CONTROL INTELIGENTE DE SISTEMAS DE ILUMINACIÓN EN EDIFICIOS

Jezzy Huamán-Rojas

Piura, abril de 2017

FACULTAD DE INGENIERÍA

Máster en Ingeniería Mecánico-Eléctrica con Mención en Automática y Optimización

Huamán, J. (2017). *Control inteligente de sistemas e iluminación en edificios* (Tesis de Máster en Ingeniería Mecánico-Eléctrica con Mención en Automática y Optimización). Universidad de Piura. Facultad de Ingeniería. Piura, Perú.

U N I V E R S I D A D D E P I U R A

FACULTAD DE INGENIERÍA

“Control inteligente de sistemas de iluminación en edificios”

Tesis Para optar el Grado de
Master en Ingeniería Mecánico –Eléctrica con
Mención en Automática y Optimización

Ing. Huamán Rojas Jezzy James

Asesor:
Dr. Ing. William Ipanaqué Alama

Piura, abril 2017

Dedicatoria:

A mi esposa que con su cariño y comprensión logra sacar lo mejor de mí a cada instante, tú eres la fuente principal que inspira mi trabajo y aprendizaje.

A mis amigos Edgar Martínez y Mario Torres quienes en vida me motivaron a dar lo mejor de mí siempre., y a quienes tengo presente a cada paso que doy, cada logro es gracias a sus consejos y el tiempo que en vida me dieron.

A mi querida madre Miriam Rojas Varillas quien a pesar de la distancia es mi mayor fortaleza, con su ejemplo forjó mi afán de lucha y superación.

A mis abuelitos Ascencio y Ervitas quienes con su amor y confianza han sido los pilares de mis éxitos.

Prologo

En la actualidad el cambio climático nos lleva a reflexionar sobre el uso y abuso que se está haciendo de las fuentes de energía, la electricidad es el contacto directo de la sociedad con la tecnología dependiendo estrechamente de ella, sin embargo no se hacen uso de tecnologías existentes para el ahorro energético, entre estas tecnologías en iluminación se cuenta con tecnología de tubos fluorescentes con arrancadores electrónicos, que no son ampliamente usados hasta ahora; también existen luminarias de tecnología led que a la actualidad son relativamente más caras pero considerando su tiempo de vida y eficiencia estás son las más viables para disminuir los consumos energéticos en iluminación, es por ello que en la presente tesis se va trabajar con la tecnología led que nos permite crear sistemas de iluminación automáticas y regulables.

En el afán de una mayor difusión de esta tecnología se ha desarrollado técnicas de control adaptativos que nos permitan instalar esta tecnología a cualquier ambiente, adaptando el sistema de iluminación de acuerdo a la función del ambiente al nivel de iluminación requerido por norma, del mismo modo permitirá la adaptación del sistema de iluminación al usuario.

Finalmente deseo expresar mi total agradecimiento a todas las personas que influyeron de manera directa o indirecta a la realización de esta tesis:

Al Gobierno del Perú que a través del CONCYTEC ha hecho posible mi ingreso a una maestría, sin su inversión hubiera sido imposible para mí, llevar una maestría de este nivel.

Al Dr. Ing. William Ipanaqué asesor de mi tesis, quien me oriento a lo largo de la maestría para culminar esta tesis exitosamente.

Al Dr. Ing. Cesar Chinguel que con su confianza y apoyo hicieron de esta maestría también una experiencia en asesoramiento e investigación en análisis de imágenes.

A mi esposa quien día a día se ha esforzado para hacer mi vida más sencilla y poder así dedicarme a esta tesis.

A todos con quienes compartimos las aulas e interminables conversaciones de cómo mejorar nuestros sistemas de control.

A todos muchas gracias.

Resumen

La presente tesis busca desarrollar un controlador adaptivo para el control inteligente de la iluminación en edificios, esto considera casas, oficinas o cualquier ambiente donde se requiera iluminar, se va presentar dos opciones de controlador adaptativo, y para considerarlo un sistema inteligente este se va adaptar al usuario haciendo uso de la tecnología de aprendizaje automático.

Los sistemas de aprendizaje automático haciendo uso de redes neuronales artificiales y árboles de decisión han sido empleados en la presente tesis para adaptar la iluminación al usuario sin dejar de cumplir las normas que se establecen por función de los ambientes, en tanto considerando la iluminación se ve afectada por muchos factores siendo estos inclusive el color de paredes y techos, los controladores no pueden ser estándar es por ello que se ha desarrollado un controlador PID adaptativo tanto con un sistema de identificación como también un sistema sin identificación.

Como conclusión se tiene un sistema embebido con árboles de decisión como sistema de adaptación al usuario y como controlador del nivel mínimo de iluminación un control PID adaptativo sin identificación.

Índice

Prologo ..... **v**

Resumen ..... **vii**

Introducción **1**

Capitulo 1: Planteamiento metodológico **3**

1.1. Introducción 3

1.2. Situación problemática 3

1.3. Justificación 5

1.4. Antecedentes 5

1.5. Objetivos 6

Capitulo 2: Marco teórico **7**

2.1. Introducción 7

2.2. La iluminación 7

2.2.1. Medidas y unidades 8

2.2.2. Niveles de iluminación 10

2.3. Modelo de la luz para iluminación 11

2.3.1. Calculo de la iluminancia 11

2.3.2. Cálculo de la iluminancia en iluminación LED 14

2.3.3. Modelo de la Iluminación en escenarios virtuales 15

2.4. Microcontroladores 16

2.4.1. El microcontrolador PIC 18

2.4.2. Arduino con microcontrolador Atmel AVR 18

2.4.3. Psoc 18

2.5. Sistemas de aprendizaje automático 18

2.5.1. Redes neuronales artificiales 19

2.5.1.1. Fundamentos biológicos de las redes neuronales 19

2.5.1.2. Red neuronal artificial 20

2.5.1.3. Función de activación sigmoidal 21

2.5.1.4. Topologías de redes neuronales artificiales 22

2.5.1.5. Aprendizaje en un perceptrón 24
2.5.1.6. Aprendizaje de red neuronal multicapa 24
2.5.1.7. Validación de las redes neuronales multicapa 32
2.5.2. Árboles de decisión 34
2.5.2.1. Aprendizaje en Árboles de decisión 35
2.5.2.2. Validación del árbol de decisión 38
2.6. Ordenadores de placa reducida 38
2.6.1. Raspberry Pi 39
2.6.2. Odroid C2 39
2.6.3. Comparación entre Raspberry Pi y Odroid 40
Capitulo 3: Diseño y control de iluminación 43
3.1. Introducción 43
3.2. Sistemas automáticos de control 43
3.2.1. Control a lazo abierto 43
3.2.2. Control a lazo cerrado 44
3.3. Control Proporcional, integral y derivativo 44
3.3.1. Modificaciones de los esquemas de control PID 46
3.3.2. Esquemas anti-windup 47
3.3.3. PID discreto 48
3.4. PID adaptativo 49
3.4.1. Método recursivo de mínimos cuadrados 49
3.4.2. PID adaptativo basado en una identificación en tiempo real 52
3.4.3. PID adaptativo sin identificación 52
3.5. Modulación por ancho de pulso 57
3.6. El control de iluminación 58
Capitulo 4: Desarrollo de software y hardware 63
4.1. Introducción 63
4.2. Sistema de aprendizaje automático 63
4.3. Sistema de control PID adaptativo 67
4.4. Pruebas en el simulador 71
4.5. Comparación entre ambos controladores 76
4.6. Diseño del hardware para el microcontrolador 80
4.7. Diseño del software para el microcontrolador 81
4.7.1. Programación por subprocesos 82
Capitulo 5: Pruebas y resultados 85
5.1. Introducción 85
5.2. Módulo de control de iluminación 85

5.3. Software de toma de datos 86
5.4. Pruebas del sensor de iluminancia 86
5.5. Pruebas de variación de factor gamma..... 88
5.6. Pruebas de variación del punto de operación..... 92

Conclusiones..... 97

Referencia Bibliográfica..... 99

ANEXOS 103

Índice de Figuras

Figura 1.- Consumo final de electricidad por sector 2004-2014 3

Figura 2.- Facturación de Energía Eléctrica por Sectores de Consumo 2014 4

Figura 3.- Ahorros energéticos por aplicación de nuevas tecnologías 4

Figura 4.- Banda de energía electromagnética 8

Figura 5.- Intensidad luminosa 9

Figura 6.- Superficie de distribución de intensidad luminosa 9

Figura 7.- Calculo de iluminancia 10

Figura 8.- Cálculo de iluminancia con ángulo 12

Figura 9.- Altura en el coeficiente de utilización 13

Figura 10.- Forma de iluminación de un led 14

Figura 11.- Calculo de posición de sensores 15

Figura 12.- Funcionamiento del Microcontrolador 17

Figura 13.- Elementos neuronales 20

Figura 14.- Red Neuronal Artificial 21

Figura 15.- Función sigmoidal 22

Figura 16.- Redes multicapa 23

Figura 17.- Red artificial con 3 capas 23

Figura 18.- Red neuronal artificial con dos salidas 27

Figura 19.- Red neuronal artificial con una salida 32

Figura 20.- Redes neuronales con diverso número de neuronas en la capa oculta 33

Figura 21.- Redes neuronales una salida y con dos salidas..... 33

Figura 22.- Redes neuronales con diferente valor de $\boldsymbol{\alpha}$ 34

Figura 23.- Primer paso por el algoritmo de aprendizaje..... 37

Figura 24.- Segundo paso por el algoritmo de aprendizaje..... 38

Figura 25.- Raspberry Pi 2 39

Figura 26.- Odroid C2 40

Figura 27.- Comparación de velocidad de procesamiento 41

Figura 28.- Comparación de velocidad de acceso a la data 41

Figura 29.- Control PID de una planta 44

Figura 30.- Control PI de una planta..... 45

Figura 31.- Interpretación de la acción derivativa como control predictivo 46

Figura 32.- Sistema de control PI-D..... 46

Figura 33.- Sistema de control I-PD..... 47

Figura 34.- Mecanismo anti-windup 48

Figura 35.- Mecanismo e interacciones de un sistema..... 53

Figura 36.- Controlador PID 54

Figura 37.- Controlador PID adaptativo sin identificación 55

Figura 38.- Modulación por ancho de pulso 58

Figura 39.- Medición de consumo de luminarias a) Luminaria fluorescente con arrancador normal, b) Luminaria fluorescente con arrancador electrónico, c) Tubo led.... 59

Figura 40.- Diagrama de flujo del control..... 60

Figura 41.- Módulo BH1750..... 60

Figura 42.- Sensor PIR 61

Figura 43.- Funcionamiento del sensor PIR..... 61

Figura 44.- Toma de datos para el sistema de aprendizaje..... 64

Figura 45.- Actualización de parámetros de sistema de aprendizaje..... 64

Figura 46.- Actualización de parámetros de sistema de aprendizaje..... 65

Figura 47.- Simulación de Escenarios: a) Habitación sin iluminación y sin personas, b) Habitación sin iluminación con personas, c) Habitación iluminada sin personas, d) Habitación iluminada con personas..... 66

Figura 48.- Árbol de decisión aprendida 67

Figura 49.- Algoritmo de control PID adaptativo con identificación..... 68

Figura 50.- Algoritmo de control PID adaptativo sin identificación..... 69

Figura 51.- Simulador de Control de Iluminación..... 70

Figura 52.- Barra de menús. a) Menú de configuración, b) Opciones de configuración del control PID adaptativo, c) Opciones de selección del sistema automático de aprendizaje..... 71

Figura 53.- Barra de menús - Graficas 71

Figura 54.- Prueba 1: Respuesta del PID adaptativo con identificación 72

Figura 55.- Prueba 1: Respuesta del PID adaptativo sin identificación 73

Figura 56.- Prueba 1: Respuesta del PID adaptativo sin identificación solución de An et al..... 73

Figura 57.- Prueba 2: Respuesta del sistema de aprendizaje. a) Redes neuronales, b) Árbol de decisión 74

Figura 58.- Prueba 2: Respuesta del PID adaptativo con identificación 75

Figura 59.- Prueba 2: Respuesta del PID adaptativo sin identificación 76

Figura 60.- Comparativa de sistemas de control en la prueba 1..... 77

Figura 61.- Rendimiento de los sistemas de control adaptativos en la prueba 1..... 78

Figura 62.- Comparativa de sistemas de control en la prueba 2..... 79

Figura 63.- Rendimiento de los sistemas de control adaptativos en la prueba 2..... 79

Figura 64.- Fase de potencia del circuito..... 80

Figura 65.- Ubicación del luxómetro BH1750 81

Figura 66.- Diagrama de Aprendizaje en el microcontrolador..... 81

Figura 67.- Diagrama de flujo de la interrupción del timer 2..... 83

Figura 68.- Diagrama de flujo de subprocesos 83
Figura 69.- Modulo de pruebas 86
Figura 70.- Software de adquisición de datos 86
Figura 71.- Funcionamiento del sensor en modo L 87
Figura 72.- Funcionamiento del sensor en modo H 88
Figura 73.- Prueba con $\boldsymbol{\gamma = 10^{-13}}$ 89
Figura 74.- Prueba con $\boldsymbol{\gamma = 10^{-12}}$ 90
Figura 75.- Prueba con $\boldsymbol{\gamma = 10^{-11}}$ 91
Figura 76.- Prueba con $\boldsymbol{\gamma = 4.10^{-12}}$ 92
Figura 77.- Prueba con punto de operación en 500 luxes 93
Figura 78.- Prueba con punto de operación en 300 luxes 94
Figura 79.- Prueba con punto de operación en 200 luxes 95

Índice de Tablas

Tabla 1.- Valores máximos de reflectancia para un fluorescente 14

Tabla 2.- Datos de entrenamiento 28

Tabla 3.- Valores de Error Máximo en MSE 31

Tabla 4.- Datos de entrenamiento extendido 31

Tabla 5.- Datos de validación 31

Tabla 6.- Datos de entrenamiento para el árbol de decisión 35

Tabla 7.- Datos de entrenamiento para el árbol de decisión modificado 36

Tabla 8.- Datos de entrenamiento tras el primer paso de elaboración 37

Tabla 9.- Datos de validación del árbol de decisión 38

Tabla 10.- Consumo energético de acuerdo a la tecnología usada 58

Tabla 118.- Datos de ejemplo para el aprendizaje desde el microcontrolador 82

Introducción

La iluminación es un tema muy amplio y estudiado por diversas ramas de la ciencia siendo necesario una buena iluminación en todo ambiente donde el ser humano se desarrolla en especial en ambientes donde se realiza trabajos con computadoras, zonas de lectura o trabajo especializado (Livingston, 2014). La iluminación también representa un 20% del consumo eléctrico en los hogares (Philips, 2011), esto para el mercado peruano equivale a unos 285 mil millones de dólares en el 2015 (Ministerio de Energía y Minas, 2015), es por ello que la presente tesis busca desarrollar un controlador que además de cumplir con lo dictaminado en normas pueda también adaptarse al usuario y funcionar de forma eficiente con lo cual se podría llegar a un ahorro de hasta el 70% (Philips, 2011) en consumo eléctrico de luminarias que equivaldría a unos 199 mil millones de dólares anuales de ahorro.

En un estudio para un sistema de control de iluminación en ambientes es necesario considerar que se va medir y como se va medir en el Capítulo 2, se ha establecido los criterios de iluminación y lo que se me mide para el cumplimiento de las normas que establecen la iluminación mínima por función del ambiente (ver Anexo A); además de entrar al enfoque de los sistemas de aprendizaje automático que se adapten al usuario, y establezca un punto de operación de las luminarias que no estén fuera de las normas pero además que de ser necesario estén por encima considerando que en muchos el usuarios requerirán niveles de iluminación superiores a los puestos en norma.

El diseño del controlador y su aplicación a luminarias es el tema del Capítulo 3, desarrollando los conceptos de un controlador adaptativo con identificación (Bobál, Böhm, Fessl, & Machácek, 2006) RLS (De Keyser, 2012) y también un controlador adaptativo sin identificación (Feng Lin, Brandt, & Saikalis, 2000), estos sistemas servirán para mantener el nivel de iluminación en un punto de operación aprendido o como mínimo lo que dicta las normas de acuerdo a la función del ambiente a iluminar.

En el capítulo 4 se ha desarrollado el software y hardware del sistema considerando primero un simulador para evaluar el rendimiento de los controladores, tras lo cual se ha sintetizado el sistema de aprendizaje automático y el controlador automático en un sistema embebido basado en un mini Arduino Pro, junto a un Odroid C2 para operar como un controlador PID adaptativo sin identificación y un sistema de aprendizaje automático basado en árboles de decisión.

En el capítulo 5 se muestran los resultados del control en tiempo real, se ha considerado evaluar primero los modos de operación del sensor BH1750, luego evaluar el factor del controlador PID sin identificación y por último se ha evaluado estos parámetros en unos puntos de operación de iluminación de acuerdo a las normas vigentes.

Capítulo 1
Planteamiento metodológico

1.1. Introducción

En este primer capítulo se plantea el porqué del desarrollo de la tesis analizando la situación problemática en el país, del mismo modo se presenta la justificación para el desarrollo de la tesis haciendo un análisis de los antecedentes se han planteado los objetivos.

1.2. Situación problemática

A nivel mundial el tema del ahorro energético y las alternativas para la generación de energías limpias son temas transversales es por ello que diversos estudios han marcado la pauta para el desarrollo de diversas gráficos comparativos donde se observa el gasto energético en todas las áreas, de ese modo el área residencial es un consumo sensible en el gasto eléctrico tal como se muestra en la figura 1 donde se puede apreciar según al Balance Energético 2014, el más reciente a la fecha de redacción, presentado por el Ministerio de Energía y Minas que el consumo nacional eléctrico nacional en el área residencial es de un 40% en el año 2014.

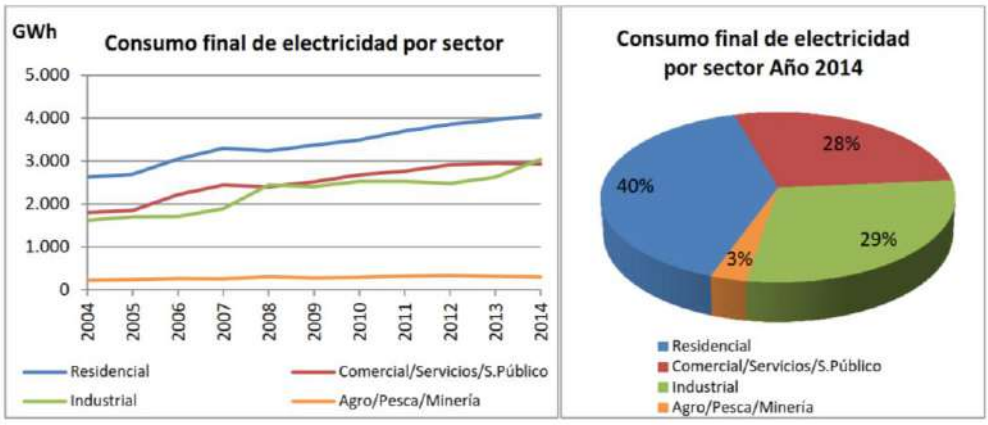


Figura 1.- Consumo final de electricidad por sector 2004-2014

Fuente: Ministerio de Energía y Minas, 2014

La figura 1 nos muestra el consumo final de la electricidad por GWh, sin embargo para calcular el potencial de ahorro, es necesario revisar el Anuario Estadístico de Electricidad (2015), donde nos muestran las facturaciones por sectores económicos y se puede analizar el gráfico circular de la figura 2.

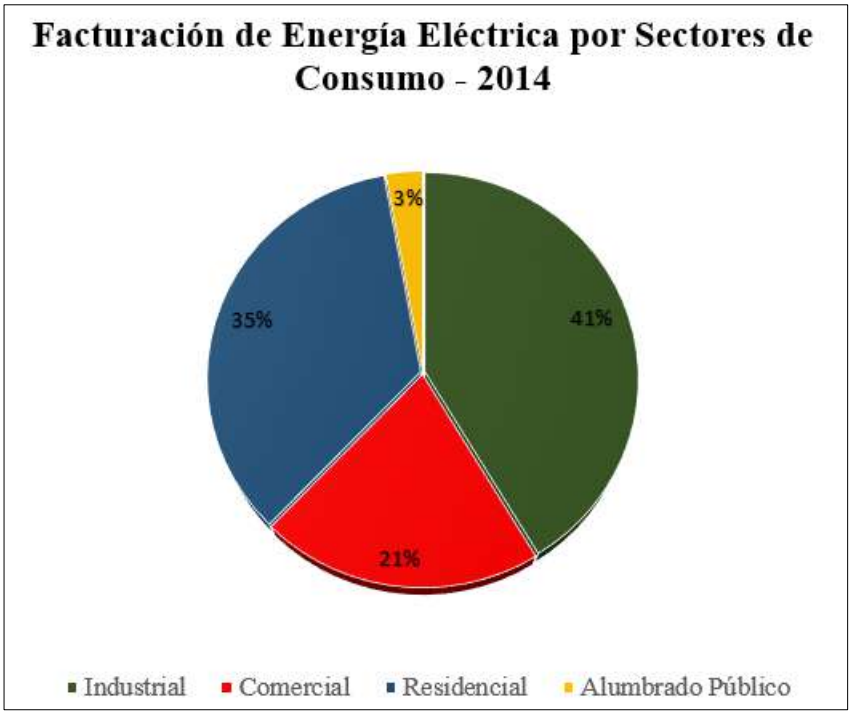


Figura 2.- Facturación de Energía Eléctrica por Sectores de Consumo 2014
Fuente: Elaboración Propia basado en datos del Ministerio de Energía y Minas, 2015

Los estudios desarrollados por la empresa Philips (2011) calculan que un total del 20% del consumo residencial se gasta en la iluminación y este gasto es debido principalmente por falta de un sistema de control y la utilización de tecnología antigua tal como es posible ver en la figura 3.

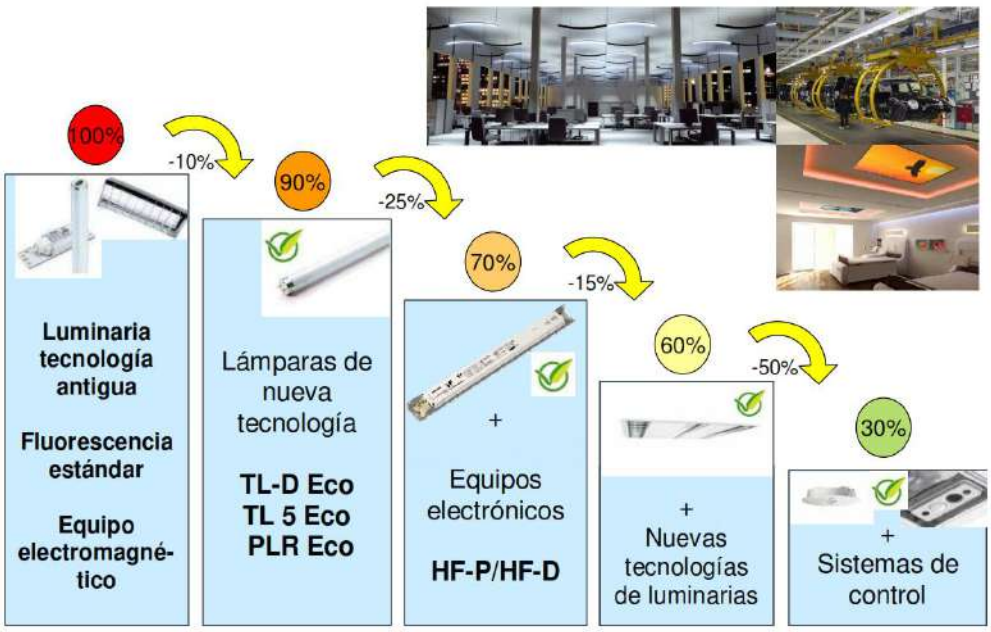


Figura 3.- Ahorros energéticos por aplicación de nuevas tecnologías
Fuente: Eficiencia energética en la Iluminación – Philips 2011

Por tanto según la figura 1 en el Perú se gasta un 8% de la energía total producida para iluminación en las residenciales que de acuerdo al Anuario Estadístico de Electricidad

(2015) son unos 281 mil millones de dólares anuales y utilizando las mejoras presentadas por la empresa Philips en la figura 3 podríamos llegar a un consumo de tan solo el 3% que significaría un ahorro de 196 mil millones de dólares anuales, considerando para ello un cambio de tecnología y la implementación de controladores inteligentes en cada hogar del Perú.

1.3. Justificación

El cambio climático es una preocupación mundial y todo ahorro es considerado como valioso, de ese modo en estudios realizados se ha concluido que con el adecuado sistema de control se logra un ahorro de gasto energético en luminarias de hasta un 70%, es por ello que en la presente tesis se va desarrollar un Control Inteligente de las luminarias considerando un estudio de un control basado en sistemas de aprendizaje automático y control adaptativo para con ello armar un prototipo utilizando sistemas embebidos, además de brindar un confort a los ocupantes de los ambientes donde se instale esta tecnología que puede ser utilizada en ambientes como:

- Oficinas.
- Sala de conferencias.
- Salas.
- Comedores.
- Dormitorios.
- Pasadizos.
- Almacenes.

1.4. Antecedentes

Previo al desarrollo de la presente tesis, como parte de un equipo de trabajo de la empresa J&H Comercialización e Ingeniería S.A. se han desarrollado algunos trabajos prácticos en diseño e instalación de sistemas automáticos de iluminación para pasadizos, con lo que se va a continuar la misma línea de investigación aportando finalmente un sistema inteligente de control.

Para encontrar los antecedentes sobre el tema se ha usado el sistema ALICIA presentado por CONCYTEC (Consejo Nacional de Ciencia Tecnología E Innovación Tecnología) que es un repositorio nacional de Tesis de Grado y Post- Grado, así mismo también se hizo uso del sistema Pirhua de la Universidad de.

Antecedentes en la Universidad de Piura:

- Iluminación Eficiente Mediante Software Dialux En Biblioteca Central De La Universidad De Piura, 2005, Guillermo Rodas Carhuajulca. Está tesis resume la manera eficiente de utilizar el software Dialux y en la parte control menciona el uso de sensores de presencia para el encendido y apagado de las luminarias.

Antecedentes encontrados en el repositorio ALICIA:

- Análisis, diseño e implementación de un prototipo de un sistema de automatización para inmuebles pequeños, 2005, Teófilo Max De La Mata Luque Pontificia Universidad Católica del Perú. En esta tesis se hace un análisis de la tecnología que se contaba en el 2005 para el desarrollo de un prototipo de un sistema de automatización con tecnología de microcontroladores PIC y CMOS.
- Sistema de Control Domótico Utilizando una Central IP PBX Basado En Software

Libre, 2012, Wally Mauro Rodríguez Bustinza, Pontificia Universidad Católica del Perú. Trabajo que resume la situación de la domótica e inmótica en el Perú para el 2012, se plantea una solución basada en un servidor web.

- Sistema Domótico de Control Centralizado con Comunicación por Línea de Poder, 2014, Miguel Ricardo Guzmán Guerra, Renzo Andreé Burga Velarde, Pontificia Universidad Católica del Perú. Se hace mención de la tecnología donde se usa la misma línea eléctrica del hogar a través del protocolo X10 para la comunicación y el manejo de tomacorrientes, cortinas y la iluminación con un controlador activado por triac.
- Implementación de una solución de domótica basado en las mejores soluciones y prácticas del mercado actual, 2015, Carlos López, Mateo Espinoza, Alfredo Barrientos Padilla, Universidad Peruana de Ciencias Aplicadas. Se hace un resumen de todas las tecnologías presentes en Perú en el área de la domótica, así también se hace un diseño de un sistema de control de iluminación con una comunicación inalámbrica usando módulos Bluetooth 4.0.

1.5. Objetivos

- Aplicación de las tecnologías de microcontroladores y su uso en un sistema de iluminación inteligente.
- Aplicación de sistemas de aprendizaje automático en el control de la iluminación.
- Aplicación de computadoras de placa reducida para la integración de sistemas embebidos en el control de la iluminación.
- Comparativa de costos de instalación y ahorro de nuevas tecnologías y sistemas de control.
- Control automático de luminarias.

Capítulo 2
Marco teórico

2.1. Introducción

El presente capitulo es un resumen del estado del arte con el que se cuenta en referencia a los temas relacionados con la tesis, de ese modo es necesario conocer conceptos como la iluminación, los microcontroladores, el aprendizaje automático y los ordenadores de placa reducida.

La iluminación como concepto para la tesis nos importa la luz visible que se mide entre los 380 y 780 nanómetros (Gordon, 2015), es en este espectro en la que se va elegir, medir y controlar la iluminación, es necesario un punto aparte tratar el modelo de la iluminación, esto principalmente para conocer las formas de calcular y la forma de como simulan los software de cálculo de iluminación.

El rápido desarrollo de las computadoras han valido también para un desarrollo en código abierto de diversos microcontroladores que son el centro de procesamiento de datos de todo sistema embebido, en la actualidad se cuenta con tecnologías de microntroladores PIC, AVR (Arduino), PSoc.

El aprendizaje automático que es una rama de la Inteligencia Artificial que hace un diverso estudio de la formas del aprendizaje de un sistema entre las principales tenemos el aprendizaje por redes neuronales y por arboles de decisión (Müller & Guido, 2017) ambas técnicas nos dan un sistema de aprendizaje automático para diversos escenarios y usuarios que usan estos sistemas.

Es necesario también contar con ordenadores de placa reducida SBC (por sus siglas en ingles "*Single Board Computer*") que son micro computadoras con las que se pueden realizar diversos cálculos más avanzados y presentar entornos gráficos, que los microcontroladores no los permiten; en el mercado existen diversos proveedores de estas SBC como son: Rasperry, Odroid, Banana Pro, Intel Galileo, etc.

2.2. La iluminación

Lo que nosotros percibimos como luz es una estrecha banda de energía electromagnética irradiada entre 380 y 780 nanómetros (Gordon, 2015), como se observa en la figura 4 lo que nuestros ojos perciben es la banda nombrada como banda visible de donde percibimos los colores y formas.

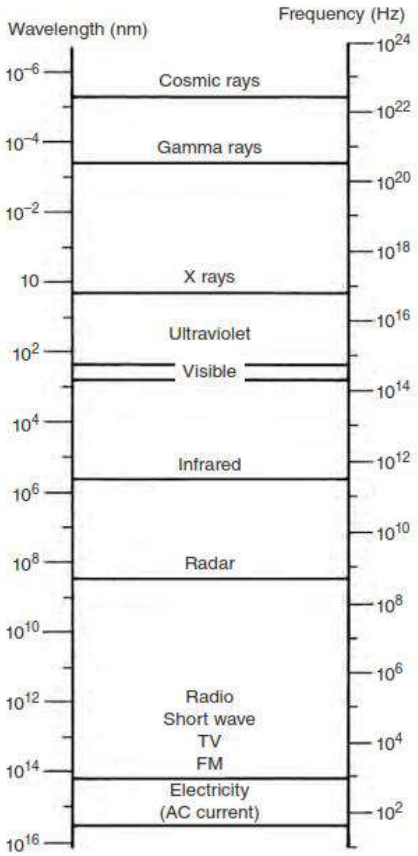


Figura 4.- Banda de energía electromagnética
Fuente: Gordon, 2015.

La luz visible irradiada por el sol nos permite trabajar cómodamente y es la que preferentemente se deberá usar como primera fuente de iluminación, la iluminación artificial se usa en caso la luz natural no sea suficiente para la labor que se está desarrollando y para la elección de la iluminación artificial se debe siempre considerar el tipo de luminaria en función a la labor para la cual se quiere iluminar.

2.2.1. Medidas y unidades

En la ingeniería es necesario tener modos de comparación cuantitativas es por ello que en electrotecnia se tiene las siguientes medidas y unidades, según el libro Como planificar con luz (2009), donde hace un resumen de las diversas técnicas para cálculos de iluminación y su aplicación.

- Flujo luminoso [Φ]

El flujo luminoso describe toda la potencia de luz dada de una fuente luminosa. Fundamentalmente, se podría registrar esta potencia de radiación como energía dada en la unidad vatio (W). No obstante, el efecto óptico de una fuente luminosa no se describe acertadamente de este modo, ya que la radiación se registra sin distinción por todo el margen de frecuencias y por ello no se tiene en cuenta la diferente sensibilidad espectral del ojo. Mediante la inclusión de la sensibilidad espectral ocular resulta la medida lumen (lm)

- Eficiencia luminosa $\left[\frac{\Phi}{P}\right]$

La eficacia luminosa describe el grado de acción de un iluminante. Se expresa

mediante la relación del flujo luminoso dado en lumen y la potencia empleada en vatios. El máximo valor teóricamente alcanzable con total conversión de la energía en luz visible sería 683 lm/W.
Las eficacias luminosas reales varían según el medio de luz, pero siempre quedan muy por debajo de este valor ideal.

- Intensidad luminosa $\left[\frac{\Phi}{\Omega}\right]$

Una fuente luminosa puntual e ideal radia su flujo luminoso de manera uniforme en todas las direcciones del espacio, su intensidad luminosa es en todas direcciones la misma. En la práctica, no obstante, siempre se da una distribución espacial irregular del flujo luminoso, que en parte es condicionada por la disposición de los medios de luz y en parte originada por la conducción consciente de la luz. Por lo tanto, es conveniente indicar una medida para la distribución espacial del flujo luminoso, es decir, la intensidad luminosa de la luz.

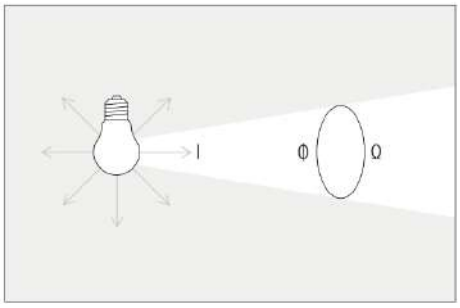


Figura 5.- Intensidad luminosa
Fuente: Ganslandt & Hofmann, 2009.

La candela como unidad de la intensidad luminosa es la única unidad base de la luminotecnia, de la cual se derivan todas las demás medidas luminotécnicas.
La distribución espacial de la intensidad luminosa de una fuente de luz da una superficie de distribución de intensidad luminosa tridimensional como se observa en la figura 6.
La sección por este cuerpo de distribución de intensidad luminosa produce la curva de distribución de intensidad luminosa, que describe la distribución de intensidad luminosa en un nivel. La intensidad luminosa se anota con ello normalmente en un sistema de coordenadas polares como función del ángulo de irradiación.
Para poder comparar directamente la distribución de la intensidad luminosa de diferentes fuentes de luz, las indicaciones se refieren cada vez a 1000 lm del flujo luminoso.

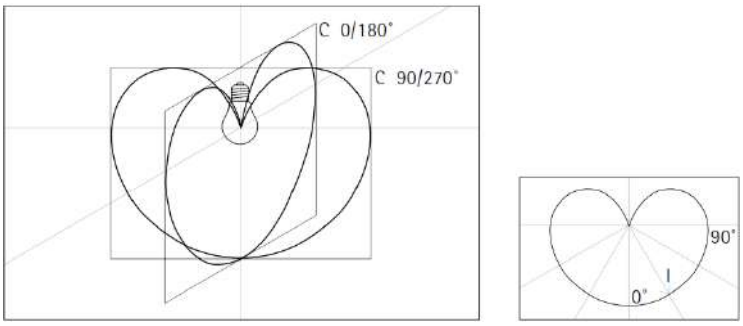


Figura 6.- Superficie de distribución de intensidad luminosa
Fuente: Ganslandt & Hofmann, 2009.

- Iluminancia

La iluminancia es una medida para la densidad del flujo luminoso. Se ha definido como la relación del flujo luminoso que cae sobre una superficie y el área de la misma. La iluminancia no está sujeta a una superficie real, se puede determinar en cualquier lugar del espacio, y puede derivar de la intensidad luminosa. La iluminancia, además, disminuye con el cuadrado de la distancia desde la fuente de luz (ley fotométrica de distancia). Su unidad de medida son los lux.
La iluminancia en un punto se calcula según la siguiente ecuación:

$$E_p[lux] = \frac{I}{a^2}$$

Donde:
I: intensidad lumínica en lumenes.
a: distancia de la luminaria en metros.

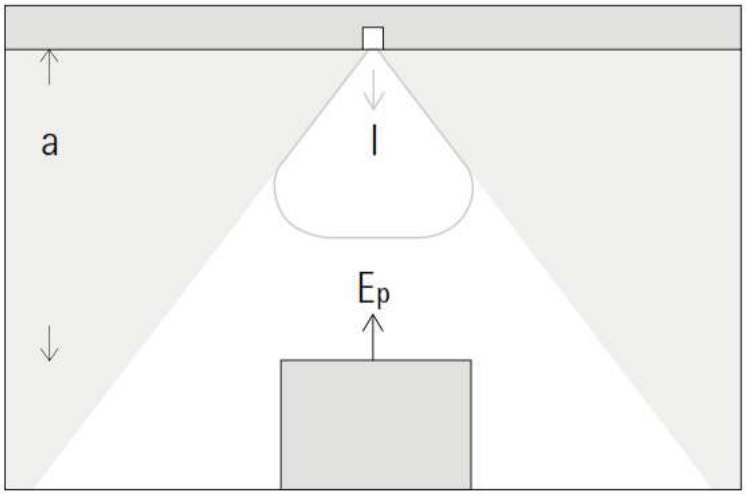


Figura 7.- Calculo de iluminancia
Fuente: Ganslandt & Hofmann, 2009.

- Luminancia

Mientras la iluminancia registra la potencia de luz que cae sobre una superficie, la luminancia describe la luz que procede de esta superficie. Esta luz, sin embargo, puede partir por sí misma de esta extensión (por ejemplo, con una luminancia de lámparas y luminarias). Aquí la luminancia se define como la relación de la intensidad luminosa y la superficie proyectada verticalmente a la dirección de irradiación.
No obstante, la luz también puede ser reflejada o transmitida por la superficie.
Con ello, la luminancia constituye la base de la claridad percibida; la sensación real de claridad, no obstante, aún queda bajo la influencia del estado de adaptación del ojo, de las proporciones de contraste del entorno y del contenido de información de la superficie vista.

2.2.2. Niveles de iluminación

La iluminación es un factor muy importante en los ambientes donde están las personas, debido que la mayor parte de información que procesamos la recibimos de los ojos, es por ello que se cuenta con diversas normas nacionales e internacionales sobre los niveles mínimos de iluminación, en Perú rige el Reglamento Nacional de

Edificaciones, el Código Nacional de Electricidad y los Decretos Generales de electricidad. La norma EM.10 del Reglamento Nacional de Edificaciones 2006 vigente hasta la actualidad nos da una referencia sobre los niveles de iluminación en luxes, es así que encontramos que en oficinas son necesarios desde 200 hasta 750 luxes o en hogares los dormitorios requieren 50 luxes en el ambiente general y 200 en la cabecera, la tabla completa con todos los valores específicos se encuentran en el Anexo A.

La Norma Nº DGE 017-AL-1-192, norma de alumbrado de interiores y campos deportivos emitida en 1982 también sirve de guía para determinar el nivel de luminosidad requerida en diversos ambientes de acuerdo a su función.

Actualmente en los niveles de iluminación no se establece de acuerdo al usuario sino de acuerdo a la función del ambiente sin embargo cuando los usuarios son adultos mayores se debe considerar una mayor iluminación que con usuarios jóvenes, esto debido que de acuerdo a la edad los ojos van envejeciendo y las pupilas se hacen más pequeñas recibiendo menor luz que un ojo joven, los resultados demuestran que las personas mayores de 60 años requieren el doble de iluminación que una persona de 20 años (Livingston, 2014), son estas las características que la presente tesis busca solucionar a través de un sistema de autoaprendizaje.

2.3. Modelo de la luz para iluminación

La iluminancia es lo que se registra por los luxómetros y es la base para el cumplimiento de los diversos reglamentos del mínimo de iluminación en ambientes habitados, los cálculos clásicos de la iluminancia han sido ampliamente estudiados en diversos libros como *Design with light* de Jason Livingston, así también lo ha estudiado Gary Gordon en su libro *Interior Lighting for Designer* y con el cambio de tecnología hacia las luminarias LED ha sido necesario nuevos estudios como lo hizo Bhardwaj, Ozcelebi, Verhoeven, & Lukkien, (2011) en cuyo paper se detalla los cálculos de la iluminancia por luminarias LED.

Del mismo modo la generación de escenarios virtuales hace necesario el modelado de la iluminación para desarrollo de imágenes tridimensionales, este se determina a partir de las fuentes de luz y su posición en coordenadas x, y, z para de ese modo generar un modelo punto a punto de la intensidad de luz en imágenes de tres dimensiones generadas por software especializados como lo es Dialux, en la presente tesis se hace mención al repositorio publicado por Ritschel, Dachsbacher, Grosch, & Kautz (2012).

2.3.1. Calculo de la iluminancia

Los libros *Design with light* e *Interior lighting for designers* mencionan la siguiente técnica de cálculo medio de la iluminancia que considerando la ecuación de iluminancia en un punto:

$$E_p[lux] = \frac{I}{a^2}$$

En caso de que el punto de iluminación no incida de forma directa la siguiente formula considera el ángulo θ de la figura 8:

$$E_p[lux] = \frac{I}{D^2}.cos(\theta)$$

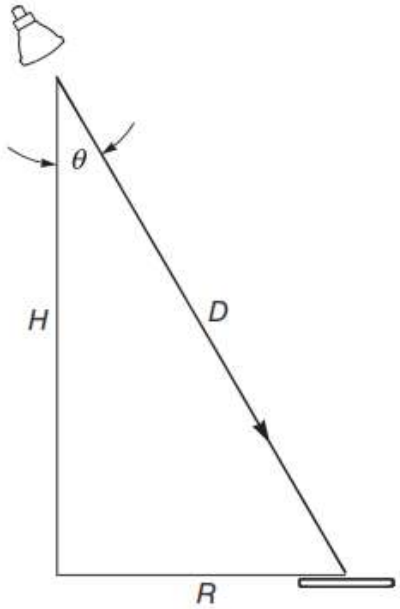


Figura 8.- Cálculo de iluminancia con ángulo
Fuente: Gordon, 2015.

Como base de estas fórmulas en áreas mayores una rápida forma de calcular la iluminancia media es usando la siguiente formula, conocido como cálculo de la iluminancia media:

$$E(fc) = \frac{n.lux_{lamp}.F_p.F_u}{A}$$

Donde:
$E(fc)$: es la media de la iluminación en lux
A: Área del plano del suelo.
lux_{lamp}: Iluminación inicial de las lámparas
F_p: Factor de pérdidas, que considera el desgaste normal de las luminarias, además de los posibles cambios en los voltajes de entrada y demás. Se analiza tres partes para considerar un valor a este coeficiente, primero la depreciación en la iluminación, cuyo valor se considera en tablas según el grado de mantenimiento, segundo un factor de suciedad, y por ultimo un factor del balasto. Para diseño donde el ambiente sea muy limpio el valor de F_p es de alrededor de 0.85, para ambientes limpios es 0.75, en ambientes sucios 0.65 y el valor de 0.55 para ambientes muy sucios.
F_u: Coeficiente de utilización, es la cantidad de iluminación que no es atrapada ni se pierde dentro de la luminaria en relación con el total producido.
Se calcula según la proporción del área de la habitación a iluminar:

$$P_h = \frac{5.h.(l+w)}{l.w}$$

Dónde: h representa la altura de utilización (figura 9), l representa la longitud y w el ancho de la habitación.

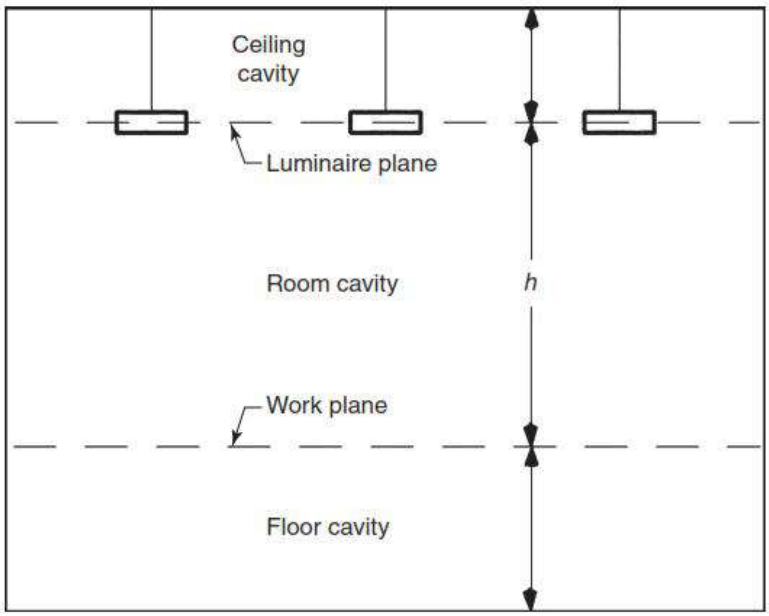


Figura 9.- Altura en el coeficiente de utilización
Fuente: Gordon, 2015.

