



Universidad de Extremadura

Escuela Politécnica de Cáceres

Departamento de Ingeniería Eléctrica Electrónica y Automática

SISTEMA DE APERTURA DE GARAJE CON ESP32

LAURA SÁNCHEZ DOSDADO

Tercer curso del Grado en Ingeniería de Telecomunicación en Sonido e Imagen

Antonio Gordillo Guerrero

November 7, 2024

Contents

1	Motivación	1
2	Estado del Arte	1
3	Resultados Obtenidos	2
4	Esquemas, gráficos, ficheros o códigos	3
5	Materiales y Costes	6
6	Futuros Desarrollos Posibles	7
7	Conclusión	7

1 Motivación

Para este proyecto, la motivación principal surgió del deseo de probarme a mi misma en programación y electrónica, áreas en las que, sinceramente, no siempre me he sentido del todo segura. En la asignatura de “Fundamentos de Programación” adquirimos conocimientos de C++ y desarrollamos un juego; sin embargo, hasta este proyecto no había trabajado con dispositivos físicos, y me pareció interesante enfrentarme a ese reto.

A lo largo de la carrera, también hemos estudiado circuitos en teoría, pero sin la experiencia de construir uno realmente. Por ello, me ilusionaba la idea de crear algo práctico, aprender de cada etapa y enfrentar las dificultades técnicas de un proyecto que, a diferencia de lo habitual en el grado, demanda tanto habilidades de programación como de montaje. Además, considero que estas oportunidades prácticas son valiosas para adquirir confianza y, sobre todo, de disfrutar el proceso.

2 Estado del Arte

Se revisaron las siguientes fuentes y recursos durante la creación del proyecto:

- **Proyecto de inspiración:** [Video en YouTube](#) (consultado el 28/10/2024).
- **Control del servomotor:**
 - [Ejemplo de control de servo con ESP32](#) en GitHub.
 - [Tutorial de servomotor en Arduino](#) en programarfacil.com.
- **Sensor LDR:** [Conexión de una fotorresistencia LDR en Arduino](#) en 330ohms.com.
- **ESP32 en formato Fritzing:** [Repositorio en GitHub](#) (consultado el 02/11/2024).
- **Introducción a LaTeX:** [Guía en Overleaf](#) (consultado el 02/11/2024).
- **Programación de un pulsador en Arduino:** [Tutorial en GeekFactory](#).

3 Resultados Obtenidos

Este proyecto consiste en una puerta de garaje automatizada, controlada mediante una ESP-WROOM-32 y una aplicación móvil desarrollada en MIT App Inventor. La comunicación entre ambos dispositivos se realiza por Bluetooth, permitiendo abrir y cerrar la puerta de manera remota desde la aplicación, la cual muestra indicadores visuales mediante los LEDs para informar sobre el estado actual de la puerta.

Además, implementé un sensor LDR que simula la detección de vehículos. Si la puerta está abierta y un vehículo está pasando, el sensor detecta la obstrucción de la luz y evita el cierre de la puerta, proporcionando así una medida de seguridad adicional. Asimismo, incorporé un pulsador para permitir el control manual en caso de fallos en la conexión Bluetooth. A continuación voy a explicar las distintas **Fases del proyecto**:

- **Fase 1 (17-23 Octubre) — Preparación y Comunicación Bluetooth**

Comencé definiendo el alcance del proyecto y reuniendo todos los componentes necesarios. Para verificar que la ESP32 funcionaba correctamente, implementé un sencillo programa de "Blink". Seguidamente, diseñé un código básico para controlar el movimiento del servo y permití que respondiera a comandos de apertura y cierre. Finalmente, creé una aplicación en MIT App Inventor que enviara estos comandos a la ESP32 mediante Bluetooth.

Para la instalación es importante utilizar un teléfono móvil con Android, ya que MIT App Inventor solo exporta en formato APK (archivo ejecutable que contiene todos los datos que se necesitan para instalar y hacer funcionar una aplicación Android).

