🏠 Sistema de Iluminación Inteligente

Documento de Visión del Proyecto

Proyecto Integrador UCQ — Sistemas Embebidos

Versión 1.0 | Febrero 2026

1. Resumen General del Proyecto

1.1 Problema de Negocio

El consumo ineficiente de energía en espacios interiores y exteriores representa un problema crítico tanto económico como medioambiental. Los sistemas de iluminación convencionales permanecen encendidos independientemente de la presencia de personas, generando desperdicio energético significativo, aumento en costos operativos y mayor huella de carbono en instituciones educativas, empresas y espacios públicos.

1.2 Declaración del Problema

El problema de:

- La iluminación estática e ineficiente en espacios de uso variable (aulas, pasillos, oficinas).
- La falta de automatización basada en la presencia real de usuarios.
- El alto consumo eléctrico generado por sistemas de iluminación no inteligentes.

Afecta a:

- Administradores institucionales que buscan reducir costos operativos.
- Usuarios finales (estudiantes, empleados) que habitan los espacios iluminados.
- El medioambiente, por el desperdicio energético acumulado.

El impacto es:

- Costos energéticos elevados sin justificación de uso real.
- Falta de control y monitoreo sobre el consumo de iluminación.
- Procesos manuales dependientes de usuarios para encender/apagar sistemas.

Una solución exitosa sería:

- Un sistema embebido que detecte movimiento y controle automáticamente la iluminación.
- Reducción estimada del 40-60% en consumo energético de iluminación.
- Operación autónoma sin intervención manual del usuario.

1.3 Posicionamiento del Proyecto

Elemento	Descripción
Para	Instituciones, empresas y espacios que buscan eficiencia energética
Quienes	Requieren automatización de iluminación sin intervención manual
El producto	Sistema de Iluminación Inteligente con detección de movimiento
Que	Controla automáticamente la iluminación basado en presencia de personas
A diferencia de	Sistemas manuales o temporizadores simples sin sensores de presencia
Nuestro producto	Integra hardware embebido (Arduino/ESP32) con lógica de control inteligente y monitoreo en tiempo real

1.4 Posicionamiento Organizacional

Este proyecto es desarrollado por el equipo de sistemas embebidos de la UCQ (Universidad de la Ciudad de Querétaro) como proyecto integrador de la materia de Sistemas Embebidos. El equipo busca demostrar competencias en diseño de firmware, integración de hardware y metodologías ágiles de desarrollo, alineando el proyecto con la misión institucional de formar ingenieros capaces de resolver problemas reales con tecnología accesible.

2. Descripción de Usuarios

2.1 Perfiles de Usuario

Tipo de Usuario	Perfil Técnico	Responsabilidades Clave	Necesidades
Administrador del Sistema	Técnico con conocimientos básicos de IoT	Configurar umbrales de detección, revisar logs, mantener el hardware	Interfaz de configuración simple, reportes de consumo
Usuario Final (Ocupante)	Sin conocimiento técnico	Habitar el espacio iluminado	Iluminación automática sin intervención, confort visual
Ingeniero de Mantenimiento	Conocimiento de electrónica básica	Revisar hardware, reemplazar componentes, calibrar sensores	Diagramas claros, acceso a pines y configuraciones
Desarrollador del Proyecto	Experto en sistemas embebidos y firmware	Programar, integrar, documentar y probar el sistema	Ambiente de desarrollo documentado, acceso a repositorio

2.2 Entorno del Usuario

- El sistema opera en espacios cerrados: aulas, pasillos, oficinas y áreas de acceso controlado.
- El hardware se instala en puntos fijos; el usuario interactúa con el espacio físico, no con la interfaz directamente.
- Plataforma actual: Arduino/ESP32 con sensores PIR y módulos de control de carga.
- Conexión a red local opcional para monitoreo remoto vía WiFi (ESP32).
- Operación 24/7 con ciclos de activación breves (segundos a minutos por detección).

