



GRUPO: 4
SEMESTRE: 04

REPORTE DE SOFTWARE

SISTEMA DE ESTACIONAMIENTO

SIMULACIÓN

PROFESORA: MANZANAREZ ALCAZAR IRIS BERENICE

INTEGRANTES:

- 1.- BUSTAMANTE MONROY JONATHAN OSVALDO
- 2.- HERNÁNDEZ MENA MELANIE LIZETH
- 3.- RAMIREZ RODRÍGUEZ KENNETH MARTIN
- 4.- ZEQUERA AYALA YAMIR
- 5.- PEREA CABALLERO RICARDO
- 6.- NAJERA MORAN ALEJANDRO

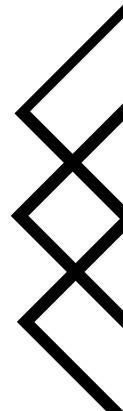




Tabla de contenido

02

| | | |
|-----|---|----|
| 1. | INTRODUCCIÓN..... | 03 |
| 2. | DISEÑO DEL SISTEMA..... | 04 |
| 2.1 | Materiales..... | 05 |
| 2.2 | Diagrama de circuitos..... | 06 |
| 3. | DESARROLLO DEL SOFTWARE..... | 07 |
| 3.1 | Librerías utilizadas..... | 08 |
| 3.2 | Inicialización y configuración..... | 08 |
| 3.3 | Configuración inicial y presentación..... | 09 |
| 3.4 | Bucle principal y actualización de la pantalla LCD..... | 10 |
| 3.5 | Función Read_Sensor..... | 11 |
| 3.6 | Resumen del código..... | 12 |
| 3.7 | Código completo..... | 12 |
| 4. | DESARROLLO DE LA MAQUETA..... | 13 |
| 4.1 | Diseño de la maqueta..... | 14 |
| 4.2 | Materiales..... | 15 |
| 4.3 | Desarrollo..... | 16 |
| 4.4 | Implementación del circuito..... | 18 |
| 5. | PRUEBAS Y ANALISIS DE RESULTADOS..... | 20 |
| 5.1 | Planteamiento de problemáticas y soluciones propuestas..... | 21 |
| 5.2 | Problema: Detección incorrecta debido a la sensibilidad de los sensores..... | 21 |
| 5.3 | Problemas de conexión y cableado..... | 22 |
| 5.4 | Recomendaciones..... | 23 |
| 6. | CONCLUSIÓN..... | 24 |
| 7. | REFERENCIAS Y FUENTES DE INFORMACIÓN..... | 26 |



INTRODUCCIÓN

CO

El presente informe resume el desarrollo de un sistema de estacionamiento basado en Arduino, que fue implementado y probado utilizando una maqueta física. Este proyecto fue realizado como parte de la materia de simulación, con el propósito de aplicar los conceptos y técnicas aprendidas en un contexto práctico.

El sistema de estacionamiento utiliza sensores para detectar la disponibilidad de espacios y guiar a los conductores hacia las áreas libres. Se desarrolló un código en Arduino para controlar el sistema y se creó una maqueta física para simular el entorno de estacionamiento.

Las pruebas realizadas demostraron un desempeño satisfactorio del sistema, con una alta precisión en la detección de espacios disponibles y una respuesta adecuada en situaciones de alta demanda. Además, el proyecto ha permitido aplicar los conocimientos teóricos de simulación en un caso práctico de ingeniería de software y electrónica.

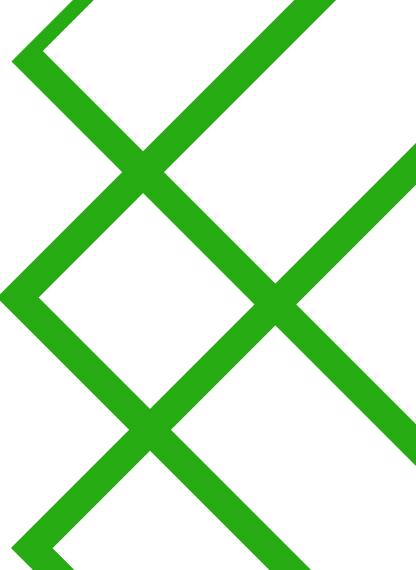
Las conclusiones obtenidas destacan la viabilidad y eficacia del sistema de estacionamiento desarrollado, así como su potencial para mejorar la gestión de los espacios de estacionamiento y la experiencia de los usuarios. Se recomienda explorar posibles mejoras y ampliaciones del sistema en futuras implementaciones.

El proyecto de sistema de estacionamiento en Arduino y la maqueta física representa una valiosa aplicación de la materia de simulación, brindando una oportunidad de adquirir habilidades prácticas y comprender las aplicaciones reales de los conceptos estudiados.

Ten en cuenta que un resumen ejecutivo es una sección breve y concisa que presenta los objetivos, resultados clave y recomendaciones principales del informe. Puedes ajustar el resumen ejecutivo generado según los aspectos más relevantes y destacados de tu proyecto.



DISEÑO DEL SISTEMA



Se describe el diseño del sistema de estacionamiento en Arduino, incluyendo los componentes utilizados, su interconexión y cualquier consideración de diseño relevante.