- **Fase 2 (23-29 Octubre) — Lógica de la Puerta y Estados**

En esta fase, trabajé en la lógica de control de la puerta y definí diferentes estados de funcionamiento. Configuré la conexión Bluetooth para que los comandos enviados desde la aplicación controlaran el servo de la puerta. Integré los LEDs, de manera que reflejaran el estado actual de la puerta y del sistema en general: LED verde para indicar que la puerta está abierta y LED rojo para indicar que está cerrada o en proceso de apertura/cierre.

- **Fase 3 (29 Octubre - 1 Noviembre) — Máquina de Estados y Pruebas**

Aquí diseñé una máquina de estados para gestionar los distintos modos de la puerta: abierta, cerrada y bloqueada (cuando un vehículo bloquea la luz del sensor). Añadí una lógica adicional para hacer parpadear el LED rojo en condiciones de baja luz. Optimicé el código para reducir el consumo de energía ajustando el uso del servo de forma que ya no tuve que usar ni la fuente, ni el regulador de voltaje, y realicé pruebas completas para asegurar la coordinación entre todos los componentes: servo, fotorresistencia, pulsador, LEDs y Bluetooth.

- **Fase 4 (1-7 Noviembre) — Documentación y Presentación**

En esta última fase, preparé la documentación técnica y el informe del proyecto en L^AT_EX, incluyendo código detallado, diagramas, y otros aspectos técnicos relevantes. Además, realicé el esquema de conexión de los componentes con Fritzing y subí todo el contenido del proyecto a mi GitHub, el enlace a este lo dejaré más adelante.

4 Esquemas, gráficos, ficheros o códigos

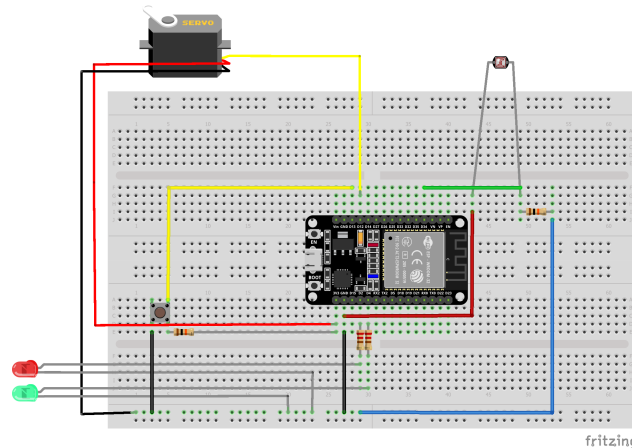


Figure 1: Esquema de conexionado del sistema

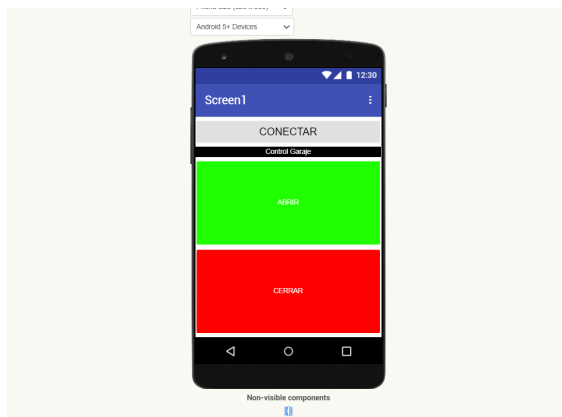


Figure 2: Interfaz de la aplicación

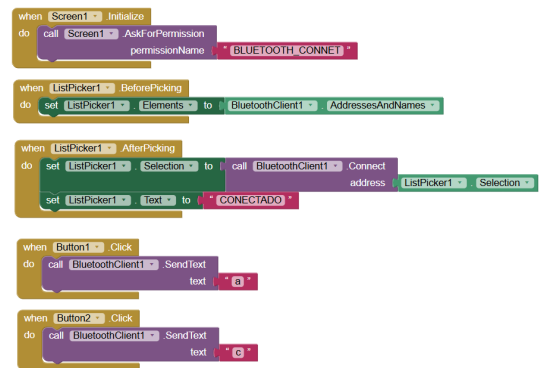


Figure 3: Bloques de la aplicación

Códigos implementados

Este es el enlace a GitHub donde se pueden ver todos los códigos que implementé durante el desarrollo del proyecto hasta llegar al código final, explicado a continuación:

https://github.com/LSD-XM/Garage_Door_System_ESP32

Código Final

El código final está disponible en GitHub en el siguiente enlace:

https://github.com/LSD-XM/Garage_Door_System_ESP32/blob/main/C%C3%B3digoArduinoIDE/