Con el valor de la proporción del área a iluminar se busca en tablas de fabricantes para encontrar el valor del coeficiente de utilización de acuerdo también a los valores de reflectancia del techo y las paredes.

Ejemplo 1:

Calcular la iluminancia media en un área de trabajo de una oficina que mide 6 metros de largo por 4 metros de ancho, considere que se utilizara 6 pares de fluorescentes que aportan 26 watts cada una, así también el ambiente al ser una oficina tiene un mantenimiento constante, las paredes son blancas y se considera un alto nivel de reflectancia.

Se tiene:

$$E(f\ ) = \frac{n\ \ l\imath\ \ _{la}\quad xF_p xF_u}{A}$$

n: 16 luminarias.

$l\imath\ \ _{la}$: del catálogo de luminarias se puede encontrar que un fluorescente de 26 watts aporta 1800 lúmenes.

F_p: 0.85

F_u: Primero se calcula el valor de $P_h = \frac{5.h.(l+w)}{l.w}$

$$P_h = \frac{5.2.\,(6+4)}{6.4} = \frac{100}{24} \approx 4$$

Por lo tanto considerando los máximos valores de reflectancia tanto del techo como de las paredes que se muestran en la tabla 1, se obtiene que el factor de utilización es 0.69.

$$E(f\ ) = \frac{12x1800x0.85x0.69}{24} \approx 528\ l\imath$$

De acuerdo al Anexo A, se puede confirmar que este nivel de iluminación es el

adecuado para una oficina.

Tabla 1.- Valores máximos de reflectancia para un fluorescente

RC	80				70			
RW	70	50	30	10	70	50	30	10
0	86	86	86	86	84	84	84	84
1	82	80	78	76	80	78	76	75
2	77	74	71	68	76	72	70	67
3	73	68	64	61	71	67	63	60
4	69	63	58	55	67	62	58	55
5	64	58	53	49	63	57	52	49
6	60	53	48	44	59	52	48	44
7	56	48	43	39	55	48	43	39
8	52	44	39	35	51	43	38	35
9	48	39	34	31	47	39	34	31
10	44	36	30	27	43	35	30	27

Fuente: Gordon, 2015.

2.3.2. Cálculo de la iluminancia en iluminación LED

El cambio tecnológico ha llevado a nuevas perspectivas al momento de calcular la iluminancia, de ese modo se tiene el siguiente modelo tomado del paper Bhardwaj et al., 2011; donde se menciona que un led ilumina en forma circular como se muestra en la figura 10, donde el ángulo θ vale entre 20 y 30 grados sexagesimales, h representa la distancia entre el led y el plano a iluminar, y r se calcula a partir de la altura y la tangente del ángulo, de acuerdo a la siguiente ecuación:

$$r = h.t\left(\frac{\theta}{2}\right)$$

La posición del sensor no tiene que ser necesariamente en el centro debajo del led tiene que ver con el ancho δ que se calcula con el número de ledes de la luminaria y r (figura 11).

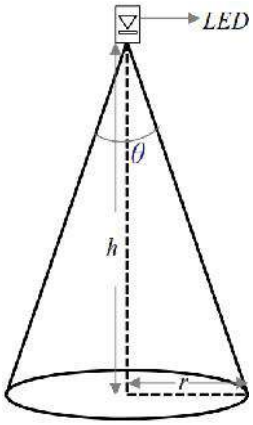


Figura 10.- Forma de iluminación de un led

Fuente: Bhardwaj, S. et al, 2011.

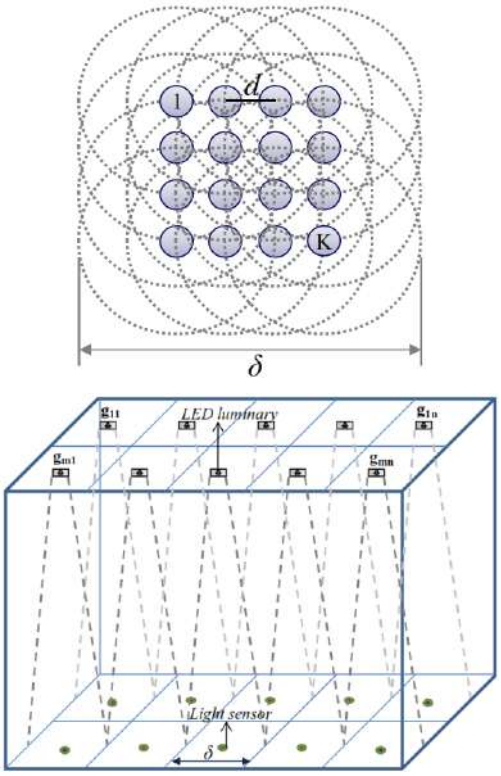


Figura 11.- Calculo de posición de sensores
Fuente: Bhardwaj, S. et al, 2011.

$$\delta = 2.r + (c-1).d$$

Tomándose como K el número de ledes de una luminaria, se tiene $K = c^2$

Para el cálculo del número de ledes necesarios en un cuarto habitado, se presenta la siguiente ecuación:

$$N°\text{-}Ledes = \frac{E.h^2}{i.F_u}$$

Donde:

E: iluminancia media del ambiente.

h: es la distancia de la altura desde la luminaria hasta la zona de trabajo.

i: Lúmenes que puede generar un led.

F_u: Factor de utilización que se puede considerar 0,77 y varía según la fabricación del led.

2.3.3. Modelo de la Iluminación en escenarios virtuales

El repositorio escrito por Ritschel et al., (2012) hace mención a la técnica desarrollada por Zhukov, Iones, & Kronin, (1998), ha ido cambiando el punto de vista del análisis con los años para mejorar la renderización y la velocidad sin embargo en aplicaciones donde el escenario no es en tiempo real esta técnica presentada por Zhukov sigue siendo usada y se describe a continuación.

Tomamos el punto P que es un punto en una escena, x pertenece a una unidad del hemisferio hS^2 centrado en P, alineado con la superficie normal en P y situándose en la parte externa de la superficie. La función $L(P, x)$ es definida como sigue:

$$L(P,x) = \begin{cases} dist(P,C) \text{ donde } C \text{ es el primer punto de intersección} \\ \quad \text{de la proyección } Px \text{ con la escena} \\ +\infty, \text{en cualquier otro caso} \end{cases}$$

Es necesario notar que para una correcta construcción de escenarios hay que considerar que las proyecciones que salen del punto P siempre tocan enfrente antes que el fondo.

Se asume que mientras la zona está más abierta más se reduce la intensidad. Por tanto la intensidad de una zona se puede tomar de la siguiente forma:

$$I(P) = \frac{2}{\pi} x I_A x \iint\limits_{x \in hS^2} \rho\big(L(P,x)\big)\, cos\alpha dx$$

Donde:

$\rho\big(L(P,x)\big)$: una función de mapeo empírico de los mapas de distancia $L(P,x)$ hacia la primera zona oscura.

α: El ángulo de la dirección de x hacia la normal.

I_A: Potencia de la luz en el ambiente

La función $\rho\big(L(P,x)\big)$ es una función monótona ascendente, por tanto necesariamente asumimos que el punto más lejano esta oscuro y que esta función tiene un límite superior. Por lo que si $L > L_{max}$ entonces $\rho\big(L(P,x)\big) = 1$, se va asumir que en toda la proyección de la dirección donde L sea menor que el máximo no existe intersección.

Definiendo la oscuridad tenemos:

$$w(P) = \frac{2}{\pi} x \iint\limits_{x \in hS^2} \rho\big(L(P,x)\big)\, cos\alpha dx$$

De donde se puede deducir que la oscuridad total es 1 y la máxima iluminación es 0.

Al momento de agregar una fuente de luz en caso se quiera simular una luminaria, la forma sencilla propuesta por Ritschel et al., (2012) explican la siguiente ecuación:

$$I(P) = (I_A + I_S).w(P) + I_S$$

Donde:

I_A: Potencia de la luz en el ambiente

I_S: La suma directa de intensidades de las fuentes de luz definido por:

$$I_S = \sum_{j=1}^{N} \delta(j).\frac{I_j}{r_j^2} cos\alpha_j$$

Donde:

N: es el número de fuentes de iluminación

I_j: es la intensidad de luz que entrega la fuente j

r_j: es la distancia entre la fuente de luz hacia el punto de la zona

$\delta(j)$: equivale a 1 si influye la luminosidad en el punto P y 0 en cualquier otro caso.

De ese modo es posible calcular la iluminancia en cada punto para un escenario en tres dimensiones.

2.4. Microcontroladores

Un microcontrolador (abreviado μC, UC o MCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un

microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Algunos microcontroladores pueden utilizar palabras de cuatro bits y funcionan a velocidad de reloj con frecuencias tan bajas como 4 kHz, con un consumo de baja potencia (mW o microvatios). Por lo general, tendrá la capacidad para mantener la funcionalidad a la espera de un evento como pulsar un botón o de otra interrupción.

Cuando es fabricado, el microcontrolador no contiene datos en la memoria ROM. Para que pueda controlar algún proceso es necesario generar o crear y luego grabar en la EEPROM[1] o equivalente del microcontrolador algún programa, el cual puede ser escrito en lenguaje ensamblador u otro lenguaje para microcontroladores; sin embargo, para que el programa pueda ser grabado en la memoria del microcontrolador, debe ser codificado en sistema numérico hexadecimal que es finalmente el sistema que hace trabajar al microcontrolador cuando éste es alimentado con el voltaje adecuado y asociado a dispositivos analógicos y discretos para su funcionamiento.

El funcionamiento del microcontrolador se realiza a través de su velocidad de reloj que depende del cristal de cuarzo instalado o de la configuración del oscilador interno; el microcontrolador tiene un número de ciclos de programa grabado en su memoria ROM con la velocidad de reloj se determina cuantas veces por segundo puede recorrer todo el programa grabado, para cada modelo de microcontrolador se puede o no dividir los tiempos de reloj, en la familia de microcontroladores PIC cada ciclo del programa se realiza en 4 tiempos del reloj.

El diagrama de flujo de la figura 12 muestra el funcionamiento del microcontrolador considerando las tareas programas en la ROM y la existencia de interrupciones.

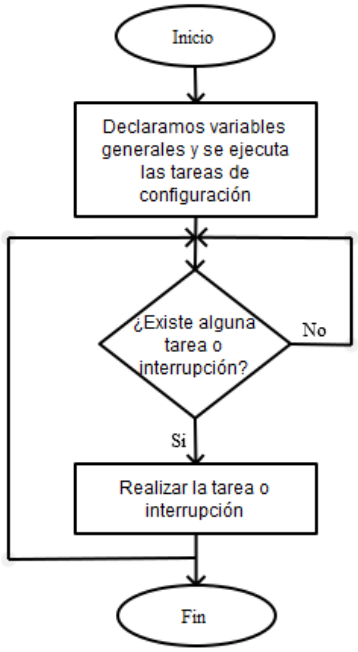


Figura 12.- Funcionamiento del Microcontrolador

Fuente: Propia (Elaborado con Pencil)

1 Son las siglas de *Electrically Erasable Programmable Read-Only Memory* (ROM programable y borrada eléctricamente).

2.4.1. El microcontrolador PIC

Los PIC son una familia de microcontroladores tipo RISC fabricados por Microchip Technology Inc. y derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de General Instrument.

En 1985 la división de microelectrónica de General Instrument se separa como compañía independiente que es incorporada como filial (el 14 de diciembre de 1987 cambia el nombre a Microchip Technology y en 1989 es adquirida por un grupo de inversores) y el nuevo propietario canceló casi todos los desarrollos, que para esas fechas la mayoría estaban obsoletos. El PIC, sin embargo, se mejoró con EEPROM para conseguir un controlador de canal programable.

2.4.2. Arduino con microcontrolador Atmel AVR

Arduino forma parte del concepto de hardware y software libre y está abierto para uso y contribución de toda la sociedad. Arduino es una plataforma de prototipos electrónicos, creado en Italia, que consiste básicamente en una placa microcontrolador, con un lenguaje de programación en un entorno de desarrollo que soporta la entrada y salida de datos y señales. Fue creado en el año 2005 con el objetivo de servir como base para proyectos de bajo coste y es lo suficientemente simple para ser utilizado por los desarrolladores.

Arduino es flexible y no requiere de un profundo conocimiento sobre el campo de la electrónica, lo que hizo que fuera muy popular entre los artistas y principiantes, además de los desarrolladores experimentados que no tienen acceso a más plataformas complejas.

Arduino es una plataforma de computación física (son sistemas digitales conectados a sensores y actuadores, que permiten construir sistemas que perciben la realidad y responden con acciones físicas), basada en una simple placa microcontrolador de entrada/salida y desarrollada sobre una biblioteca que simplifica la escritura de la programación en C/C++.

2.4.3. Psoc

La compañía Cypress Semiconductor desarrollo estos circuitos integrados como un potente microcontrolador que tiene tres espacios de memoria separados: SRAM para los datos, Flash para instrucción y registros de entrada y salida.

Estos microcontroladores están especializados en el tratamiento de entradas analógicas contando con bloques analógicos como filtros para ser agregados desde la programación, desde la versión del PSoC 4 se hace uso de un núcleo ARM Cortex M0 de 32 bits, que le dan una potencia de procesamiento muy superior a otros microcontroladores, en contra parte su programación es de mayor complejidad.

2.5. Sistemas de aprendizaje automático

El aprendizaje automático es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permiten a las computadoras aprender. En los últimos años este campo ha sido muy explotado generándose nuevos conocimientos prácticamente a diario, autores como Raschka, (2015) en su libro *Python machine learning* describen al aprendizaje automático como una herramienta para convertir la data en conocimiento, esto debido que actualmente la humanidad produce y obtiene

información del mundo que le rodea por millones de bites al día, pero en si estos datos no generan conocimiento si no son analizados.

En el desarrollo de la presente tesis se va hacer uso de redes neuronales y árboles de decisión como sistemas de aprendizaje automático en la toma de decisión al encender o apagar una luminaria o un conjunto de estas en los diversos ambientes.

2.5.1. Redes neuronales artificiales

El libro Inteligencia artificial con aplicaciones a la ingeniería del Dr. Ponce Cruz, (2010) define las redes neuronales artificiales como aproximadores no lineales a la forma en que funciona el cerebro; por lo tanto no deben compararse directamente con el cerebro ni confundir los principios que fundamentan el funcionamiento de las redes neuronales artificiales y el cerebro, ni pensar que las redes neuronales se basan únicamente en las redes biológicas ya que sólo emulan en una parte muy simple el funcionamiento del cerebro humano. Además se debe considerar que las redes biológicas son generadoras de procesos neurobiológicos en que se establecen relaciones de complejidad muy alta, las cuales no se puede lograr con redes monocapas ni con redes multicapas.

Son conocidas como aproximadores universales desde el punto de vista matemático.

2.5.1.1. Fundamentos biológicos de las redes neuronales

Una neurona biológica es una célula especializada en procesar información (Ponce Cruz, 2010). Está compuesta por el cuerpo de la célula (soma) y dos tipos de ramificaciones: el axón y las dendritas. La neurona recibe las señales (impulsos) de otras neuronas a través de sus dendritas y transmite señales generadas por el cuerpo de la célula a través del axón. La figura 13 muestra los elementos de una red neuronal natural.

En general, una neurona consta de un cuerpo celular más o menos esférico, de 5 a 10 micras de diámetro, del que salen una rama principal, el axón y varias ramas más cortas que corresponden a las dendritas.

Una de las características de las neuronas es su capacidad de comunicarse. En forma concreta, las dendritas y el cuerpo celular reciben señales de entrada; el cuerpo celular las combina e integra y emite señales de salida. El axón transmite dichas señales a las terminales axónicos, los cuales distribuyen la información. Se calcula que en el cerebro humano existen aproximadamente 10^{15} conexiones.

Las señales que se utilizan son de dos tipos: eléctricas y químicas. La señal generada por la neurona y transportada a lo largo del axón es un impulso eléctrico, mientras que la señal que se transmite entre los terminales axónicos de una neurona y las dendritas de la otra es de origen químico.

Para establecer una similitud directa entre la actividad sináptica y la analogía con las redes neuronales artificiales podemos considerar que las señales que llegan a la sinapsis son las entradas a la neurona; éstas son ponderadas (atenuadas o simplificadas) a través de un parámetro denominado peso, asociado a la sinapsis correspondiente. Estas señales de entrada pueden excitar a la neurona (sinapsis con peso positivo) o inhibirla (peso negativo).

El efecto es la suma de las entradas ponderadas. Si la suma es igual o mayor que el umbral de la neurona, entonces la neurona se activa (da salida). Esta es una

de todo o nada; cada neurona se activa o no se activa. La facilidad de transmisión de señales se altera mediante la actividad del sistema nervioso. Las sinapsis son susceptibles a la fatiga, deficiencia de oxígeno y la presencia de anestésicos, entre otros. Esta habilidad de ajustar señales es un mecanismo de aprendizaje.

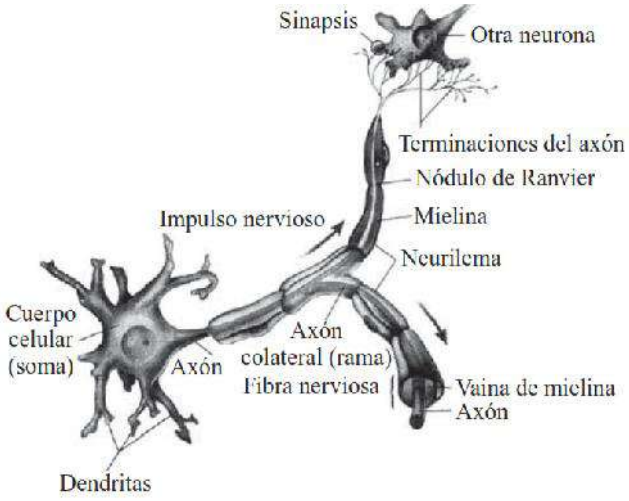


Figura 13.- Elementos neuronales
Fuente: Ponce Cruz, 2010

2.5.1.2. Red neuronal artificial

Las redes neuronales artificiales (RNA) se definen según (Ponce Cruz, 2010) como sistemas de mapeos no lineales cuya estructura se basa en principios observados en los sistemas nerviosos de humanos y animales. Constan de un número grande de procesadores simples ligados por conexiones con pesos. Las unidades de procesamiento se denominan neuronas. Cada unidad recibe entradas de otros nodos y genera una salida simple escalar que depende de la información local disponible, guardada internamente o que llega a través de las conexiones con pesos.
Pueden realizarse muchas funciones complejas dependiendo de las conexiones.
Las neuronas artificiales simples fueron introducidas por McCulloch y Pitts en 1943. Una red neuronal se caracteriza por los siguientes elementos:

a) Un conjunto de unidades de procesamiento o neuronas.
b) Un estado de activación para cada unidad, equivalente a la salida de la unidad.
c) Conexiones entre las unidades, generalmente definidas por un peso que determina el efecto de una señal de entrada en la unidad.
d) Una regla de propagación, que determina la entrada efectiva de una unidad a partir de las entradas externas.
e) Una función de activación que actualiza el nuevo nivel de activación basándose en la entrada efectiva y la activación anterior.
f) Una entrada externa que corresponde a un término determinado como bias para cada unidad.
g) Un método para reunir la información, correspondiente a la regla del aprendizaje.
h) Un ambiente en el que el sistema va a operar, con señales de entrada e incluso señales de error.

En muchas redes las unidades de proceso tienen respuesta de la forma:

$$y = f\left(\sum_k \omega_k . x_k\right)$$

Donde:

x_k: son señales de la salida de otros nodos o entradas externas.

ω_k: pesos de las ligas de conexión

$f(.)$: función no lineal simple

Una red puede tener una estructura arbitraria, pero las capas que contienen estas estructuras están definidas de acuerdo con su ubicación en la topología de la red neuronal. Las entradas externas son aplicadas en la primera capa, y las salidas se consideran la última capa. Las capas internas que no se observan como entradas o salidas se denominan capas ocultas. Por convención, las entradas no se consideran como capa porque no realizan procesamiento, la figura 14 nos muestra el esquema básico de una red neuronal artificial.

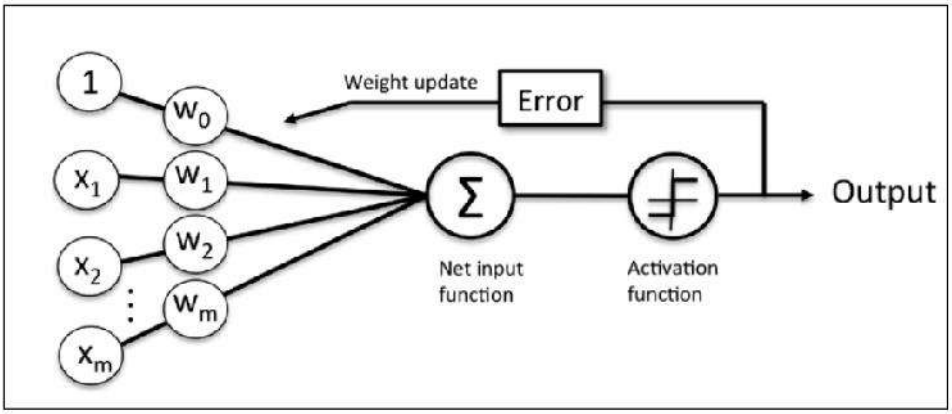


Figura 14.- Red Neuronal Artificial

Fuente: Raschka, 2015

2.5.1.3. Función de activación sigmoidal

Una red neuronal artificial como la mostrada en la figura 14 es el modelo básico que se puede operar teniendo una sola entrada x, un bias b y un peso w, donde se opera la salida del siguiente modo:

$$y = f(w.x + b)$$

Donde "y" esta es la respuesta a la función de activación también conocida como función de transferencia, se puede tener los siguientes valores y operarlos.

$$x = 3$$
$$w = 0.5$$

b=1

$$y = f(2.5)$$

Como se ha menciona depende totalmente de la función de transferencia que puede ser lineal o no lineal.

Entre las funciones lineales tenemos:

- Función de escalón
- Función Lineal

Funciones de activación no lineales

- Función sigmoidal

- Función tangente hiperbólica

De todas estas opciones la más usada en redes multicapa es la función sigmoidal (Hagan, Beale, De Jess, & Demuth, 2014) porque es derivable en cualquier punto, en la figura 15 se observa su gráfica y su función está representada por:

$$f(u) = \frac{1}{1 + e^{-u}}$$

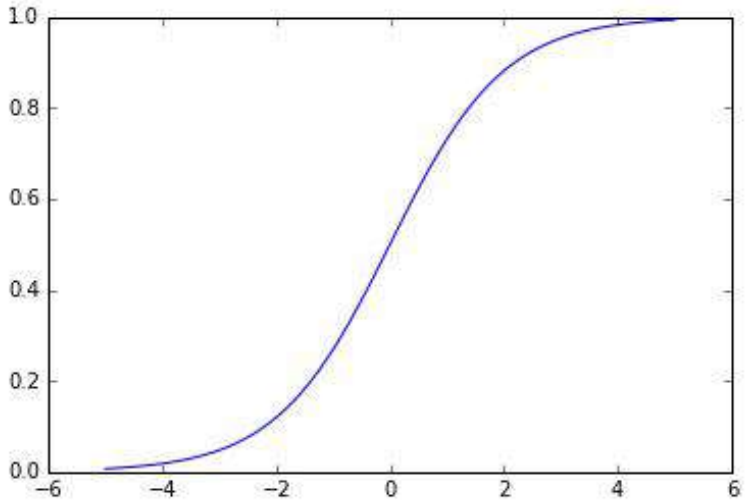


Figura 15.- Función sigmoidal
Fuente: Propia (elaborado con Python)

La principal ventaja que presenta este tipo de activación se ve en el algoritmo de aprendizaje debido a que la función sigmoidal tiene derivada en cualquier punto.

2.5.1.4. Topologías de redes neuronales artificiales

Existen dos principales topologías la primera es una red monocapa, donde la relación entre entradas y salidas es directa, y una segunda topología conocida como redes multicapa como el que muestra la figura 16.

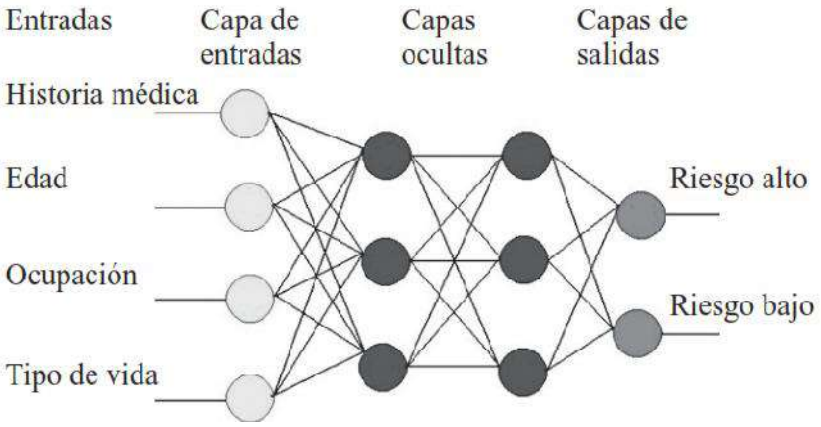


Figura 16.- Redes multicapa
Fuente: Ponce Cruz, 2010

Como se observa en la figura 16 las redes multicapa cuentan con una capa de entradas, un número determinado de capas ocultas y una capa de salida, una red de este tipo se puede describir de forma matricial considerando una matriz W para los

pesos de cada entrada en relación con cada neurona, un vector b del bias correspondiente a cada capa, las entradas representadas por un vector p, y la salida como un vector a.

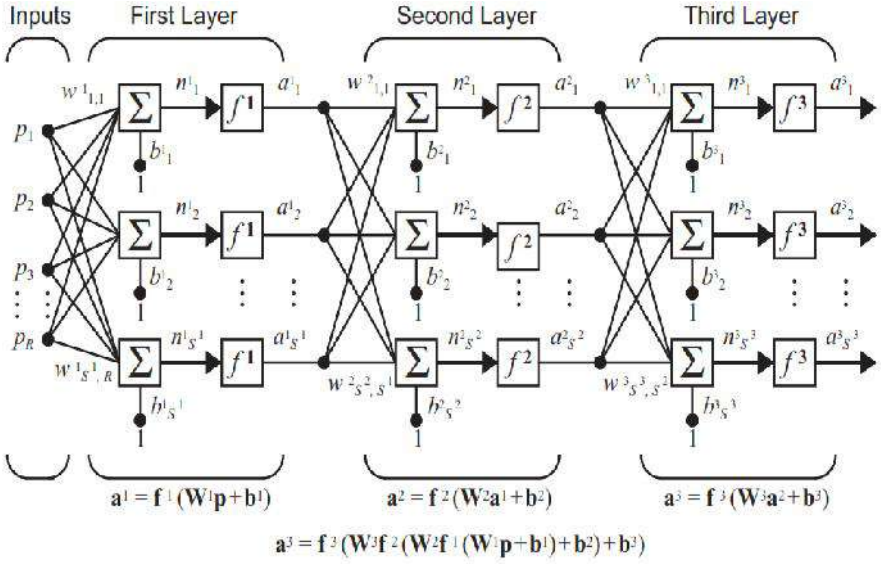


Figura 17.- Red artificial con 3 capas

Fuente: (Hagan et al., 2014)

En la figura 17 se muestra otra forma de representar las redes neuronales y las ecuaciones que definen la primera capa son las siguientes:

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_R \end{bmatrix}$$

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{1,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,1} & \cdots & w_{S,R} \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_S \end{bmatrix}$$

La representación de la salida de la primera red neuronal artificial se define como:

$$a = f(W.p + b)$$

En la segunda capa las entradas son ahora la salida de la primera capa por tanto tenemos lo siguiente:

$$a^2 = f(W^2 f(W.p + b) + b^2)$$

Donde los superíndices en la ecuación simbolizan el número de capa.

Para un M números de capas tenemos la siguiente representación:

$$a^{m+1} = f^{m+1}(W^{m+1}a^m + b^{m+1})$$

Para $m = 1,2,3, \dots, M-1$

$$a^0 = p$$

Las salidas del sistema son cuando $a = a^M$

2.5.1.5. Aprendizaje en un perceptrón

El aprendizaje de las redes neuronales artificiales se puede definir de forma sencilla en considerando el error en un perceptrón (Hagan et al., 2014):

$$e = t - a$$

Donde:

e: error del perceptrón

t: salida correcta

a: valor encontrado por la red neuronal

Considerando el error, los nuevos valores de la matriz de pesos W puede volverse actualizar del siguiente modo:

$$W_n = W_a + p.e$$

Donde:

p: representa a las entradas

Al momento de entrenar una red neuronal de una sola capa, podemos también colocar el bias, que representa el off set de la entrada y esta se puede establecer con la siguiente ecuación.

$$b_n = b_a {}_o + e$$

Una red neuronal de una sola neurona no podría manejar decisiones tan complejas es por ello que se hace necesario estudiar también el aprendizaje de una red neuronal con varias neuronas que viene hacer la secuencia del anterior modo de aprendizaje pero aplicándolo neurona a neurona y de forma matricial.

$${}_iW_n = {}_iW_a + {}_ip^T.e$$

Y el bias:

$${}_ib_n = {}_ib_a + e$$

2.5.1.6. Aprendizaje de red neuronal multicapa

Las redes multicapa requieren un algoritmo más sofisticado para el aprendizaje, el algoritmo de retropropagación conocido como *backpropagation* (Hagan et al., 2014) basándose en el método del descenso de gradiente logra un cálculo de los pesos de la red neuronal aceptables.

El algoritmo requiere valores de entrada con sus respectivos valores de salida, que son el comportamiento del sistema.

El algoritmo modificara los valores de los pesos y los bias minimizando el cuadrado del error medio.

Para el desarrollo del aprendizaje se requiere muestras de aprendizaje representadas por $\{p_1, t_1\}, \{p_2, t_2\}, \dots, \{p_Q, t_Q\}$ donde p son las entradas y t son las salidas deseadas, el algoritmo busca minimizar el error medio elevado al cuadrado.

$$F(x) = E[e^T e] = E[(t-a)^T (t-a)]$$

Consideramos una aproximación del error medio al cuadrado:

$$\hat{F}(x) = (t_k - a_k)^T (t_k - a_k) = e_k^T . e_k$$

Donde se ha cambiado el error total, por el error tras un número de iteraciones del algoritmo.

Y la solución de los pesos y bias se observa en las siguientes ecuaciones:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{F}(x)}{\partial w_{i,j}^m}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \hat{F}(x)}{\partial b_i^m}$$

Donde α es el factor de aprendizaje.

Resolver las derivadas parciales es la parte más complicada del método que se desarrolla a partir de la regla de la cadena usada para resolver derivadas que dependen de una tercera función:

$$\frac{\partial \ \big(n(w)\big)}{\partial} = \frac{\partial(n)}{\partial} x \frac{\partial}{\partial}$$

Utilizando esta regla tenemos las siguientes soluciones:

$$\frac{\partial \hat{F}(x)}{\partial w_{i,j}^m} = \frac{\partial \hat{F}}{\partial n_i^m} \frac{\partial n_i^m}{\partial w_{i,j}^m}$$

$$\frac{\partial \hat{F}(x)}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \frac{\partial n_i^m}{\partial b_i^m}$$

Los segundos términos de esas derivadas parciales se pueden calcular.

Considerando:

$$n_i^m = \sum_{j-1}^{S^{m-1}} w_{i,j}^m . a_j^{m-1} + b_i^m$$

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}$$

$$\frac{\partial n_i^m}{\partial b_i^m} = 1$$

Definiendo de ese modo la función de sensibilidad de la retropropagación.

$$S_i^m = \frac{\partial \hat{F}}{\partial n_i^m}$$

Entonces la solución de las ecuaciones diferenciales tenemos:

$$\frac{\partial \hat{F}(x)}{\partial w_{i,j}^m} = S_i^m . a_j^{m-1}$$

$$\frac{\partial \hat{F}(x)}{\partial b_i^m} = S_i^m$$

Por tanto aplicando el método de descenso de gradiente tendríamos lo siguiente:

$$W^m(k+1) = W^m(k) - \ \alpha . s^m (a^{m-1})^T$$

$$b^m(k+1) = b^m(k) - \ \alpha . s^m$$

Donde:

$$s^m = \frac{\partial \hat{F}}{\partial s^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial n_1^m} \\ \frac{\partial \hat{F}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial n_{S^m}^m} \end{bmatrix}$$

La solución de la matriz s^m requiere volver aplicar la regla de la cadena para resolverlo, además se resuelve recursivamente utilizando el valor del término anterior para el siguiente.

$$\frac{\partial n^{m+1}}{\partial n^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_1^{m+1}}{\partial n_{S^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \cdots & \frac{\partial n_2^{m+1}}{\partial n_{S^m}^m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_1^m} & \cdots & \frac{\partial n_{S^{m+1}}^{m+1}}{\partial n_{S^m}^m} \end{bmatrix}$$

Y para calcular los términos de esta matriz se utiliza la siguiente ecuación:

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial\left(\sum_{j=1}^{S^{m-1}} w_{i,j}^m . a_j^{m-1} + b_i^m\right)}{\partial n_j^m} = w_{i,j}^{m+1} . \frac{\partial a_j^m}{\partial n_j^m}$$

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = w_{i,j}^{m+1} . \frac{\partial f^m\left(n_j^m\right)}{\partial n_j^m} = w_{i,j}^{m+1} . \dot{f}^m\left(n_j^m\right)$$

Donde

$$\dot{f}^m\left(n_j^m\right) = \frac{\partial f^m\left(n_j^m\right)}{\partial n_j^m}$$

Por tanto el Jacobiano puede ser escrito como:

$$\dot{F}(n^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{S^m}^m) \end{bmatrix}$$

Por tanto podemos reescribir la matriz de sensibilidad de retropropagación del siguiente modo:

$$s^m = \frac{\partial \hat{F}}{\partial n^m} = \left(\frac{\partial n^{m+1}}{\partial n^m}\right)^T \frac{\partial \hat{F}}{\partial n^{m+1}} = \dot{F}(n^m)(W^{m+1})^T \frac{\partial \hat{F}}{\partial n^{m+1}}$$

$$s^m = \dot{F}(n^m)(W^{m+1})^T s^{m+1}$$

Por tanto el algoritmo toma su nombre debido a esta relación que toma los valores de atrás hacia adelante.

El último valor a calcular es el primer valor desde donde partir estos cálculos iterativos, para ello necesitamos hacer los cálculos en la capa final de la red neuronal.

$$s_i^M = \frac{\partial \hat{F}}{\partial n_i^M} = \frac{\partial (t-a)^T (t-a)}{\partial n_i^M} = \frac{\partial \sum_{j=1}^{S^{m-1}} \left(t_j - a_j\right)^2}{\partial n_i^M} = -2\left(t_j - a_j\right) \frac{\partial a_i}{\partial n_i^M}$$

Ahora considerando;

$$\frac{\partial a_i}{\partial n_i^M} = \frac{\partial n_i^M}{\partial n_i^M} = \frac{\partial f^M(n_i^M)}{\partial n_i^M} = \dot{f}^M(n_i^M)$$

$$s_i^M = -2\left(t_j - a_j\right)\dot{f}^M(n_i^M)$$

Expresado matricialmente tendremos:

$$S^M = -2\dot{F}^M(n^M)(t-a)$$

En el sistema diseñado para aprender del usuario cuenta con 2 entradas, una del sensor de presencia y el otro el sensor de iluminación, ambas entradas han sido escaladas al mismo nivel, de este modo la iluminación va desde 1 a 10, y el sensor de presencia marca 1 si no detecta ninguna persona y 10 en caso contrario.

El libro "*Introduction to Machine Learning with Python*" recomienda tener dos salidas, una para la acción de encender y otra para de apagar, sin embargo al ser entradas complementarias podría ser posible poner una sola salida, es por ello que se va probar ambas soluciones con 1 y con 2 salidas.

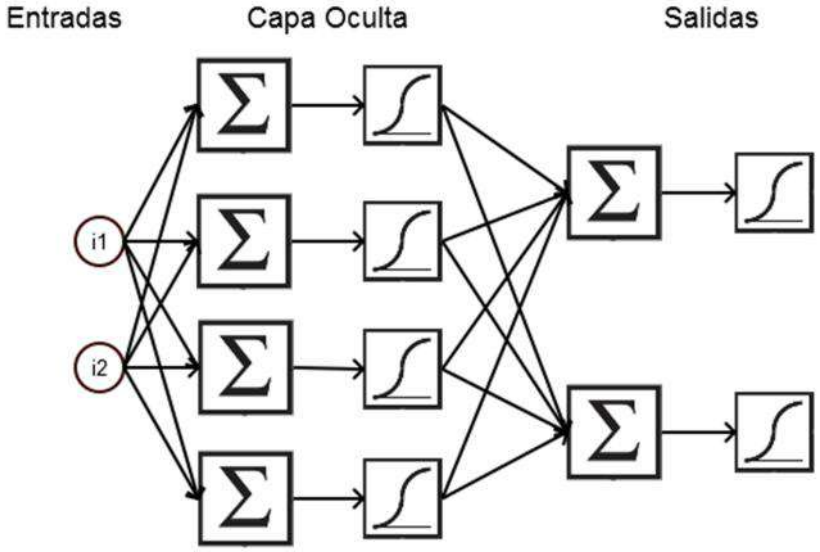


Figura 18.- Red neuronal artificial con dos salidas
Fuente: Propia (Elaborado con Pencil)

La figura 18 nos muestra la red neuronal diseñada para aprender de un usuario y decidir encender o apagar una luminaria, esta red cuenta con dos entradas denominadas i_1 e i_2 en la capa oculta se tiene cuatro neuronas, y la salida la o_1 se activa en caso de encender la luminaria y o_2 para apagar dicha luminaria.
La salida de la capa oculta:

$$a^1 = s \qquad (W^1.p + b^1)$$

La salida de la red neuronal:

$$a^2 = s \qquad (W^2.a^1 + b^2)$$

El primer paso es dar valores pequeños aleatorios a los pesos en cada capa, del mismo modo se debe dar valores pequeños a los bias.

$$w^1_{(0)} = \begin{bmatrix} 0.25 & 0.4 \\ 0.5 & 0.1 \\ 0.5 & 0.1 \\ 0.25 & 0.4 \end{bmatrix}$$

$$w^2_{(0)} = \begin{bmatrix} 0.2 & 0.35 & 0.35 & 0.2 \\ 0.3 & 0.25 & 0.25 & 0.3 \end{bmatrix}$$

$$b^1_{(0)} = \begin{bmatrix} -0.1 \\ -0.2 \\ -0.2 \\ -0.1 \end{bmatrix}$$

$$b^2_{(0)} = \begin{bmatrix} 0.2 \\ 0.6 \end{bmatrix}$$

Los siguientes valores han sido sacados del software desarrollado para obtener datos del comportamiento de un usuario, que se analiza en el capítulo 4.

Tabla 2.- Datos de entrenamiento

Sensor de Presencia	Luxómetro	Luminaria Prendida	Luminaria Apagada
1	2	0	1
1	7	0	1
10	2.5	1	0
10	7.2	0	1

Fuente: Propia

Para una primera iteración de entrenamiento vamos a tomar la primera fila de datos, que consiste que la luminaria debe estar apagado debido a que no existe una persona en el ambiente a pesar que la iluminación no es adecuada.

$$p = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$t = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$a^1 = s \left(\begin{bmatrix} 0.25 & 0.4 \\ 0.5 & 0.1 \\ 0.5 & 0.1 \\ 0.25 & 0.4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} -0.1 \\ -0.2 \\ -0.2 \\ -0.1 \end{bmatrix} \right)$$

$$a^1 = s \left(\begin{bmatrix} 0.95 \\ 0.50 \\ 0.50 \\ 0.95 \end{bmatrix} \right)$$

$$a^1 = \begin{bmatrix} 0.7211 \\ 0.6225 \\ 0.6225 \\ 0.7211 \end{bmatrix}$$

$$a^2 = s \left(\begin{bmatrix} 0.2 & 0.35 & 0.35 & 0.2 \\ 0.3 & 0.25 & 0.25 & 0.3 \end{bmatrix} \cdot \begin{bmatrix} 0.7211 \\ 0.6225 \\ 0.6225 \\ 0.7211 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.6 \end{bmatrix} \right)$$

$$a^2 = \begin{bmatrix} 0.7159 \\ 0.7931 \end{bmatrix}$$

Por tanto el error va ser:

$$e_1 = t - a = \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.7159 \\ 0.7931 \end{bmatrix} = \begin{bmatrix} -0.7159 \\ 0.2069 \end{bmatrix}$$

Calculamos las derivadas de las funciones de activación:

$$\dot{f}^m(n) = \frac{\partial \left(\frac{1}{1+e^{-n}} \right)}{\partial} = \frac{e^{-n}}{(1+e^{-n})^2} = \left(\frac{1}{1+e^{-n}} \right) \left(1 - \frac{1}{1+e^{-n}} \right)$$

$$\dot{f}^m(n) = (a^m)(1 - a^m)$$

Hallamos las matrices de sensibilidad de retropropagación definidas anteriormente:

$$S^M = -2.\dot{F}^M(n^M).(t - a)$$

$$S^2 = -2.\dot{F}^2(n^2).e_1$$

$$\dot{F}^2(n^2) = \begin{bmatrix} f^2(a_1^2) & 0 \\ 0 & f^2(a_2^2) \end{bmatrix}$$

$$f^2(a_1^2) = (1 - a_1^2)(a_1^2)$$

$$f^2(n^2) = (1 - 0.7159)(0.7159) = 0.2034$$

$$f^2(n^2) = (1 - a_2^2)(a_2^2)$$
$$f^2(n^2) = (1 - 0.7931)(0.7931) = 0.1641$$
$$S^2 = -2x \begin{bmatrix} 0.2034 & 0 \\ 0 & 0.1641 \end{bmatrix} x \begin{bmatrix} -0.7159 \\ 0.2069 \end{bmatrix}$$
$$S^2 = \begin{bmatrix} 0.2912 \\ -0.0679 \end{bmatrix}$$

$$s^m = \dot{F}^m(n^m)(W^{m+1})^T s^{m+1}$$
$$s^1 = \dot{F}^1(n^1)(W^2)^T s^2$$
$$\dot{F}^1(n^1) = \begin{bmatrix} f^1(n_1^1) & 0 & 0 & 0 \\ 0 & f^1(n_2^1) & 0 & 0 \\ 0 & 0 & f^1(n_3^1) & 0 \\ 0 & 0 & 0 & f^1(n_4^1) \end{bmatrix}$$
$$f^1(n_1^1) = (1 - a_1^1)(a_1^1)$$
$$f^1(n_1^1) = (1 - 0.7211)(0.7211) = 0.2011$$
$$f^1(n_2^1) = (1 - a_2^1)(a_2^1)$$
$$f^1(n_2^1) = (1 - 0.6225)(0.6225) = 0.2350$$
$$f^1(n_3^1) = (1 - a_3^1)(a_3^1)$$
$$f^1(n_3^1) = (1 - 0.6225)(0.6225) = 0.2350$$
$$f^1(n_4^1) = (1 - a_4^1)(a_4^1)$$
$$f^1(n_4^1) = (1 - 0.7211)(0.7211) = 0.2011$$
$$\dot{F}^1(n^1) = \begin{bmatrix} 0.2011 & 0 & 0 & 0 \\ 0 & 0.2350 & 0 & 0 \\ 0 & 0 & 0.2350 & 0 \\ 0 & 0 & 0 & 0.2011 \end{bmatrix}$$

$$s^1 = \begin{bmatrix} 0.2011 & 0 & 0 & 0 \\ 0 & 0.2350 & 0 & 0 \\ 0 & 0 & 0.2350 & 0 \\ 0 & 0 & 0 & 0.2011 \end{bmatrix} \begin{bmatrix} 0.2 & 0.35 & 0.35 & 0.2 \\ 0.3 & 0.25 & 0.25 & 0.3 \end{bmatrix}^T \begin{bmatrix} 0.2349 \\ -0.0679 \end{bmatrix}$$
$$s^1 = \begin{bmatrix} 0.0076 \\ 0.0200 \\ 0.0200 \\ 0.0076 \end{bmatrix}$$

Hallamos los valores de actualización de los pesos, usando un valor de $\alpha = 1$:

$$W^m(k+1) = W^m(k) - \alpha . s^m (a^{m-1})^T$$
$$b^m(k+1) = b^m(k) - \alpha . s^m$$
$$W^1(1) = W^1(0) - \alpha . s^1 (a^0)^T$$
$$W^1(1) = \begin{bmatrix} 0.25 & 0.4 \\ 0.5 & 0.1 \\ 0.5 & 0.1 \\ 0.25 & 0.4 \end{bmatrix} - 1. \begin{bmatrix} 0.0076 \\ 0.0200 \\ 0.0200 \\ 0.0076 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}^T$$
$$W^1(1) = \begin{bmatrix} 0.2424 & 0.3848 \\ 0.48 & 0.0601 \\ 0.48 & 0.601 \\ 0.2424 & 0.3848 \end{bmatrix}$$
$$b^1(1) = b^1(0) - \alpha . s^1$$

$$b^1(1) = \begin{bmatrix} -0.1 \\ -0.2 \\ -0.2 \\ -0.1 \end{bmatrix} - 1. \begin{bmatrix} 0.0076 \\ 0.0200 \\ 0.0200 \\ 0.0076 \end{bmatrix}$$

$$b^1(1) = \begin{bmatrix} -0.1076 \\ -0.2200 \\ -0.2200 \\ -0.1076 \end{bmatrix}$$

$$W^2(1) = W^2(0) - \alpha . s^2 (a^1)^T$$

$$W^2(1) = \begin{bmatrix} 0.2 & 0.35 & 0.35 & 0.2 \\ 0.3 & 0.25 & 0.25 & 0.3 \end{bmatrix} - 1. \begin{bmatrix} 0.2912 \\ -0.0679 \end{bmatrix} \begin{bmatrix} 0.7211 \\ 0.6225 \\ 0.6225 \\ 0.7211 \end{bmatrix}^T$$

$$W^2(1) = \begin{bmatrix} -0.0100 & 0.1687 & 0.1687 & -0.0100 \\ 0.3490 & 0.2923 & 0.2923 & 0.3490 \end{bmatrix}$$

$$b^2(1) = b^2(0) - \alpha . s^2$$

$$b^1(1) = \begin{bmatrix} 0.2 \\ 0.6 \end{bmatrix} - 1. \begin{bmatrix} 0.2912 \\ -0.0679 \end{bmatrix}$$

$$b^1(1) = \begin{bmatrix} -0.0912 \\ 0.6672 \end{bmatrix}$$

El libro de Müller & Guido, (2017) haciendo referencia a los datos de entrenamiento y a las redes neuronales explica que entre los criterios más importantes podemos considerar los siguientes:

- Cada red neuronal es diferente y se debe evaluar de acuerdo a la experiencia del diseñador.
- Los datos de entrenamiento debe considerar un número similar de las diversas salidas del sistema.
- El principal criterio para determinar si una red neuronal aprendió una regla determinada es el error mínimo cuadrado (MSE por sus siglas en inglés).

