

**Universidad Autónoma Metropolitana – Azcapotzalco**

**División de Ciencias Básicas e Ingeniería**

**Licenciatura en Ingeniería en Computación**

**Dispositivo de iluminación LED para habitaciones con  
incorporación de electrónica digital y control desde Android por  
bluetooth.**

Modalidad: Proyecto tecnológico

Trimestre 2016-Otoño

Hernández Alves Mario Alberto  
209365718  
Marbeto\_lvs@hotmail.com

Asesor

Dr. José Alejandro Reyes Ortiz  
Profesor Asociado  
jaro@correo.azc.uam.mx  
Departamento de Sistemas

Coasesor

Dr. Leonardo Daniel Sánchez Martínez  
Profesor Asociado  
ldsm1985@gmail.com  
Departamento de Sistemas

07 de diciembre de 2016



## Declaratoria

Yo, Dr. Leonardo Daniel Sánchez Martínez, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Yo, Dr. José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Yo, Mario Alberto Hernández Alves, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



## Resumen

En este reporte de proyecto terminal se explica la forma en que se realizó el diseño e implementación de un dispositivo de iluminación LED para habitaciones con incorporación de electrónica digital y control desde Android por bluetooth. El dispositivo incorpora funcionalidades que intentan romper el paradigma de la iluminación actual.

Se describen los procesos de diseño e implementación de hardware y software para el desarrollo del proyecto, las herramientas que se utilizaron, las técnicas requeridas y los procesos de construcción de cada elemento que conforma el dispositivo.

Se incluyen todas las etapas del desarrollo desde el diseño hasta la construcción final, también se explica el funcionamiento del dispositivo desde la perspectiva del usuario final y se analizan los resultados obtenidos.

## Tabla de contenido

|        |  |    |
|--------|--|----|
| 1      | Introducción.....                              | 1  |
| 2      | Antecedentes.....                              | 3  |
| 2.1    | Proyectos terminales o de integración.....     | 3  |
| 2.2    | Hardware y software comercial.....             | 3  |
| 3      | Justificación.....                             | 4  |
| 4      | Objetivos.....                                 | 5  |
| 4.1    | Objetivo general.....                          | 5  |
| 4.2    | Objetivos particulares.....                    | 5  |
| 5      | Marco teórico.....                             | 6  |
| 5.1    | Electrónica.....                               | 6  |
| 5.2    | Circuito eléctrico.....                        | 6  |
| 5.3    | La corriente, el voltaje y la resistencia..... | 7  |
| 5.4    | Tipos de corriente y frecuencia.....           | 7  |
| 5.5    | Potencia.....                                  | 8  |
| 5.6    | Ley de ohm.....                                | 8  |
| 5.7    | Circuito en serie y paralelo.....              | 8  |
| 5.8    | Componentes de circuitos eléctricos.....       | 9  |
| 5.9    | Electrónica digital.....                       | 9  |
| 5.9.1  | Los sensores.....                              | 10 |
| 5.9.2  | Los microcontroladores.....                    | 10 |
| 5.9.3  | Los transmisores.....                          | 10 |
| 5.9.4  | Los módulos digitales.....                     | 11 |
| 5.10   | Hardware.....                                  | 11 |
| 5.10.1 | Desarrollo de hardware.....                    | 12 |
| 5.11   | Software.....                                  | 12 |
| 5.11.1 | Lenguajes de programación.....                 | 12 |
| 5.11.2 | Entornos de desarrollo integrados.....         | 13 |
| 5.11.3 | Los diagramas de flujo.....                    | 13 |
| 5.11.4 | Los casos de uso de texto.....                 | 13 |
| 5.11.5 | Paradigmas de programación.....                | 14 |
| 6      | Desarrollo del proyecto.....                   | 15 |

|  |    |
|--|----|
| 6.1 El módulo de control a distancia .....                                       | 16 |
| 6.1.1 Diseño .....   | 17 |
| 6.1.2 Implementación .....   | 19 |
| 6.2 El módulo de recepción de instrucciones por bluetooth .....                  | 22 |
| 6.2.1 La parte de hardware.....  | 22 |
| 6.2.2 La parte de software.....  | 23 |
| 6.3 Para el módulo de recepción de señales de presencia y botón de control ..... | 25 |
| 6.3.1 Diseño del circuito electrónico.....                                       | 25 |
| 6.3.2 Diseño del circuito impreso .....  | 28 |
| 6.3.3 Construcción del circuito impreso.....                                     | 30 |
| 7 Para el módulo de potencia.....  | 37 |
| 7.1 Diseño del circuito electrónico.....   | 37 |
| 7.2 Diseño del circuito impreso .....  | 40 |
| 7.3 Construcción del circuito impreso.....                                       | 43 |
| 8 Para el módulo de iluminación .....  | 49 |
| 8.1 Diseño .....   | 49 |
| 8.2 Construcción .....   | 50 |
| 9 Diagrama de conexión entre módulos .....                                       | 54 |
| 10 Para el módulo de procesamiento de instrucciones.....                         | 56 |
| 10.1 Para la parte de hardware .....   | 56 |
| 10.2 Para la parte de software .....   | 57 |
| 10.2.1 La parte del modelado del sistema.....                                    | 57 |
| 10.2.2 Implementación del sistema.....   | 62 |
| 11 Desarrollo de carcasa .....   | 66 |
| 11.1 Diseño .....  | 66 |
| 11.1.1 Parte posterior .....   | 66 |
| 11.1.2 Parte frontal .....   | 67 |
| 11.2 Construcción .....  | 67 |
| 12 Ensamblado .....  | 71 |
| 13 Resultados .....  | 77 |
| 14 Conclusiones .....  | 79 |
| 15 Referencias .....   | 80 |

|                    |     |
|--------------------|-----|
| 16 Apéndices ..... | 81  |
| 16.1 A .....       | 81  |
| 16.2 B .....       | 137 |

## Tabla de Figuras:

|   |    |
|---|----|
| Figura 1. Ejemplo de los elementos de un circuito eléctrico básico.....   | 6  |
| Figura 2. Movimiento de los electrones en un circuito.....  | 7  |
| Figura 3. Circuito en serie.....  | 9  |
| Figura 4. Circuito en paralelo.....   | 9  |
| Figura 5. Descripción general de la interacción entre los dispositivos. ....  | 15 |
| Figura 6. Descripción gráfica del dispositivo de control fijo.....  | 16 |
| Figura 7. Transformación de acciones sobre la pantalla táctil a instrucciones bluetooth mediante el módulo de control a distancia. .... | 16 |
| Figura 8. Descripción gráfica del dispositivo de control a distancia. ....  | 17 |
| Figura 9. Etapa de diseño (diagrama de flujo) para el módulo de control a distancia. ....   | 17 |
| Figura 10. Diagrama de flujo del módulo de control a distancia.....   | 19 |
| Figura 11. Evento para seleccionar un sistema de iluminación y disparar un procedimiento correspondiente. ....                          | 20 |
| Figura 12. Eventos que se disparan al cambiar las coordenadas del acelerómetro. ....  | 20 |
| Figura 13. Controladores de bluetooth, de acelerómetro y de notificaciones agregados al proyecto. ....                                  | 20 |
| Figura 14. Implementación de la vista en el IDE App Inventor 2.....   | 21 |
| Figura 15. Descripción general del módulo de recepción de instrucciones por bluetooth. ....   | 22 |
| Figura 16. Módulo bluetooth HC-06. ....   | 22 |
| Figura 17. Desarrollo del programa de envío de instrucciones.....   | 23 |
| Figura 18. Conexión del Arduino a la Interfaz USB. ....   | 24 |
| Figura 19. Conexión de los puertos seriales y alimentación entre el módulo bluetooth y Arduino. ....                                    | 24 |
| Figura 20. Detección de presencia o actividad del botón de control. ....  | 25 |
| Figura 21. Componente electrónico: pulsador normalmente abierto.....  | 25 |
| Figura 22. Sensor Pir HC-Sr501.....   | 26 |
| Figura 23. Área de diseño de EasyEda con los componentes del circuito desconectados. ....   | 26 |
| Figura 24. Herramientas de alambrado de EasyEda. ....   | 27 |
| Figura 25. Opción para exportar el diseño a imagen o pdf en EasyEda.....  | 27 |
| Figura 26. Diagrama final del circuito electrónico para el módulo de recepción de señales de presencia y botón de control.....          | 27 |
| Figura 27. Función de EasyEda para convertir los componentes y rutas de un circuito electrónico en un circuito impreso. ....            | 28 |
| Figura 28. Colocación de los componentes dentro del área del circuito impreso. ....   | 28 |
| Figura 29. Proceso de creación de las rutas (pistas) de forma manual para el circuito impreso. ....                                     | 29 |
| Figura 30. Creación de orificios para el circuito impreso.....  | 29 |
| Figura 31. Diseño final del circuito impreso. ....  | 29 |
| Figura 32. Pistas del circuito impreso final.....   | 30 |
| Figura 33. Router CNC X2001. ....   | 30 |
| Figura 34. Software de control numérico computarizado Mach 3. ....  | 31 |
| Figura 35. Software CAM Vectric Aspire. ....  | 31 |
| Figura 36. Circuito transformado en vectores por Vectric Aspire. ....   | 32 |
| Figura 37. Generación de código G en Vectric Aspire.....  | 32 |



|  |    |
|--|----|
| Figura 38. Placa fenólica sin ser fresada. ....  | 33 |
| Figura 39. Código G cargado en el software CNC Mach 3. ....  | 33 |
| Figura 40. Proceso de fresado de la placa fenólica. ....   | 34 |
| Figura 41. Colocación de componentes y buses. ....   | 34 |
| Figura 42. Componentes electrónicos soldados en el circuito impreso. ....  | 35 |
| Figura 43. Comprobando el funcionamiento correcto de los componentes. ....   | 35 |
| Figura 44. El módulo de recepción de señales de presencia y botón de control terminado. ....                                 | 36 |
| Figura 45. Amplificación de señal por el módulo de potencia. ....  | 37 |
| Figura 46. Transistor TIP120 Darlington. ....  | 37 |
| Figura 47. Diodo rectificador 1N4001. ....   | 38 |
| Figura 48. Resistor de un 1K. ....   | 38 |
| Figura 49. Área de diseño de EasyEda con los componentes del circuito desconectados. ....                                    | 38 |
| Figura 50. Herramientas de alambrado de EasyEda. ....  | 39 |
| Figura 51. Opción para exportar el diseño a imagen o pdf en EasyEda. ....  | 39 |
| Figura 52. Diagrama final del circuito electrónico para el módulo de potencia. ....  | 40 |
| Figura 53. Función de EasyEda para convertir los componentes y rutas de un circuito electrónico en un circuito impreso. .... | 40 |
| Figura 54. Colocación de los componentes dentro del área del circuito impreso. ....  | 41 |
| Figura 55. Proceso de creación de las rutas (pistas) de forma manual para el circuito impreso. ....                          | 41 |
| Figura 56. Creación de orificios para el circuito impreso. ....  | 42 |
| Figura 57. Diseño final del circuito impreso. ....   | 42 |
| Figura 58. Pistas del circuito impreso final. ....   | 43 |
| Figura 59. Orificios del circuito impreso final. ....  | 43 |
| Figura 60. Circuito transformado en vectores por Vectric Aspire. ....  | 44 |
| Figura 61. Generación de código G en Vectric Aspire. ....  | 44 |
| Figura 62. Placa fenólica sin ser fresada. ....  | 45 |
| Figura 63. Código G cargado en el software CNC Mach 3. ....  | 45 |
| Figura 64. Proceso de fresado de la placa fenólica. ....   | 46 |
| Figura 65. Colocación de componentes y buses. ....   | 46 |
| Figura 66. Componentes electrónicos soldados en el circuito impreso. ....  | 47 |
| Figura 67. Comprobando el funcionamiento correcto de los componentes. ....   | 47 |
| Figura 68. El módulo de potencia terminado. ....   | 48 |
| Figura 69. Descripción gráfica del módulo de iluminación. ....   | 49 |
| Figura 70. Tira de Leds SMD 5050. ....   | 49 |
| Figura 71. Simulación 3D de la base para la tira LED en el software Vectric Aspire 8. ....                                   | 50 |
| Figura 72. Material base en madera de pino. ....   | 51 |
| Figura 73. Medidas marcadas para los cortes de las bases. ....   | 51 |
| Figura 74. Bases para las tiras LED cortadas y lijadas. ....   | 51 |
| Figura 75. Pintado de las bases para las tiras LED. ....   | 52 |
| Figura 76. Orificios y ranuras para los cables de alimentación. ....   | 52 |
| Figura 77. Orificios para la alimentación de las tiras LED. ....   | 52 |
| Figura 78. Extremo del módulo de iluminación central, parte delantera y posterior. ....                                      | 53 |
| Figura 79. Módulo de iluminación terminado. ....   | 53 |
| Figura 80. Diagrama de conexión entre módulos para formar el dispositivo general. ....                                       | 54 |

|  |    |
|--|----|
| Figura 81. Opción para exportar el diseño a imagen o pdf en EasyEda. ....                                  | 55 |
| Figura 82. Funcionamiento general del módulo de procesamiento de instrucciones.....                        | 56 |
| Figura 83. Arquitectura Arduino Uno R3.....  | 56 |
| Figura 84. Procedimiento principal.....  | 62 |
| Figura 85. Flujo alternativo para el procedimiento de encendido. ....                                      | 63 |
| Figura 86. Algunas variables tipo byte. ....   | 63 |
| Figura 87. Uso de memoria dinámica del sistema. ....   | 64 |
| Figura 88. Procesos independientes para subir o bajar intensidad.....                                      | 64 |
| Figura 89. Proceso para realizar un conteo de programación de apagado, independientemente del tiempo. .... | 65 |
| Figura 90. Información final de la compilación del sistema. ....   | 65 |
| Figura 91. Pre-visualización 3D de la parte posterior de la carcasa. ....                                  | 66 |
| Figura 92. Pre-visualización 3D de la parte frontal de la carcasa. ....                                    | 67 |
| Figura 93. Software CAM Vectric Aspire. ....   | 67 |
| Figura 94. Vectores para la parte posterior de la carcasa.....   | 68 |
| Figura 95. Parte posterior de la carcasa después de ser fresada. ....                                      | 68 |
| Figura 96. Parte frontal de la carcasa después de ser fresada. ....  | 69 |
| Figura 97. Parte posterior de la carcasa después de ser pintada. ....                                      | 69 |
| Figura 98. Carcasa lista.....  | 70 |
| Figura 99. Fijación de la arquitectura de Arduino en la carcasa.....                                       | 71 |
| Figura 100. Fijación del módulo de potencia a la carcasa.....  | 71 |
| Figura 101. Fijación del módulo de recepción de señales de presencia y botón de control. ....              | 72 |
| Figura 102. Fijación del módulo bluetooth. ....  | 72 |
| Figura 103. Colocación de botón de control y sensor PIR.....   | 73 |
| Figura 104. Cables de salida para la conexión de los módulos de iluminación.....                           | 73 |
| Figura 105. Nudos para evitar desplazamiento de los cables hacia el exterior. ....                         | 74 |
| Figura 106. Conexión entre módulos.....  | 74 |
| Figura 107. Unión de la parte posterior y frontal de la carcasa. ....                                      | 75 |
| Figura 108. Proceso de conexión de los módulos de iluminación.....   | 75 |
| Figura 109. El dispositivo totalmente armado.....  | 76 |

# 1 Introducción

La tecnología LED es un invento de la década de los 60 del siglo pasado pero su uso, para fines de iluminación doméstica, es reciente y tiene enormes ventajas respecto a la iluminación incandescente e iluminación halógena. Una bombilla LED, con un promedio de vida de 50 000 horas, es 30 veces más duradera que una bombilla incandescente y 25 veces más que una bombilla halógena. En cuanto a eficiencia y ahorro las bombillas que incorporan tecnología LED son capaces de brindar la misma cantidad de luz (medida en lúmenes<sup>1</sup>) que una bombilla incandescente pero con un ahorro de 80% de energía, esto se debe a que las bombillas incandescentes y halógenas gastan hasta el 90% de la energía que consumen en calor [1], proceso conocido como “efecto Joule”, y sólo el 10% se transforma en iluminación. Entre otras características a favor de la tecnología LED podemos listar las siguientes:

- Libre de mercurio y tóxicos.
- No producen luz ultravioleta ni infrarroja.
- Son reciclables.
- Reduce el cansancio visual.
- Su uso reduce el problema del calentamiento global.

Entonces, si las ventajas de usar tecnología LED son tantas ¿por qué la mayoría de las habitaciones de los hogares siguen utilizando bombillas incandescentes y halógenas?

La tecnología LED, a pesar de sus ventajas, mantiene un modo de funcionamiento igual al de la luz incandescente y halógena, es decir, no tiene funcionalidades extras. Por ello los usuarios lo consideran un gasto elevado innecesario. Para acercar a la gente común a la tecnología LED empresas como Apple inc. y Philips han implementado dispositivos que aprovechan la corriente directa e incorporan electrónica digital para crear iluminadores novedosos con algunas funcionalidades extras como cambio de color de la luz emitida, control de intensidad, encendido y apagado desde un Smartphone; elevando notoriamente el precio de éstos.

Por otra parte desarrolladores independientes han logrado crear versiones modificables de muchos dispositivos innovadores como por ejemplo impresoras 3D, maquinas CNC y cuadricopteros de electrónica libre cuyas especificaciones, diagramas esquemáticos y códigos son de acceso público ya sea de forma gratuita o de pago. Este tipo de dispositivos electrónicos, libres y modificables, son de fácil acceso para las personas con conocimientos en electrónica pero de baja capacidad adquisitiva, además estimulan la

---

<sup>1</sup> El lumen (símbolo: lm) es la unidad del Sistema Internacional de Medidas para medir el flujo luminoso, una medida de la potencia luminosa emitida por la fuente. El flujo luminoso se diferencia del flujo radiante en que el primero contempla la sensibilidad variable del ojo humano a las diferentes longitudes de onda de la luz y el último involucra toda la radiación electromagnética emitida por la fuente según las leyes de Wien y de Stefan-Boltzmann sin considerar si tal radiación es visible o no.

creatividad de la comunidad para realizar adaptaciones, generar nuevas ideas y funciones para los dispositivos. Por ello es de gran importancia la creación de éste tipo de dispositivos que hacen más inclusivo el desarrollo de tecnología en el mundo.

En este trabajo se diseñó e implementó un dispositivo de iluminación para habitaciones con uso de la tecnología LED y funciones atractivas para los usuarios que extiende el paradigma de la iluminación actual, tomando como base los dispositivos creados por Apple Inc. y Philips, pero haciéndolo de electrónica libre y fácil fabricación, de tal forma que cualquier persona con conocimientos básicos en electrónica lo puede manufacturar a bajo costo y las personas con conocimientos intermedios en electrónica pueden realizar modificaciones de hardware y software al dispositivo.

Durante el desarrollo de éste proyecto se abordará la implementación y el desarrollo de los módulos:

- Módulo de control a distancia: se encargará de enviar instrucciones mediante la tecnología inalámbrica bluetooth desde un dispositivo Android al sistema de iluminación.
- Módulo de recepción de instrucciones por bluetooth: se encargará de recibir las instrucciones y transmitir las al sistema de iluminación.
- Módulo de recepción de señales de presencia y botón de control: se encargará de generar las señales necesarias para interpretar los eventos del sensor de presencia y el pulsador.
- Módulo de potencia: se encargará de realizar las transformaciones de energía correspondientes.
- Módulo de procesamiento de instrucciones: se encargará de procesar las instrucciones que brindan las funcionalidades del sistema de iluminación.
- Módulo de iluminación: se encargará de presentar los procesos como eventos visibles mediante iluminación LED.

Para finalizar se mostrará una serie de procesos de diseño de carcasa y embalaje.

## 2 Antecedentes

### 2.1 Proyectos terminales o de integración:

2.1.1.- **Control de iluminación con tecnología LED** [2]: Este proyecto terminal busca identificar y evaluar los diferentes tipos de control para tiras LED. La propuesta se relaciona con este proyecto ya que ambos utilizarán circuitos digitales para controlar luces de tecnología LED pero la diferencia es que en este proyecto se diseñó el circuito digital y no se usan los circuitos de control existentes en el mercado.

2.1.2.- **Iluminación inteligente de escenarios** [3]: Este proyecto terminal busca controlar las luces LED de escenarios a través de un programa informático de una computadora. Se relaciona con este proyecto ya que en ambos se busca el control digital a distancia de la iluminación, pero ahora se utiliza un medio inalámbrico y no una conexión por cable, además de que el controla desde un Smartphone y no desde una computadora.

2.1.3.- **Sistema de iluminación con control inalámbrico infrarrojo basado en tecnología leds** [4]: Este proyecto terminal tiene como finalidad controlar iluminación LED de forma inalámbrica utilizando la tecnología infrarroja. Este proyecto tiene similitudes en cuanto al control de iluminación a distancia pero se utiliza tecnología bluetooth que es omnidireccional y no unidireccional como el infrarrojo.

### 2.2 Hardware y software comercial:

2.2.1.- **Bombillas Hue de Philips** [5]: Es una luz LED personal controlada a distancia por Wireless, capaz de emular cualquier color de una paleta de colores. Tiene similitud con este proyecto ya que también es iluminación controlada a distancia, pero a diferencia de Hue mi dispositivo no genera dependencia de un modem ya que funciona por bluetooth.

2.2.2.- **Luz Wi-Fi – Bombillas inteligentes** [6]: Es una bombilla de luz LED con control de intensidad y cambio de color a distancia por Smartphone. Se relaciona con mi trabajo ya que integra una funcionalidad de control de intensidad, pero se diferencia de mi proyecto ya que requiere de un modem para generar la red Wi-Fi y los procesos se llevan a cabo en el Smartphone y no en la bombilla; en mi proyecto los procesos se realizan en el dispositivo de control fijo.

2.2.3.- **Control de iluminación via eNet de Jung** [7]: Es un control de iluminación y componentes de domótica. Se relaciona con mi trabajo ya que también se enfoca en el control de iluminación a distancia y control de tiempos, pero la gran diferencia con mi proyecto es que Jung enfoca su dispositivo al control del hogar en general: persianas, luces, temperatura etc. Mi proyecto sólo se focaliza en el control de la iluminación.

### 3 Justificación

El número de dispositivos de electrónica libre es muy reducido comparado con la enorme cantidad de dispositivos privativos que se lanzan al mercado cada año. Esto provoca que muchos de los productos electrónicos de nuevo lanzamiento terminen siendo “cajas negras” para los usuarios, aún para los que tienen conocimientos en computación y/o electrónica.

El hardware y software libre ayudan a la didáctica, la innovación y el desarrollo ya que desarrolladores independientes los usan como base para desarrollar sus propios proyectos y así contribuir, al igual que desarrolladores de hardware y software privativo, al desarrollo de nuevas tecnologías.

Este proyecto sirve como una aportación a la comunidad interesada por los dispositivos de electrónica libre y también para los usuarios con conocimientos básicos en electrónica que debido a su baja capacidad adquisitiva no tienen acceso a los dispositivos como los creados por Apple Inc. y Philips, o que por hobby e interés desean una base para sumergirse en el mundo del desarrollo de dispositivos de domótica.

El dispositivo que se diseñó e implementó tiene una gama de funcionalidades más amplia que la de los dispositivos propuestos e implementados por las empresas antes mencionadas, esto con la finalidad de brindar una base suficientemente rica en posibilidades para posteriores desarrollos.

## **4 Objetivos**

### **4.1 Objetivo general:**

Construir un dispositivo de iluminación habitacional que integra tecnología LED y funcionalidades que pueden ser ejecutadas desde un dispositivo Android.

### **4.2 Objetivos particulares:**

- Diseñar e implementar la aplicación de envío de instrucciones para el sistema operativo Android.
- Diseñar e implementar el circuito de mando y de potencia.
- Diseñar e implementar el código de control para la recepción y reenvío de instrucciones por bluetooth.
- Diseñar e implementar el código del sistema para el análisis, ejecución y administración de instrucciones.
- Diseñar e implementar los componentes de iluminación independientes.

## 5 Marco teórico

La electrónica y la computación han llegado para quedarse; y con ellas los circuitos electrónicos que son la base de la mayoría de las tecnologías en el mundo. ¿Te imaginas no contar con estas tecnologías?, es decir, no tener una computadora para los sistemas de información de tu negocio o un Smartphone para comunicarte con tus familiares ante una emergencia. La electrónica y la computación son parte de nuestras vidas, por ello es necesario apostar por la creación de este tipo de herramientas para el futuro pero para ello debemos conocer los conceptos, técnicas y leyes que rigen el mundo de la electrónica y la informática.

### 5.1 Electrónica

La electricidad se puede comprender como un tipo de energía la cual, mediante una manipulación correcta, puede generar efectos mecánicos, luminosos, caloríficos y químicos. La manipulación de ésta energía mediante dispositivos que son capaces de recibir, procesar o enviar información para el uso de la industria se le conoce como **electrónica**.

### 5.2 Circuito eléctrico

Existen múltiples componentes capaces de manipular la electricidad, éstos se organizan dentro de un sistema para lograr un objetivo general. A este sistema se le conoce como **circuito eléctrico**.

Un circuito eléctrico básico consta de un mínimo de cuatro elementos disponibles en la Figura 1, estos permiten el flujo y la utilización de los electrones:

1. La fuente de energía.
2. Los conductores
3. La carga.
4. El dispositivo de control

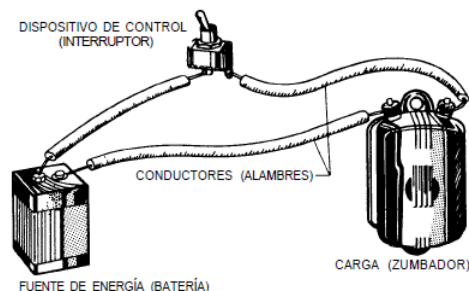


Figura 1. Ejemplo de los elementos de un circuito eléctrico básico.



### 5.3 La corriente, el voltaje y la resistencia

Para entender cómo se realiza la manipulación de la energía, en específico los electrones se deben comprender algunas variables de interés, por ejemplo: **el voltaje** es la fuerza electromotriz que permite que los electrones se muevan a través de un circuito eléctrico, se mide en volts. **La corriente** es la cantidad de electrones que se mueven en un segundo dentro de un circuito eléctrico, el movimiento de electrones se muestra en la Figura 2 y se mide en amperes. Un ampere corresponde al movimiento de  $6.28 \times 10^{18}$  electrones en un segundo. Y **la resistencia** es la oposición al flujo de corriente que se genera por los choques entre electrones, esto reduce el número de electrones que se mueven a través un conductor y su unidad de medida es el ohm ( $\Omega$ ).

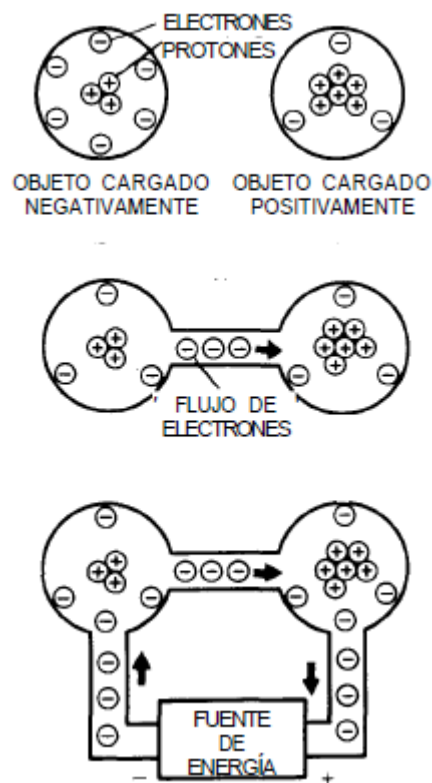


Figura 2. Movimiento de los electrones en un circuito.

### 5.4 Tipos de corriente y frecuencia

La corriente (flujo de electrones) puede ser de dos tipos: **corriente continua** y **corriente alterna**. La corriente continua se produce cuando un circuito tiene una fuente de voltaje constante, es decir, las cargas en los polos negativo y positivo no cambian a través de tiempo. Y la corriente alterna se produce cuando un circuito tiene una fuente de voltaje cuya polaridad de las cargas en los polos negativo y positivo se invierten en un determinado tiempo. Al número de veces que un ciclo de corriente alterna puede completarse en un segundo se le llama **frecuencia** y se mide en herts (Hz).

## 5.5 Potencia

Otro término interesante es el de **potencia**, hace referencia a la velocidad con la que se realiza un trabajo en un circuito eléctrico, es decir, es la velocidad con la que la energía puede cambiar de forma. Su unidad de medida es el Watt (W).

## 5.6 Ley de ohm

Todos hemos escuchado hablar de **la ley de Ohm**. Esta ley describe formalmente la relación entre las variables voltaje, corriente, energía y potencia. Dentro de los circuitos eléctricos, cuando los electrones han salido de la fuente, la única oposición al flujo de estos es la resistencia, además, este flujo tiene un determinado voltaje y corriente, por ello existen relaciones entre ellas. Estas relaciones, según la ley de ohm, se determinan de la siguiente forma:

1. El voltaje necesario para establecer cierta intensidad de corriente a través de un circuito es igual al producto de la corriente y de la resistencia del circuito como se muestra en la Ecuación 1.

$$V = I \times R \text{ (Ecuación 1)}$$

Donde:

V= voltaje, I= corriente y R= resistencia.

2. La intensidad de corriente de un circuito es igual al voltaje que se aplica al circuito, dividido entre la resistencia del circuito como se muestra en la Ecuación 2.

$$I = V / R \text{ (Ecuación 2)}$$

3. La resistencia de un circuito es igual al voltaje que se aplica al circuito, dividido entre la cantidad de corriente del circuito como se muestra en la Ecuación 3.

$$R = V / I \text{ (Ecuación 3)}$$

4.- La potencia de un circuito es igual al voltaje que se aplica al circuito, multiplicado por la cantidad de corriente del circuito como se muestra en la Ecuación 4.

$$P = V \times I \text{ (Ecuación 4)}$$

El flujo de la corriente se puede dar de dos formas, dependiendo del tipo de circuito: circuito en serie o circuito en paralelo.

## 5.7 Circuito en serie y paralelo

En el **circuito en serie**, mostrado en la Figura 3, los elementos se conectan uno después de otro. De ésta forma sólo existe una ruta por la que los electrones pueden circular. Se utiliza para controlar y proteger sistemas eléctricos. En un circuito de éste tipo la suma de las caídas de voltaje en el circuito es igual al voltaje aplicado, este hecho se conoce como **la ley de voltaje de Kirchhoff**.

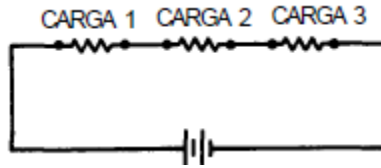


Figura 3. Circuito en serie.

En un **circuito en paralelo**, mostrado en la Figura 4, las cargas se conectan de forma ramificada, si una de las ramas se desconecta o abre las ramas restantes continuaran operando [8].

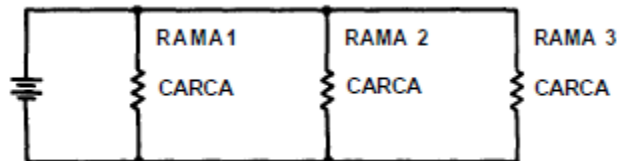


Figura 4. Circuito en paralelo.

## 5.8 Componentes de circuitos eléctricos

Dentro de estos circuitos podemos encontrar algunos componentes electrónicos que determinan su funcionamiento, por ejemplo:

- **El diodo rectificador.** Es un dispositivo semiconductor que permite el paso de la corriente eléctrica en una única dirección.
- **El diodo emisor de Luz (LED).** Es un dispositivo semiconductor que transforma la energía eléctrica en luz de un espectro determinado cuando se polariza de forma correcta.
- **El resistor.** Es un componente eléctrico diseñado para introducir una resistencia eléctrica determinada entre dos puntos de un circuito eléctrico. Comúnmente se conocen como resistencias.
- **El transistor.** Es un dispositivo electrónico especialmente diseñado para entregar una señal de salida en respuesta a una señal de entrada, puede cumplir las funciones de conmutador, oscilador o amplificador.

Estos componentes han revolucionado la industria de la electrónica y han dado lugar a la aparición de otras disciplinas como la electrónica digital.

## 5.9 Electrónica digital

La **electrónica digital** es la rama más moderna de la electrónica y que evoluciona más rápidamente. Su aplicación es la construcción de dispositivos electrónicos en los que la información está codificada en estados discretos. Los enormes avances tecnológicos han logrado que se desarrollen componentes electrónicos mucho más avanzados, capaces de

realizar tareas complejas y formar parte de sistemas robustos que, a su vez, logran más avances tecnológicos. Este es el efecto domino de las tecnologías actuales [9].

Algunos componentes recientemente creados con electrónica digital son, por ejemplo:

### 5.9.1 Los sensores

Son objetos capaces de detectar magnitudes físicas o químicas y transformarlas en variables eléctricas. Las magnitudes físicas pueden ser:

- Intensidad lumínica.
- Temperatura.
- Distancia.
- Aceleración.
- Inclinación.
- Presión.
- Fuerza.
- Unidad.
- Movimiento.
- Torsión.

### 5.9.2 Los microcontroladores

Son circuitos integrados programables, capaces de ejecutar las ordenes grabadas en su memoria. Contienen en su interior tres unidades funcionales:

- Unidad central de procesamiento.
- Memoria.
- Periféricos de entrada y salida.

**El microcontrolador AVR atmega 328p** por ejemplo, es un microcontrolador de la marca ATMEL que funciona a 8 bits y está integrada por una memoria principal de 2kb, 32kb de memoria flash, 32 registros de propósito general y 23 puertos de entrada y salida; además de tener la capacidad de correr con un ciclo de reloj máximo de 20Mhz.

### 5.9.3 Los transmisores

Son instrumentos que captan las señales y son capaces de transmitir las a distancia hacia un receptor para la transferencia de información. Pueden ser de varios tipos e implementar distintas tecnologías como:

- Wifi. [10]
- Bluetooth. [11]
- Radio frecuencia determinada.

