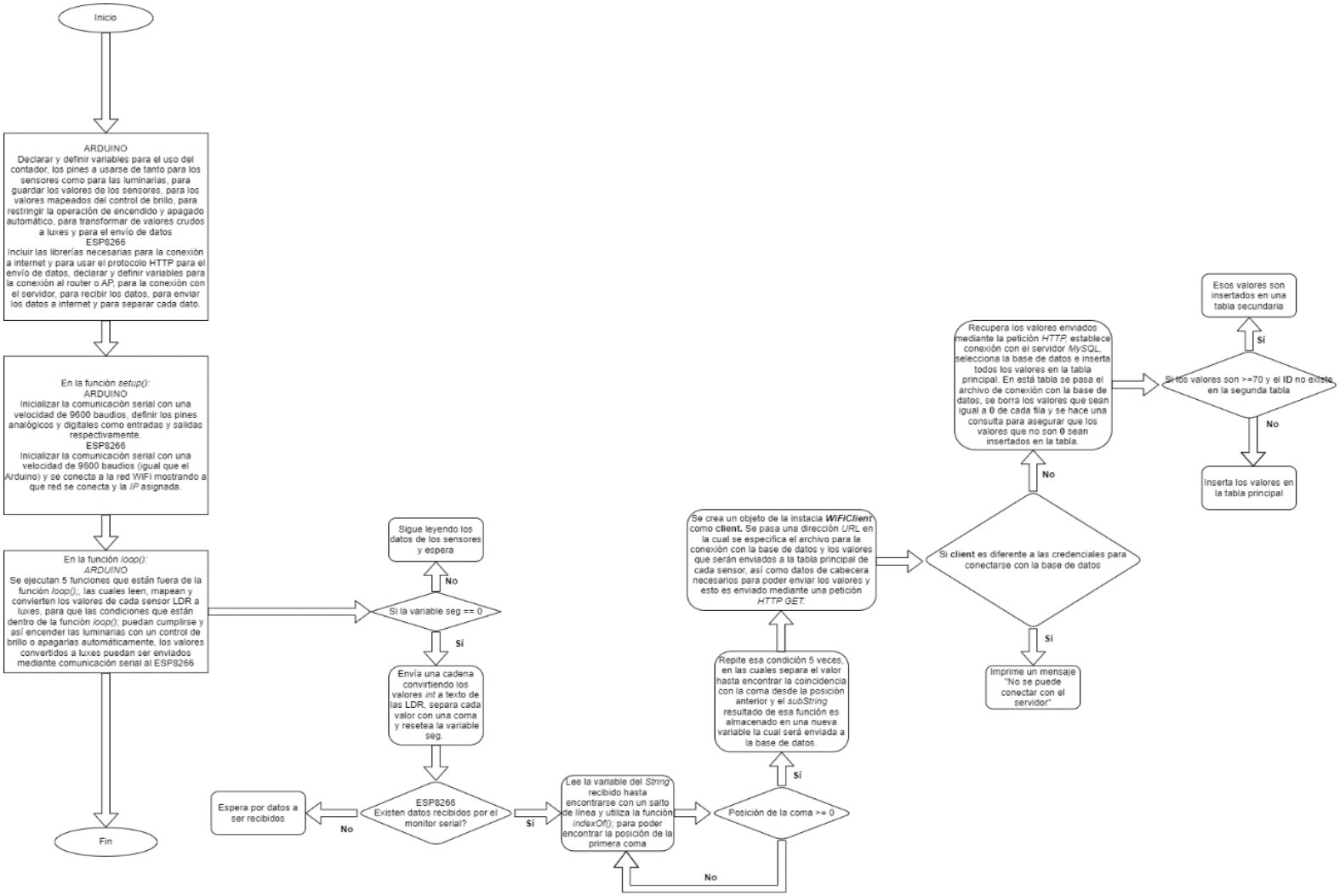



**Figura 3.32:** Flujograma de circuito completo

### Cálculo para transformar la lectura de los sensores LDR (valores crudos) a Luxes

Para empezar, se hizo una investigación de cómo se puede transformar los valores crudos de una *LDR* a luxes, se encontró una relación entre dichos valores y la intensidad de luz que incide sobre los sensores [16]. La relación que existe entre estos 2 valores es lineal, debido a que cuanto mayor sea la incidencia de luz sobre el sensor, menor será el valor crudo y viceversa. Pero para efectos de estudio y del presente proyecto, en este caso se hizo una relación directa o proporcional, esto quiere decir que, a mayor sea la cantidad de incidencia de luz, mayor serán los valores crudos. Partiendo de esta investigación, se hizo uso de la ecuación de la recta que se presenta a continuación:

$$y = m\mathrm{x} + \mathrm{b}$$

Ecuación 3.1: Ecuación de la recta [17].

Donde:

*y* : lux

*x* : valores crudos (vc)

Entonces, se sustituye esas relaciones en la **Ecuación 3.1,** y se obtiene:

$$lux = m * \mathrm{vc} + \mathrm{b}$$

Ecuación 3.2: Ecuación para transformar a luxes.

Partiendo de la **Ecuación 3.2,** se procede a calcular ***m*** y ***b,*** debido a que estos valores no son conocidos. Para calcular ***m,*** se puede usar la fórmula de la *pendiente de la recta,* la cual se presenta a continuación:

$$m = \frac{\mathrm{y2} - \mathrm{y1}}{\mathrm{x2} - \mathrm{x1}}$$

Ecuación 3.3: Pendiente de la recta [17].

Donde:

*y2, y1, x1, x2* : son puntos conocidos de la recta.

Para saber *y2, y1, x2, x1,* se realizó pruebas con el sensor *LDR* con diferentes tipos de iluminación y se obtuvieron los puntos que se observan en la **Figura 3.33.**

| LDR | Luxes |
|---|---|
| 280 | 0 |
| 303 | 3 |
| 326 | 6 |
| 349 | 9 |
| 372 | 12 |
| 395 | 15 |
| 418 | 18 |
| 441 | 21 |
| 464 | 24 |
| 487 | 27 |
| 510 | 30 |
| 533 | 33 |
| 556 | 36 |
| 579 | 39 |
| 602 | 42 |
| 625 | 45 |
| 648 | 48 |
| 671 | 51 |
| 694 | 54 |
| 717 | 57 |
| 740 | 60 |
| 763 | 63 |
| 786 | 66 |
| 809 | 69 |
| 832 | 72 |
| 855 | 75 |
| 878 | 78 |
| 901 | 81 |
| 924 | 84 |
| 947 | 87 |
| 970 | 90 |
| 993 | 93 |
| 1016 | 96 |
| 1023 | 99 |

**Figura 3.33:** Puntos de referencia.

Para obtener la pendiente de la recta **"m"**, de la **Ecuación 3.3**, se escogió los siguientes valores:

*y1: 280*

*y2: 1023*

*x1: 0*

*x2: 99*

Se reemplaza los valores en la **Ecuación 3.3**, se obtiene:

$$m = 0.133243$$

Debido a que entre más dígitos decimales se incluya en el código, el microprocesador tardará mucho más en ejecutar y procesar la función matemática [18], por lo tanto, se decidió usar 2 de los 6 dígitos decimales.