Para calcular la media del error cuadrático se usa la siguiente formula (Heaton, 2015):

$$M = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Donde:
$\hat{y}_i$: son las salidas de la red neuronal.
y_i: son las salidas deseadas.
n: número de datos de corroboración

Heaton, (2015) explica los factores que deben considerarse en este tipo de validación de las redes neuronales son los datos de la tabla 3 que presenta en su libro, de ese modo los datos que se tiene para el entrenamiento son el número de los datos de entrada para obtener el máximo error, el autor clasifica estas bases de datos en pequeña, media, grande y muy grande sin considerar valores específicos para estos valores, sin embargo hay que considerar siempre la experiencia del diseñador de redes neuronales para clasificar la base de datos con la que se entrena la red neuronal.

y_i: son las salidas deseadas.
n: número de datos de corroboración

Heaton, (2015) explica los factores que deben considerarse en este tipo de validación de las redes neuronales son los datos de la tabla 3 que presenta en su libro, de ese modo los datos que se tiene para el entrenamiento son el número de los datos de entrada para obtener el máximo error, el autor clasifica estas bases de datos en pequeña, media, grande y muy grande sin considerar valores específicos para estos valores, sin embargo hay que considerar siempre la experiencia del diseñador de redes neuronales para clasificar la base de datos con la que se entrena la red neuronal.

Tabla 3.- Valores de Error Máximo en MSE

Base de datos	Error Máximo
Pequeña	0.01
Media	0.251
Grande	1.002
Muy Grande	100.216

Fuente: Heaton, 2015.

El entrenamiento de la red neuronal ha requerido ampliar el cuadro de datos de entrada considerando más ejemplos de entradas para que el sistema encienda, siguiendo la recomendación de los autores mencionados.

Tabla 4.- Datos de entrenamiento extendido

Sensor de Presencia	Luxómetro	Luminaria Prendida	Luminaria Apagada
1	2	0	1
10	2.5	1	0
1	7	0	1
10	1.5	1	0
10	7.2	0	1
10	5	1	0

Fuente: Propia.

El entrenamiento considera factores como α que es el factor de aprendizaje, el número de salidas que de acuerdo a la recomendación de Sarah Guido se usan dos como se muestra en la figura 18, sin embargo también se usara una red neuronal similar pero con una sola salida como se muestra en la figura 19, además se va probar aumentar el número de neuronas en la capa oculta y para la validación de los datos se va usar los datos de la tabla 5.

Tabla 5.- Datos de validación

Sensor de Presencia	Luxómetro	Luminaria Prendida	Luminaria Apagada
1	3	0	1
10	3	1	0
1	10	0	1
10	1	1	0

10	8	0	1
10	5	1	0

Fuente: Propia.

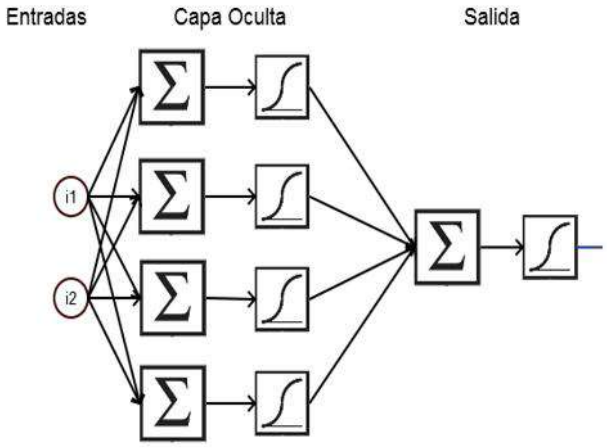
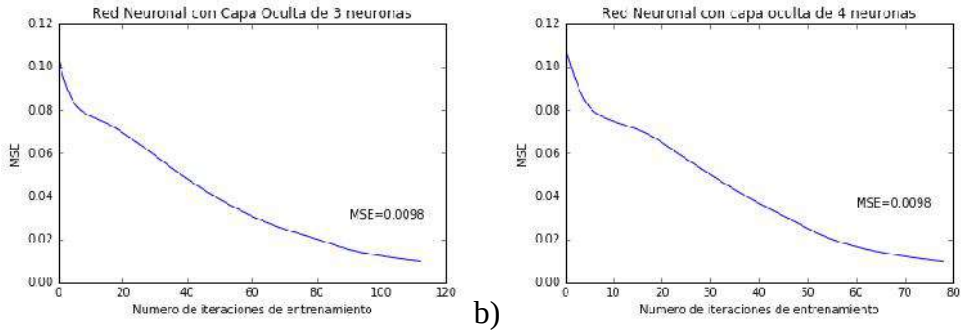


Figura 19.- Red neuronal artificial con una salida

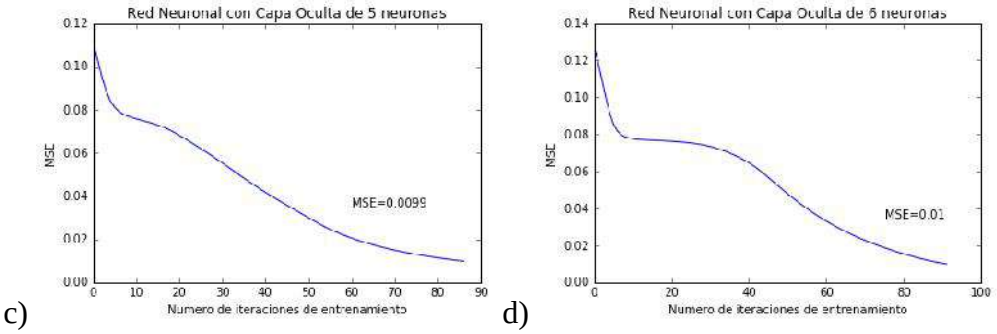
Fuente: Propia (Elaborado en Pencil)

2.5.1.7. Validación de las redes neuronales multicapa

Una vez estudiando el algoritmo de aprendizaje se ha diseñado un programa en Python para la actualización de pesos sinápticos de las redes neuronales y el valor del bias, se va analizar diversas configuraciones de redes neuronales se va evaluar primero el uso de diversos números de capas ocultas, luego evaluar el número de salidas debido a que a pesar de que la bibliografía recomienda poner dos salidas, también es válido probar con una salida y comparar los resultados por último se va evaluar el factor de aprendizaje que recomiendan valores bajos de aprendizaje, y vamos evaluar cuanto afecta su valor en la velocidad de aprendizaje de la red neuronal artificial, las figuras 20, 21, 22 muestra los resultados de la media del error cuadrático en los diversos escenarios ya comentados.



a) b)

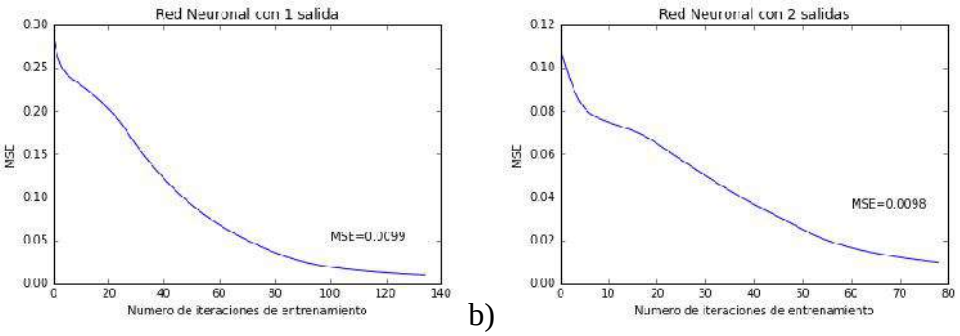


c) d)

Figura 20.- Redes neuronales con diverso número de neuronas en la capa oculta

Fuente: Propia (Elaborado en Python)

En la figura 20 se analizó el trabajo que cumple el número de capas ocultas y como se puede observar más de 4 neuronas no hay mejora respecto al error mínimo cuadrado además de que los ciclos de entrenamiento aumentan, para estas pruebas se ha usado un el valor de $\alpha = 0.1$ y dos salidas.



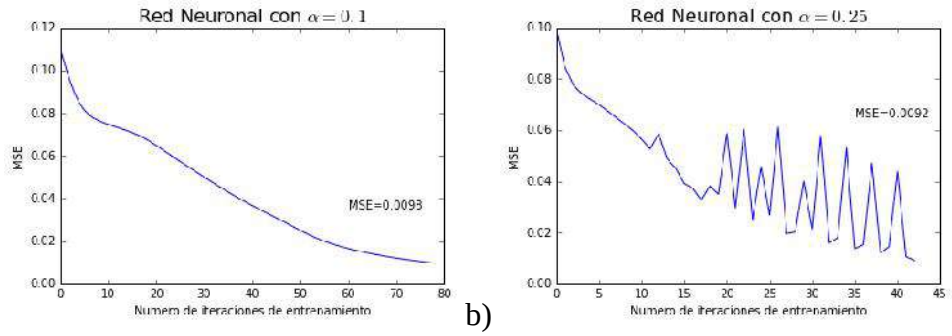
a) b)

Figura 21.- Redes neuronales una salida y con dos salidas

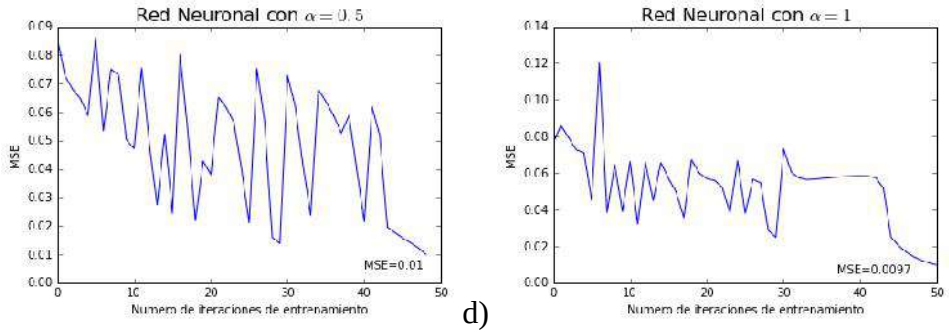
Fuente: Propia (Elaborado en Python)

En la figura 21 se muestra los resultados de la diferencia entre tener una sola salida para dos acciones y tener una salida para cada acción en este caso nosotros tenemos 2 acciones, la de prender y la de apagar, y al usar dos salidas el aprendizaje es más rápido, es por ello que en adelante se usarán dos salidas como recomienda la bibliografía.

En la figura 22 se va analizar la acción del factor de aprendizaje alfa, la bibliografía recomienda usar valores pequeños y en las pruebas precedentes se han usado este factor con un valor de $\alpha = 0.1$ y en estas pruebas se hace variar su valor hasta llegar al valor de $\alpha = 1$



a) b)



c) d)

Figura 22.- Redes neuronales con diferente valor de α

Fuente: Propia (Elaborado en Python)

Como se observa a valores mayores de α es más errático el camino hacia una convergencia aunque sea más pequeño el número de iteraciones de entrenamiento, sin embargo esa falta de regularidad en el decrecimiento continuo no hace muy confiable a un sistema con un valor de α alto.

2.5.2. Árboles de decisión

El libro *Data mining with Decision Trees* de Rokach & Maimon, (2015) en referencia a los árboles de decisión lo definen como un modelo de predicción utilizado en diversos ámbitos que van desde la inteligencia artificial hasta la Economía. Dado un conjunto de datos se fabrican diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema, estos son muy utilizados en el análisis de la toma de decisiones de individuos.

Debido a estas características se va aplicar esta teoría para compararla frente a una red neuronal multicapa.

Algunas ventajas de los árboles de decisión son;

- Fácil de entender e interpretar, los arboles pueden ser visualizados.
- Requiere poca preparación de la data aunque es necesario que no existan datos incompletos.
- El costo de uso informático del árbol es logarítmico en relación al número de puntos usados para entrenar el árbol.
- Se puede analizar data categórica junto con la data numérica.
- Es posible tener salidas categóricas y numéricas.
- Los datos son interpretables debido a que el modelo de árbol de decisión es una caja blanca.

Las desventajas de un árbol de decisión incluyen:

- El mal uso de la complejidad puesta en un árbol de decisión puede quitarle generalización, para evitar este error es necesario considerar establecer la profundidad máxima del árbol o la poda al considerar un número mínimo de datos para determinada decisión.
- Los árboles de decisión pueden ser inestables porque pequeñas variaciones en los datos pueden dar lugar a que se genere un árbol completamente diferente. Este problema se mitiga utilizando árboles de decisión dentro de un conjunto.
- La optimización del aprendizaje de un árbol de decisión conduce a complejos algoritmos que no garantizan un óptimo global.

- Existen formas lógicas que son difíciles de aprendes como el concepto de XOR o de paridad debido a que no es fácil poner estos conceptos en si como un árbol de decisión.
- Es necesario tener la data equilibrada.

2.5.2.1. Aprendizaje en Árboles de decisión

Los intentos por crear algoritmos que generen árboles de decisión en base a ejemplos dados comienza en las décadas de 1970, y en Quinlan, (1986) presenta un paper sobre la técnica ID3, para la elaboración de árboles de decisión considerando los ejemplos de aprendizaje, y a lo largo de los años diversos artículos científicos se han dedicado analizar la preparación de los datos, el cual podemos resumir en los siguiente:

- La data debe estar equilibrada.
- No es necesaria data en el mismo rango.

Con estas recomendaciones podemos rehacer nuestra tabla de datos de entrenamiento para el árbol de decisión.

Tabla 6.- Datos de entrenamiento para el árbol de decisión

Sensor de Presencia	Luxómetro	Estado de la Luminaria
0	200	Apagado
1	250	Prendido
0	700	Apagado
1	150	Prendido
1	720	Apagado
1	500	Prendido

Fuente: Propia.

El primer concepto necesario en el análisis de datos es la entropía que en informática se define como una medida de información, y realiza una suma ponderada de la información transmitida por cada clase por la porción de los elementos de la misma, en esta medida se refleja la variabilidad existente, o el grado de desorden, en un conjunto de ejemplos presentados frente a las diferentes particiones o clases posibles.

$$E \quad \text{ía}(E, A_i, v_j) = -\sum_{k=1}^{C} \frac{n_i}{n_i} l\text{o}_2\left(\frac{n_i}{n_i}\right)$$

Donde:

E ía(E, A_i, v_j): es la entropía de los ejemplos E cuando el atributo A_i tiene un valor v_j

n_i : es el número de ejemplos que tienen el valor v_j del atributo A_i y pertenecen a la clase C_k

n_i : es el número de ejemplos que tienen el valor v_j del atributo A_i

v_j: es el valor del atributo

C: es el número de clases.

A lo largo de la elaboración del árbol de decisión se va eligiendo en cada nodo de decisión el atributo que mayor ganancia de información aporte, y la métrica que se usa en esta comparación es la entropía ya definida como ganancia de información.

$$G \quad (E, A_i) = En \quad \text{ía}(E) - \sum_{v_j \in V} \frac{n_i}{n} E \quad \text{ía}(E, A_i, v_j)$$

Para obtener la máxima ganancia y considerando que el primer factor de la entropía del conjunto es la misma, se va buscar hallar el menor del segundo factor como mejor atributo.

Por tanto con los datos presentados en la tabla 6 se va proceder a elaborar la métrica para los dos atributos.

$$I(A_1) = \sum_{i=1}^{n\ (A_1)} \frac{n_i}{n} Ii \ = \sum_{i=1}^{2} \frac{n_i}{6} Ii \ = \frac{n_1}{6} I_1 \ + \frac{n_1}{6} I_1$$

$$I_1 \ = -\sum_{j=1}^{2} \frac{n_j}{n_k} lo \ _2\left(\frac{n_j}{n_k}\right) = -\frac{2}{2} lo \ _2\left(\frac{2}{2}\right) - \frac{0}{2} lo \ _2\left(\frac{0}{2}\right) = 0$$

$$I_1 \ = -\sum_{j-1}^{2} \frac{n_j}{n_k} lo \ _2\left(\frac{n_j}{n_k}\right) = -\frac{1}{4} lo \ _2\left(\frac{1}{4}\right) - \frac{3}{4} lo \ _2\left(\frac{3}{4}\right) = 0.81$$

$$I(A_1) = \frac{2}{6}0 + \frac{4}{6}0.81 = 0.54$$

$$I(A_2) = 0$$

Este valor de la ganancia de información debido al atributo dos, es cero porque toda la información de esta clase esta dispersada porque no se ha tomado ninguna métrica antes de tratar la data, es por ello que primero antes de considerar este atributo se va sacar la media de los datos y actualizar el valor convirtiendo estos datos en dos clases posibles, convirtiendo los datos de la tabla 6 en la tabla 7.

Tabla 7.- Datos de entrenamiento para el árbol de decisión modificado

Sensor de Presencia	Luxómetro	Estado de la Luminaria
0	0	Apagado
1	0	Prendido
0	1	Apagado
1	0	Prendido
1	1	Apagado
1	1	Prendido

Fuente: Propia.

Desde esta nueva tabla se ha vuelto a calcular la métrica para el atributo 2.

$$I(A_2) = 0.92$$

Considerando ambas métricas se puede realizar el primer nodo de decisión:

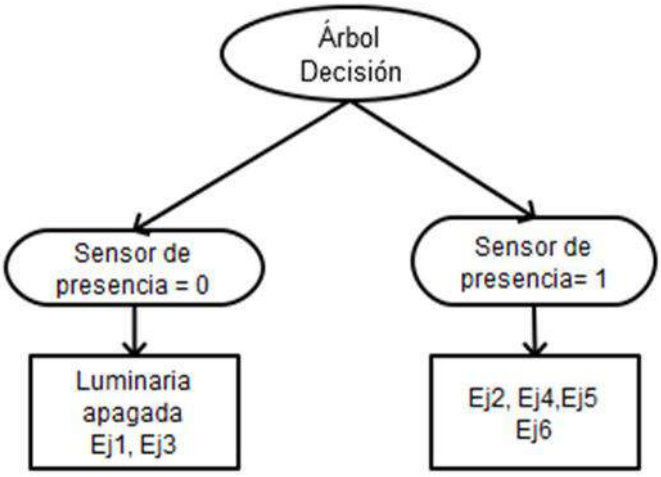


Figura 23.- Primer paso por el algoritmo de aprendizaje

Fuente: Propia (Elaborado en Pencil)

Ahora se crea una tabla con los datos sobrantes del primer paso .y se evalúa nuevamente la entropía de los datos restantes para volver a encontrar un nuevo nodo de clasificación.

Tabla 8.- Datos de entrenamiento tras el primer paso de elaboración

Sensor de Presencia	Luxómetro	Estado de la Luminaria
1	0	Prendido
1	0	Prendido
1	1	Apagado
1	1	Prendido

Fuente: Propia.

Haciendo los cálculos para los atributos se consiguen los siguientes valores:

$$I(A_1) = 0.81$$

$$I(A_2) = 0.5$$

Por tanto con el nuevo corte se hace de acuerdo al atributo A_2

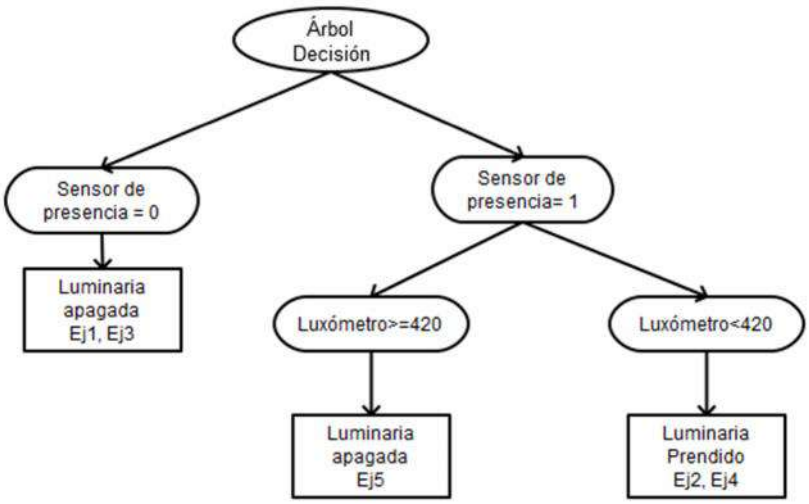


Figura 24.- Segundo paso por el algoritmo de aprendizaje

Fuente: Propia (Elaborado en Pencil)

Tras un segundo paso por el algoritmo el ejemplo 6 ha quedado sin clasificación sin embargo debido a que está dentro de un error normal, se puede dejar el árbol de decisión tal como ha quedado.

2.5.2.2. Validación del árbol de decisión

Existen diversos métodos de validación un árbol de decisión entre ellas la combinación de la sensibilidad y la especificidad dan como resultado a la presión, que es un valor que nos permite determinar lo bien entrenado que está el árbol de decisión.

$$S = \frac{p \quad v}{p}$$

$$E = \frac{n \quad ve}{n}$$

Donde la sensibilidad evalúa que tan bien el clasificador reconoce los ejemplos positivos, y la especificidad evalúa el rendimiento en los ejemplos negativos, con estos dos valores se puede evaluar la precisión de un árbol de decisión:

$$P \quad \acute{o}n = S \quad . \frac{p}{p \quad + n}$$

$$+ E \quad . \frac{n}{p \quad + n}$$

Considerando estas fórmulas podemos determinar la precisión con los datos de prueba de la tabla 9.

Tabla 9.- Datos de validación del árbol de decisión

Sensor de Presencia	Luxómetro	Estado de la Luminaria
0	400	Apagado
1	180	Prendido
0	700	Apagado
1	260	Prendido
1	600	Apagada
1	500	Prendida

Fuente: Propia.

$$S = \frac{2}{3} = 0.666$$

$$E = \frac{3}{3} = 1$$

$$P \quad is \ \acute{o}n = 0.666.\frac{3}{6} + 1.\frac{3}{6} = 0.83$$

Teniendo una precisión para este árbol de decisión de 0.83 que considerando el número de datos está dentro de un margen de precisión aceptable.

2.6. Ordenadores de placa reducida

Un ordenador de placa reducida, conocida también como SBC (por sus siglas en ingles *Single Board Computer*) es una computadora completa en un solo circuito, el diseño se centra en un solo microprocesador con RAM, entradas y salidas y todas las demás

características de un computador funcional en una tarjeta que suele ser de tamaño reducido y que tiene todo lo que necesita en la placa base.

Las aplicaciones de estas SBC van desde trabajos en entornos industriales o en sistemas embebidos como centro de comandos de un sistema mayor, hasta proyectos de adquisición de datos remotos y domótica.

Entre los principales modelos de SBC se tiene el Raspberry Pi, el Gumstix, Intel Galileo y el Odroid.

2.6.1. Raspberry Pi

El Raspberry Pi es un SBC de origen británico que nace a raíz de crear una computadora de bajo coste para estimular la enseñanza de las ciencias de la computación en las escuelas, hoy en día es el SBC más usado por aficionados a desarrollar prototipos propios y laboratorios, las principales características del Raspberry Pi 2 son:

- 4 puertos USB, 40 pines de entrada y salida general.
- Puerto de HDMI
- Puerto Ethernet
- Base de Micro SD
- Núcleo de Video 3D, 1 Gb de RAM
- CPU ARM Cortex-A7 de 900 MHz
- Precio: 32 dólares

La figura 25 muestra el modelo Raspberry Pi 2.

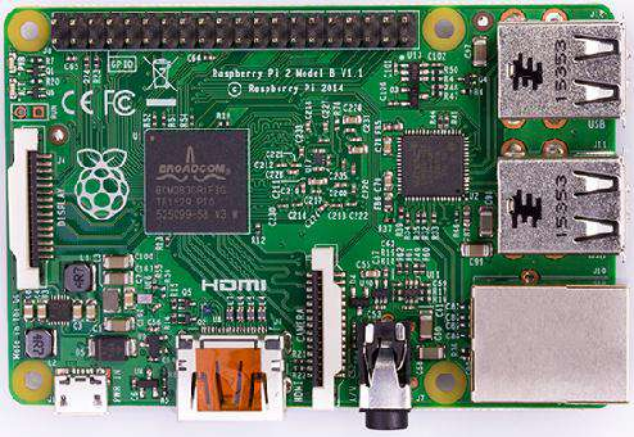


Figura 25.- Raspberry Pi 2
Fuente: Raspberry Pi, 2014

2.6.2. Odroid C2

El Odroid es una SBC de origen surcoreano creado por la empresa Hardkernel, cuyo nombre viene de Open Android, y en la búsqueda de una plataforma de desarrollo para aplicaciones Android se creó este sistema totalmente compatible con el sistema operativo Linux y Android de ese modo esta pequeña SBC tiene características superiores de procesador y memoria.

El Odroid-C2 incorpora muchas de las conexiones que presentan los típicos ordenadores, 4 puertos USB, un puerto OTG microUSB, un puerto Ethernet que soporta velocidades de transferencia Gigabit, un conector HDMI 2.0 para monitores

que puedan soportar resolución de hasta 4k, un conector de alimentación de 5V/2A. Además de estas entradas convencionales, el C2 también incluye un puerto GPIO de 40 pines, un puerto de consola serie USB-UART, un conector para módulos eMMC y una ranura para tarjetas microSD, el precio de un SBC de esta compañía son de 40 dólares.

Figura 26.- Odroid C2
Fuente: Hardkernel, 2015

2.6.3. Comparación entre Raspberry Pi y Odroid

El concepto de Odroid nace como un clon de Raspberry Pi y en cada versión van eclipsando el original en rendimiento (Ruppi, 2014).
En comparación con el Raspberry Pi (RPI), el C1 tiene 4 veces el número de núcleos CPU y la frecuencia de reloj es aproximadamente 2 veces más rápido. Además, el tamaño de la RAM también es 2 veces mayor y el acceso a la RAM es dos veces más rápido. El C1 también incluye un puerto Ethernet Gigabit que permite altas velocidades de transmisión de datos, alrededor de 500 Mbps en el mundo real. El C1 tiene 4 puertos USB host así como un puerto USB-OTG para su rápida conexión con gadgets Linux
Además los creadores de Odroid han incluido lectores eMMC, que son memorias de almacenamiento ultra rápido que en un momento llegaron a ser superadas por las memorias SD pero en la actualidad se usan de modo análogo a los discos solidos de las computadoras de escritorio.
La figura 27 nos da una clara idea de las diferencias de procesamiento entre el RPI y el Odroid C1, es fácil diferenciar las velocidades de acceso a memoria, de acceso a datos de la red, velocidad de copiado de archivos y esto gracias a la integración de memorias eMMC para acelerar en general el procesamiento de datos, sin embargo con el fin de demostrar que en condiciones similares en este caso corriendo el sistema operativo desde una memoria SD se realizó otro ensayo donde ambos SBC tenían una memoria SD para correr el sistema operativo, la gráfica de esta prueba se puede observar en el figura 28.

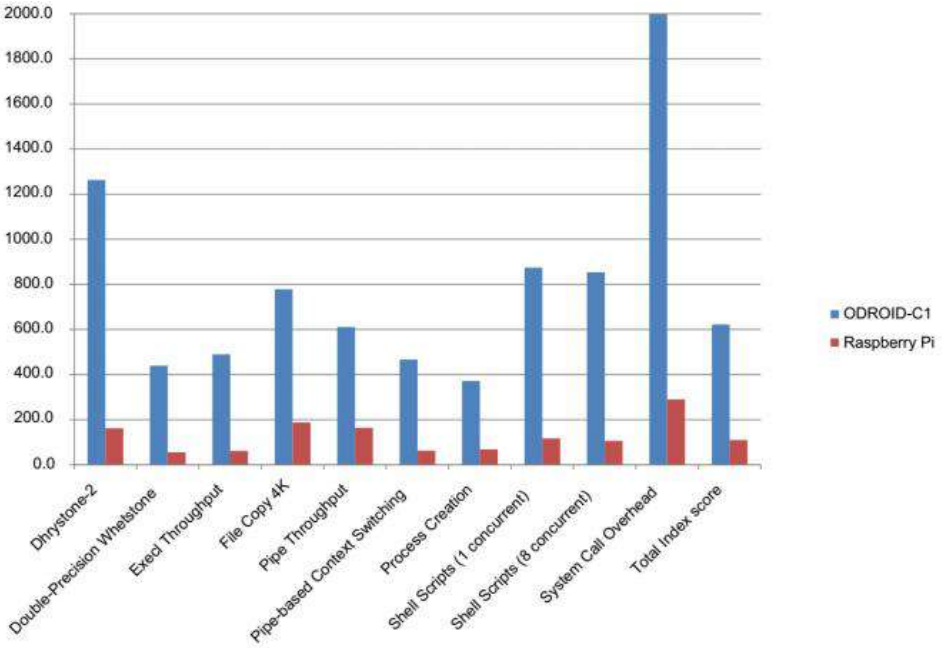


Figura 27.- Comparación de velocidad de procesamiento
Fuente: Ruppi, 2014

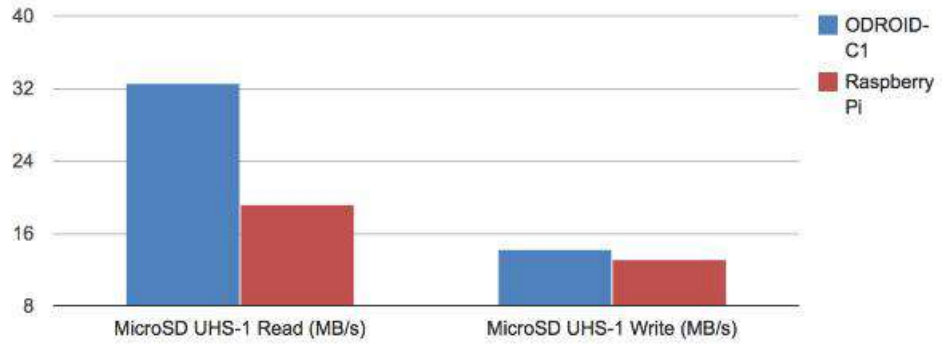


Figura 28.- Comparación de velocidad de acceso a la data
Fuente: Ruppi, 2014

Por las comparaciones obvias el Odroid es una plataforma muy completa para el desarrollo de prototipos y sus aplicaciones en la domótica, esta comparación realizada por Hardkernel demuestra clara ventaja del Odroid sobre el Raspberry, si bien la última versión del Raspberry trae mejores características aún no se equipara con una versión C2 del Odroid.

Capítulo 3
Diseño y control de iluminación

3.1. Introducción

El presente capitulo presenta la teoría básica de todo controlador, y su aplicación a la iluminación.

La iluminación elegida para el desarrollo de la tesis es la tecnología LED y los controladores para los ledes son diversos entre ellos los de corte de fase (Yan & Chen, 2014) y también los esquemas de modulación por ancho de pulso (Vasca & Iannelli, 2012) o PWM por sus siglas en inglés, sin embargo la conclusión final de diversos investigadores en el área tales como Mahadeokar & Sardeshmukh, (2015) sugieren que la modulación por ancho de pulso es la opción más eficiente, barata y simple; es por ello que en la presente tesis se va analizar y usar este tipo de controlador.

La tesis tiene como objetivo entregar una luminaria que se regule bien en cualquier ambiente, es por ello que no se puede diseñar un controlador estándar, sino más bien un controlador adaptativo, en el lado del usuario la tarea de la adaptación es aprendida por los sistemas de aprendizaje automático, y en el lado del control de la iluminación se va implantar un controlador proporcional, integral y derivativo discreto con auto sintonización o adaptativo.

3.2. Sistemas automáticos de control

El control automático está relacionado con el mantenimiento de sus variables a un valor deseado de operación (Ipanaqué, 2012) en el caso de la presente tesis la variable a controlar es el nivel de iluminación medida en luxes, entre las principales ventajas que menciona el autor del libro "Control Automático de Procesos: innovando procesos productivos" están: reducción de costos, aumento de la calidad y reducción de errores, ahorro de tiempo y mejor administración de recursos.

Para los procesos discretos normalmente los pasos del sistema de automatización son secuenciales, y dichos pasos son ejecutados por un microcontrolador o una computadora, éstos son repetitivos y cíclicos es decir de tipo secuencial, por eventos o por tiempo.

3.2.1. Control a lazo abierto

También llamado regulación en adelanto o *feedforward* cuya estrategia es alcanzar el nivel deseado con un solo comando, en otras palabras en un sistema de control de

lazo abierto no se mide la salida ni se realimenta para compararla con la entrada (Ogata, 2010).

Un sistema de lazo abierto se caracteriza porque el valor deseado genera una señal de comando hacia el proceso y como salida de éste se obtiene un valor aproximado al deseado. Puede resultar ineficaz por las posibles desviaciones ocasionadas por disturbios. Para poder realizar un control adecuado a un proceso, es necesario medir la salida del sistema y considerando este valor tomar la decisión más adecuada para que pueda corregirla llevándola hacia el valor esperado (Ipanaqué, 2012).

3.2.2. Control a lazo cerrado

Este sistema también llamado realimentado o *feedback* realiza la medición de la variable de la salida, la cual es retroalimentada al regulador o controlador, en base a una estrategia, el controlador busca que la salida corresponda al valor deseado. La señal de control de un sistema de lazo abierto, permanece constante mientas no se varíe la entrada, en cambio, en un sistema de lazo cerrado la señal de control puede variar debido al cambio del valor deseado o debido a una desviación de la salida. (Ipanaqué, 2012).

En un sistema de control de lazo cerrado se alimenta al controlador la señal del error de actuación, que es la diferencia entre la señal de entrada y la señal de realimentación (que puede ser la propia señal de salida o una función de la señal de salida y sus derivadas y/o integrales), con el fin de reducir el error y llevar la salida del sistema a un valor deseado. (Ogata, 2010)

3.3. Control Proporcional, integral y derivativo

En el área del control una parte fundamental lo sigue ocupando los controladores PID, estos controladores presentes en prácticamente todas las industrias son de diseño general que se pueden aplicar a casi cualquier sistema de control y ajustar en el sitio de aplicación, las cuales ya se tienen una amplia forma de sintonización siguiendo unas reglas determinadas, por ejemplo las reglas de Ziegler-Nichols. La figura 29 muestra el esquema básico de un PID.

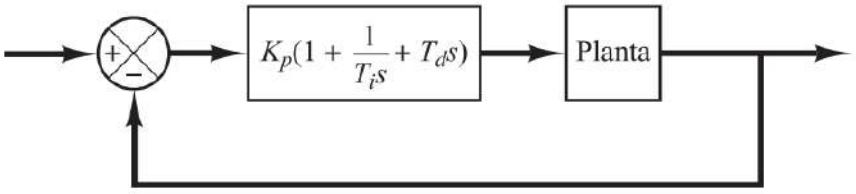


Figura 29.- Control PID de una planta

Fuente: Ogata, 2010

Las ecuaciones en tiempo continuo que describen a un control PID son las siguientes:

$$u_{(t)} = K_p \left(e_{(t)} + \frac{1}{T_i} \int_0^t e_{(t)} dt + T_d \frac{de_{(t)}}{dt} \right) +$$

Donde:

$u_{(t)}$: señal de control.

$e_{(t)}$: error de control (salida deseada - salida)

K_p: ganancia proporcional

T_i: tiempo integral
T_d: tiempo derivativo
La señal de control es por tanto la suma de los tres términos y cada uno de ellos conlleva una acción que fue estudiado por Åström, Hägglund, Dormido, & Guzmán Sánchez, (2009).

- Acción proporcional, donde la ley de control solo es proporcional a la acción de control además si se agrega un u_b que será una señal de base que podría fijarse en $\frac{(u_{max}-u_{\min})}{2}$ se tendría la siguiente ecuación para la acción proporcional.

$$u_{(t)} = K_p.e_{(t)} + u_b$$

Este tipo de controlador tiene el defecto de llevar a un error estático que se puede arreglar variando el u_b que será el valor calculado para llegar a valor deseado de la salida, sin embargo no va mantener un error estático cero en presencia de disturbios.

- Acción integral, cuya función principal de la acción integral es asegurarse de que la salida del proceso coincida con el punto de consigna en estado estacionario, un controlador con acción integral siempre dará error cero en estado estacionario, en la figura 30 se observa un diagrama de bloques de un controlador PI

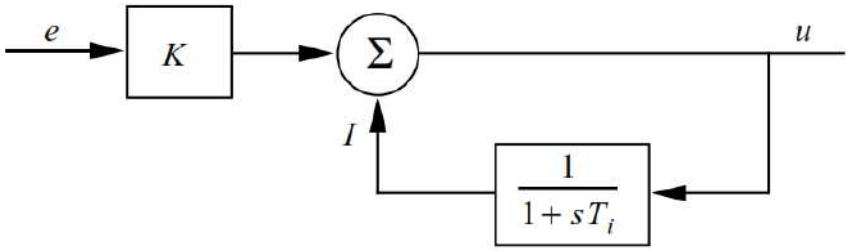


Figura 30.- Control PI de una planta
Fuente: Åström et al., 2009

De acuerdo con la figura 30 tendremos la siguiente ecuación:

$$u = Ke + I = T_i\frac{dI}{dt} + I$$

Por lo tanto tentemos:

$$T_i\frac{dI}{dt} = Ke$$

$$dI = \frac{K}{T_i}e.dt$$

Integrando tenemos:

$$I = \frac{K}{T_i}\int e.dt$$

De donde tenemos nuestro modelo PI:

$$u = K\left(e + \frac{1}{T_i}\int e.dt\right)$$

- Acción derivativa, el objetivo de esta acción es mejorar la estabilidad en lazo cerrado; la desventaja típica de los tipos de controladores no predictivos es que el error tiene que introducirse en el sistema para que el controlador actué, en el control PID se busca mitigar este error con la acción derivativa, que se puede

interpretar como si el control se hiciese proporcional a la salida predicha del proceso, donde la predicción se hace extrapolando el error por la tangente de la curva del error como se muestra en la figura 31.

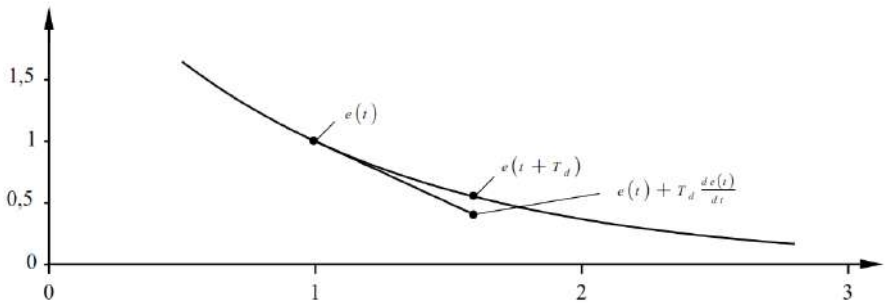


Figura 31.- Interpretación de la acción derivativa como control predictivo
Fuente: Åström et al., 2009

La estructura básica de un controlador PD es:

$$u_{(t)} = K\left(e_{(t)} + T_d \frac{de_{(t)}}{dt}\right)$$

3.3.1. Modificaciones de los esquemas de control PID

El extenso uso de los controladores PID ha reconocido también diversos problemas que existen tal como la "patada en el punto de consigna" debido al uso de una modificación necesario en un PID real donde el término derivativo se representa:

$$\frac{T_d s}{1 + \gamma T_d s}$$

Donde el parámetro γ toma valores alrededor de 0.1 (Ogata, 2010) por tanto cuando la entrada de referencia es una función escalón la variable manipulada no contendrá una función impulso sino una forma de un pulso estrecho y ese es el problema o fenómeno antes mencionado.

La primera solución que se ha presentado para evitar ese fenómeno es el control PI-D, cuyo diagrama de bloques se muestra en la figura 32.

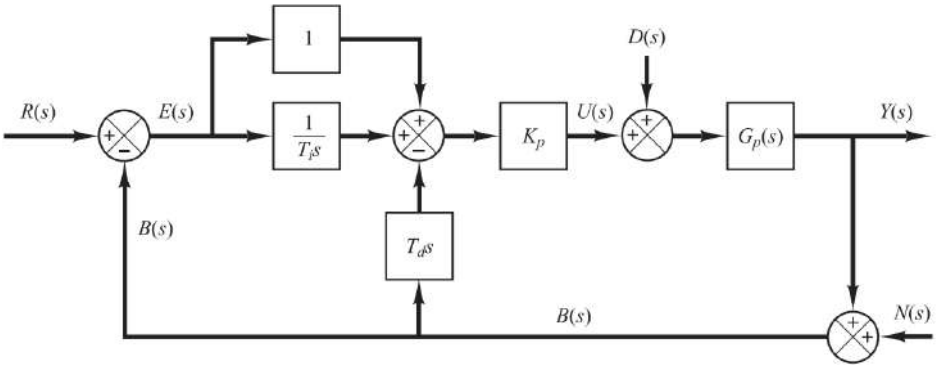


Figura 32.- Sistema de control PI-D
Fuente: Ogata, 2010

La función de transferencia en lazo cerrado del sistema de control PI-D se obtiene mediante:

$$\frac{Y_{(s)}}{R_{(s)}} = \left(1 + \frac{1}{T_i s}\right).\left(\frac{K_p G_p(s)}{1 + \left(1 + \frac{1}{T_i s} + T_d s\right) K_p G_p(s)}\right)$$

Otra posible configuración de un controlador PID es I-PD considerando otra vez el caso en el que la entrada de referencia es una función escalón. Tanto el control PID como el control PI-D implican una función escalón en la señal manipulada. En muchas ocasiones, tal cambio escalón en la señal manipulada puede no resultar conveniente. Por tanto, puede convenir mover la acción proporcional y la acción derivativa al camino de realimentación, a fin de que estas acciones sólo afecten a la señal de realimentación (Ogata, 2010), la figura 33 muestra tal esquema de control, que se denomina control I-PD.