- Zigbee.
- Lifi.

#### 5.9.4 Los módulos digitales

Son dispositivos que se encargan de aprovechar la unión de varios dispositivos electrónicos y realizar tareas específicas. Su objetivo puede ser de distintos tipos:

- **Módulos de control.**  
El **módulo de control Arduino Uno R3** por ejemplo, es una plataforma de prototipos electrónicos basados en hardware y software de código abierto. Integra un microcontrolador de la marca ATMEL atmega328p a una velocidad de 16 MHz [13].
- **Módulos de iluminación.**  
**Los módulos led smd 5050(tiras led)**, son un conjunto de leds de montaje superficial organizados en un circuito recortable y auto-adherible.
- **Módulos de transmisión.**  
El **módulo bluetooth HC-06** por ejemplo, es un transmisor de tecnología bluetooth V2.0 diseñado para trabajar como servidor configurable mediante comandos AT (parecidos a los comandos de una terminal “tonta”) y con una frecuencia de 2.4 GHz, con un alcance de 10 metros a un voltaje de 3.3v.
- **Módulos con sensores.**  
El **módulo PIR HC-SR501** por ejemplo, es un detector de presencia que utiliza un sensor de temperatura y analizar sus cambios para detectar la presencia de una persona dentro de un área determinada.

Algunos de los sistemas electrónicos más complejos creados hasta el momento son:

- **Las computadoras.** Son máquinas electrónicas capaces de almacenar información y tratarla automáticamente mediante operaciones matemáticas y lógicas controladas por programas informáticos.
- **Los teléfonos inteligentes o Smartphone.** Son dispositivos semejantes a minicomputadoras y con una mayor conectividad que un teléfono móvil convencional, el término “inteligente” se utiliza con fines comerciales y hace referencia a la capacidad de usarse como un computador personal.

Estos dispositivos son tan complejos que hace falta comprender muchas cosas antes de poder entender cómo es que se diseñan y construyen, en un principio podemos destacar dos partes esenciales: El hardware y el software.

#### 5.10 Hardware

Dentro de un sistema digital o informático, es el conjunto de elementos tangibles o físicos (procesador, dispositivos de almacenamiento permanente, memoria principal, etc.).

### 5.10.1 Desarrollo de hardware

Para cada uno de los componentes electrónicos que utilicemos, podemos usar internet para encontrar su **documentación (datasheet)** que el fabricante pone a disposición pública y así saber los detalles pertinentes acerca de la especificación técnica de determinado componente. Esto facilita la utilización de un componente dentro de un circuito electrónico y reduce el tiempo de diseño ya que no es necesario probar los componentes uno a uno para saber acerca de su funcionamiento.

Para el diseño de circuitos electrónicos existe software que nos permiten desarrollar diagramas, simular el funcionamiento del circuito, detectar errores y diseñar un circuito impreso. Esto permite crear circuitos más confiables y que se pueden probar de forma virtual antes de fabricarlos. **EasyEda** por ejemplo, es un entorno web para el desarrollo de circuitos electrónicos, placas de circuito impreso y simulación basado en la web y la nube.

Existen distintas normas que indican como se debe realizar el desarrollo de **circuitos impresos** y como se pueden evitar errores que no son predecibles mediante los sistemas de diseño de circuitos, además de incluir metodologías para la realización de circuitos impresos ya sea de forma rudimentaria o de forma industrial. También existen sistemas de software que permiten realizar diseños y modelado en 3D para la presentación del producto, algunos de ellos brindan secuencias de control numérico para maquinaria industrial que se encargará de la fabricación física. **UNE** por ejemplo, son un conjunto de normas básicas para el diseño de placas de circuito impreso.

## 5.11 Software

Dentro de un sistema digital o informático, hace referencia a los códigos o programas que permiten el uso y administración de los dispositivos de hardware. Es la parte intangible de un computador o Smartphone, como los sistemas operativos. **Android** por ejemplo, es un sistema operativo basado en el núcleo Linux, diseñado para dispositivos móviles con pantalla táctil; este sistema operativo es propiedad de google inc.

Los sistemas informáticos actuales son complejos, por ello han surgido metodologías, normas, estándares, técnicas, documentos y herramientas que nos permiten sobrellevar la gran carga de trabajo y abstracción que la construcción de un sistemas electrónicos avanzado requiere.

### 5.11.1 Lenguajes de programación

Para el desarrollo de programas existen **lenguajes de programación** que permiten, mediante un lenguaje formal, describir procedimientos, actividades o acciones que un sistema informático o digital debe realizar. La mayoría de lenguajes de programación integran un conjunto de instrucciones cuyo léxico es similar al lenguaje común para hacerlo más entendible para el programador y al conjunto de instrucciones se le suele llamar **código fuente**. Algunos de los más conocidos son:

- **Java** es un lenguaje de programación de propósito general y orientado a objetos; los programas generados con este lenguaje requieren de una máquina virtual lo que permite que el mismo programa pueda correr sobre diferentes dispositivos.
- **C** es un lenguaje de programación estructurado que a pesar de ser de alto nivel permite usar instrucciones de muy bajo nivel. Este lenguaje de programación está orientado a la implementación de sistemas operativos.

### 5.11.2 Entornos de desarrollo integrados

Existen **entornos de desarrollo integrados (IDE)** que son aplicaciones que proporcionan servicios para facilitar el desarrollo de software. Por ejemplo:

- **IDE Arduino** es un entorno de desarrollo de sistemas para microcontroladores bajo la plataforma de hardware libre Arduino y permite, mediante el uso de un compilador y un depurador, la programación en un lenguaje propio de Arduino basado en **Wiring**<sup>2</sup> (pequeña arquitectura de hardware parecida a la de un computador personal) y el lenguaje de programación C.
- **APP Inventor 2** es un entorno de desarrollo basado en programación orientado a eventos, hace uso de la librería Open Blocks de Java y permite la creación de aplicaciones para dispositivos con el sistema operativo Android.

Se han creado **metodologías de desarrollo de software** que ayudan a identificar los elementos clave que deben incluirse en el desarrollo de un determinado programa además se han descrito las reglas necesarias para el trabajo en equipo y brindar soporte para la generación de la documentación necesaria para el desarrollo de proyectos. Por ejemplo:

### 5.11.3 Los diagramas de flujo

Nos permite abstraer de forma gráfica los elementos de un proceso o algoritmo para poder ser comprendido y posteriormente programado. **LucidChart** por ejemplo, es un entorno web que permite utilizar una versión de prueba (30 días) o de pago para realizar diagramas de flujo que permitan abstraer un procedimiento antes de programarlo.

### 5.11.4 Los casos de uso de texto,

Es una metodología enfocada al descubrimiento de los requerimientos de un sistema. En específico proveen el entendimiento de las funciones de un escenario principal de éxito y el análisis de los flujos alternativos, es decir, nos permite saber lo que realmente se necesita en un sistema complejo.

---

<sup>2</sup> Es una micro-arquitectura que provee las condiciones necesarias para que un microcontrolador pueda funcionar.

### 5.11.5 Paradigmas de programación

Por otra parte han surgido **paradigmas de programación** que son distintos enfoques o filosofías acerca de cómo generar soluciones de software. Nos brindan una forma específica de abstraer los elementos del problema así como la forma de relacionarlos e integrarlos en una solución. Un paradigma de programación puede determinar la estructura y abstracción de uno o varios lenguajes de programación por ejemplo:

- IDE de Arduino hace uso de **programación estructurada** que considera una ejecución secuencia de varias subrutinas lo cual permite mejorar la calidad, claridad y tiempo de desarrollo.
- IDE App Inventos 2 hace uso de **programación orientada a eventos** en la que el flujo de ejecución del programa está determinado por los sucesos que ocurran en el sistema y definidos por el usuario.

La mayoría de las tecnologías informáticas actuales se desarrollan utilizando distintos paradigmas y lenguajes de programación.



## 6 Desarrollo del proyecto

De forma general se tendrán tres dispositivos<sup>3</sup>: un dispositivo Android (contendrá la aplicación de control), un dispositivo de control fijo y un dispositivo de iluminación.

De forma general la interacción entre los dispositivos sería como en la Figura 5.

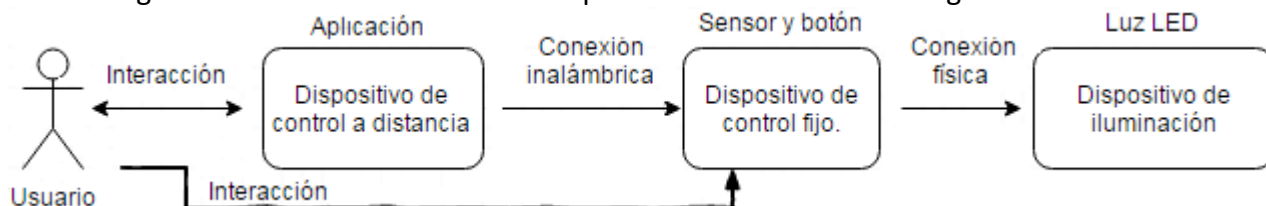


Figura 5. Descripción general de la interacción entre los dispositivos.

Se tendrán 4 modos de funcionamiento:

- Modo 1: El sistema de mantendrá las iluminaciones apagadas en espera de instrucciones desde el dispositivo de control a distancia.
- Modo 2: El sistema encenderá la iluminación a un 75%
- Modo 3: El sistema encenderá la iluminación a un 40%
- Modo 4: El sistema encenderá por la acción el sensor de movimiento.

Todos los modos de funcionamiento serán accesibles desde el dispositivo de control fijo y desde el dispositivo de control a distancia y en conjunto brindarán las siguientes funciones al dispositivo general:

- Encendido/apagado progresivo (manual y mediante aplicación Android).
- Almacenamiento de intensidad preferida.
- Encendido completo o encendido parcial.
- Control de intensidad mediante aplicación Android.
- Programación de apagado automático mediante aplicación Android.
- Efectos de iluminación activables desde la aplicación Android.
- Encendido mediante sensor de movimiento.
- Control de estados mediante aplicación Android y mediante el control fijo.
- Simulación de estancia.

Para lograr estas funcionalidades se implementarán y desarrollarán los siguientes módulos:

- Módulo de control a distancia (aplicación Android).
- Módulo de recepción de instrucciones por bluetooth.
- Módulo de recepción de señales de presencia y botón de control.
- Módulo de potencia.

<sup>3</sup>Para fines prácticos: con *dispositivo* **no** nos referimos a un módulo sino a un conjunto de uno a más módulos del proyecto.

- Módulo de procesamiento de instrucciones.
- Módulo de iluminación.

El dispositivo de control fijo visto como “caja negra” tendrá una estructura como la de la Figura 6.

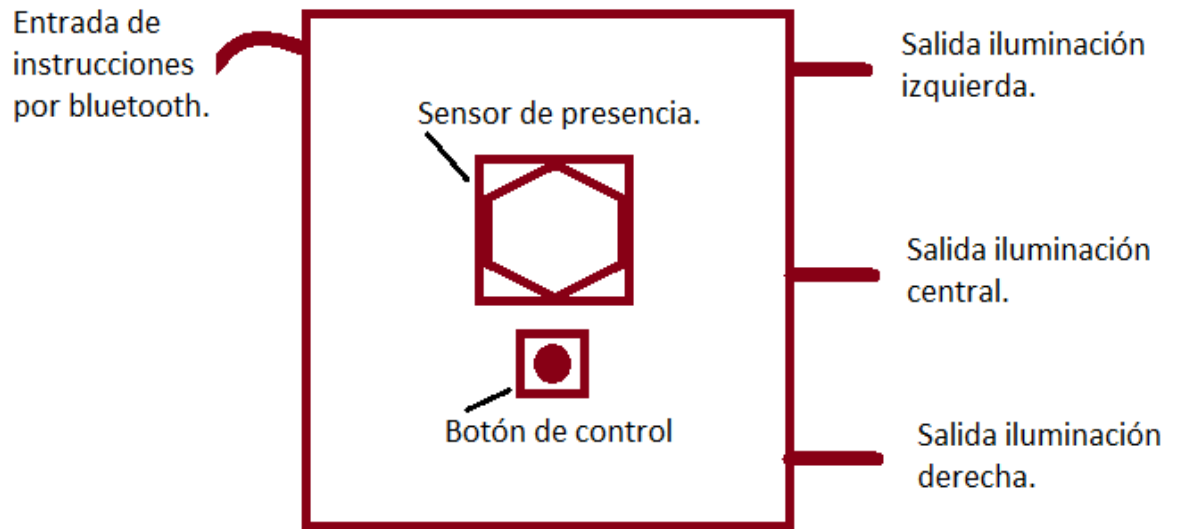


Figura 6. Descripción gráfica del dispositivo de control fijo.

El desarrollo de un proyecto debe seguir un riguroso orden para sobrellevar la carga de trabajo que éste implique. Por ello se decidió realizar un diseño previo para cada módulo el cual nos permita identificar requerimientos y consideraciones antes de la implementación.

### 6.1 El módulo de control a distancia

Su objetivo es enviar instrucciones u órdenes de iluminación de forma inalámbrica a partir de las acciones realizadas por el usuario sobre la aplicación como se muestra en la Figura 7.

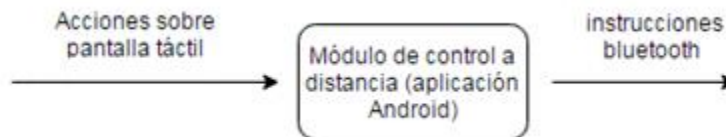


Figura 7. Transformación de acciones sobre la pantalla táctil a instrucciones bluetooth mediante el módulo de control a distancia.

Una vez que el módulo de control a distancia se cargue a un Smartphone Android dará lugar a un dispositivo de control a distancia similar a lo que se ve en la Figura 8.

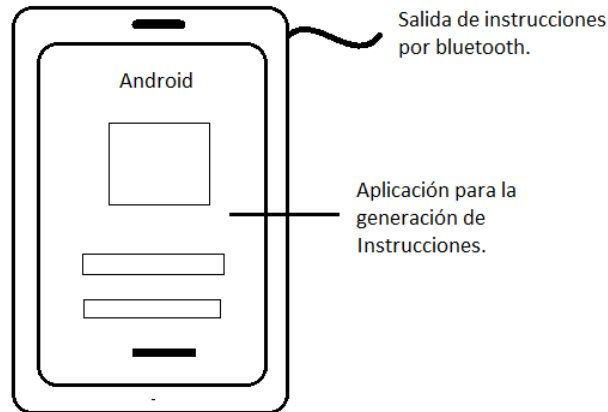


Figura 8. Descripción gráfica del dispositivo de control a distancia.

El módulo de control a distancia (aplicación Android) se realizó en dos etapas, la etapa de diseño y la etapa de implementación.

### 6.1.1 Diseño

Se utilizó la herramienta LucidChart para diseñar el diagrama de flujo. Se adoptó una metodología iterativa, es decir, se realizó el primer diagrama de flujo como en la Figura 9, posteriormente se revisó el diagrama de flujo para identificar flujos no deseados o bucles infinitos para su posterior corrección. Se realizaron un total de cuatro iteraciones, cada una con cambios menores. Recordemos que el número de iteraciones de revisión de un diagrama de flujo dependerá de la experiencia o habilidad de la persona que lo realiza.

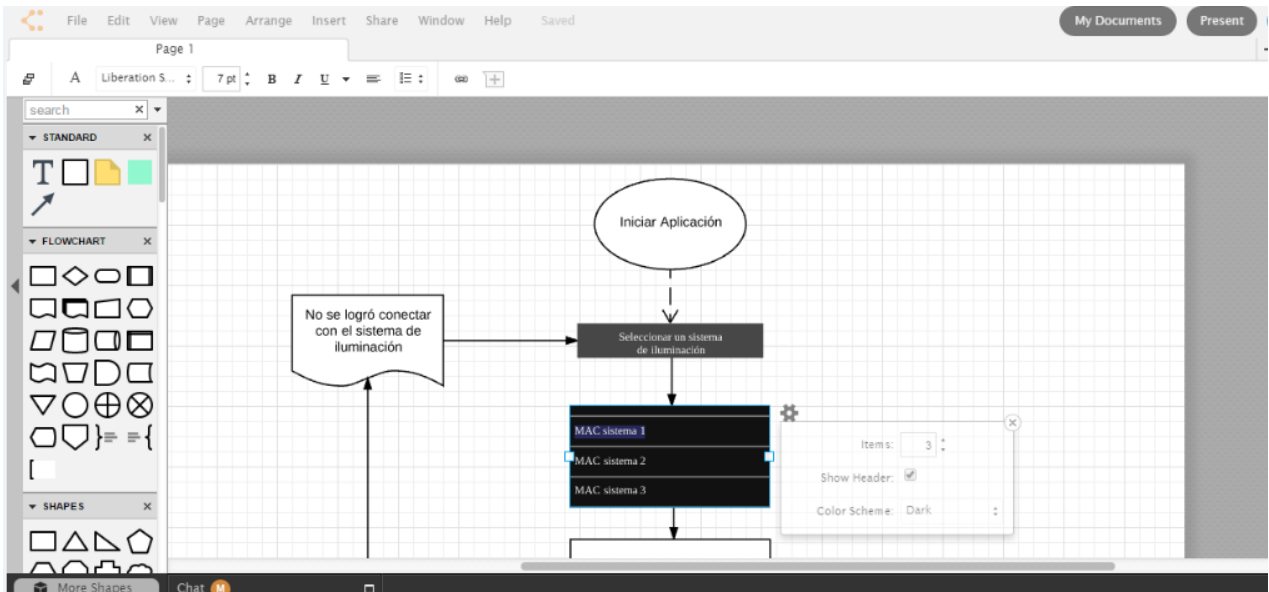
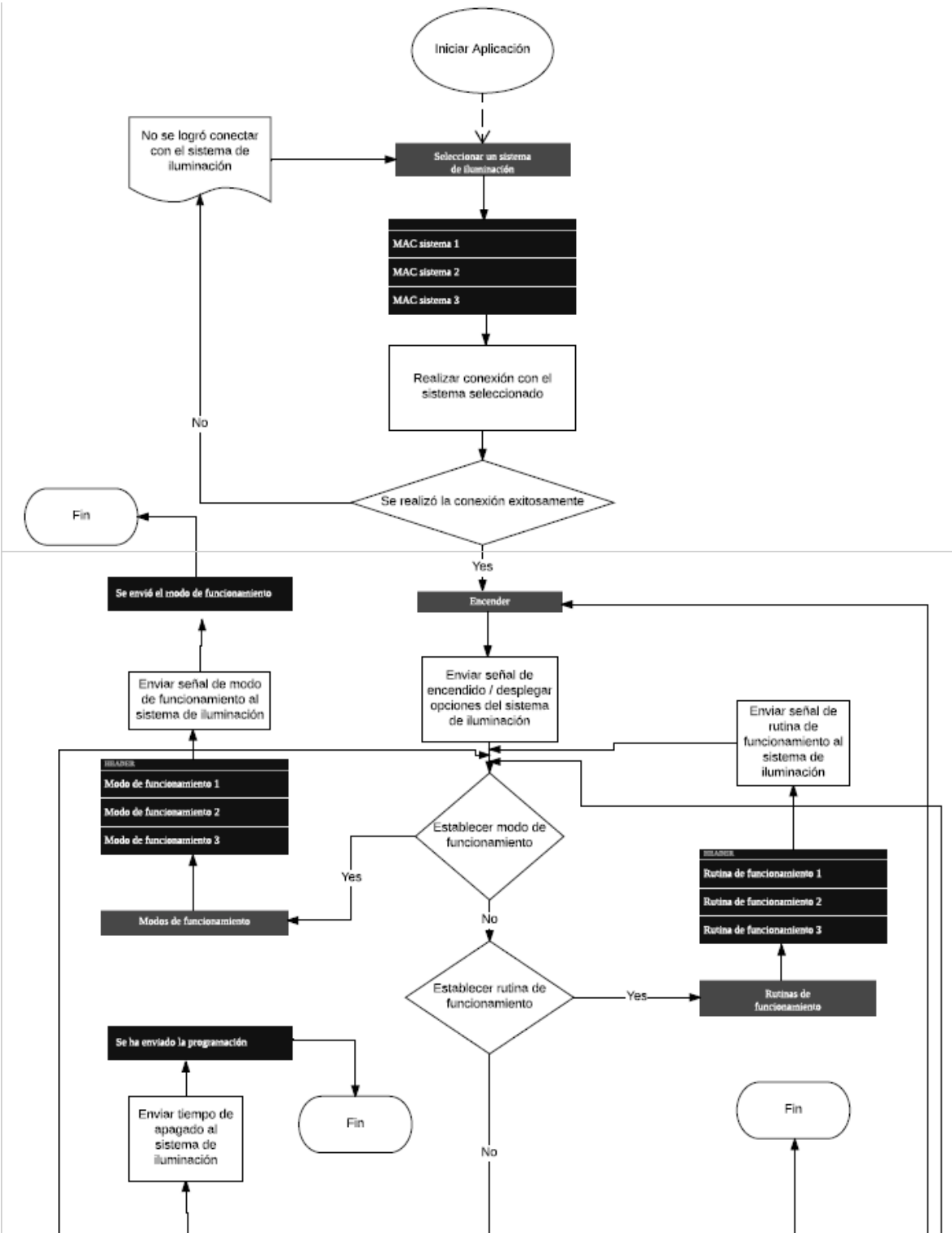


Figura 9. Etapa de diseño (diagrama de flujo) para el módulo de control a distancia.

La etapa de diseño del módulo de control a distancia fue de aproximadamente 10 horas y se obtuvo el diagrama de flujo de la Figura 10.



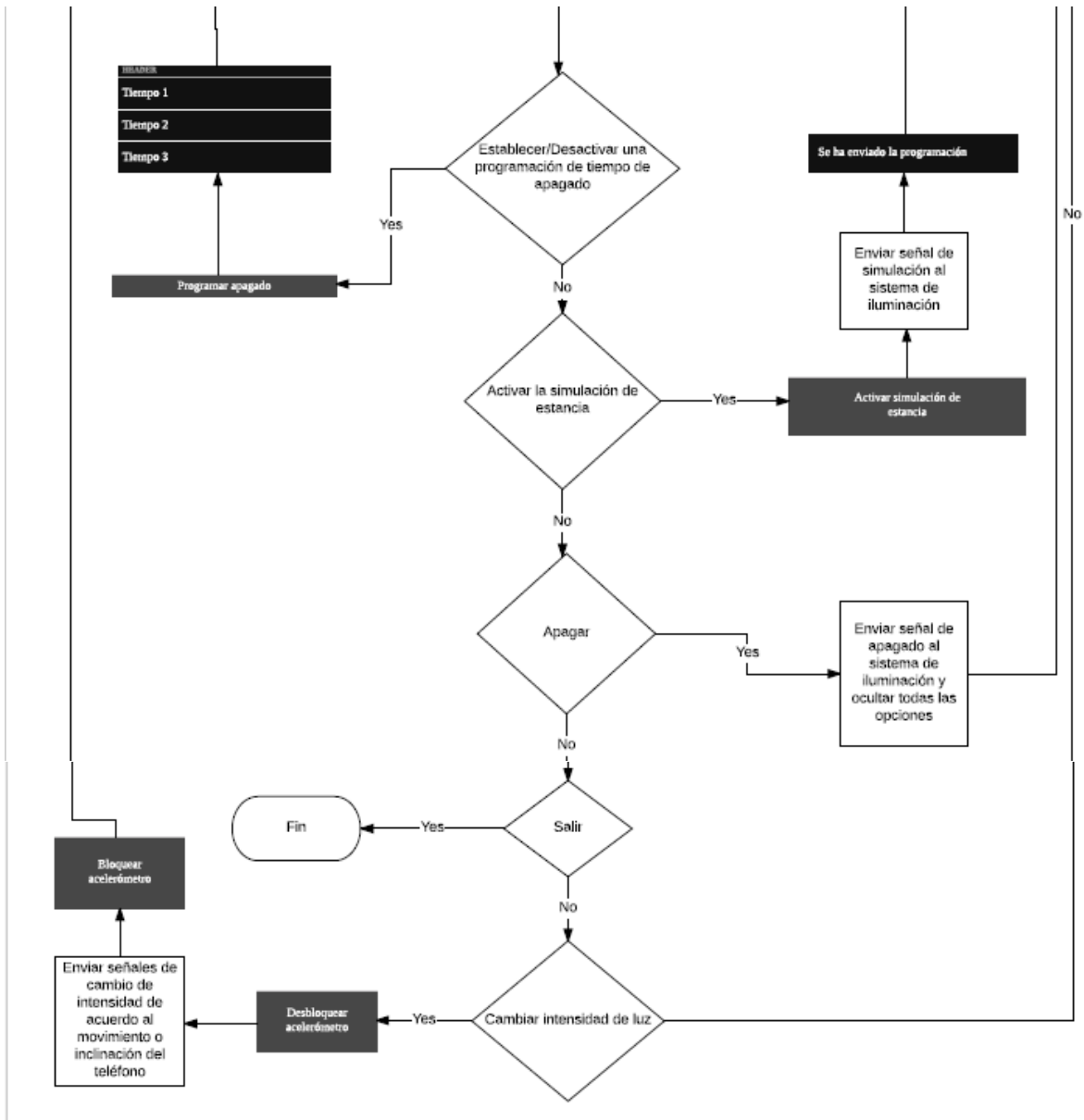


Figura 10. Diagrama de flujo del módulo de control a distancia.

### 6.1.2 Implementación

Se utilizó el patrón de arquitectura de software de Modelo-Vista-Controlador (MVC) con el entorno de desarrollo integrado App Inventor 2 con programación orientada a eventos.

- **El modelo:** cada línea de transición del diagrama de flujo se consideró como un evento programable y cada elemento se convirtió en un procedimiento determinado como se muestra en las Figuras 11 y 12.

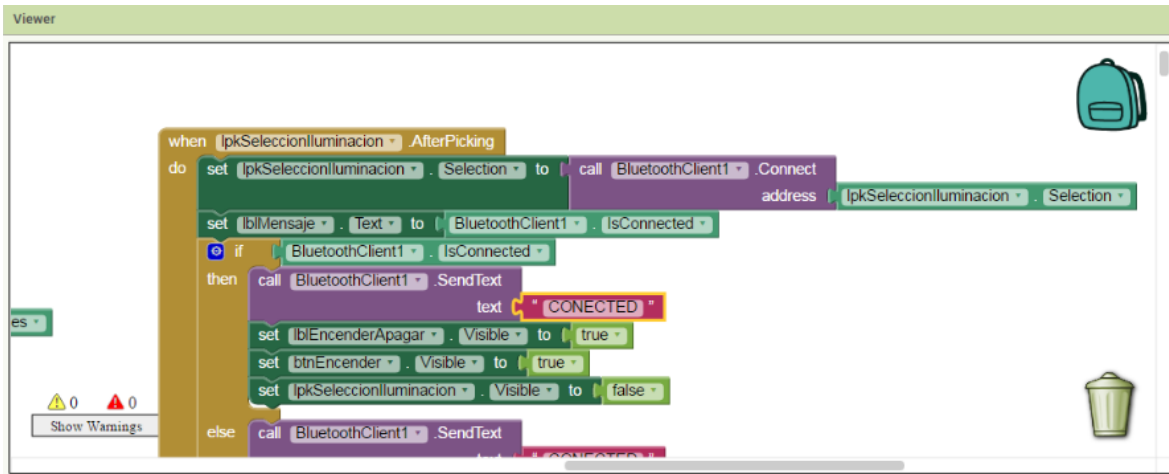


Figura 11. Evento para seleccionar un sistema de iluminación y disparar un procedimiento correspondiente.

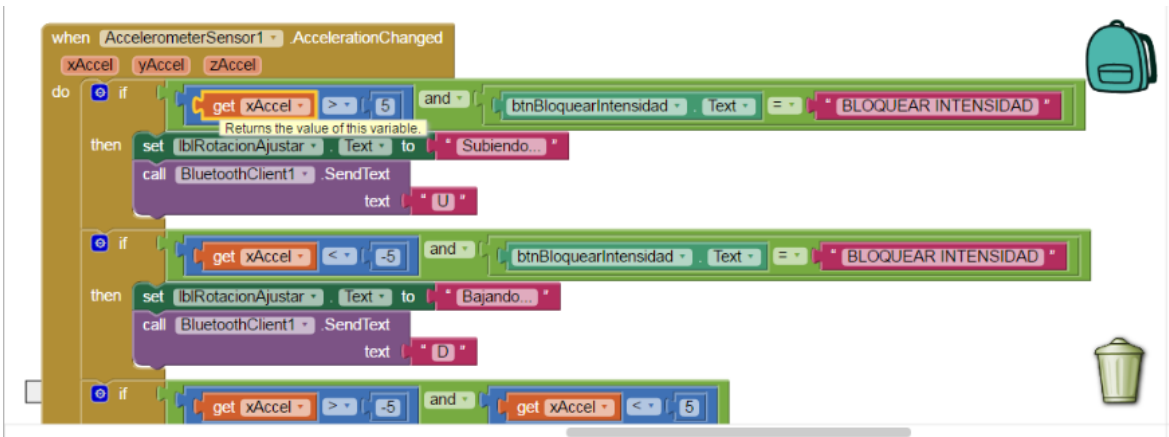


Figura 12. Eventos que se disparan al cambiar las coordenadas del acelerómetro.

- **El controlador:** se agregaron al proyecto los controladores de bluetooth, de acelerómetro y de notificaciones como se muestra en la Figura 13.

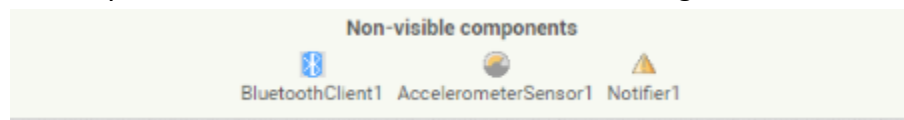


Figura 13. Controladores de bluetooth, de acelerómetro y de notificaciones agregados al proyecto.

- El controlador de bluetooth nos permite acceder al recurso de hardware para la transmisión de información desde el dispositivo.
- El controlador de acelerómetro nos permite acceder al recurso de hardware del acelerómetro y leer los datos de éste.
- El controlador de notificaciones nos permite acceder a las notificaciones del sistema operativo Android.

- **La vista:** se utilizaron los elementos estándar del sistema operativo Android como botones, listas, textos, etc. con la finalidad de facilitar su comprensión para los usuarios como en la Figura 14. Las imágenes utilizadas se descargaron del repositorio libre “pixabay.com” y se tomaron las siguientes consideraciones de programación visual:
- Se evitó lo más posible el uso del color blanco que es color más brillante en la pantalla y puede llegar a cansar la vista.
  - Se evitó colocar elementos que puedan confundir al usuario como imágenes no relacionadas con la acción del botón o mensajes confusos.
  - Se colocó el color rojo, a pesar de estar contra indicado para aplicaciones de uso recurrente ya que nos sirve para indicar de forma intuitiva un bloqueo de la intensidad. Se procuró utilizar el rojo en un tono oscuro para disminuir el impacto en ojos del usuario.
  - Se cuidó que todos los elementos funcionen de forma habitual como en todas las aplicaciones móviles para disminuir la curva de aprendizaje.
  - Se incorporó prevención de errores mediante mensajes de texto en partes estratégicas de la aplicación para evitar que el usuario pierda el flujo de ejecución.

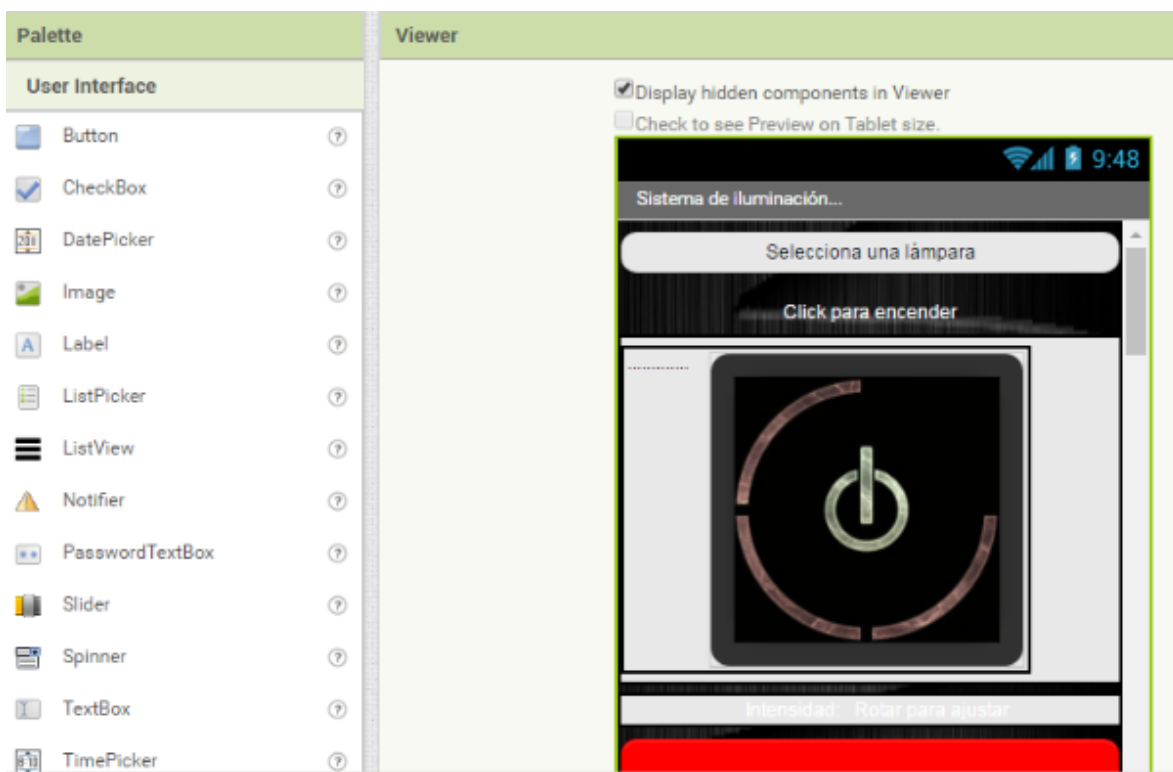


Figura 14. Implementación de la vista en el IDE App Inventor 2.