Ya teniendo la pendiente de la recta **"m",** se puede calcular **"b".** Se empieza dividiendo **"x"** en la **Ecuación 3.1,** quedaría de la siguiente manera:

$$\frac{y}{\mathrm{x}} = \mathrm{m} + \frac{b}{\mathrm{x}}$$

Ecuación 3.4: Dividiendo "x" en ambos lados.

Para despejar **"b"** de la **Ecuación 3.4,** se puede reemplazar "**x"** con cualquier punto conocido **(x1, y1)**. Despejando **"b"** de la **Ecuación 3.4** y haciendo los cambios descritos anteriormente, se obtiene:

$$b = y1 + \mathrm{m} * \mathrm{x1}$$

Ecuación 3.5: Punto de corte con el eje "y".

Usando la **Ecuación 3.5,** se obtiene:

$$b = -37.30821$$

Al igual que con la pendiente de la recta **"m",** solo se usan 2 de los 5 dígitos decimales, debido a que el tercer dígito es mayor a **5**, se redondea el segundo dígito al número entero superior, se usará el siguiente resultado:

$$b = -37.31$$

## 3.4 Generación de una base histórica de información

Para la realización de la tabla histórica de los datos enviados desde el ESP8266, se hizo uso de un código en *PHP,* el cual extrae los valores de la base de datos de una tabla en la cual son añadidos todos los valores enviados por el ESP8266.

En el código se empieza añadiendo la línea para poder conectarse con la base de datos. A continuación, se define una consulta *SQL* para eliminar las filas de la tabla, la cuales contengan valores iguales a **0** en todas las columnas y se comprueba si se produjo algún error en la consulta.

Después, se ejecuta una nueva consulta *SQL* para recuperar los datos de la tabla. Luego se define otra consulta *SQL*, la cual inserta datos en una nueva tabla, los cuales cumplan con las siguientes condiciones:

- Que los datos aún no hayan sido insertados en la nueva tabla (no se repitan).

- Que los valores sean iguales o superiores a 70 en cualquiera de las columnas y que el ***ID*** del valor no exista en la segunda tabla para ser insertados.

Finalmente, en el código *PHP* se muestra un mensaje diciendo que los datos han sido insertados correctamente en la nueva tabla.

El resto del código, es la creación de una página en *HTML* en el cual se genera una tabla en la cual se irán mostrando los valores que se obtiene de la base de datos.

El flujograma del código descrito, se puede observar en la **Figura 3.34.**

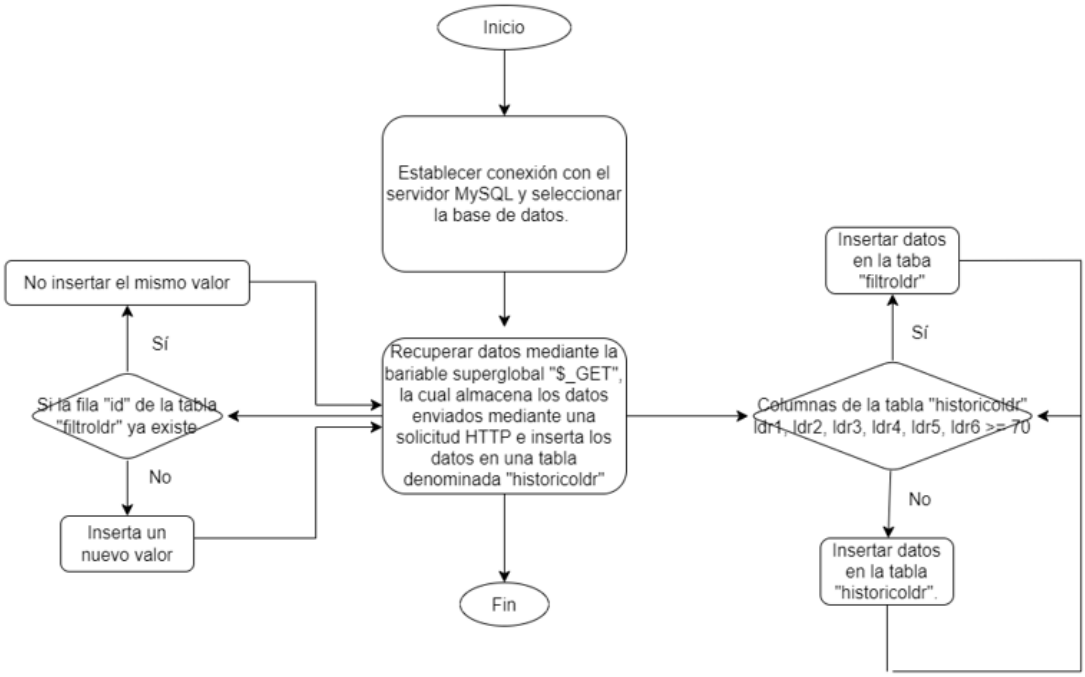



**Figura 3.34:** Flujograma de la creación de las tablas de los datos.

## 3.5 Pruebas de funcionamiento

Para la ejecución de las pruebas, se dividió el funcionamiento del sistema de alumbrado público inteligente en 4 partes: comparación del algoritmo tradicional con el algoritmo desarrollado en el proyecto, funcionamiento de los sensores *LDR*, control de brillo de las luminarias y la transmisión de información a la base de datos.

### Comparación del algoritmo tradicional con el algoritmo desarrollado en el proyecto

1. **Razón de la prueba:** esta prueba se realizó para comparar el algoritmo tradicional que se tiene hoy en día en el alumbrado público y destacar las ventajas que tiene el algoritmo desarrollado para el presente proyecto.

2. **Descripción:** el alumbrado público tradicional las luminarias se encienden cuando la luminosidad que incide sobre los sensores que tienen llega a un valor que se tiene establecido y las luminarias se encienden con toda su intensidad en todo momento hasta que se vuelve a incidir una luminosidad establecida sobre los sensores y estas se apaguen.

   En el algoritmo desarrollado para el presente proyecto, las luminarias son encendidas dependiendo un valor preestablecido, pero no se encienden con toda su intensidad desde el inicio, estas se van encendiendo gradualmente con un control del brillo de cada luminaria dependiendo de la luminosidad que incida sobre los sensores, al igual que se irán apagando gradualmente dependiendo de la luminosidad que vaya captando los sensores *LDR.*

3. **Comentarios:** el algoritmo desarrollado para el presente proyecto tiene algunas ventajas en comparación del algoritmo tradicional:

- Las luminarias tendrán un mayor tiempo de vida debido a que no están encendidas con todo su brillo en todo el tiempo que permanezcan encendidas.
- Tiene un menor consumo energético gracias a que las luminarias ajustan su brillo gradualmente en comparación del alumbrado tradicional, las cuales tienen un encendido y apagado fijo.
- Tiene una menor contaminación lumínica en zonas en las cuales no se necesita tanta iluminación gracias a que se ajusta automáticamente el brillo de las luminarias.