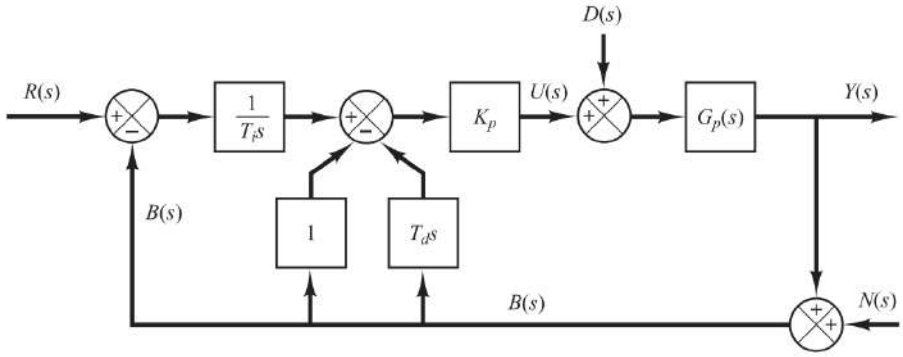


Figura 33.- Sistema de control I-PD
Fuente: Ogata, 2010

La señal manipulada está dada por:

$$U_{(s)} = \left(K_p \frac{1}{T_i s}.R_{(s)}\right) - K_p\left(1 + \frac{1}{T_i s} + T_d s\right) B_{(s)}$$

Es necesario tener en cuenta que el valor de la referencia solo aparece en la parte del control integral, por tanto en el control I-PD es imprescindible tener la acción del control integral para una operación del sistema de control.

3.3.2. Esquemas anti-windup

Cuando un sistema de control funciona en un amplio rango de condiciones operativas puede suceder que el valor de salida que da el regulador alcance los límites del actuador. En ese momento se rompe el lazo de realimentación ya que el actuador permanece en su valor límite independientemente del valor de la variable controlada. En estas condiciones si se está utilizando un regulador que posee acción integral, el error continúa integrándose, esto significa que el término integral aumenta aun cuando no debería hacerlo. La consecuencia de este fenómeno es que cualquier regulador que posea acción integral y no tenga mecanismos para contrarrestar este efecto puede dar lugar a grandes transitorios cuando se verifica saturación en el actuador. (Ipanaqué, 2012).

Conocido este fenómeno se ha desarrollado diversos esquemas para contrarrestar el efecto windup, el principal mencionado por muchos autores es el de Åström et al., (2009) que se observa en la figura 34.

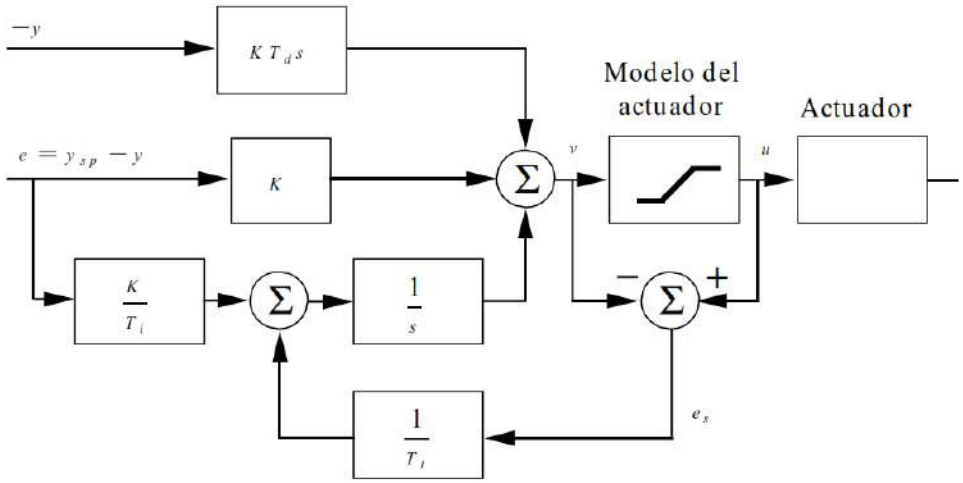


Figura 34.- Mecanismo anti-windup
Fuente: Åström et al., 2009

El esquema conocido como un recalculo que trabaja cuando la salida se satura, recalcula el término integral en el controlador de forma que su nuevo valor da una salida en el límite de la saturación, se ha estudiado también que es beneficioso no resetear el integrador de inmediato sino a través de una ganancia T_t.

3.3.3. PID discreto

Para la implementación de un controlador PID en un sistema embebido es necesario tener dicho controlador discretizado en este caso el controlador PID se discretiza según (Bobál et al., 2006) de una forma simple si consideramos T_0 un tiempo de muestreo suficientemente pequeño, la derivada se puede reemplazar con una diferencia de primer orden:

$$\frac{de}{dt} \approx \frac{e_{(k)} - e_{(k-1)}}{T_0} = \frac{\Delta e_{(k)}}{T_0}$$

Del mismo modo la integral se puede discretizar en la siguiente ecuación:

$$\int_0^t e_{(t)} dt \approx T_0 \sum_{i=1}^{k} e_{(i-1)}$$

Por tanto

$$u_{(k)} = K_p \left(e_{(k)} + \frac{T_0}{T_i} \sum_{i=1}^{k} e_{(i-1)} + \frac{T_d}{T_0} \left(e_{(k)} - e_{(k-1)} \right) \right)$$

También es posible discretizar recursivamente usando el método rectangular hacia atrás teniendo el PID discreto de la siguiente manera:

$$u_{(k)} = K_p \left(e_{(k)} + \frac{T_0}{T_i} \sum_{i=1}^{k} e_{(i)} + \frac{T_d}{T_0} \left(e_{(k)} - e_{(k-1)} \right) \right)$$

Si se requiere un alto grado de exactitud se puede discretizar la integral con el método trapezoidal:

$$\int_0^t e_{(t)} dt \approx T_0 \sum_{i=1}^{k} \frac{e_{(i)} + e_{(i-1)}}{2}$$

Por tanto el PID tendrá la siguiente forma:

$$u_{(k)} = K_p\left(e_{(k)} + \frac{T_0}{T_i}\sum_{i=1}^{k}\frac{e_{(i)} + e_{(i-1)}}{2} + \frac{T_d}{T_0}\left(e_{(k)} - e_{(k-1)}\right)\right)$$

Si el tiempo de muestro es suficientemente rápido la diferencia entre las aproximaciones de la integral no es significativa, siendo la más usada el método rectangular hacia atrás (Bobál et al., 2006).

Estas presentaciones discretizadas el PID requieren que el microprocesador este almacenando toda la data del error a lo largo del proceso, y muchas veces eso sobrepasa la capacidad de memoria, es por ello que una función recursiva de la forma discretizada del PID conocida como algoritmo incremental se presenta del siguiente modo:

$$u_{(k)} = \Delta u_{(k)} + u_{(k-1)}$$

$$\Delta u_{(k)} = K_p\left(e_{(k)} - e_{(k-1)} + \frac{T_0}{T_i}e_{(k)} + \frac{T_d}{T_0}\left(e_{(k)} - 2e_{(k-1)} + e_{(k-2)}\right)\right)$$

Con esta versión solo se requiere guardar los dos últimos errores y la última entrada del sistema, y hace mucho más sencillo su implementación en sistemas embebidos.

3.4. PID adaptativo

La teoría de control en su búsqueda de un controlador universal ha desarrollado diversos modelos de identificación en tiempo real y su aplicación al diseño de controladores como los vistos en *"Digital Self-tuning Controllers: Algorithms, Implementation and Applications"*, (2006), así también se han desarrollado diversos controladores que sin identificar el sistema pueden mantenerlo en un punto de control como el visto en el paper *"Self-tuning of PID controllers by adaptive interaction"*, (2000) valiéndose de métodos como el descenso de gradiente mantienen la tendencia del error en 0.

La técnica conocida como descenso de gradiente se basa en encontrar la gradiente del error, normalmente el error cuadrático, y seguirla de forma descendente, este método nos asegura una rápida convergencia al buscar un valor mínimo, si la gradiente de una función es descendiente entonces en ese punto se encuentra un valor mínimo y si en cambio es ascendente es decir que el valor mínimo ya se ha pasado.

3.4.1. Método recursivo de mínimos cuadrados

El método recursivo de mínimos cuadrados conocido como RLS (por sus siglas en inglés *Recursive Least Squares*) es un método usado para la identificación de sistemas tanto fuera de línea o en tiempo real, la teoría desarrollado por Robin De Keyser, (2012) permite una identificación en tiempo real como es la necesaria en un control PID de auto sintonización o adaptativo.

Supongamos una relación entre dos series de tiempo: $y(.)$, $u(.)$ ambas relacionadas por una ganancia K teniendo en cuenta que al no tener elementos dinámicos el proceso transforma la medida de u a una salida y.

Este método al ser usado en tiempo discreto tendremos una secuencia de medidas como por ejemplo: $\{u(1), y(1), u(2), y(2), \dots, \dots, u(i), y(i)\}$ por lo tanto en el caso ideal tendremos:

$$y(.) = Ku(.)$$

Donde K es la ganancia desconocida que queremos estimar, es necesario considerar un disturbio presente en la medición de la señal o un error en el modelo u otro; por tanto es necesario considerar un término de disturbio estocástico $v(.)$ por lo que el modelo quedara representado del siguiente modo:

$$y(.) = Ku(.) + v(.)$$

Debido al último factor no es posible considerar que $K = {y(.)}/{u(.)}$ por tanto deberíamos tener una secuencia de entradas y salidas como se muestra en la siguiente lista.

$$\begin{cases} y(1) = Ku(1) + v(1) \\ y(2) = Ku(2) + v(2) \\ \quad . \\ \quad . \\ \quad . \\ y(i) = Ku(i) + v(i) \\ \quad . \\ \quad . \\ \quad . \\ y(t) = Ku(t) + v(t) \end{cases}$$

Ahora estos datos se suman y obtenemos lo siguiente:

$$\sum_{i=1}^{t} y(i) = K \sum_{i=1}^{t} u(i) + \sum_{i=1}^{t} v(i)$$

Considerando que el disturbio modelado en $v(t)$ tiene una media de cero, por lo tanto ahora si podemos definir la ganancia K:

$$\widehat{K}_t = \frac{\sum_{i=1}^{t} y(t)}{\sum_{i=1}^{t} u(t)}$$

En conclusión si tenemos la información de entradas y salidas suficientemente grande ya podríamos identificar el sistema de este primer modo, sin embargo también podemos recurrir a pequeños artilugios para ordenar la ecuación de forma recursiva:

$$\widehat{K}_t = \widehat{K}_{t-1} + \frac{1}{\alpha_t}.\left[y(t) - \widehat{K}_{t-1}u(t)\right]$$

Donde:

$$\alpha_t = \alpha_{t-1} + u(t)$$

Esta formulación recursiva es la base de todos los algoritmos de identificación en tiempo real, comenzando con un $\widehat{K}_0$ y α_0 se va estimando los nuevos valores.

La formulación de un sistema dinámico de entrada $u(t)$ y salida $y(t)$ medidos en un tiempo discreto $t = 1, 2, 3, \ldots$ pueden ser formulados del siguiente modo:

$$y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) + v(t)$$

Donde podemos reconocer a $v(t)$ como un disturbio de carácter desconocido, por conveniencia se va usar el operador de retraso:

$$q^{-1} y(t) = y(t-1)$$

Por tanto reescribimos como:

$$A(q^{-1})y(t) = B(q^{-1})u(t) + v(t)$$

Donde $A(q^{-1})$ y $B(q^{-1})$ son polinomios con el operador de retraso definido como:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}$$
$$B(q^{-1}) = b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}$$

Con lo que podemos definir el vector de parámetros:

$$\theta^T = \left[a_1 \dots a_{n_a}; b_1 \dots b_{n_b}\right]$$

Y como complemento de este vector de parámetros tenemos el vector de la información de entradas y salidas retrasadas:

$$\varphi^T(t) = [-y(t-1) \dots - y(t-n_a); u(t-1) \dots - u(t-n_b)]$$

Y la salida se reescribe:

$$y(t) = \varphi^T(t)\theta + v(t)$$

Este modelo describe una variable observada $y(t)$ como una combinación lineal desconocida de los componentes del vector de observación φ^T más el disturbio que podría pertenecer al ruido; en control $\varphi^T(t)$ es llamado el vector de medidas, y θ el vector de parámetros.

En base a esta última representación el método recursivo de mínimos cuadrados requiere dos vectores tales son:

$$\hat{\theta}(t) = f\left[\hat{\theta}(t-1), P(t), \varphi(t)\right]$$
$$P(t) = g\left[P(t-1), \hat{\theta}(t-1), \varphi(t)\right]$$

Donde los factores que definen las funciones $f(.,.,.)$ y $g(.,.,.)$ son factores previamente conocidos, la estimación anterior $\hat{\theta}(t)$, la información actual $\varphi(t)$ y la variable auxiliar $P(t)$. La elección de las funciones f y g llevan a muchos diversos métodos de identificación, sin embargo la usada en la presente tesis, es conocido como RLS con una función variable en el tiempo basada en:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t).\left[y(t) - \varphi(t)\hat{\theta}(t-1)\right]$$
$$K(t) = P(t)\varphi(t)$$
$$P(t) = P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{1 + \varphi^T(t)P(t-1)\varphi(t)}$$

Por tanto si iniciamos las variables como $\hat{\theta}(0) = 0$ y un $P(0) = cI$ donde c tiene que ser un valor grande como 10^3.

Con este RLS se puede hacer una identificación tanto en tiempo real como una identificación externa, sin embargo en sistemas que van variando en el tiempo es necesario agregar un factor adicional que nos permita una identificación dinámica, un factor de olvido λ que se podría relacionar como una longitud de memoria obtenida con la siguiente formula: $\frac{1}{1-\lambda}$, por ejemplo si usamos un valor de $\lambda = 1$ obtenemos el mismo RLS ya visto, en cambio con un valor de $\lambda = 0.98$ tenemos una longitud de memoria de 50 tomas.

Este factor es incluido en el vector de disturbios:

$$V(\theta) = \sum_{k=1}^{t} \lambda^{t-k}[y(k) - \varphi^T(k)\theta]^2$$

Por tanto el método recursivo de mínimos cuadrados con factor de olvido queda expresado:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + P(t)\varphi(t).\left[y(t) - \varphi^T(t)\hat{\theta}(t-1)\right]$$

$$P(t) = \frac{1}{\lambda}\left[P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{\lambda + \varphi^T(t)P(t-1)\varphi(t)}\right]$$

Estas expresiones ya nos permiten tener una identificación en tiempo real con factores que pueden ser usados en el diseño de controladores PID adaptativos.

3.4.2. PID adaptativo basado en una identificación en tiempo real

Un controlador basado en la identificación en tiempo real requiere que los valores identificados se vayan actualizando, el controlador de Dahlin, (1968) se basa en una identificación con el método recursivo de mínimos cuadrados donde se tiene un vector de parámetros estimados, Bobál et al., (2006) han rediseñado el trabajo de Dahlin, y haciendo uso de un vector de parámetros estimados se calcula los valores de la salida proporcional, integral y derivativa.

$$\hat{\theta}^T(t) = [a_1, a_2, b_1]$$

Además considerando el vector de datos del siguiente modo:

$$\varphi^T(t-1) = [-y(t-1), -y(t-2); u(t-1)]$$

Para el presente PID se ha considerado un PID discreto presentando antes.

$$u_{(k)} = \Delta u_{(k)} + u_{(k-1)}$$

$$\Delta u_{(k)} = K_p\left(e_{(k)} - e_{(k-1)} + \frac{T_0}{T_i}e_{(k)} + \frac{T_d}{T_0}\left(e_{(k)} - 2e_{(k-1)} + e_{(k-2)}\right)\right)$$

Donde se definen los parámetros del controlador:

$$K_p = -\frac{(a_1 + 2.a_2)Q}{b_1}$$

$$T_i = -\frac{T_0}{\frac{1}{a_1 + 2.a_2} + 1 + \frac{T_d}{T_0}}$$

$$T_d = \frac{T_0 a_2 Q}{K_p b_1}$$

Una vez identificado el sistema se tienen valores para $\hat{\theta}^T(t)$ por tanto solo se requiere evaluar el factor Q, que viene definida:

$$Q = 1 - e^{-\frac{T_0}{B}}$$

Donde el factor B es definida por el diseñador del controlador, considerando que mientras más pequeño sea B se tendrá una respuesta más rápida en lazo cerrado, T_0 es el tiempo de muestreo.

Es necesario considerar que el vector de parámetros deben tener todos los valores diferentes de 0, para evitar una división de cero.

3.4.3. PID adaptativo sin identificación

El control PID adaptativo sin identificación se basa en la teoría de la interacción de un sistema completo dividido en N subsistemas llamados mecanismos donde cada mecanismo esta enumerado por $n \in N = \{1,2,3,\ldots,N\}$ y cada mecanismo consiste en una entrada x_n y una salida y_n relacionadas en una función F_n descrita como una función causal (Feng Lin et al., 2000).

$$F_n = x_n \to y_n, n \in N$$

Donde x_n, y_n son los espacios de entradas y salidas respectivamente, estas relaciones

pueden ser escritas como:

$$y_n = (F_n \circ x_n)(t) = F_n[x_n(t)]$$

Donde ∘ representa la composición.

Una interacción desde un mecanismo a otro se refiere a una conexión que lleva información denotada por c, y tenemos a C que es el conjunto de todas las conexiones, las etiquetas pre_c denotan el mecanismo que envía la información a través de una conexión c, y la etiqueta $post_c$ es quien recibe dicha información. Suponemos que la $I_n = \{c: pre_c = \text{n}\}$ que representa el conjunto de conexiones de la interacción de las entradas de enésimo mecanismo, del mismo modo podemos denotar por $O_n = \{c: post_c = \text{n}\}$ las enésimas interacciones entre las salidas de un mecanismo (An, Yuan, & Li, 2016).

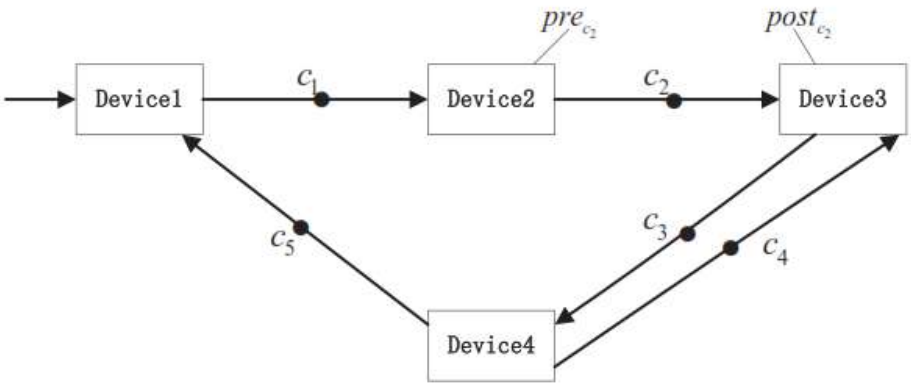


Figura 35.- Mecanismo e interacciones de un sistema

Fuente: An et al., 2016

En la figura 35 se muestran la simbología descrita antes, por ejemplo de la figura podemos ver que el conjunto de interacciones de entrada del mecanismo 3 (*Device3*) son $I_3 = \{c_2, c_4\}$ y para el mismo mecanismo las interacciones de salida son $O_3 = \{c_3\}$.

Considerando una interacción lineal entre los componentes del PID, tenemos que $u_n(t)$ es una entrada externa, por tanto $x_n(t)$ puede ser expresado como:

$$x_n(t) = u_n(t) + \sum_{n \in I_n} \alpha_c y_{pre_c}(t)$$

Donde α_c son los pesos de conexión, con esta lineal interacción, la dinámica del sistema puede ser descrito como:

$$y_n(t) = F_n\left[u_n(t) + \sum_{n \in I_n} \alpha_c y_{pre_c}(t)\right]$$

El propósito del algoritmo adaptativo es el adaptar los pesos de conexión α_c en orden que el índice de actuación $E(y_1 \dots y_n, u_1 \dots u_n)$ pueda ser minimizado. Si los pesos de las conexiones α_c son adaptados entonces $E(y_1 \dots y_n, u_1 \dots u_n)$ puede ir decreciendo con el tiempo.

$$\dot{\alpha}_c = \left(\sum_{s \epsilon O_{post_c}} \alpha_s \dot{\alpha}_c \frac{\frac{dE}{dy_{post_c}} \circ F'_{post_s}[x_{post_s}]}{\frac{dE}{dy_{post_c}} \circ F'_{post_s}[x_{post_s}] \circ y_{post_c}} - \gamma \frac{\partial E}{\partial y_{post_c}} \right) \circ F'_{post_s}[x_{post_s}] \circ y_{pre_c}$$

Donde γ es un coeficiente de adaptación.

El algoritmo PID de un sistema de control es un caso especial de la aplicación de la formulación descrita, descomponiendo el sistema de un controlador PID aplicado a un sistema SISO (*single input, single output*) como se muestra en la figura 36, el mecanismo 1 es el proporcional, el mecanismo 2 es la parte integral, y el mecanismo 3 es la parte derivativa, siendo el sistema a controlar o la planta el mecanismo 4.

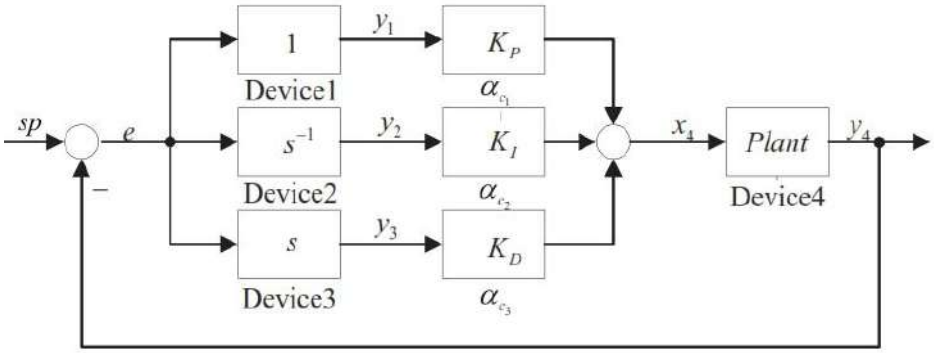


Figura 36.- Controlador PID

Fuente: An et al., 2016

Los mecanismos están conectados a través de α_c=K_p, K_t ~~o~~ K_d, si no consideramos la realimentación del sistema tenemos que $O_4 = \emptyset$, podemos desarrollar la ecuación de adaptación del siguiente modo:

$$\dot{\alpha}_c = -\gamma \frac{\partial E}{\partial y_4} \circ F'_4[x_4] \circ y_{pre_c}$$

Y la interacción de actuación a minimizar sería el error elevado al cuadrado:

$$E = e^2 = (sp - y_4)^2$$

Considerando la teoría de la derivada parcial de Fréchet de la planta $\frac{\partial E}{\partial y_4}$ sería:

$$\frac{\partial E}{\partial y_4} = -2(sp - y_4) = -2e$$

Obteniendo por tanto un algoritmo de sintonización basado en Fréchet.

El símbolo $\circ$ puede ser reemplaza por el producto porque E y F son funciones trascendentales en el algoritmo de interacción adaptativa.

$$\dot{K}_p = 2\gamma e F'_4[x_4] \circ y_1$$
$$\dot{K}_i = 2\gamma e F'_4[x_4] \circ y_2$$
$$\dot{K}_d = 2\gamma e F'_4[x_4] \circ y_3$$

Donde $F'_4[x_4]$ es la derivada de Fréchet de la planta, que se puede aproximar como sigue:

$$F'_4[x_4] \circ h = \int_0^t f_x(x(\tau), \tau) h(\tau) d\,\tau$$

Que a su vez es aproximada usando otro valor constante β

$$F_4'[x_4] \circ h = \beta h$$

Por tanto podemos hallar los siguientes valores considerando que el factor de adaptación incluye al factor 2β:

$$\dot{K}_p = \gamma e y_1$$
$$\dot{K}_i = \gamma e y_2$$
$$\dot{K}_d = \gamma e y_3$$

Con estos factores ya se puede formar un controlador PID con auto sintonización que en un diseño continuo es posible implementar por ejemplo en el software Simulink como se muestra en la figura 37.

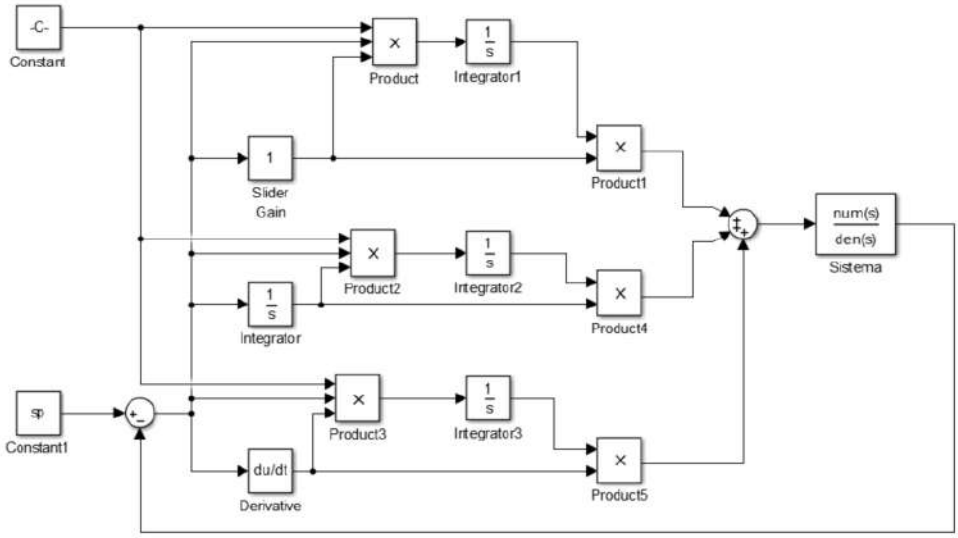


Figura 37.- Controlador PID adaptativo sin identificación
Fuente: Propia (elaborado en Simulink)

Debido a la utilización de un microcontroladores digitales es necesario discretizar las soluciones del PID adaptativo de sintonización automática, para ello podríamos usar las formulaciones de la derivada e integral como sigue:

$$\frac{de_{(t)}}{dt} = \frac{e_{(k)} - e_{(k-1)}}{T}$$

$$\int_0^t e_{(t)} dt = \sum_{i=0}^{k-1} T e_{(i)}$$

Y desarrollando la formulación presentado en la figura 36, tenemos que:

$$e = sp - y$$

$$u = k_p + k_i + k_d$$

$$u = \gamma . e . \int_0^t e^2{}_{(t)} dt + \gamma . \int_0^t e_{(t)} dt \, . \int_0^t \int_0^t e_{(t)} dt . e_{(t)} dt +$$

$$\gamma \frac{de_{(t)}}{dt} \int_0^t \frac{de_{(t)}}{dt} . e_{(t)} dt$$

De donde podríamos separar los siguientes factores y discretizarlos según las formulaciones ya presentadas:

$$k_p = \gamma . e . \int_0^t e^2{}_{(t)} dt$$

$$kd_{p(k)} = \gamma . e_k . \sum_{i=0}^{k-1} T_s e^2{}_{(i)}$$

$$k_i = \gamma . \int_0^t e_{(t)} dt \, . \int_0^t + \int_0^t e_{(t)} dt . e_{(t)} dt$$

$$kd_{i(k)} = \gamma . \sum_{i=0}^{k-1} T_s e_{(i)} \, . \sum_{j=0}^{k-1} \sum_{m=0}^{k-1} T_s e_{(m)} T_s e_{(j)} +$$

$$k_d = \gamma \frac{de_{(t)}}{dt} \int_0^t \frac{de_{(t)}}{dt} . e_{(t)} dt$$

$$kd_{d(k)} = \gamma \frac{e_{(k)} - e_{(k-1)}}{T_s} \sum_{i=0}^{k-1} \frac{e_{(k)} - e_{(k-1)}}{T_s} . T_s e_{(i)}$$

$$kd_{d(k)} = \gamma \frac{e_{(k)} - e_{(k-1)}}{T_s} \sum_{i=0}^{k-1} \left(e_{(k)} - e_{(k-1)} \right) . e_{(i)}$$

$$u_k = kd_{p(k)} + kd_{i(k)} + kd_{d(k)}$$

Otra solución posible la presenta An et al formulando la integral con la siguiente ecuación:

$$\int_0^{kT_s} f(t) dt = y(k) \approx y(k-1) + \frac{f(k) + f(k-1)}{2} T_s$$

Donde T_s es el tiempo de muestreo, $y(k)$ es la señal de salida y $f(k)$ es la señal de entrada de cada integral.

Desarrollando para cada término tendremos:

$$\dot{K}_p = \gamma e(t) . e(t)$$

$$\dot{K}_p = \gamma e(t)^2$$

$$\int_0^{kT_s} \gamma e(t)^2 dt = \gamma T_s \frac{e(k)^3}{3}$$

$$Kp_{(k)} = \gamma T_s \left[\frac{e(k-1)^3}{3} + \frac{e(k)^2 + e(k-1)^2}{2} \right]$$

$$\dot{K}_i = \gamma e(t) \int_0^{kT_s} e(t) dt$$

$$\int_0^{kT_s} e(t) dt = T_s \frac{e(k)^2}{2} = T_s \left[\frac{e(k-1)^2}{2} + \frac{e(k) + e(k-1)}{2} \right]$$

$$\dot{K}_i = \gamma e(t) T_s \left[\frac{e(k) + e(k-1)^2 + e(k-1)}{2} \right]$$

$$F_i(k) = \left[\frac{e(k) + e(k-1)^2 + e(k-1)}{2} \right]$$

$$K_i = \int_0^{kT_s} \gamma e(t) T_s F_i(k) dt$$

$$K_i = \gamma F_i(k) T_s \int_0^{kT_s} e(t) dt$$

$$Ki_{(k)} = \gamma {T_s}^2 F_i(k)^2$$

$$\dot{K}_d = \gamma e(t) \frac{de(t)}{dt}$$

$$K_d = \int_0^{kT_s} \gamma e(t) \frac{de(t)}{dt} dt$$

$$K_d = \gamma T_s e(k)[e(k-1)+1] - \int_0^{kT_s} \gamma e(t) \frac{de(t)}{dt} dt$$

$$Kd_{(k)} = \gamma T_s \frac{e(k)[e(k-1)+1]}{2}$$

Con estos valores hallados An et al., proceden a discretizar el PID de acuerdo a la siguiente formulación:

$$u_k = u_{k-1} + a.e_{(k)} + b.e_{(k-1)} + c.e_{(k-2)}$$

$$a = Kp_{(k)} + Ki_{(k)}.\frac{T_s}{2} + \frac{Kd_{(k)}}{T_s}$$

$$b = -Kp_{(k)} + Ki_{(k)}.\frac{T_s}{2} - \frac{2.Kd_{(k)}}{T_s}$$

$$c = \frac{Kd_{(k)}}{T_s}$$

Las dos soluciones se van a comparar en el siguiente capítulo.

3.5. Modulación por ancho de pulso

La modulación por ancho de pulso (PWM por sus siglas en inglés) es la base para el control de la electrónica de potencia que con la excepción de algunos convertidores resonantes, la gran mayoría de circuitos electrónicos de potencia son controlados por señales PWM de varias formas (Vasca & Iannelli, 2012).

Otra forma de entender a la modulación por ancho de pulso es definirla como una técnica para obtener resultados analógicos con medios digitales, a través de un control digital, digamos un pin del Arduino se crea una señal cuadrada de conmutación entre encendido y apagado, este patrón de encendido y apagado puede simular tensiones en el medio completo y tensión de desactivación, es decir totalmente encendido y totalmente apagado.

La característica principal de una modulación por ancho de pulso es la frecuencia de oscilación es decir la frecuencia en donde suceden estos ciclos de encendido y apagado de la señal, para los ledes el tiempo debe ser más rápido de lo que el ojo humano pueda detectar para evitar el efecto de flicker, es por ello que una velocidad por encima de 200 Hz es aceptado (Rosen, 2011).

En la figura 38 las líneas verdes representan la frecuencia de oscilación que son de periodo regular, para el Arduino la frecuencia es de 500 Hz que es superior a los 200

Hz sugeridos en la bibliografía, por tanto no se quiere sino hacer un llamado a la función del Arduino para la utilización del PWM, en la misma figura podemos observar los diversos valores de la función *analogWrite* del Arduino de acuerdo al ciclo de trabajo, que son la designación del tiempo de encendido y tiempo de apagado en cada ciclo a la frecuencia.

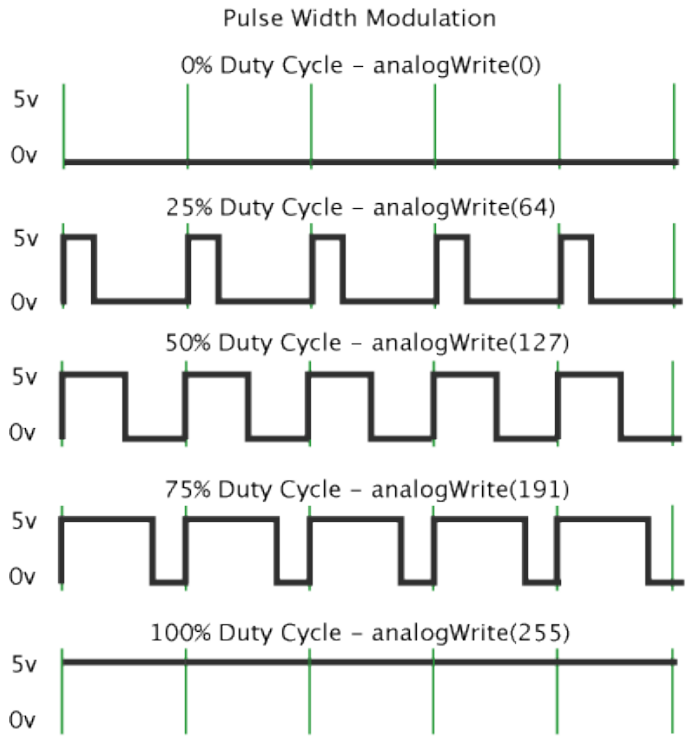


Figura 38.- Modulación por ancho de pulso
Fuente: Arduino.cc, 2010

3.6. El control de iluminación

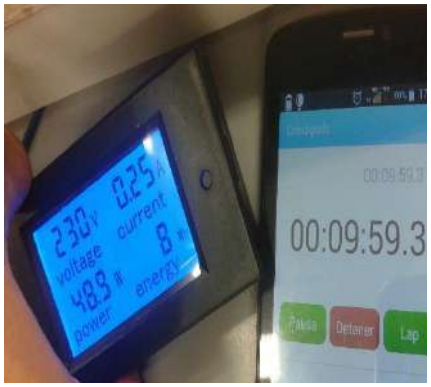
En el desarrollo de la presente tesis se ha elaborado una serie de pruebas de diversas tecnologías de iluminación destacando principalmente la tecnología de iluminación fluorescente y la tecnología led llegando a la conclusión de la necesidad de un cambio de tecnología de luminarias; de las luminarias convencionales a la tecnología led debido principalmente a las ventajas en ahorro de consumo de hasta un 54% menos según la tabla 10 que son mediciones realizadas en el laboratorio como se muestra en la figura 38.

Tabla 10.- Consumo energético de acuerdo a la tecnología usada

Tipo	Consumo por hora(W)	Ahorro Anual	Costo de la Tecnología
Balasto y arrancador	78	0	S/.30
Balastro Electrónico	48	38%	S/.40
Tubo Led	36	54%	S/.70

Fuente: Propia

a) 

b) 

c) 

Figura 39.- Medición de consumo de luminarias a) Luminaria fluorescente con arrancador normal, b) Luminaria fluorescente con arrancador electrónico, c) Tubo led

Fuente: Propia

La ventaja de la tecnología led es evidente, considerando además que el tubo led estaba en su máxima capacidad durante la prueba; las luminarias de tecnología led dan la posibilidad de control y con ello reducir más su consumo aumentando la eficiencia y el confort del consumidor.

El esquema de control se muestra en la figura 40, donde se nota que el primer paso tras la lectura de los sensores es la toma de decisión del sistema de aprendizaje automático, y luego el control de la iluminación PID adaptativo; en el desarrollo se ha estudiado dos tipos de sistemas automáticos de control y dos sistemas de PID adaptativo en el siguiente capítulo se analizará su implementación en sistemas embebidos y su comparación.

Como se observa en el diagrama de flujo el sistema de aprendizaje automático también va ser quien determine el nivel de operación de las luminarias, es posible usar luminarias convencionales en este esquema en el cual se obviaría el control PID y sería un control de encendido y apagado controlado solo por el sistema de aprendizaje automático.

El controlador que se ha desarrollado se adapta a cualquier usuario y a cualquier ambiente, esto debido a dos características principales: el sistema automático de aprendizaje y el PID adaptativo.

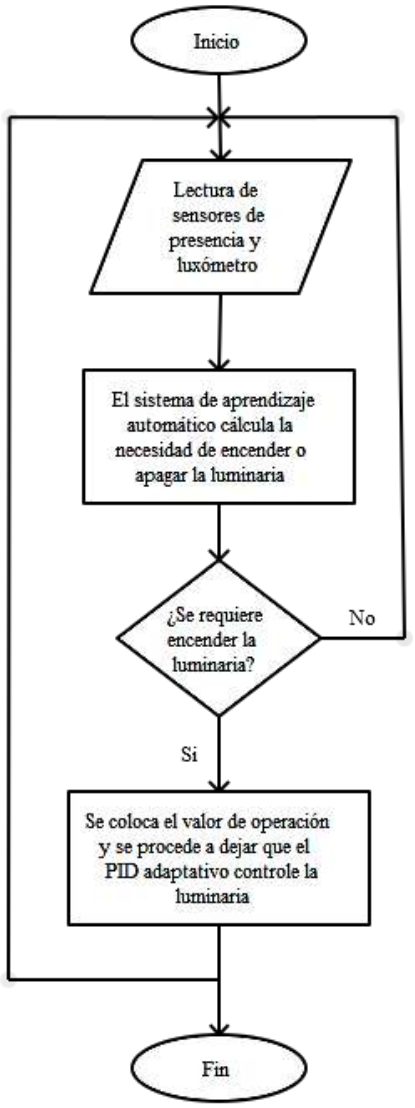


Figura 40.- Diagrama de flujo del control

Fuente: Propia (Elaborado en Pencil)

El sensor de iluminación es el BH-1750 cuyo rango de medición va desde 1 – 65535 lux con un tiempo mínimo de medición de 16 ms (ROHM Semiconductor, 2011), la figura 40 nos muestra el modulo del sensor, este sensor se comunica al microprocesador a través de comunicación IIC.

Figura 41.- Módulo BH1750

Fuente: ROHM Semiconductor, 2011

El sensor de presencia PIR (*passive infrared*) es un sensor con tecnología de infrarrojos que detectan los niveles de la irradiación infrarroja y obtiene un estado al encender, una vez que tiene este estado está a la espera de un cambio de estado que originará una salida, este sensor tiene una sensibilidad en distancia de hasta 6 metros en frente formando una semicircunferencia que le da un ángulo de detección de 110° x 70° (Adafruit Industries, 2016).

Figura 42.- Sensor PIR
Fuente: Adafruit Industries, 2016

El sensor permite la configuración de la sensibilidad, el tiempo de encendido una vez detectado un movimiento, y la posibilidad de usar un *retriggering* que es la habilidad de seguir activado mientras siga detectando movimiento, este comportamiento se puede observar en la gráfica de la figura 43

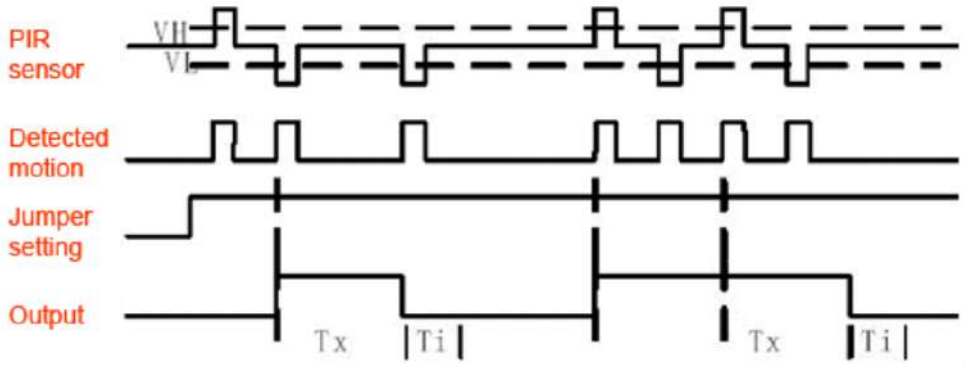


Figura 43.- Funcionamiento del sensor PIR
Fuente: Adafruit Industries, 2016

Donde el Tx es el tiempo de encendido configurable desde una resistencia variable, el tiempo mínimo es de 2.5 segundos y el máximo de 2 minutos aproximadamente, Ti es el tiempo de inhabilitación de detección que es de 1.5 segundos y es para dar una mayor estabilidad al sensor.
Como se observa la segunda activación de la salida tiene un tiempo de duración mayor debido al comportamiento de volverse activar conocido como retriggering.

Capítulo 4
Desarrollo de software y hardware

4.1. Introducción

En el desarrollo de la presente tesis se han estudiado dos formas de aprendizaje automático y dos esquemas de control adaptativo en el presente capítulo se va armar y desarrollar los algoritmos para estos sistemas.

Antes de pasar a desarrollar un algoritmo para el sistema embebido se ha visto necesario probar los diferentes sistemas de aprendizaje y de control en un simulador creado en Python 2.7.12 que nos permite evaluar las capacidades de estos sistemas y de eso modo poder diseñar un sistema embebido estable y de ese modo armar un prototipo que incluya el microprocesador y el SBC Odroid.

4.2. Sistema de aprendizaje automático

Los algoritmos estudiados en el capítulo 2, los cuales fueron el algoritmo de aprendizaje basado en redes neuronales y el algoritmo de aprendizaje basado en árboles de decisiones han demostrado ser algoritmos muy estables y se van a probar ambos algoritmos en su aplicación en un sistema embebido considerando primero el algoritmo que se muestra en el diagrama de flujo de la figura 44 que nos muestra el primer paso necesario en el sistema de aprendizaje, en este paso se va almacenando los datos para el aprendizaje automático. En el capítulo 2 se ha trabajado con 6 datos de entrenamiento y otros 6 datos de validación, con el cual el sistema de aprendizaje automático llego a converger en un tiempo razonable, por tanto es posible usar este mínimo de datos en total 12 o usar más datos con un máximo de 18 datos de entrenamiento, si bien va ser posible ir guardando más datos para el entrenamiento se poner ese límite para acelerar el aprendizaje, además de darle la posibilidad de ser un sistema adaptativo, es decir si un nuevo usuario realiza un número suficiente de cambios el sistema se va adaptar al nuevo usuario olvidándose de las reglas aprendidas del usuario anterior, el algoritmo de la actualización de parámetros se observa en la figura 45, este diagrama fue diseñado desde el punto de vista del sistema embebido.