La incorporación completa de los tres elementos (Modelo-Vista-Controlador) nos brinda el funcionamiento completo de la aplicación. Finalmente se utilizó el compilador de App Inventor 2 para generar un archivo APK instalable en cualquier sistema operativo Android (Tablet o Smartphone) versión 2.4 o superior. Ésta aplicación se instaló de forma habitual en un Smartphone con Android 2.4.

La implementación del módulo de control a distancia duró aproximadamente 36 horas.

## 6.2 El módulo de recepción de instrucciones por bluetooth

Este módulo validará la conexión con el módulo de control a distancia y posteriormente recibirá las instrucciones enviadas de forma inalámbrica para repetirlas a un medio físico hacia el módulo de procesamiento de instrucciones como se ve en la Figura 15.



Figura 15. Descripción general del módulo de recepción de instrucciones por bluetooth.

Este módulo consta de dos partes: hardware y software.

### 6.2.1 La parte de hardware

Se utilizó el módulo bluetooth HC-06, mostrado en la Figura 16, el cual se debe programar para poder definir su funcionamiento como servidor.

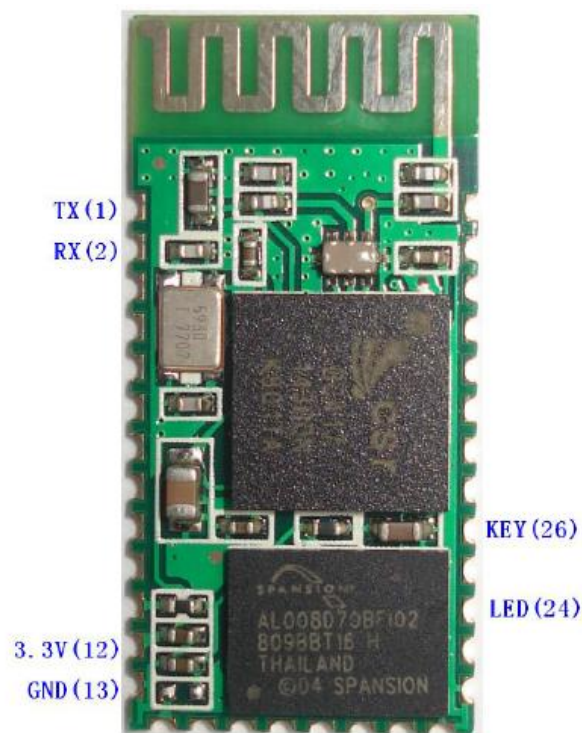


Figura 16. Módulo bluetooth HC-06.

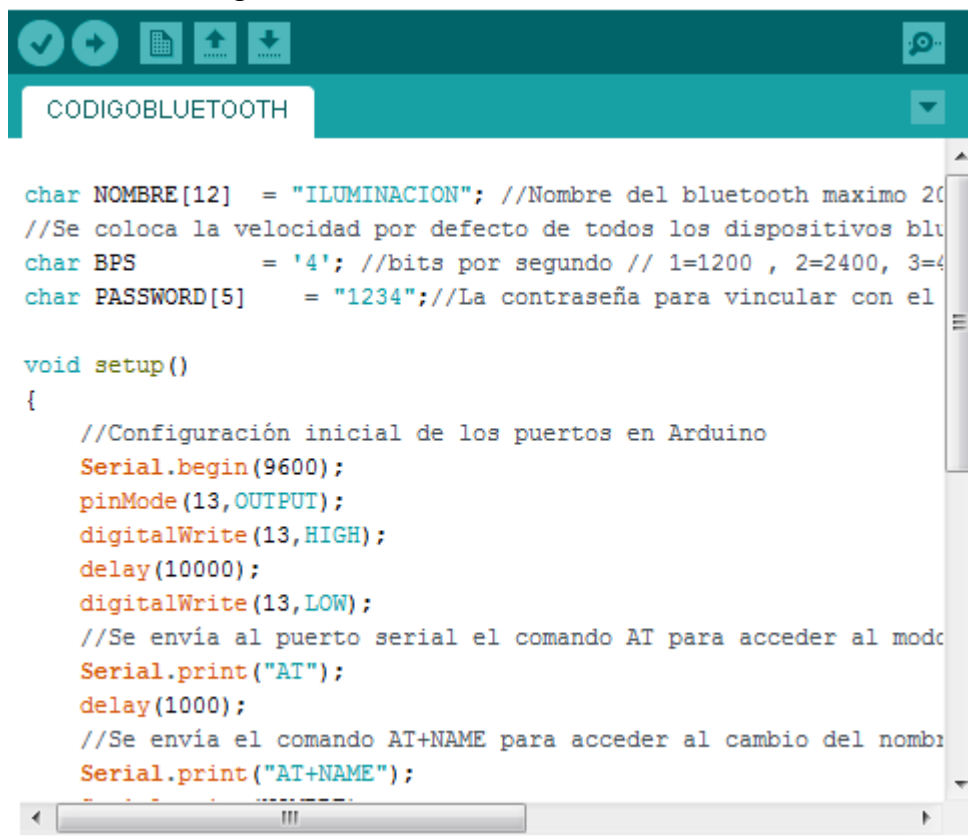


## 6.2.2 La parte de software

Se utilizó el entorno de desarrollo de Arduino para generar el programa necesario que configura el módulo bluetooth como servidor y también los parámetros necesarios para su identificación y funcionamiento: velocidad de ciclo de reloj, nombre del dispositivo bluetooth y contraseña de acceso. La programación se realizó mediante comandos AT<sup>4</sup>.

Al igual que una “terminal tonta” puede cargar la configuración a un router en redes, un módulo de Arduino uno R3 puede hacer la función de la “terminal tonta” y cargar la configuración al módulo bluetooth HC-06. Para ello se realizó el siguiente procedimiento:

1. Se desarrolló el programa de envío de configuración en el IDE de Arduino mostrado en la Figura 17.

The image shows a screenshot of the Arduino IDE interface. At the top, there is a toolbar with icons for checking, running, saving, and uploading. Below the toolbar, the file name 'CODIGOBLUETOOTH' is visible. The main area contains C++ code for an Arduino sketch. The code defines variables for the device name, baud rate, and password, and includes a setup function that configures the serial port and sends AT commands to the Bluetooth module.

```
char NOMBRE[12] = "ILUMINACION"; //Nombre del bluetooth maximo 20
//Se coloca la velocidad por defecto de todos los dispositivos blu
char BPS      = '4'; //bits por segundo // 1=1200 , 2=2400, 3=4
char PASSWORD[5] = "1234"; //La contraseña para vincular con el

void setup()
{
    //Configuración inicial de los puertos en Arduino
    Serial.begin(9600);
    pinMode(13,OUTPUT);
    digitalWrite(13,HIGH);
    delay(10000);
    digitalWrite(13,LOW);
    //Se envía al puerto serial el comando AT para acceder al modo
    Serial.print("AT");
    delay(1000);
    //Se envía el comando AT+NAME para acceder al cambio del nombre
    Serial.print("AT+NAME");
```

Figura 17. Desarrollo del programa de envío de instrucciones.

2. Se depuró y cargó el programa desarrollado en la arquitectura de Arduino Uno R3 mediante la interfaz USB como se muestra en la Figura 18.

<sup>4</sup> Los **comandos AT (Attention)** son instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal modem.



Figura 18. Conexión del Arduino a la Interfaz USB.

3. Se conectó el puerto serial RX del módulo bluetooth al puerto serial TX del Arduino y viceversa como se muestra en la Figura 19.
4. Se alimentó el módulo bluetooth con la fuente de 5V que brinda el Arduino como se muestra en la Figura 19.

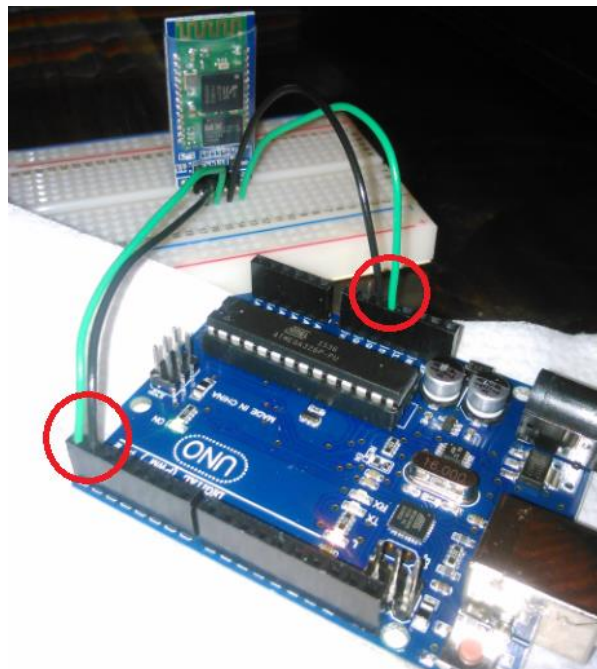


Figura 19. Conexión de los puertos seriales y alimentación entre el módulo bluetooth y Arduino.

5. Se envió la configuración al módulo bluetooth a través de las interfaces de los puertos seriales del Arduino.

El diseño e implementación del módulo de recepción de instrucciones por bluetooth duró aproximadamente 6 horas.

### 6.3 Para el módulo de recepción de señales de presencia y botón de control

Su función es identificar presencia<sup>5</sup> en la habitación o pulsaciones del botón de control y transformarlas en señales que se enviarán por medio físico al módulo de procesamiento de instrucciones como en la Figura 20.

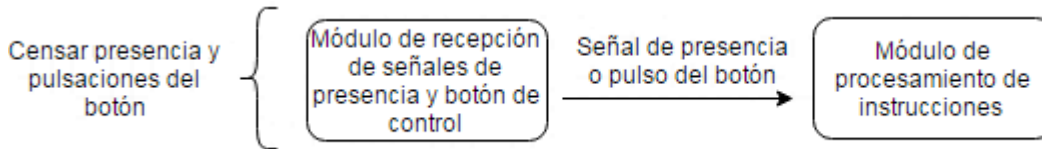


Figura 20. Detección de presencia o actividad del botón de control.

Éste módulo es un dispositivo de hardware que se desarrolló en tres etapas: etapa de diseño del circuito electrónico, etapa de diseño del circuito impreso y etapa de construcción del circuito impreso.

#### 6.3.1 Diseño del circuito electrónico

Se consideraron los siguientes componentes electrónicos para el circuito:

- **Pulsador (botón):** A nivel técnico nos permite introducir un bit 1 al dispositivo general el cual disparará una interrupción en el sistema. Se muestra en la Figura 21.



Figura 21. Componente electrónico: pulsador normalmente abierto.

---

<sup>5</sup>Una detección de presencia se refiere a un cambio de movimiento y calor dentro de una habitación. Esto se hace mediante un sensor PIR de los muchos disponibles en el mercado.

- **Sensor PIR HC-sr501 (sensor de presencia):** A nivel técnico nos permite detectar la presencia de una persona en la habitación y como respuesta al evento enviar un bit 1 al dispositivo el cual disparará una interrupción en el sistema. Mostrado en la Figura 22.



Figura 22. Sensor Pir HC-Sr501.

Nota: El módulo de recepción de instrucciones por bluetooth también va conectado a éste circuito, pero debido a su complejidad se decidió abordarlo como un módulo independiente (el módulo de recepción de instrucciones por bluetooth).

Posteriormente se utilizó el entorno de desarrollo EasyEda (software para el diseño de circuitos) para realizar el diagrama del circuito electrónico.

El procedimiento realizado es el siguiente:

1. Se colocan los componentes electrónicos en el área de diseño, se les asigna su orientación y posición correspondiente. Todo en forma visual como se observa en la Figura 23.

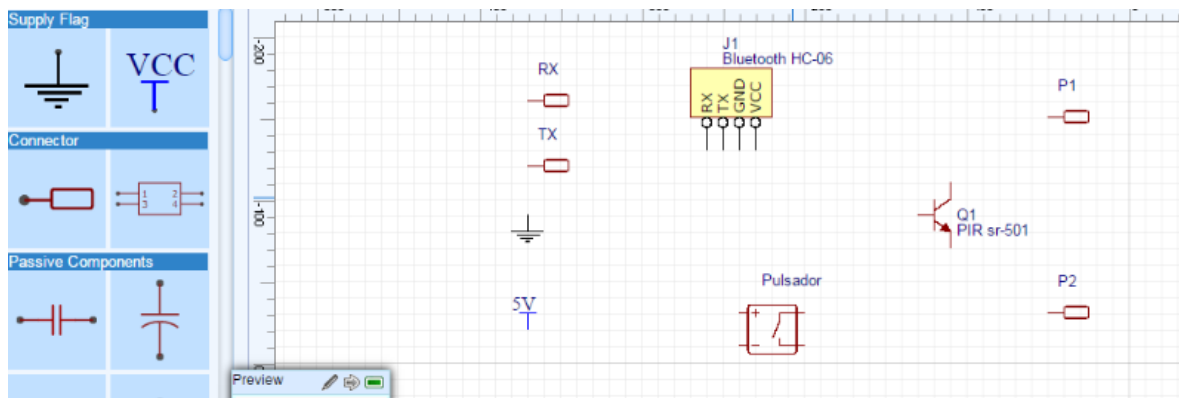


Figura 23. Área de diseño de EasyEda con los componentes del circuito desconectados.

2. Se unen las entradas y salidas de los componentes electrónicos con la herramienta de alambrado manual mostrada en la Figura 24.

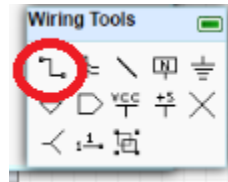


Figura 24. Herramientas de alambrado de EasyEda.

3. Se exporta el diseño final a un formato de imagen o pdf, como se muestra en la Figura 25, ya que es una herramienta de abstracción del circuito y se debe poder revisar en cualquier momento sin necesidad de abrir el entorno de desarrollo.

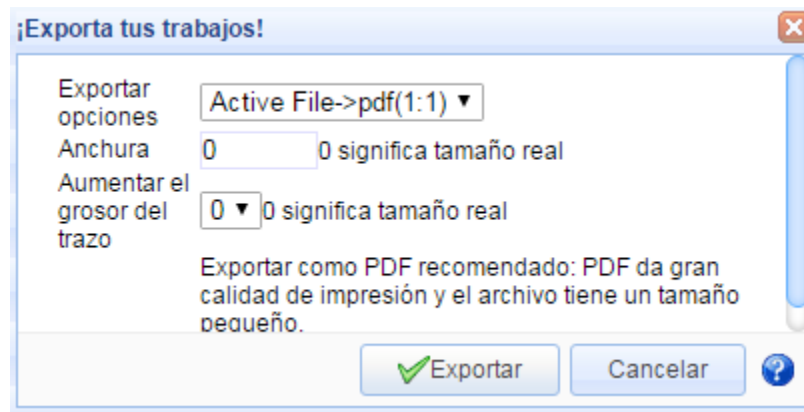


Figura 25. Opción para exportar el diseño a imagen o pdf en EasyEda.

En un principio el sensor PIR HC-Sr501 se expresó como si fuera un transistor, esto permite la comprensión del diagrama ya que el sensor en realidad es un transistor que se satura con cambios de temperatura, posteriormente se indica sólo como una salida de 3 pines para el diagrama final mostrado en la Figura 26.

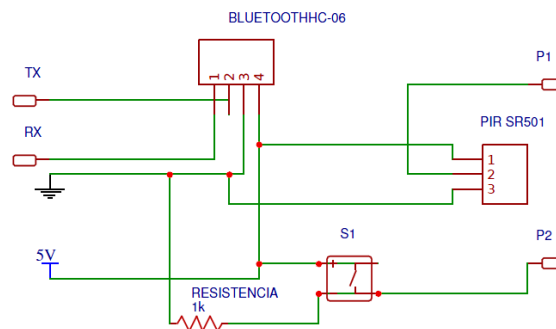


Figura 26. Diagrama final del circuito electrónico para el módulo de recepción de señales de presencia y botón de control.

La resistencia de 1K utilizada en éste circuito podría ser de cualquier valor ya que sus únicas funciones son: mantener un bit bajo y evitar que la corriente fluya directamente a tierra cuando el pulsador está cerrado.

### 6.3.2 Diseño del circuito impreso

Se utilizó el entorno de desarrollo de EasyEda, este entorno es capaz de convertir un diagrama de circuito electrónico en un circuito impreso. El proceso es:

1. Convertimos los componentes y rutas de alambrado en un diseño de circuito impreso con la herramienta de conversión de EasyEda mostrada en la Figura 27.

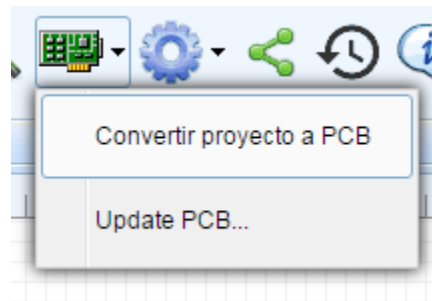


Figura 27. Función de EasyEda para convertir los componentes y rutas de un circuito electrónico en un circuito impreso.

2. Vamos colocando los componentes del circuito en la posición y orientación deseada para el circuito impreso como se muestra en la Figura 28.



Figura 28. Colocación de los componentes dentro del área del circuito impreso.

3. Al terminar de colocar los componentes dentro del área se procede a crear las rutas (pistas) que conformarán el circuito impreso, todas se deben crear de forma manual. Esto se muestra en la Figura 29.

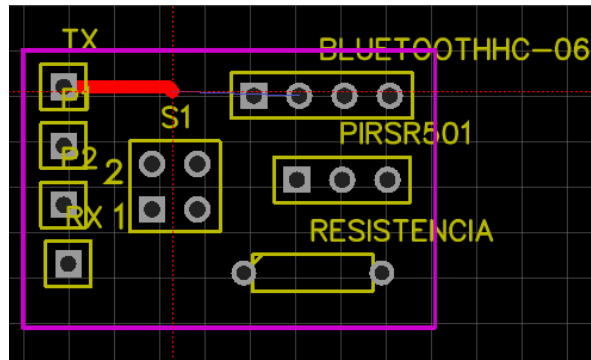


Figura 29. Proceso de creación de las rutas (pistas) de forma manual para el circuito impreso.

- Al terminar el proceso de creación de rutas se procede a la creación de orificios donde se conectarán cada uno de los componentes electrónicos. La separación y la posición de los puntos son calculados automáticamente por el entorno de desarrollo así que sólo se configura el radio. El resultado se muestra en la Figura 30.

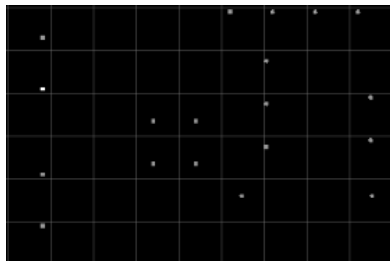


Figura 30. Creación de orificios para el circuito impreso.

Para el circuito impreso final que consta de pistas y orificios, como se muestra en la Figura 31, se realizaron las dos acciones siguientes:

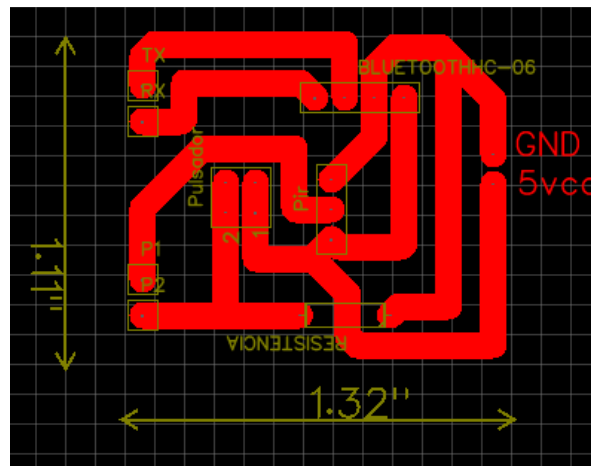


Figura 31. Diseño final del circuito impreso.

1. Se exportaron las pistas del diseño del circuito impreso final como una imagen en blanco y negro como se ve en la Figura 32.

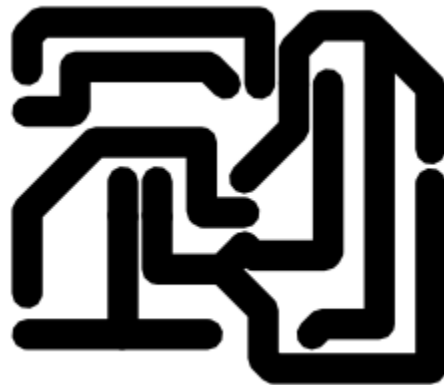


Figura 32. Pistas del circuito impreso final.

2. Se exportaron los orificios del diseño del circuito impreso final.

### 6.3.3 Construcción del circuito impreso

De los múltiples procesos que existen en la industria para la construcción de un circuito impreso se seleccionó el proceso de fresado por control numérico computarizado ya que es el único proceso que no utiliza químicos contaminantes los cuales son un peligro si no se toman las medidas adecuadas de manipulación y desecho como el ácido férrico.

Se utilizó un router CNC X2001, mostrado en la Figura 33, controlado por el software de control numérico computarizado Mach 3 mostrado en la Figura 34.



Figura 33. Router CNC X2001.



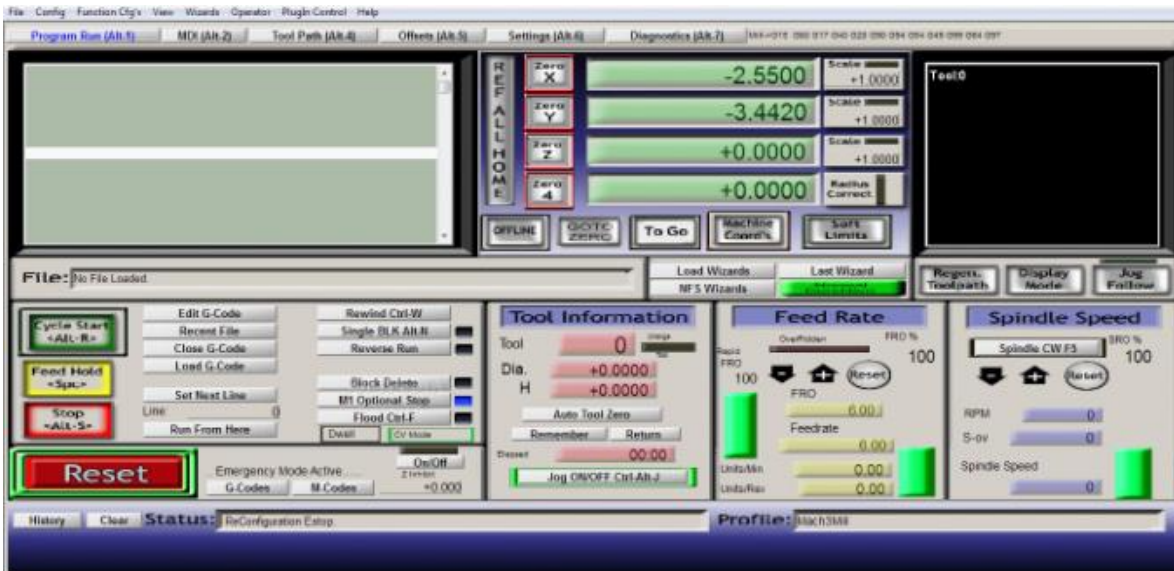


Figura 34. Software de control numérico computarizado Mach 3.

Para la generación de código G, que es el tipo de programación que se requiere para el control numérico computarizado, se utilizó el software de modelado (CAM) Vectric Aspire 8 mostrado en la Figura 35.

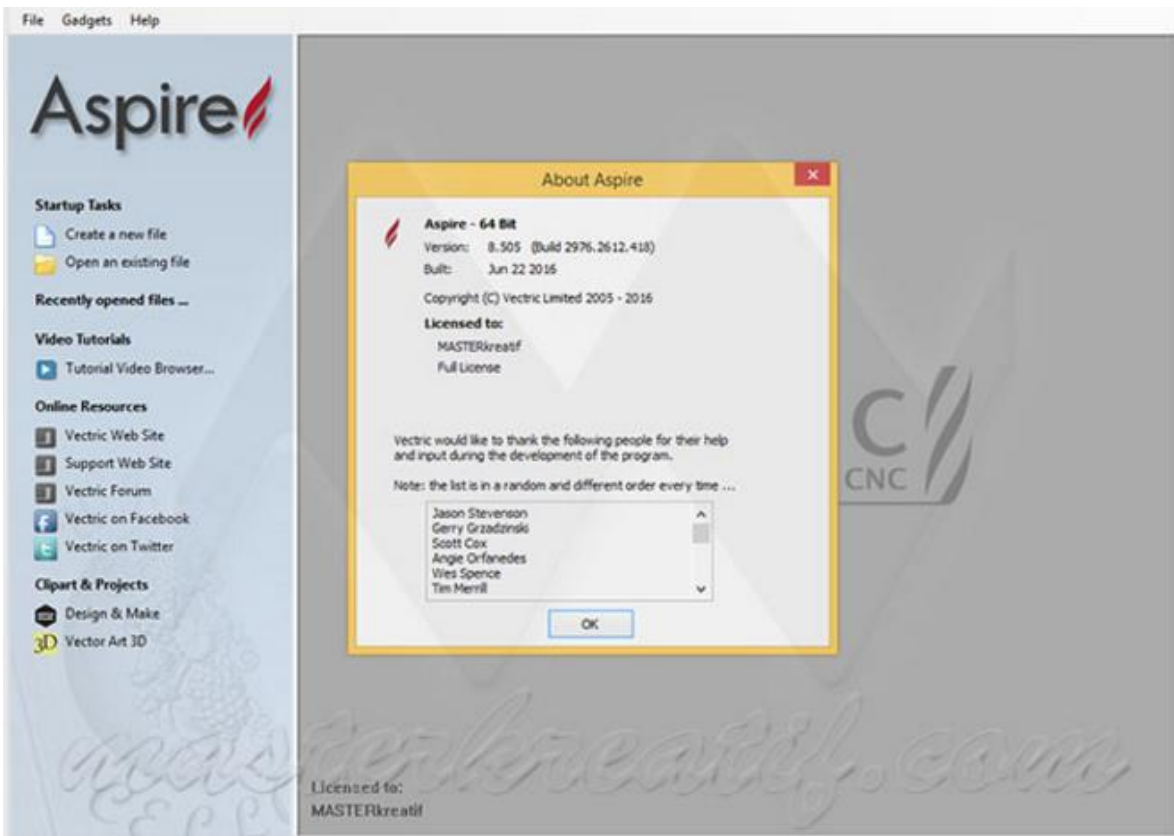


Figura 35. Software CAM Vectric Aspire.

La construcción del circuito impreso se realizó sobre una placa fenólica de 1.5 milímetros de base aislante y 0.105 milímetros de lámina de cobre (conductor) tal como lo recomienda la norma UNE 20-621-84/3[14]. El proceso fue el siguiente:

1. Con ayuda del software de modelado Vectric Aspire 8, se exportaron las imágenes de las pistas y orificios creados en la etapa de diseño del circuito impreso y se generaron los vectores correspondientes a éstos. Se muestra en la Figura 36.

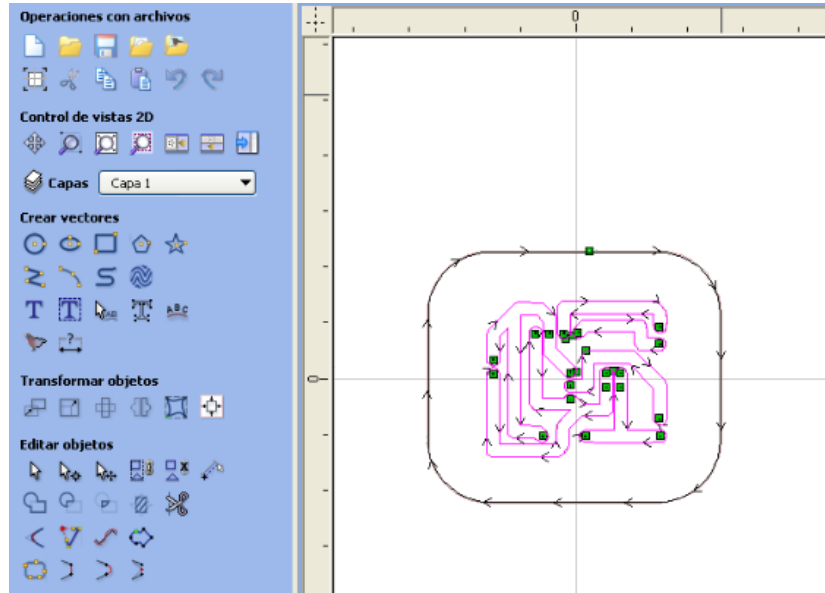


Figura 36. Circuito transformado en vectores por Vectric Aspire.

2. Se generó el código G configurando el fresado de la siguiente forma: los orificios del circuito impreso con un diámetro de 0.5mm y la separación entre pistas de 2mm, como se ve en la Figura 37, que es una de las recomendaciones en la norma UNE 20-552-75[14].

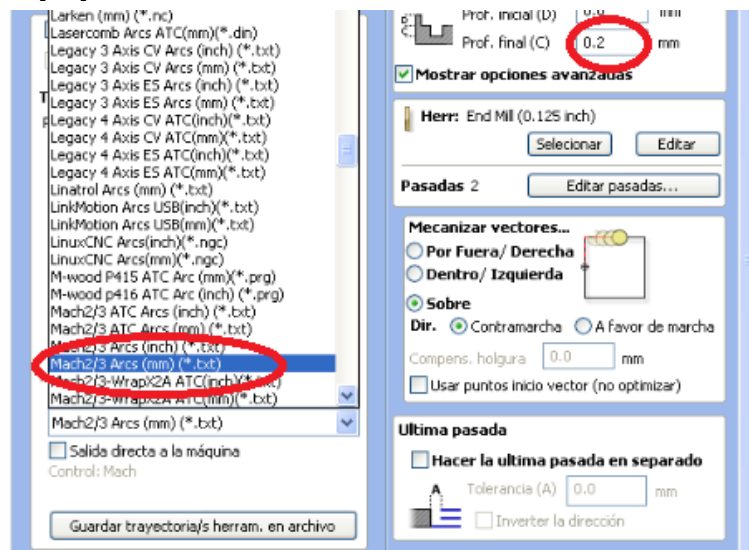


Figura 37. Generación de código G en Vectric Aspire.

- Se fijó la placa fenólica a la base del router CNC y se determinó el centro de la placa de forma manual y se colocó la herramienta de corte sobre éste como se ve en la Figura 38.



Figura 38. Placa fenólica sin ser fresada.

- Se ejecutó el software de control numérico Mach 3 y se cargó el código generado en el punto 2 como se observa en la Figura 39.

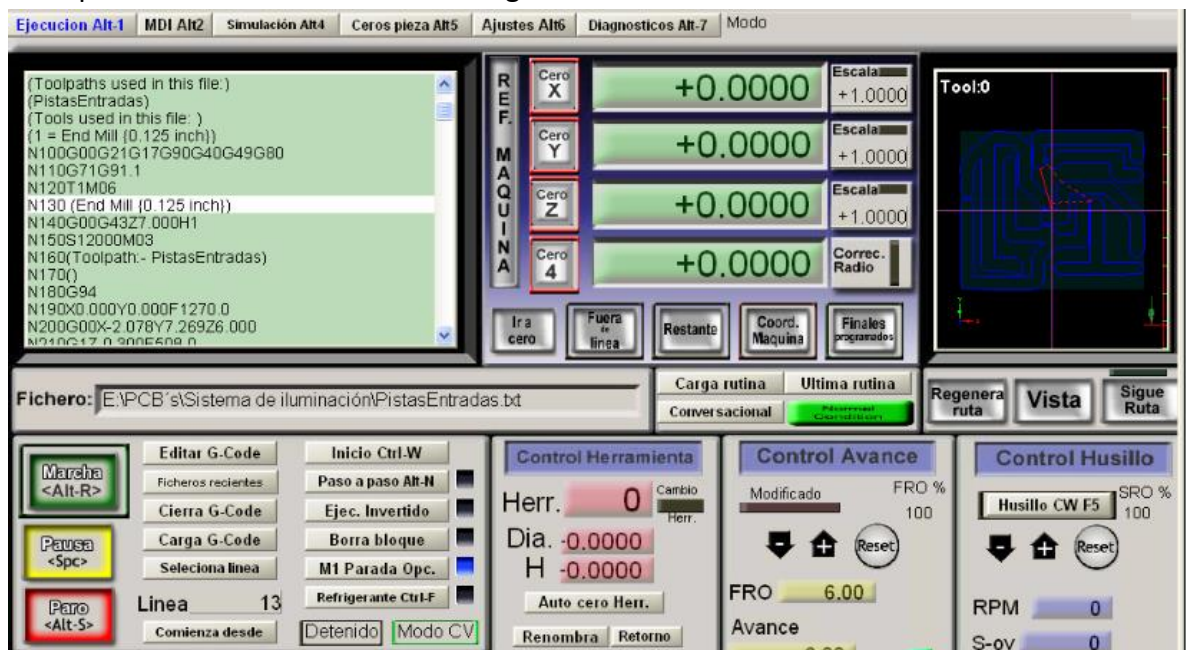


Figura 39. Código G cargado en el software CNC Mach 3.