### Funcionamiento de los sensores LDR

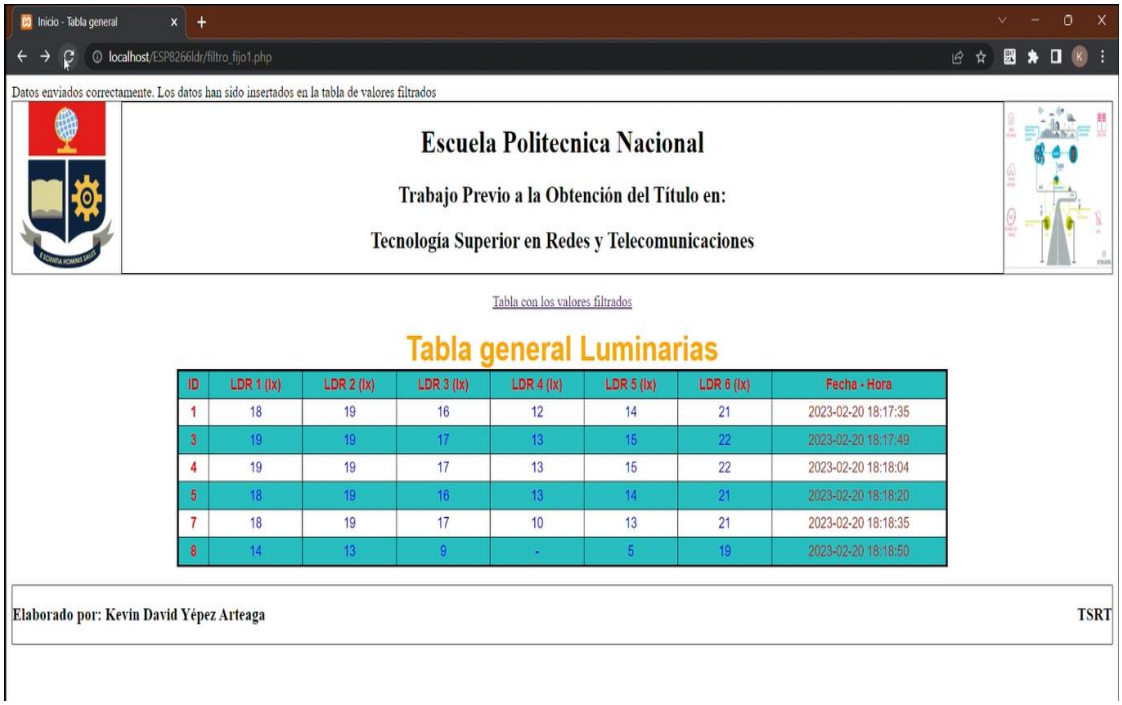
1. **Razón de la prueba:** la prueba se realizó para comprobar que los sensores capten y envíen los datos correctamente al Arduino para procesarlos.
2. **Descripción:** los sensores fueron colocados en la maqueta en la parte superior del “poste”, como se observa en la **Figura 3.35**, para detectar la luz que incide desde la arriba.

**Figura 3.35:** Disposición de las LDR en la maqueta.

3. **Comentarios:** estos sensores en su lectura arrojan valores de 0 a 1023 (rango en el que trabaja el Arduino).
4. **Correcciones:** se limitó los valores de lectura como se explicó en la sección **Implementación del prototipo,** para así tener un mejor control del encendido y apagado de las luminarias.

### Control de brillo de las luminarias

1. **Razón de la prueba:** en esta prueba se comprobó que las luminarias se encendieran con un brillo controlado, según la luz que incida sobre los sensores.
2. **Descripción:** las luminarias se encenderán con un brillo controlado por la luz que esté incidiendo sobre los sensores, los valores de intensidad de brillo que emiten las luminarias son los que son insertados en las tablas, como se muestra en las **Figura 3.36, Figura 3.37, Figura 3.38, Figura 3.39.**




| ID | LDR 1 (lx) | LDR 2 (lx) | LDR 3 (lx) | LDR 4 (lx) | LDR 5 (lx) | LDR 6 (lx) | Fecha - Hora |
|---|---|---|---|---|---|---|---|
| 1 | 18 | 19 | 16 | 12 | 14 | 21 | 2023-02-20 18:17:35 |
| 3 | 19 | 19 | 17 | 13 | 15 | 22 | 2023-02-20 18:17:49 |
| 4 | 19 | 19 | 17 | 13 | 15 | 22 | 2023-02-20 18:18:04 |
| 5 | 18 | 19 | 16 | 13 | 14 | 21 | 2023-02-20 18:18:20 |
| 7 | 18 | 19 | 17 | 10 | 13 | 21 | 2023-02-20 18:18:35 |
| 8 | 14 | 13 | 9 | - | 5 | 19 | 2023-02-20 18:18:50 |

**Figura 3.36:** Intensidad de luminosidad detectada por los sensores.


**Figura 3.37:** Intensidad reflejada en las luminarias.

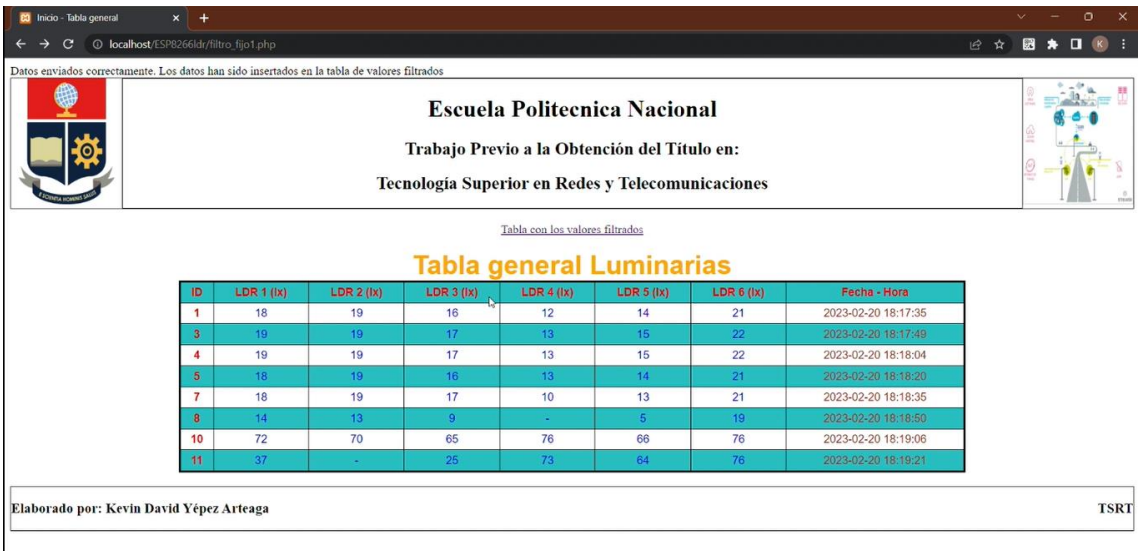



**Figura 3.38:** Cambio de luminosidad detectada en los sensores.

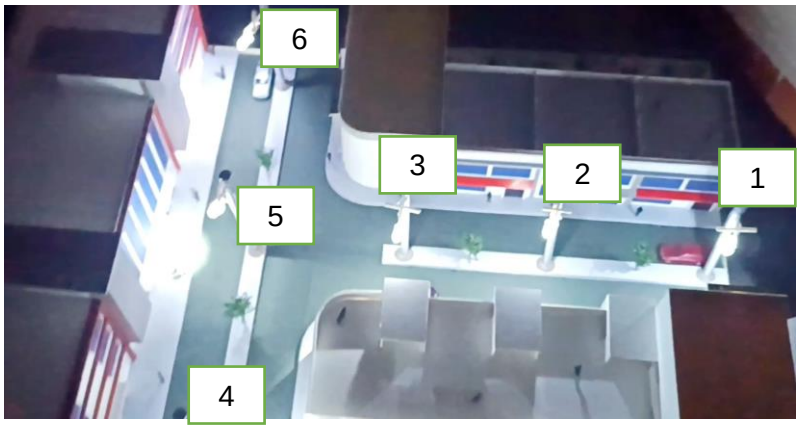



**Figura 3.39:** Cambio reflejado en la intensidad de las luminarias.

3. **Comentarios:** se puede observar que las luminarias reflejan el cambio de luminosidad que detectan los sensores.