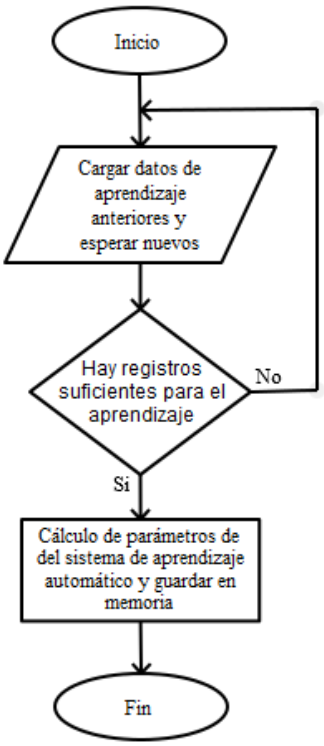


Figura 44.- Toma de datos para el sistema de aprendizaje
Fuente: Propia (Elaborado con Pencil)

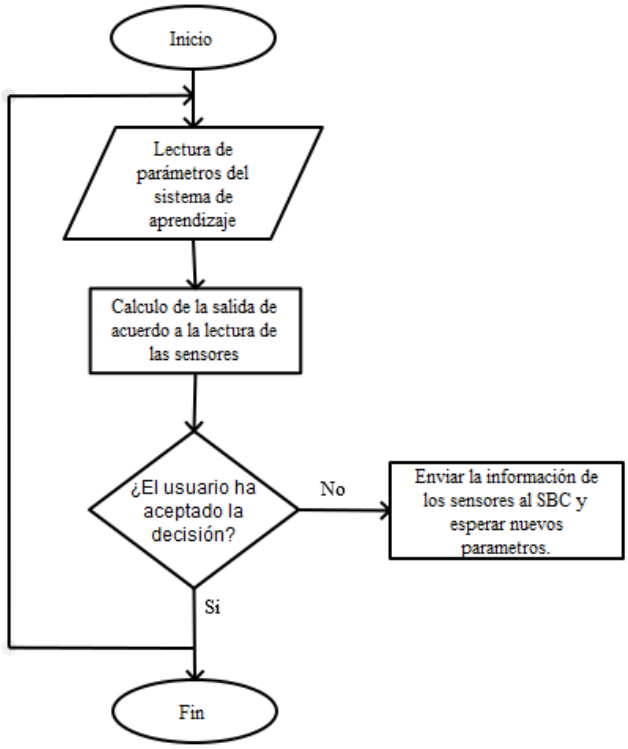


Figura 45.- Actualización de parámetros de sistema de aprendizaje
Fuente: Propia (Elaborado con Pencil)

Basado en estos diagramas se ha desarrollado el programa de simulación cuyo código está en el Anexo B, el diagrama de flujo del programa es el que se muestra en la figura 46.

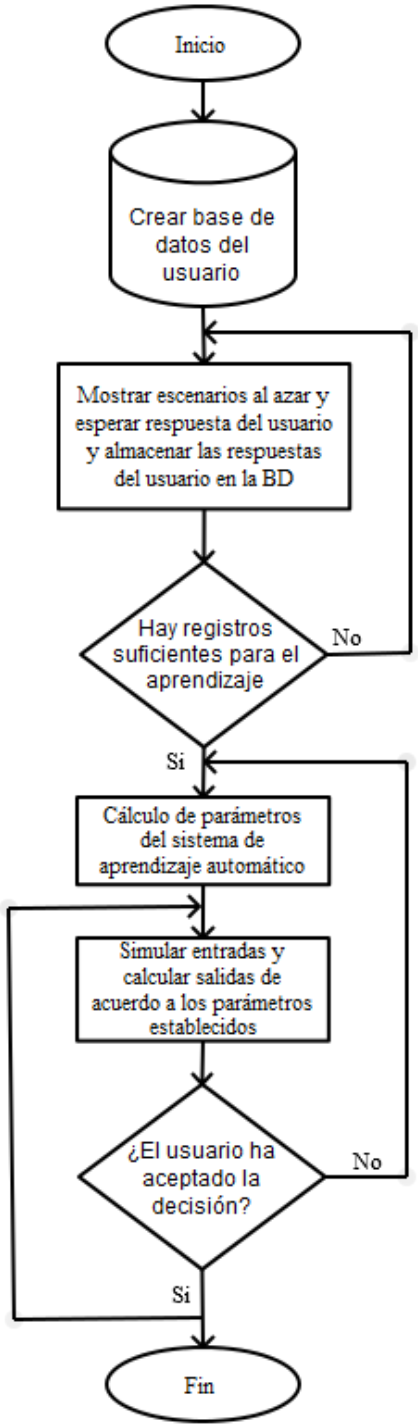


Figura 46.- Actualización de parámetros de sistema de aprendizaje
Fuente: Propia (Elaborado con Pencil)

La figura 47 muestra una captura del programa de simulación del sistema automático de aprendizaje, donde nos muestra los diversos escenarios que genera al azar, la imagen de la puerta indica la presencia de alguna persona en el ambiente y el brillo de la imagen representa el nivel de iluminación, que también se muestra en una etiqueta debajo de la imagen.

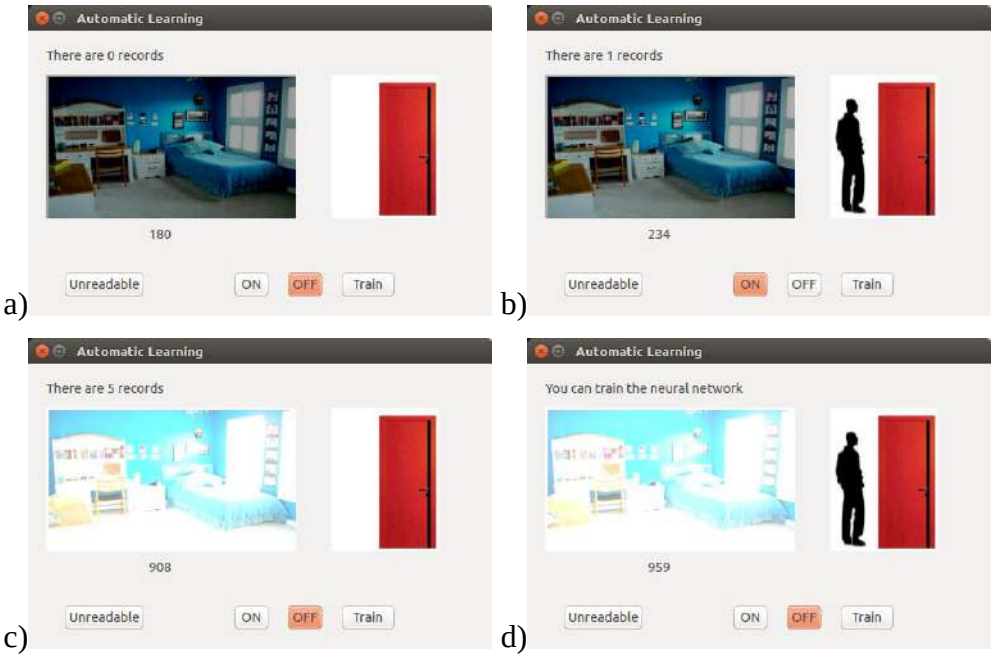


Figura 47.- Simulación de Escenarios: a) Habitación sin iluminación y sin personas, b) Habitación sin iluminación con personas, c) Habitación iluminada sin personas, d) Habitación iluminada con personas

Fuente: Propia (Elaborado con Python)

El sistema de aprendizaje usa la librería Scikit-learn (Pedregosa et al., 2011) que otorga módulos de aprendizaje de redes neuronales y árboles de decisión que nos permite desarrollar un programa de acuerdo al diagrama de flujo de la figura 46.

Considerando la notación estudiada en el capítulo 2, tenemos la siguiente representación de la red neuronal y sus valores.

$$s_2 = sigmoidal(w_2.s_1 + b_2)$$

$$s_1 = sigmoidal(w_1.p + b_1)$$

$$w_1 = \begin{bmatrix} -0.2966 & 0.4205 & -0.5254 & -0.2908 \\ -0.3017 & -0.9925 & -0.3437 & 0.5644 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} -0.1287 \\ 0.0572 \\ +0.0961 \\ -0.2556 \end{bmatrix}$$

$$w_2 = \begin{bmatrix} -0.03563 & 0.2054 \\ 0.8841 & -1.3374 \\ -0.4157 & -0.3744 \\ -0.8409 & 0.4166 \end{bmatrix}$$

$$b_2 = \begin{bmatrix} -0.0066 \\ 0.3683 \end{bmatrix}$$

Estos valores pueden ser transferidos a un sistema embebido para su utilización, así también si el usuario modifica una acción estos coeficientes se actualizan.

El segundo modelo de aprendizaje basado en arboles de decisión se muestra en la figura 48.

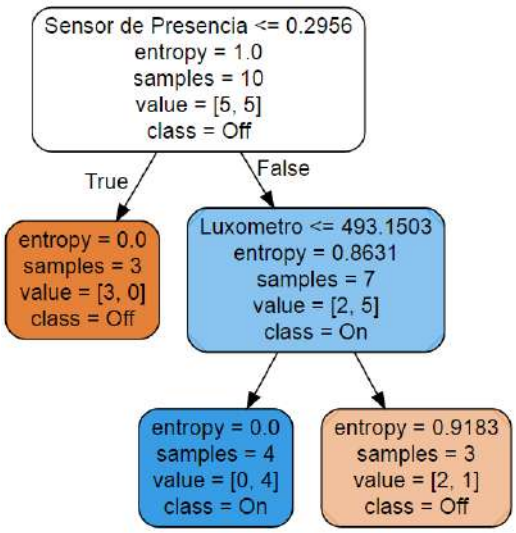


Figura 48.- Árbol de decisión aprendida
Fuente: Propia (Elaborado con Python)

Para trasladar este esquema al sistema embebido se debe considerar solo el valor del tercer nodo que es valor desde el cual considera baja la iluminación, debido a que la decisión respecto al sensor de presencia se mantendrá constante, al mantener ese esquema se da mayor robustez al sistema.

4.3. Sistema de control PID adaptativo

Los algoritmos estudiados en el capítulo 3 sobre el control PID adaptativo o de auto sintonización, estos se pueden diseñar con identificación en tiempo real y sin identificación cuyos diagramas de flujo se pueden observar en las figuras 49 y 50, ambos algoritmos se van a comprobar en un simulador elaborado en Python para dicho fin.

El simulador cuenta con un menú para configurar los parámetros de los PID adaptativos, así también un menú para seleccionar el tipo de sistema de aprendizaje automático que se va utilizar.

La librería necesaria para crear un simulador de sistemas continuos o discretos en Python es Scipy (Jones, Oliphant, & Peterson, 2001) con esta librería es posible analizar funciones de transferencia en tiempo continuo o discreto y así también simular con tiempos específicos como los que se realiza comúnmente en Matlab y Simulink, las ventajas son una mayor flexibilidad y el uso de software libre.

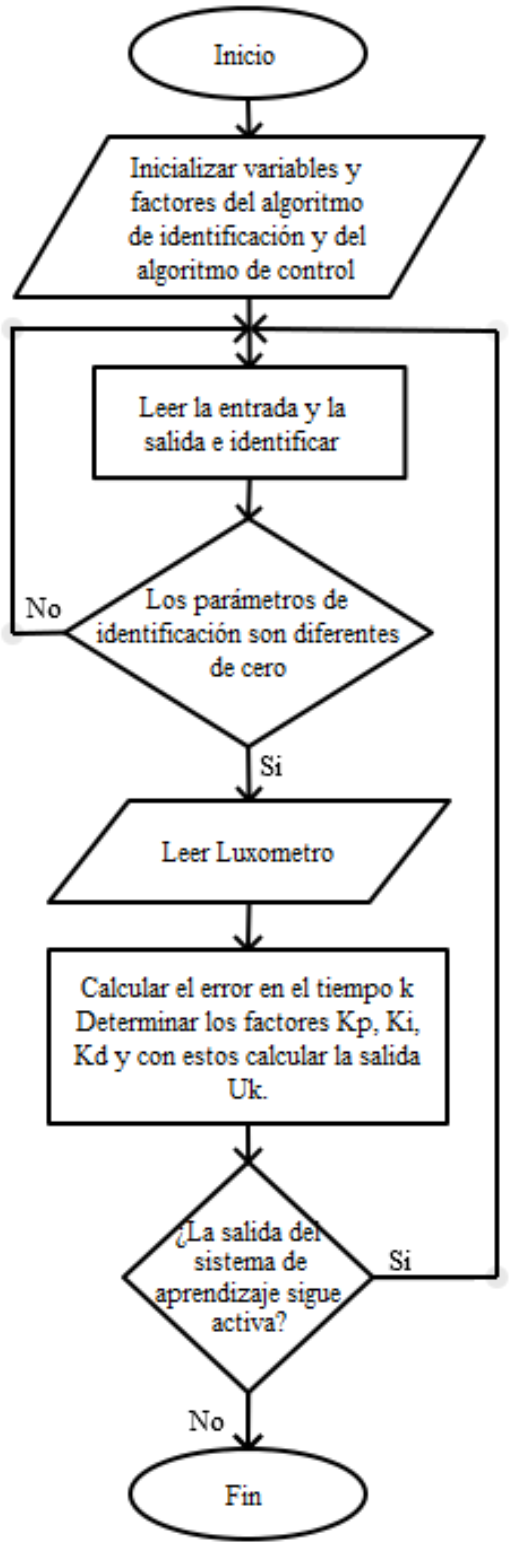


Figura 49.- Algoritmo de control PID adaptativo con identificación

Fuente: Propia (Elaborado con Pencil)

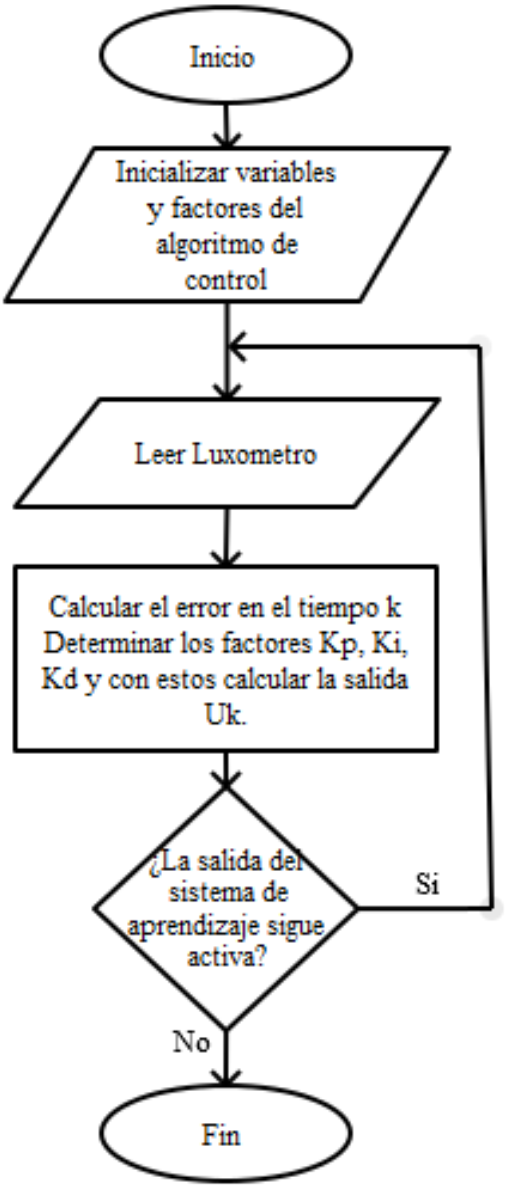


Figura 50.- Algoritmo de control PID adaptativo sin identificación
Fuente: Propia (Elaborado con Pencil)

Para la simulación se ha utilizado dos funciones de transferencias que fueron identificadas en un módulo de ensayos desarrollado en el laboratorio de Sistemas Automáticos de Control de la Universidad de Piura cuyo código fuente y respuesta a entrada escalón se encuentran en el Anexo C, estas funciones de transferencia son las siguientes:

$$G1(s) = \frac{4836}{s + 44.14}$$

$$G2(s) = \frac{2354}{s + 71.49}$$

La base para las simulaciones será el sistema $G2(s)$ en el simulador que se muestra en la figura 50 cuyo código fuente se encuentra en el Anexo D.

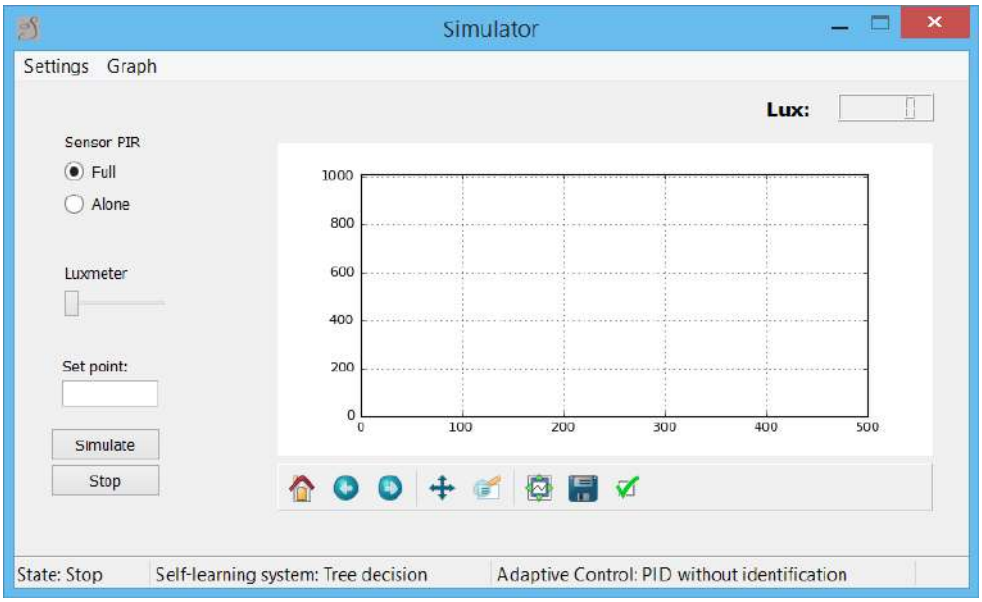


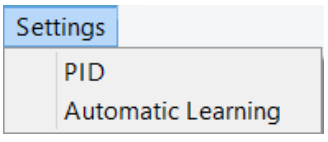
Figura 51.- Simulador de Control de Iluminación

Fuente: Propia (Elaborado con ¨Python)

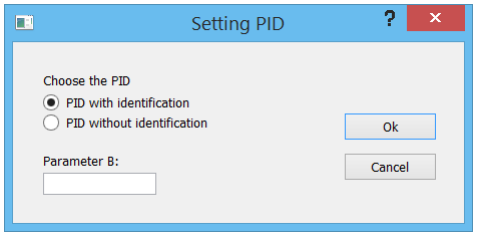
La figura 51 nos muestra la pantalla principal del simulador a la izquierda se encuentran los controles que sirven de entrada tales como el sensor de presencia nos permite elegir entre un ambiente ocupado o vacío, una barra deslizadora que nos permite seleccionar el nivel de iluminación del ambiente, un cuadro de texto para ingresar el valor del punto de operación, y los controles para iniciar la simulación o detenerla. Se cuenta con una gráfica desarrollada en matplotlib de Python donde se visualizara las diversas imágenes que nos permite mostrar el simulador tales como: salida y punto de operación, señal de control, la evolución de los parámetros del control PID, y una gráfica del funcionamiento de la toma de decisiones.

En la parte inferior se cuenta con una barra de estados que muestran el estado de la simulación, el tipo de sistema de autoaprendizaje, el tipo de control adaptativo y en derecha aparece el estado de la luminaria en el momento de la simulación. El diseño se ha pensado de forma que el uso sea de forma intuitiva.

a)



b)



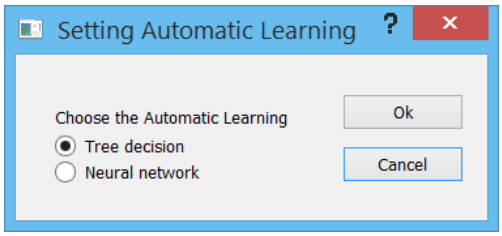
c) 

Figura 52.- Barra de menús. a) Menú de configuración, b) Opciones de configuración del control PID adaptativo, c) Opciones de selección del sistema automático de aprendizaje.

Fuente: Propia (Elaborado con ¨Python)

En la figura 52 podemos ver el menú de configuraciones, donde podemos configurar el único parámetro necesario en los PIDs adaptativos, así también se puede elegir qué tipo de sistema de autoaprendizaje se desea usar para la simulación, estos sistemas son los que exporta el programa de la simulación.

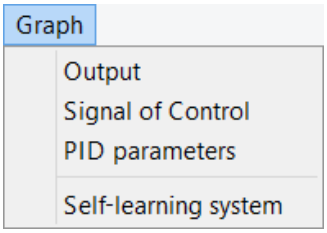


Figura 53.- Barra de menús - Graficas

Fuente: Propia (Elaborado con ¨Python)

La figura 53 nos muestra las opciones las gráficas de salida que existen y al dar clic en ellas durante la simulación se mostraran la gráfica de la opción seleccionada.

4.4. Pruebas en el simulador

Para la primera prueba se va considerar un ambiente que se va iluminando poco a poco con la luz natural y un punto de operación como si se tratara de una oficina de uso general que debería contar con 500 luxes tal como se observa en el Anexo A.

Prueba 1:

Punto de operación 500 lux

Estado inicial: ambiente con personas, y sin luz natural

Al instante 50 la luz natural aumenta a 40 luxes

Al instante 100 la luz natural aumenta a 80 luxes

Al instante 150 la luz natural aumenta a 120 luxes

Al instante 200 la luz natural reduce hasta 40 luxes

Al instante 250 la luz natural aumenta a 720 luxes

La figura 54 nos muestra los resultados en el simulador de estas condiciones considerando un controlador PID con identificación, cuyo parámetro es $B = 1.2$ que le proporciona una estabilidad frente a los disturbios, es necesario considerar en sistemas de control con identificación en tiempo real usar un sensor exclusivo para detectar la respuesta del sistema y otro que cense también los disturbios, a partir del instante 250 el punto de operación está por debajo de la iluminación que brinda la luz natural, es por ello que la señal de control se queda en 0.

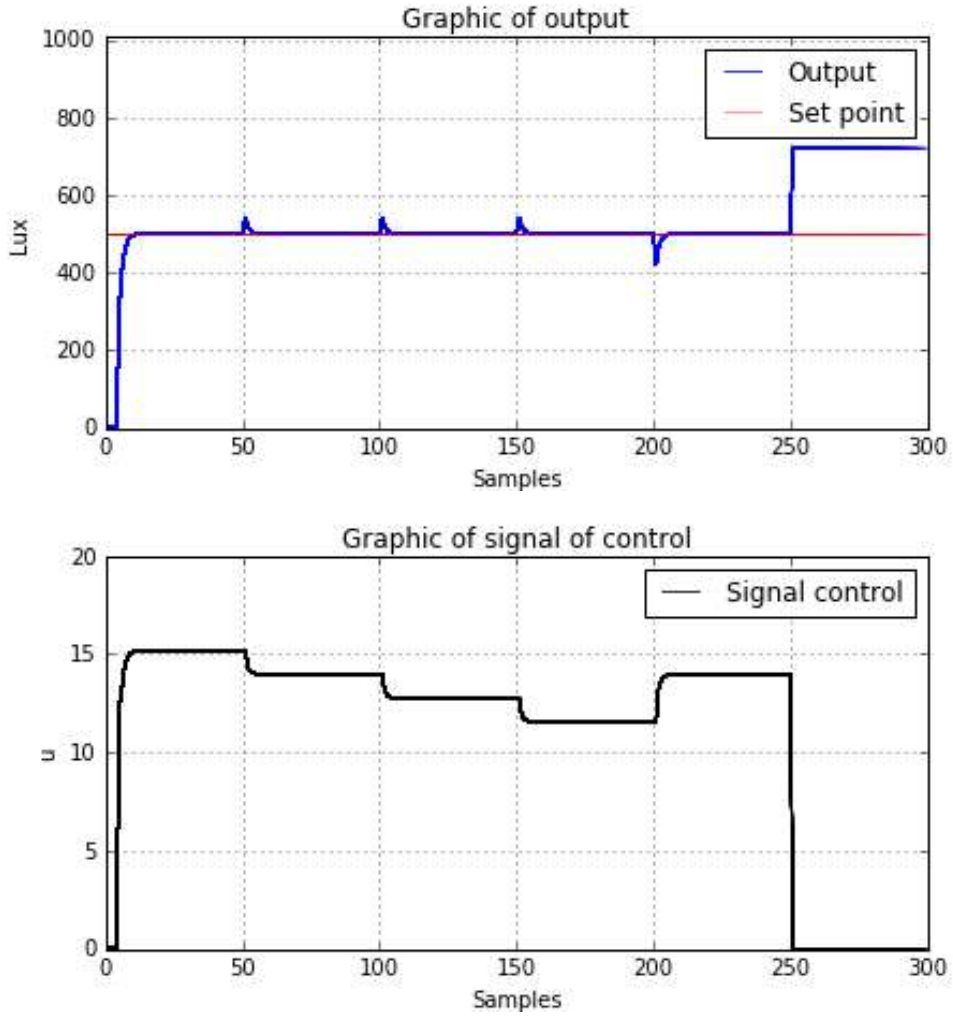
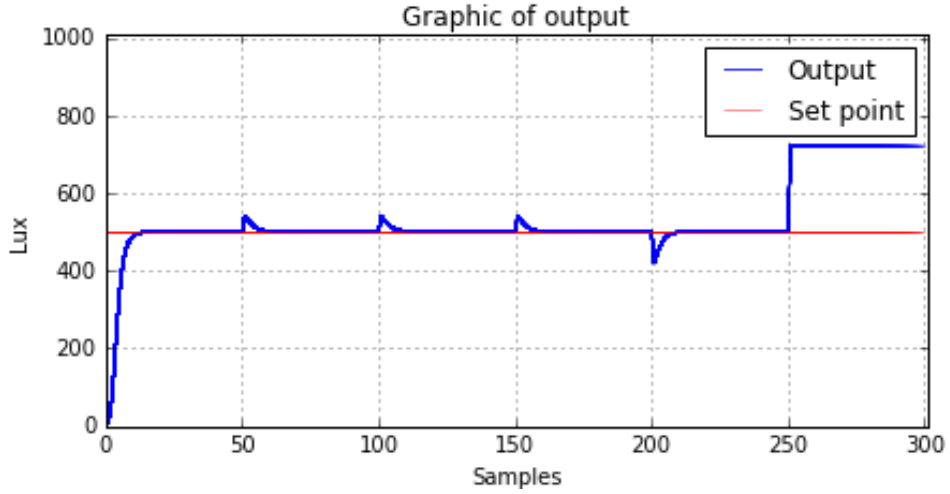


Figura 54.- Prueba 1: Respuesta del PID adaptativo con identificación
Fuente: Propia (Elaborado con Python)

La figura 55 nos muestra los resultados en el simulador de las condiciones determinadas para la prueba 1 con un controlador PID adaptativo sin identificación, cuyos parámetros son de $\gamma = 10^{-14}$ que al igual que el sistema con identificación tiene una respuesta rápida y no es necesario tener dos sensores independientes en este caso.



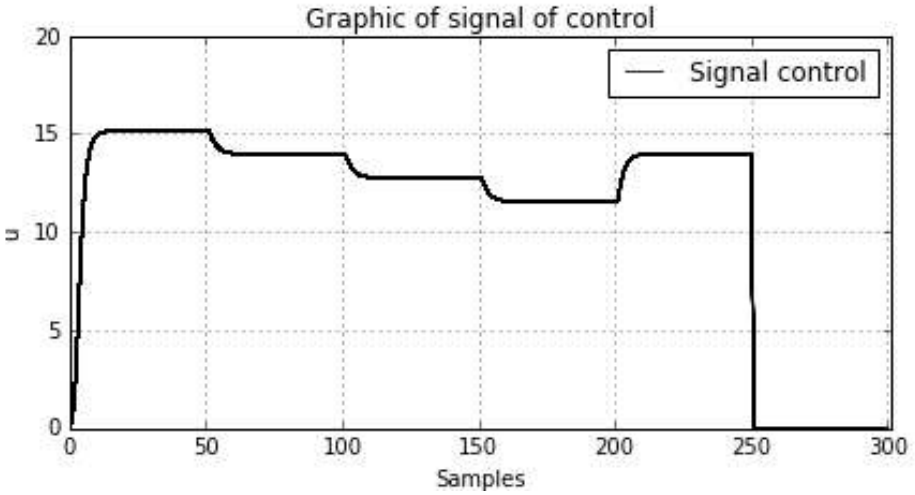


Figura 55.- Prueba 1: Respuesta del PID adaptativo sin identificación

Fuente: Propia (Elaborado con ¨Python)

En la discretización de los parámetros del PID adaptativo sin identificación se habían encontrado dos formas de hacerlo uno según la conversión en sumatorias de la integral y otra por una formulación presentada por An et al, en la imagen 55 es la solución resolviendo las integrales por sumatorias, en la figura 56 se observa la solución presentada por An et al con un valor de $\gamma = 10^{-12}$, y como se observa no llega a converger al punto de operación en ningún momento, es por ello que en lo sucesivo se va utilizar la solución de la integral por sumatorias al referirnos al control del PID adaptativo sin identificación, con los ensayos se ha determinado que el factor γ debe ser muy pequeño en sistemas donde error es grande y debe ser algo más grande en sistemas con errores pequeños, al aumentar γ es más rápido el tiempo de establecimiento, sin embargo puede conducir a sobre oscilaciones y perdida de estabilidad de este método al usar valores γ mayores.

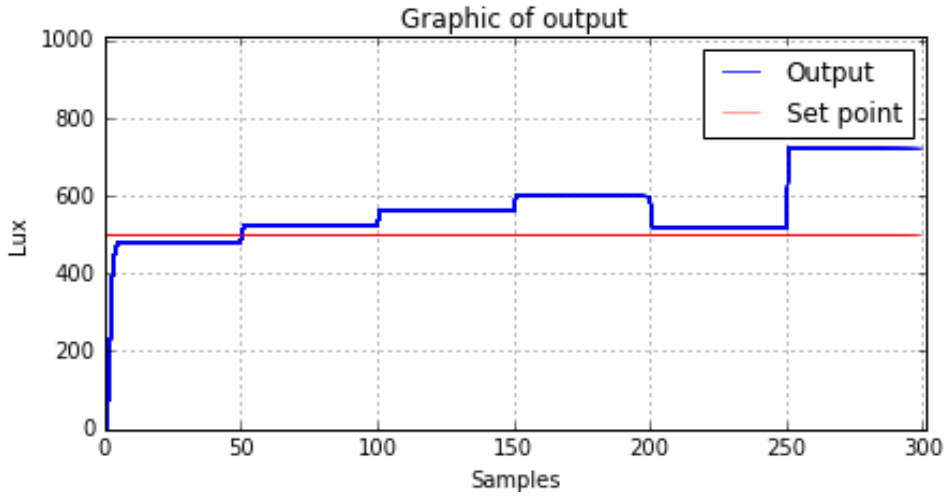


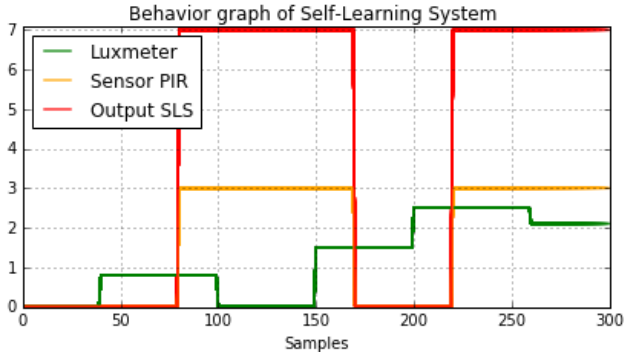
Figura 56.- Prueba 1: Respuesta del PID adaptativo sin identificación solución de An et al.

Fuente: Propia (Elaborado con Python)

Para una segunda prueba se va considerar la iluminación mínima de la cabecera de la cama de una habitación que son 200 luxes, considerando que por momentos las

personas van a salir de la habitación.
Prueba 2:
Punto de operación 200 lux
Estado inicial: ambiente sin personas, y sin luz natural
Acción de las personas:
Al instante 80 una persona entra en el ambiente
Al instante 170 el ambiente queda desocupado
Al instante 220 una persona entra en el ambiente
Al instante 270 el ambiente queda desocupado
Acción de la luz natural
Al instante 40 la luz natural aumenta a 80 luxes
Al instante 100 la luz natural disminuye a 0 lux
Al instante 150 la luz natural aumenta a 150 luxes
Al instante 200 la luz natural aumenta hasta 250 luxes
Al instante 250 la luz natural aumenta hasta 210 luxes
En la prueba 1, el sistema de aprendizaje utilizado fue el de árbol de decisión, para esta segunda prueba se observara la respuesta de ambos sistemas de autoaprendizaje con la gráfica correspondiente, que se puede observar en la figura 57, donde se ve como el sistema se va encendiendo y apagando de acuerdo al sensor de presencia y al luxómetro, la línea roja en 7 significa que la red neuronal ha decidido encender la luminaria y en 0 es la decisión de apagarla, la línea verde es la respuesta del luxómetro escalado de 0 a 10, y la línea anaranjada es la respuesta del sensor de presencia 3 en caso de detectar una presencia y 0 en caso contrario.

a)



b)

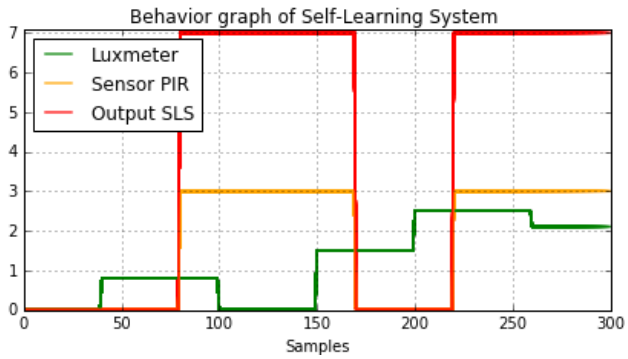


Figura 57.- Prueba 2: Respuesta del sistema de aprendizaje. a) Redes neuronales, b) Árbol de decisión

Fuente: Propia (Elaborado con Python)

Esta sistema de autoaprendizaje fue entrenado considerando un mínimo de iluminación de 500 luxes es por ello que a pesar de estar sobre los 200 lux que es el punto de operación para esta habitación el sistema sigue intentando encender la luminaria sin embargo como se observa en la figura 58 y 59 los controladores PID no alimentan la luminaria en esos instantes que la luz natural mantiene la habitación por encima del punto de operación.

Con el entrenamiento debido cada ambiente podría tener sus valores estándares y de ahí comenzaría un aprendizaje propia del usuario, además deberemos de considerar los puntos de operación fijados por las normas como mínimos y no como máximos debido a que cada persona puede fijar su máximo de acuerdo a su necesidad, considerando nuevamente que se requiere diferentes niveles de iluminación conforme se avanza en la edad.

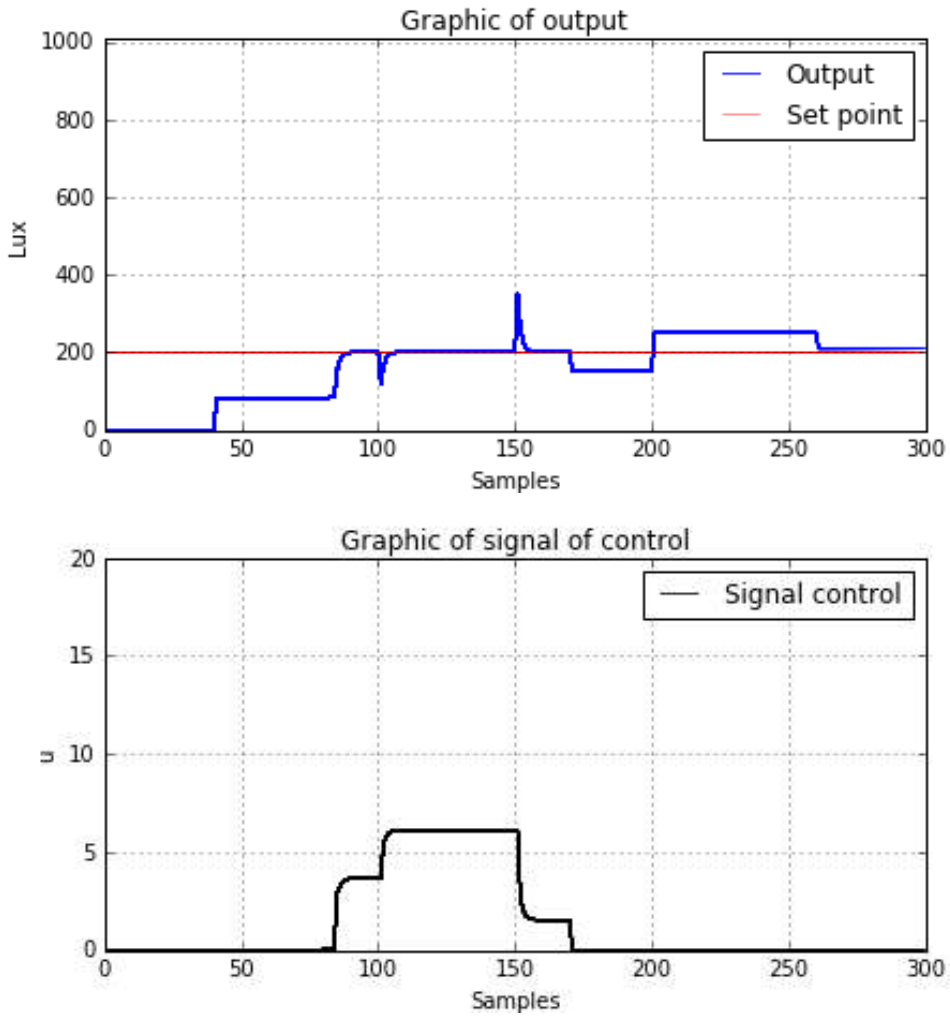


Figura 58.- Prueba 2: Respuesta del PID adaptativo con identificación

Fuente: Propia (Elaborado con Python)

La figura 58 nos muestra los resultados del PID adaptativo con identificación con el factor $B = 1.2$ este factor no depende del error, como si depende el sistema de control PID adaptativo sin identificación que se observa en la figura 58 en este caso el factor que se ha usado fue de $\gamma = 5.10^{-13}$ como se ha mencionado antes se debe considerar a menores errores un mayor factor γ, son solo estos factores que se sintonizan en estos

tipos de control y no responden más que al criterio del tiempo de respuesta, sin requerir mayor conocimiento del proceso o del controlador.

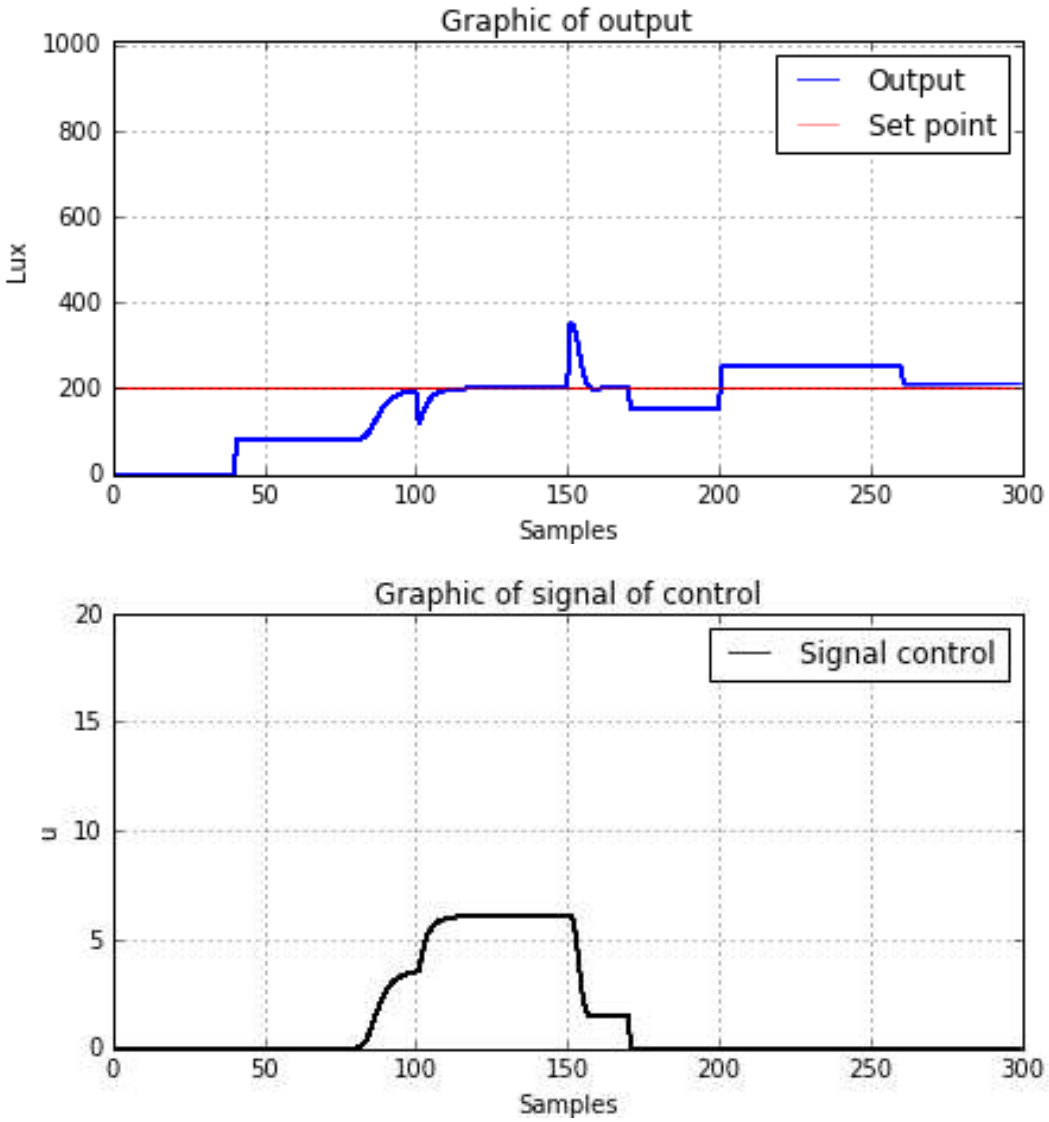


Figura 59.- Prueba 2: Respuesta del PID adaptativo sin identificación
Fuente: Propia (Elaborado con Python)

4.5. Comparación entre ambos controladores

Como se ha ido viendo en el capítulo ya se tienen ambos controladores implementados en un simulador, lo que nos va permitir compararlos de acuerdo a las pruebas 1 y 2 vistas en el apartado anterior.

El objetivo final de la tesis es implementar un algoritmo de control adaptativo en un sistema embebido y para ello se requiere considerar el menor uso de tiempo en el procesamiento, es por ello que se va medir el tiempo entre cada acción de control de ambos algoritmos el PID adaptativo con identificación y sin identificación.

Tal como se ha observado en las figuras anteriores ambos sistemas son estables frente a los cambios realizados, y en la figura 60 y 61 se muestra una comparativa entre ambos sistemas en las pruebas antes descritas.

Para la prueba 1 se ha considerado que el ambiente siempre ha estado con personas.