PuertaGarajeFINAL/PuertaGarajeFINAL.ino

Bibliotecas y Configuración Bluetooth

Para el control del servo y la comunicación Bluetooth, se utilizan las bibliotecas `ESP32Servo` y `BluetoothSerial`:

```
1 #include <ESP32Servo.h>
2 #include <BluetoothSerial.h>
```

La configuración de Bluetooth permite el control remoto desde una aplicación o dispositivo que envíe comandos a través de este protocolo. El dispositivo se inicializa con el nombre `ESP32_Puerta`.

Lógica de Control

La lógica de control de la puerta sigue una **máquina de estados** para gestionar el proceso de apertura, cierre y bloqueo de la puerta, basándose en el estado actual de la puerta y las condiciones de luz detectadas.

La **máquina de estados** incluye tres estados clave:

- **Cerrado:** La puerta está en la posición de cierre total, con el servo en el ángulo mínimo `ANGULO_CERRADO` y el LED rojo encendido. En este estado, el comando de apertura o una pulsación del pulsador abrirán la puerta.
- **Abierto:** La puerta está completamente abierta, con el servo en `ANGULO_ABIERTO` y el LED verde encendido. En este estado, el comando de cierre o el pulsador intentarán cerrar la puerta si las condiciones de luz lo permiten.
- **Bloqueo por condiciones de baja luz:** Si se intenta cerrar la puerta en condiciones de luz baja (valor de LDR menor que `UMBRAL_LUZ_BAJA`), la puerta permanece abierta y el LED rojo parpadea, indicando que no es seguro cerrarla.

En el código, el estado de la puerta se gestiona mediante la variable `estadoPuerta`.

```
1 //Variables de Control
2 const int ANGULO_ABIERTO = 90;
3 const int ANGULO_CERRADO = 0;
4 const int VELOCIDAD_SERVO = 20;
5 int estadoPuerta = ANGULO_CERRADO;
```

Definición de Pines y Constantes

Los pines principales incluyen LEDs (rojo y verde) para indicar el estado de la puerta, un LDR para medir las condiciones de luz, un pulsador y un servo para abrir o cerrar la puerta. La constante `UMBRAL_LUZ_BAJA` representa el nivel de luz mínimo necesario para permitir el cierre de la puerta:

```
1 #define LED_PIN_ROJO 4
2 #define LED_PIN_VERDE 2
3 #define SERVO_PIN 12
4 #define LDR_PIN 34
5 #define UMBRAL_LUZ_BAJA 1500
```

Loop Principal

El bucle principal ejecuta dos tareas fundamentales:

- **Control Bluetooth:** Cuando se recibe un dato ('a' para abrir o 'c' para cerrar), el sistema verifica el estado de la puerta. Si se envía el comando para cerrar y las condiciones de luz son insuficientes, se activa una alerta.
- **Control mediante pulsador:** El pulsador permite alternar entre apertura y cierre de la puerta. Si la luz es baja, se activa un aviso a través del LED rojo.

Funciones Principales

Función abrirPuerta()

Esta función abre la puerta ajustando el ángulo del servo hasta alcanzar el valor definido en la constante `ANGULO_ABIERTO`. Además, enciende el LED verde para indicar el estado de puerta abierta:

```
1 void abrirPuerta() {
2     digitalWrite(LED_PIN_ROJO, HIGH);
3     digitalWrite(LED_PIN_VERDE, LOW);
4     for (int angulo = ANGULO_CERRADO; angulo <= ANGULO_ABIERTO; angulo++) {
5         Motor.write(angulo);
6         delay(VELOCIDAD_SERVO);
7     }
8     estadoPuerta = ANGULO_ABIERTO;
9     digitalWrite(LED_PIN_VERDE, HIGH);
10 }
```

Función cerrarPuerta()

Esta función es similar a `abrirPuerta()`, pero en sentido inverso. El servo se mueve de `ANGULO_ABIERTO` a `ANGULO_CERRADO`, y se enciende el LED rojo.