MATERIALES

GO

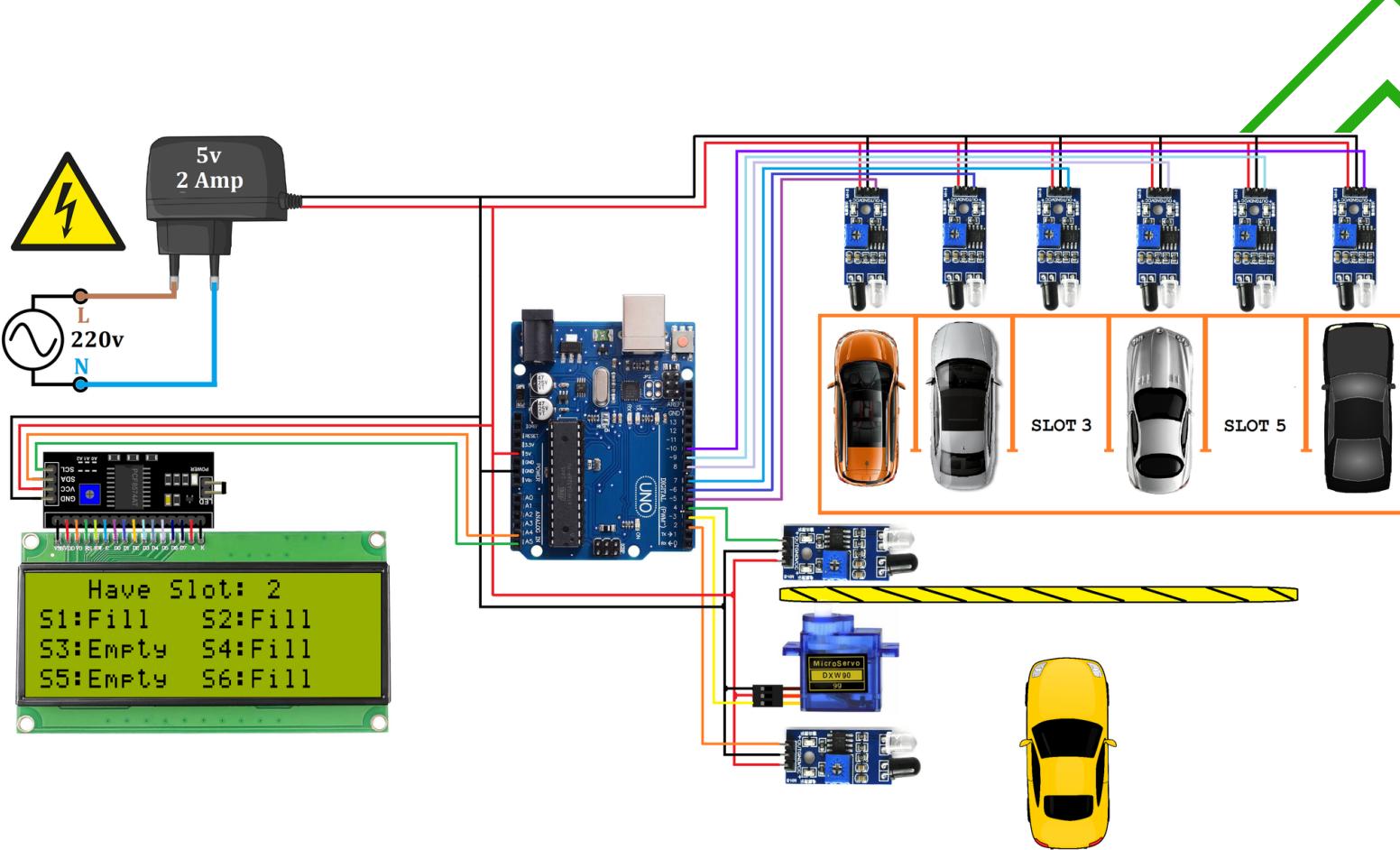
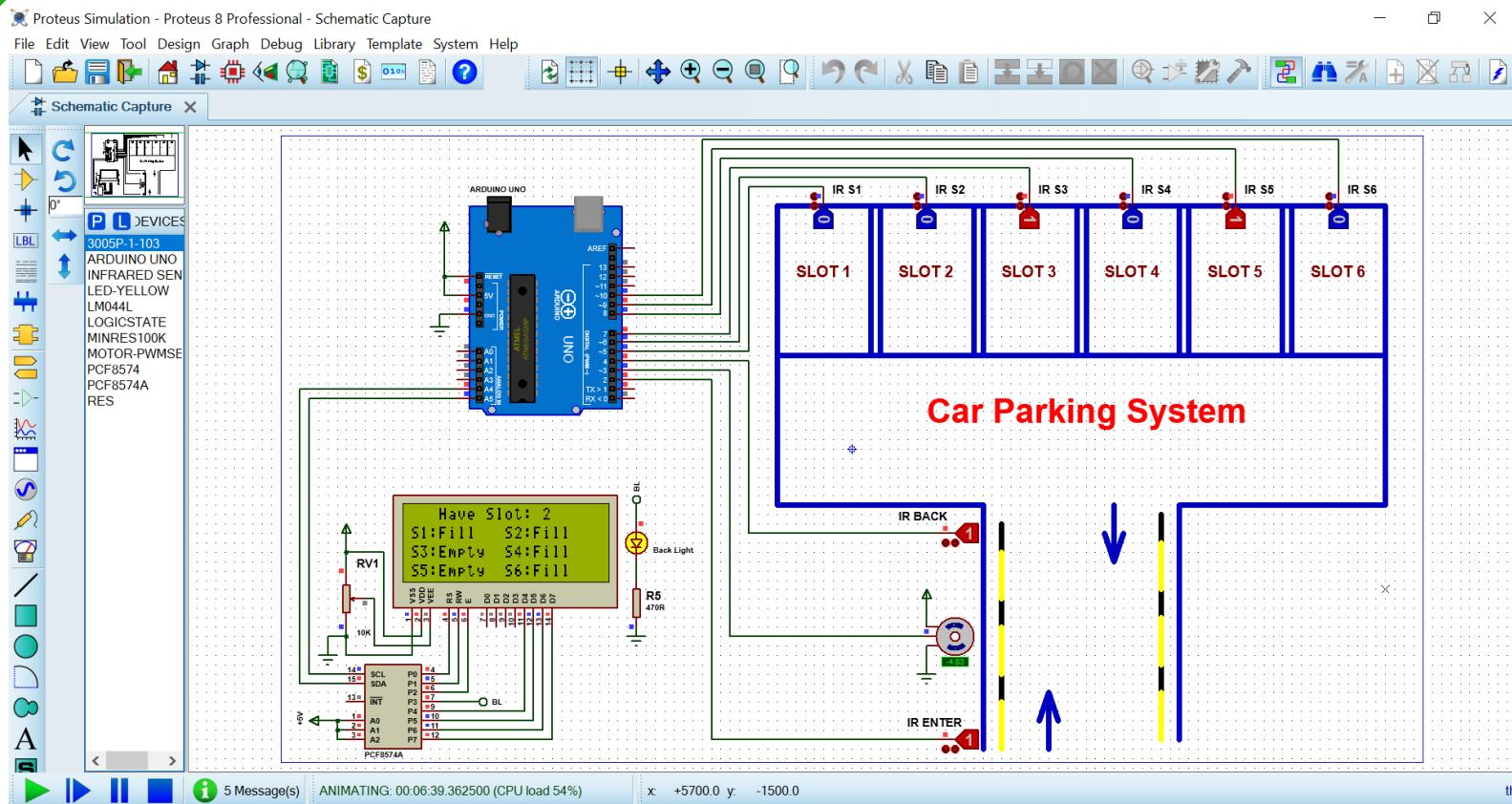
Para la implementación del sistema de estacionamiento en Arduino, se utilizaron los siguientes materiales:

- 1. Arduino Uno:** Se empleó una placa Arduino Uno como el microcontrolador principal del sistema. Esta placa proporciona una plataforma versátil y de fácil programación para controlar los componentes del sistema.
- 2. 16x2 LCD Display:** Se utilizó un display LCD de 16x2 caracteres para mostrar información relevante, como la disponibilidad de plazas de estacionamiento. Este display permitió una interfaz visual clara y legible para los usuarios.
- 3. I2C LCD Module:** Se empleó un módulo I2C para el display LCD, el cual simplifica la conexión y permite una comunicación más eficiente entre el Arduino y el display. Esta opción I2C facilita el cableado y reduce el número de pines utilizados en el Arduino.
- 4. IR Sensor x 8:** Se utilizaron ocho sensores infrarrojos (IR) para detectar la presencia de vehículos en las plazas de estacionamiento. Estos sensores permiten una detección precisa y confiable, y su implementación en conjunto permite una cobertura completa del área de estacionamiento.
- 5. Mini Servo Motor SG-90:** Se utilizó un mini servo motor SG-90 para simular la apertura y cierre de las barreras del estacionamiento. Este motor permite un movimiento controlado y preciso de las barreras, brindando una experiencia realista y funcional en la maqueta física.
- 6. Female DC Power Jack:** Se empleó un conector hembra DC Power Jack para facilitar la conexión de la fuente de alimentación al Arduino. Este conector proporciona una conexión segura y estable para suministrar energía al sistema.
- 7. Fuente de alimentación de 5V y 2Amp:** Se utilizó una fuente de alimentación externa de 5V y 2Amp para proporcionar la energía necesaria al Arduino y a los componentes del sistema. Esta fuente de alimentación garantiza un suministro de energía estable y adecuado para el funcionamiento correcto del sistema.
- 8. Protoboard:** Se empleó una protoboard para facilitar la conexión y la disposición de los componentes en el circuito. La protoboard permite una organización ordenada de los cables y los componentes, lo cual simplifica el proceso de montaje y pruebas.
- 9. Jumpers macho a macho:** Se utilizaron 20 jumpers macho a macho para realizar las conexiones entre los componentes en el circuito. Estos jumpers proporcionan una conexión fácil y rápida, permitiendo una configuración flexible y modificaciones en el diseño del circuito.
- 10. Jumpers macho a hembra:** Se emplearon 20 jumpers macho a hembra para realizar las conexiones entre el Arduino y los diferentes componentes. Estos jumpers permiten una conexión segura y confiable, facilitando la interconexión del Arduino con el display LCD, los sensores, el servo motor y otros componentes del sistema.

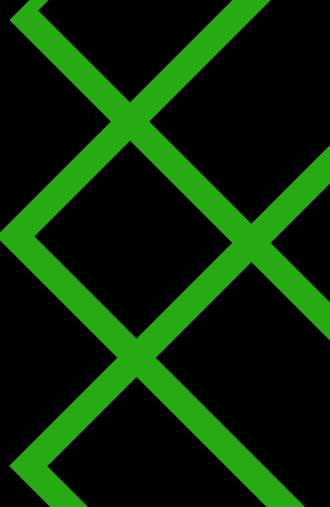
La combinación de estos materiales proporcionó una base sólida para la implementación exitosa del sistema de estacionamiento en Arduino. Cada uno de estos elementos desempeñó un papel fundamental en el funcionamiento y la interacción de los componentes, permitiendo una experiencia completa y funcional en la maqueta física desarrollada.

DIAGRAMA DE CIRCUITOS

90



DESARROLLO DEL SOFTWARE



Se describe cómo se desarrolló el código, las funciones principales y cómo se relaciona con la configuración física.

A continuación se presenta una explicación detallada del código desarrollado en Arduino, para controlar un sistema de estacionamiento. Se describirán las funciones principales, las librerías utilizadas y toda la lógica tras este software.

Librerías utilizadas:

El código hace uso de las siguientes librerías:

1. **Servo:** Esta librería permite controlar un servo motor conectado al Arduino.
2. **Wire:** Esta librería es necesaria para la comunicación I2C utilizada para el control de la pantalla LCD.
3. **LiquidCrystal_I2C:** Esta librería proporciona funciones para interactuar con una pantalla LCD basada en el protocolo I2C.

```

1 //Estas son las librerias que vamos a ocupar
2 #include <Servo.h>
3 #include <Wire.h>
4 #include <LiquidCrystal_I2C.h>
5 // LiquidCrystal_I2C lcd(0x3f,16,2); // si no te sale con esta direccion puedes usar (0x3f,16,2) (0x27,16,2) (0x20,16,2)
6

```

Inicialización y configuración:

El código Arduino comienza con la inicialización y configuración necesaria antes de que comience el bucle principal.