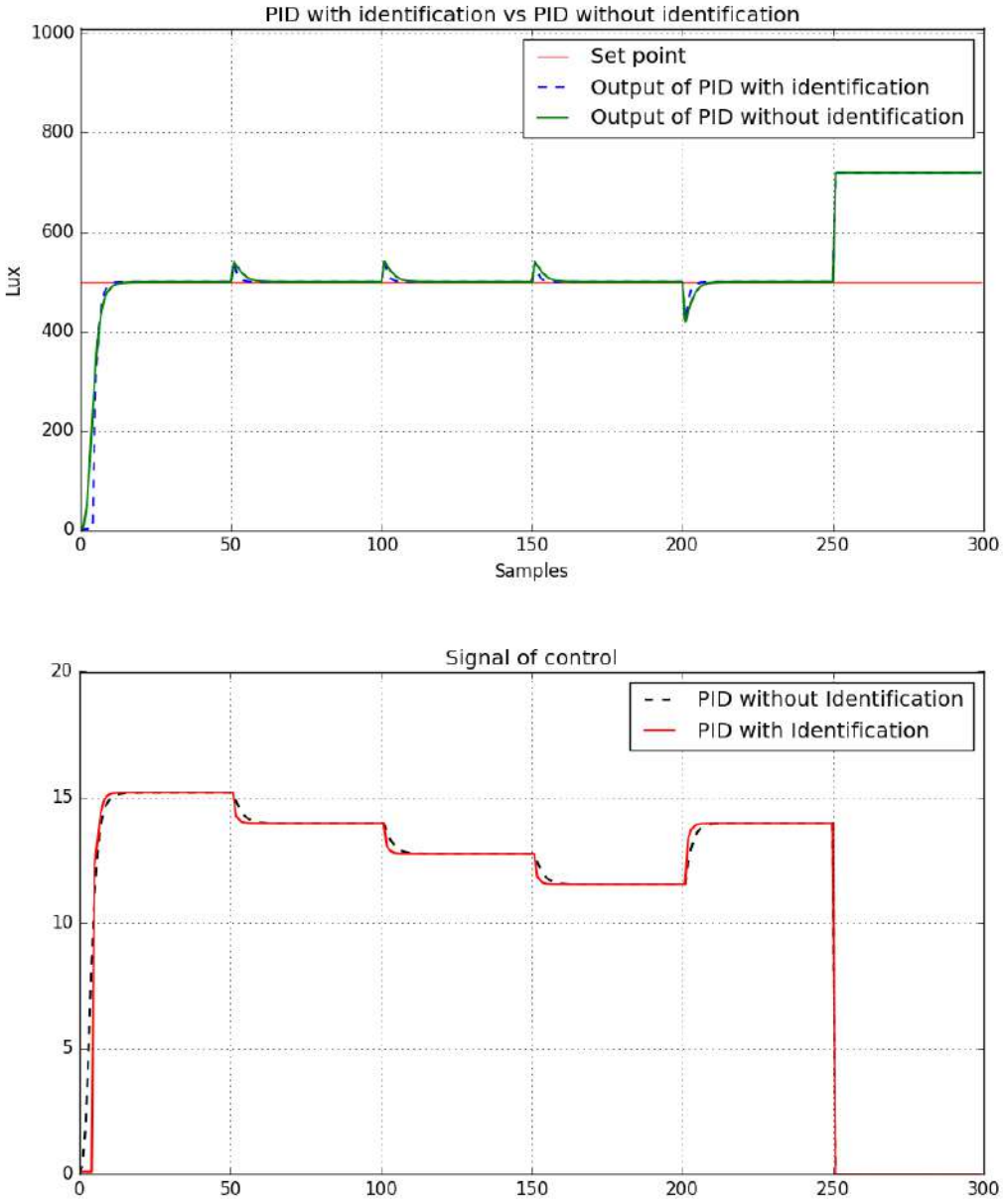


Figura 60.- Comparativa de sistemas de control en la prueba 1
Fuente: Propia (Elaborado con Python)

En la figura 61 se puede ver los tiempos en segundos consumidos por cada proceso de control, el promedio para un control PID adaptativo con identificación es de 70 milisegundos y para un control PID sin identificación es de 68 milisegundos, si bien la diferencia es aproximadamente de solo 3% esta velocidad de procesamiento en un mini Arduino podría ser muy significativo, para la enorme cantidad de veces que el proceso de control ocurre en cada segundo.

Es necesario también analizar el número de variables a declarar en cada caso, para una identificación se requiere declarar 3 matrices y 10 variables todos decimales, en contra parte para un control sin identificación sólo se requiere 7 variables decimales, por tanto bajo este punto de vista también es una mejor opción el control adaptativo sin identificación.

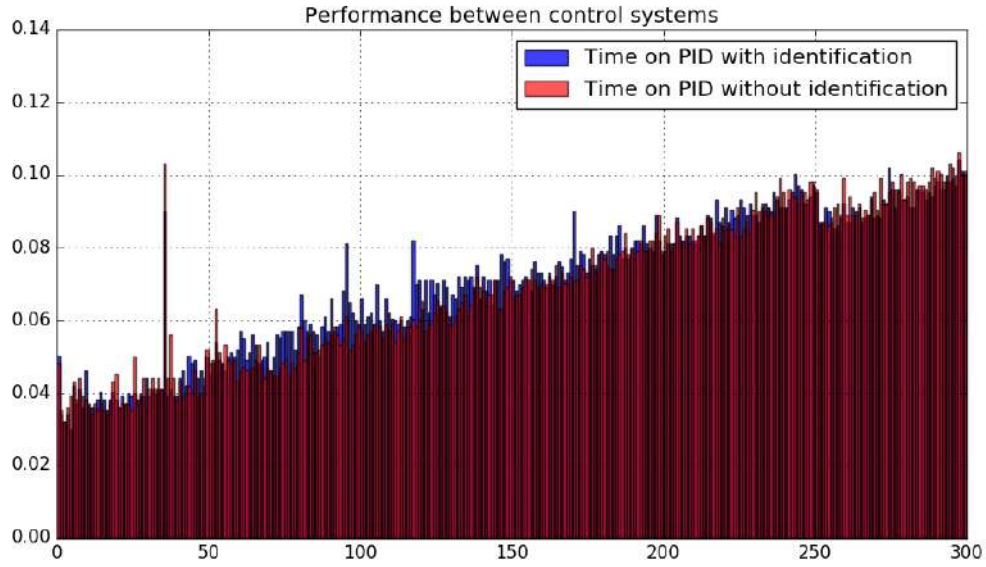
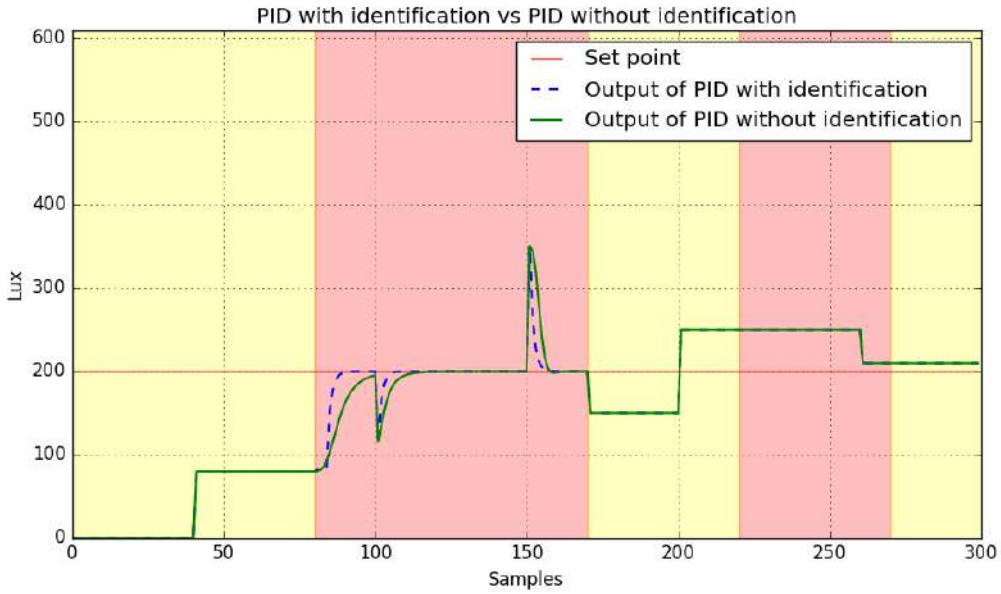


Figura 61.- Rendimiento de los sistemas de control adaptativos en la prueba 1
Fuente: Propia (Elaborado con Python)

En la figura 62 se puede apreciar la comparación entre los sistemas de control adaptativos en la prueba 2, en esta prueba se ha tratado de simular el ingreso y salida de las personas, y en la figura las áreas sombradas con amarillo como escenarios donde no había personas, y las partes sombreadas de rojo son escenarios donde había personas en el ambiente.



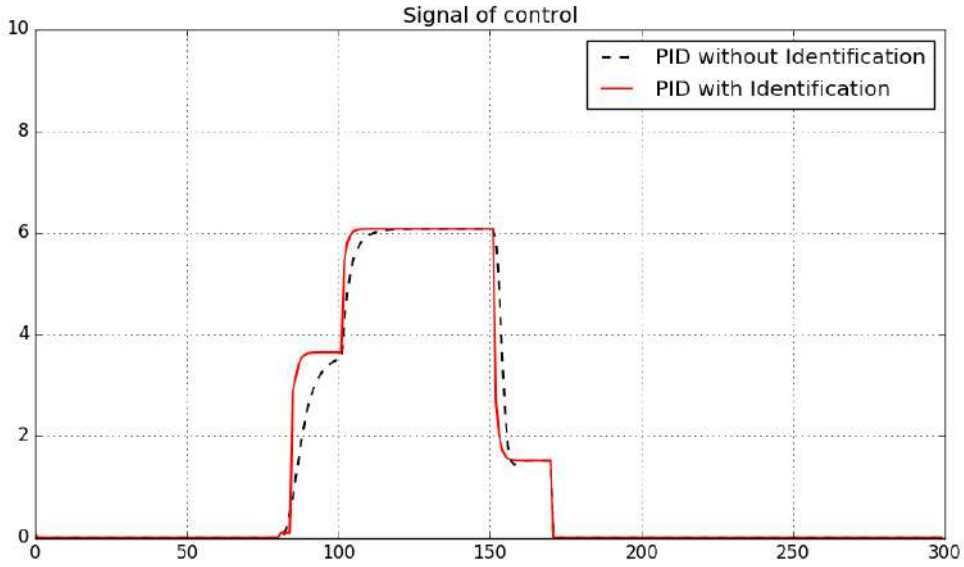


Figura 62.- Comparativa de sistemas de control en la prueba 2
Fuente: Propia (Elaborado con Python)

Como en la prueba 1 se observa la estabilidad de ambos sistemas sin embargo en el caso del sistema de control adaptativo sin identificación se observa un sistema algo más lento pero igual de estable.

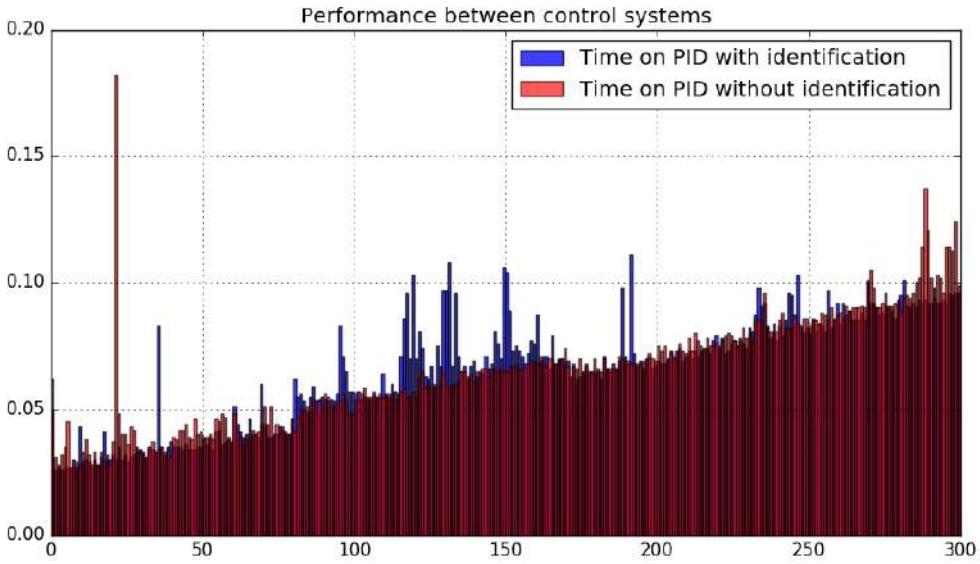


Figura 63.- Rendimiento de los sistemas de control adaptativos en la prueba 2
Fuente: Propia (Elaborado con Python)

En este caso el tiempo promedio que toma cada acción de control del sistema con identificación es de 65 milisegundos, mientras que para un sistema de control sin identificación es de 64 milisegundos, nuevamente el sistema de control con identificación es relativamente más lento, esto por la cantidad de procesamiento que hay que hacer en cada iteración, primero la identificación y luego el control, así también como fue estudiado por Gundogdu & Komurgoz, (2013) el sistema de control

adaptativo sin identificación tiene poco error y es muy rápido, coinciendo con lo visto en la presente tesis.

Por tanto se va considerar un diseño para el sistema embebido un mini Arduino Pro, en base a un sistema de autoaprendizaje el árbol de decisión y como sistema de control un controlador PID sin identificación; el árbol de decisiones resulta más sencillo de implementar que una red neuronal y por lo visto hasta el momento ambos tienen un comportamiento similar que es lo que va determinar el uso de este sistema de aprendizaje automático; el sistema de control PID sin identificación también resulta más sencillo de implementar por su número de variables.

4.6. Diseño del hardware para el microcontrolador

La parte física que consta de los sensores y el controlador se ha diseñado siguiendo las pautas de las fichas técnicas de estos sensores, para la fase de potencia se ha utilizado un mosfet IRFZ44N y un driver para mosfet TC4420 de cuyas notas de aplicación (Hussain, 2002) se extrajo el circuito de potencia que se observa en la figura 64 con este circuito se realiza el control PWM desde los pines PWM del Arduino.

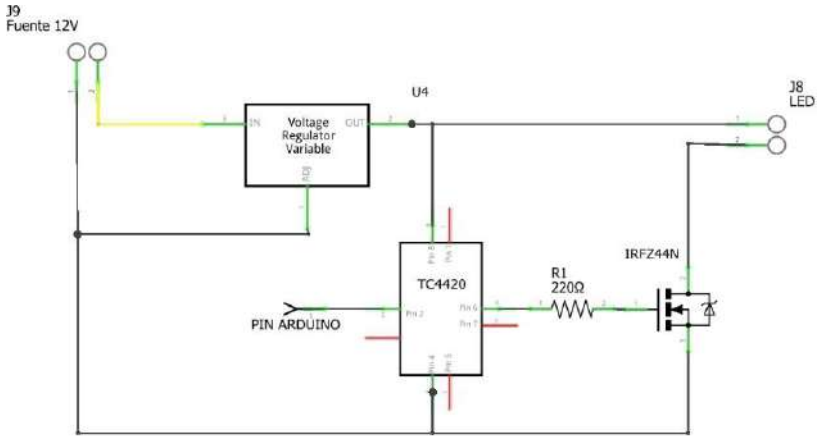


Figura 64.- Fase de potencia del circuito

Fuente: Propia (Elaborado con Fritzing)

El sistema tiene las siguientes entradas:

- Sensor de presencia (D2).
- Luxómetro (I2C).
- Interruptor (D7).

Salidas:

- Relevador (D6)
- Salida PWM al led (D10)
- Salidas de comunicación serial

El diseño presentado en el Anexo D, requiere de dos fuentes de voltaje una de 5 volteos para conectar el mini Arduino Pro y otra de 12 volteos para conectar el led, el sensor de presencia deberá ser instalado en un punto donde detecte el movimiento general de quien ocupe el ambiente y el luxómetro será conectado cerca al centro de la luminaria, si consideramos una rejilla para luminarias donde se instalan dos tubos fluorescentes, el sensor deberá ser ubicado al centro de estas como se observa en la figura 65.

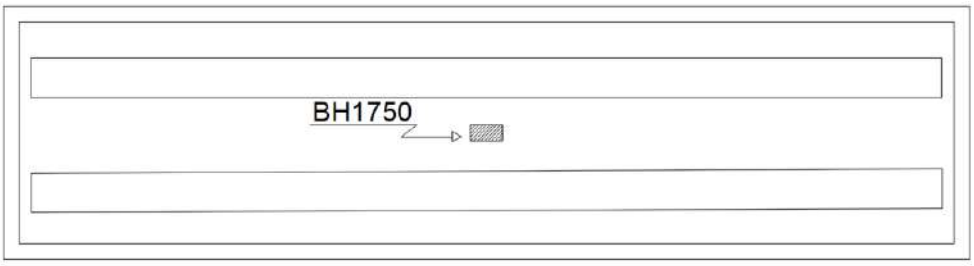


Figura 65.- Ubicación del luxómetro BH1750
Fuente: Propia (Elaborado en Paint)

4.7. Diseño del software para el microcontrolador

Considerando que tanto el sistema de autoaprendizaje y el sistema de control se han especificado como un sistemas de autoaprendizaje basado en árbol de decisión y control PID adaptativo sin identificación; en base al diagrama de flujo de la figura 40, se ha desarrollado un programa en Arduino para la aplicación de este sistema de control inteligente, cuyo código fuente se encuentra en el Anexo D, incluyendo el esquemático de componentes. El Arduino podrá ir conectado al puerto serial del Odroid C2+ para a través de este punto de comunicación el Arduino pasaría los datos obtenidos para el aprendizaje y el Odroid tras un ciclo de entrenamiento entregará el valor promedio aprendido, si un promedio es menor que lo sugerido en normas, se designará lo consignado en normas para no incumplir con estás, sin embargo si es superior se guardará dicho valor, esto debido a que el usuario requiere mayor iluminación que lo que dice las normas y esto podría deberse a la edad u otros factores.

Es posible darle independencia al circuito de la SBC Odroid respecto al aprendizaje si se guarda en memoria los últimos 10 valores del aprendizaje y se opera de acuerdo al diagrama de flujo de la figura 66.

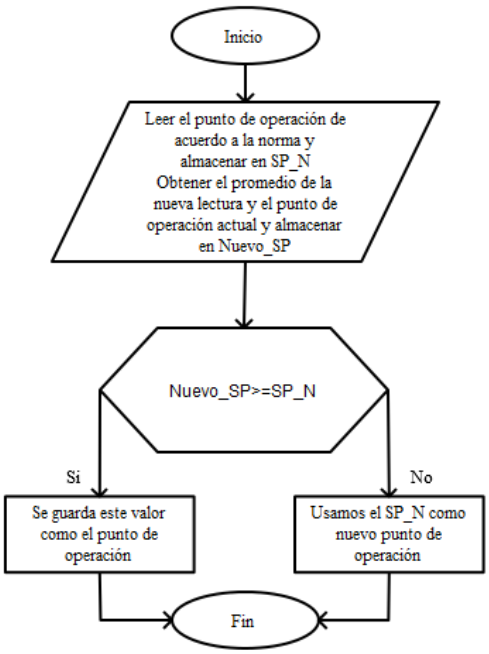


Figura 66.- Diagrama de Aprendizaje en el microcontrolador
Fuente: Propia (Elaborado en Pencil)

Por ejemplo consideremos los datos de la tabla 11, con los cuales el sistema de autoaprendizaje nos dio un primer promedio de 152.8 que está debajo de lo que establece la norma para una habitación en la parte de la cabecera de la cama que por reglamento debe ser 200 lux, por tanto el punto de operación sería 200 luxes.

Tabla 11.- Datos de ejemplo para el aprendizaje desde el microcontrolador

Sensor de Presencia	Luxómetro	Decisión Aprendida
1	78	1
1	250	0
0	350	0
1	0	1
0	150	0
1	150	1
1	40	0
1	220	1
1	240	0
1	50	1

Fuente: Propia

En un primer caso consideremos que un usuario ingresa cuando el nivel de iluminación es de 220 luxes pero el sistema se mantiene apagado, entonces el usuario va y enciende la luminaria, en este momento se promedia la nueva medida y el punto de operación obteniendo 210 luxes que sería el nuevo punto de operación.
En un segundo caso el usuario entra a la habitación cuando la iluminación es de 180 luxes, al entrar la luminaria se enciende, pero el usuario decide apagarlo, por tanto el sistema de aprendizaje ahora tratara de cambiar el valor a 195 luxes que está por debajo de lo requerido por norma, por tanto asignamos el punto de operación en el mínimo de la norma.

4.7.1. Programación por subprocesos

La programación se ha desarrollado por subprocesos paralelos controlados por interrupciones internas del Arduino, quiere decir que a pesar de tener un solo procesador se han creado 4 subprocesos que van ocupando el procesador de acuerdo a un tiempo de que los controla, es así como la toma de datos del sistema ocurre cada 120 milisegundos aproximadamente y el subproceso de control cada 130 milisegundos y la decisión del sistemas de autoaprendizaje cada 1 segundo, estos subprocesos se pueden observar en el diagrama de flujo de la figura 63, el flujo de control de los procesos se realiza utilizando la interrupción por desbordamiento del timer, que es un contador interno con él que cuentan los microprocesadores, con esta técnica se evita la utilización de retados en la programación, el diagrama de flujo de la interrupción se puede observar en la figura 67.

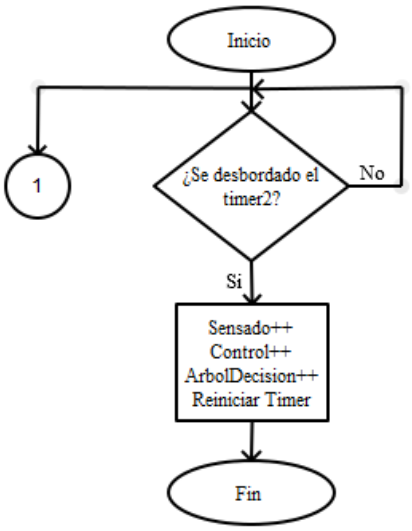


Figura 67.- Diagrama de flujo de la interrupción del timer 2
Fuente: Propia (Elaborado con Pencil)

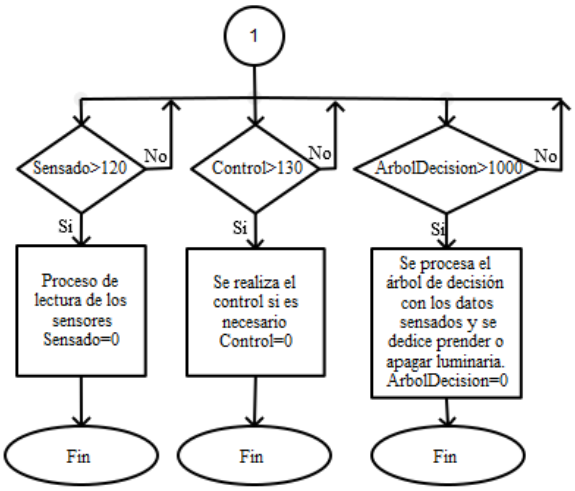


Figura 68.- Diagrama de flujo de subprocesos
Fuente: Propia (Elaborado con Pencil)

Como se observa en la figura 68, se puede ir creando subprocesos que van a funcionar en tiempos determinados y la sincronización de estos subprocesos depende del funcionamiento general.

En nuestro sistema el subproceso que debe ocurrir con mayor frecuencia es la lectura de los sensores, y con datos de lectura se pasa a controlar, acción de decisión por parte del sistema de autoaprendizaje de encender y apagar la luminaria debe ocurrir un número de veces menor para darle tiempo al control de llegar al punto de operación, para considerar en el tiempo de configuración o comunicación con el Odroid se puede detener estos subprocesos reiniciando los valores de control y pausando las interrupciones.

Capítulo 5
Pruebas y resultados

5.1. Introducción

En el presente capitulo se muestran las pruebas realizados en un módulo de control de iluminación que se ha creado para ese fin además se ha creado otro software en Python para la conexión y toma de datos del sistema en tiempo real.
Se han realizado pruebas de la lectura del sensor para la verificación de los modos con los que cuenta el sensor, y luego se ha realizado las pruebas considerando el punto de operación de 500 luxes que es el mínimo en oficinas.

5.2. Módulo de control de iluminación

Para tener un correcto lugar de ensayos de los modelos de control, se ha desarrollado un módulo de pruebas que consta:

- 2 Tiras Led de 12 v, de 1 metro cada una.
- Mini Arduino Atmega 328
- Controlador PWM
- 1 Sensor de Presencia
- 1 Relevador de 5 v
- 1 Luxómetro – BH1750

Las imágenes del módulo creado se pueden observar en la figura 69, no es parte del módulo pero también se puede observar en la imagen de la Vista superior un vatímetro con el cual se han medido los consumos para del sistema.

a) Vista Superior del módulo

b) Vista Interior del módulo

Figura 69.- Modulo de pruebas
Fuente: Propia

5.3. Software de toma de datos

En la figura 70 vemos la pantalla principal del capturador de datos, que cuenta con dos espacios para graficar la salida del sistema de acuerdo al punto de operación y la salida del controlador.

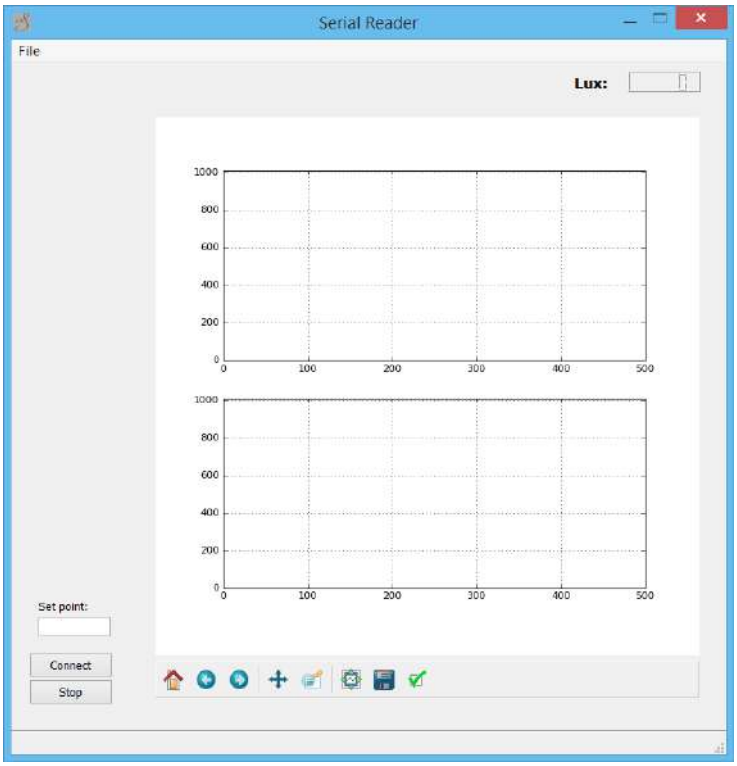


Figura 70.- Software de adquisición de datos
Fuente: Propia (Elaborado en Python)

Es posible asignar un nuevo punto de operación desde el cuadro de texto colocado para ese fin.

5.4. Pruebas del sensor de iluminancia

El sensor BH1750 tienes 2 modos de funcionamiento, en el modo “L” lee en baja resolución pero a una velocidad de 16 ms, esta fue usada para la identificación en el

programa del Anexo C, en una primera prueba se usó este mismo modo para el control, sin embargo como se puede observar en la figura 71 este modo ingresa mucho ruido al sistema, en la identificación no representaba un problema debido a que el estimador considera que el ruido tiene un promedio igual a 0, y no afecta en la identificación, sin embargo en el control si tiene efecto aunque debido al robusto sistema de control PID adaptativo no muestra cambios significativos de la señal de control.

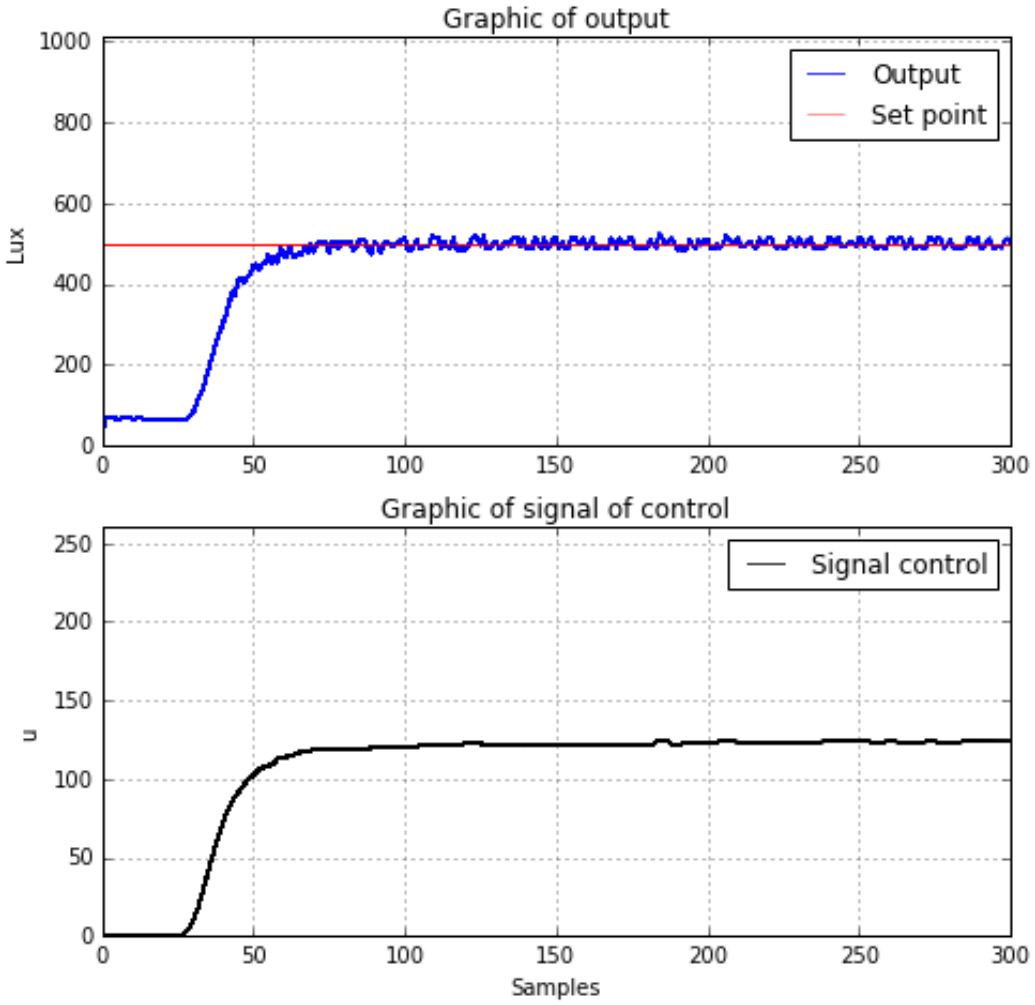


Figura 71.- Funcionamiento del sensor en modo L
Fuente: Propia (Elaborado con Python)

Un segundo modo llamado modo “H” del sensor BH1750 es el de lectura cada 120 ms pero con una mayor resolución y una señal muy limpia de salida, esto también nos obliga a cambiar los tiempos de control, pero tras la modificación requerida el sistema funciona como se muestra en la figura 72, con una salida y señal mucho más estable, evitando cualquier incomodidad a la vista del usuario.

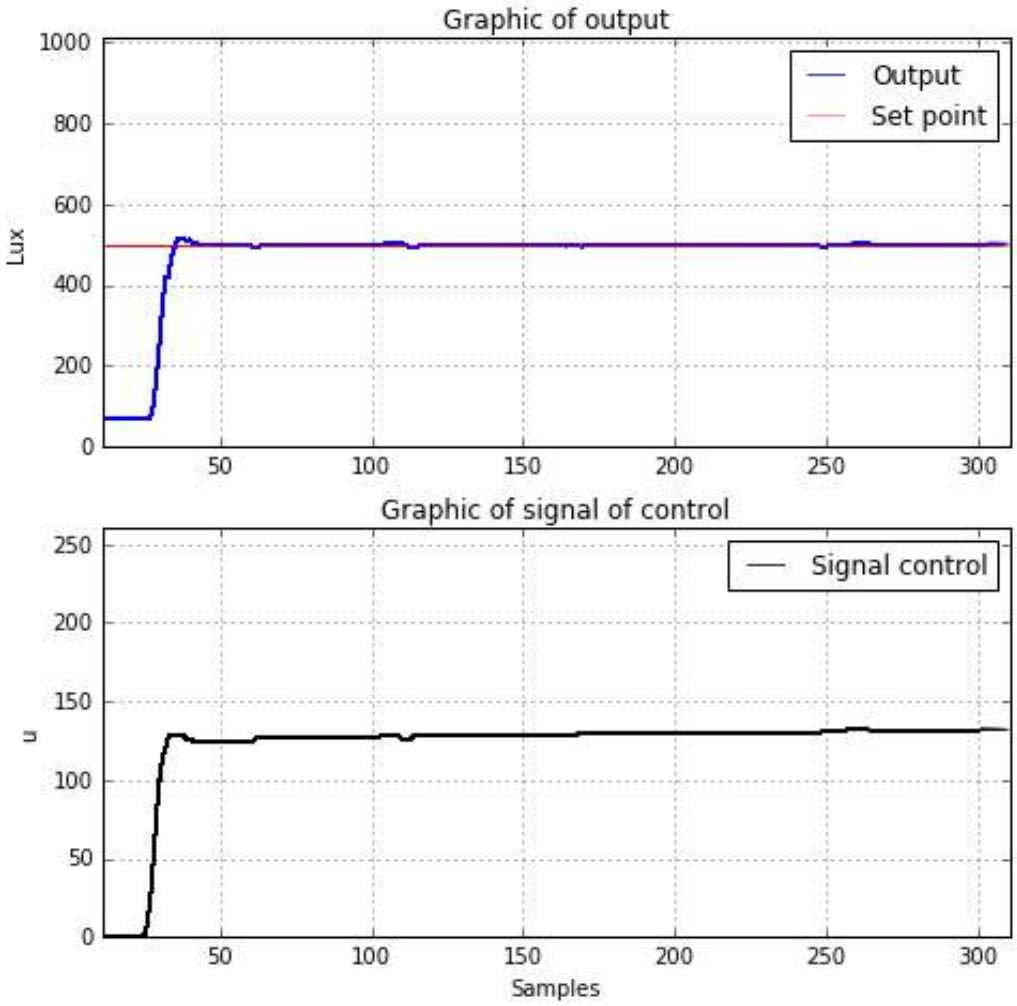


Figura 72.- Funcionamiento del sensor en modo H

Fuente: Propia (Elaborado con Python)

5.5. Pruebas de variación de factor gamma

El único factor a sintonizar es el factor γ, para la realización de pruebas se han considerado valores de $\gamma = 10^{-13}$, $\gamma = 10^{-12}$, $\gamma = 10^{-11}$ y $\gamma = 4.10^{-12}$. El programa del Arduino conectado al Odroid da la opción de establecer un nuevo valor de γ para sintonizar el controlador en cada nueva instalación que se realice.

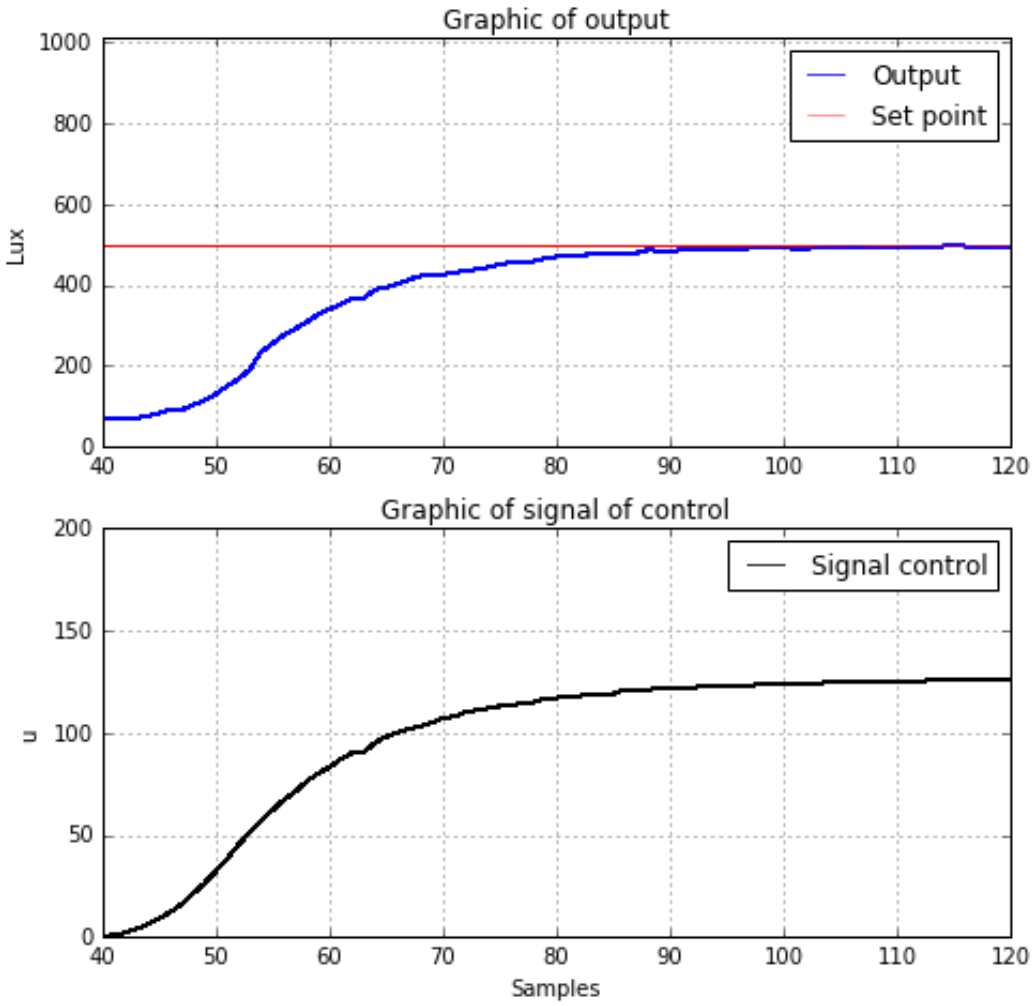


Figura 73.- Prueba con $\gamma = 10^{-13}$

Fuente: Propia (Elaborado con Python)

El número de tiempos de muestreo que demora el sistema es de más de 50 ciclos, entonces todavía es se puede ir aumentando el valor de γ.

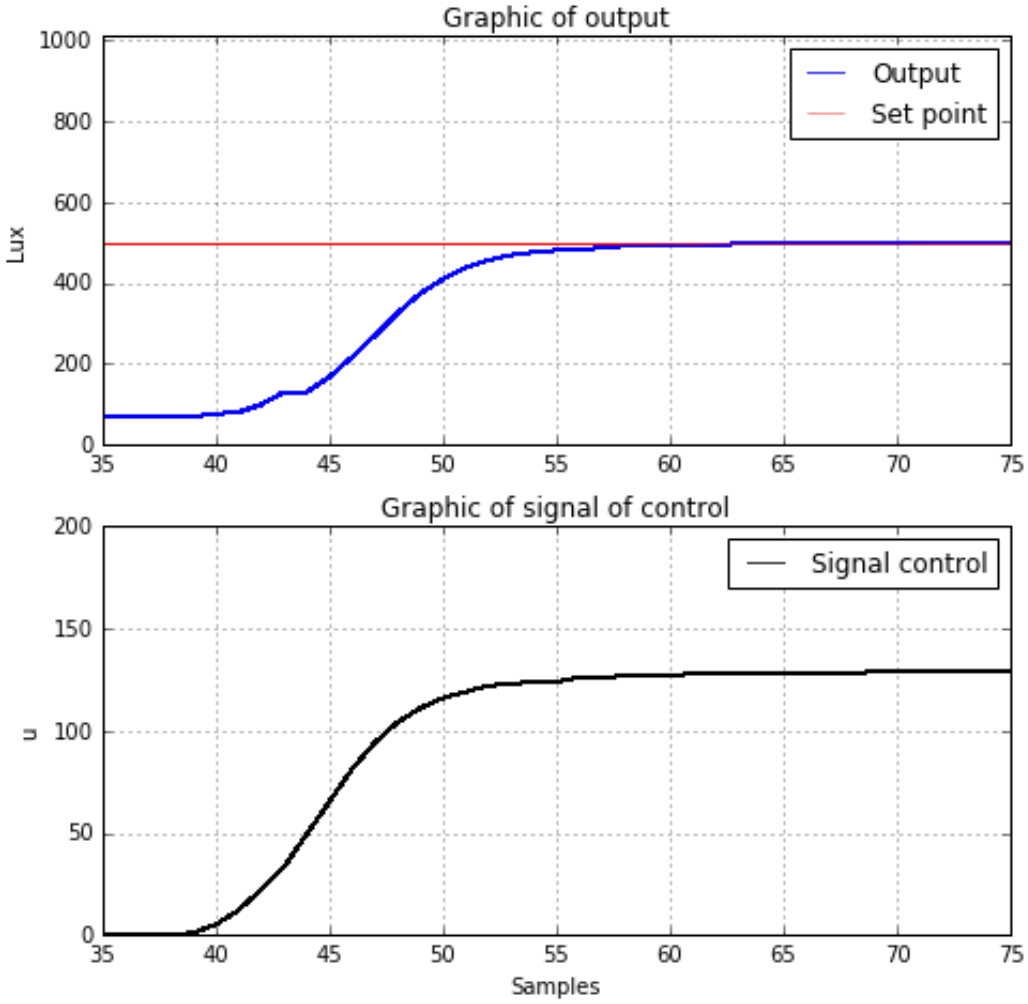


Figura 74.- Prueba con $\gamma = 10^{-12}$

Fuente: Propia (Elaborado con Python)

En este caso el tiempo que demora el sistema en estabilizarse son de 20 veces el tiempo de muestreo.

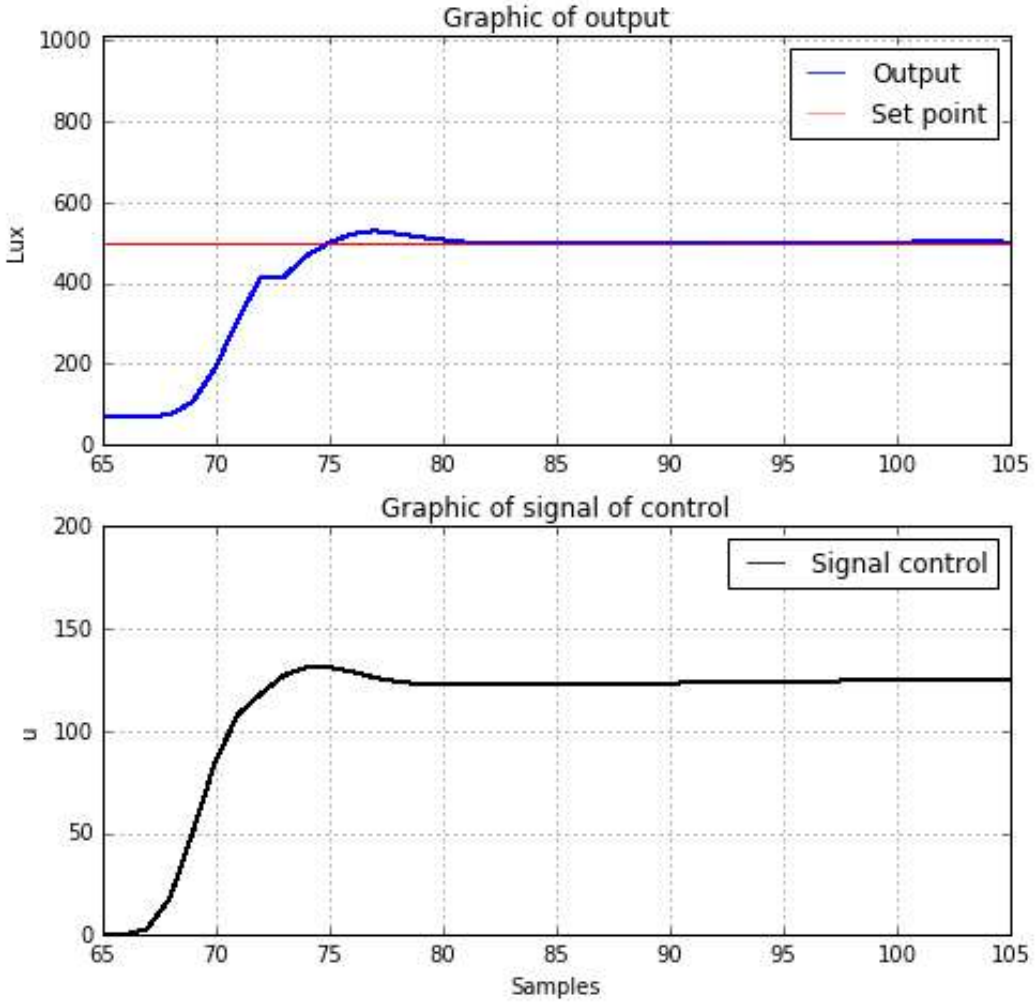


Figura 75.- Prueba con $\gamma = 10^{-11}$
Fuente: Propia (Elaborado con Python)

En este caso el sistema presenta una pico de subida para llegar al punto de operación, y el tiempo se reduce a 15 veces el tiempo de muestreo, es posible que en cambios bruscos estos picos lleguen a desestabilizar el sistema es por ello que no se va usar este factor.