### Transmisión de información a la base de datos

1. **Razón de la prueba:** esta prueba fue realizada para evidenciar el correcto funcionamiento del módulo ESP8266, así como el correcto envío de los valores hacia la base de datos.
2. **Descripción:** mientras se realiza el cambio de luminosidad sobre los sensores *LDR* y se espera el tiempo establecido para el envío de los valores a la base de datos, los valores son ingresados en la tabla principal, donde se muestra la información de todos los valores enviados, como se muestra en la **Figura 3.40.** También se tiene una segunda tabla como se muestra en la **Figura 3.41**, la cual filtra los valores de la tabla principal dependiendo de la condición establecida, como se menciona en la sección **Generación de una base histórica de información.**

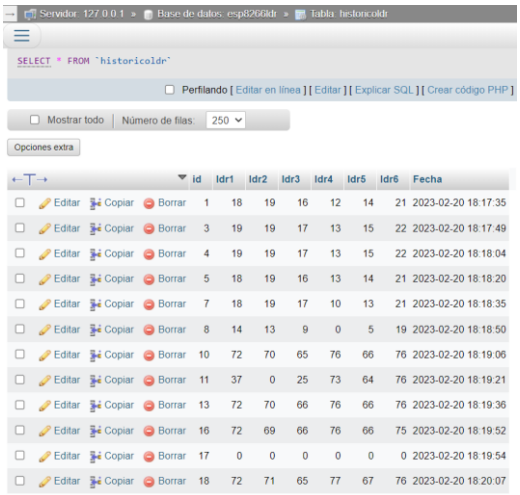



**Figura 3.40:** Tabla de todos los datos enviados por el ESP8266.


**Figura 3.41:** Tabla con los valores filtrados.

La comprobación del funcionamiento completo del sistema se puede observar escaneando el siguiente código *QR* que se muestra en la **Figura 3.42**, o a través del siguiente link: https://youtu.be/E7Yf9swLZ1E.


**Figura 3.42:** Código QR del video.

# 4 CONCLUSIONES

- La investigación sobre la intensidad de luz ambiental y los factores que puedan alterar esta misma luz ambiental como los días lluviosos, días nublados, soleados, etc, fue fundamental para hacer la elección de que tipo de sensor se iba a utilizar para el proyecto. Partiendo de eso se implementó un prototipo de sistema que sea capaz de detectar la luz ambiental para que mediante los sensores puedan tener un brillo controlado de las luminarias.
- Se hizo un estudio necesario para la selección de los dispositivos, los cuales puedan cumplir con todas las exigencias del sistema como el suministrar la energía necesaria para controlar de manera eficiente las luminarias y sensores, así como para energizar el módulo de comunicación inalámbrica el cual ayudará con el envío de datos.
- *Xampp* es una herramienta muy útil para los desarrolladores de páginas web, primeramente, porque es una aplicación multiplataforma lo cual permite probar las aplicaciones realizadas en diferentes sistemas operativos. Permite crear un entorno controlado de un servidor local para realizar pruebas de la página web o la base de datos que se está creando antes de que esta sea publicada oficialmente.
- El prototipo se puede implementar a gran escala incluyendo dispositivos de potencia como pueden ser **relés***,* para controlar luminarias a un voltaje más alto como el que se tiene en el tendido eléctrico, así también el uso de dispositivos que protejan el contraflujo de corriente en el caso del tendido eléctrico, estos dispositivos pueden ser **diodos rectificadores, puentes de diodo,** etc
- Los códigos que se diseñaron, son los encargados de dar las instrucciones a todos los componentes del sistema para que puedan comunicarse de forma correcta entre sí. También es importante el tipo de comunicación que se llevó a cabo para enlazar el módulo ESP8266 y la tarjeta Arduino Uno para el envío y recepción de los datos, el tipo de comunicación que se usó entre estas 2 placas fue comunicación serial **Asíncrona.**
- la comunicación serial **Asíncrona** la cual no usa un reloj compartido, con esto se tiene flexibilidad a la del envío y recepción de datos, ya que no se necesita que ambos equipos estén sincronizados de manera precisa para su correcto funcionamiento. También requiere menos ancho de banda que la comunicación síncrona, debido a que esta no necesita de señales de sincronización y, por

último, este tipo de comunicación es compatible con una gran variedad de dispositivos como sensores, microcontroladores, etc.

- La transmisión de los datos mediante comunicación inalámbrica es fundamental en el proyecto, así que se debe escoger correctamente el módulo que ayudará con esta función ya que este debe tener un buen alcance al igual que una transmisión confiable para tener los valores correctos en la base de datos para después ser extraídos y mostrados mediante una página web la cual contendrá una tabla en la cual los datos podrán ser visualizados de una manera ordenada, la medida en la que están los datos, así como la fecha y hora en la que el dato fue ingresado al sistema.
- Existen varios de tipos de comunicación inalámbrica, entre ellos está: *zigbee, bluetooth, IR (Infrarrojo), WiFi, etc.* De las tecnologías mencionadas anteriormente se usó la comunicación inalámbrica mediante WiFi, debido a que esta tecnología es superior a las demás por su alcance en cuestión de distancia y también a que se puede enviar una mayor cantidad de información.
- Para terminar, las pruebas que se realizó a cada componente del sistema, fueron necesarias para corregir los valores en los cuales los componentes trabajan correctamente. Estas pruebas ayudaron a determinar el rango de trabajo correcto de cada componente. Con esto se analizó que el sistema de alumbrado público inteligente funcionara correctamente en todas sus etapas, como: la comunicación inalámbrica mediante *WiFi,* el monitoreo constante de los valores de cada sensor, el control de brilla de cada luminaria y el registro de los valores en la base de datos.