1. Se crea un objeto **LiquidCrystal_I2C** llamado **lcd** para interactuar con la pantalla LCD. En este objeto se establece la dirección I2C del dispositivo y se definen los pines de conexión.

```

8 //Se crea el objeto de la libreria LiquidCrystal_I2C y se le pasa como parametros los pines que se van a usar
9 LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

```

2. Se crea un objeto **Servo** llamado **myservo** para controlar el servo motor. Este objeto se adjunta al pin correspondiente al servo motor.

```

11 //Se crea el objeto de la libreria del servomotor
12 Servo myservo;
37 //Se define que pin va a usar el servomotor para que se le pasen ordenes
38 myservo.attach(3);

```

3. Se definen constantes y variables que representan los pines utilizados por los sensores infrarrojos y las variables para llevar el registro de la disponibilidad de los espacios.

```

14 //Se definen las variables que hacen referencia a los pines del arduino
15 #define ir_enter 2
16 #define ir_back 4
17
18 #define ir_car1 5
19 #define ir_car2 6
20 #define ir_car3 7
21
22 //se crean variables que nos van a ayudar a crear la lógica de los lugares disponibles
23 int S1 = 0, S2 = 0, S3 = 0;
24 int flag1 = 0, flag2 = 0;
25 int slot = 3;

```

Configuración inicial y presentación:

Después de la inicialización y configuración, el programa Arduino pasa a la función **setup()**, donde se realizan las siguientes tareas:

1. Se configuran los pines de los sensores y del servo motor como entradas o salidas según corresponda.

```

29 | //Se declara que los pines van a ser usado para leer datos
30 | pinMode(ir_car1, INPUT);
31 | pinMode(ir_car2, INPUT);
32 | pinMode(ir_car3, INPUT);
33 |
34 | pinMode(ir_enter, INPUT);
35 | pinMode(ir_back, INPUT);

```

2. Se establece la posición inicial del servo motor en 90 grados utilizando la función **myservo.write(90)**.

```

40 | //Le da la orden de que el servo motor se ponga en la posicion de 90 grados
41 | myservo.write(90);

```

3. Se inicializa la pantalla LCD con un mensaje de bienvenida utilizando la función **lcd.begin(16, 2)** y se muestra el mensaje en la pantalla utilizando la función **lcd.print()**.

```

48 | //Tamaño de pixeles de la pantalla
49 | lcd.begin(16, 2);
50 | //Que se posiciones en la cordenada dada
51 | lcd.setCursor(0, 0);
52 | //Que ponga un texto
53 | lcd.print(" Bienvenido ");
54 | //Que se posiciones en la cordenada dada
55 | lcd.setCursor(0, 1);
56 | //Que ponga un texto
57 | lcd.print(" Estacionamiento");

```

```

59 | //Esto es un delay del arduino
60 | delay(2000);
61 | //Se limpie la pantalla del arduino
62 | lcd.clear();

```

4. Se llama a la función **Read_Sensor()** para leer el estado de los sensores infrarrojos y actualizar las variables de disponibilidad de los espacios.

```

64 | //Llamada a una funcion
65 | Read_Sensor();

```

5. Se calcula la cantidad de espacios disponibles restando el número de espacios ocupados ($S1 + S2 + S3$) a la cantidad total de espacios (slot = 3).

```

67 | int total = S1 + S2 + S3;
68 | slot = slot - total;

```

Bucle principal y actualización de la pantalla LCD:

Después de la configuración inicial, el programa Arduino entra en el bucle principal **loop()**, donde se realizan las siguientes acciones repetidamente:

Se llama a la función **Read_Sensor()** para leer el estado actual de los sensores infrarrojos y actualizar las variables de disponibilidad de los espacios.

```
71 void loop() {
72     Read_Sensor();
```

2. Se actualiza la pantalla LCD con la información sobre la cantidad de espacios disponibles y el estado de cada espacio.

```
75     lcd.setCursor(0, 0);
76     lcd.print(" Disponibles: ");
77     lcd.print(slot);
78     lcd.print("   ");
79
80     lcd.setCursor(0, 1);
81     if (S1 == 1) {
82         lcd.print("S1:F ");
83     } else {
84         lcd.print("S1:E");
85     }
86
87     lcd.setCursor(6, 1);
88     if (S2 == 1) {
89         lcd.print("S2:F ");
90     } else {
91         lcd.print("S2:E");
92     }
93
94     lcd.setCursor(12, 1);
95     if (S3 == 1) {
96         lcd.print("S3:F ");
97     } else {
98         lcd.print("S3:E");
99     }
```

3. Se verifica si se ha presionado el botón de entrada (**ir_enter**). Si hay espacios disponibles y no se ha activado la bandera **flag1**, se activa la bandera y se mueve el servo motor a 180 grados para permitir la entrada de un vehículo. Además, se actualiza la cantidad de espacios disponibles.

```
102    if (digitalRead(ir_enter) == 0 && flag1 == 0) {
103        if (slot > 0) {
104            flag1 = 1;
105            if (flag2 == 0) {
106                myservo.write(180);
107                slot = slot - 1;
108            }
109        } else {
110            lcd.setCursor(0, 0);
111            lcd.print(" Sin lugares ");
112            delay(1500);
```

4. Se verifica si se ha presionado el botón de salida (**ir_back**). Si no se ha activado la bandera **flag2**, se activa la bandera y se mueve el servo motor a 180 grados en la dirección opuesta para permitir la salida de un vehículo. Además, se actualiza la cantidad de espacios disponibles.

```
116  if (digitalRead(ir_back) == 0 && flag2 == 0) {  
117      flag2 = 1;  
118      if (flag1 == 0) {  
119          myservo.write(180);  
120          slot = slot + 1;
```

5. Si ambas banderas **flag1** y **flag2** están activadas, se espera un segundo y se mueve el servo motor a 90 grados para restablecer su posición inicial. Luego, se desactivan ambas banderas.

```
124  if (flag1 == 1 && flag2 == 1) {  
125      delay(1000);  
126      myservo.write(90);  
127      flag1 = 0, flag2 = 0;
```

6. Se agrega un retraso de 1 milisegundo en cada iteración del bucle para permitir el funcionamiento adecuado del programa.

```
130  delay(1);
```

Función **Read_Sensor()** :

La función **Read_Sensor()** se encarga de leer el estado de los sensores infrarrojos que detectan la presencia de vehículos en cada espacio de estacionamiento. A continuación se muestra una descripción de dicha función:

```
133 void Read_Sensor() {  
134     S1 = 0, S2 = 0, S3 = 0;  
135  
136     if (digitalRead(ir_car1) == 0) { S1 = 1; }  
137     if (digitalRead(ir_car2) == 0) { S2 = 1; }  
138     if (digitalRead(ir_car3) == 0) { S3 = 1; }
```

La función **Read_Sensor()** se encarga de leer el estado de los sensores infrarrojos conectados a los pines **ir_car1**, **ir_car2** y **ir_car3**. Estos sensores son utilizados para detectar la presencia de vehículos en los espacios de estacionamiento correspondientes.