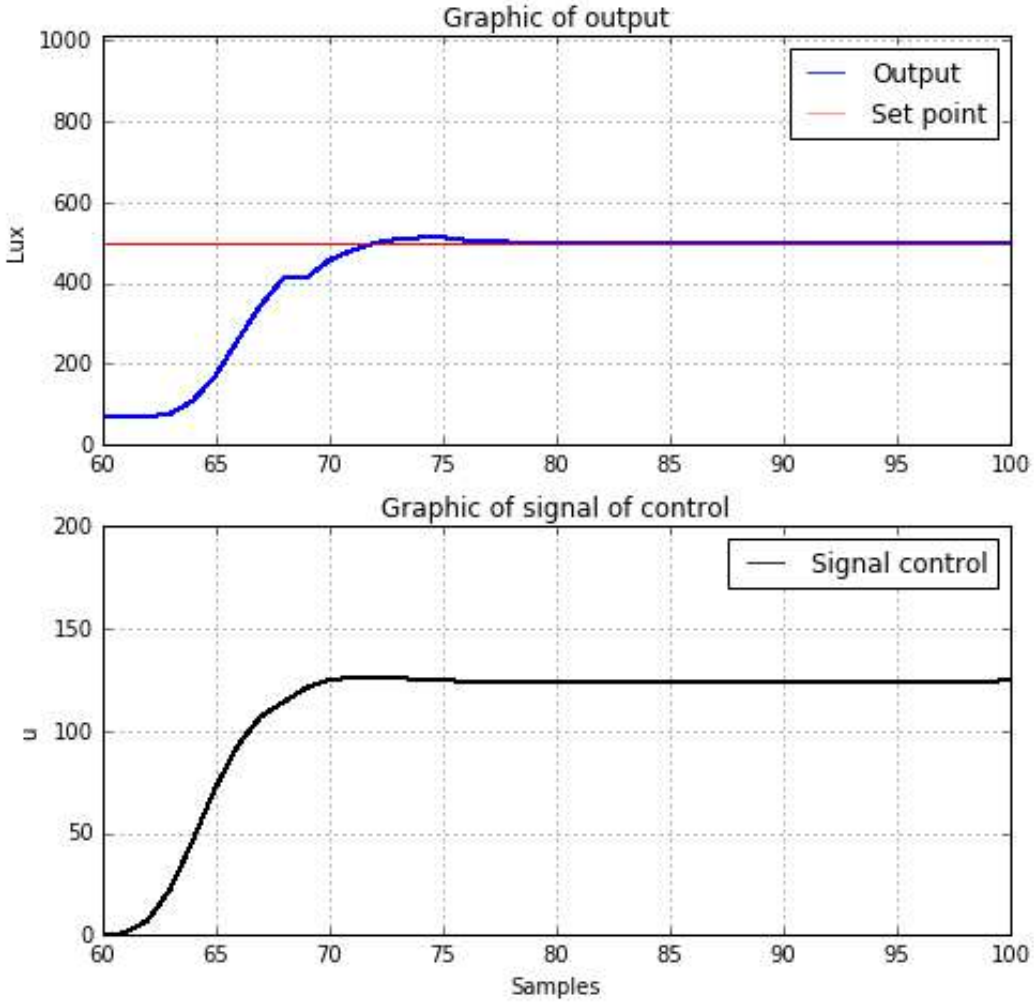


Figura 76.- Prueba con $\gamma = 4.10^{-12}$

Fuente: Propia (Elaborado con Python)

Con un factor intermedio logramos una velocidad de 12 veces el tiempo de muestreo y un muy pequeño pico, que no traería dificultados.

En la figura 76 se muestra una simulación en un tiempo mayor con un factor de $\gamma = 4.10^{-12}$.

5.6. Pruebas de variación del punto de operación

Según la Norma EM 010 del Reglamento Nacional de Edificaciones (Anexo A) los ambiente como la cocina del hogar en el espacio general debe contar con 300 luxes, la cabecera de la cama del dormitorio debe contar con 200 luxes y una oficina de uso general debería contar con 500 luxes, estos valores serán los puntos de operación del sistema a simular, considere que el modulo está en un ambiente donde existen otras fuentes de iluminación.

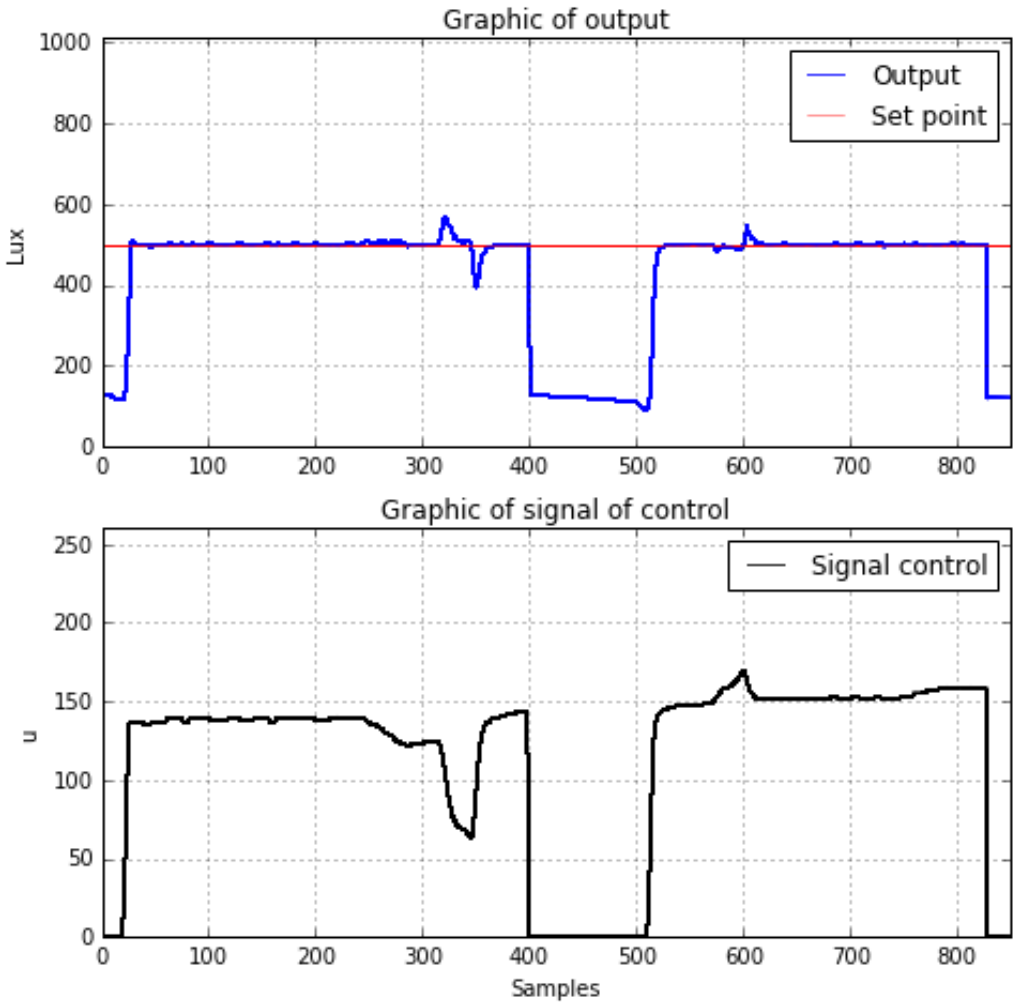


Figura 77.- Prueba con punto de operación en 500 luxes

Fuente: Propia (Elaborado con Python)

En este caso se estado probando ingresar diversas fuentes de iluminación externa y esperando su respuesta del sistema, en el instante 300 se ha puesto una luminaria externa, entonces la señal de control ha bajado para mantener el punto de operación, también se ha cubierto un poco el sensor de iluminación para de ese modo simular una falta de luz como se observa desde el instante 560 hasta el 605, el sistema intenta llegar al punto de operación y luego cuando la obstrucción es retirada el control vuelve a ser estable. En los instante 400 y 810 la señal de control es 0 debido a que en esos momentos el árbol de decisión no ha visto necesario encender la luminaria.

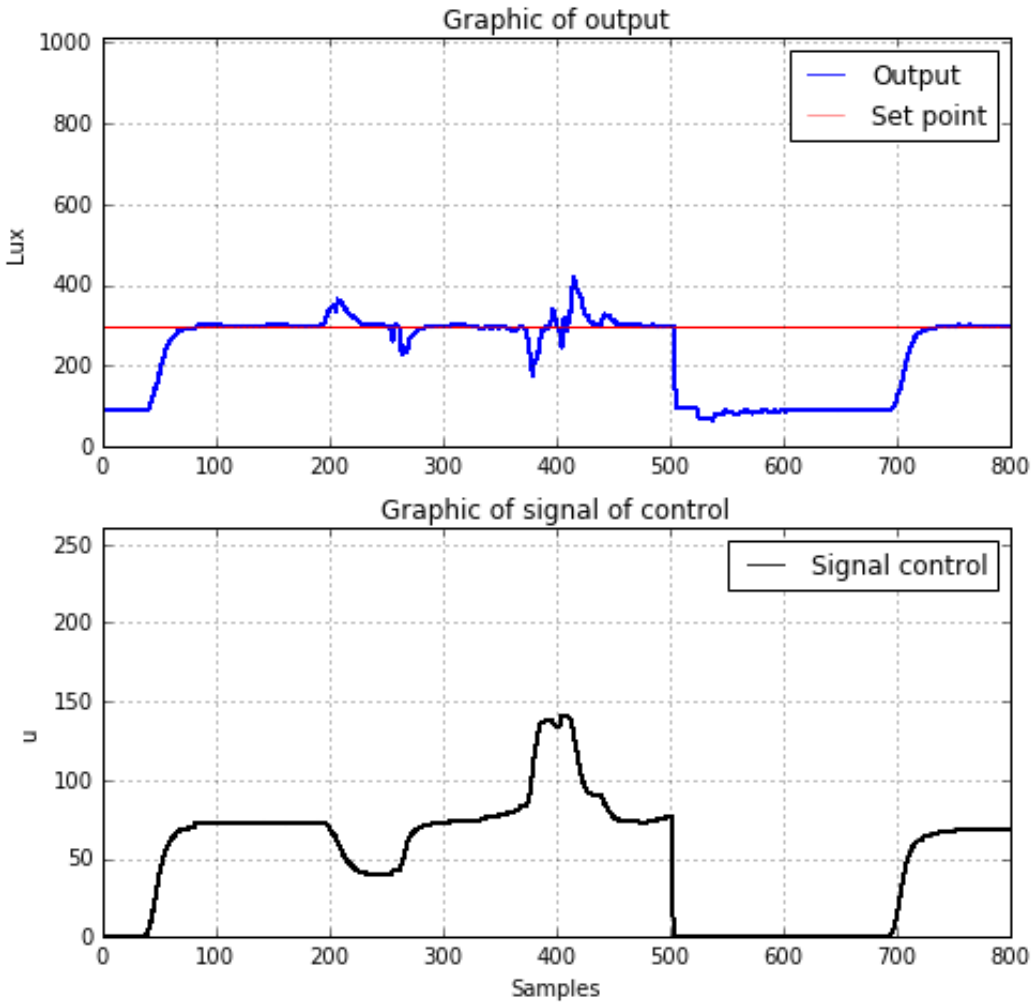


Figura 78.- Prueba con punto de operación en 300 luxes

En este caso podemos observar variaciones debido a ingreso de fuentes externas, es necesario mencionar que estas fuentes externas o disturbios no requieren ser medidos para ser controlados automáticamente por el sistema, en el instante de 380 a 450 se ve un cambio en el ambiente es decir se ha oscurecido es por eso que el sistema debe aportar mayor iluminación.

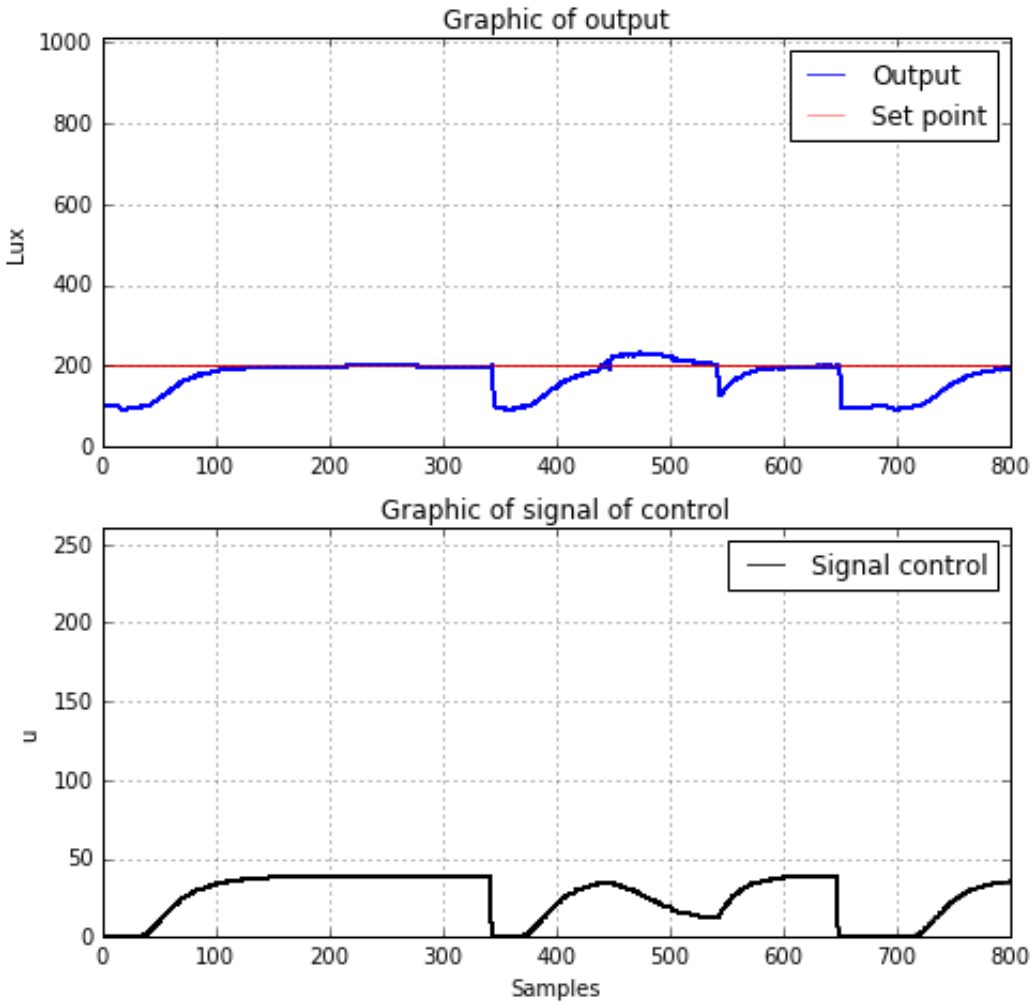


Figura 79.- Prueba con punto de operación en 200 luxes

Considerando la iluminación en la cabecera de un dormitorio que deben ser como mínimo de 200 luxes, en este caso como se ha mantenido el mismo factor gamma para todas las pruebas y considerando que en este caso los errores son menores el sistema demora más en llegar al punto de operación, pero nunca pierde el control del sistema, como se observa en la figura 79.

En esta prueba se ingresa una iluminación externa entre aproximadamente el tiempo 440 y se retira en el tiempo 500, y el sistema responde a todos esos cambios, los 200 luxes serían por ejemplo en la cabecera de una cama de habitación que el sistema lo mantiene lo más estable posible y cercano siempre al punto de operación establecido por normas o aprendido.

Conclusiones

- Los sistemas de iluminación deben basarse como mínimo en luminarias ahorradoras con arrancadores electrónicos, desechando los balastos y sus arrancadores que como se ha visto en la tesis consumen más de doble de energía eléctrica en comparación con la tecnología led.

- Tras el estudio de las tecnologías actuales de iluminación se ha llegado a la conclusión que los sistemas de iluminación basados en tecnologías led son las más eficientes y adaptables para un control automático.

- Los sistemas de aprendizaje automático basados en redes neuronales artificiales y árboles de decisión funcionando como clasificadores, son ideales para aprender y emular acciones de cuando encender una luminaria en un ambiente específico. Estos sistemas de aprendizaje automático tienen guardada la información de los valores mínimos específicos que debe cumplir cada ambiente de acuerdo a su función y también si el usuario así lo requiere estos sistemas aprenden automáticamente una mayor necesidad de iluminación, como es la que se requiere de acuerdo a como a la edad de las personas.

- El sistema de identificación en tiempo real para sistemas variables en el tiempo propuesto por el Dr. Robin De Keyser nos entrega unos coeficientes que se van amoldando al sistema de acuerdo el sistema de identificación siga conectado al sistema.

- Los sistemas de identificación deben tener una frecuencia de muestreo adecuada dependiendo del sistema, en el sistema de la presente tesis se ha identificado considerando una frecuencia de 62.5 Hz que es la máxima frecuencia que nos entrega el luxómetro BH1750.

- Los sistemas de control PID adaptativos son rápidos además de sencillos de sintonizar al tener en general un único factor de sintonía, es así como un control PID adaptativo sin identificación solo requiere un factor de sintonía, siendo este del orden de 10^{-11} en caso de sistemas con errores menores a 150 unidades para llegar al estacionario; y siendo un factor del orden de 10^{-13} en presencia de errores mayores a 150 unidades.
  Y en caso de controladores PID adaptativos con identificación también es necesario considerar y sintonizar el factor de olvido del sistema de identificación, el tiempo

de muestreo de estos sistemas y por último el factor del algoritmo que de acuerdo a la bibliografía a valores menores más rápida la respuesta del controlador.

- Para trasladar todo el sistema de control de la simulación en un prototipo se ha seleccionado el sistema de aprendizaje automático conocido como árbol de decisión, y debido a que un sistema de control PID con identificación requiere sintonizar muchos factores y una carga computacional mayor, se ha seleccionado el controlador PID adaptativo sin identificación para su implementación en un sistema embebido.

- Se puede sintetizar un controlador PID adaptativo sin identificación en un sistema embebido para su instalación en cualquier ambiente, considerando los sensores de presencia y el luxómetro BH1750.

- Python ha demostrado ser un completo lenguaje de programación orientado a objetos y de fácil aplicación en investigaciones científicas contando con librerías de aprendizaje automático, simulación de sistemas LTI (Lineales e invariables en el tiempo), dando un entorno sencillo y práctico, con la posibilidad de agregar interfaces gráficas de usuario (GUI), tal como se mostro en la presente tesis.

Referencia Bibliográfica

Adafruit Industries. (2016). *PIR Motion Sensor* (Technical note). Recuperado a partir de https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor

An, S., Yuan, S., & Li, H. (2016). Self-tuning of PID controllers design by adaptive interaction for quadrotor UAV. En *Guidance, Navigation and Control Conference (CGNCC), 2016 IEEE Chinese* (pp. 1547–1552). IEEE.

Arduino.cc. (2010). Arduino - PWM. *Tutorial*. Recuperado a partir de https://www.arduino.cc/en/Tutorial/PWM

Åström, K. J., Hägglund, T., Dormido, S., & Guzmán Sánchez, J. L. (2009). *Control PID avanzado*. Madrid [etc.: Pearson Educación.

Bhardwaj, S., Ozcelebi, T., Verhoeven, R., & Lukkien, J. (2011). Smart indoor solid state lighting based on a novel illumination model and implementation. *IEEE Transactions on Consumer Electronics*, *57*(4), 1612–1621.

Bobál, V., Böhm, J., Fessl, J., & Machácek, J. (2006). *Digital Self-tuning Controllers: Algorithms, Implementation and Applications*. Springer Science & Business Media.

Dahlin, D. B. (1968). Designing and tuning digital controllers. *Instruments and Control systems*, *42*(6), 77–83.

De Keyser, R. (2012). *Recursive Parameter Estimation* (pp. 9–15). Bélgica: Ghent University, Faculty of Engineering.

Feng Lin, Brandt, R. D., & Saikalis, G. (2000). Self-tuning of PID controllers by adaptive interaction (pp. 3676–3681 vol.5). IEEE. https://doi.org/10.1109/ACC.2000.879256

Ganslandt, R., & Hofmann, H. (2009). Cómo planificar con luz. *Madrid: Erco*.

Gordon, G. (2015). *Interior lighting for designers*. John Wiley & Sons.

Gundogdu, T., & Komurgoz, G. (2013). Adaptive PID controller design by using adaptive interaction approach theory (pp. 1–5). IEEE. https://doi.org/10.1109/EPECS.2013.6713095

Hagan, M. T., Beale, M. H., De Jess, O., & Demuth, H. B. (2014). Neural network design.

Hardkernel. (2015). Odroid C2 [Blog]. Recuperado a partir de http://www.hardkernel.com/main/products/prdt_info.php?g_code=G145457216438

Heaton, J. (2015). *Deep learning and neural networks*. St. Louis, MO: Heaton Research, Inc.

Hussain, A. (2002). Driving Power MOSFETs in High-Current, Switch Mode Regulators. *Microchip Technology, Inc.* Recuperado a partir de http://www.microchip.com/wwwAppNotes/AppNotes.aspx?appnote=en011878

Ipanaqué, W. (2012). *Control Automático de Procesos: innovando procesos productivos*. Piura, Perú: CONCYTEC.

Jones, E., Oliphant, T., & Peterson, P. (2001). *SciPy: Open source scientific tools for Python*. Recuperado a partir de http://www.scipy.org/

Livingston, J. (2014). *Designing With Light: The Art, Science and Practice of Architectural Lighting Design*. John Wiley & Sons.

Mahadeokar, S., & Sardeshmukh, M. (2015). Energy efficient PWM Dimmable Smart Digital LED driver (pp. 306–311). IEEE. https://doi.org/10.1109/ICESA.2015.7503361

Ministerio de Energía y Minas. (2014). *Balance Energético 2014* (Anuario). Perú: Ministerio de Energía y Minas.

Ministerio de Energía y Minas. (2015). *Anuario Estadístico de Electricidad 2015* (Informe Estadístico Anual). Perú: Ministerio de Energía y Minas.

Müller, A. C., & Guido, S. (2017). Introduction to machine learning with Python.

Ogata, K. (2010). *Ingeniería de Control Moderna*. S.l.: PRENTICE HALL.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Philips. (2011). Eficiencia Energética en la Iluminación. En S. Julían (Ed.) (p. 2,7). Presentado en Eficiencia Energética en la Iluminación, Philips.

Ponce Cruz, P. (2010). *Inteligencia artificial: con aplicaciones a la ingeniería*. México, D.F.: Alfaomega.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Raschka, S. (2015). *Python machine learning*. Packt Publishing Ltd.

Raspberry Pi. (2014). Raspberry Pi 2 [Blog]. Recuperado a partir de https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

Ritschel, T., Dachsbacher, C., Grosch, T., & Kautz, J. (2012). The state of the art in interactive global illumination. En *Computer Graphics Forum* (Vol. 31, pp. 160–188). Wiley Online Library.

ROHM Semiconductor. (2011). *Digital 16bit Serial Output Type Ambient Light Sensor IC* (Technical note).

Rokach, L., & Maimon, O. (2015). *Data mining with decision trees: theory and applications* (Second edition). Hackensack, New Jersey: World Scientific.

Rosen, R. (2011). *Dimming Techniques for Switched-Mode LED Drivers* (Application Note). Texas: Texas Instruments.

Ruppi, K. (2014). Odroid C1 vs Raspberry Pi B. *Magazine Odroid, 12*, 15–17.

Vasca, F., & Iannelli, L. (Eds.). (2012). *Dynamics and control of switched electronic systems: advanced perspectives for modeling, simulation and control of power converters*. New York: Springer.

Yan, L., & Chen, B. (2014). New developments in dimmable LED driver controllers (pp. 92–96). IEEE. https://doi.org/10.1109/SSLCHINA.2014.7127229

Zhukov, S., Iones, A., & Kronin, G. (1998). An ambient light illumination model. En *Rendering Techniques' 98* (pp. 45–55). Springer.

ANEXOS

ANEXO A:
Norma EM 10 – Reglamento Nacional de Edificaciones – Recuperado del repositorio del Diario El Peruano

III.4. INSTALACIONES ELECTRICAS Y MECÁNICAS

NORMA EM. 010

INSTALACIONES ELÉCTRICAS INTERIORES

Artículo 1º.- GENERALIDADES

Las instalaciones eléctricas interiores están tipificadas en el Código Nacional de Electricidad y corresponde a las instalaciones que se efectúan a partir de la acometida hasta los puntos de utilización.

En términos generales comprende a las acometidas, los alimentadores, subalimentadores, tableros, sub-tableros, circuitos derivados, sistemas de protección y control, sistemas de medición y registro, sistemas de puesta a tierra y otros.

Las instalaciones eléctricas interiores deben ajustarse a lo establecido en el Código Nacional de Electricidad, siendo obligatorio el cumplimiento de todas sus prescripciones, especialmente las reglas de protección contra el riesgo eléctrico.

Artículo 2º.- ALCANCE

Las prescripciones de esta Norma son de aplicación obligatoria a todo proyecto de instalación eléctrica interior tales como: Viviendas, Locales Comerciales, Locales Industriales, Locales de Espectáculos, Centros de Reunión, Locales Hospitalarios, Educacionales, de Hospedaje, Locales para Estacionamiento de Vehículos, Playas y Edificios de Estacionamiento, Puesto de Venta de Combustible y Estaciones de Servicio.

En general en cualquier instalación interior en todo el territorio de la República.

Artículo 3º.- CÁLCULOS DE ILUMINACIÓN

En la elaboración de proyectos de instalaciones eléctricas interiores, los proyectistas están obligados a realizar cálculos de iluminación en locales tales como: Comerciales, Oficinas, Locales de Espectáculos, Aeropuertos, Puertos, Estaciones de Transporte Terrestre y Similares, Locales Deportivos, Fábricas y Talleres, Hospitales, Centros de Salud, Postas Médicas y Afines, Laboratorios, Museos y afines.

A continuación se presenta la Tabla de Iluminancias mínimas a considerar en lux, según los ambientes al interior de las edificaciones, definiendo la calidad de la iluminación según el tipo de tarea visual o actividad a realizar en dichos ambientes.

Los proyectistas deben observar las disposiciones del Código Nacional de Electricidad y las Normas DGE relacionadas a la iluminación

TABLA DE ILUMINANCIAS PARA AMBIENTES AL INTERIOR

AMBIENTES	ILUMINANCIA EN SERVICIO (lux)	CALIDAD
Áreas generales en edificios		
Pasillos, corredores	100	D - E
Baños	100	C - D
Almacenes en tiendas	100	D - E
Escaleras	150	C - D
Líneas de ensamblaje		
Trabajo pesado (ensamble de maquinarias)	300	C - D
Trabajo normal (industria liviana)	500	B - C
Trabajo fino (ensambles electrónicos)	750	A - B
Trabajo muy fino (ensamble de instrumentos)	1500	A - B
Industrias químicas y plásticos		
En procesos automáticos	150	D - E
Plantas al interior	300	C - D
Salas de laboratorios	500	C - D
Industria farmacéutica	500	C - D
Industrias del caucho	500	C - D
Inspección	750	A - B
Control de colores	1000	A - B
Fábricas de vestimenta		
Planchado	500	A - B
Costura	750	A - B
Inspección	1000	A - B
Industrias eléctricas		
Fabricación de cables	300	B - C
Bobinados	500	A - B
Ensamblaje de partes pequeñas	1000	A - B
Pruebas y ajustes	1000	A - B
Ensamble de elementos electrónicos	1500	A - B
Industrias alimentarias		
Procesos automáticos	200	D - E
Áreas de trabajo general	300	C - D
Inspección	500	A - B
Trabajos en vidrio y cerámica		
Salas de almacén	150	D - E
Áreas de mezclado y moldeo	300	C - D
Áreas de acabados manuales	300	B - C
Áreas de acabados mecánicos	500	B - C
Revisión gruesa	750	A - B
Revisión fina – Retoques	1000	A - B
Trabajos en hierro y acero		
Plantas automáticas	50	D - E
Plantas semi – automáticas	200	D - E
Zonas de trabajo manual	300	D - E
Inspección y control	500	A - B
Industrias de cuero		
Áreas de trabajo en general		
Prensado, curtiembre, costura	300	B - C
Producción de calzados	750	A - B
Control de calidad	1000	A - B
Trabajos de maquinado (forjado – torno)		
Forjado de pequeñas piezas	200	D - E
Maquinado en tornillo de banco	400	B - C
Maquinado simple en torno	750	A - B
Maquinado fino en torno e inspección de pequeñas partes	1500	A - B
Talleres de pintado		
Preparación de superficies	500	C - D
Pintado general	750	B - C
Pintado fino, acabados, control	1000	A - B
Fábricas de papel		
Procesos automáticos	200	D - E
Elaboración semi automática	300	C - D
Inspección	500	A - B
Imprentas – Construcción de libros		
Salas de impresión a máquina	500	C - D
Encuadernado	500	A - B
Composición, edición, etc.	750	A - B
Retoques	1000	A - B
Reproducciones e impresiones a color	1500	A - B
Grabados en acero y cobre	2000	A - B
Industrias textiles		
Área de desembalaje	200	D - E
Diseño	300	D - E
Hilados, cardados, teñidos	500	C - D
Hilados finos, entrelazados	750	A - B
Cosido, inspección	1000	A - B
Industrias en madera		
Aserradero	200	D - E
Ensamble en tornillo de banco	300	C - D
Trabajo con máquinas	500	B - C
Acabados	750	A - B
Inspección control calidad	1000	A - B
Oficinas		
Archivos	200	C - D
Salas de conferencia	300	A - B
Oficinas generales y salas de cómputo	500	A - B
Oficinas con trabajo intenso	750	A - B
Salas de diseño	1000	A - B
Centros de enseñanza		
Salas de lectura	300	A - B
Salones de clase, laboratorios, talleres, gimnasios	500	A - B
Tiendas		
Tiendas convencionales	300	B - C
Tiendas de autoservicio	500	B - C
Tiendas de exhibición	750	B - C
Edificios Públicos		
Salas de cine	150	B - C
Salas de conciertos y teatros	200	B - C
Museos y galerías de arte	300	B - C
Iglesias		
- nave central	100	B - C
- altar y púlpito	300	B - C

AMBIENTES	ILUMINANCIA EN SERVICIO (lux)	CALIDAD
Viviendas		
Dormitorios		
- general	50	B - C
- cabecera de cama	200	B - C
Baños		
- general	100	B - C
- área de espejo	500	B - C
Salas		
- general	100	B - C
- área de lectura	500	B - C
Salas de estar	100	B - C
Cocinas		
- general	300	B - C
- áreas de trabajo	500	B - C
Área de trabajo doméstico	300	B - C
Dormitorio de niños	100	B - C
Hoteles y restaurantes		
Comedores	200	B - C
Habitaciones y baños		
- general	100	B - C
- local	300	B - C
Áreas de recepción, salas de conferencia	300	B - C
Cocinas	500	B - C
Subestaciones eléctricas al interior		
Alumbrado general	200	B - C
Alumbrado local	500	A - B
Alumbrado de emergencia	50	B - C
Hospitales – Centros Médicos		
Corredores o pasillos		
- durante la noche	50	A - B
- durante el día	200	A - B
Salas de pacientes		
- circulación nocturna	1	A - B
- observación nocturna	5	A - B
- alumbrado general	150	A - B
- exámenes en cama	300	A - B
Salas de exámenes		
- alumbrado general	500	A - B
- iluminación local	1000	A - B
Salas de cuidados intensivos		
- cabecera de cama	50	A - B
- observación local	750	A - B
Sala de enfermeras	300	A - B
Salas de operaciones		
- sala de preparación	500	A - B
- alumbrado general	1000	A - B
- mesa de operaciones	100000	A - B
Salas de autopsias		
- alumbrado general	750	A - B
- alumbrado local	5000	A - B
Laboratorios y farmacias		
- alumbrado general	750	A - B
- alumbrado local	1000	A - B
Consultorios		
- alumbrado general	500	A - B
- alumbrado local	750	A - B

CALIDAD DE LA ILUMINACIÓN POR TIPO DE TAREA VISUAL O ACTIVIDAD

CALIDAD	TIPO DE TAREA VISUAL O ACTIVIDAD
A	Tareas visuales muy exactas
B	Tareas visuales con alta exigencia. Tareas visuales de exigencia normal y de alta concentración
C	Tareas visuales de exigencia y grado de concentración normales; y con un cierto grado de movilidad del trabajador.
D	Tareas visuales de bajo grado de exigencia y concentración, con trabajadores moviéndose frecuentemente dentro de un área específica.
E	Tareas de baja demanda visual, con trabajadores moviéndose sin restricción de área.

Artículo 4º.- EVALUACIÓN DE LA DEMANDA

Los proyectos deberán incluir un análisis de la potencia instalada y máxima demanda de potencia que requerirán las instalaciones proyectadas.

La evaluación de la demanda podrá realizarse por cualquier de los dos métodos que se describen:

Método 1. Considerando las cargas realmente a instalarse, los factores de demanda y simultaneidad que se obtendrán durante la operación de la instalación.

Método 2. Considerando las cargas unitarias y los factores de demanda que estipula el Código Nacional de Electricidad o las Normas DGE correspondientes; el factor de simultaneidad entre las cargas será asumido y justificado por el proyectista.

El valor mínimo de la demanda máxima y el tipo de suministro para la elaboración del Proyecto de Subsistema de Distribución Secundaria, que requiere una habilitación de tierras para ser dotada del servicio público de electricidad, están establecidos en la Norma DGE «Calificación Eléctrica para la Elaboración de Proyectos de Subsistemas de Distribución Secundaria».

Artículo 5º.- COMPONENTES DE UN PROYECTO DE INSTALACIÓN ELÉCTRICA INTERIOR

Para los efectos de la presente Norma se considera que un proyecto de instalación eléctrica interior consta de lo siguiente:

- Memoria Descriptiva
- Factibilidad y Punto de Entrega del Servicio Público
- Memoria de Cálculo
- Especificaciones Técnicas
- Planos
- Certificado de Habilitación de Proyectos

Memoria Descriptiva

Descripción de la naturaleza del proyecto y la concepción del diseño de cada una de las instalaciones que conforman el sistema proyectado.

Factibilidad y Punto de Entrega del Servicio Público de Electricidad

Cartas con la factibilidad y punto de entrega (suministro) para el servicio público de electricidad, otorgada por el respectivo concesionario.

Memoria de Cálculo

Descripción y formulación de los parámetros de cálculo de los diferentes diseños, complementado con las respectivas hojas de cálculo.

Especificaciones Técnicas

Descripción de las características específicas y normas de fabricación de cada uno de los materiales y/o equipos a utilizarse; así como, los métodos constructivos a seguirse.

Planos

Los planos deben ser presentados en hojas de tamaño y formatos normalizados según la NTP 272.002 y NTP 833.001, doblados al tamaño A4 conforme a la NTP 833.002 debiendo quedar a la vista el rótulo respectivo donde debe figurar el nombre completo y número de registro del Colegio de Ingenieros del Perú del Profesional Responsable (Ing. Electricista o Ing. Mecánico-Electricista); así como su firma y sello oficial.

De acuerdo a la naturaleza y magnitud del proyecto los planos pueden ser:

- Planos Generales: Para que mediante aplicación de los símbolos gráficos normalizados en electricidad se haga la distribución de las salidas, diagramas unifilares y demás elementos de los diseños del proyecto. El plano debe ser desarrollado en escala 1:50.
- Planos de Conjunto: Para identificar la posición relativa de las distintas partes y/o elementos de un sistema, que por su tamaño sea necesario hacerlo. El plano debe ser desarrollado en escala 1:100, 1:200 ó 1:500.
- Planos de Detalle: Para una mejor identificación o comprensión de algunos elementos o parte de los diseños del proyecto, tales como esquemas generales, planos isométricos etc., sean necesarios. Los detalles deben ser desarrollados en escala 1:20 ó 1.25.

Certificado de Habilitación de Proyectos

Documento emitido por el Consejo Departamental del Colegio de Ingenieros del Perú, por la que certifica que el Profesional que se menciona se encuentra hábil y esta autorizado para desarrollar un proyecto de su especialidad.

Artículo 6º.- DISEÑO DE LAS INSTALACIONES ELÉCTRICAS

El diseño de instalaciones eléctricas, deberá realizarse de acuerdo con el Código Nacional de Electricidad.

ANEXO B:

Programa: Automatic Learning

Este programa nos va presentando diversos escenarios de iluminación con un ambiente con personas o vacío, y espera una decisión de acuerdo a estos escenarios, estas decisiones y el estado de escenario lo almacena en una base de datos para luego usar esos datos para obtener los factores de aprendizaje.

```
# -*- coding: utf-8 -*-
"""
Created on Thu June 26 12:02:57 2016
@author: Ing. Huamán Rojas Jezzy James
"""
import sys
from PIL import Image, ImageQt
from PyQt4 import QtCore, QtGui
import JJutiles as Ujj
import random
from Simulador_AA import *
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
import numpy as np
from sklearn import tree
import pydotplus
from IPython.display import Image as IPImage

#Estado del aprendizaje
estadoGeneral=0 #0 - En modo aprendizaje; #1 - En modo predictor

#Datos Maximos de entrada para entrenamiento
var_senp=1
var_luxm=1000
r_senp=0
r_luxm=0
vec=0

#Salida
var_salida=0

#Base datos
bd_jj='base_'+'UDEP'

#Confiabilidad
rgMax=10
Rpta_NN=0
Rpta_Tree=0

#Red Neuronal
nn=MLPClassifier(algorithm='sgd',activation='logistic',random_state=1,
alpha=0.1,hidden_layer_sizes=[4],max_iter=1000)
arbol = DecisionTreeClassifier(criterion='entropy',max_depth=2,random_state=0)
X=[]
y=[]
X_scaled=[]
Y_scaled=[]

def Definir_Redes():
    global nn,arbol,X,y,X_scaled,Y_scaled
    ##DEFINIR RED NEURONAL
    X=Ujj.Select_Entrada(bd_jj)
    y=Ujj.Select_Salida(bd_jj)
    X_scaled=[]
    Y_scaled=[]
    ##Escalar los datos del sensor
    for i in X:
        X_fila=[]
        cont=1
        for j in i:
            if cont%2==1:
                X_fila.append(Ujj.remap(j,0,var_senp,0,rgMax))
            if cont%2==0:
                X_fila.append(Ujj.remap(j,0,var_luxm,0,rgMax))
            cont=cont+1
        X_scaled.append(X_fila)
```

```
    #Formar doble salida
    for i in y:
        Y_fila=[]
        if i==1:
            Y_fila=[1,0]
        if i==0:
            Y_fila=[0,1]
        Y_scaled.append(Y_fila)
    nn.fit(X_scaled,Y_scaled)
    print ("Los coeficientes de la red neuronal son: ")
    print (nn.coefs_    )
    ##DEFINIR ARBOL DE DECISION
    arbol.fit(X, y)

def NN_Say(sp,lx):
    sp=Ujj.remap(sp,0,var_senp,0,rgMax)
    lx=Ujj.remap(lx,0,var_luxm,0,rgMax)
    Xne=np.array([[sp,lx]])
    sadn=nn.predict(Xne)
    for i in sadn:
        bs=i
    return bs[0]

def TREE_Say(sp,lx):
    Xne=np.array([[sp,lx]])
    sadn=arbol.predict(Xne)
    for i in sadn:
        bs=i
    return bs

def Renovar_valores():
    global r_senp, r_luxm, vec, Rpta_NN, Rpta_Tree, estadoGeneral
    r_senp=random.randrange(var_senp+1)
    r_luxm=random.randrange(var_luxm+1)
    vec=Ujj.remap(r_luxm,0,var_luxm,-150,150)
    #Toma de decisión
    if estadoGeneral==1:
        Rpta_NN=NN_Say(r_senp,r_luxm)
        Rpta_Tree=TREE_Say(r_senp,r_luxm)
    return

class Principal(QtGui.QDialog):
    def __init__(self,parent=None):
        QtGui.QWidget.__init__(self,parent)
        self.ui=Ui_frm_VentanaPrincipal()
        self.ui.setupUi(self)

        QtCore.QObject.connect(self.ui.cmd_on,QtCore.SIGNAL('clicked()'),self.Comando_On)
QtCore.QObject.connect(self.ui.cmd_offQtCore.SIGNAL('clicked()'),self.Comando_Off)
QtCore.QObject.connect(self.ui.cmd_train,QtCore.SIGNAL('clicked()'),self.Entrenar)
QtCore.QObject.connect(self.ui.cmd_ange,QtCore.SIGNAL('clicked()'),self.VerImagen)
        self.Iniciacion()
        self.ui.lbl_conteo.setText("There are "+ str(Ujj.Nro_Datos(bd_jj)) +" records")
        self.VerImagen()

    def Iniciacion(self):
        global bd_jj, estadoGeneral
        name,respt=QtGui.QInputDialog.getText(None, 'Text ', 'Enter name:')
        if respt:
            bd_jj='base_'+name
            Ujj.crear_bd(bd_jj)
        if Ujj.Nro_Datos(bd_jj)>=10:
            estadoGeneral=1
            Definir_Redes()
        else:
            estadoGeneral=0

    def Entrenar(self):
        global estadoGeneral
        nro_datos=Ujj.Nro_Datos(bd_jj)
        if nro_datos<6:
            self.Mensaje_Train()
        else:
            if estadoGeneral==0:
                Definir_Redes()
                estadoGeneral=1
                self.VerImagen()
```

```
        self.Mensaje_Train_ok()

    def Mensaje_Train(self):
        QtGui.QMessageBox.information(None, QtGui.qApp.tr("Train"),
                QtGui.qApp.tr("Fall short of data""\n"
                              " ""\n"
                              "Click Ok to continue."),
                QtGui.QMessageBox.Ok)

    def Mensaje_Train_ok(self):
        QtGui.QMessageBox.information(None, QtGui.qApp.tr("Train"),
                QtGui.qApp.tr("The neural network is ready""\n"
                              " ""\n"
                              "Click Ok to continue."),
                QtGui.QMessageBox.Ok)

    def Mensaje_Train_upd(self):
        QtGui.QMessageBox.information(None, QtGui.qApp.tr("Update"),
                QtGui.qApp.tr("The neural network was update""\n"
                              " ""\n"
                              "Click Ok to continue."),
                QtGui.QMessageBox.Ok)

    def Verificar_nro_datos(self):
        global estadoGeneral
        nro_datos=Ujj.Nro_Datos(bd_jj)

        if nro_datos>5:
            self.ui.lbl_conteo.setText("You can train the neural network")

        if nro_datos>9:
            Definir_Redes()
            self.ui.lbl_conteo.setText("The neural network is ready")
            estadoGeneral=1

    def Comando_On(self):
        global estadoGeneral, Rpta_NN, Rpta_Tree
        if estadoGeneral==0:
            #Modo de aprendizaje
            if(Ujj.Nro_Datos_Si(bd_jj)<=Ujj.Nro_Datos_No(bd_jj)):
                Ujj.Insertar_dato(bd_jj,r_senp,r_luxm,1,1)
            else:
                Ujj.Insertar_dato(bd_jj,r_senp,r_luxm,1,0)
            self.Verificar_nro_datos()
        else:
            ##Modo de predicción
            if Rpta_NN==0:
                a=Ujj.Select_Positivo(bd_jj)
                print a[0]
                Ujj.Update_dato(bd_jj,r_senp,r_luxm,1,0,a[0])
                Ujj.Insertar_dato(bd_jj,r_senp,r_luxm,1,1)
                Definir_Redes()
                Rpta_NN=NN_Say(r_senp,r_luxm)
                self.Mensaje_Train_upd()
                if Rpta_NN==1:
                    self.ui.lbl_conteo.setText("The NN decided: ON")
                else:
                    self.ui.lbl_conteo.setText("The NN decided: OFF")
                return
        self.VerImagen()

    def Comando_Off(self):
        global estadoGeneral, Rpta_NN, Rpta_Tree
        if estadoGeneral==0:
            #Guardar datos en base de datos
            if(Ujj.Nro_Datos_No(bd_jj)<=Ujj.Nro_Datos_Si(bd_jj)):
                Ujj.Insertar_dato(bd_jj,r_senp,r_luxm,0,1)
            else:
                Ujj.Insertar_dato(bd_jj,r_senp,r_luxm,0,0)
            self.Verificar_nro_datos()
        else:
            ##Modo de predicción
            if Rpta_NN==1:
                self.Mensaje_Train_upd()
                a=Ujj.Select_Negativo(bd_jj)
                print a[0]
                Ujj.Update_dato(bd_jj,r_senp,r_luxm,0,0,a[0])
```

```
                Ujj.Insertar_dato(bd_jj,r_senp,r_luxm,0,1)
                Definir_Redes()
                Rpta_NN=NN_Say(r_senp,r_luxm)
                if Rpta_NN==1:
                    self.ui.lbl_conteo.setText("The NN decided: ON")
                else:
                    self.ui.lbl_conteo.setText("The NN decided: OFF")
                return
        self.VerImagen()


    def VerImagen(self):
        global estadoGeneral
        Renovar_valores()
        #Cargar label luxometro
        self.ui.lbl_luxm.setText(str(r_luxm))
        #Cargar Imagen de habitacion
        Im_habitacion = Image.open('Im/i_hab1.jpg')
        Im_hab_brillo = Im_habitacion.point(lambda x: x+vec)
        QtImage1 = ImageQt.ImageQt(Im_hab_brillo)
        QtImage2 = QtGui.QImage(QtImage1)
        pixmap = QtGui.QPixmap.fromImage(QtImage2)
        self.ui.lbl_habitacion.setPixmap(pixmap)
        self.ui.lbl_habitacion.setScaledContents(True)
        #Cargar Imagen de persona
        if r_senp==1:
            Im_persona = Image.open('Im/i_persona.jpg')
        else:
            Im_persona = Image.open('Im/i_sin_persona.jpg')
        QtImage3 = ImageQt.ImageQt(Im_persona)
        QtImage4 = QtGui.QImage(QtImage3)
        pixmap = QtGui.QPixmap.fromImage(QtImage4)
        self.ui.lbl_persona.setPixmap(pixmap)
        self.ui.lbl_persona.setScaledContents(True)
        nro_datos=Ujj.Nro_Datos(bd_jj)
        if nro_datos<6:
            self.ui.lbl_conteo.setText("There are "+ str(Ujj.Nro_Datos(bd_jj)) +" records")
        if estadoGeneral==1:
            if Rpta_NN==1:
                self.ui.lbl_conteo.setText("The NN decided: ON")
            else:
                self.ui.lbl_conteo.setText("The NN decided: OFF")

    def closeEvent(self, event):
        exit()


if __name__ == '__main__':
    app=0
    app=QtGui.QApplication(sys.argv)
    #QtGui.QImageReader.supportedImageFormats()
    myapp=Principal()
    myapp.show()
    sys.exit(app.exec_())
```

ANEXO C:

Programa: Identificación de un sistema de iluminación led controlado por señal PWM.
Este programa fue desarrollado en mi labor de asesor de un grupo del curso de Sistemas Automáticos de Control dictado por el Dr. William Ipanaqué durante el semestre académico 2016-I. Se ha usado un programa en Arduino que lee las señales del sensor y las envía al Matlab 2014 a través del puerto serial, donde se identifica a través de RLS [27].