2.3 Necesidades Actuales del Usuario

- Automatización total: el sistema debe encenderse y apagarse sin intervención humana.
- Tiempo de respuesta rápido (<1 segundo) desde la detección de movimiento hasta la activación de luz.
- Temporización configurable: el tiempo que permanece encendida la luz tras la última detección debe ser ajustable.
- Consumo energético mínimo en estado de espera (modo bajo consumo cuando no hay movimiento).
- Confiabilidad: el sistema debe operar de forma continua sin fallos por al menos 30 días.

2.4 Soluciones Alternativas

Alternativa 1 — Sensores comerciales independientes (ej. Philips Hue Motion): Fáciles de instalar pero costosos, propietarios y sin capacidad de personalización del firmware.

Alternativa 2 — Temporizadores manuales: Económicos pero no detectan presencia real, lo que puede dejar luces encendidas en espacios vacíos o apagarlas con personas presentes.

Solución elegida: Sistema embebido propio con ESP32/Arduino, que permite personalización total, bajo costo y integración con sistemas de monitoreo.

3. Descripción General del Sistema

3.1 Perspectiva del Sistema

El Sistema de Iluminación Inteligente es un sistema embebido autónomo que integra sensores de movimiento PIR, microcontroladores (Arduino Uno / ESP32) y cargas de iluminación LED. Opera de forma independiente con capacidad de expansión para monitoreo remoto. Se comunica con el hardware de iluminación mediante señales digitales PWM o relés, y opcionalmente reporta eventos vía WiFi a un servidor local o dashboard web.

3.2 Resumen de Capacidades

Función del Sistema	Beneficio Clave
Detección de movimiento mediante sensor PIR	Activación automática de iluminación sin intervención manual
Control de tiempo de encendido configurable	Evita que las luces permanezcan encendidas sin necesidad
Modo bajo consumo en standby	Reduce consumo energético cuando no hay presencia detectada
Indicador visual de estado del sistema	Facilita diagnóstico sin necesidad de herramientas externas
Monitoreo remoto vía WiFi (ESP32)	Permite supervisión del estado del sistema desde una interfaz web
Log de eventos de detección	Posibilita análisis de patrones de uso del espacio

3.3 Supuestos y Dependencias

- Se asume disponibilidad de suministro eléctrico estable en el área de instalación.
- Los sensores PIR compatibles (HC-SR501 o similar) estarán disponibles para el equipo de desarrollo.
- El entorno de desarrollo incluye Arduino IDE o PlatformIO con soporte para ESP32.
- La red WiFi local estará disponible para las funciones de monitoreo remoto (opcional).
- El hardware será proporcionado por la UCQ o adquirido por el equipo.

4. Características del Sistema

ID	Característica	Descripción	Prioridad
F-01	Detección de movimiento	El sistema detecta presencia mediante sensor PIR con rango de 3-7 metros y ángulo de 110°	Alta
F-02	Control automático de iluminación	Enciende y apaga la iluminación según detección de movimiento	Alta
F-03	Temporización ajustable	El tiempo de encendido post-detección es configurable entre 10 segundos y 10 minutos	Alta
F-04	Modo bajo consumo	El microcontrolador entra en modo sleep cuando no hay actividad	Media
F-05	Indicador LED de estado	LED de diagnóstico muestra el estado del sistema (activo, espera, error)	Media
F-06	Monitoreo WiFi remoto	ESP32 publica eventos de detección vía MQTT o servidor HTTP local	Baja
F-07	Registro de eventos	Almacenamiento local de timestamps de detección para análisis posterior	Baja
F-08	Calibración de sensibilidad	Ajuste de sensibilidad del sensor PIR mediante potenciómetro o firmware	Media

5. Requisitos Adicionales

5.1 Restricciones

- El sistema debe utilizar únicamente pines digitales GPIO del microcontrolador (restricción del proyecto académico).
- El código debe seguir el coding standard definido por el equipo (documentado en /docs).
- El diseño de hardware debe ser reproducible con componentes disponibles en el mercado local de Querétaro.
- El costo total de componentes no debe exceder $500 MXN por unidad.

5.2 Rangos de Calidad

- Tiempo de respuesta: < 1 segundo desde detección hasta activación de luz.
- Confiabilidad: Operación continua por 30 días sin reinicio manual.
- Consumo en standby: < 50mA en modo bajo consumo.
- Cobertura de detección: mínimo 3 metros de rango efectivo.
- Precisión de detección: < 5% de falsos positivos en condiciones normales.