En la primera línea de la función, se inicializan las variables **S1**, **S2** y **S3** con el valor 0 para indicar que inicialmente los espacios de estacionamiento están vacíos.

A continuación, se utilizan las sentencias **if** para verificar el estado de cada sensor. Si el valor leído en alguno de los sensores es igual a 0, se asigna el valor 1 a la variable correspondiente (**S1**, **S2** o **S3**). Esto indica que hay un vehículo presente en el espacio de estacionamiento respectivo.

Resumen del código:

1. Importar las librerías necesarias: **Servo.h**, **Wire.h** y **LiquidCrystal_I2C.h**.
2. Crear un objeto **LiquidCrystal_I2C** llamado **Lcd** con los parámetros de inicialización correspondientes.
3. Crear un objeto **Servo** llamado **myservo**.
4. Definir las constantes que representan los pines del Arduino: **ir_enter**, **ir_back**, **ir_car1**, **ir_car2** y **ir_car3**.
5. Declarar las variables **S1**, **S2** y **S3** e inicializarlas con el valor 0. Estas variables representan la disponibilidad de los espacios de estacionamiento.
6. Declarar las variables **flag1** y **flag2** e inicializarlas con el valor 0. Estas variables controlan el estado de los sensores de entrada y salida.
7. Declarar la variable **slot** e inicializarla con el valor 3. Esta variable representa la cantidad de espacios de estacionamiento disponibles inicialmente.
8. En la función **setup()**, configurar los pines como entradas o salidas según corresponda.
9. Asociar el pin del servo motor al objeto **myservo** usando **myservo.attach(3)**.
10. Mover el servo motor a la posición 90 grados usando **myservo.write(90)**.
11. Inicializar el objeto **Lcd** y configurar la pantalla mostrando un mensaje de bienvenida durante 2 segundos.
12. Llamar a la función **Read_Sensor()** para obtener el estado de los sensores de los espacios de estacionamiento.
13. Calcular la cantidad total de espacios ocupados sumando **S1**, **S2** y **S3**, y actualizar el valor de **slot** restando la cantidad total de espacios ocupados.
14. En el bucle **loop()**, llamar a la función **Read_Sensor()** para obtener el estado actual de los sensores.
15. Actualizar la pantalla LCD con la cantidad de espacios disponibles y el estado de cada espacio de estacionamiento.
16. Verificar si se presionó el botón de entrada (**ir_enter**) y si **flag1** es 0. Si hay espacios disponibles, mover el servo motor a la posición 180 grados, disminuir **slot** en 1 y actualizar **flag1** a 1.
17. Si no hay espacios disponibles, mostrar un mensaje en la pantalla indicando que no hay lugares.
18. Verificar si se presionó el botón de salida (**ir_back**) y si **flag2** es 0. Mover el servo motor a la posición 180 grados, aumentar **slot** en 1 y actualizar **flag2** a 1.
19. Si se presionaron ambos botones, esperar 1 segundo, mover el servo motor a la posición 90 grados y restablecer **flag1** y **flag2** a 0.
20. Agregar un pequeño retardo de 1 milisegundo al final del bucle **loop()** para evitar un procesamiento excesivo.

Código completo:

https://drive.google.com/drive/folders/1Xw1EVUdrf_M9vzDj1Eo1E7GbjVTLTt3G?usp=share_link

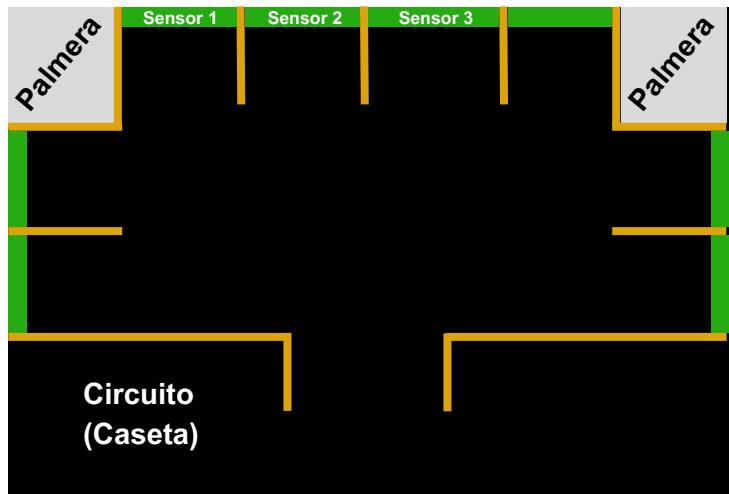
DESARROLLO DE LA MAQUETA



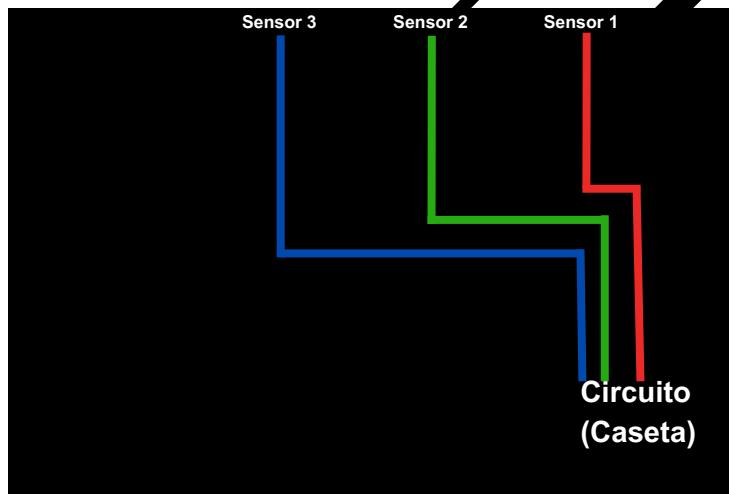
Se describe la maqueta física que has construido para probar el sistema. Menciona los materiales utilizados, el diseño de la maqueta y cómo se relaciona con la implementación del sistema en Arduino.