5. Se inició el proceso de fresado el cual duró alrededor de 2 minutos. El proceso intermedio se ve en la Figura 40.

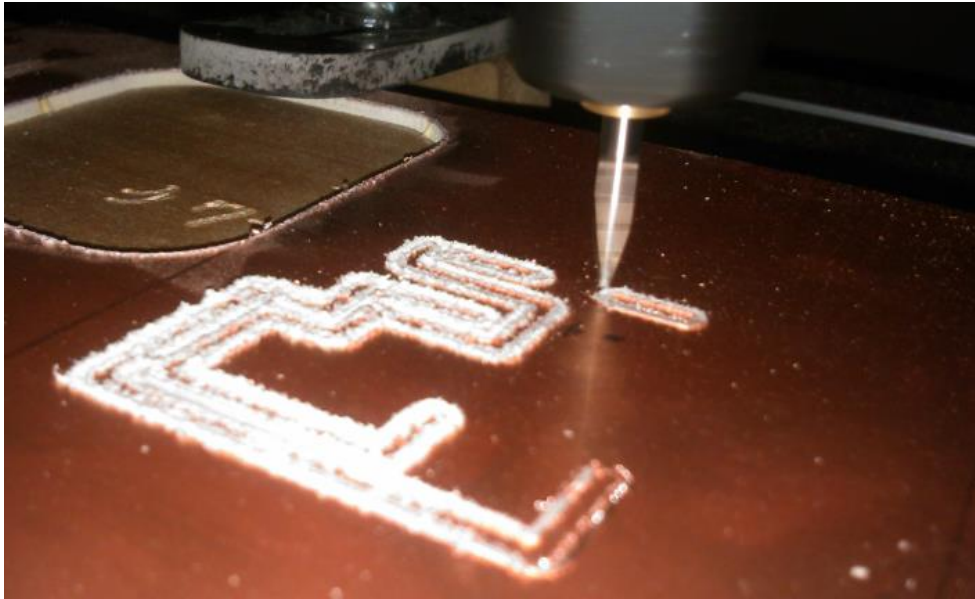


Figura 40. Proceso de fresado de la placa fenólica.

6. Se tomó la placa fenólica fresada y se colocaron los componentes y buses electrónicos correspondientes, según el diseño del circuito impreso. El proceso intermedio se ve en la Figura 41.

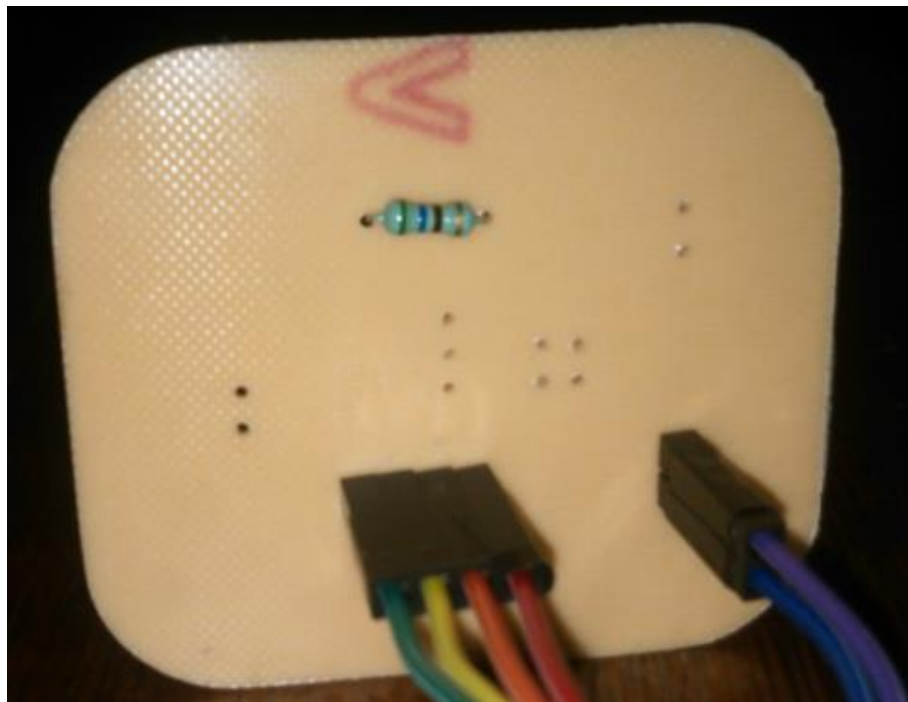


Figura 41. Colocación de componentes y buses.

7. Se soldaron los componentes uno a uno. El resultado se muestra en la Figura 42.

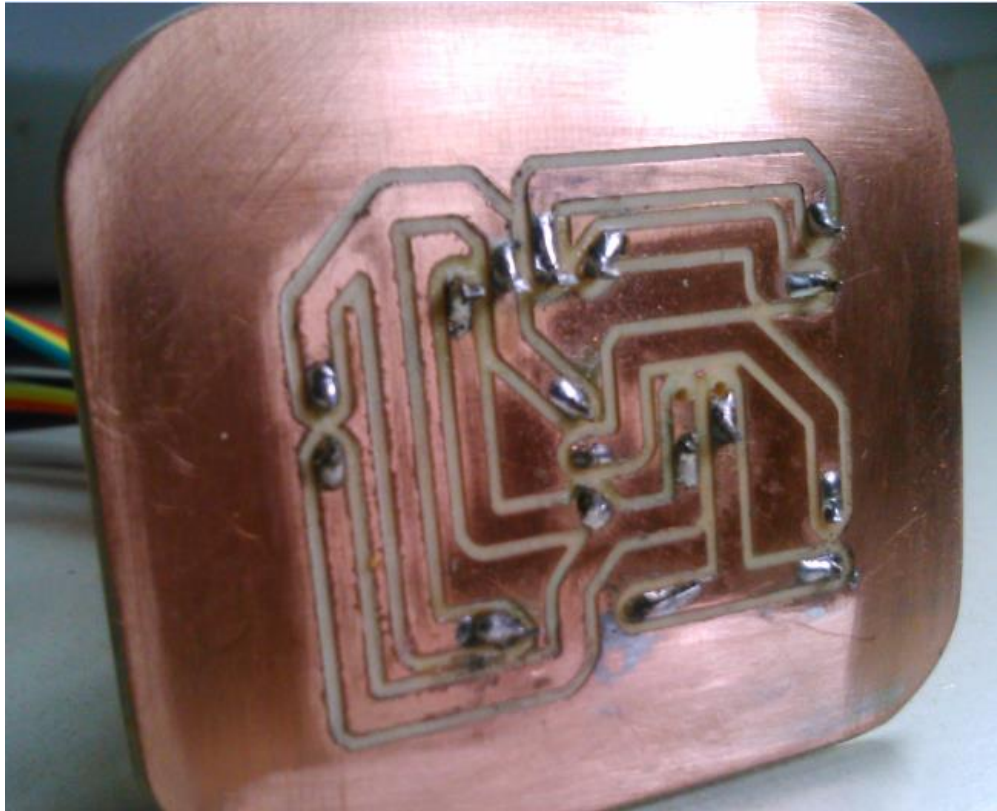


Figura 42. Componentes electrónicos soldados en el circuito impreso.

8. Finalmente se comprobó la conexión correcta de cada componente como se puede ver en la Figura 43.



Figura 43. Comprobando el funcionamiento correcto de los componentes.

Después de este procedimiento el módulo de recepción de señales de presencia y botón de control se da por terminado, éste se muestra en la Figura 44. El proceso duró aproximadamente 48 horas.

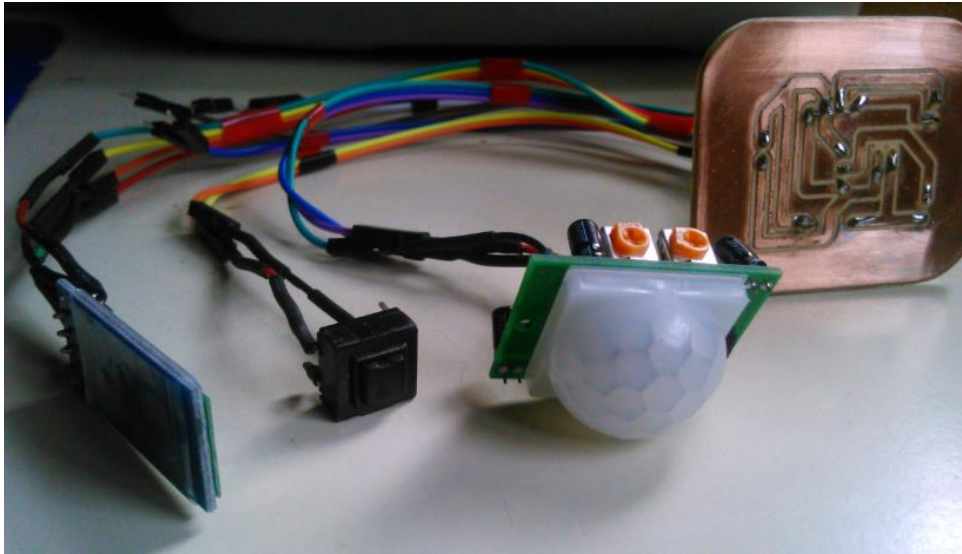


Figura 44. El módulo de recepción de señales de presencia y botón de control terminado.

## 7 Para el módulo de potencia

Su función es la de amplificar las señales PWM<sup>6</sup>, provenientes del módulo de procesamiento de instrucciones, a una señal PWM de mayor voltaje y enviarla al módulo de iluminación como se ve en la Figura 45.

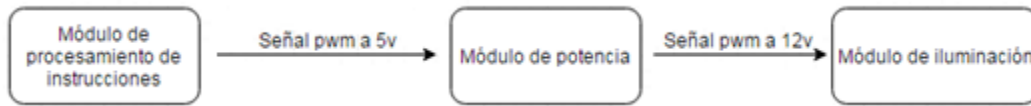


Figura 45. Amplificación de señal por el módulo de potencia.

Éste módulo es un dispositivo de hardware que se desarrolló en tres etapas: etapa de diseño del circuito electrónico, etapa de diseño del circuito impreso y etapa de construcción del circuito impreso.

### 7.1 Diseño del circuito electrónico

Se consideraron los siguientes componentes electrónicos para el circuito:

- **Transistor TIP120 Darlington:** A nivel técnico nos permite, mediante el uso de su área de saturación, transformar las señales PWM de 5v provenientes del módulo de procesamiento de instrucciones en señales de 12v que es el voltaje al que trabaja la iluminación. El componente se muestra en la Figura 46.

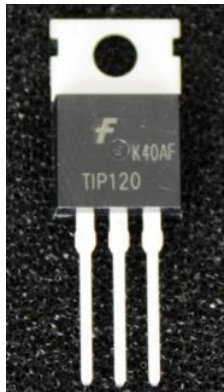


Figura 46. Transistor TIP120 Darlington.

- **Diodo rectificador 1N4001:** A nivel técnico nos permite proteger las salidas PWM de algún error humano, al conectar el circuito, o de algún componente con mal funcionamiento. El componente se muestra en la Figura 47.

---

<sup>6</sup> La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de Pulse-Width-Modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una [senoidal](#) o una [cuadrada](#)), de forma digital sin necesidad de usar potenciómetros.



Figura 47. Diodo rectificador 1N4001.

- **Resistor de 1k:** A nivel técnico nos permite que la corriente proveniente de las salidas PWM mantengan al transistor TIP120 en su área de saturación. El componente se muestra en la Figura 48.



Figura 48. Resistor de un 1K.

Posteriormente se utilizó el entorno de desarrollo EasyEda para realizar el diagrama del circuito electrónico.

El procedimiento realizado es el siguiente:

1. Se colocan los componentes electrónicos en el área de diseño, se les asigna su orientación y posición correspondiente. Todo de forma visual como se ve en la Figura 49.

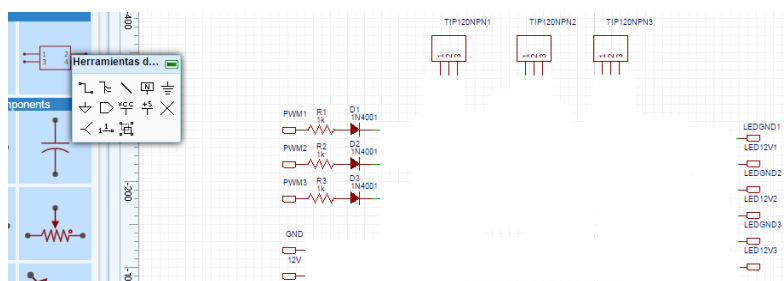


Figura 49. Área de diseño de EasyEda con los componentes del circuito desconectados.



2. Se unen las entradas y salidas de los componentes electrónicos con la herramienta de alambrado manual. También de forma visual como se ve en la Figura 50.

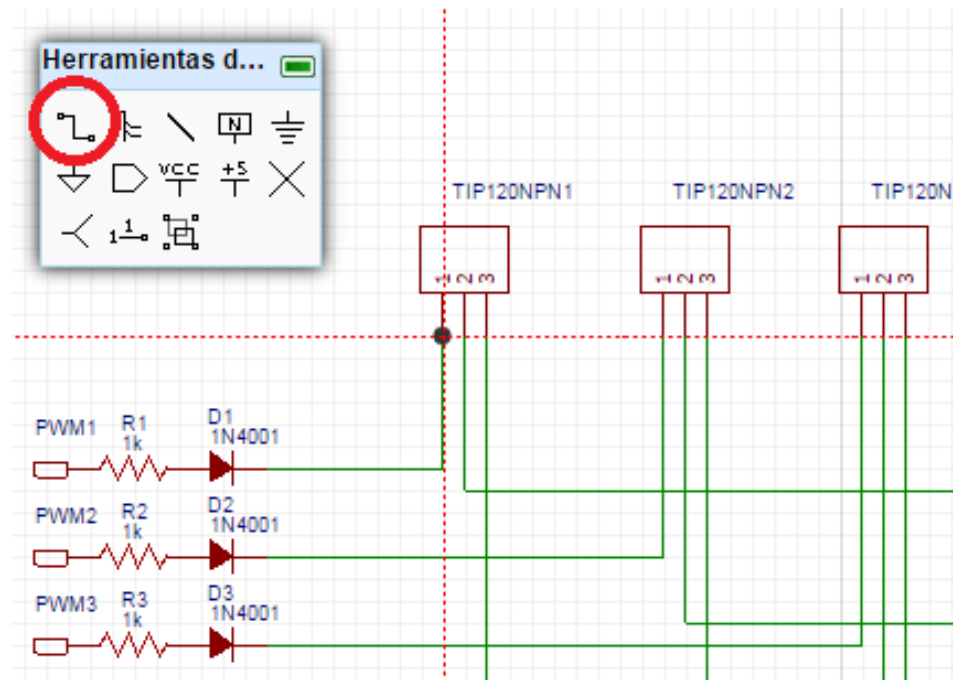


Figura 50. Herramientas de alambrado de EasyEda.

3. Se exporta el diseño final a un formato de imagen o pdf, como se observa en la Figura 51, ya que es una herramienta de abstracción del circuito y se debe poder revisar en cualquier momento sin necesidad de abrir el entorno de desarrollo.

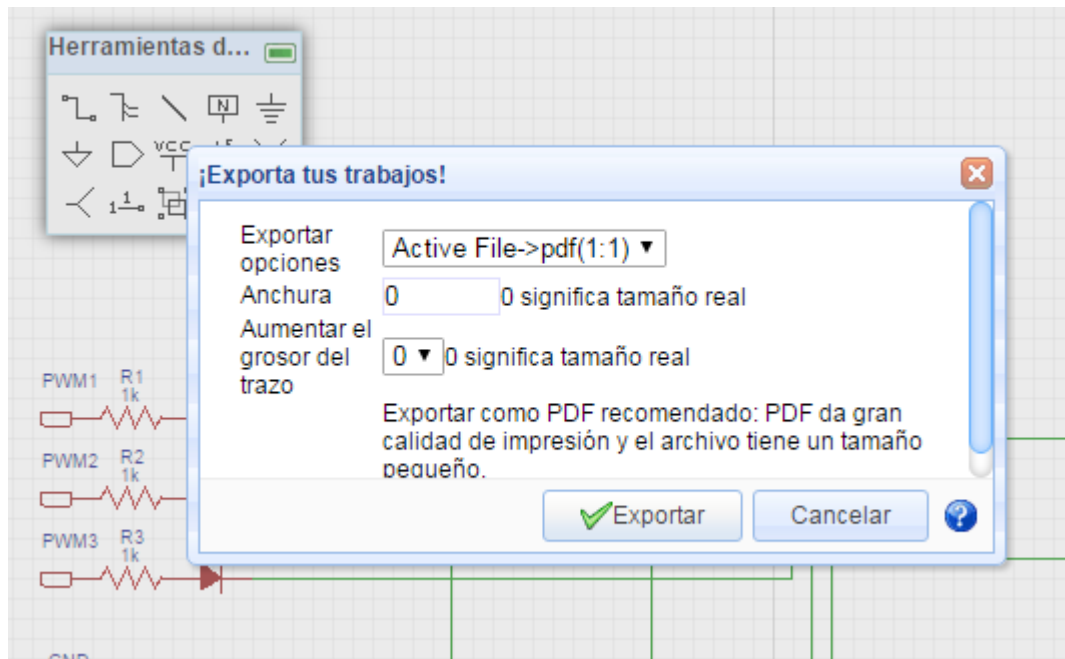


Figura 51. Opción para exportar el diseño a imagen o pdf en EasyEda.

Al terminar éste procedimiento se da por finalizado el diseño del circuito impreso para el módulo de potencia. El diagrama final se ve en la Figura 52.

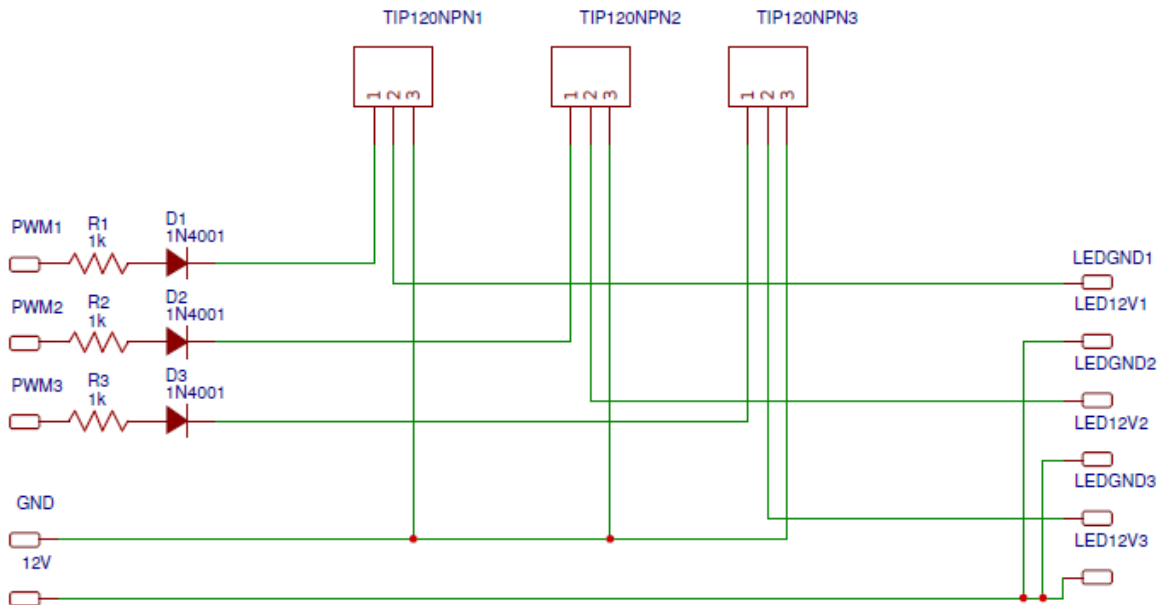


Figura 52. Diagrama final del circuito electrónico para el módulo de potencia.

## 7.2 Diseño del circuito impreso

Se utilizó el entorno de desarrollo de EasyEda, este entorno es capaz de convertir un diagrama de circuito electrónico generando los componentes y rutas para la creación del circuito impreso. El proceso es:

1. Convertimos los componentes y rutas de alambrado en un diseño de circuito impreso con la herramienta de conversión de EasyEda. La opción se muestra en la Figura 53.

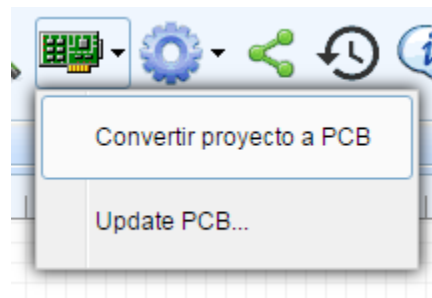


Figura 53. Función de EasyEda para convertir los componentes y rutas de un circuito electrónico en un circuito impreso.

2. Vamos colocando los componentes del circuito en la posición y orientación deseada para el circuito impreso como se muestra en la Figura 54.

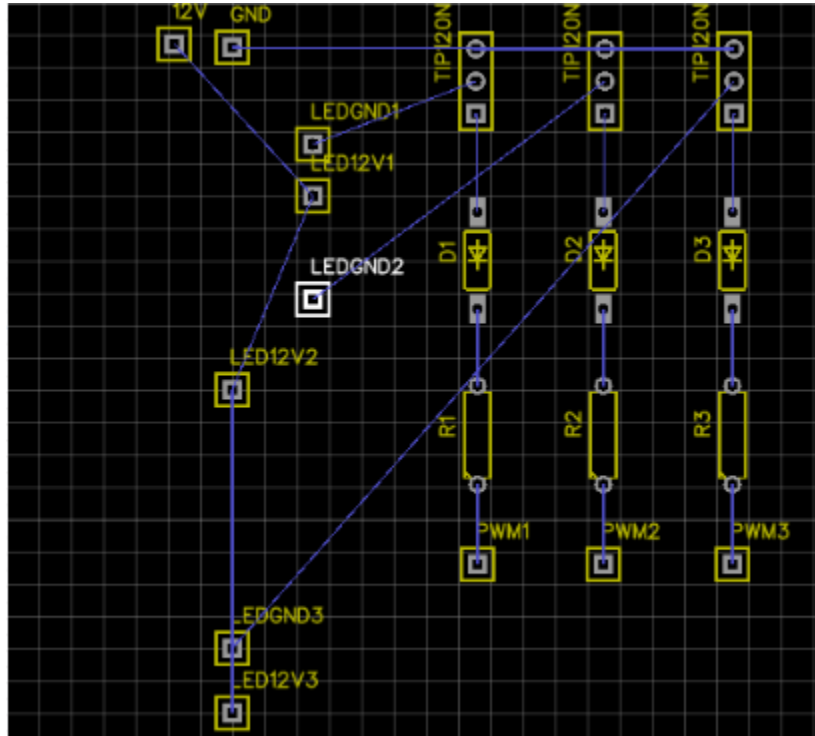


Figura 54. Colocación de los componentes dentro del área del circuito impreso.

- Al terminar de colocar los componentes dentro del área se procede a crear las rutas (pistas) que conformarán el circuito impreso, todas se deben crear de forma manual. El proceso se puede apreciar en la Figura 55.

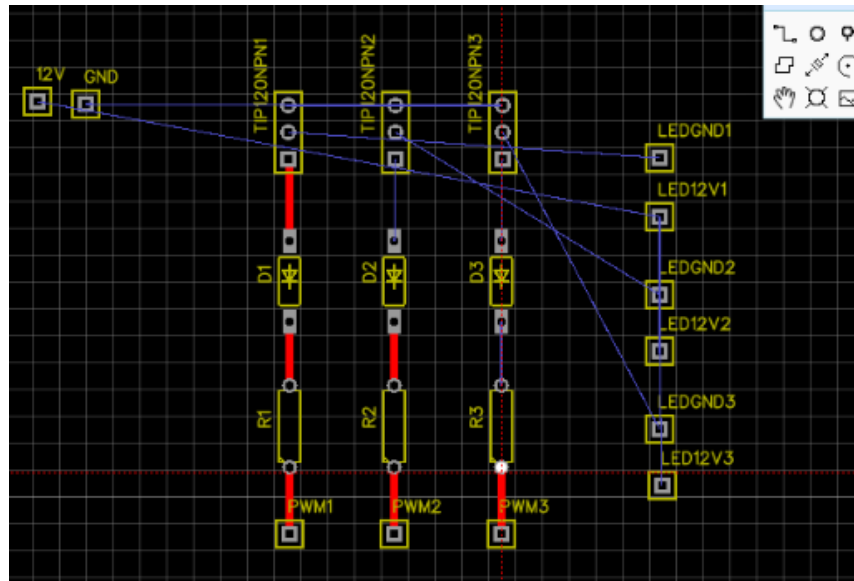


Figura 55. Proceso de creación de las rutas (pistas) de forma manual para el circuito impreso.

- Al terminar el proceso de creación de rutas se procede a la creación de orificios donde se conectarán cada uno de los componentes electrónicos. La separación

y la posición de los puntos son calculados automáticamente por el entorno de desarrollo así que sólo se configura el radio. El resultado se muestra en la Figura 56.

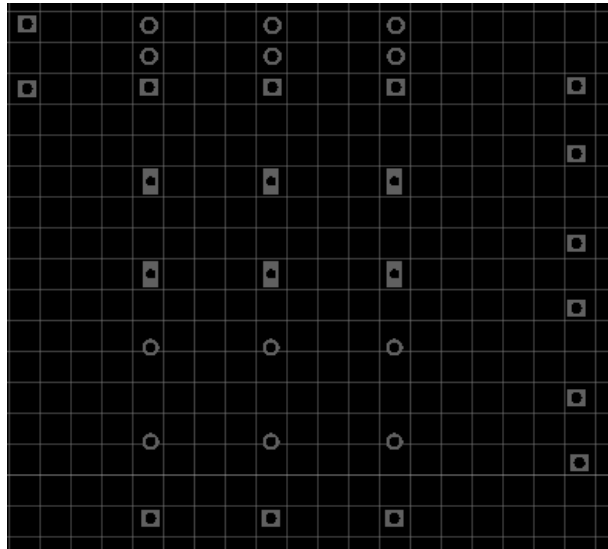


Figura 56. Creación de orificios para el circuito impreso.

Para el circuito impreso final que consta de pistas y orificios, se observa en la Figura 57, se realizaron las dos acciones siguientes:

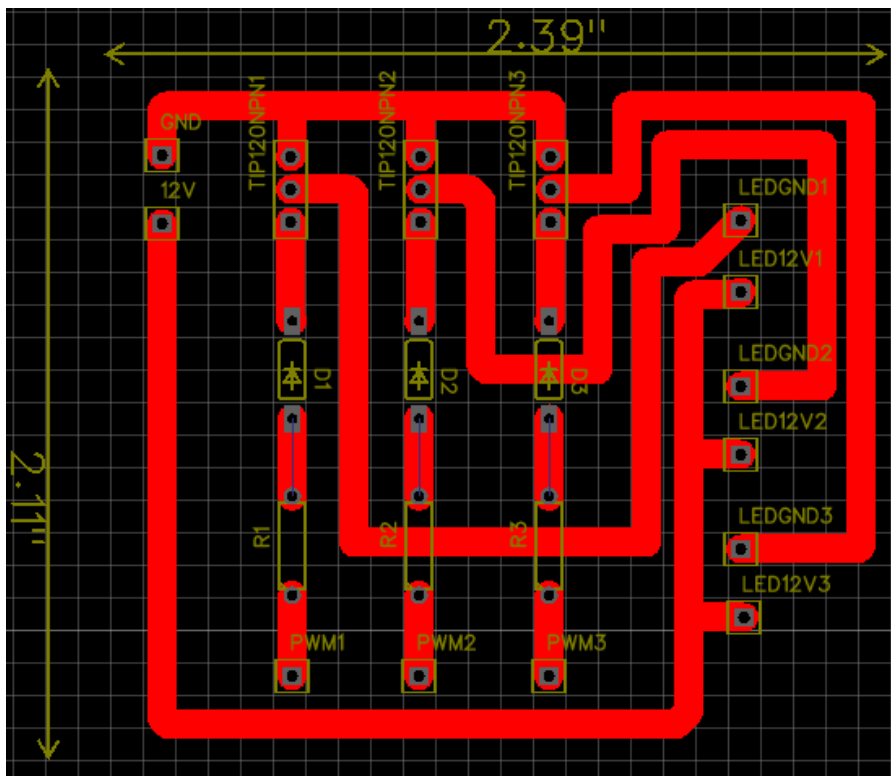


Figura 57. Diseño final del circuito impreso.

1. Se exportaron las pistas del diseño del circuito impreso final como una imagen en blanco y negro como se ve en la Figura 58.

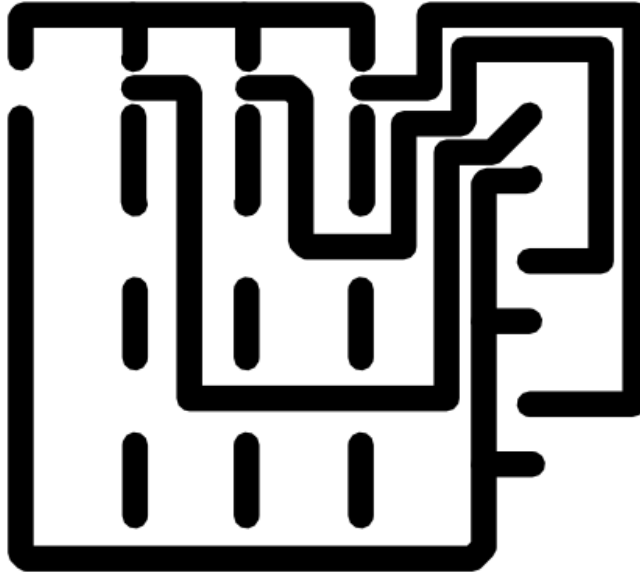


Figura 58. Pistas del circuito impreso final.

2. Se exportaron los orificios del diseño del circuito impreso final como una imagen en blanco y negro como se ve en la Figura 59.

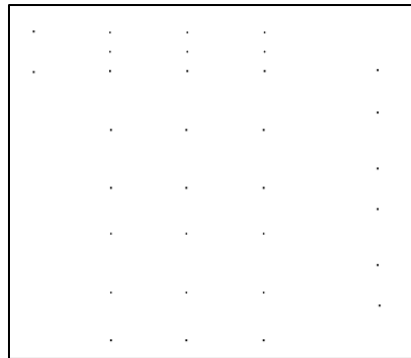


Figura 59. Orificios del circuito impreso final.

### 7.3 Construcción del circuito impreso

Se utilizó, nuevamente, un router CNC X2001 controlado por el software de control numérico computarizado Mach 3.

Para la generación de código G, que es el tipo de programación que se requiere para el control numérico computarizado, se utilizó, nuevamente, el software de modelado (CAM) Vectric Aspire 8.

La construcción del circuito impreso se realizó sobre una placa fenólica de 1.5 milímetros de base aislante y 0.105 milímetros de lámina de cobre (conductor) tal como lo recomienda la norma UNE 20-621-84/3. El proceso fue el siguiente:

1. Con ayuda del software de modelado Vectric Aspire 8, se exportaron las imágenes de las pistas y orificios creados en la etapa de diseño del circuito impreso y se generaron los vectores correspondientes a éstos. Se muestra en la Figura 60.

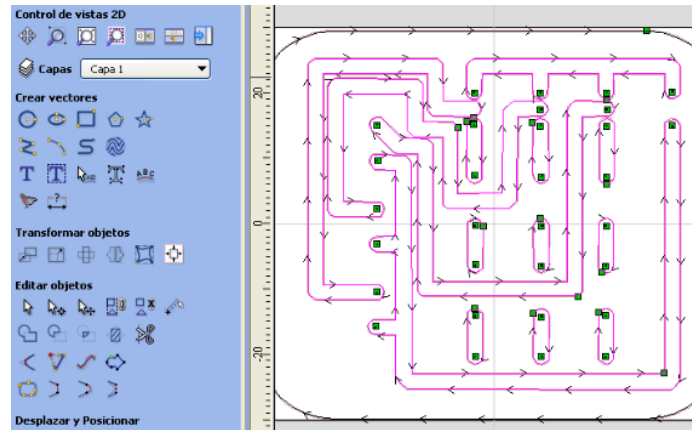


Figura 60. Circuito transformado en vectores por Vectric Aspire.

2. Se generó el código G configurando el fresado de la siguiente forma: los orificios del circuito impreso con un diámetro de 0.5mm y la separación entre pistas de 2mm, como se ve en la Figura 61, que es una de las recomendaciones en la norma UNE 20-552-75.

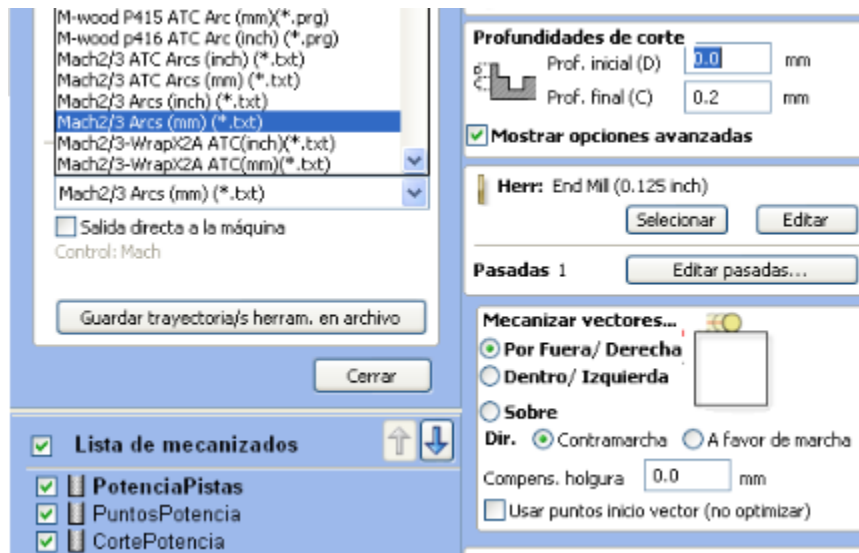


Figura 61. Generación de código G en Vectric Aspire.

3. Se fijó la placa fenólica a la base del router CNC y se determinó el centro de la placa de forma manual y se colocó la herramienta de corte sobre éste como se observa en la Figura 62.



Figura 62. Placa fenólica sin ser fresada.