# 5 RECOMENDACIONES

- Es importante definir adecuadamente los valores del sensor de luz, debido a que si se escoge valores muy bajos o muy altos en ciertas ocasiones las luminarias no tendrán un encendido y apagado correcto.
- Los programas usados tanto como para el diseño y levantamiento de servidores, permiten una simulación del funcionamiento del sistema. A pesar de que se puede realizar simulaciones es recomendable ejecutar la implementación física del sistema para evidenciar su funcionamiento en tiempo real.
- Esta es una aplicación que puede ser implementada en lugares donde la iluminación es escasa como en zonas rurales de la ciudad, así como en lugares donde la tecnología está en su punto más alto como podría ser las *Smart Cities,* que son lugares en los cuales se puede explotar de mejor manera esta idea.
- Se recomienda hacer un análisis del área en la cual va a ser implementado el sistema, debido a que no en todos los sectores la luz ambiental es la misma al igual que las condiciones climáticas pueden variar de un lugar a otro, para que a futuro el sistema se comporte de una manera correcta en cuanto al brillo de las luminarias.
- La página web que muestra los datos de las luminarias al igual que las bases de datos en la cual se recolectan los valores, pueden ser mejoradas añadiendo una experiencia más vistosa, atractiva y útil, al igual que se podría añadir un *backup* de los datos en caso de necesitarlos, esto con el uso de programas especializados para el diseño de páginas web en los que se pueda añadir botones dinámicos, espacios específicos para la recolección de datos y una base de datos en tiempo real.
- Es recomendable usar el módulo WiFi ESP8266 *v.3*, debido a que este ya es una placa de desarrollo la cual cuenta con más pines en comparación de sus generaciones anteriores, ya que en esta versión se puede programar conectando el módulo directamente a la PC simplemente con la ayuda de un cable *USB* y se puede cargar los códigos en el mismo lenguaje que el Arduino Uno, sus anteriores generaciones necesitan de comandos especiales para funcionar correctamente y la forma de cargar los códigos es más complicado ya que se debe realizar conexiones especiales.

# 6 REFERENCIAS BIBLIOGRÁFICAS


[1] «Wikipedia,» 12 09 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Arduino_IDE. [Último acceso: 06 12 2022].

[2] K. Söderby, «Docs Arduino,» 12 11 2022. [En línea]. Available: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2. [Último acceso: 28 11 2022].

[3] «ARDUINO.cl,» [En línea]. Available: https://arduino.cl/arduino-uno/. [Último acceso: 15 01 2023].

[4] «TECmikro,» 2022. [En línea]. Available: https://tecmikro.com/content/17-arduino-uno-r3-caracteristicas. [Último acceso: 29 11 2022].

[5] «NAYLAMP Mechatronics,» [En línea]. Available: https://naylampmechatronics.com/blog/56_usando-esp8266-con-el-ide-de-arduino.html. [Último acceso: 15 01 2023].

[6] «ESPRESSIF,» 10 2022. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. [Último acceso: 21 12 2022].

[7] «Industrias GSL,» 30 01 2022. [En línea]. Available: https://industriasgsl.com/blogs/automatizacion/sensor-ldr. [Último acceso: 21 12 2022].

[8] «MecatrónicaLATAM,» 23 04 2021. [En línea]. Available: https://www.mecatronicalatam.com/es/tutoriales/sensores/sensor-de-luz/ldr/. [Último acceso: 15 01 2023].

[9] A. Oram, «Books Google,» 06 2002. [En línea]. Available: https://books.google.com.ec/books?hl=es&lr=&id=9c-pkLaNmqoC&oi=fnd&pg=PR5&dq=what+is+mysql+protocol&ots=G2svlTP5oa&sig=

xdWPeJvTfcPcwQKSiPj1Jmc7KJU&redir_esc=y#v=onepage&q=what%20is%20mysql%20protocol&f=false. [Último acceso: 29 12 2022].

[10] «SkySQL,» 08 2012. [En línea]. Available: http://rozero.webcindario.com/disciplinas/fbmg/abd3/MariaDB_vs_MySQL.pdf. [Último acceso: 04 01 2023].

[11] E. C. F. &. S. Godbole, «Springer,» 08 11 2016. [En línea]. Available: https://link.springer.com/chapter/10.1007/978-1-4842-1191-5_24. [Último acceso: 04 01 2023].

[12] «Apache Friends,» 2022. [En línea]. Available: https://www.apachefriends.org/es/about.html. [Último acceso: 04 01 2023].

[13] X.-j. C. Yi-yuan Fang, «IEEEexplore,» 13 06 2011. [En línea]. Available: https://sci-hub.ru/https://ieeexplore.ieee.org/abstract/document/5873448. [Último acceso: 18 01 2023].

[14] «ITQ,» [En línea]. Available: http://www.itq.edu.mx/carreras/IngElectronica/archivos_contenido/Apuntes%20de%20materias/ETD1022_Microcontroladores/4_SerialCom.pdf. [Último acceso: 18 010 2023].

[15] «W3 Schools,» [En línea]. Available: https://www.w3schools.com/tags/ref_httpmethods.asp. [Último acceso: 30 01 2023].

[16] Mac, «Foros de Electrónica,» 25 04 2008. [En línea]. Available: https://www.forosdeelectronica.com/threads/rango-de-medidas-de-una-ldr.92258/page-2. [Último acceso: 20 03 2023].

[17] «Nodo Universitario,» 2016. [En línea]. Available: https://oa.ugto.mx/oa/oa-enmsir-0000001/2_formas_de_la_ecuacin_de_la_recta.html. [Último acceso: 12 02 2023].

[18] «Aprendiendo Arduino,» 2015. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2015/03/26/tipos-de-datos/. [Último acceso: 16 02 2023].

# 7 ANEXOS

ANEXO I. Certificado de originalidad

ANEXO II. Enlaces

ANEXO III. Conjunto de datos extensos

## ANEXO I: Certificado de Originalidad

**CERTIFICADO DE ORIGINALIDAD**

Quito, D.M. 2 de marzo de 2022

De mi consideración:

Yo, ITALO ALEXANDER CARREÑO MENDOZA, en calidad de Director del Trabajo de Integración Curricular titulado IMPLEMENTACIÓN DE UN PROTOTIPO DE ALUMBRADO PÚBLICO INTELIGENTE elaborado por el estudiante KEVIN DAVID YÉPEZ ARTEAGA de la carrera en TECNÓLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 17%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

https://epnecuador-my.sharepoint.com/:b:/g/personal/italo_carreno_epn_edu_ec/EVPdH5kFMGFAsPDqMcsTMjsBhOGl2oP0KKxWlg6K5hoNQg?e=uiQLji