Diseño de la maqueta:

Parte delantera:



Parte trasera:



Materiales:

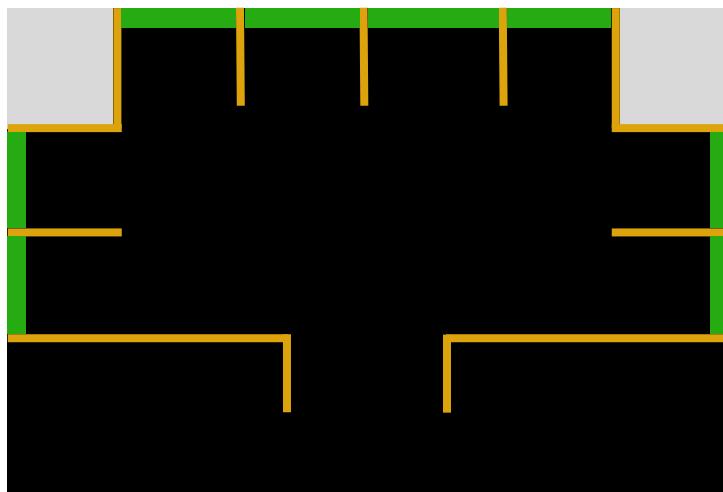
- Un tamaño considerable de cartón
- Cinta doble cara
- Una hoja de fomi amarillo
- Una hoja de fomi blanco
- 3 rojos de papel terminados
- Tijeras
- Silicon frío
- Unicel
- Pintura verde
- Pintura café
- Pincel

Desarrollo:

- Se pinto el cartón para la base del estacionamiento, de un color negro para simular el cemento
- Con silicon se empezó a pegar tiras de fomi amarillo para simular, las líneas que delimitan cada espacio del estacionamiento. Igual se pego 2 cuadros de fomi blanco



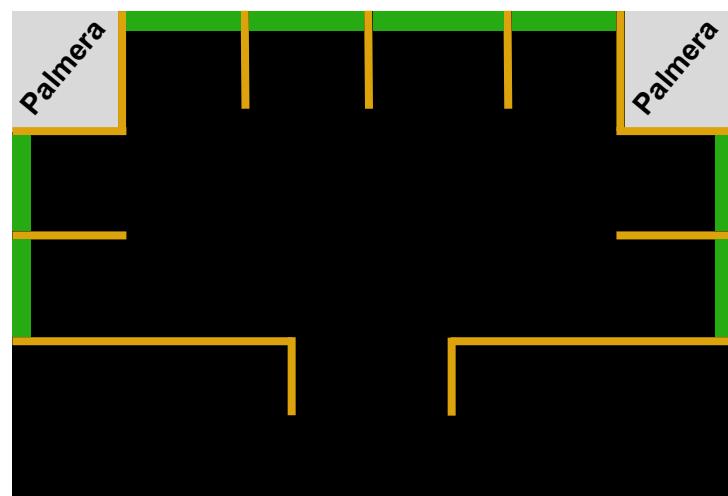
- Se pintaron de verde, 6 rectángulos de unicel, para simular un tipo de jardinera tope



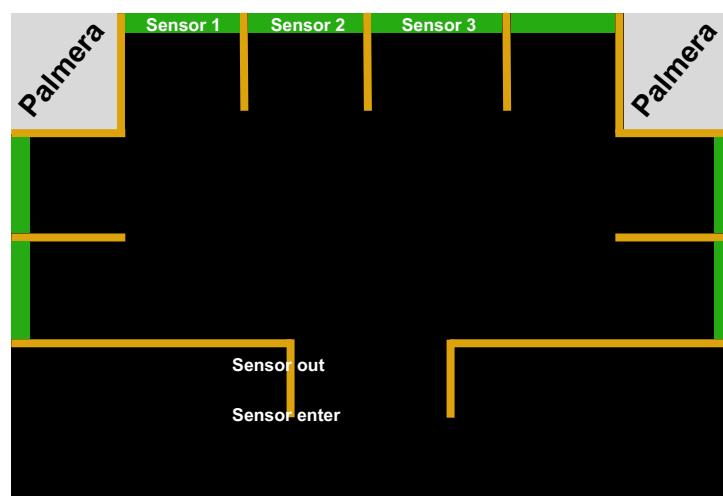
- Con rollos de papel se hizo unas palmeras: Primero se enrollaron dos para hacer el palo de la palmera, luego con otro rollo se hace las hojas de la palmera, luego se pegan con siliconón en la parte superior del palo, para después pintar la palmera y colocarles con siliconón una base cuadrada de cartón.



- Después las palmeras se colocan con siliconón en donde están los cuadrados de fomi blanco.

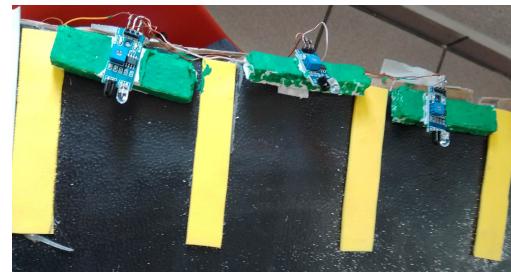
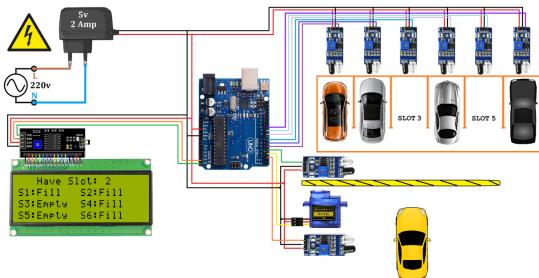


- Luego con cinta doble cara, se colocan los sensores arriba de los rectángulos de unicel ver