4. En el software de control numérico Mach 3 se cargó el código generado en el punto 2. Esto se observa en la Figura 63.



Figura 63. Código G cargado en el software CNC Mach 3.

5. Se inició el proceso de fresado el cual duró alrededor de 2 minutos. El proceso intermedio se ve en la Figura 64.

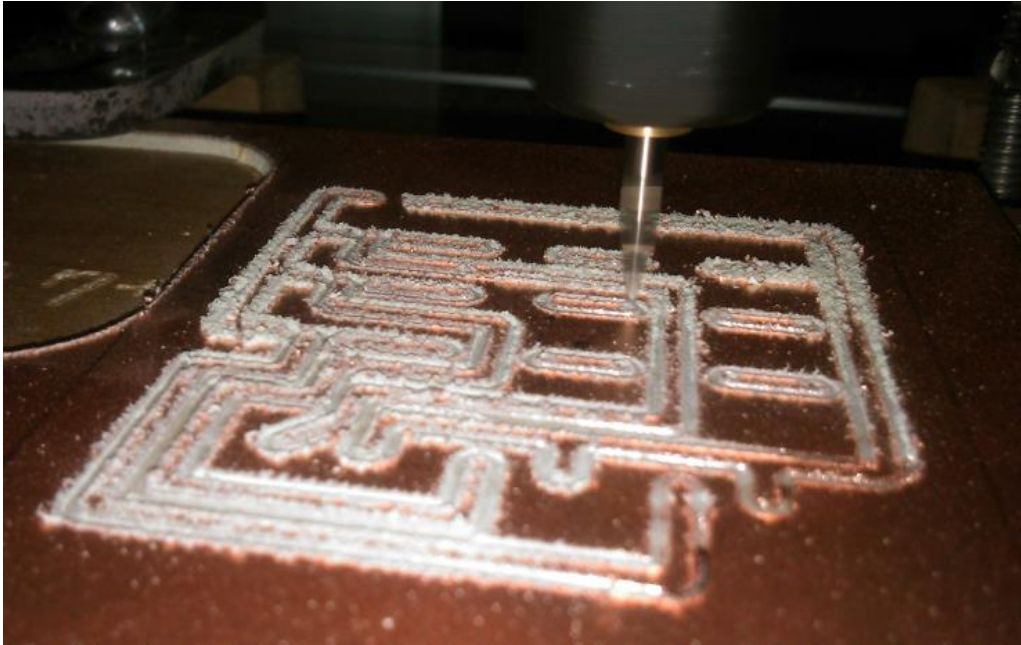


Figura 64. Proceso de fresado de la placa fenólica.

6. Se tomó la placa fenólica fresada y se colocaron los componentes y buses electrónicos correspondientes como se muestra en la Figura 65, según el diseño del circuito impreso.

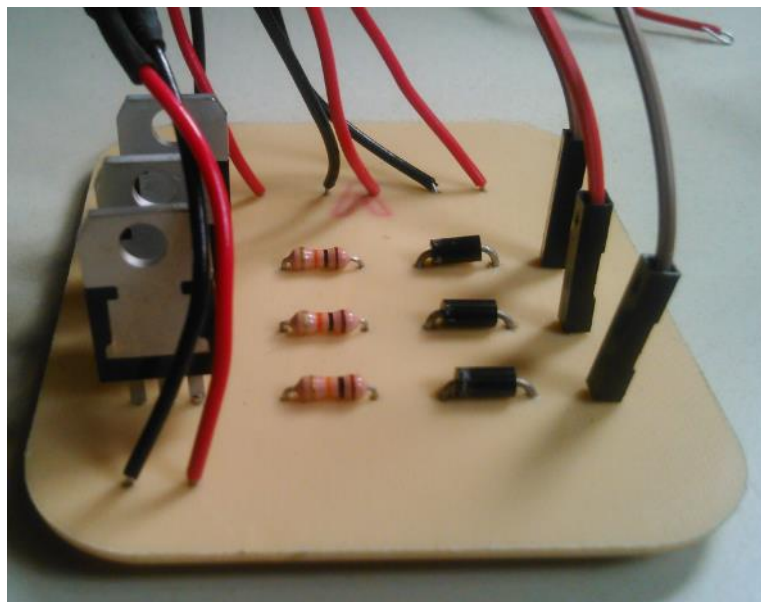


Figura 65. Colocación de componentes y buses.



7. Se soldaron los componentes uno a uno mediante el procedimiento normal. El resultado se muestra en la Figura 66.

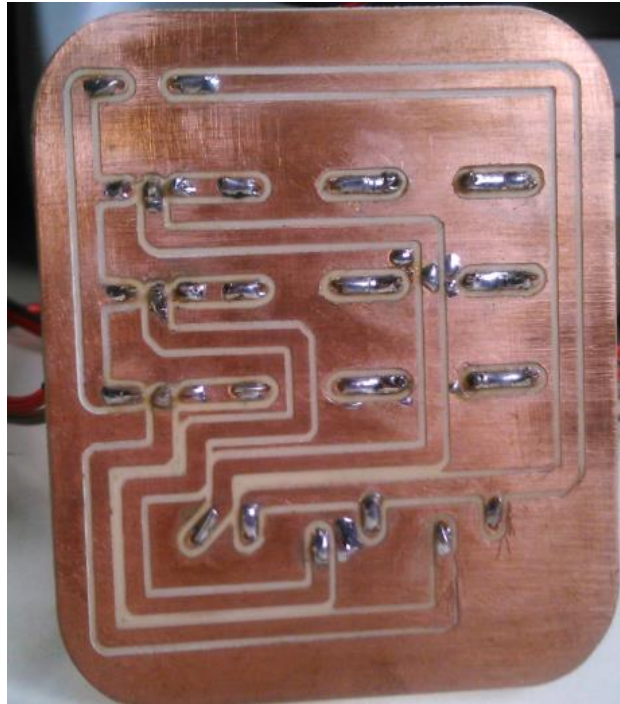


Figura 66. Componentes electrónicos soldados en el circuito impreso.

8. Finalmente se comprobó la conexión correcta de cada componente. Esto se ve en la Figura 67.

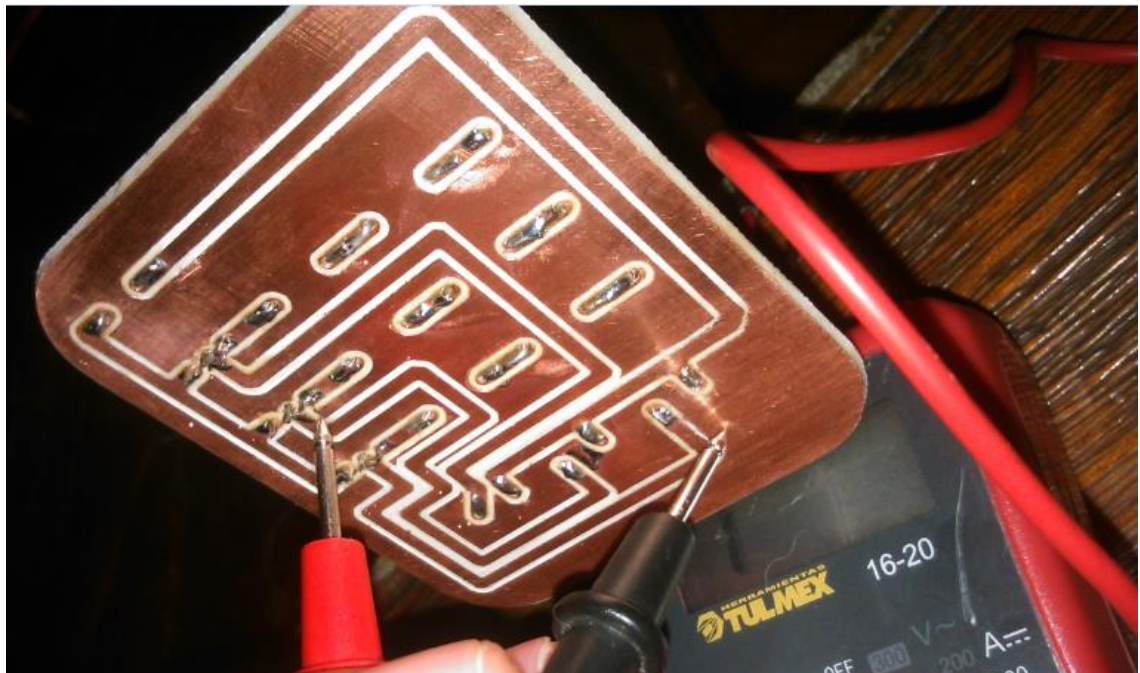


Figura 67. Comprobando el funcionamiento correcto de los componentes.

Después de este procedimiento el módulo de potencia se da por terminado y el resultado se muestra en la Figura 68. El proceso duró aproximadamente 36 horas.

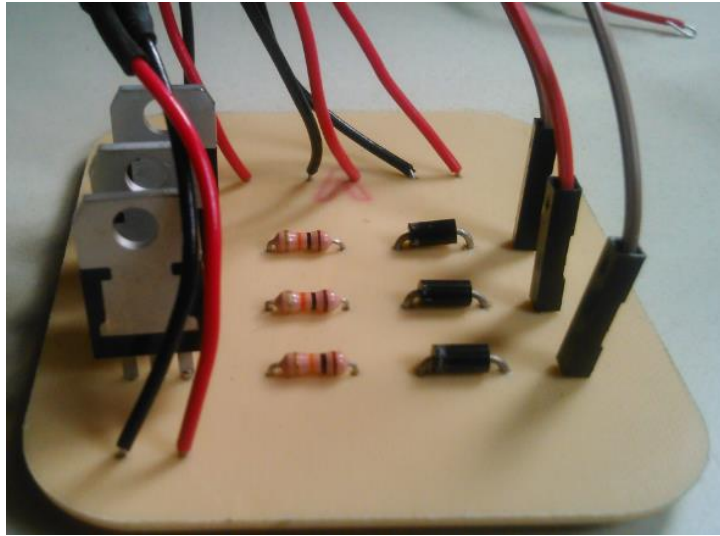


Figura 68. El módulo de potencia terminado.

## 8 Para el módulo de iluminación

Éste módulo será el encargado de brindar una respuesta de iluminación visible al usuario a partir de las instrucciones que se están ejecutando. Dentro de una habitación tendrá la capacidad de iluminar desde tres ángulos diferentes similar a lo que podemos observar en la Figura 69.

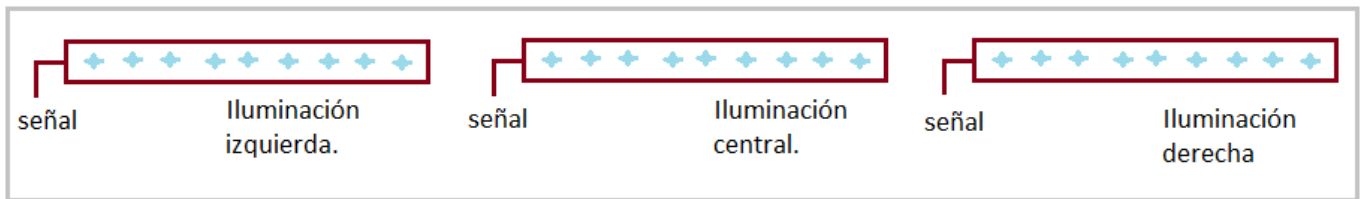


Figura 69. Descripción gráfica del módulo de iluminación.

El módulo de iluminación es un componente de hardware visible que influye en la presentación del dispositivo general. Se desarrolló en dos etapas: la etapa de diseño y la etapa de construcción.

### 8.1 Diseño

Se utilizó, nuevamente, el software CAM Vectric Aspire 8 para realizar el diseño de los vectores correspondientes. Es importante destacar que el diseño del módulo de iluminación depende, en gran medida, del gusto del diseñador y puede cambiar sin afectar el funcionamiento del dispositivo general.

De la gran cantidad de componentes de iluminación LED que existen en el mercado se decidió utilizar aquellos que trabajan a 12V ya que es conjunto de componentes más variados; de éstos se seleccionó la tira LED smd 50x50 que se muestra en la Figura 70, por ser la más comercial.



Figura 70. Tira de Leds SMD 5050.

**Consideraciones sobre la tira LED SMD 5050:**

- Éste componente contiene 3 Leds SMD 5050 por cada 5 cm y cada LED brinda un flujo luminoso de 15 lúmenes.
- Una lámpara convencional para una habitación emite un flujo luminoso estándar de aproximadamente 1000 lúmenes.

Entonces el diseño del módulo debe ser adaptado a éste componente.

Se diseñó una base para la tira LED de 18mm por 480mm, a ésta se le agregó un desnivel de 13 mm y en los extremos se colocaron dos orificios de 2mm; todo esto sobre un material de 18mm de espesor<sup>7</sup>. Esto se puede de visualizar en la Figura 71.

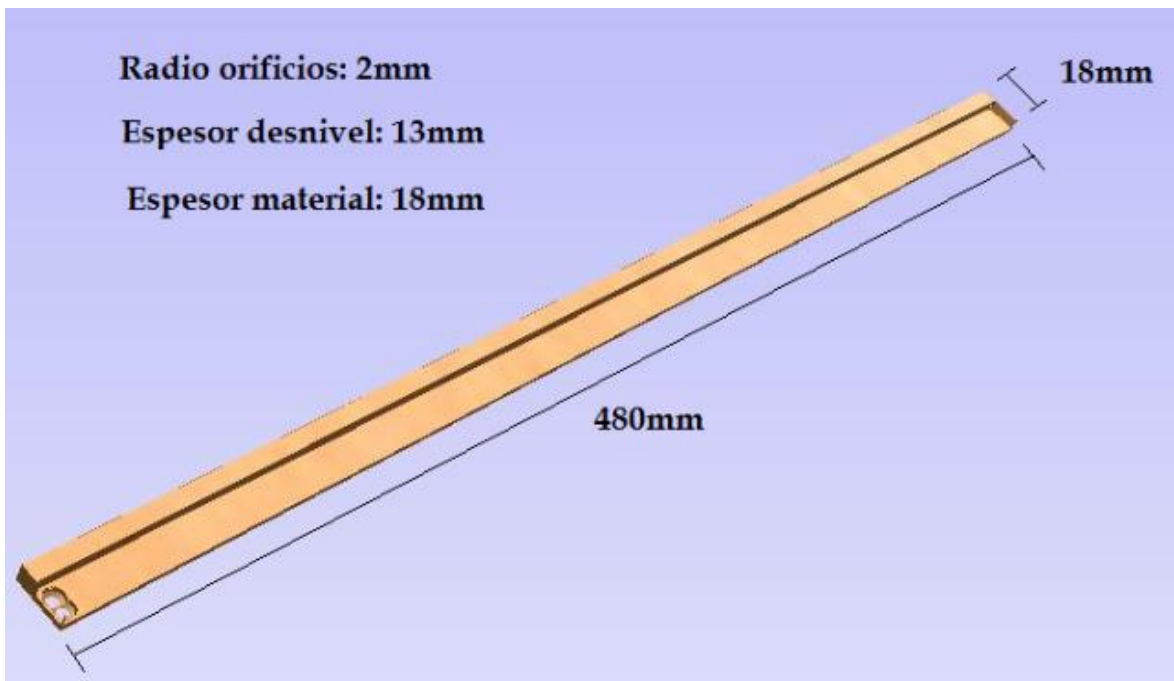


Figura 71. Simulación 3D de la base para la tira LED en el software Vectric Aspire 8.

## 8.2 Construcción

De acuerdo al diseño, como se mostró en la Figura 69, necesitamos tres módulos de iluminación: módulo de iluminación izquierda, módulo de iluminación derecha y módulo de iluminación central. Entre los tres módulos deben brindar un flujo luminoso de 1000 lúmenes que equivale a aproximadamente 120 centímetros de la tira LED.

Se dividieron los 120 centímetros de la siguiente forma:

- 45 cm para el módulo de iluminación izquierdo
- 45 cm para el módulo de iluminación derecho
- 30 cm para el módulo de iluminación central

<sup>7</sup> Las medidas de la base fueron, intencionalmente, realizadas para que la base coincida con una estructura comercial llamada "esquinero" y se puede encontrar en diversos materiales en el mercado como: plástico, madera o aluminio.

Debido a la simplicidad de la estructura diseñada se decidió realizar la construcción con técnicas rudimentarias utilizando como material la madera de pino. La estructura base se adquirió por metro como se ve en la Figura 72 y el procedimiento de adaptación realizado es el siguiente:



Figura 72. Material base en madera de pino.

1. Se cortaron las bases para los módulos de iluminación dejando 3 cm de tolerancia para colocar la tira led como se ve en la Figura 73, es decir:
  - 48 cm de base para el módulo de iluminación izquierdo.
  - 48 cm de base para el módulo de iluminación derecho.
  - 33 cm de base para el módulo de iluminación central.



Figura 73. Medidas marcadas para los cortes de las bases.

2. Se cortaron las bases y se lijaron los extremos para eliminar imperfecciones del corte. El resultado se ve en la Figura 74.



Figura 74. Bases para las tiras LED cortadas y lijadas.

3. Se pintaron las bases con pintura acrílica en aerosol negra para dar una mejor presentación. El proceso se ve en la Figura 75.



Figura 75. Pintado de las bases para las tiras LED.

4. Se realizaron los orificios y ranuras para los cables de alimentación según el diseño realizado anteriormente, como se muestra en la Figura 76.

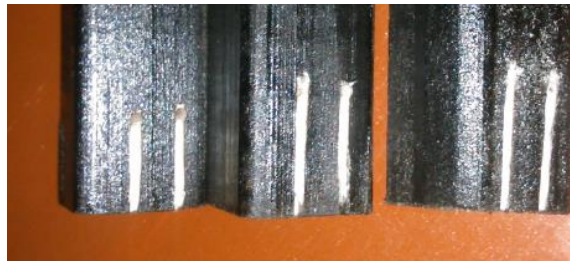


Figura 76. Orificios y ranuras para los cables de alimentación.

5. Con ayuda de la parte posterior auto-adherible de las tiras LED se colocaron sobre las bases construidas procurando que el extremo quedara junto a los orificios de la base como se observa en la siguiente Figura 77.



Figura 77. Orificios para la alimentación de las tiras LED.

6. Se introdujeron los cables de alimentación, se soldaron y se introdujeron en las ranuras que evitan su desplazamiento, como se ven la Figura 78.



Figura 78. Extremo del módulo de iluminación central, parte delantera y posterior.

Al terminar el procedimiento anterior se da por finalizada la construcción del módulo de iluminación que duró aproximadamente 24 horas. El resultado se muestra en la Figura 79.



Figura 79. Módulo de iluminación terminado.

## 9 Diagrama de conexión entre módulos

Antes de desarrollar el último módulo del proyecto se elaboró el diagrama de conexión entre módulos ya que para éste último módulo necesitamos conectar todos los módulos y verificar su funcionamiento.

Se utilizó el entorno de desarrollo EasyEda para realizar el diagrama de conexiones entre módulos.

El procedimiento realizado es el siguiente:

1. Se colocaron los componentes electrónicos en el área de diseño, se les asigna su orientación y posición correspondiente.
2. Se unieron las entradas y salidas de los componentes electrónicos con la herramienta de alambrado manual. El resultado se muestra en Figura 80.

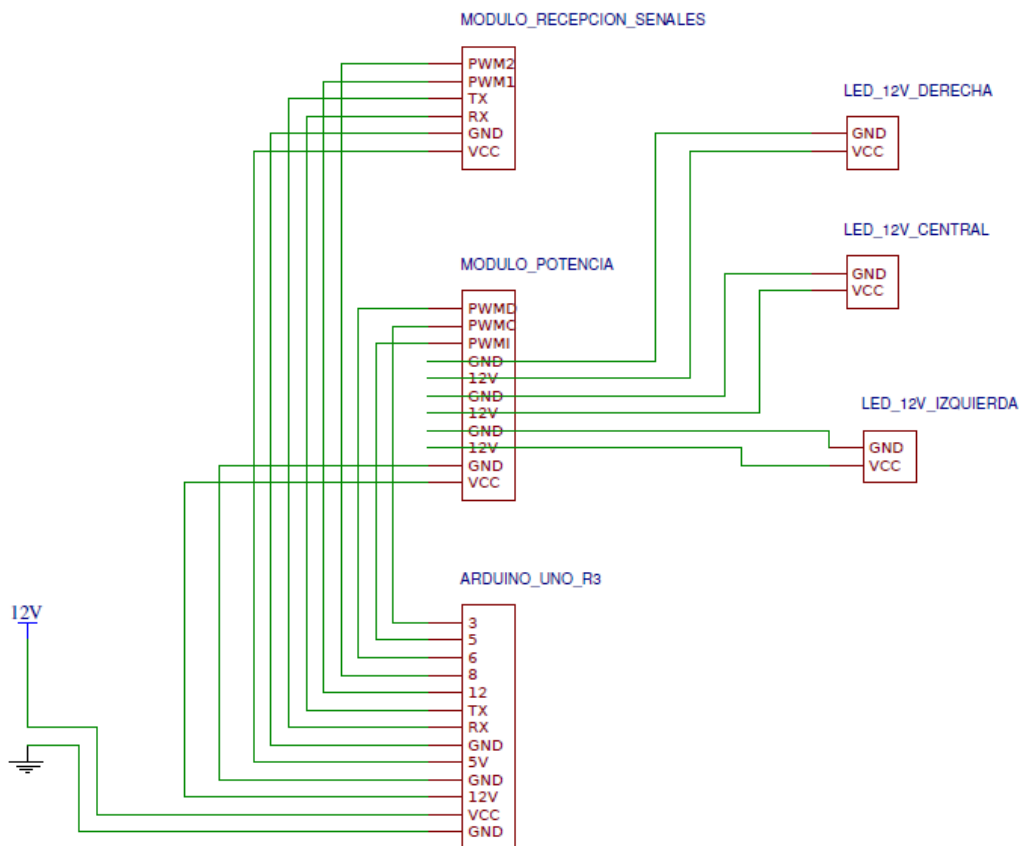


Figura 80. Diagrama de conexión entre módulos para formar el dispositivo general.

3. Se exporta el diseño final a un formato de imagen o pdf ya que es una herramienta de abstracción del dispositivo y se debe poder revisar en cualquier momento sin necesidad de abrir el entorno de desarrollo. La herramienta se muestra en la Figura 81.



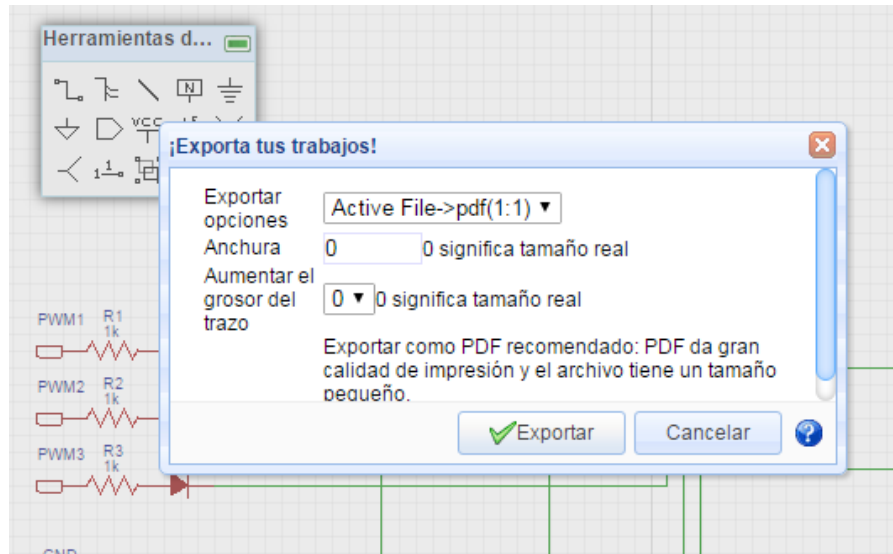


Figura 81. Opción para exportar el diseño a imagen o pdf en EasyEda.

## 10 Para el módulo de procesamiento de instrucciones

Su función es la de recibir, interpretar, ejecutar y administrar las instrucciones de iluminación inalámbricas del módulo de recepción de instrucciones por bluetooth así como procesar las señales del módulo de recepción de señales de presencia y botón de control para generar las señales PWM como se observa en la Figura 82.

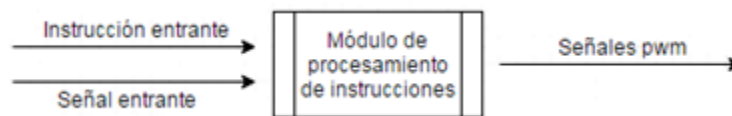


Figura 82. Funcionamiento general del módulo de procesamiento de instrucciones.

Éste módulo es el más complejo de todos los que forman el del proyecto ya que se encarga de administrar los recursos de hardware y brindar los servicios necesarios para ejecutar las diferentes funciones del dispositivo. Está constituido por dos partes: hardware y software.

### 10.1 Para la parte de hardware:

Se decidió utilizar la arquitectura de Arduino Uno R3, que se observa en la Figura 83, es decir, tenemos una arquitectura de hardware con las siguientes características básicas:

- Una velocidad de procesamiento de 16 MHz.
- Una memoria de almacenamiento permanente de 32kb, para sistema y datos permanentes.
- Una memoria principal de 2kb para procesos y datos temporales.
- Dos puertos seriales RX y TX.
- 14 puertos digitales.
- Seis puertos analógicos.



Figura 83. Arquitectura Arduino Uno R3.

Se seleccionó esta arquitectura debido a que es una de las arquitecturas más económicas del mercado aunque muy limitada en sus recursos.

## **10.2 Para la parte de software:**

El sistema desarrollado debe ser capaz de ejecutarse perfectamente en la arquitectura de hardware seleccionada, es decir:

- El sistema para ésta arquitectura no puede exceder los 32kb.
- Cada proceso con sus correspondientes variables locales y variables globales no puede equivaler a más de 2kb cargado en memoria principal
- El sistema para ésta arquitectura no debe generar uso de memoria principal para datos que pueda saturarla por completo en tiempo de ejecución y provocar el fallo total del sistema.
- El sistema para ésta arquitectura no puede exceder los 32kb de almacenamiento permanente para su funcionamiento.
- El sistema para ésta arquitectura debe administrar de forma correcta los tiempos de ejecución de las interrupciones y procesos.
- El sistema no debe contener procedimientos que monopolicen la CPU, cada proceso debe contener al menos una interrupción de salida esto evita que el sistema pueda quedar pasmado en determinado estado.

El desarrollo del software para éste módulo se realizó en dos partes: la parte de modelado del sistema y la parte de la implementación del sistema.

### **10.2.1 La parte del modelado del sistema:**

El desarrollo de un sistema de robustez media o alta no se debe realizar mediante programación exhaustiva sino que se debe realizar un modelado previo.

La parte más compleja del sistema era identificar los requerimientos necesarios del sistema. Una incorrecta identificación en los requerimientos ocasionaría un mal funcionamiento del sistema: flujos indeseados, procesos perdidos, retardos en el tiempo de respuesta, etc.

Para el descubrimiento de los requerimientos del sistema se decidió utilizar un modelo de casos de uso de texto el cual se desarrolló de forma iterativa, es decir, se realizó el primer modelo de casos de uso de texto; posteriormente se analizó mediante corridas de escritorio en busca de errores y se corrigieron en la siguiente iteración.

El desarrollo del modelo tomó un aproximado de 6 iteraciones, hasta llegar al modelo de casos de uso de texto final que se muestra a continuación.

## Modelo de casos de uso de texto para el módulo de procesamiento de instrucciones

### Consideraciones previas:

La distribución de puertos de la arquitectura Arduino es la siguiente:

- El puerto 3 analógico corresponde a la salida de iluminación central.
- El puerto 5 analógico corresponde a la salida de iluminación izquierda.
- El puerto 6 analógico corresponde a la salida de iluminación derecha.
- El puerto 8 digital corresponde a la entrada de bits del sensor de presencia.
- El puerto 12 digital corresponde a la entrada de bits del pulsador.
- Los puertos seriales tx y rx trabajan en conjunto para mantener la comunicación con el módulo de recepción de instrucciones por bluetooth.
- La variable IntensidadRespaldo almacena la intensidad establecida por el usuario.

### Escenario principal:

1. El sistema inicia y se coloca automáticamente en “modo 1” y establece las señales de los puertos analógicos: 3=0, 5=0 y 6=0.
2. El sistema está en “modo 1” y recibe un bit ‘1’ por el puerto digital 12.
3. El sistema realiza el cambio a “modo 2” y establece las señales de los puertos analógicos: 5=170, 6=170 y 3=0.
4. El sistema está en “modo 2” y recibe un bit ‘1’ por el puerto digital 12.
5. El sistema realiza el cambio a “modo 3” y establece las señales de los puertos analógicos: 3=150, 5=0 y 6=0.
6. El sistema está en “modo 3” y recibe un bit ‘1’ por el puerto digital 12.
7. El sistema realiza el cambio a “modo 4” y establece las señales de los puertos analógicos: 3=0, 5=0 y 6=0.
8. El sistema está en “modo 4” y recibe un bit ‘1’ por el puerto digital 12.
9. El sistema realiza el cambio a “modo 1” y establece las señales de los puertos analógicos: 3=0, 5=0 y 6=0.

### Flujos alternativos

**El flujo del sistema puede cambiar en cualquier momento a menos exista una transición en curso: a, b, c...**

- 1a. El sistema recibe una instrucción de conexión por los puertos seriales tx y rx.
  1. El sistema permanece en “modo 1” y establece las señales de los puertos analógicos: 3=0, 5=0 y 6=0.
  2. El sistema se mantiene verificando las instrucciones entrantes por los puertos seriales tx y rx.

3. El sistema recibe una instrucción de “Encendido”
  - 3.1 El sistema establece las señales de los puertos analógicos: 3=IntensidadRespaldo, 5=IntensidadRespaldo y 6= IntensidadRespaldo.
  - 3.2 El sistema salta al punto 1a-2.
4. El sistema recibe una instrucción de “Apagado”.
  - 4.1 El sistema establece las señales de los puertos analógicos: 3=0, 5=0 y 6=0.
  - 4.2 El sistema salta al punto 1a-2.
5. El sistema recibe una instrucción de “Incremento de intensidad”.
  - 5.1 El sistema incrementa la intensidad de respaldo en dos unidades.
  - 5.2 El sistema establece las señales de los puertos analógicos: 3=IntensidadRespaldo, 5=IntensidadRespaldo y 6= IntensidadRespaldo.
  - 5.3 El sistema salta al punto 1a-2.
6. El sistema recibe una instrucción de “Decremento de intensidad”.
  - 6.1 El sistema realiza el decremento de la intensidad de respaldo en dos unidades.
  - 6.2 El sistema establece las señales de los puertos analógicos: 3=IntensidadRespaldo, 5=IntensidadRespaldo y 6= IntensidadRespaldo.
  - 6.3 El sistema salta al punto 1a-2.
7. El sistema recibe una instrucción de “Programación de apagado”.
  - 7.1 El sistema inicializa el conteo de la programación de apagado recibida.
  - 7.2 El sistema recibe una instrucción de “Cancelación del conteo”.
    - 7.2.1 El sistema cancela el conteo.
    - 7.2.2 El sistema salta al punto 1a-2.
  - 7.3 El sistema recibe una instrucción de “Incremento de intensidad”.
    - 7.3.1 El sistema realiza la interrupción del conteo.
    - 7.3.2 El sistema incrementa la intensidad de respaldo en dos unidades.
    - 7.3.3 El sistema regresa al conteo de la programación de apagado.
  - 7.4 El sistema recibe una instrucción de “Decremento de intensidad”.
    - 7.4.1 El sistema realiza la interrupción del conteo

- 7.4.2 El sistema realiza el decremento de la intensidad de respaldo en dos unidades.
- 7.4.3 El sistema regresa al conteo de la programación de apagado.
- 7.5 El sistema recibe una instrucción de “Programación de apagado”.
  - 7.5.1 El sistema salta al punto 1a-7-7.1
- 7.6 El sistema recibe un bit ‘1’ por el puerto digital 12.
  - 7.6.1 El sistema cancela la programación de apagado.
  - 7.6.2 El sistema realiza el cambio a “modo 1”, establece las señales de los puertos analógicos: 5=0, 6=0 y 3=0 y salta al punto 1.
- 8. El sistema recibe una instrucción de “Modo de programación 2”.
  - 8.1 El sistema salta al punto 3.
- 9. El sistema recibe una instrucción de “Modo de programación 3”.
  - 9.1 El sistema salta al punto 5.
- 10. El sistema recibe una instrucción de “Modo de programación 4”.
  - 10.1 El sistema salta al punto 7.
- 11. El sistema recibe una instrucción de “Realiza un efecto”.
  - 11.1 El sistema ejecuta el efecto recibido.
  - 11.2 El sistema recibe una instrucción de “apagado”.
    - 11.2.1 El sistema termina la ejecución del efecto y salta al punto 1a-1.
    - 11.2.2 El sistema detiene la ejecución del efecto y salta al punto 3.
  - 11.3 El sistema recibe una instrucción de conexión por el puerto serial
    - 11.3.1 El sistema detiene la ejecución del efecto, establece las señales de los puertos analógico: 3=0, 5=0 y 6=0 y salta al paso 1a-2.
- 12. El sistema recibe una instrucción de “Simular estancia”.
  - 12.1 El sistema comienza la simulación de estancia.
  - 12.2 El sistema recibe un bit ‘1’ por el puerto digital 12.
    - 12.2.1 El sistema detiene la simulación de estancia, realiza el cambio a “modo 1”, establece las señales de los puertos analógicos: 5=0, 6=0 y 3=0 y salta al punto 3.

12.3 El sistema recibe una instrucción de conexión por el puerto serial.

12.3.1 El sistema detiene la simulación de estancia, realiza el cambio a “modo 1”, establece las señales de los puertos analógicos: 5=0, 6=0 y 3=0 y salta al punto 1a-2.

3a. El sistema recibe una instrucción de conexión por los puertos seriales tx y rx.