Atentamente,

ITALO ALEXANDER CARREÑO MENDOZA

Docente

Escuela de Formación de Tecnólogos

# ANEXO II:  Enlaces

**Anexo II.I** Código QR de la implementación y pruebas de funcionamiento


https://youtu.be/E7Yf9swLZ1E

## ANEXO III: Códigos Fuente

## CÓDIGO DE LA PLACA ARDUINO UNO

```
//Variables reseteo de funcion millis
extern volatile unsigned long timer0_millis;
unsigned long new_value = 15;
int seg = 0;

//Variables para guardar los valores del LDR
int LDR = 0;
volatile int valorldr1 = 0;
volatile int valorldr2 = 0;
volatile int valorldr3 = 0;
volatile int valorldr4 = 0;
volatile int valorldr5 = 0;
volatile int valorldr6 = 0;

//Variables mapeadas para la intensidad de los LEDs
volatile int l1 = 0;
volatile int l2 = 0;
volatile int l3 = 0;
volatile int l4 = 0;
volatile int l5 = 0;
volatile int l6 = 0;

//Variables para restringir la intensidad
int max = 1023;
int min = 280;

//Variables de las iluminarias
const int Lum1 = 3;
const int Lum2 = 5;
const int Lum3 = 6;
const int Lum4 = 9;
const int Lum5 = 10;
const int Lum6 = 11;

//Variables para transformar a luxes
float m = 0.13; //Pendiente de la recta
volatile float b = -37.31; //Intercepto
int maxled = 100;
int minled = 0;
volatile int lux1 = 0;
volatile int lux2 = 0;
volatile int lux3 = 0;
```

```
volatile int lux4 = 0;
volatile int lux5 = 0;
volatile int lux6 = 0;

//Variable a transmitir
String valores;

void setup() {
  Serial.begin(9600);
  for (LDR = 14; LDR <=19; LDR++){
    pinMode(LDR, INPUT);
  }
  pinMode(Lum1, OUTPUT);
  pinMode(Lum2, OUTPUT);
  pinMode(Lum3, OUTPUT);
  pinMode(Lum4, OUTPUT);
  pinMode(Lum5, OUTPUT);
  pinMode(Lum6, OUTPUT);

}

void loop() {
  seg = new_value - (millis() / 1000);

  ldr_1();
  if (valorldr1 > 280 || valorldr1 <=1023) {
    analogWrite (Lum1, l1);
  }
  else {
    digitalWrite (Lum1, LOW);
  }

  ldr_2();
    if (valorldr2 > 280 || valorldr2 <= 1023) {
    analogWrite (Lum2, l2);
  }
    else {
    digitalWrite (Lum2, LOW);
  }

  ldr_3();
  if (valorldr3 > 280 || valorldr3 <= 1023) {
    analogWrite (Lum3, l3);
  }
  else {
    digitalWrite (Lum3, LOW);
  }
```

```
  ldr_4();
  if (valorldr4 > 280 || valorldr4 <= 1023) {
    analogWrite (Lum4, l4);
  }
  else {
    digitalWrite (Lum4, LOW);
  }

  ldr_5();
    if (valorldr5 > 280 || valorldr5 <= 1023) {
    analogWrite (Lum5, l5);
  }
  else {
    digitalWrite (Lum5, LOW);
  }

  ldr_6();
    if (valorldr6 > 280 || valorldr6 <= 1023) {
    analogWrite (Lum6, l6);
  }
  else {
    digitalWrite (Lum6, LOW);
  }

  //Valores a transmitir y reseteo del contador
  if (seg == 0){
    valores = String(lux1) + "," + String(lux2) + "," + String(lux3) +
"," + String(lux4) + "," + String(lux5) + "," + String(lux6);
    Serial.println(valores);
    setMillis(new_value);
  }
}


//Funciones de lectura de los sensores LDR
void ldr_1 (){
  valorldr1 = analogRead (14);
  valorldr1 = constrain (valorldr1, min, max);
  l1 = map (valorldr1, max, min, 0, 255);
  lux1 = (m*valorldr1) + b;
  lux1 = constrain (lux1, minled, maxled);

  delay (50);
}

void ldr_2 (){
  valorldr2 = analogRead (15);
  valorldr2 = constrain (valorldr2, min, max);
```

```
  l2 = map (valorldr2, max, min, 0, 255);
  lux2 = (m*valorldr2) + b;
  lux2 = constrain (lux2, minled, maxled);

  delay (50);
}

void ldr_3 (){
  valorldr3 = analogRead (16);
  valorldr3 = constrain (valorldr3, min, max);
  l3 = map (valorldr3, max, min, 0, 255);
  lux3 = (m*valorldr3) + b;
  lux3 = constrain (lux3, minled, maxled);

  delay (50);
}

void ldr_4 (){
  valorldr4 = analogRead (17);
  valorldr4 = constrain (valorldr4, min, max);
  l4 = map (valorldr4, max, min, 0, 255);
  lux4 = (m*valorldr4) + b;
  lux4 = constrain (lux4, minled, maxled);

  delay (50);
}

void ldr_5 (){
  valorldr5 = analogRead (18);
  valorldr5 = constrain (valorldr5, min, max);
  l5 = map (valorldr5, max, min, 0, 255);
  lux5 = (m*valorldr5) + b;
  lux5 = constrain (lux5, minled, maxled);

  delay (50);
}

void ldr_6 (){
  valorldr6 = analogRead (19);
  valorldr6 = constrain (valorldr6, min, max);
  l6 = map (valorldr6, max, min, 0, 255);
  lux6 = (m*valorldr6) + b;
  lux6 = constrain(lux6, minled, maxled);

  delay (50);
}

//Función reseteo de millis
```

```
void setMillis(unsigned long new_millis){
  uint8_t oldSREG = SREG;
  cli();
  timer0_millis = new_millis;
  SREG = oldSREG;
}
```

## Código del módulo ESP8266

```
#include <ESP8266HTTPClient.h>
#include <ESP8266WiFiMulti.h>

// defino credenciales red
static const char* ssid = "INTERNET_CNT YEPEZ";
static const char* password = "Titokeluckyfer12345";

//Variables para conexión con el servidor
const char* host = "192.168.1.12";
const int port = 80;

//Variables a enviar
String serialrcv;
String sub_str;
int ldr_1 = 0;
int ldr_2 = 0;
int ldr_3 = 0;
int ldr_4 = 0;
int ldr_5 = 0;
int ldr_6 = 0;

//Variables para separar el String
int coma1 = 0;
int coma2 = 0;
int coma3 = 0;
int coma4 = 0;
int coma5 = 0;

void setup() {
  Serial.begin(9600);
  while(!Serial){;
  }

  WiFi.begin(ssid, password);
  Serial.print("Conectando...");
  while (WiFi.status()!= WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Conectado a: ");
  Serial.println(ssid);
  Serial.print("IP Local: ");
  Serial.println(WiFi.localIP());
  Serial.println();
```

```
}

void loop() {
  if (Serial.available()){
    serialrcv = Serial.readStringUntil ('\n'); //creamos y llenamos
buffer hasta un "enter"
    Serial.println (serialrcv);

    coma1 = serialrcv.indexOf(",");

    if (coma1 >= 0){
      sub_str = serialrcv.substring(0, coma1);
      ldr_1 = sub_str.toInt();
      Serial.print("LDR 1: ");
      Serial.println(ldr_1);

      coma2 = serialrcv.indexOf(",", coma1 + 1);

      if (coma2 >= 0){
        sub_str = serialrcv.substring(coma1 + 1, coma2);
        ldr_2 = sub_str.toInt();
        Serial.print("LDR 2: ");
        Serial.println(ldr_2);

        coma3 = serialrcv.indexOf(",", coma2 + 1);

        if (coma3 >= 0){
          sub_str = serialrcv.substring(coma2 + 1, coma3);
          ldr_3 = sub_str.toInt();
          Serial.print("LDR 3: ");
          Serial.println(ldr_3);

          coma4 = serialrcv.indexOf(",", coma3 + 1);

          if (coma4 >= 0){
            sub_str = serialrcv.substring(coma3 + 1, coma4);
            ldr_4 = sub_str.toInt();
            Serial.print("LDR 4: ");
            Serial.println(ldr_4);

            coma5 = serialrcv.indexOf(",", coma4 + 1);

            if (coma5 >= 0){
              sub_str = serialrcv.substring(coma4 + 1, coma5);
              ldr_5 = sub_str.toInt();
              Serial.print("LDR 5: ");
              Serial.println(ldr_5);
```

```
              sub_str = serialrcv.substring(coma5 + 1);
              ldr_6 = sub_str.toInt();
              Serial.print("LDR 6: ");
              Serial.println(ldr_6);

              WiFiClient client;
              if (!client.connect(host, port)){
                Serial.println("No se pude conectar al servidor");
                return;
                }
              String url = "/ESP8266ldr/enviodatos.php?ldr1=";
              url += ldr_1;
              url += "&ldr2=";
              url += ldr_2;
              url += "&ldr3=";
              url += ldr_3;
              url += "&ldr4=";
              url += ldr_4;
              url += "&ldr5=";
              url += ldr_5;
              url += "&ldr6=";
              url += ldr_6;

              //Envio de petición al servidor
              client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                           "Host: " + host + "\r\n" +
                           "Conexion: cerrada \r\n\r\n");
              unsigned long timeout = millis();
              while (client.available() == 0) {
                if (millis() - timeout > 10000) {
                  Serial.println(">>> Client Timeout !");
                  client.stop();
                  return;
                  }
                }
              //Respuesta del servidor
              while (client.available()){
                String line = client.readStringUntil('\r');
                Serial.println(line);
              }
            }
          }
        }
      }
    }
    delay(1);
  }
}
```