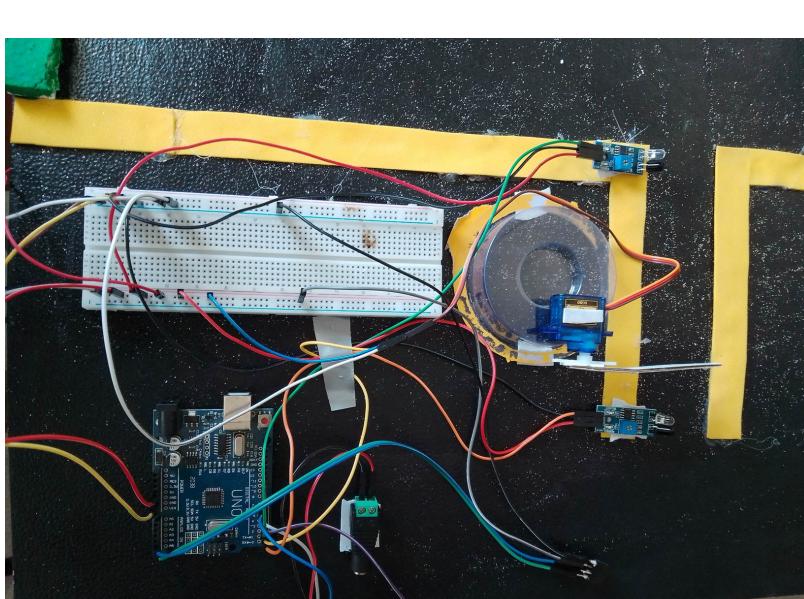
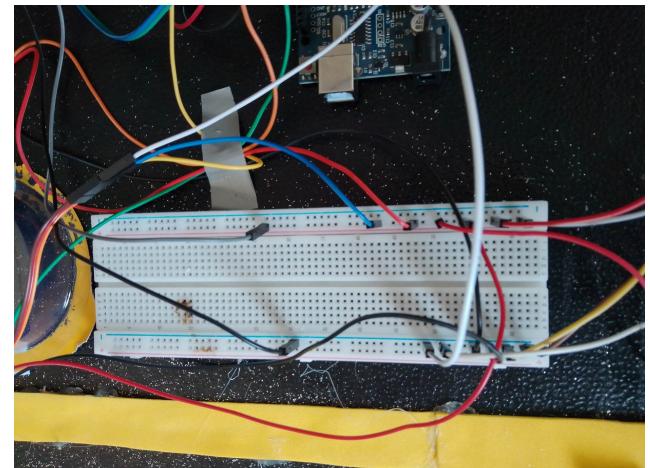
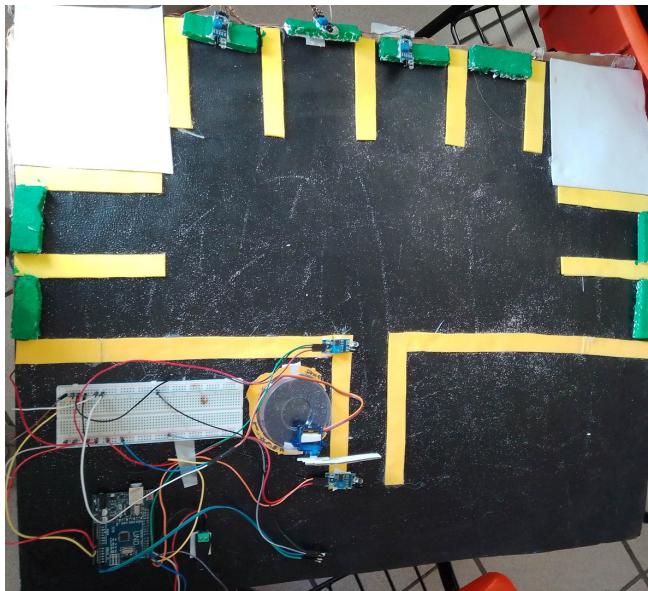


Implementación del circuito:

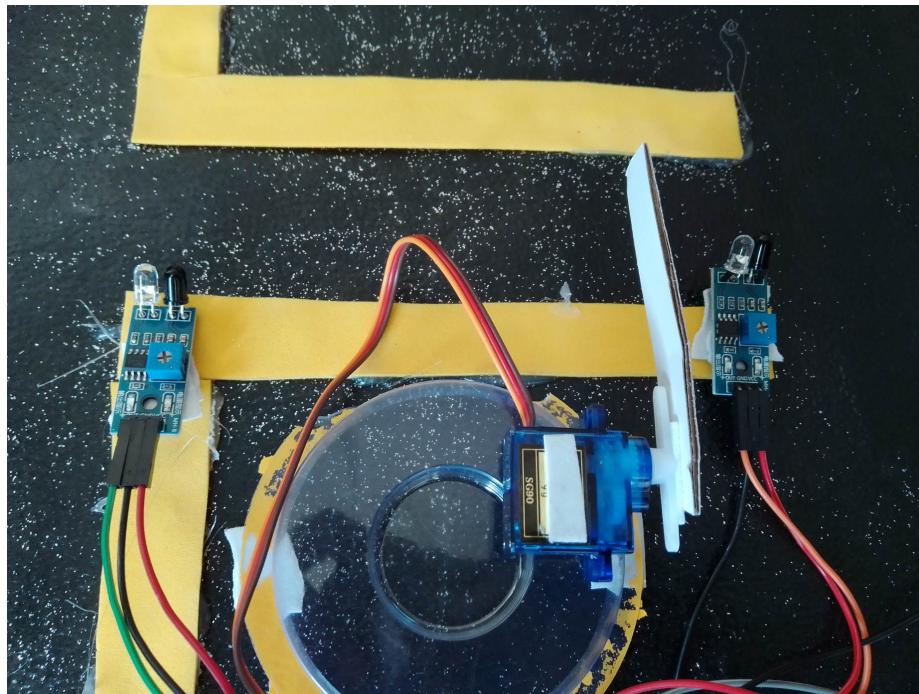
- Se soldó cables en los tres pines de cada sensor: en el de en medio va el negativo, en el de la derecha va el que manda la información (ceros y unos) y en el de la izquierda el positivo.



- Luego los cables se conectan al protoboard y al arduino



- También se conecta el servo motor al arduino, y se le coloca con cinta doble cara, un rectángulo de cartón blanco, para simular el paso del estacionamiento.



- Se coloca la pantalla LCD, al circuito.