1. El sistema cambia a “modo 1”, establece las señales de los puertos analógicos: 3=0, 5=0 y 6=0.
2. El sistema salta al punto 1a-2

5a. El sistema recibe una instrucción de conexión por los puertos seriales tx y rx.

1. El sistema cambia a “modo 1”, establece las señales de los puertos analógicos: 3=0, 5=0 y 6=0.
2. El sistema salta al punto 1a-2

7a. El sistema recibe una instrucción de conexión por los puertos seriales tx y rx.

1. El sistema cambia a “modo 1”, establece las señales de los puertos analógicos: 3=0, 5=0 y 6=0.
2. El sistema salta al punto 1a-2.

7b. El sistema recibe un bit ‘1’ por el puerto digital 8.

1. El sistema establece las señales de los puertos analógicos: 3=IntensidadRespaldo, 5=IntensidadRespaldo y 6= IntensidadRespaldo.
2. El sistema comienza a verificar el puerto digital 8 cada 125 milisegundos
3. El sistema recibe un bit ‘0’ por el puerto digital 8.
4. El sistema realiza un conteo de 15 segundos.

4.1 El sistema recibe un bit ‘1’ por el puerto digital 8.

4.1.1 El sistema reinicia el conteo de 15 segundos y regresa al punto 7b-4.

4.2 El sistema termina el conteo de los 15 segundos

4.3 El sistema establece las señales de los puertos analógicos: 3= 0, 5=0 y 6= 0 y regresa al punto 7.

### 10.2.2 Implementación del sistema

La implementación del sistema fue la única etapa del desarrollo del proyecto en la que se presentaron problemas no triviales. Más adelante se mencionan, así como la solución implementada para éstos.

Para escribir el código de programación para la implementación del sistema se utilizó, nuevamente, el entorno de desarrollo integrado de Arduino que hace uso de un lenguaje de programación propio, basado en el lenguaje de programación C. El sistema se desarrolló a partir del modelo de casos de uso de texto. No se realizó pseudocódigo ni diagrama de flujo sino que se transformó directamente del modelo con los siguientes pasos:

1. Se programó el escenario principal, es decir, los puntos 1-9 del modelo de casos de uso de texto. Un proceso por cada caso. Los procesos creados en éste punto son ejecutados por el procedimiento loop mostrado en la Figura 84, equivalente al proceso main en C.

```
Version1.5
pinMode(salidaLedCentral, OUTPUT);
pinMode(sensorPresencia,INPUT);//Se determina el puert
pinMode(pulsador,INPUT);
}

//El loop() es el ciclo que se encargará de ejecutar el
void loop() {
  //El siguiente ciclo se encarga de vaciar el buffer de
  while (Serial.available()) {
    delay(3);
    char c = Serial.read();
    readString += c; //Esto es una concatenación ya que
  }
  //Condición que engloba a todos los procedimientos que
  //Es muy importante ésta condición de lo contrario en
  if (readString.length() >0) {
    //Serial.println(readString);
    //Cuando un dispositivo android se conecta mediante
    if(readString == "CONECTED")
    {
      registroModo=1;
      banderaEncendidoApagado=0;
      analogWrite(salidaLedIzquierda,0);
      analogWrite(salidaLedDerecha,0);
      analogWrite(salidaLedCentral,0);
    }
    //Procedimiento para encendido paulatino activado de
    if (readString == "ON" && banderaEncendidoApagado==0
```

Figura 84. Procedimiento principal.



2. Cada proceso creado en el punto anterior fue complementado con sus correspondientes flujos alternativos. Un ejemplo de esto se ve en la Figura 85.

```
Version1.5
if (readString == "ON" && banderaEncendidoApagado==0 && regis
{
  if (banderaEncendidoApagado == 0)
  {
    //ciclo que enciende paulatinamente la iluminación a la m
    for(intensidad=0;intensidad <256;intensidad++)
    {
      analogWrite (salidaLedIzquierda,intensidad);
    }
  }
}
```

Figura 85. Flujo alternativo para el procedimiento de encendido.

3. De acuerdo al modelo de casos de uso notamos que existen sub-flujos alternativos por lo tanto se completaron uno a uno los flujos alternativos con sus correspondientes sub-flujos alternativos.
4. Se corrió el compilador del IDE de Arduino sobre el código escrito para buscar errores léxicos y sintácticos.
5. Se corrigieron todos los errores léxicos y sintácticos.
6. Se corrió nuevamente el compilador del IDE de Arduino sobre el código escrito y se verificaron los espacios requeridos por el código objeto generado.

Después de éste procedimiento se presenta el primer problema: las variables locales y globales para los procesos ocupan más de 2kb en memoria principal.

Solución:

Se analizaron las variables globales y se identificó cuáles de ellas no necesitaban más de un byte en memoria principal que es el valor mínimo que podemos asignar a una variable ya que la arquitectura de Arduino Uno R3 es de 8 bits. Algunas variables de ejemplo se muestran en la Figura 86.

```
byte banderaProcesoApagado = 0;//Es una bandera para el sistema.
byte banderaSalidaCiclos=0;//Es una bandera que nos ayuda para ate
int contSimulacionEstancia=0;//Es un contador que tomará valores
byte banderaEstadoSimulacion=0;//Es una bandera que indica en que
byte registroModo=1;//Es un dato que el sistema utiliza para saber

byte salidaLedIzquierda = 5;//Determina el puerto analógico por el
byte salidaLedDerecha = 6;//Determina el puerto analógico por el c
byte salidaLedCentral = 3;//Determina el puerto analógico por el c

byte sensorPresencia = 8;////Determina el puerto digital por el qu
byte pulsador = 12;//Determina el puerto digital por el que entr

byte banderaEncendidoApagado=0;//Es una bandera que indica al sist
```

Figura 86. Algunas variables tipo byte.

Al realizar los cambios el uso de memoria se redujo en gran medida, hasta un 13% del total utilizable como se muestra en la Figura 87.

```
Compilado
Global variables use 269 bytes (13%) of dynamic memory, leaving
1,779 bytes for local variables. Maximum is 2,048 bytes.
```

Figura 87. Uso de memoria dinámica del sistema.

Después de solucionar el primer problema se presentó inmediatamente el segundo problema: algunos de los procesos son demasiado pesados y su ejecución se realiza con retardos, es decir, el sistema no parece funcionar en tiempo real. Se realizó el siguiente procedimiento para solucionar el problema:

1. Se identificaron aquellos procesos que se podían dividir como un conjunto de subprocesos independientes. Por ejemplo el proceso de cambiar intensidad se dividió en los dos procesos subir intensidad y bajar intensidad como se muestra en la Figura 88 donde el proceso "U" corresponde a subir intensidad y el proceso "D" corresponde a bajar intensidad.

```
    }
    banderaEncendidoApagado = 0;
}
//Procedimiento para ajustar la intensidad (aumentar UP).
if(readString == "U" && intensidad < 254 && banderaEncendidoApagado == 1)
{
    intensidad++;
    intensidad++;
    analogWrite(salidaLedIzquierda, intensidad);
    analogWrite(salidaLedDerecha, intensidad);
    analogWrite(salidaLedCentral, intensidad);
    registroIntensidad=intensidad;
}
//Procedimiento para ajustar la intensidad (reducir DOWN).
if(readString == "D" && intensidad > 4 && banderaEncendidoApagado == 1)
{
    intensidad--;
    intensidad--;
    analogWrite(salidaLedIzquierda, intensidad);
    analogWrite(salidaLedDerecha, intensidad);
}
```

Figura 88. Procesos independientes para subir o bajar intensidad.

- De todos los procesos resultantes, se identificó aquellos que compartían tareas similares y se creó un proceso independiente que brindara ese servicio para todos esos procesos. Por ejemplo para los procesos de programación de apagado el conteo se realiza de la misma forma, por ello se creó un proceso que se encarga de realizar estos conteos para todas las programaciones de apagado. El inicio de éste proceso se observa en la Figura 89.

```
Version1.5
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
}
//Este es el proceso que se encarga de realizar el conteo del tiempo de apagad
if(banderaProcesoApagado == 1)
{
  while(contMinutos >=0)
  {
    while(contCuartosMinutos <= 15000)//Equivalente a 15 segundos que son 1500
    {
      contCuartosMinutos=contCuartosMinutos+1;
    }
  }
}
```

Figura 89. Proceso para realizar un conteo de programación de apagado, independientemente del tiempo.

- Se corrigieron los errores léxicos y sintácticos
- Finalmente se generó el código objeto del sistema y se verificó que el tamaño no excediera los 32kb disponibles para el sistema y datos permanentes.

Estos procedimientos de prueba y corrección se realizaron de forma iterativa, cada iteración generó una versión nueva del sistema. Al final se realizaron cinco iteraciones y se generaron las versiones del sistema 1.1, 1.2, 1.3, 1.4, y 1.5.

**Para ver el código resultante ir al apéndice A en la sección de apéndices.**

Este sistema que es la versión final (versión 1.5) del sistema ocupa 50% del espacio de almacenamiento permanente y 18% del espacio cargado en memoria principal como se observa en la Figura 90. El sistema se cargó a la arquitectura Arduino de forma habitual con el cable de datos USB.

```
Compilado
Sketch uses 16,248 bytes (50%) of program storage space. Maximum is 32,256 bytes.
Global variables use 370 bytes (18%) of dynamic memory, leaving 1,678 bytes for local variables. Maximum is 2,048 bytes.
1743
```

Figura 90. Información final de la compilación del sistema.

El desarrollo del sistema se realizó en un aproximado de 70 horas.

## 11 Desarrollo de carcasa

Para finalizar se desarrolló la carcasa para el dispositivo general, se realizó en dos etapas: la etapa de diseño y la etapa de construcción.

### 11.1 Diseño:

Se consideraron los siguientes requerimientos para el diseño de la carcasa:

1. La carcasa debe tener el espacio necesario para albergar los módulos de potencia, recepción de instrucciones por bluetooth, recepción de señales de presencia y botón de control y el de procesamiento de instrucciones.
2. La carcasa no debe generar interferencias sobre alguno de los componentes y debe ser de un material ligero, maleable y fácil de trabajar.
3. La carcasa debe permitir al sensor de presencia un rango de mínimo 120 grados para censar.
4. La carcasa debe tener expuestas la salida USB del Arduino así como la de alimentación, además debe tener orificios para las salidas de los módulos de iluminación.
5. La carcasa debe permitir que los indicadores de los dispositivos estén a la vista.
6. La carcasa debe permitir el flujo de aire frío hacia los transistores para evitar sobre calentamiento.

La carcasa se diseñó en dos partes: la parte frontal y la parte posterior. Los diseños fueron realizados mediante el software de diseño Vectric Aspire 8 en el cual se elaboraron los vectores necesarios y se realizaron las simulaciones 3D.

#### 11.1.1 Parte posterior

Para la parte posterior se seleccionó el material MDF por ser barato y tener propiedades aptas para los requerimientos: ligero y maleable. Después de realizar los vectores se realizó una simulación en 3D, mostrada en la Figura 91, para verificar el diseño antes de su construcción.



Figura 91. Pre-visualización 3D de la parte posterior de la carcasa.

### 11.1.2 Parte frontal:

Para la parte frontal se seleccionó el material acrílico por ser fácil de conseguir y tener propiedades aptas para los requerimientos: ligero, translucido, maleable y pueden obtenerse buenos acabados. Después de realizar los vectores se realizó una simulación en 3D, como se muestra en la Figura 92, para verificar el diseño antes de su construcción.

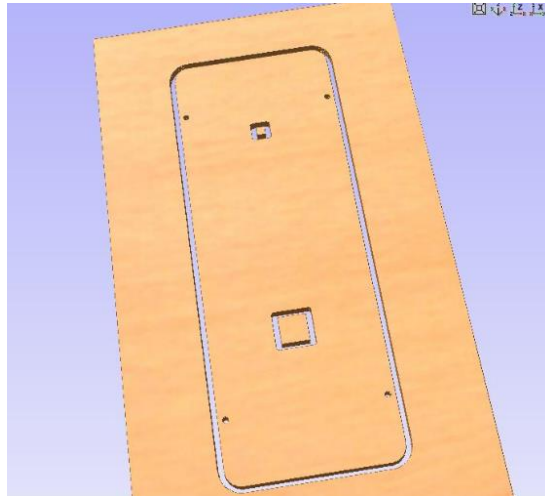


Figura 92. Pre-visualización 3D de la parte frontal de la carcasa.

### 11.2 Construcción:

De los múltiples procesos que existen en la industria se seleccionó el proceso de fresado por control numérico computarizado.

Se utilizó nuevamente un router CNC X2001 controlado por el software de control numérico computarizado Mach 3.

Para la generación de código G, se utilizó el software de modelado (CAM) Vectric Aspire 8. Mostrado en la Figura 93.

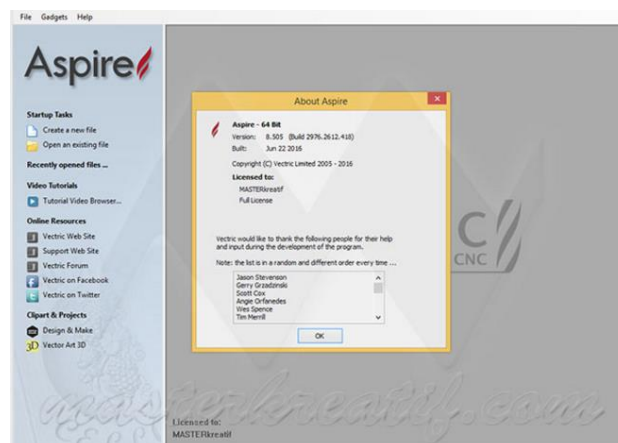


Figura 93. Software CAM Vectric Aspire.

El proceso fue el siguiente:

1. Con ayuda del software de modelado Vectric Aspire 8, se generaron los vectores correspondientes a éstos. Por ejemplo los vectores para la parte frontal se muestran en la Figura 94.

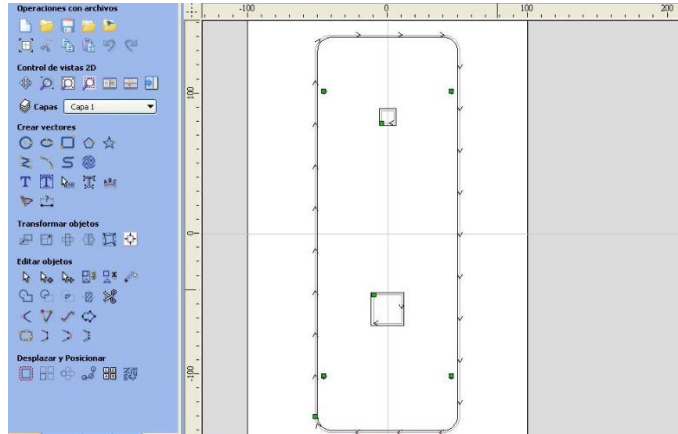


Figura 94. Vectores para la parte posterior de la carcasa.

2. Se generó el código G para realizar el fresado. De forma similar a como se realizó con los circuitos electrónicos.
3. Se fijaron los materiales a la base del router CNC.
4. Se ejecutó el software de control numérico Mach 3 y se cargó el código generado en el punto 2.
5. Se inició el proceso de fresado el cual duró alrededor de 15 minutos.
6. Al terminar el proceso de fresado se sacaron y limpiaron las piezas. Las piezas fresadas se muestran en las Figuras 95 y 96. Las piezas coinciden con los diseños en forma y medidas.



Figura 95. Parte posterior de la carcasa después de ser fresada.



Figura 96. Parte frontal de la carcasa después de ser fresada.

7. La parte posterior de la carcasa se pintó de negro. El resultado se muestra en la Figura 97.

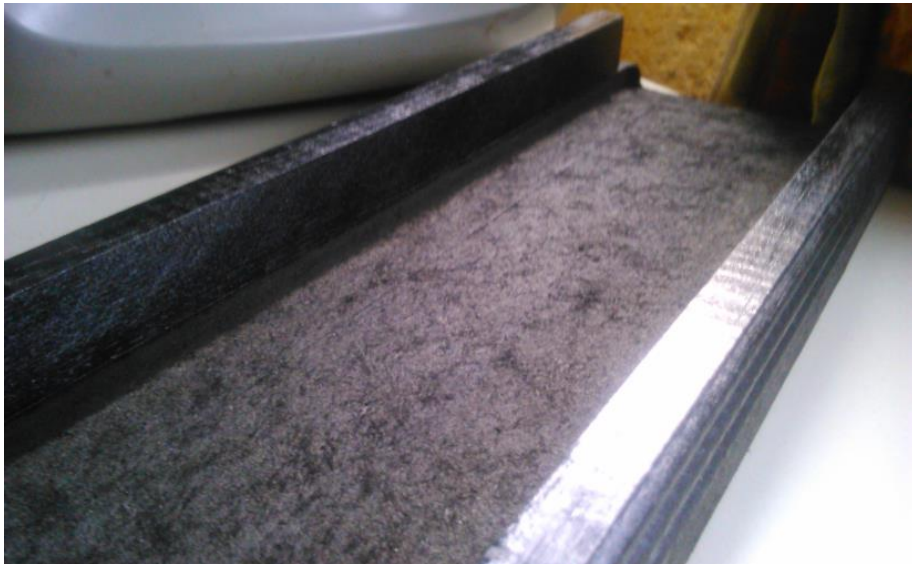


Figura 97. Parte posterior de la carcasa después de ser pintada.

8. Finalmente se comprobó que las piezas empalmaban perfectamente y estaban listas para su armado como se muestra en la Figura 98.



Figura 98. Carcasa lista.



## 12 Ensamblado

Todos los elementos fueron diseñados para encajar perfectamente por lo cual el proceso de construcción final fue relativamente simple y se siguió el siguiente proceso.

1. Se fijaron los componentes dentro la parte posterior de la carcasa con ayuda de tornillos de 1/16 de pulgada como se muestra en las Figuras 99, 100, 101 y 102.

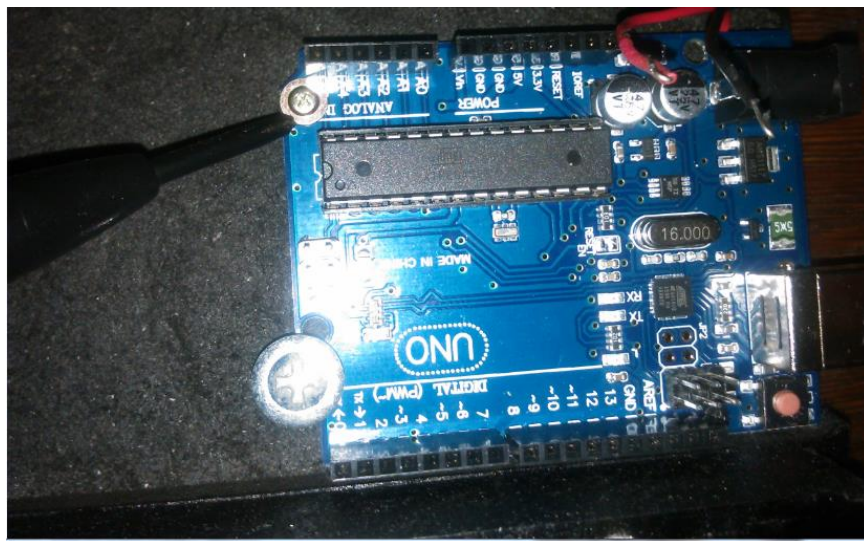


Figura 99. Fijación de la arquitectura de Arduino en la carcasa.

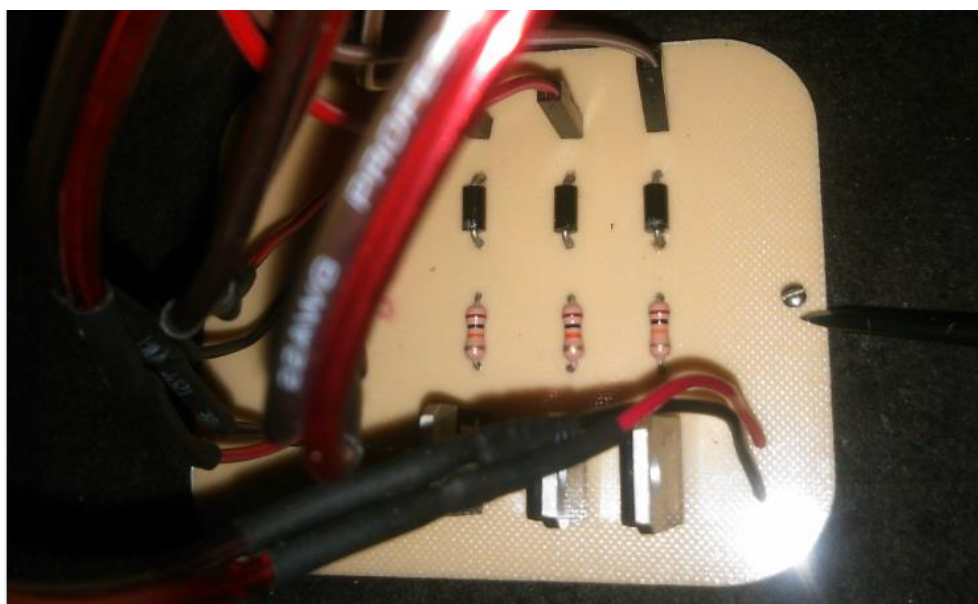


Figura 100. Fijación del módulo de potencia a la carcasa.

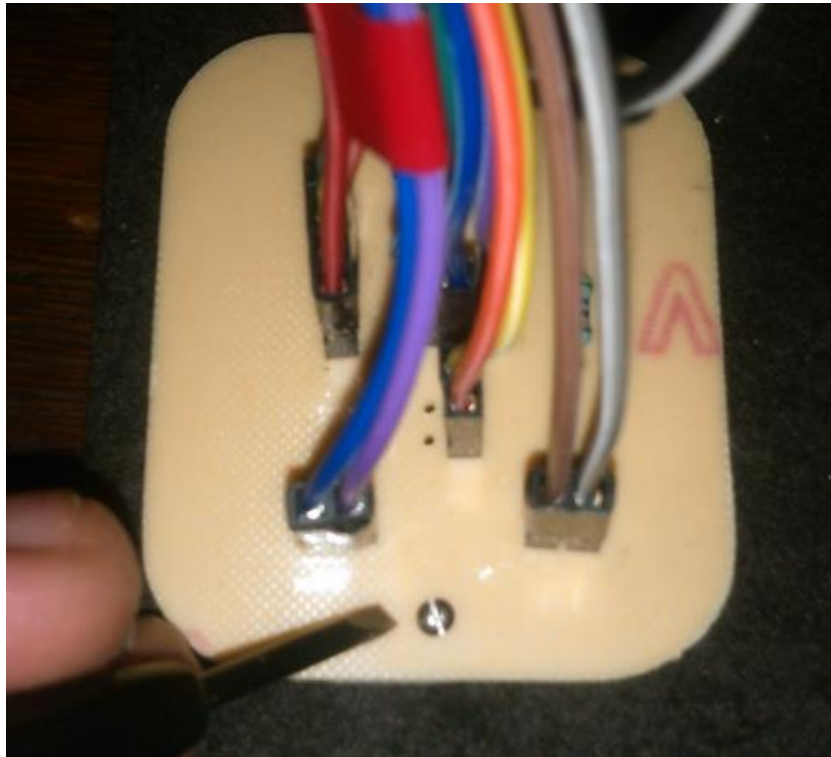


Figura 101. Fijación del módulo de recepción de señales de presencia y botón de control.

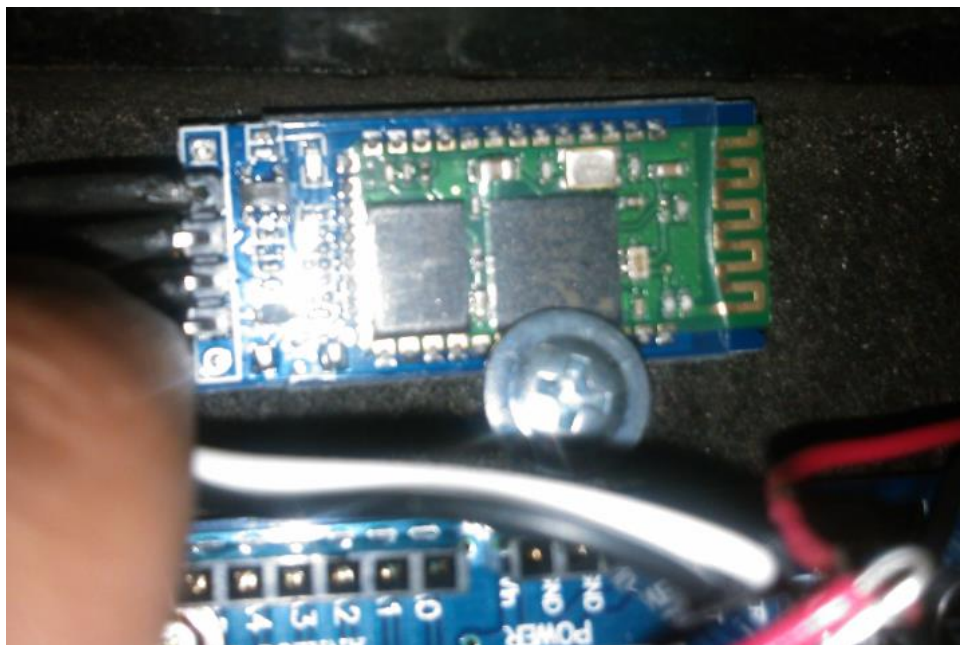


Figura 102. Fijación del módulo bluetooth.

2. Se colocó el botón de control y el sensor PIR en la parte frontal de la carcasa. Estos componentes entran sólo a presión como se muestra en la Figura 103.

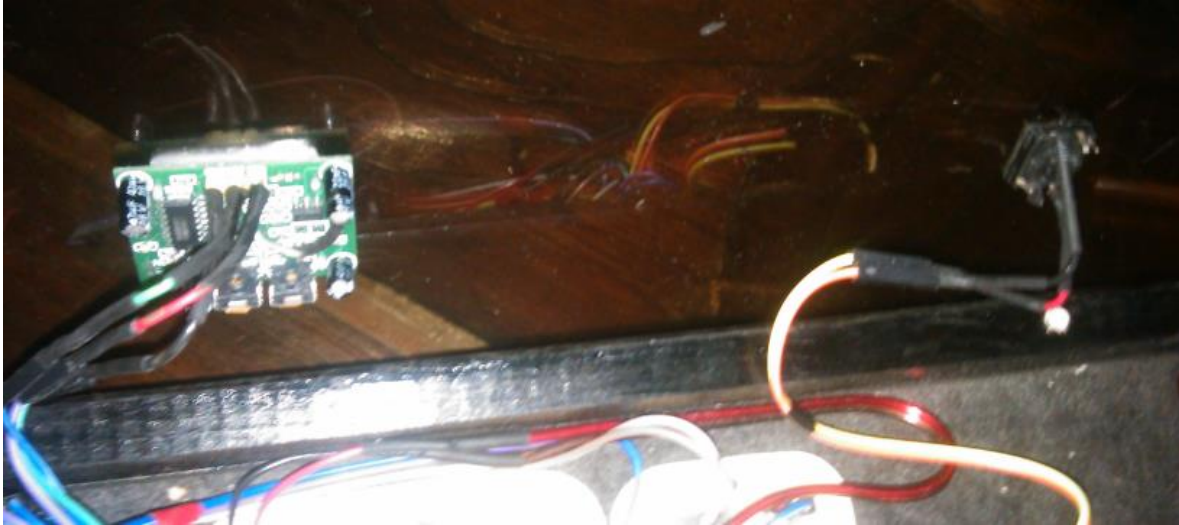


Figura 103. Colocación de botón de control y sensor PIR.

3. Se realizaron las salidas para los módulos de iluminación por los orificios de la parte posterior de la carcasa como se muestra en las Figuras 104 y 105.

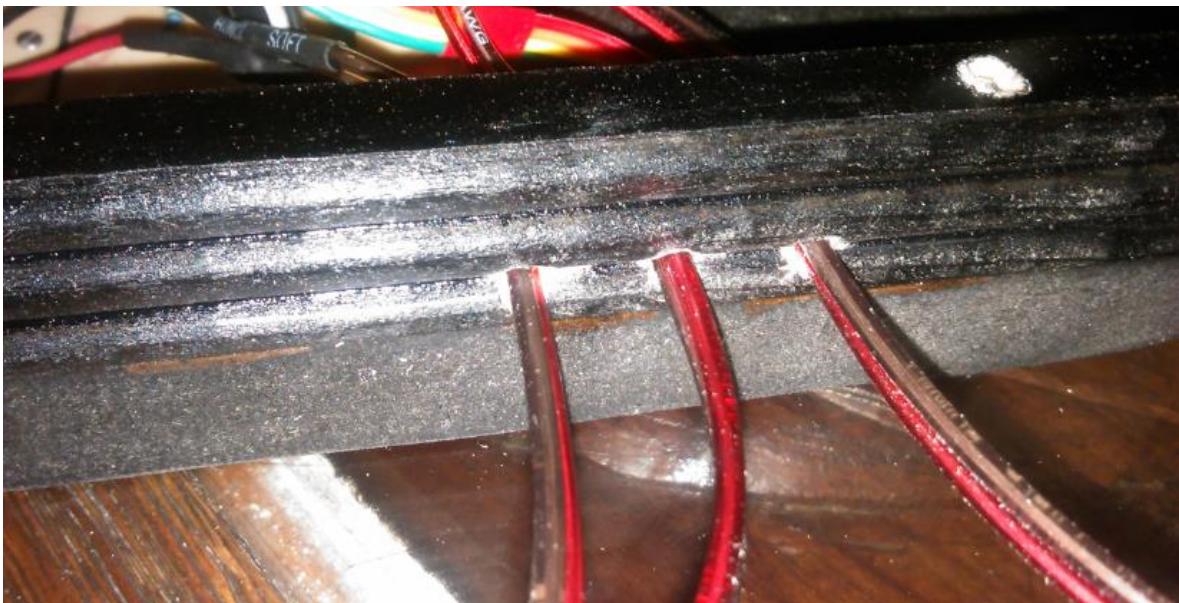


Figura 104. Cables de salida para la conexión de los módulos de iluminación.



Figura 105. Nudos para evitar desplazamiento de los cables hacia el exterior.

4. Se realizaron las conexiones correspondientes según el diagrama de conexión entre módulos. En la Figura 106 se observan los módulos interconectados.

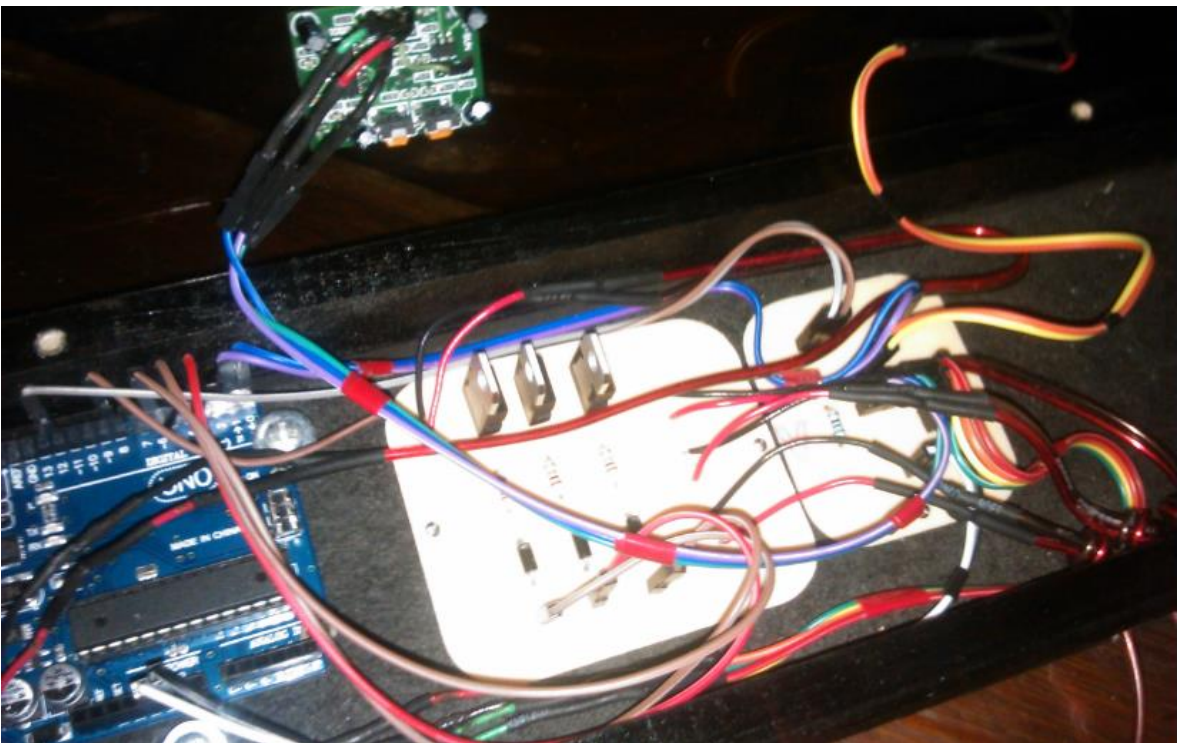


Figura 106. Conexión entre módulos.

5. Se colocaron los tornillos de 1/8 de pulgada para unir la parte posterior y frontal de la carcasa como se muestra en la Figura 107.