5.3 Estándares Aplicables

- Lenguaje de programación: C/C++ para Arduino/ESP32 con estilo de código modular.
- Control de versiones: Git con commits convencionales (feat, fix, chore, docs).
- Documentación: Markdown en repositorio GitHub + comentarios Doxygen en código.
- Metodología: SCRUM con sprints de 1 semana.

5.4 Requisitos del Sistema

- Microcontrolador: Arduino Uno (prototipo) / ESP32 (versión con WiFi).
- Sensor: PIR HC-SR501 o equivalente.
- Carga: LED de 5V o relé para cargas de 110V AC.
- Alimentación: 5V DC (USB o fuente regulada).
- Entorno de desarrollo: Arduino IDE 2.x / PlatformIO + VS Code.

5.5 Requisitos de Rendimiento

- El sistema debe detectar movimiento y activar la iluminación en menos de 1 segundo.
- El microcontrolador debe procesar las interrupciones del sensor en tiempo real sin pérdida de eventos.
- En modo monitoreo WiFi, la latencia de reporte no debe exceder 2 segundos.

5.6 Requisitos de Documentación

- README.md con instrucciones de instalación y uso.
- Documento SRS (System Requirements Specification) en /docs.
- Diagramas de hardware (esquemáticos) en /hardware.
- Código fuente comentado con Doxygen en /src.
- Resultados de pruebas documentados en /tests.

6. Definition of Ready (DoR) y Definition of Done (DoD)

6.1 Definition of Ready (DoR)

Una historia de usuario está lista para ser trabajada en un sprint cuando cumple todos los siguientes criterios:

Criterio	Descripción
Descripción clara	La historia tiene título, contexto y objetivo comprensibles para todo el equipo
Criterios de aceptación	Se definen condiciones verificables y medibles para considerar la historia completa
Dependencias identificadas	Se listan otros módulos, hardware o historias que deben existir previamente
Estimación en Story Points	El equipo ha estimado el esfuerzo usando la escala de Fibonacci (1, 2, 3, 5, 8, 13)
Hardware disponible	Los componentes físicos necesarios están disponibles o confirmados para el sprint
Sin bloqueos conocidos	No existen impedimentos técnicos u organizacionales sin resolver al inicio del sprint

6.2 Definition of Done (DoD)

Una historia de usuario está terminada cuando cumple todos los siguientes criterios:

Criterio	Descripción
Compila sin warnings	El código compila exitosamente sin advertencias en Arduino IDE / PlatformIO
Coding standard	El código sigue las convenciones de nomenclatura y estructura definidas por el equipo
Pruebas realizadas	Se han ejecutado pruebas funcionales documentadas en /tests con resultados registrados
Integrado en main	El código ha sido mergeado a la rama main mediante Pull Request aprobado
Documentado	El código tiene comentarios Doxygen y el README refleja los cambios realizados
Demo funcional	Se ha demostrado el funcionamiento ante al menos un miembro del equipo o el Scrum Master

7. Planeación de Sprints

El proyecto se desarrolla en 6 sprints de 1 semana cada uno, siguiendo metodología SCRUM.

Sprint	Objetivo	Entregables Principales
Sprint 0Semana 1	Definición del producto	Project Vision, Problem Statement, Stakeholders, Backlog inicial, Arquitectura propuesta, Selección HW/SW, DoR y DoD
Sprint 1Semana 2	Setup y prototipo base	Entorno de desarrollo configurado, primer programa de detección PIR funcional, esquemático de hardware v1
Sprint 2Semana 3	Control de iluminación	Lógica de encendido/apagado con temporización configurable, pruebas de detección documentadas
Sprint 3Semana 4	Optimización y bajo consumo	Modo sleep implementado, calibración de sensibilidad, pruebas de confiabilidad (24h continuas)
Sprint 4Semana 5	Monitoreo y comunicación	Módulo WiFi (ESP32) funcional, reporte de eventos, dashboard básico o log serial
Sprint 5Semana 6	Integración y entrega final	Sistema integrado completo, documentación final, demo funcional, presentación al profesor