Funciones Auxiliares

La función `onOffLeds()` hace que los LEDs parpadeen al inicio para indicar que el sistema está listo. La función `luzBaja()` lee el valor del sensor LDR y determina si las condiciones de luz son adecuadas para cerrar la puerta. Cuando el sensor detecta poca luz y se intenta cerrar la puerta, la función `parpadearLedRojo()` enciende y apaga el LED rojo como señal de advertencia.

5 Materiales y Costes

El tiempo estimado para la realización del proyecto fue de entre 30 y 40 horas. En cuanto a los costes económicos, los principales componentes del proyecto son material prestado por el profesor y pertenecen a SOL ([Smart Open Lab](#)).









Estado	Componente	Función	Precio (€)
	ESP-WROOM-32	Controlar la puerta y comunicación vía Bluetooth	12.00
	LDR o Fotorresistencia	Simular la presencia de un coche en el paso	0.30
	Servomotor HS-705MG	Mover la puerta	15.00
-	Un Smartphone con Android	App y control vía Bluetooth	—
	LEDs (verde y rojo)	Indicar el estado de la puerta	0.50
	Pulsador 4 patas	Activar manualmente el sistema	0.20
	Resistencias 220 Ω (x2)	Limitar corriente de los LEDs	0.10
	Resistencias 10 k Ω (x2)	Configuración del pulsador	0.10
	Cables y protoboard	Conexiones del circuito	3.00
Total			31.20

Table 1: Tabla de Materiales y Costes del Proyecto

6 Futuros Desarrollos Posibles

Este sistema de apertura de puerta de garaje, aunque ya es funcional y cuenta con los elementos básicos, tiene un gran potencial para ampliarse y adaptarse a diferentes necesidades del mundo real. Por ejemplo, se podrían añadir sensores adicionales que detecten condiciones ambientales con mayor precisión, como sensores de temperatura o de movimiento, para un control más completo y seguro de la puerta en diversas situaciones.

Además, reemplazar la conexión Bluetooth por WiFi sería una mejora significativa, ya que ampliaría el rango de control, permitiría acceder al sistema desde cualquier lugar con conexión a Internet y mejoraría la estabilidad de la comunicación. Esto podría ser especialmente útil en un entorno residencial o comercial, donde los usuarios necesitan monitorear y controlar la puerta de forma remota.

Otra mejora importante sería añadir un sistema de seguridad, como una alarma sonora mediante un zumbador piezoeléctrico, que se active si la puerta no puede cerrarse debido a condiciones de baja luz o a la presencia de un obstáculo. Esta funcionalidad no solo incrementaría la seguridad al alertar al usuario de posibles obstrucciones, sino que también haría el sistema más eficiente al informar rápidamente sobre incidencias.

Finalmente, montar el circuito en una pequeña carcasa o caseta de cartón no solo protegería los componentes electrónicos del entorno, sino que también mejoraría la presentación y facilitaría su manejo. Estas mejoras potenciales, además de optimizar el sistema, lo harían más versátil y permitirían su aplicación en distintos contextos, como sistemas de acceso inteligentes para garajes.

7 Conclusión

A través de este proyecto, he podido aplicar y consolidar conocimientos adquiridos en distintas asignaturas, además de profundizar en la plataforma ESP32, similar a Arduino en su enfoque de código abierto. Durante el desarrollo, no solo avancé en el manejo de microcontroladores, sino que también gané confianza en la programación, explorando herramientas que antes me resultaban desconocidas. Este proyecto me permitió ver cómo los conocimientos teóricos en electrónica y programación pueden cobrar vida en un sistema funcional, abordando problemas reales.

A lo largo del proceso, descubrí que, aunque ya tenía una base en C++ gracias a otras asignaturas, llevarlo al ámbito de Arduino y la ESP32 abrió nuevas perspectivas. También experimenté con componentes electrónicos como el LDR, LEDs y el servomotor, entendiendo no solo cómo funcionan, sino también cómo interactúan entre sí en un circuito real.

En conclusión, este proyecto me mostró la importancia de complementar los conocimientos teóricos con la práctica, superando los obstáculos propios del proceso y logrando que cada problema resuelto se convierta en un aprendizaje significativo.