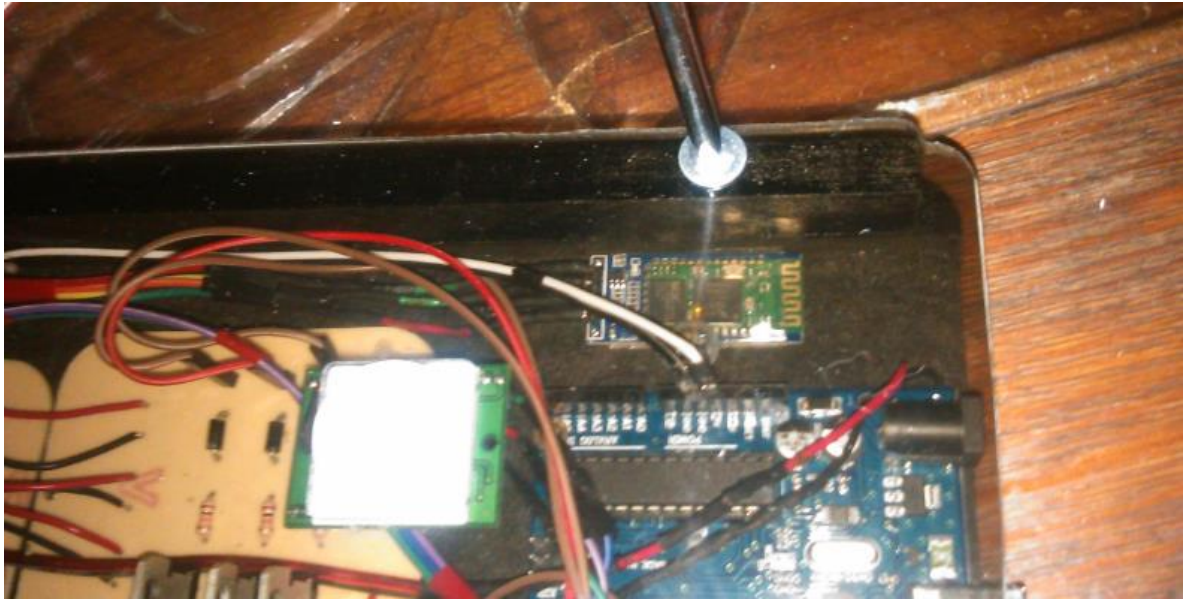


Figura 107. Unión de la parte posterior y frontal de la carcasa.

6. Finalmente se conectaron los módulos de iluminación según el diagrama de conexión entre módulos como se muestra en la Figura 108.

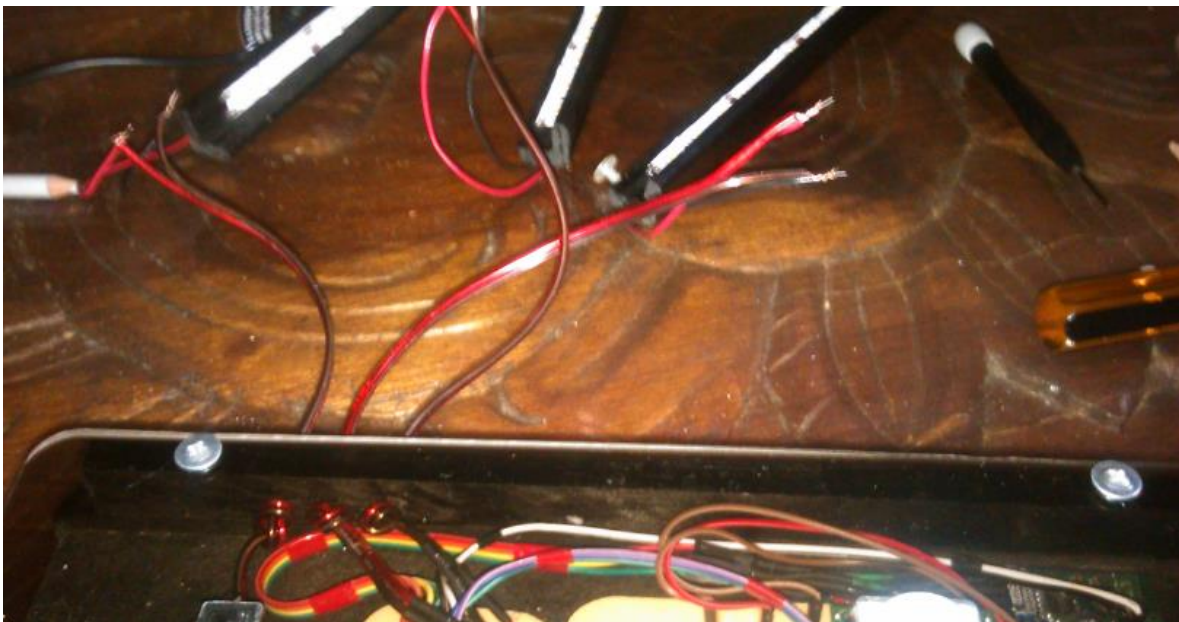


Figura 108. Proceso de conexión de los módulos de iluminación.

Después de este procedimiento el dispositivo está listo para realizar las pruebas necesarias para evaluar el cumplimiento de sus objetivos. El dispositivo se muestra en la Figura 109.

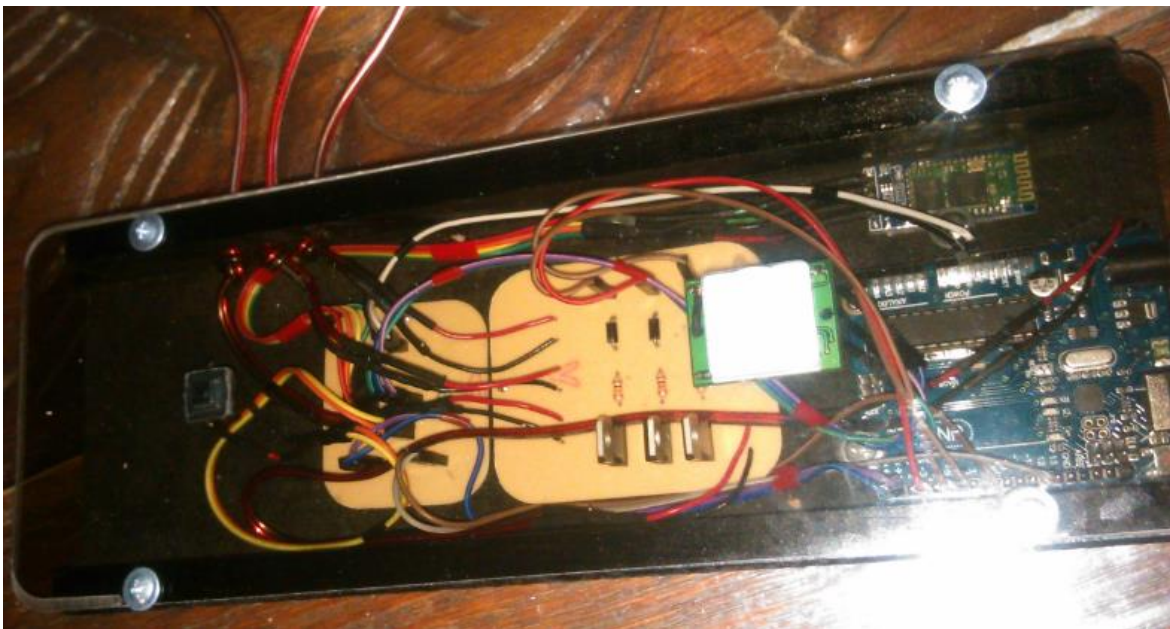


Figura 109. El dispositivo totalmente armado.

## 13 Resultados

Para analizar los resultados podemos abordar las siguientes condiciones que determinan cuando el proyecto se puede dar por terminado.

**El dispositivo general puede realizar todas las funciones propuestas:**

- El sistema se conecta a la alimentación e inicia en modo 1, que es apagado, de forma inmediata como se esperaba.
- Al presionar el botón de control el sistema cambia de modo 1 a modo 2 y encienden las iluminaciones izquierda y derecha a una intensidad de 80% como se tenía programado.
- Al presionar el botón de control nuevamente el sistema de iluminación cambia de modo 2 a modo 3 y enciende la iluminación central a un 60% como se programó previamente.
- Al presionar el botón de control nuevamente el sistema de iluminación cambia de modo 3 a modo 4 y se activa el sensor de movimiento el cual funciona correctamente: enciende sólo ante la presencia de una persona y el tiempo de encendido es de 15 segundos tal como se tenía planeado.
- Al presionar el botón de control nuevamente el sistema regresa al modo 1 tal como se tenía previsto.
- Al conectar el sistema de iluminación con un dispositivo Android mediante la aplicación, el sistema indica la conexión apagándose y respondiendo a las instrucciones enviadas desde el celular.
- El encendido desde el dispositivo Android se realiza de forma progresiva y se respeta la intensidad guardada tal como se tenía previsto.
- El apagado desde el dispositivo Android se realiza de forma progresiva como se tenía pensado.
- Al rotar el dispositivo Android hacia el lado derecho la intensidad del sistema de iluminación aumenta y al rotarlo hacia el lado izquierdo la intensidad disminuye tal y como se programó.
- Al enviar una programación de apagado desde un dispositivo Android el sistema se apaga en el tiempo previsto tal y como debería de suceder. Para saber esto se utilizó un cronometro.
- Al enviar una funcionalidad de prueba desde un dispositivo Android el sistema de iluminación reacciona de forma esperada de acuerdo a cada funcionalidad
- Al enviar un modo de funcionamiento desde un dispositivo Android el sistema de iluminación se coloca en el modo correspondiente.

- Al activar la simulación de estancia desde un dispositivo Android el sistema de iluminación realiza el encendido y apagado dentro de los rangos previstos.

**El dispositivo general no presenta ninguna anomalía de las comunes en circuitos electrónicos como calentamientos o reinicios repentinos:**

Se probó el dispositivo a su máxima intensidad durante 24 horas continuas y no presentó ningún tipo de calentamiento o anomalía.

**El dispositivo general no presenta problemas de programación como ejecución de instrucciones no solicitadas o retrasos notables en la ejecución de los procesos que impidan su correcto funcionamiento:**

El dispositivo ejecuta todas las instrucciones de forma inmediata y la experiencia de uso es buena. Después de cada evento el sistema de iluminación reacciona de la forma esperada en tiempo y forma de realizar los procesos.

**El dispositivo general puede funcionar durante horas continuas sin presentar anomalías de hardware o software:**

El sistema de iluminación se puso en funcionamiento durante una semana ininterrumpida y no presento ningún problema o anomalía de hardware o software.



## 14 Conclusiones

El control de la iluminación mediante sistemas informáticos es posible; se puede controlar el comportamiento de la iluminación en la vida real mediante software ejecutado sobre determinada arquitectura adaptada a nuestras necesidades. Este tipo de implementaciones se pueden realizar con ayuda de las tecnologías actuales.

Las metodologías de diseño para hardware y software sí son efectivas para sobrellevar la carga de trabajo que puede suponer el desarrollo de un proyecto de domótica y brindan la abstracción suficiente para satisfacer los requerimientos encontrados para el proyecto.

El control de la iluminación dentro de una habitación mediante un dispositivo Android realmente brinda una experiencia distinta a la forma habitual de utilizar la iluminación y es posible que pueda convertirse en una necesidad a corto plazo.

Para proyectos posteriores se puede pensar en diseñar e implementar un dispositivo similar pero que funcione mediante la tecnología Wifi. También podemos pensar en modificaciones que integren inteligencia artificial al diseño actual o un software para programar efectos mediante una interfaz gráfica sin necesidad de utilizar código.

## 15 Referencias

- [1]"Revista de iluminación online", *Iluminet.com*, 2016. [Online]. Available: <http://www.iluminet.com/>. [Último acceso: 27- Nov- 2016].
- [2] R. González Ramírez, "*Control de iluminación con tecnología LED*" proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2015.
- [3] C. Hernández Borja, "*Iluminación inteligente de escenarios*" proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2005.
- [4] C. Pérez Carbajal, "*Sistema de iluminación con control inalámbrico infrarrojo basado en tecnología leds*" proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2015.
- [5]"Philips Hue", *Www2.meethue.com*, 2016. [Online]. Available: <http://www2.meethue.com/es-mx/>. [Último acceso: 27- Nov- 2016].
- [6]"Bombillas Led Wifi controladas con el móvil", *Luzwifi.com*, 2016. [Online]. Available: <http://www.luzwifi.com/>. [Último acceso: 27- Nov- 2016].
- [7]"JUNG- eNet Control de la iluminación Técnica", *Jung.de*, 2016. [Online]. Available: <http://www.jung.de/es/925/productos/tecnica/control-de-la-iluminacion/enet/>. [Último acceso: 27- Nov- 2016].
- [8] P. Buban M.L. Schmitt, "*Electricidad y Electrónica Básicas - Conceptos y Aplicaciones*", McGraw-Hill, 1983.
- [9] P. M. Buckley, A. H. Hoskyns, "*Basic Electronic Circuits*", E. & F. N. Spon Ltd, 1980.
- [10]"Wi-Fi alliance ", *Wi-Fi.org*, 2016. [Online]. Available: <http://www.wi-fi.org/>. [Último acceso: 03- Dic- 2016].
- [11]"Bluetooth", *bluetooth.com*, 2016. [Online]. Available: <http://www.bluetooth.com/>. [Último acceso: 03- Dic- 2016].
- [13] Becky Stewart, "*Adventures in Arduino*", Wiley, 2015.
- [14]"Circuitos Eléctricos", [ftp.ehu.es](http://ftp.ehu.es), 2016. [Online]. Available: [http://ftp.ehu.es/cidira/dptos/depjt/Tecnologia/BK-ANGEL/Presentaciones/02\\_Circuitos%20Impresos.pdf/](http://ftp.ehu.es/cidira/dptos/depjt/Tecnologia/BK-ANGEL/Presentaciones/02_Circuitos%20Impresos.pdf/). [Último acceso: 03- Dic- 2016].

## 16 Apéndices

### 16.1 A

int contMinutos = -1; //Sirve para realizar los conteos de tiempo, se inicializa en -1 para indicar al sistema que no hay ninguna programación de apagado al iniciar. Se expresa en cuartos de minuto

int contCuartosMinutos=0;//Sirve para contar 1/4 de minuto es decir 15000 milisegundos (15 segundos).

int contSensorPir = -1;//Sirve para contar el tiempo apartir de que el sensor de presencia envió un bit alto.

//Nota: byte es el tipo de dato más chico en arduino, NO se manejan datos tipo bit por software, sólo por lectura de hardware.

//Nota: Los puertos analógicos se manejan por un ancho de pulso que puede variar desde 0 a 255.

byte banderaProcesoApagado = 0;//Es una bandera para el sistema. Toma el valor de 1 cuando existe un proceso de apagado en curso y 0 cuando no existe un proceso de apagado.

byte banderaSalidaCiclos=0;//Es una bandera que nos ayuda para atender las interrupciones y salir de los ciclos anidados.(1)se detecta una interrupción y (0) No hay interrupción.

int contSimulacionEstancia=0;//Es un contador que tomará valores random para realizar simulación la de estancia.

byte banderaEstadoSimulacion=0;//Es una bandera que indica en que parte del proceso de simulación se está en un tiempo determinado.(1)Simulando encendido(0)Simulando apagado.

byte registroModo=1;//Es un dato que el sistema utiliza para saber en que modo de funcionamiento se encuentra en cada momento.

byte salidaLedIzquierda = 5;//Determina el puerto analógico por el que saldrá la señal de la iluminación izquierda (Ver en la arquitectura de arduino).

byte salidaLedDerecha = 6;//Determina el puerto analógico por el que saldrá la señal de la iluminación derecha (Ver en la arquitectura de arduino).

byte salidaLedCentral = 3;//Determina el puerto analógico por el que saldrá la señal de la iluminación central (Ver en la arquitectura de arduino).

byte sensorPresencia = 8;////Determina el puerto digital por el que entraran los bits del sensor de presencia (Ver en la arquitectura de arduino).

byte pulsador = 12;//Determina el puerto digital por el que entraran los bits del pulsador (Ver en la arquitectura de arduino).

byte banderaEncendidoApagado=0;//Es una bandera que indica al sistema si la iluminación esta encendida o apagada. (1)Encendido y (0) Apagado.

int intensidad=0;//Nos ayuda a la modificación de la intensidad de forma paulatina en los ciclos.

int registroIntensidad=100;//Guarda la última intensidad programada.

```
int i=0;//Auxiliar para ciclos.
```

```
int a=0;//Auxiliar para ciclos.
```

```
int b=0;//Auxiliar para ciclos.
```

```
int c=0;//Auxiliar para ciclos.
```

```
String readString;// Nos ayuda a almacenar el bufer de datos provenientes del módulo bluetooth.
```

```
void setup() {
```

```
    Serial.begin(9600);// Se configura el puerto serial a una velocidad de 9600 bits por segundo al igual que se configuró el puerto serial del bluetooth. Será velocidad de comunicación entre ambos.
```

```
    pinMode(salidaLedIzquierda, OUTPUT);//Se determina el puerto analógico como puerto de salida.
```

```
    pinMode(salidaLedDerecha, OUTPUT);
```

```
    pinMode(salidaLedCentral, OUTPUT);
```

```
    pinMode(sensorPresencia,INPUT);//Se determina el puerto digital como puerto de entrada.
```

```
    pinMode(pulsador,INPUT);
```

```
}
```

```
//El loop() es el ciclo que se encargará de ejecutar el sistema de forma ciclica.
```

```
void loop() {
```

```
    //El siguiente ciclo se encarga de vaciar el buffer del puerto serial y lo almacena en la variable readString, siempre que el puerto tenga datos en espera.
```

```
    while (Serial.available()) {
```

```
        delay(3);
```

```
        char c = Serial.read();
```

```
        readString += c; //Esto es una concatenación ya que la lectura se realiza caracter por caracter.
```

```
    }
```

```
//Condición que engloba a todos los procedimientos que dependen de los datos del puerto serial.
```

```
//Es muy importante ésta condición de lo contrario en cada ciclo se analizará todo el código y los procesos en ejecución serán muy lentos.
```

```
if (readString.length() >0) {
```

```
    //Serial.println(readString);
```

```
    //Cuando un dispositivo android se conecta mediante la aplicación adecuada se cambia a modo 1 y se cede el control a la aplicación
```

```

if(readString == "CONECTED")
{
registroModo=1;
banderaEncendidoApagado=0;
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
analogWrite(salidaLedCentral,0);
}

//Procedimiento para encendido paulatino activado desde la aplicación.
if (readString == "ON" && banderaEncendidoApagado==0 && registroModo == 1)
{

if(banderaEncendidoApagado == 0)
{
//ciclo que enciende paulatinamente la iluminación a la máxima intensidad.
for(intensidad=0;intensidad <256;intensidad++)
{
analogWrite(salidaLedIzquierda,intensidad);
analogWrite(salidaLedDerecha,intensidad);
analogWrite(salidaLedCentral,intensidad);
delay(5);
}
//ciclo que ajusta paulatinamente la intensidad a la intensidad guardada en registroIntensidad.
for(intensidad=255;intensidad > registroIntensidad;intensidad--)
{
analogWrite(salidaLedIzquierda,intensidad);
analogWrite(salidaLedDerecha,intensidad);
analogWrite(salidaLedCentral,intensidad);
delay(3);
}
intensidad=registroIntensidad;
}
}

```

```

analogWrite(salidaLedIzquierda, intensidad);
analogWrite(salidaLedDerecha,intensidad);
analogWrite(salidaLedCentral,intensidad);
banderaEncendidoApagado=1;
}
//Procedimiento para apagado paulatino activado desde la aplicación.
if (readString == "OFF" && banderaEncendidoApagado==1 && registroModo == 1)
{
registrolIntensidad=intensidad;//Antes de apagar se respalda la intensidad, por si se realizó algún ajuste.
for(i=intensidad; i >= 10;i--)
{
analogWrite(salidaLedIzquierda,i);
analogWrite(salidaLedDerecha,i);
analogWrite(salidaLedCentral,i);
delay(10);
}
for(i=10; i >= 0;i--)
{
analogWrite(salidaLedIzquierda,i);
analogWrite(salidaLedDerecha,i);
analogWrite(salidaLedCentral,i);
delay(15);
}
banderaEncendidoApagado = 0;
}
//Procedimiento para ajustar la intensidad (aumentar UP).
if(readString == "U" && intensidad < 254 && banderaEncendidoApagado ==1 && registroModo == 1)
{
intensidad++;
intensidad++;
analogWrite(salidaLedIzquierda,intensidad);
analogWrite(salidaLedDerecha,intensidad);

```

```

    analogWrite(salidaLedCentral,intensidad);
    registroIntensidad=intensidad;
}
//Procedimiento para ajustar la intensidad (reducir DOWN).
if(readString == "D" && intensidad > 4 && banderaEncendidoApagado ==1 && registroModo == 1)
{
    intensidad--;
    intensidad--;
    analogWrite(salidaLedIzquierda,intensidad);
    analogWrite(salidaLedDerecha,intensidad);
    analogWrite(salidaLedCentral,intensidad);
    registroIntensidad=intensidad;
}
// Procedimiento que establece una programación de apagado de 30 segundos que equivalen a 2/4 de
minuto.
if(readString == "30S" && banderaEncendidoApagado==1 && registroModo == 1)
{
    //Las siguientes 11 lineas ayudan a visualizar la llegada de la programación al dispositivo de iluminación.
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedDerecha,0);
    delay(100);
    analogWrite(salidaLedIzquierda,150);
    analogWrite(salidaLedDerecha,150);
    delay(100);
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedDerecha,0);
    delay(100);
    analogWrite(salidaLedIzquierda,intensidad);
    analogWrite(salidaLedDerecha,intensidad);
    contCuartosMinutos=0;
    contMinutos=2;
    banderaProcesoApagado=1;
}

```

```

}

// Procedimiento que establece una programación de apagado de 5 minutos que equivalen a 20/4 de
minuto.

if(readString == "5" && banderaEncendidoApagado==1 && registroModo == 1)
{
  analogWrite(salidaLedIzquierda,0);
  analogWrite(salidaLedDerecha,0);
  delay(100);
  analogWrite(salidaLedIzquierda,150);
  analogWrite(salidaLedDerecha,150);
  delay(100);
  analogWrite(salidaLedIzquierda,0);
  analogWrite(salidaLedDerecha,0);
  delay(100);
  analogWrite(salidaLedIzquierda,intensidad);
  analogWrite(salidaLedDerecha,intensidad);
  contCuartosMinutos=0;
  contMinutos=20;
  banderaProcesoApagado=1;
}

// Procedimiento que establece una programación de apagado de 15 minutos que equivalen a 60/4 de
minuto.

if(readString == "15" && banderaEncendidoApagado==1 && registroModo == 1)
{
  analogWrite(salidaLedIzquierda,0);
  analogWrite(salidaLedDerecha,0);
  delay(100);
  analogWrite(salidaLedIzquierda,150);
  analogWrite(salidaLedDerecha,150);
  delay(100);
  analogWrite(salidaLedIzquierda,0);
  analogWrite(salidaLedDerecha,0);
  delay(100);
}

```



```

    analogWrite(salidaLedIzquierda,intensidad);
    analogWrite(salidaLedDerecha,intensidad);
    contCuartosMinutos=0;
    contMinutos=60;
    banderaProcesoApagado=1;
}

// Procedimiento que establece una programación de apagado de 30 minutos que equivalen a 120/4 de
minuto.

if(readString == "30" && banderaEncendidoApagado==1 && registroModo == 1)
{
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedDerecha,0);
    delay(100);
    analogWrite(salidaLedIzquierda,150);
    analogWrite(salidaLedDerecha,150);
    delay(100);
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedDerecha,0);
    delay(100);
    analogWrite(salidaLedIzquierda,intensidad);
    analogWrite(salidaLedDerecha,intensidad);
    contCuartosMinutos=0;
    contMinutos=120;
    banderaProcesoApagado=1;
}

// Procedimiento que establece una programación de apagado de 1 hora que equivalen a 240/4 de
minuto.

if(readString == "1H" && banderaEncendidoApagado==1 && registroModo == 1)
{
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedDerecha,0);
    delay(100);
    analogWrite(salidaLedIzquierda,150);

```

```

analogWrite(salidaLedDerecha,150);
delay(100);
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
delay(100);
analogWrite(salidaLedIzquierda,intensidad);
analogWrite(salidaLedDerecha,intensidad);
contCuartosMinutos=0;
contMinutos=240;
banderaProcesoApagado=1;
}

// Procedimiento que establece una programación de apagado de 30 minutos que equivalen a 480/4 de
minuto.
if(readString == "2H" && banderaEncendidoApagado==1 && registroModo == 1)
{
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
delay(100);
analogWrite(salidaLedIzquierda,150);
analogWrite(salidaLedDerecha,150);
delay(100);
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
delay(100);
analogWrite(salidaLedIzquierda,intensidad);
analogWrite(salidaLedDerecha,intensidad);
contCuartosMinutos=0;
contMinutos=480;
banderaProcesoApagado=1;
}

// procedimiento que establece los parametros necesarios para interrumpir y cancelar cualquier
programación de apagado en curso.
if(readString == "0" && banderaEncendidoApagado==1 && registroModo == 1)

```

```

{
  analogWrite(salidaLedIzquierda,0);
  analogWrite(salidaLedDerecha,0);
  delay(100);
  analogWrite(salidaLedIzquierda,150);
  analogWrite(salidaLedDerecha,150);
  delay(100);
  analogWrite(salidaLedIzquierda,0);
  analogWrite(salidaLedDerecha,0);
  delay(100);
  analogWrite(salidaLedIzquierda,intensidad);
  analogWrite(salidaLedDerecha,intensidad);
  contMinutos=-1;
  banderaProcesoApagado=0;
  contCuartosMinutos=0;
}

//Da soporte a la simulación de estancia, si se desea cambiar el intervalo de simulación se deben cambiar
los parametros de la función random, expresado en cuartos de minutos.

if(readString == "SIMFUNCION" && registroModo ==
1)////////////////////////////////////
///
{
  contCuartosMinutos=0;

  //Se inicia la simulación con las iluminaciones izquierda y derecha encendida.
  analogWrite(salidaLedIzquierda,220);
  analogWrite(salidaLedDerecha,220);
  analogWrite(salidaLedCentral,0);

  //Se coloca la bandera en uno para indicar que la simulación está con la iluminación encendida.
  banderaEstadoSimulacion=1;

  while(0==0)
  {

    if(banderaEstadoSimulacion==1)

```

```

{
//Se genera un número random que equivale a el número de cuartos de segundo a simular.
contSimulacionEstancia= random(1,8);
while(contSimulacionEstancia > 0)
{
//Ciclo que se encarga de realizar el conteo de cada cuarto de segundo y da soporte a las
interrupciones cada 50 milisegundos.
while(contCuartosMinutos <= 300)
{
contCuartosMinutos=contCuartosMinutos+1;
delay(50);//Éste, indica cada cuanto se debe verificar una interrupción.

//Interrupción del tipo instrucción del puerto serial
if(Serial.available())
{
banderaSalidaCiclos = 1;
break;
}
//Interrupción del tipo un bit alto por el puerto digital del pulsador
if(digitalRead(pulsador) == HIGH)
{
banderaSalidaCiclos = 1;
break;
}

//Se ha completado un cuarto de minuto y se inicia de nuevo el conteo
if(contCuartosMinutos ==300)
{
contSimulacionEstancia = contSimulacionEstancia -1;
contCuartosMinutos = 0;
break;
}
}

```

```

    }
    if(banderaSalidaCiclos == 1)
    {
        break;
    }
}

//Se apagan las iluminaciones encendidas
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
//Se coloca la bandera en cero para indicar que la simulación está con la iluminación apagada.
banderaEstadoSimulacion=0;
}

//Da soporte a la simulación apagada
if(banderaEstadoSimulacion==0)
{
    //Se genera un número aleatorio que equivale a los cuartos de minutos que se simulará apagado.
    contSimulacionEstancia= random(1,8);
    while(contSimulacionEstancia > 0)
    {
        //Ciclo que da soporte al conteo de cuartos de minuto durante el apagado y da soporte a las
        //interrupciones cada 50 milisegundos
        while(contCuartosMinutos <= 300)
        {
            contCuartosMinutos=contCuartosMinutos+1;
            delay(50);
        }
        //Interrupción del tipo intrucción por el puerto serial.
        if(Serial.available())
        {
            banderaSalidaCiclos = 1;
            break;
        }
    }
}

```

```

}

//Interrupción del tipo bit alto por el puerto digital del pulsador
if(digitalRead(pulsador) == HIGH)
{
    banderaSalidaCiclos = 1;
    break;
}

//Se ha completado el conteo de 15s.
if(contCuartosMinutos ==300)
{
    contSimulacionEstancia = contSimulacionEstancia -1;
    contCuartosMinutos = 0;
    break;
}
}

if(flagSalidaCiclos == 1)
{
    break;
}
}

//Se enciende la iluminación.
analogWrite(salidaLedIzquierda,220);
analogWrite(salidaLedDerecha,220);

//Se coloca la bandera en uno para indicar que la simulación está con la iluminación encendida.
banderaEstadoSimulacion=1;
}

//Salida para atender una interrupción.
if(flagSalidaCiclos == 1)
{

```

```

    banderaSalidaCiclos = 0;

    break;

}

}

}

//Se realiza el cambio a modo de funcionamiento 2 mediante la instrucción SET2 recibida desde el puerto
serial.

if(readString == "SET2" && registroModo == 1)

{

registroModo = 2;

banderaEncendidoApagado = 1;

analogWrite(salidaLedCentral,0);

delay(100);

analogWrite(salidaLedCentral,150);

delay(100);

analogWrite(salidaLedCentral,0);

delay(100);

analogWrite(salidaLedCentral,150);

delay(100);

analogWrite(salidaLedCentral,0);

analogWrite(salidaLedIzquierda,170);

analogWrite(salidaLedDerecha,170);

}

//Se realiza el cambio a modo de funcionamiento 3 mediante la instrucción SET3 recibida desde el puerto
serial.

if(readString == "SET3" && registroModo == 1)

{

registroModo = 3;

analogWrite(salidaLedIzquierda,0);

analogWrite(salidaLedDerecha,0);

analogWrite(salidaLedCentral,0);

delay(100);

analogWrite(salidaLedCentral,150);

```

```

delay(100);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
}
//Se realiza el cambio a modo de funcionamiento 4 mediante la instrucción SET4 recibida desde el puerto
serial.

if(readString == "SET4" && registroModo == 1)
{
registroModo = 4;
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
}

```



```

analogWrite(salidaLedCentral,0);

delay(100);

analogWrite(salidaLedCentral,150);

delay(100);

analogWrite(salidaLedCentral,0);
}

//Este es el proceso que se encarga de realizar el conteo del tiempo de apagado y ejecutar el apagado
cuando termine el conteo.

if(banderaProcesoApagado == 1)
{
while(contMinutos >=0)
{
while(contCuartosMinutos <= 15000)//Equivalente a 15 segundos que son 15000 milisegundos.
{
contCuartosMinutos=contCuartosMinutos+1;

delay(1);// Este procedimiento no debe monopolizar las ejecuciones por ello la verificación de
interrupciones se realiza cada milisegundo. Unidad minima posible.

//Analiza si hay datos en espera en el puerto serial y debe realizar una interrupción.
if(Serial.available())
{
banderaSalidaCiclos = 1;

break;
}

//Analiza si hay un bit alto del pulsador y debe realizar la salida forzada del proceso.
if(digitalRead(pulsador) == HIGH)
{
banderaSalidaCiclos = 1;

break;
}

//Finaliza el conteo de 15 segundos y reinicia.
if(contCuartosMinutos ==15000)

```

```

{
  contMinutos = contMinutos -1;
  contCuartosMinutos = 0;
  break;
}
}
//En caso de interrupcion
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}

//Termina el conteo de los minutos y se realiza el procedimiento de apagado.
if(contMinutos == 0)
{
  registroIntensidad=intensidad;
  for(i=intensidad; i >= 10;i--)
  {
    analogWrite(salidaLedIzquierda,i);
    analogWrite(salidaLedDerecha,i);
    analogWrite(salidaLedCentral,i);
    delay(10);
  }
  for(i=10; i >= 0;i--)
  {
    analogWrite(salidaLedIzquierda,i);
    analogWrite(salidaLedDerecha,i);
    analogWrite(salidaLedCentral,i);
    delay(15);
  }
  contMinutos=-1;
}

```

```

    banderaProcesoApagado=0;
    banderaEncendidoApagado=0;
}
}
}
if(readString == "EFECTO1" && registroModo == 1)
{
while(0 == 0)
{
if(Serial.available())
{
break;
}
if(digitalRead(pulsador) == HIGH)
{
break;
}
delay(40);
analogWrite(salidaLedIzquierda,255);
analogWrite(salidaLedDerecha,255);
analogWrite(salidaLedCentral,255);
delay(40);
if(Serial.available())
{
break;
}
if(digitalRead(pulsador) == HIGH)
{
break;
}
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);

```

```

    analogWrite(salidaLedCentral,0);
}
}
if(readString == "EFECTO2" && registroModo == 1)
{

while(0 == 0)
{
    b=60;
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedCentral,0);
    for(a=0;a<=50;a++)
    {
        analogWrite(salidaLedIzquierda,120);
        delay(b);
        if(Serial.available())
        {
            banderaSalidaCiclos = 1;
            break;
        }
        if(digitalRead(pulsador) == HIGH)
        {
            banderaSalidaCiclos = 1;
            break;
        }
        delay(b);
        if(Serial.available())
        {
            banderaSalidaCiclos = 1;
            break;
        }
    }
}
}

```

```
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
delay(b);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
delay(b);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
delay(b);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
```

```

}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,120);
delay(b);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
delay(b);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
delay(b);
if(Serial.available())

```

```
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
delay(b);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
delay(b);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
analogWrite(salidaLedDerecha,0);
```

```
b=b-1;
}
if(banderaSalidaCiclos == 1)
{
    banderaSalidaCiclos = 0;
    break;
}

for(a=0;a<=20;a++)
{
    analogWrite(salidaLedIzquierda,120);
    delay(50);
    if(Serial.available())
    {
        banderaSalidaCiclos = 1;
        break;
    }
    if(digitalRead(pulsador) == HIGH)
    {
        banderaSalidaCiclos = 1;
        break;
    }
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedDerecha,120);
    delay(50);
    if(Serial.available())
    {
        banderaSalidaCiclos = 1;
        break;
    }
}
```



```

if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
analogWrite(salidaLedDerecha,0);
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}

for(a=0;a<=20;a++)
{
  analogWrite(salidaLedIzquierda,120);
  delay(30);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
  if(digitalRead(pulsador) == HIGH)
  {
    banderaSalidaCiclos = 1;
    break;
  }
  analogWrite(salidaLedIzquierda,0);
  analogWrite(salidaLedDerecha,120);
  delay(30);
  if(Serial.available())