## Código de la conexión con la base de datos

```php
<?php

error_reporting(E_ALL & ~E_WARNING);

$ldr1 = $_GET['ldr1'];
$ldr2 = $_GET['ldr2'];
$ldr3 = $_GET['ldr3'];
$ldr4 = $_GET['ldr4'];
$ldr5 = $_GET['ldr5'];
$ldr6 = $_GET['ldr6'];


$usuario = "root";
$contraseña = "";
$servidor = "localhost";
$basededatos = "esp8266ldr";

$conexion = mysqli_connect ( $servidor, $usuario, "") or die ("No se ha podido
conectar con el servidor");

$db = mysqli_select_db ($conexion, $basededatos) or die ("No se ha podido
seleccionar la Base de Datos");

$consulta = "INSERT INTO `historicoldr` (`id`, `ldr1`, `ldr2`, `ldr3`, `ldr4`,
`ldr5`, `ldr6`, `Fecha`) VALUES (NULL, '$ldr1', '$ldr2', '$ldr3', '$ldr4',
'$ldr5', '$ldr6', CURRENT_TIMESTAMP)";

$resultado = mysqli_query( $conexion, $consulta );

echo "Datos enviados correctamente. ";


?>
```

## Código de la tabla principal

```php
<?php

include "enviodatos.php";
error_reporting(E_ALL & ~E_WARNING);


// Borra las filas que contengan valores igual a 0
$delete_sql = "DELETE FROM `historicoldr` WHERE ldr1 = 0 AND ldr2 = 0 AND ldr3
= 0 AND ldr4 = 0 AND ldr5 = 0 AND ldr6 = 0";
$delete_query = mysqli_query($conexion, $delete_sql);

// Chequea por errores en la consulta DELETE
if (!$delete_query) {
    die("Error en la consulta de eliminación: " . mysqli_error($conexion));
}


// Define la consulta SELECT para recibir los valores de la base de datos
$sql = "SELECT id, ldr1, ldr2, ldr3, ldr4, ldr5, ldr6, Fecha FROM
`historicoldr`";

// Ejecuta la consulta
$query = mysqli_query($conexion, $sql);

// Chequea errores
if (!$query) {
    die("Error en la consulta: " . mysqli_error($conexion));
}


// Verifica si hay por lo menos una fila de datos devuelta por la consulta
antes realizada
if (mysqli_num_rows($query) > 0) {

    // Define la consulta SQL INSERT INTO para poder insertar los datos en la
tabla "filtroldr", dependiendo de las condiciones
    $tabla2 = "INSERT INTO `filtroldr` (id, ldr1, ldr2, ldr3, ldr4, ldr5, ldr6,
Fecha) SELECT id, ldr1, ldr2, ldr3, ldr4, ldr5, ldr6, Fecha FROM `historicoldr`
WHERE id NOT IN (SELECT id FROM `filtroldr`) AND ldr1 >= 70 OR ldr2 >= 70 OR
ldr3 >= 70 OR ldr4 >= 70 OR ldr5 >= 70 OR ldr6 >= 70 ON DUPLICATE KEY UPDATE
ldr1 = VALUES(ldr1), ldr2 = VALUES(ldr2), ldr3 = VALUES(ldr3), ldr4 =
VALUES(ldr4), ldr5 = VALUES(ldr5), ldr6 = VALUES(ldr6), Fecha = VALUES(Fecha)";


    // Ejecuta la nueva consulta
    $result = mysqli_query($conexion, $tabla2);

    // Chequea por errores
    if (!$result) {
        die("Error en la consulta: " . mysqli_error($conexion));
    }

    // Muestra un mensaje diciendo que los datos han sido insertados
 echo "Los datos han sido insertados en la tabla de valores filtrados";
```

```
}


else {
    // Muestra un mensaje indicando que no hay datos que coincidan con la
consulta
    echo "No hay datos que cumplan con los criterios especificados";
}


?>

<!---Formación del documento--->
<!DOCTYPE html>
<html>
  <!---Encabezado del documento--->
  <head>
    <!---Título de la página web--->
    <title>Inicio - Tabla general</title>

    <!---Estilo de la tabla--->
    <style type="text/css">
    table{
      border-collapse: collapse;
      width: 70%;
      border-style: solid;
      border-width: 3px;
      border-color: #000000;
      font-family: Sans-serif;
      font-size: 15px;
      text-align: center;
    }

    th{
      background-color: #28BEC1;
      color: white;
    }
    tr:nth-child(even) {
      background-color: #28BEC1;
    }

    </style>
  </head>

  <!---Cuerpo del documento--->
  <body>
    <!--Formación del encabezado--->
    <div style="border: 1px solid black">
      <img src="https://univercimas.com/wp-content/uploads/2021/05/Logo-de-la-
Escuela-Politecnica-Nacional-EPN-300x300.png" alt="EPN" style=" float:left;
width: 150px; height: 170px;"/>
      <img src="https://static.esmartcity.es/media/2017/12/cci-3-figura-1-
solucion-stelaria-elt-iluminacion-inteligente.png" alt="Iluminacion" style="
float:right; width: 150px; height: 170px;"/>
      <div style="border: 1px solid black; overflow:auto;">
```

```
        <h1 style= "text-align: center;">Escuela Politecnica Nacional </h1>
        <h2 style= "text-align: center;">Trabajo Previo a la Obtención del
Título en:</h2>
        <h2 style="text-align: center;">Tecnología Superior en Redes y
Telecomunicaciones</h2>
        <h2 style= "text-align: center;"></h2>
      </div>
    </div>
    <?php echo "<br>"; ?>

    <!---Botones de acceso--->
    <center>
      <a href="http://localhost/ESP8266ldr/datos_filtrados.php" class="button
button-pill button-primary" target='_blank'>Tabla con los valores filtrados</a>
    </center>
    <br>

    <table align= "center" width= "70%" cellpadding= "5" cellspacing= "5"
border= "1">

        <caption style="color: orange; font-size: 36px"><b>Tabla general
Luminarias</b></caption>


  <tr>
    <tr><th style="color:red;">ID</th><th style="color:red;">LDR 1 (lx)</th><th
style="color:red;">LDR 2 (lx)</th><th style="color:red;">LDR 3 (lx)</th><th
style="color:red;">LDR 4 (lx)</th><th style="color:red;">LDR 5 (lx)</th><th
style="color:red;">LDR 6 (lx)</th><th style="color:red;">Fecha - Hora</th>
  </tr>

  <?php while ($row = mysqli_fetch_array($query)){ ?>
  <tr>
    <?php if ($query->num_rows > 0): ?>
    <!-- Mostrar la tabla aquí -->
    <td style="color:red;"><b><?php if (!empty($row['id'])) {
          echo $row['id'];
        } else {
          echo "-";
        } ?></b></td>
    <td style="color:blue;"><?php if (!empty($row['ldr1'])) {
          echo $row['ldr1'];
        } else {
          echo "-";
        } ?></td>
    <td style="color:blue;"><?php if (!empty($row['ldr2'])) {
          echo $row['ldr2'];
        } else {
          echo "-";
        } ?></td>
    <td style="color:blue;"><?php if (!empty($row['ldr3'])) {
          echo $row['ldr3'];
        } else {
          echo "-";
        } ?></td>
```

```
    <td style="color:blue;"><?php if (!empty($row['ldr4'])) {
          echo $row['ldr4'];
        } else {
          echo "-";
        } ?></td>
    <td style="color:blue;"><?php if (!empty($row['ldr5'])) {
          echo $row['ldr5'];
        } else {
          echo "-";
        } ?></td>
    <td style="color:blue;"><?php if (!empty($row['ldr6'])) {
          echo $row['ldr6'];
        }  else {
          echo "-";
        } ?></td>
    <td style="color:brown;"><?php if (!empty($row['Fecha'])) {
          echo $row['Fecha'];
        } ?></td>

    <?php endif; ?>

</tr>
<?php

} ?>

</table>

<!---Apéndice--->
  <footer>
    <?php echo "<br>" ;?>
    <div STYLE="border: 1px solid black; overflow: auto">
      <h3 style="float: left;">Elaborado por: Kevin David Yépez Arteaga</h3>
      <h3 style="float: right;">TSRT</h3>
    </div>
  </footer>
</html>
```