8. Product Backlog Inicial

Formato: ID único | User Story | Prioridad (Alta/Media/Baja) | Story Points (Fibonacci)

ID	User Story	Sprint	Prioridad	Story Points
US-001	Como administrador, quiero que el sistema encienda la luz automáticamente al detectar movimiento, para no depender de interruptores manuales.	1	Alta	5
US-002	Como usuario, quiero que la luz permanezca encendida durante un tiempo configurable tras la última detección, para no quedarme sin luz mientras estoy en el espacio.	1	Alta	3
US-003	Como administrador, quiero configurar el tiempo de encendido entre 10s y 10min, para adaptar el sistema a diferentes espacios.	2	Alta	3
US-004	Como ingeniero, quiero que el sistema entre en modo bajo consumo cuando no hay movimiento, para reducir el gasto energético.	3	Media	5
US-005	Como técnico, quiero un LED indicador de estado del sistema, para diagnosticar fallos sin herramientas externas.	2	Media	2
US-006	Como administrador, quiero recibir reportes de eventos de detección vía WiFi, para monitorear el uso del espacio remotamente.	4	Baja	8
US-007	Como desarrollador, quiero que el código compile sin warnings y tenga pruebas documentadas, para garantizar calidad del firmware.	1	Alta	3
US-008	Como administrador, quiero un diagrama de hardware claro y esquemático en el repositorio, para replicar la instalación.	1	Alta	2
US-009	Como usuario, quiero que la detección sea precisa (< 5% falsos positivos), para evitar encendidos innecesarios.	3	Media	5

ID	User Story	Sprint	Prioridad	Story Points
US-010	Como equipo, quiero el sistema completamente integrado y documentado, para presentar un demo funcional al profesor.	5	Alta	8

9. Matriz de Roles por Participante

Integrante	GitHub / Rol Principal	Product Owner	Scrum Master	Firmware	Hardware	Integración	Testing & QA
Luis Antonio	Project Manager	✓	—	✓	—	✓	—
Uriel Everardo	Hardware Lead	—	✓	—	✓	✓	—
Luis Alejandro	Software Lead	—	—	✓	—	✓	✓
Juan Luis	Testing & QA	—	—	✓	✓	—	✓

Nota: Al ser un equipo de 4 personas, cada integrante asume múltiples roles según las necesidades del sprint. El Product Owner define y prioriza el backlog; el Scrum Master facilita las ceremonias y elimina bloqueos; el Development Team es responsable de la implementación técnica en sus áreas de especialización.

9.1 Responsabilidades por Rol

Rol	Responsable	Descripción de Responsabilidades
Product Owner	Luis Antonio	Define y prioriza el Product Backlog. Representa los intereses del cliente/profesor. Valida que los entregables cumplan los criterios de aceptación. Gestiona el alcance del proyecto.
Scrum Master	Uriel Everardo	Facilita las ceremonias Scrum (Sprint Planning, Daily, Review, Retrospectiva). Elimina impedimentos y protege al equipo durante el sprint. Cuida el cumplimiento de la metodología ágil.
Dev Team — Firmware / Software Lead	Luis Alejandro	Desarrollo del código fuente en C/C++ para Arduino/ESP32. Implementación de lógica de detección, control de carga y comunicación WiFi.
Dev Team — Hardware Lead	Uriel Everardo	Diseño y ensamblaje de circuitos. Elaboración de esquemáticos y selección de componentes electrónicos. Documentación de hardware en /hardware.
Dev Team — Integración	Luis Antonio, Luis Alejandro	Integración de módulos firmware y hardware. Gestión del repositorio GitHub, process de PR/merge y mantenimiento de rama main.

Rol	Responsable	Descripción de Responsabilidades
Dev Team — Testing & QA	Juan Luis	Diseño y ejecución de casos de prueba. Documentación de resultados en /tests. Validación de criterios de aceptación (DoD). Reporte de bugs.

— Fin del Documento de Visión del Proyecto —

Repositorio: https://github.com/RedShock2/smart-lighting-system

UCQ — Sistemas Embebidos — Febrero 2026