```

```

{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
analogWrite(salidaLedDerecha,0);
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}

for(a=0;a<=40;a++)
{
  analogWrite(salidaLedIzquierda,120);
  delay(20);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
}

```

```
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,120);
delay(20);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
analogWrite(salidaLedDerecha,0);
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}

for(a=200;a<=255;a++)
{
  analogWrite(salidaLedIzquierda,a);
  analogWrite(salidaLedDerecha,a);
  analogWrite(salidaLedCentral,a);
  delay(18);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
}
```

```

}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}

for(a=0;a<=100;a++)
{
  analogWrite(salidaLedIzquierda,255);
  analogWrite(salidaLedDerecha,255);
  analogWrite(salidaLedCentral,255);
  delay(10);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
  if(digitalRead(pulsador) == HIGH)
  {
    banderaSalidaCiclos = 1;
    break;
  }
}
if(banderaSalidaCiclos == 1)

```

```

{
  banderaSalidaCiclos = 0;
  break;
}

for(a=255;a>=0;a--)
{
  analogWrite(salidaLedIzquierda,a);
  analogWrite(salidaLedDerecha,a);
  analogWrite(salidaLedCentral,a);
  delay(1);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
  if(digitalRead(pulsador) == HIGH)
  {
    banderaSalidaCiclos = 1;
    break;
  }
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}

for(a=0;a<=300;a++)
{

```

```

analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
analogWrite(salidaLedCentral,0);
delay(10);
if(Serial.available())
{
banderaSalidaCiclos = 1;
break;
}
if(digitalRead(pulsador) == HIGH)
{
banderaSalidaCiclos = 1;
break;
}
}
if(banderaSalidaCiclos == 1)
{
banderaSalidaCiclos = 0;
break;
}
}
}
if(readString == "EFECTO3" && registroModo == 1)
{
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
while(0 == 0)
{
for(a=0;a <256;a++)
{
analogWrite(salidaLedIzquierda,a);

```

```

analogWrite(salidaLedCentral,a);
delay(4);
if(Serial.available())
{
    banderaSalidaCiclos = 1;
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    banderaSalidaCiclos = 1;
    break;
}
}
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedCentral,0);
if(banderaSalidaCiclos == 1)
{
    banderaSalidaCiclos = 0;
    break;
}
for(a=0;a <256;a++)
{
    analogWrite(salidaLedDerecha,a);
    analogWrite(salidaLedCentral,a);
    delay(4);
    if(Serial.available())
    {
        banderaSalidaCiclos = 1;
        break;
    }
}
if(digitalRead(pulsador) == HIGH)
{

```

```

    banderaSalidaCiclos = 1;
    break;
}
}
analogWrite(salidaLedDerecha,0);
analogWrite(salidaLedCentral,0);
if(flagSalidaCiclos == 1)
{
    banderaSalidaCiclos = 0;
    break;
}
}
}
if(readString == "EFECTO4" && registroModo == 1)
{
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedDerecha,0);
    analogWrite(salidaLedCentral,0);
    while(1 == 1)
    {
        for(a=0;a <200;a++)
        {
            analogWrite(salidaLedIzquierda,a);
            delay(4);
            if(Serial.available())
            {
                banderaSalidaCiclos = 1;
                break;
            }
        }
        if(digitalRead(pulsador) == HIGH)
        {
            banderaSalidaCiclos = 1;

```



```

    break;
}
}
if(banderaSalidaCiclos == 1)
{
    banderaSalidaCiclos = 0;
    break;
}
for(a=200;a>=0;a--)
{
    analogWrite(salidaLedIzquierda,a);
    delay(4);
    if(Serial.available())
    {
        banderaSalidaCiclos = 1;
        break;
    }
    if(digitalRead(pulsador) == HIGH)
    {
        banderaSalidaCiclos = 1;
        break;
    }
}
if(banderaSalidaCiclos == 1)
{
    banderaSalidaCiclos = 0;
    break;
}
for(a=0;a <200;a++)
{
    analogWrite(salidaLedCentral,a);
    delay(4);
}
}
}

```

```

if(Serial.available())
{
    banderaSalidaCiclos = 1;
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    banderaSalidaCiclos = 1;
    break;
}
}
if(banderaSalidaCiclos == 1)
{
    banderaSalidaCiclos = 0;
    break;
}
for(a=200;a>=0;a--)
{
    analogWrite(salidaLedCentral,a);
    delay(4);
    if(Serial.available())
    {
        banderaSalidaCiclos = 1;
        break;
    }
    if(digitalRead(pulsador) == HIGH)
    {
        banderaSalidaCiclos = 1;
        break;
    }
}
if(banderaSalidaCiclos == 1)

```

```

{
  banderaSalidaCiclos = 0;
  break;
}
for(a=0;a <200;a++)
{
  analogWrite(salidaLedDerecha,a);
  delay(4);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
  if(digitalRead(pulsador) == HIGH)
  {
    banderaSalidaCiclos = 1;
    break;
  }
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}
for(a=200;a>=0;a--)
{
  analogWrite(salidaLedDerecha,a);
  delay(4);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
}

```

```

}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}
}
}
if(readString == "EFECTO5" && registroModo == 1)
{
  analogWrite(salidaLedIzquierda,0);
  analogWrite(salidaLedDerecha,0);
  analogWrite(salidaLedCentral,0);
  while(0 == 0)
  {
    for(a=0;a <200;a++)
    {
      analogWrite(salidaLedDerecha,a);
      delay(4);
      if(Serial.available())
      {
        banderaSalidaCiclos = 1;
        break;
      }
    }
  }
  if(digitalRead(pulsador) == HIGH)
  {

```

```

banderaSalidaCiclos = 1;
break;
}
}
if(banderaSalidaCiclos == 1)
{
banderaSalidaCiclos = 0;
break;
}
for(a=200;a>=0;a--)
{
analogWrite(salidaLedDerecha,a);
delay(4);
if(Serial.available())
{
banderaSalidaCiclos = 1;
break;
}
if(digitalRead(pulsador) == HIGH)
{
banderaSalidaCiclos = 1;
break;
}
}
if(banderaSalidaCiclos == 1)
{
banderaSalidaCiclos = 0;
break;
}
for(a=0;a <200;a++)
{
analogWrite(salidaLedCentral,a);

```

```

delay(4);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}
for(a=200;a>=0;a--)
{
  analogWrite(salidaLedCentral,a);
  delay(4);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
  if(digitalRead(pulsador) == HIGH)
  {
    banderaSalidaCiclos = 1;
    break;
  }
}
}

```

```

if(banderaSalidaCiclos == 1)
{
banderaSalidaCiclos = 0;
break;
}
for(a=0;a <200;a++)
{
analogWrite(salidaLedIzquierda,a);
delay(4);
if(Serial.available())
{
banderaSalidaCiclos = 1;
break;
}
if(digitalRead(pulsador) == HIGH)
{
banderaSalidaCiclos = 1;
break;
}
}
if(banderaSalidaCiclos == 1)
{
banderaSalidaCiclos = 0;
break;
}
for(a=200;a>=0;a--)
{
analogWrite(salidaLedIzquierda,a);
delay(4);
if(Serial.available())
{
banderaSalidaCiclos = 1;

```

```

    break;
}
if(digitalRead(pulsador) == HIGH)
{
    banderaSalidaCiclos = 1;
    break;
}
}
if(banderaSalidaCiclos == 1)
{
    banderaSalidaCiclos = 0;
    break;
}
}
}
if(readString == "EFECTO6" && registroModo == 1)
{
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedDerecha,0);
    while(0 == 0)
    {
        for(a=0;a <256;a++)
        {
            analogWrite(salidaLedCentral,a);
            delay(30);
            if(Serial.available())
            {
                banderaSalidaCiclos = 1;
                break;
            }
        }
        if(digitalRead(pulsador) == HIGH)
        {

```



```

banderaSalidaCiclos = 1;
break;
}
}
if(banderaSalidaCiclos == 1)
{
banderaSalidaCiclos = 0;
break;
}
for(int b=255;b >= 0;b--)
{
analogWrite(salidaLedCentral,b);
delay(15);
if(Serial.available())
{
banderaSalidaCiclos = 1;
break;
}
if(digitalRead(pulsador) == HIGH)
{
banderaSalidaCiclos = 1;
break;
}
}
if(banderaSalidaCiclos == 1)
{
banderaSalidaCiclos = 0;
break;
}
for(int c=0;c<=40;c++)
{
analogWrite(salidaLedCentral,0);

```

```

delay(50);
if(Serial.available())
{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}
}
}
if(readString == "EFECTO7" && registroModo == 1)
{
  analogWrite(salidaLedIzquierda,0);
  analogWrite(salidaLedDerecha,0);
  analogWrite(salidaLedCentral,0);
  while(0 == 0)
  {
    for(a=0;a <256;a++)
    {
      analogWrite(salidaLedCentral,a);
      delay(30);
      if(Serial.available())

```

```

{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
}
if(banderaSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}
for(int b=255;b >= 0;b--)
{
  analogWrite(salidaLedCentral,b);
  delay(15);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
  if(digitalRead(pulsador) == HIGH)
  {
    banderaSalidaCiclos = 1;
    break;
  }
}
if(banderaSalidaCiclos == 1)
{

```

```

    banderaSalidaCiclos = 0;
    break;
}
for(a=0;a <256;a++)
{
analogWrite(salidaLedIzquierda,a);
analogWrite(salidaLedDerecha,b);
delay(30);
if(Serial.available())
{
    banderaSalidaCiclos = 1;
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    banderaSalidaCiclos = 1;
    break;
}
}
if(flagSalidaCiclos == 1)////////////////////////////////////
{
    banderaSalidaCiclos = 0;
    break;
}
for(int b=255;b >= 0;b--)
{
analogWrite(salidaLedIzquierda,b);
analogWrite(salidaLedDerecha,b);
delay(15);
if(Serial.available())
{
    banderaSalidaCiclos = 1;

```

```

    break;
}
if(digitalRead(pulsador) == HIGH)
{
    banderaSalidaCiclos = 1;
    break;
}
}
if(banderaSalidaCiclos == 1)
{
    banderaSalidaCiclos = 0;
    break;
}
}
}
if (readString == "EFECTO8" && registroModo == 1)
{
    while(0 == 0)
    {
        if(Serial.available())
        {
            break;
        }
        if(digitalRead(pulsador) == HIGH)
        {
            break;
        }
        analogWrite(salidaLedIzquierda,150);
        analogWrite(salidaLedDerecha,150);
        analogWrite(salidaLedCentral,150);
        delay(100);
    }
}
}

```

```
if(Serial.available())
{
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    break;
}
delay(100);
if(Serial.available())
{
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    break;
}
delay(100);
if(Serial.available())
{
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    break;
}
delay(100);
if(Serial.available())
{
    break;
}
if(digitalRead(pulsador) == HIGH)
```

```
{
    break;
}
delay(100);
if(Serial.available())
{
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    break;
}
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
analogWrite(salidaLedCentral,0);
delay(100);
if(Serial.available())
{
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    break;
}
delay(100);
if(Serial.available())
{
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    break;
```

```

    }
    delay(100);
    if(Serial.available())
    {
        break;
    }
    if(digitalRead(pulsador) == HIGH)
    {
        break;
    }
    delay(100);
    if(Serial.available())
    {
        break;
    }
    if(digitalRead(pulsador) == HIGH)
    {
        break;
    }
    delay(100);
}
}
if(readString == "EFECTO9" && registroModo == 1)
{
    while(0 == 0)
    {
        for(a=0;a <120;a++)
        {
            analogWrite(salidaLedIzquierda,a);
            analogWrite(salidaLedDerecha,a);
            analogWrite(salidaLedCentral,a);
            delay(45);
        }
    }
}

```



```

if(Serial.available())
{
    banderaSalidaCiclos = 1;
    break;
}
if(digitalRead(pulsador) == HIGH)
{
    banderaSalidaCiclos = 1;
    break;
}
}
if(banderaSalidaCiclos == 1)
{
    banderaSalidaCiclos = 0;
    break;
}
for(b=120;b >= 0;b--)
{
    analogWrite(salidaLedIzquierda,b);
    analogWrite(salidaLedDerecha,b);
    analogWrite(salidaLedCentral,b);
    delay(25);
    if(Serial.available())
    {
        banderaSalidaCiclos = 1;
        break;
    }
}
if(digitalRead(pulsador) == HIGH)
{
    banderaSalidaCiclos = 1;
    break;
}
}

```

```

    }
    if(banderaSalidaCiclos == 1)
    {
        banderaSalidaCiclos = 0;
        break;
    }
    for(c=0;c<=40;c++)
    {
        analogWrite(salidaLedIzquierda,0);
        analogWrite(salidaLedDerecha,0);
        analogWrite(salidaLedCentral,0);
        delay(50);
        if(Serial.available())
        {
            banderaSalidaCiclos = 1;
            break;
        }
        if(digitalRead(pulsador) == HIGH)
        {
            banderaSalidaCiclos = 1;
            break;
        }
    }
    if(banderaSalidaCiclos == 1)
    {
        banderaSalidaCiclos = 0;
        break;
    }
}
}
if(readString == "EFECTO10" && registroModo == 1)

```

```

{
  while(0 == 0)
  {
    for(a=20;a <220;a++)
    {
      analogWrite(salidaLedIzquierda,a);
      analogWrite(salidaLedDerecha,a);
      analogWrite(salidaLedCentral,a);
      delay(45);
      if(Serial.available())
      {
        banderaSalidaCiclos = 1;
        break;
      }
      if(digitalRead(pulsador) == HIGH)
      {
        banderaSalidaCiclos = 1;
        break;
      }
    }
    if(banderaSalidaCiclos == 1)
    {
      banderaSalidaCiclos = 0;
      break;
    }
    for(b=220;b >= 20;b--)
    {
      analogWrite(salidaLedIzquierda,b);
      analogWrite(salidaLedDerecha,b);
      analogWrite(salidaLedCentral,b);
      delay(25);
      if(Serial.available())

```

```

{
  banderaSalidaCiclos = 1;
  break;
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
}
if(flagSalidaCiclos == 1)
{
  banderaSalidaCiclos = 0;
  break;
}
for(c=0;c<=20;c++)
{
  analogWrite(salidaLedIzquierda,20);
  analogWrite(salidaLedDerecha,20);
  analogWrite(salidaLedCentral,20);
  delay(50);
  if(Serial.available())
  {
    banderaSalidaCiclos = 1;
    break;
  }
}
if(digitalRead(pulsador) == HIGH)
{
  banderaSalidaCiclos = 1;
  break;
}
}
}

```

```

    if(banderaSalidaCiclos == 1)
    {
        banderaSalidaCiclos = 0;
        break;
    }

}

}

readString=""; //Se elimina la ultima instrucción recibida desde el puerto serial.
}

//Da soporte al caso en el que se encendió la iluminación desde un dispositivo android y luego se requiere
apagarla de modo manual mediante el pulsador.

if(registroModo == 1 && digitalRead(pulsador) == HIGH && banderaEncendidoApagado == 1)
{
registroModo = 1;

banderaEncendidoApagado = 0;
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
}

//Realiza el cambio de modo 1 a modo 2 mediante la recepción de un bit alto del pulsador por el puerto
digital.

//En el "modo 1" la iluminación esta cuando controlada por un dispositivo Android o esta apagada.
//En el "modo 2" se enciende la iluminación derecha e izquierda y la iluminación central permanece
apagada.

if(registroModo == 1 && digitalRead(pulsador) == HIGH && banderaEncendidoApagado == 0)
{
registroModo = 2;
banderaEncendidoApagado = 1;

```

```

analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
analogWrite(salidaLedIzquierda,170);
analogWrite(salidaLedDerecha,170);
}

//Realiza el cambio de modo 2 a modo 3 mediante la recepción de un bit alto del pulsador por el puerto
digital.

//En el "modo 3" la iluminación central permanece encendida y las iluminaciones izquierda y derecha
permanecen apagadas.

if(registroModo == 2 && digitalRead(pulsador) == HIGH)
{
registroModo = 3;
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
delay(100);
}

```

```

    analogWrite(salidaLedCentral,150);
}

//Realiza el cambio de modo 3 a modo 4 mediante la recepción de un bit alto del pulsador por el puerto
digital.

//En el "modo 4" las tres iluminaciones encenderán respondiendo a las señales del sensor de presencia.
if(registroModo == 3 && digitalRead(pulsador) == HIGH)
{
registroModo = 4;
    analogWrite(salidaLedIzquierda,0);
    analogWrite(salidaLedDerecha,0);
    analogWrite(salidaLedCentral,0);
    delay(100);
    analogWrite(salidaLedCentral,150);
    delay(100);
    analogWrite(salidaLedCentral,0);
    delay(100);
    analogWrite(salidaLedCentral,150);
    delay(100);
    analogWrite(salidaLedCentral,0);
    delay(100);
    analogWrite(salidaLedCentral,150);
    delay(100);
    analogWrite(salidaLedCentral,0);
    delay(100);
    analogWrite(salidaLedCentral,150);
    delay(100);
    analogWrite(salidaLedCentral,0);

}

//Se realiza el cambio del modo 4 al modo 1.
if(registroModo == 4 && digitalRead(pulsador) == HIGH)
{

```

```

registroModo = 1;
banderaEncendidoApagado = 0;
analogWrite(salidaLedIzquierda,0);
analogWrite(salidaLedDerecha,0);
analogWrite(salidaLedCentral,0);
delay(100);
analogWrite(salidaLedCentral,150);
delay(100);
analogWrite(salidaLedCentral,0);
}
//Da soporte al modo 4
if(registroModo == 4)
{
//Se recibe un bit alto del sensor de presencia y se inicia el proceso de encendido automático por
movimiento.
if(digitalRead(sensorPresencia) == HIGH )
{
contSensorPir=120;//Se inicializa en 120 que son 120/8 de segundo que es equivalente a 15 segundos.
//Se realiza un encendido paulatino hasta la máxima intensidad.
for(intensidad=0;intensidad <256;intensidad++)
{
analogWrite(salidaLedIzquierda,intensidad);
analogWrite(salidaLedDerecha,intensidad);
analogWrite(salidaLedCentral,intensidad);
delay(5);
}
//Se realiza un ajuste paulatino hasta la última intensidad guardada.
for(intensidad=255;intensidad >= registroIntensidad;intensidad--)
{
analogWrite(salidaLedIzquierda,intensidad);
analogWrite(salidaLedDerecha,intensidad);
analogWrite(salidaLedCentral,intensidad);
}
}
}

```



```

delay(3);
}
intensidad=registroIntensidad;
analogWrite(salidaLedIzquierda, intensidad);
analogWrite(salidaLedDerecha,intensidad);
analogWrite(salidaLedCentral,intensidad);
//Proceso de conteo para en encendido por sensor de presencia = 15 segundo a partir del último bit alto
recibido del sensor pir.
while(contSensorPir >= 0)
{
    delay(125);//Este proceso se puede interrumpir en cualquier momento por ello cada 1/8 de segundo se
    verifica si existe una interrupcion pendiente por atender.
    contSensorPir = contSensorPir-1;
    //interrupcion por parte del mismo sensor pir que reinicia el contador ante movimientos consecutivos.
    if(digitalRead(sensorPresencia) == HIGH )
    {
        contSensorPir = 120;
    }
    if(Serial.available())
    {
        break;
    }
    if(digitalRead(pulsador) == HIGH)
    {
        registroModo = 1;
        banderaEncendidoApagado = 0;
        analogWrite(salidaLedIzquierda,0);
        analogWrite(salidaLedDerecha,0);
        analogWrite(salidaLedCentral,0);
        delay(100);
        analogWrite(salidaLedCentral,150);
        delay(100);
        analogWrite(salidaLedCentral,0);
    }
}

```

```

        break;
    }
}
//Se termina el proceso de conteo y se realiza un apagado paulatino.
if(registroModo == 4 && contSensorPir == -1)
{
for(i=intensidad; i >= 10;i--)
    {
    analogWrite(salidaLedIzquierda,i);
    analogWrite(salidaLedDerecha,i);
    analogWrite(salidaLedCentral,i);
    delay(10);
    }
for(i=10; i >= 0;i--)
    {
    analogWrite(salidaLedIzquierda,i);
    analogWrite(salidaLedDerecha,i);
    analogWrite(salidaLedCentral,i);
    delay(15);
    }
}
}
}

```

## 16.2 B

The image displays three sections of Scratch code blocks:

- Top Section:** A 'when Screen1 BackPressed' event triggers a 'do' loop. Inside, it checks if 'btnEfecto1' is visible; if so, it calls 'cambiarPantallaAtras1'. If 'btnModoAndroid' is visible, it calls 'cambiarModoAndroid'. Otherwise, it closes the application. Additionally, a 'when btnSalir Click' event also closes the application, and a 'when lpkSeleccionIluminacion BeforePicking' event sets 'lpkSeleccionIluminacion Elements' to 'BluetoothClient1 Address'.
- Middle Section:** A 'when spnProgramacion AfterSelecting' event triggers a 'selection' block. It then checks if the selection is 'Cancelar programación'. If yes, it sends '0' via Bluetooth, enables 'btnEncender', sets 'btnBloquearIntensidad' text to 'DESBLOQUEAR INTENSIDAD' and background color to red, and shows a dialog: 'Se ha cancelado la programación de apagado de la iluminación.' with title 'Atención' and button 'Entendido'. It then checks if the selection is 'Cancelar programación' again and sets it to 'Ninguna'. It then checks for '5 min', sending '5' via Bluetooth and showing a dialog: 'Programación enviada, la iluminación se apagará en 5 minutos.' with title 'Atención' and button 'Entendido'.
- Bottom Section:** It checks for '15 min', sending '15' via Bluetooth and showing a dialog: 'Programación enviada, la iluminación se apagará en 15 minutos.' with title 'Atención' and button 'Entendido', then calls 'cambiarPantalla2'. It then checks for '30 min', sending '30' via Bluetooth and showing a dialog: 'Programación enviada, la iluminación se apagará en 30 minutos.' with title 'Atención'.

```



    buttonText " Entendido "
    call cambiarPantalla2
  if
    get selection = " 1 hrs "
  then
    call BluetoothClient1 .SendText
      text " 1H "
    call Notifier1 .ShowMessageDialog
      message " Programación enviada, la iluminación se apagará en 1 hora. "
      title " Atención "
      buttonText " Entendido "
    call cambiarPantalla2
  if
    get selection = " 2 hrs "
  then
    call BluetoothClient1 .SendText
      text " 2H "
    call Notifier1 .ShowMessageDialog
      message " Programación enviada, la iluminación se apagará en 2 horas. "
      title " Atención "
      buttonText " Entendido "
    call cambiarPantalla2
  if
    get selection = " 30 seg "
  then
    call BluetoothClient1 .SendText
      text " 30S "
    call Notifier1 .ShowMessageDialog
      message " Programación enviada, la iluminación se apagará en 30 segundos. "
      title " Atención "
      buttonText " Entendido "
    call cambiarPantalla2

```

```

to cambiarPantallaModos
do
  set lblEncenderApagar .Visible to false
  set btnSimulacion .Visible to false
  set lblSimulacion .Visible to false
  set lblUltimaProgramacion .Visible to false
  set btnEncender .Visible to false
  set lblProgramación .Visible to false
  set lblIntensidad .Visible to false
  set lblRotacionAjustar .Visible to false
  set btnBloquearIntensidad .Visible to false
  set lblProgramación .Visible to false
  set spnProgramacion .Visible to false
  set btnPaletaEfectos .Visible to false
  set btnModos .Visible to false
  set btnModoAndroid .Visible to true
  set btnModoSensor .Visible to true

```

 0 
  0  
 Show Warnings







```

set btnModoTipo1 . Visible to true
set btnModoTipo2 . Visible to true


to cambiarModoAndroid
do
set lblEncenderApagar . Visible to true
set btnSimulacion . Visible to true
set lblSimulacion . Visible to true
set lblUltimaProgramacion . Visible to true
set btnEncender . Visible to true
set lblProgramacion . Visible to true
set lblIntensidad . Visible to true
set lblRotacionAjustar . Visible to true
set btnBloquearIntensidad . Visible to true
set btnPaletaEfectos . Visible to true
set lblProgramacion . Visible to true
set spnProgramacion . Visible to true
set btnModos . Visible to true
set btnModoAndroid . Visible to false
set btnModoSensor . Visible to false
set btnModoTipo1 . Visible to false
set btnModoTipo2 . Visible to false

to cambiarPantallaAtras1
do
set btnSimulacion . Visible to true
set lblSimulacion . Visible to true
set lblEncenderApagar . Visible to true
set lblUltimaProgramacion . Visible to true
set btnEncender . Visible to true
set lblProgramacion . Visible to true
set lblIntensidad . Visible to true
set lblRotacionAjustar . Visible to true
set btnBloquearIntensidad . Visible to true
set btnPaletaEfectos . Visible to true
set lblProgramacion . Visible to true
set spnProgramacion . Visible to true
set btnModos . Visible to true
set btnModoAndroid . Visible to false
set btnModoSensor . Visible to false
set btnModoTipo1 . Visible to false
set btnModoTipo2 . Visible to false
set btnAtras . Visible to false
set btnEfecto1 . Visible to false
set btnEfecto2 . Visible to false
set btnEfecto3 . Visible to false
set btnEfecto4 . Visible to false
set btnEfecto5 . Visible to false
set btnEfecto6 . Visible to false
set btnEfecto7 . Visible to false
set btnEfecto8 . Visible to false


```


0 0  
Show Warnings



0 0  
Show Warnings



0 0  
Show Warnings



```

set btnEfecto9 . Visible to false
set btnEfecto10 . Visible to false

to cambiarPantallaEfectos
do
set lblEncenderApagar . Visible to false
set lblSimulacion . Visible to false
set btnSimulacion . Visible to false
set lblUltimaProgramacion . Visible to false
set btnEncender . Visible to false
set lblProgramación . Visible to false
set lblIntensidad . Visible to false
set lblRotacionAjustar . Visible to false
set btnBloquearIntensidad . Visible to false
set lblProgramación . Visible to false
set spnProgramacion . Visible to false
set btnPaletaEfectos . Visible to false

set btnModos . Visible to false
set btnEfecto1 . Visible to true
set btnEfecto2 . Visible to true
set btnEfecto3 . Visible to true
set btnEfecto4 . Visible to true
set btnEfecto5 . Visible to true
set btnEfecto6 . Visible to true
set btnEfecto7 . Visible to true
set btnEfecto8 . Visible to true
set btnEfecto9 . Visible to true
set btnEfecto10 . Visible to true
set btnAtras . Visible to true

```

0 0  
Show Warnings

```

when btnAtras . Click
do call cambiarPantallaAtras1

```

```

when btnPaletaEfectos . Click
do call cambiarPantallaEfectos

```

```

to cambiarPantallaOff
do
set lblSimulacion . Visible to false
set btnSimulacion . Visible to false
set spnProgramacion . Visible to false
set btnBloquearIntensidad . Visible to false
set lblIntensidad . Visible to false
set lblRotacionAjustar . Visible to false
set lblUltimaProgramacion . Visible to false
set lblProgramación . Visible to false
set btnModos . Visible to false
set btnPaletaEfectos . Visible to false

```

```
to cambiarPantallaOn
do
  set spnProgramacion . Visible to true
  set btnModos . Visible to true
  set btnSimulacion . Visible to true
  set btnPaletaEfectos . Visible to true
  set btnBloquearIntensidad . Visible to true
  set lblIntensidad . Visible to true
  set lblRotacionAjustar . Visible to true
  set lblSimulacion . Visible to true
  set lblUltimaProgramacion . Visible to true
  set lblProgramación . Visible to true
```

```
to cambiarPantalla2
do
  set btnSimulacion . Visible to false
  set lblSimulacion . Visible to false
  set btnSalir . Visible to true
  set lblDesactivarProgramacion . Visible to true
  set lblEncenderApagar . Visible to false
  set lblAuxiliar . Visible to false
  set btnModos . Visible to false
  set btnPaletaEfectos . Visible to false
  set btnEncender . Visible to false
  set lblIntensidad . Visible to false
  set lblRotacionAjustar . Visible to false
  set btnBloquearIntensidad . Visible to false
  set lblProgramación . Visible to false
  set spnProgramacion . Visible to false
  set lblUltimaProgramacion . Visible to false
```

```
to cambiarPantalla1
do
  set btnSimulacion . Visible to false
  set lblSimulacion . Visible to false
  set btnSalir . Visible to true
  set lblSimulacionEstancia . Visible to true
  set lblEncenderApagar . Visible to false
  set lblAuxiliar . Visible to false
  set btnModos . Visible to false
  set btnPaletaEfectos . Visible to false
  set btnEncender . Visible to false
  set lblIntensidad . Visible to false
  set lblRotacionAjustar . Visible to false
  set btnBloquearIntensidad . Visible to false
  set lblProgramación . Visible to false
  set spnProgramacion . Visible to false
  set lblUltimaProgramacion . Visible to false
```

```
when btnEfecto1 . Click
do
  call BluetoothClient1 . SendText
  text " EFECTO1 "
```

```
when btnEfecto2 . Click
do
  call BluetoothClient1 . SendText
  text " EFECTO2 "
```

```
when btnEfecto3 . Click
do
  call BluetoothClient1 . SendText
  text " EFECTO3 "
```

```
when btnEfecto4 . Click
do
  call BluetoothClient1 . SendText
  text " EFECTO4 "
```





```

when IpSeleccionIluminacion . AfterPicking
do
  set IpSeleccionIluminacion . Selection to call BluetoothClient1 . Connect
  address IpSeleccionIluminacion . Selection
  set lblMensaje . Text to BluetoothClient1 . IsConnected
  if BluetoothClient1 . IsConnected
  then
    call BluetoothClient1 . SendText
    text " CONECTED "
    set lblEncenderApagar . Visible to true
    set btnEncender . Visible to true
    set IpSeleccionIluminacion . Visible to false
  set lblMensaje . Text to if BluetoothClient1 . IsConnected
  then " Conectado con la iluminación..."
  else " No se logró conectar con la iluminación, vuelva a intentarlo."

```

0 0

```

when btnEncender . Click
do
  set lblEncenderApagar . Text to if " Click para apagar " = lblEncenderApagar . Text
  then " Click para encender "
  else " Click para apagar "
  set btnEncender . Image to if " Click para apagar " = lblEncenderApagar . Text
  then " ENCENDIDO.png "
  else " APAGADO.png "
  if " Click para apagar " = lblEncenderApagar . Text
  then
    call BluetoothClient1 . SendText
    text " ON "
    set btnBloquearIntensidad . Text to " DESBLOQUEAR INTENSIDAD "
    set btnBloquearIntensidad . BackgroundColor to red
    call cambiarPantallaOn
  if " Click para encender " = lblEncenderApagar . Text
  then
    call BluetoothClient1 . SendText
    text " OFF "
    call cambiarPantallaOff

```

0  
Show

```

when AccelerometerSensor1 . AccelerationChanged
xAccel yAccel zAccel
do
  if get xAccel > 5 and btnBloquearIntensidad . Text = " BLOQUEAR INTENSIDAD "
  then
    set lblRotacionAjustar . Text to " Subiendo..."
    call BluetoothClient1 . SendText
    text " U "
  if get xAccel < -5 and btnBloquearIntensidad . Text = " BLOQUEAR INTENSIDAD "
  then
    set lblRotacionAjustar . Text to " Bajando..."
    call BluetoothClient1 . SendText
    text " D "
  if get xAccel > -5 and get xAccel < 5
  then
    set lblRotacionAjustar . Text to " Rotar para ajustar "

```

```

when btnBloquearIntensidad .Click
do
  set btnBloquearIntensidad .Text to
  if btnBloquearIntensidad .Text = "DESBLOQUEAR INTENSIDAD"
  then "BLOQUEAR INTENSIDAD"
  else "DESBLOQUEAR INTENSIDAD"
  if btnBloquearIntensidad .Text = "DESBLOQUEAR INTENSIDAD"
  then set btnBloquearIntensidad .BackgroundColor to red
  if btnBloquearIntensidad .Text = "BLOQUEAR INTENSIDAD"
  then
    set btnEncender .Enabled to true
    set btnBloquearIntensidad .BackgroundColor to gray

when btnModos .Click
do
  call cambiarPantallaModos

when btnModoAndroid .Click
do
  call cambiarModoAndroid

```

```

when btnModoSensor .Click
do
  set btnModos .Visible to false
  set btnModoAndroid .Visible to false
  set btnModoSensor .Visible to false
  set btnModoTipo1 .Visible to false
  set btnModoTipo2 .Visible to false
  call BluetoothClient1 .SendText
  text "SET2"
  call Notifier1 .ShowMessageDialog
  message "Se ha activado el modo de iluminación 2."
  title "Atención"
  buttonText "Entendido"
  set btnSalir .Visible to true
  set lblMensaje1 .Visible to true

```

```

when btnModoTipo2 .Click
do
  set btnModos .Visible to false
  set btnModoAndroid .Visible to false
  set btnModoSensor .Visible to false
  set btnModoTipo1 .Visible to false
  set btnModoTipo2 .Visible to false
  call BluetoothClient1 .SendText
  text "SET4"
  call Notifier1 .ShowMessageDialog
  message "Se ha activado el modo de iluminación 4. Sensor de movimiento."
  title "Atención"
  buttonText "Entendido"
  set btnSalir .Visible to true
  set lblMensajeSensor .Visible to true

```

```
when btnModoTipo1 .Click
do
  set btnModos . Visible to false
  set btnModoAndroid . Visible to false
  set btnModoSensor . Visible to false
  set btnModoTipo1 . Visible to false
  set btnModoTipo2 . Visible to false
  call BluetoothClient1 .SendText
    text "SET3 "
  call Notifier1 .ShowMessageDialog
    message "Se ha activado el modo de iluminación 3. "
    title "Atención "
    buttonText "Entendido "
  set btnSalir . Visible to true
  set lblMensaje2 . Visible to true
```