## Código de la tabla secundaria

```php
<?php

include "enviodatos.php";
error_reporting(E_ALL & ~E_WARNING);


    // Define the SELECT query to insert the filtered data into the "alertldr"
table
$sql = "SELECT id, ldr1, ldr2, ldr3, ldr4, ldr5, ldr6, Fecha FROM `filtroldr`";

$query = mysqli_query($conexion, $sql);

if (!$query) {
    die("Error en la consulta: " . mysqli_error($conexion));
}


?>

<!---Formación del documento--->
<!DOCTYPE html>
<html>
  <!---Encabezado del documento--->
  <head>
    <!---Título de la página web--->
    <title>Tabla valores filtrados</title>

    <!---Estilo de la tabla--->
    <style type="text/css">
    table{
      border-collapse: collapse;
      width: 70%;
      border-style: solid;
      border-width: 3px;
      border-color: #000000;
      font-family: Sans-serif;
      font-size: 15px;
      text-align: center;
    }

    th{
      background-color: #28BEC1;
      color: white;
    }
    tr:nth-child(even) {
      background-color: #28BEC1;
    }

    </style>
  </head>

  <!---Cuerpo del documento--->
  <body>
    <!--Formación del encabezado--->
```

```
    <div style="border: 1px solid black">
      <img src="https://univercimas.com/wp-content/uploads/2021/05/Logo-de-la-
Escuela-Politecnica-Nacional-EPN-300x300.png" alt="EPN" style=" float:left;
width: 150px; height: 170px;"/>
      <img src="https://static.esmartcity.es/media/2017/12/cci-3-figura-1-
solucion-stelaria-elt-iluminacion-inteligente.png" alt="Iluminacion" style="
float:right; width: 150px; height: 170px;"/>
      <div style="border: 1px solid black; overflow:auto;">
        <h1 style= "text-align: center;">Escuela Politecnica Nacional </h1>
        <h2 style= "text-align: center;">Trabajo Previo a la Obtención del
Título en:</h2>
        <h2 style="text-align: center;">Tecnología Superior en Redes y
Telecomunicaciones</h2>
        <h2 style= "text-align: center;"></h2>
      </div>
    </div>
    <?php echo "<br>"; ?>


<table width = "70%" align = "center" cellpadding = "5" cellspacing = "5"
border = 1>

  <caption style="color: orange; font-size: 36px"><b>Tabla filtrada con los
valores más altos de las luminarias</b></caption>

  <tr>
    <tr><th style="color:red;">ID</th><th style="color:red;">LDR 1 (lx)</th><th
style="color:red;">LDR 2 (lx)</th><th style="color:red;">LDR 3 (lx)</th><th
style="color:red;">LDR 4 (lx)</th><th style="color:red;">LDR 5 (lx)</th><th
style="color:red;">LDR 6 (lx)</th><th style="color:red;">Fecha - Hora</th>
  </tr>


  <?php while ($row = mysqli_fetch_array($query)){ ?>
  <tr>
    <?php if ($query->num_rows > 0): ?>
    <!-- Mostrar la tabla aquí -->
    <td style="color:red;"><b><?php if ($row['id']) {
          echo $row['id'];
        } else {
          echo "-";
        } ?></b></td>
    <td style="color:blue;"><?php if ($row['ldr1'] >= 70) {
          echo $row['ldr1'];
        } else {
          echo "-";
        } ?></td>
    <td style="color:blue;"><?php if ($row['ldr2'] >= 70) {
          echo $row['ldr2'];
        } else {
          echo "-";
        } ?></td>
    <td style="color:blue;"><?php if ($row['ldr3'] >= 70) {
          echo $row['ldr3'];
        } else {
```

```
            echo "-";
          } ?></td>
      <td style="color:blue;"><?php if ($row['ldr4'] >= 70) {
            echo $row['ldr4'];
          } else {
            echo "-";
          } ?></td>
      <td style="color:blue;"><?php if ($row['ldr5'] >= 70) {
            echo $row['ldr5'];
          } else {
            echo "-";
          } ?></td>
      <td style="color:blue;"><?php if ($row['ldr6'] >= 70) {
            echo $row['ldr6'];
          }  else {
            echo "-";
          } ?></td>
      <td style="color:brown;"><?php if ($row['Fecha']) {
            echo $row['Fecha'];
          } ?></td>

      <?php endif; ?>

  </tr>
  <?php

  } ?>

  </table>


  <!---Apéndice--->
    <footer>
      <?php echo "<br>" ;?>
      <div STYLE="border: 1px solid black; overflow: auto">
        <h3 style="float: left;">Elaborado por: Kevin David Yépez Arteaga</h3>
        <h3 style="float: right;">TSRT</h3>
      </div>
    </footer>
  </html>
```