Anexo C1: Código fuente del Arduino

```
#include <Wire.h>
#include "BH1750FVI.h"

BH1750FVI LightSensor;

int valor = 30;
int led = 9;

void setup() {
        Serial.begin(9600);
        pinMode(led, OUTPUT);
        LightSensor.begin();
        LightSensor.SetMode(Continuous_L_resolution_Mode);
        LightSensor.SetAddress(Device_Address_L);
        analogWrite(led,valor);
}

void loop() {
    uint16_t lux = LightSensor.GetLightIntensity();
    Serial.print(valor0);
    Serial.print(" ");
    Serial.println(lux);
    delay(500);
}
```

Anexo C2: Archivo .m del programa de identificación

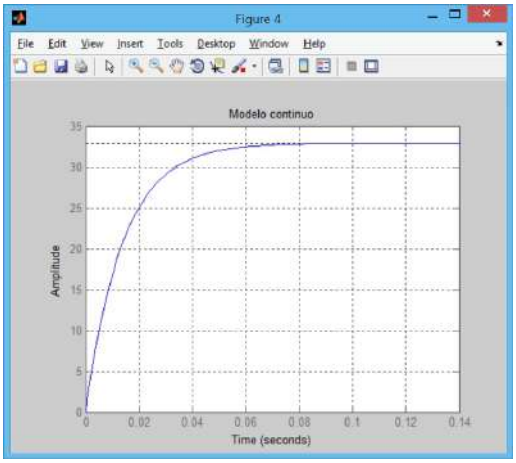
```
clc
clear all
close all
Ysalida=[0 150 200 280 500];
Ientrada=[0 30 60 120 255];
Param=[];

Ybuf=zeros(10,1); Ubuf=zeros(10,1);
Theta=zeros(2,1); P=100*eye(2);
Param=[];
amp=1; data=[];
Ns=10000;
figure(1); axis([0 Ns -10 140]); hold on;
puerto='COM4';
Ns2=num2str(120);
delete(instrfind({'Port'},{puerto}));
ps=serial(puerto);
set(ps,'Baudrate',9600);
set(ps,'DataBits',8);
set(ps,'Parity','none');
set(ps,'StopBits',1);
set(ps,'FlowControl','none');
set(ps,'Terminator','CR/LF')
fopen(ps);
for Tm=1:Ns,
   if(Tm==1)
       fprintf(ps,'%s\n',Ns2)
   end
   Valor=fscanf(ps)
   [in,out] = strtok(Valor, ' ');
   Yt=str2num(out)
   %%Yt=Ysalida(Tm);
   Ut=str2num(in)
   %%Ut=Ientrada(Tm);
   Fie=[-Ybuf(1); Ubuf(1)]; Lambda=0.99;
   ThetaOld=Theta; %use this if you want an animated plot (AP)
   [Theta,P]=RLS(Yt,Fie,Theta,P,Lambda); % my RLS function
   plot([Tm Tm-1],[Theta'; ThetaOld'],'Erasemode','none'); drawnow; %only for AP!
```

```
   Ybuf=[Yt; Ybuf(1:9)]; Ubuf=[Ut; Ubuf(1:9)];
   data=[data; [Ut Yt]];
   Param=[Param;Theta(1) Theta(2)];
end;
fprintf(ps,0);
fclose(ps);

delete(ps)
clear ps
A=[1 Param(Ns,1)];
B=[0 Param(Ns,2)];
sys30=tf(B,A,0.016)
figure
step(sys30)
title('Modelo discreto')
figure
sys30c=d2c(sys30)
step(sys30c)
title('Modelo continuo')
```

En una primera simulación se identificó el sistema con un PWM de 30, que tiene una salida del sistema identificado:

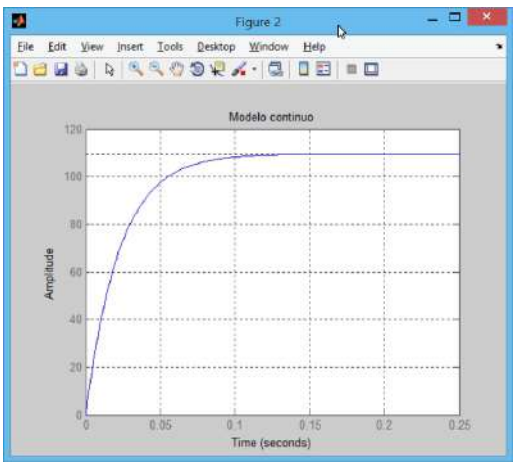
$$G\ (s) = \frac{4}{s + 4\ .1}$$

Este sistema corresponde a alimentar el tubo led con una entrada PWM 30.
La respuesta a una entrada escalón del sistema $G\ (s)$ identificado se observa a continuación.



Se ha identificado un segundo sistema a una entrada al sistema de 120 PWM, cuya función de transferencia es:

$$G\ (s) = \frac{2}{s + 7\ .4}$$



ANEXO D:

Programa: Simulador en Python.

Este es el código del programa simulador que principalmente hace uso de las librerías de Sklearn, numpy y scipy.

```
# -*- coding: utf-8 -*-
"""
Created on Set 18 10:50:40 2016
Tesis: Control Inteligente de sistemas de iluminación en Edificios

@author: Jezzy James Huamán Rojas
"""

from sklearn.externals import joblib
from PyQt4.uic import loadUiType
from PyQt4 import QtGui,QtCore
from matplotlib.figure import Figure
from matplotlib.backends.backend_qt4agg import (
    FigureCanvasQTAgg as FigureCanvas,
    NavigationToolbar2QT as NavigationToolbar)
import sys
import numpy as np
import JJutiles as libJJ
from scipy.signal import lti, lsim2

Ui_MainWindow, QMainWindow = loadUiType('Simu.ui')
Ui_diagPID, QMainPID=loadUiType('Diag_PID.ui')
Ui_diagAL,QMainAL=loadUiType('Diag_AL.ui')

#Armando la gráfica principal
fig1 = Figure()
ax1f1 = fig1.add_subplot(111)
ax1f1.grid('on')
ax1f1.axis([0,500,0,1010])

#Sistema Lineal
num=[2354]
den=[1,71.49]
Lti_luz=lti(num,den)
mt_T=[]
mt_yout=[0]
mt_xout=[.0]

#Variables generales
estado_S=0 #0 Stop - 1 Running
sr_lux=0
sr_pre=0
ml=0  #0 Tree decision / 1 Neural Network
t_pid=1 #0 PID with Identification / 1 PID without Identification
setpoint=0
t_graph=1 #1 Set point and output /2 Signal control /3 Evolution PID /4 Neural Network

#Variables de la Red Neuronal
var_senp=1
var_luxm=1000
rgMax=10

#Importar redes neuronales
AU_Tree=joblib.load('AuLearn/TREE.pkl')
AU_NN=joblib.load('AuLearn/NN.pkl')

#PID WITHOUT IDENTIFICATION
S1kp=0
S2ki=0
S3kd=0
er_pidwo=0
er_k_1_pidwo=0
fc_g=0.0000000000001
U_inp_pidwo=0

#PID WITHOUT IDENTIFICATION AN ET AL
er_pidwo_2=0.0
er_k_1_pidwo_2=0.0
er_k_2_pidwo_2=0.0
f_ki=0.0
```

```
f_ki_1=0.0
fc_gan=0.00000001
U_ant=0.0

#PID WITH IDENTICATION
Lambda=0.99
fc_b=1
P=1000*np.eye(3)
Theta=np.zeros((3,1))
Fie=np.array([[0],[0],[0]])
er_pidwi=0.0
er_pidwi_k_1=0.0
er_pidwi_k_2=0.0
u_ant_pidwi=0.0
f_Q=0.0
Y_t_1=0.0
Y_t_2=0.0
U_t_1=0.0

def NN_Say(sp,lx):
    sp=libJJ.remap(sp,0,var_senp,0,rgMax)
    lx=libJJ.remap(lx,0,var_luxm,0,rgMax)
    Xne=np.array([[sp,lx]])
    sadn=AU_NN.predict(Xne)
    for i in sadn:
        bs=i
    print "La NN: "+str(bs[0])
    return bs[0]

def TREE_Say(sp,lx):
    Xne=np.array([[sp,lx]])
    sadn=AU_Tree.predict(Xne)
    for i in sadn:
        bs=i
    print "El arbol: "+str(bs)
    return bs

class Dialog_PID(QtGui.QDialog, Ui_diagPID):
    def __init__(self, parent=None):
        QtGui.QDialog.__init__(self, parent)
        self.setupUi(self)

QtCore.QObject.connect(self.cmd_pid_cancel,QtCore.SIGNAL('clicked()'),self.Canc_PID)
        QtCore.QObject.connect(self.cmd_pid_ok,QtCore.SIGNAL('clicked()'),self.ok_PID)
        self.rbtn_widen.toggled.connect(lambda:self.type_PID(self.rbtn_widen))
        self.rbtn_woident.toggled.connect(lambda:self.type_PID(self.rbtn_woident))
        self.tt_pid=0
        self.parameter=0
        self.txtinp_parameter.setValidator(QtGui.QDoubleValidator())

    def Canc_PID(self):
        print "No se han registrado los cambios"
        self.close()

    def ok_PID(self):
        global fc_b,fc_g,t_pid
        print "ok"
        if self.tt_pid==0:
            fc_b=float(self.txtinp_parameter.text())
            print fc_b
            t_pid=0
        else:
            fc_g=float(self.txtinp_parameter.text())
            print fc_g
            t_pid=1
        self.close()

    def type_PID(self,btn):
        if btn.text()=="PID with identification":
            if btn.isChecked()==True:
                print "PID con identificacion"
                self.tt_pid=0
                self.lbl_parameter.setText("Parameter B:")
        if btn.text()=="PID without identification":
            if btn.isChecked()==True:
                print "PID without identification"
                self.lbl_parameter.setText("Parameter Gamma:")
```

```
                self.tt_pid=1

class Dialog_AL(QtGui.QDialog, Ui_diagAL):
    def __init__(self, parent=None):
        QtGui.QDialog.__init__(self, parent)
        self.setupUi(self)
        QtCore.QObject.connect(self.cmd_al_cancel,QtCore.SIGNAL('clicked()'),self.Canc_AL)
        QtCore.QObject.connect(self.cmd_al_ok,QtCore.SIGNAL('clicked()'),self.ok_AL)
        self.rbtn_tree.toggled.connect(lambda:self.type_AL(self.rbtn_tree))
        self.rbtn_nn.toggled.connect(lambda:self.type_AL(self.rbtn_nn))
        self.tt_au=0

    def Canc_AL(self):
        self.close()

    def ok_AL(self):
        global ml
        print "ok"
        if self.tt_au==0:
            ml=0
        else:
            ml=1
        self.close()

    def type_AL(self,btn):
        if btn.text()=="Tree decision":
            if btn.isChecked()==True:
                self.tt_au=0

        if btn.text()=="Neural network":
            if btn.isChecked()==True:
                self.tt_au=1

class Main(QMainWindow, Ui_MainWindow):
    def __init__(self, ):
        super(Main, self).__init__()
        self.setupUi(self)
        QtCore.QObject.connect(self.cmd_simular,QtCore.SIGNAL('clicked()'),self.Simular)
        QtCore.QObject.connect(self.cmd_stop,QtCore.SIGNAL('clicked()'),self.Stop)
        self.slr_lux.valueChanged.connect(self.Sli_move)
        self.rbtn_off.toggled.connect(lambda:self.Sensor_presense(self.rbtn_off))
        self.rbtn_on.toggled.connect(lambda:self.Sensor_presense(self.rbtn_on))
        self.action_PID.triggered.connect(lambda:self.showdialog_PID())
        self.action_ML.triggered.connect(lambda:self.showdialog_ML())
        self.action_Graph_Output.triggered.connect(lambda:self.Change_Graph1())
        self.action_Graph_Scontrol.triggered.connect(lambda:self.Change_Graph2())
        self.action_Graph_PID_const.triggered.connect(lambda:self.Change_Graph3())
        self.action_Graph_Neural.triggered.connect(lambda:self.Change_Graph4())
        self.txtinp_sp.setValidator(QtGui.QIntValidator())
        self.txtinp_sp.editingFinished.connect(self.Change_SP)
        self.Contador1=0
        self.OutLux=0
        self.Y=[]
        self.setpoint_g=[]
        self.luxmeter_g=[]
        self.sensorPIR_g=[]
        self.SAL_say_g=[]
        self.ucontrol_g=[]
        self.kp_g=[]
        self.ki_g=[]
        self.kd_g=[]

        self.Ts=0.02
        self.State1=QtGui.QLabel("State: Running")
        self.State1.setFixedWidth(110)
        self.State2=QtGui.QLabel("Self-learning system: Neural Network")
        self.State2.setFixedWidth(280)
        self.State3=QtGui.QLabel("Adaptive Control: PID without identification")
        self.State3.setFixedWidth(350)
        self.State4=QtGui.QLabel(" ")
        self.State4.setFixedWidth(60)
        self.Inicializate()

    def Inicializate(self):
        self.addmpl(fig1)
        self.Sensor_presense(self.rbtn_on)
        self.state_main.addWidget(self.State1)
```

```
        self.state_main.addWidget(self.State2)
        self.state_main.addWidget(self.State3)
        self.state_main.addWidget(self.State4)
        self.Status_Bar_Edition()

    def Change_Graph1(self):
        global t_graph, estado_S
        if estado_S==0:
            t_graph=1

    def Change_Graph2(self):
        global t_graph, estado_S
        if estado_S==0:
            t_graph=2

    def Change_Graph3(self):
        global t_graph, estado_S
        if estado_S==0:
            t_graph=3

    def Change_Graph4(self):
        global t_graph, estado_S
        if estado_S==0:
            t_graph=4

    def Change_SP(self):
        global setpoint
        setpoint=int(self.txtinp_sp.text())
        print "New set point is: "+ str(setpoint)

    def showdialog_PID(self):
        dialog=Dialog_PID()
        dialog.setAttribute(QtCore.Qt.WA_DeleteOnClose)
        dialog.exec_()
        self.Status_Bar_Edition()

    def showdialog_ML(self):
        dia_ML = Dialog_AL()
        dia_ML.setAttribute(QtCore.Qt.WA_DeleteOnClose)
        dia_ML.exec_()
        self.Status_Bar_Edition()

    def Status_Bar_Edition(self):
        global estado_S,ml,t_pid

        if estado_S==0: #0 Stop - 1 Running
            self.State1.setText('State: Stop')
            self.State4.setText('')
        else:
            self.State1.setText('State: Running')

        if ml==0:
            self.State2.setText('Self-learning system: Tree decision')
        else:
            self.State2.setText('Self-learning system: Neural Network')

        if t_pid==0:
            self.State3.setText('Adaptive Control: PID with identification')
        else:
            self.State3.setText('Adaptive Control: PID without identification')

    def Sensor_presense(self,btn):
        global sr_pre
        if btn.text()=="Full":
            if btn.isChecked()==True:
                sr_pre=1
        if btn.text()=="Alone":
            if btn.isChecked()==True:
                sr_pre=0

    def Sli_move(self):
        global sr_lux
        sr_lux=self.slr_lux.value()
        print "Lux en el ambiente: " + str(self.slr_lux.value())

    def Resetear_PIDwo(self):
        global S1kp,S2ki,S3kd,er_pidwo,er_k_1_pidwo
```

```
        S1kp=0
        S2ki=0
        S3kd=0
        er_pidwo=0
        er_k_1_pidwo=0

    def Resetear_PIDwI(self):
        global
P,Theta,Fie,er_pidwi,er_pidwi_k_1,er_pidwi_k_2,u_ant_pidwi,f_Q,Y_t_1,Y_t_2,U_t_1
        P=1000*np.eye(3)
        Theta=np.zeros((3,1))
        Fie=np.array([[0],[0],[0]])
        er_pidwi=0.0
        er_pidwi_k_1=0.0
        er_pidwi_k_2=0.0
        u_ant_pidwi=0.0
        f_Q=0.0
        Y_t_1=0.0
        Y_t_2=0.0
        U_t_1=0.0

    def Simular(self):
        global estado_S
        self.Contador1=0
        estado_S=1
        self.Y=[]
        self.setpoint_g=[]
        self.luxmeter_g=[]
        self.sensorPIR_g=[]
        self.SAL_say_g=[]
        self.ucontrol_g=[]
        self.kp_g=[]
        self.ki_g=[]
        self.kd_g=[]
        self.OutLux=sr_lux
        ax1f1.cla()
        ax1f1.grid('on')
        ax1f1.axis([0,500,0,1010])
        self.Resetear_PIDwo()
        self.Resetear_PIDwI()
        self.Status_Bar_Edition()
        self.update()

    def Stop(self):
        global estado_S
        estado_S=0
        self.Status_Bar_Edition()

    def addmpl(self, fig):
        self.canvas = FigureCanvas(fig)
        self.plot_vl.addWidget(self.canvas)
        self.canvas.draw()
        self.toolbar = NavigationToolbar(self.canvas,
                self.ctn_plot, coordinates=True)
        self.plot_vl.addWidget(self.toolbar)

    def Rpta_AL(self):
        if ml==0: #Tree decision
            Rpta=TREE_Say(sr_pre,sr_lux)
        else:
            Rpta=NN_Say(sr_pre,sr_lux)
        return Rpta

    def Calculo_PID_woI(self,ss_lux):
        global S1kp,S2ki,S3kd,er_pidwo,er_k_1_pidwo,fc_g,setpoint,sr_lux
        if setpoint<sr_lux:
            u=0
            kp=0
            ki=0
            kd=0
        else:
            er_pidwo=setpoint-ss_lux
            print S1kp,S2ki,S3kd,er_pidwo,er_k_1_pidwo,fc_g,setpoint,ss_lux
            S1kp=S1kp+er_pidwo**2
            kp=fc_g*er_pidwo*S1kp
            S2ki=S2ki+S1kp*er_pidwo
            ki=fc_g*S1kp*S2ki
```

```
            S3kd=S3kd+(er_pidwo-er_k_1_pidwo)*er_pidwo
            kd=fc_g*(er_pidwo-er_k_1_pidwo)*S3kd
            er_k_1_pidwo=er_pidwo+0.0
            u=kp+ki+kd

        return u,kp,ki,kd

    def Calculo_PID_woI2(self,ss_lux):
        global setpoint,sr_lux,
er_pidwo_2,er_k_1_pidwo_2,er_k_2_pidwo_2,f_ki,f_ki_1,fc_g2,U_ant
        if setpoint<sr_lux:
            u=0
            kp=0
            ki=0
            kd=0
        else:
            print
er_pidwo_2,er_k_1_pidwo_2,er_k_2_pidwo_2,f_ki,f_ki_1,fc_g2,U_ant,setpoint,ss_lux
            er_pidwo_2=setpoint-ss_lux
            kp=fc_g2*((er_k_1_pidwo_2**3)/3+(er_pidwo_2**2+er_k_1_pidwo_2**2)/2)
            f_ki=(er_pidwo_2+er_k_1_pidwo_2**2+er_k_1_pidwo_2)/2
            ki=fc_g2*f_ki**2
            kd=fc_g2*((er_pidwo_2*(er_k_1_pidwo_2+1))/2)
            a=kp+ki/2-kd
            b=-kp+ki/2-2*kd
            c=kd
            u=U_ant+a*er_pidwo_2+b*er_k_1_pidwo_2+c*er_k_2_pidwo_2
            U_ant=u+0.0
            er_k_2_pidwo_2=er_k_1_pidwo_2+0.0
            er_k_1_pidwo_2=er_pidwo_2+0.0
            f_ki_1=f_ki+0.0
        return u,kp,ki,kd

    def Calculo_PID_Ident(self,ss_lux):
        global fc_b,er_pidwi,er_pidwi_k_1,er_pidwi_k_2,u_ant_pidwi,f_Q,setpoint,sr_lux
        if setpoint<sr_lux:
            u=0
            kp=0
            ki=0
            kd=0
        else:
            er_pidwi=setpoint-ss_lux
            a1=Theta[0]
            a2=Theta[1]
            b1=Theta[2]
            f_Q=1-np.exp(-(1.0/fc_b))
            if a1!=0 and  a2!=0 and b1!=0:
                kp=(-(a1+2*a2)*f_Q)/b1
                kd=(a2*f_Q)/(kp*b1)
                ki=-1/((1/(a1+2*a2))+1+kd)
            else:
                kp=0.0001
                ki=1
                kd=0.01
                u=0.1
                return u,kp,ki,kd

            du=kp*(er_pidwi-er_pidwi_k_1+(1/ki)*er_pidwi+kd*(er_pidwi-
2*er_pidwi_k_1+er_pidwi_k_2))
            u=du+u_ant_pidwi
            u_ant_pidwi=u+0.0
            er_pidwi_k_1=er_pidwi+0.0
            er_pidwi_k_2=er_pidwi_k_1+0.0
        return u,kp,ki,kd

    def PIDwid_RLS(self):
        global Y_t_1,Y_t_2,U_t_1,U_inp_pidwo,Lambda,P,Theta,Fie
        if t_pid==0:
            Fie=np.array([[-Y_t_1],[-Y_t_2],[U_t_1]])
            [Theta,P]=libJJ.RLS(self.OutLux,Fie,Theta,P,Lambda)
            Y_t_2=Y_t_1+0.0
            Y_t_1=self.OutLux
            U_t_1=U_inp_pidwo

    def update(self):
```

```
        global
estado_S,sr_lux,ax1f1,U_inp_pidwo,Lti_luz,mt_T,mt_yout,mt_xout,setpoint,var_senp,var_luxm,r
gMax
        if self.Rpta_AL()==1:
            self.State4.setText("ON")
            if t_pid==1:
                U_inp_pidwo,kp,ki,kd=self.Calculo_PID_woI(self.OutLux)
            else:
                U_inp_pidwo,kp,ki,kd=self.Calculo_PID_Ident(self.OutLux)
            print "Ucalculada: " + str(U_inp_pidwo)
            if setpoint>sr_lux:
                if t_pid==1:
                    mt_T,mt_yout,mt_xout=lsim2(Lti_luz,T=np.linspace(0,0.5,5),

U=np.ones((5,1))*U_inp_pidwo,X0=mt_xout[len(mt_xout)-1])
                else:
                    self.PIDwid_RLS()
                    mt_T,mt_yout,mt_xout=lsim2(Lti_luz,T=np.linspace(0,self.Ts,5),

U=np.ones((5,1))*U_inp_pidwo,X0=mt_xout[len(mt_xout)-1])

                self.OutLux=sr_lux+mt_yout[len(mt_yout)-1]
                print "salida C: "+str(self.OutLux)

            else:
                self.OutLux=sr_lux
                self.Resetear_PIDwo()


        if self.Rpta_AL()==0:
            self.State4.setText("OFF")
            self.OutLux=sr_lux
            self.Resetear_PIDwo()
            self.Resetear_PIDwI()

        self.lcd_lux.display(self.OutLux)
        self.Contador1=self.Contador1+1

        if t_graph==1:
            if self.Contador1>500:
                ax1f1.axis([self.Contador1-500,self.Contador1,0,1010])
            else:
                ax1f1.axis([0,500,0,1010])

            self.Y.append(self.OutLux)
            self.setpoint_g.append(setpoint)
            X=np.arange(len(self.Y))
            ax1f1.plot(X,self.Y,ms=100,color='blue',lw=1,alpha=.8, label='Output')
            ax1f1.hold(True)
            ax1f1.plot(X,self.setpoint_g,ms=100,color='red',lw=0.5,alpha=.8,label='Set
point')
            if self.Contador1==1:
                ax1f1.legend()

        elif t_graph==2:
            if self.Contador1>500:
                ax1f1.axis([self.Contador1-500,self.Contador1,0,20])
            else:
                ax1f1.axis([0,500,0,20])
            self.ucontrol_g.append(U_inp_pidwo)
            X=np.arange(len(self.ucontrol_g))
            ax1f1.plot(X,self.ucontrol_g,ms=100,color='blue',lw=1,alpha=.8,label='Signal
control')
            if self.Contador1==1:
                ax1f1.legend()

        elif t_graph==3:
            print "para Grafico"
            print kp,ki,kd
            if self.Contador1>500:
                ax1f1.axis([self.Contador1-500,self.Contador1,0,20])
            else:
                ax1f1.axis([0,500,0,20])
            self.kp_g.append(kp)
            self.ki_g.append(ki)
            self.kd_g.append(kd)
            self.ucontrol_g.append(U_inp_pidwo)
```

```
            X=np.arange(len(self.kp_g))
            ax1f1.plot(X,self.ucontrol_g,ms=100,color='red',lw=1.5,alpha=.7,label='Signal
control')
            ax1f1.hold(True)
            ax1f1.plot(X,self.kp_g,ms=100,color='orange',lw=1.5,alpha=.8,label='Kp')
            ax1f1.plot(X,self.ki_g,ms=100,color='blue',lw=1.,alpha=.9,label='Ki')
            ax1f1.plot(X,self.kd_g,ms=100,color='green',lw=1.5,alpha=1, label='Kd')
            if self.Contador1==1:
                ax1f1.legend()

        elif t_graph==4:
            if self.Contador1>500:
                ax1f1.axis([self.Contador1-500,self.Contador1,0,7.1])
            else:
                ax1f1.axis([0,500,0,7.1])
            g_sp=libJJ.remap(sr_pre,0,var_senp,0,3)
            g_lx=libJJ.remap(sr_lux,0,var_luxm,0,rgMax)
            self.luxmeter_g.append(g_lx)
            self.sensorPIR_g.append(g_sp)
            self.SAL_say_g.append(self.Rpta_AL()*7)
            X=np.arange(len(self.luxmeter_g))

ax1f1.plot(X,self.luxmeter_g,ms=100,color='green',lw=1.5,alpha=.8,label='Luxmeter')
            ax1f1.hold(True)

ax1f1.plot(X,self.sensorPIR_g,ms=100,color='orange',lw=1.5,alpha=.8,label='Sensor PIR')
            ax1f1.plot(X,self.SAL_say_g,ms=100,color='red',lw=1.5,alpha=.8,label='Output
SSL')
            if self.Contador1==1:
                ax1f1.legend()
        self.canvas.draw()


        if estado_S==1:
            QtCore.QTimer.singleShot(10, self.update) # QUICKLY repeat

        if estado_S==0:
            self.Status_Bar_Edition()

if __name__ == '__main__':
    ##Programa para hacer correr todo

    app = QtGui.QApplication(sys.argv)
    main = Main()
    main.show()


    app.exec_()
    print "Fin de programa"
```

ANEXO E:

Programa: Sistema embebido del Control inteligente de iluminación en edificios

Este programa es la síntesis de la tesis en un microcontrolador mini Arduino Pro.

ANEXO E1: Codigo fuente del Arduino

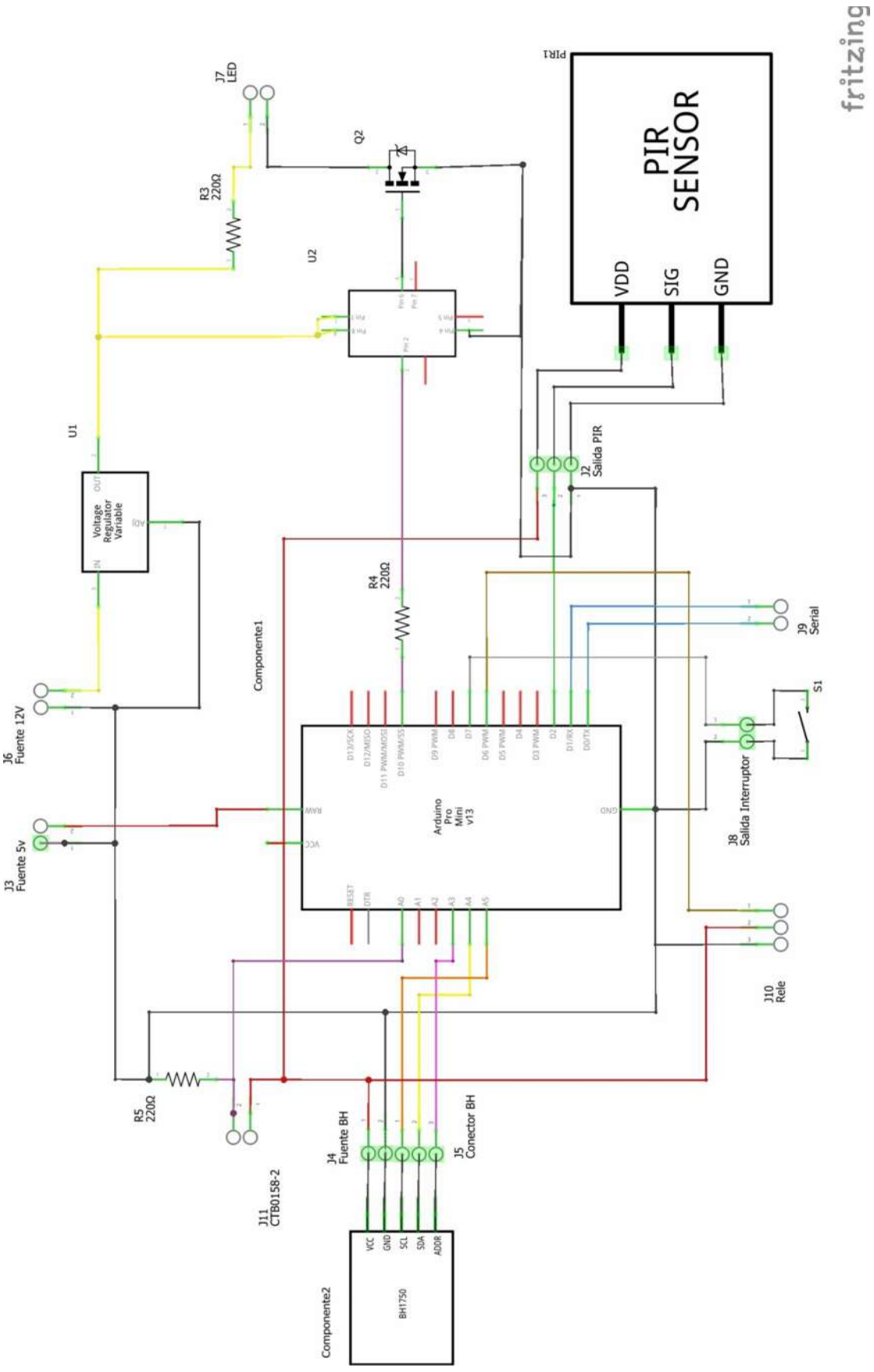
```
#include <Wire.h>
#include "BH1750FVI.h"
#include <EEPROM.h>
//Definimos pines de operación
int Led=10; //Led
int S_pr=2; //Sensor de presencia
int Rele=6; //Relay para la fuente
int Switch=7; //Interruptor
BH1750FVI LightSensor; //Luxometro

//Lineas de control
int TreeD=0;
int ControlPID=0;
int Sensado=0;
int Interruptor=0;

//Variables compartidas
uint16_t lux =0; //Medida del luxometro
uint16_t lux_a =0; //Medida del luxometro
int lux_dn=0;    //Medida de la luminaria al momento de decidir encender la luz
int ss_pr=0;     //Señal del PIR
int estado=0;    //0 en caso de apagado y 1 en caso de encendido
float ST=0.13;        //Tiempo de muestreo
float fc_general=0;

//Variables de TreeD
int Nro_register=0;//Numero de registro
int Ty_Learn=0;//0 para aprendizaje en microprocesador 1 para aprendizaje en SBC

//Variables del PID
int Setpoint=0; //Para el funcionamiento
int Setpp_N=0;  //Punto de operación de acuerdo a norma
float u;        //Variable de control
float dEr;      //derivada del error
float er,er a,S1kp,S2ki,S3kd,a S1kp,a_S2ki,a_S3kd; //factores de proceso del PID
float kp,ki,kd; //salidas del proceso
float gamma=4*10e-12; //El factor gamma
float u_c=0;          //Otra variable de control
float u_a=0;          //Variable de control anterior

void InicioVariables(){
  er=0;
  er_a=0;
  u=0;
  u_a=0;
  u_c=0;
  dEr=0;
  S1kp=0;
  S2ki=0;
  S3kd=0;
  a_S1kp=0;
  a_S2ki=0;
  a_S3kd=0;
}

float Calc_pid(float sensor){
  er=Setpoint-sensor;
  dEr=(er-er a)/ST;
  S1kp=a_S1kp+(er*er)*ST;
  kp=gamma*er*S1kp;
  S2ki=a_S2ki+S1kp*er*ST;
  ki=gamma*S1kp*S2ki;
  S3kd=a_S3kd+dEr*er;
  kd=gamma*dEr*S3kd;
  u=kp+ki+kd;
```

```
  if (u>0 && u<256){
    a_S1kp=S1kp;
    a_S2ki=S2ki;
    a_S3kd=S3kd;
    er_a=er;
  }else{
    if(u<0){
      u=0;
    }
    if (u>255){
      u=255;
    }
  }
  if (abs(u-u_a)<1){
    u=u_a;
  }else{
    u_a=u;
  }
  u=(int)u;
  return u;
}

void Primera_Configuracion(){
  EEPROM.write(1,62);
  EEPROM.write(2,3);//Centena de Setpoint
  EEPROM.write(3,0);//Unidades y decenas de Setpoint
  EEPROM.write(6,1);//Comenzar a grabar desde el primer registro
  EEPROM.write(7,0);//Aprendizaje en Microprocesador
  EEPROM.write(8,10);//Factor de correción
  EEPROM.write(9,3);//Setpoint de acuerdo a norma
  EEPROM.write(10,0);//unidades y decenas de la norma
  //De 20 - 50 se guardan las últimas configuraciones
  delay(100);
}

void Lectura_prev_value(){
  //setpoint
  byte p1=EEPROM.read(2);
  byte p2=EEPROM.read(3);
  Setpoint=p1*100+p2;
  //Numero de registro
  Nro_register=EEPROM.read(6);
  //Tipo de aprendizaje
  Ty_Learn=EEPROM.read(7);
  //Factor de corrección
  fc_general=EEPROM.read(8)/10;
  //Set point de acuerdo a norma
  byte p5=EEPROM.read(9);
  byte p6=EEPROM.read(10);
  Setpp_N=p5*100+p6;
}

void setup() {   // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(Led, OUTPUT);
  analogWrite(Led,0);
  pinMode(S_pr, INPUT);
  pinMode(Switch, INPUT_PULLUP);
  digitalWrite(Rele,LOW);
  LightSensor.begin();
  LightSensor.SetAddress(Device_Address_L);//Address 0x5C
  LightSensor.SetMode(Continuous_H_resolution_Mode);//Resolución máxima con tiempo de
muestreo mayor a 120 ms
  int temp=0;
  temp=EEPROM.read(1);
  if (temp!=62){
    Primera_Configuracion();
  }
  Lectura_prev_value();
  noInterrupts();           // disable all interrupts
  TCCR2A = 0;
  TCCR2B = 0;
  TCNT2 = 0;            // preload timer 65536-16MHz/256/2Hz 34286
  TCCR2B |= (1 << CS12);    // 256 prescaler
  TIMSK2 |= (1 << TOIE2);   // enable timer overflow interrupt
  interrupts();             // enable all interrupts
  InicioVariables();
```

```
  Sensar_JS();
  Decisiones();
}

ISR(TIMER2_OVF_vect){//Leer sensor
  TreeD=TreeD+1;
  ControlPID=ControlPID+1;
  Sensado=Sensado+1;
  Interruptor=Interruptor+1;
  TCNT2= 0;            // preload timer
}

bool Tree_say(int sensorp,int luxmeter){
  if (sensorp==1){
    if (luxmeter<Setpoint){
      return true;
    }else{
      return false;
    }
  }else{
    return false;
  }
}

void Aprender_DT(float luxuser){
  float nsp=(Setpoint+luxuser)/2;
  if (nsp>=Setpp_N){
    Setpoint=nsp;
    int uyd=(int)Setpoint%100;
    int cmSp=(int)(Setpoint-uyd)/100;
    EEPROM.write(2,cmSp);//Centena de Setpoint
    EEPROM.write(3,uyd);//Unidades y decenas de Setpoint
  }else{
    Setpoint=Setpp_N;
    int uyd=(int)Setpoint%100;
    int cmSp=(int)(Setpoint-uyd)/100;
    EEPROM.write(2,cmSp);//Centena de Setpoint
    EEPROM.write(3,uyd);//Unidades y decenas de Setpoint
  }
}

void Aprendizaje_Tree(float luxo, int rptaUser){
  //Guardar en la memoria
  int pos=Nro_register*2+18;
  int value=(int)luxo/4-1;
  EEPROM.write(pos,value);
  EEPROM.write(pos+1,rptaUser);
  Nro_register=Nro_register+1;
  if (Nro_register>10){
    EEPROM.write(6,1);
    Nro_register=1;
  }else{
    EEPROM.write(6,Nro_register+1);
  }
  Serial.print("modo: ");
  Serial.println(Ty_Learn);
  if (Ty_Learn==0){ //Aprender de datos
    Aprender_DT(luxo);
  }
}

void Sensar_JS(){
  lux = LightSensor.GetLightIntensity();
  lux=lux*fc_general;
  if (lux>1024){
    lux=1024;
  }
  ss_pr=digitalRead(S_pr);
}

void Controlar_JS(){
  u_c=Calc_pid(lux);
  if (u_c!=0){
    digitalWrite(Rele,HIGH);
    analogWrite(Led,u_c);
  }else{
    digitalWrite(Rele,LOW);
```

```
    analogWrite(Led,0);
  }
}

void Decisiones(){
  if(Tree_say(ss_pr,lux_dn)){
    estado=1;
  }else{
    estado=0;
    lux_dn=lux;
  }
}


void loop() {
  if (Sensado>125){
    Sensar_JS();
    Sensado =0;
  }

  if(ControlPID>130){
    if (estado==1 && Setpoint>lux_dn){
      Controlar_JS();

    }else{
      InicioVariables();
      digitalWrite(Rele,LOW);
      analogWrite(Led,0);
    }
    Serial.print(Setpoint);
    Serial.print(",");
    Serial.print(lux);
    Serial.print(",");
    Serial.print(ss_pr);
    Serial.print(",");
    Serial.print(estado);
    Serial.print(",");
    Serial.print(u_c);
    Serial.println(",");
    ControlPID=0;

  }

  if (Interruptor>350){
    int R_Switch=digitalRead(Switch);
    if (R_Switch==0){
      int rppta=0;
      if (estado==1){
        rppta=0;
      }else{
        rppta=1;
      }
      Aprendizaje_Tree(lux_dn,rppta);
    }
    Interruptor=0;
  }

  if(TreeD>1024){
    Decisiones();
    TreeD=0;
  }
}
```

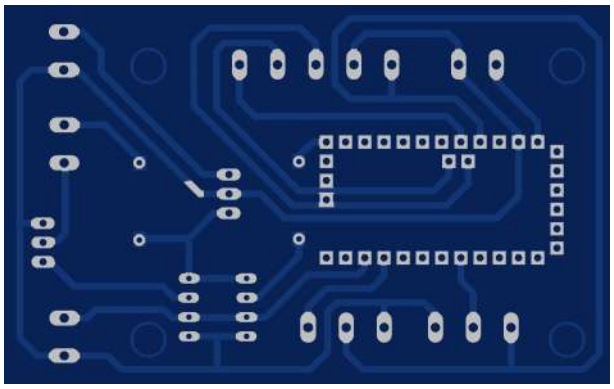
ANEXO E2: Esquema del circuito electrónico del prototipo elaborado en Fritzing



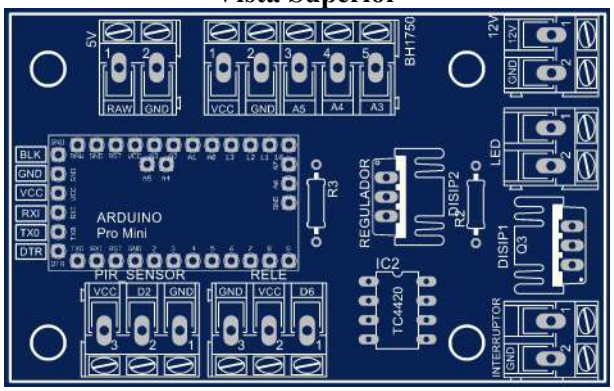
ANEXO E3: Diseño del PCB

La siguiente placa fue diseñadó por el Téc. Julio C. Salazar Morales y elaborado en el Laboratorio de Sistemas Automáticos de Control de la Universidad de Piura

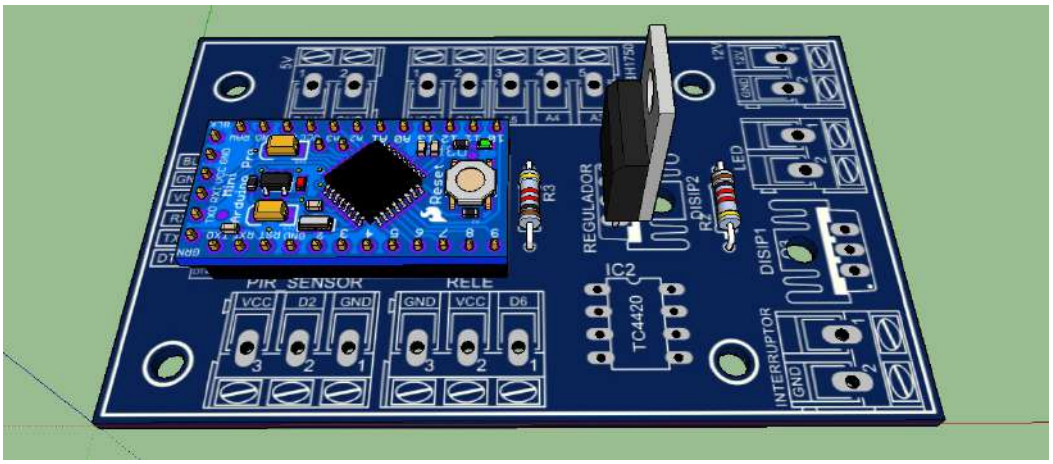
Vista Inferior



Vista Superior



Vista 3D



Fuente Propia (Elaborado con CadSoft Eagle)