PRUEBAS Y ANALISIS DE RESULTADOS

Explica cómo se llevaron a cabo las pruebas del sistema de estacionamiento y qué resultados se obtuvieron. Describe cualquier problema encontrado durante las pruebas y cómo se resolvieron. Incluye datos cuantitativos o cualitativos relevantes, como la precisión del sistema o el rendimiento en diferentes situaciones.

Planteamiento de problemáticas y soluciones propuestas:

Problema: Detección incorrecta debido a la sensibilidad de los sensores

Durante las pruebas de implementación, se encontró un problema relacionado con la sensibilidad de los sensores infrarrojos utilizados para detectar la presencia de vehículos en los espacios de estacionamiento. El problema radica en que los sensores presentan una sensibilidad excesiva, lo que resulta en una detección incorrecta y la indicación errónea de que todos los espacios de estacionamiento están ocupados. Esto puede generar confusión y afectar el funcionamiento adecuado del sistema.

Solución propuesta:

Para abordar este problema, se pueden considerar las siguientes soluciones:

1. **Ajuste de la sensibilidad de los sensores:** Es necesario ajustar la sensibilidad de los sensores infrarrojos para asegurarse de que detecten correctamente la presencia de vehículos. Esto se puede lograr mediante la configuración de los parámetros de sensibilidad disponibles en los sensores o utilizando componentes adicionales, como potenciómetros, para ajustar manualmente la sensibilidad. Realiza pruebas y ajustes graduales para encontrar el equilibrio adecuado que permita una detección confiable y precisa de los vehículos.
2. **Verificación de la distancia de detección:** Verifica la distancia de detección de los sensores infrarrojos y asegúrate de que se ajuste a tus necesidades. Algunos sensores infrarrojos tienen un alcance limitado, por lo que es importante verificar que la distancia de detección se encuentre dentro de los límites deseados. Puedes hacer pruebas colocando vehículos a diferentes distancias de los sensores y verificando si se detectan correctamente.
3. **Pruebas exhaustivas de detección:** Realiza pruebas exhaustivas para verificar la capacidad de los sensores para detectar la presencia de vehículos en diferentes condiciones. Prueba con diferentes objetos y superficies opacas para evaluar la precisión de la detección. Además, verifica el comportamiento de los sensores en diferentes condiciones ambientales, como variaciones de luz o temperatura, para garantizar la confiabilidad de la detección en situaciones reales.
4. **Calibración y ajuste automático:** Considera implementar un mecanismo de calibración o ajuste automático de la sensibilidad de los sensores. Esto permitirá que el sistema se adapte automáticamente a cambios en las condiciones ambientales y mejore la precisión de la detección. Puedes utilizar algoritmos de calibración que ajusten la sensibilidad en función de la variabilidad de la señal detectada.

Análisis: La solución fue ir ajustando la sensibilidad e ir probando con varios objetos, lo que se noto es que entre mas negro sea el objeto, mas se le dificulta al sensor detectar dicho objeto, así que también se ajusto la altura en donde esta el sensor para que sea mas fácil detectar a la mayoría de carros.

Problemas de conexión y cableado:

Durante la implementación del sistema de estacionamiento, es fundamental prestar atención a las conexiones y al cableado utilizado. Los problemas de conexión suelta o cables defectuosos pueden generar fallas en la detección y en la comunicación entre los componentes del sistema, lo que afectará negativamente su funcionamiento. A continuación, se detallan algunas consideraciones importantes para abordar los problemas de conexión y cableado:

Solución propuesta:

Para abordar este problema, se pueden considerar las siguientes soluciones:

1. **Verificación de conexiones:** Asegúrate de que todas las conexiones estén realizadas correctamente. Verifica que los cables estén firmemente conectados a los pines correspondientes de los componentes, como los sensores y el Arduino. Inspecciona visualmente cada conexión para identificar cualquier conexión suelta o mal conectada. Si es necesario, vuelve a conectar los cables de manera adecuada y asegúrate de que estén firmemente sujetos.
2. **Comprobación de cables defectuosos:** Los cables defectuosos pueden causar problemas de conexión e interrupciones en la transmisión de datos. Realiza una inspección minuciosa de los cables utilizados en el sistema y verifica si hay signos de daño, como cables pelados, roturas o dobleces excesivos. Si encuentras algún cable defectuoso, reemplázalo por uno nuevo y asegúrate de utilizar cables de buena calidad que sean adecuados para la aplicación.
3. **Utilización de conectores adecuados:** Para evitar conexiones sueltas o inestables, utiliza conectores adecuados para realizar las conexiones entre los componentes. Los conectores, como conectores tipo macho y hembra, proporcionan una conexión segura y confiable. Asegúrate de utilizar los conectores adecuados para los cables utilizados y sigue las instrucciones de conexión recomendadas por los fabricantes.
4. **Organización y asegurado de cables:** Mantén el cableado del sistema de manera organizada y asegurada. Utiliza bridás, sujetadores o canaletas para mantener los cables en su lugar y evitar que se enreden o se muevan de forma inadvertida. Un cableado desordenado aumenta el riesgo de conexiones sueltas y dificulta la identificación y solución de problemas.
5. **Pruebas de continuidad:** Realiza pruebas de continuidad en los cables para verificar si hay conexiones interrumpidas o cables rotos. Utiliza un multímetro o una herramienta de prueba de continuidad para verificar que la señal eléctrica pueda fluir correctamente a través de cada cable. Esto te ayudará a identificar y solucionar problemas de conexión y cableado.
6. **Resguardo de conexiones expuestas:** Si hay conexiones expuestas, como empalmes de cables, asegúrate de resguardarlos adecuadamente para evitar cortocircuitos o contacto accidental con otros componentes o superficies. Utiliza cinta aislante o cubiertas protectoras para proteger las conexiones expuestas y garantizar la seguridad y confiabilidad del sistema.

Análisis: Al verificar los cables, algunos estaban haciendo contacto, así que los aislamos con silicón, lo que se aprende de esto, es que no se debe pelar de mas los cables, solo lo necesario.

Recomendaciones:

Para garantizar el correcto funcionamiento del código, se recomienda realizar pruebas exhaustivas y realizar un análisis de su rendimiento. A continuación, se presentan algunas recomendaciones y consideraciones para llevar a cabo las pruebas y el análisis del código:

1. Pruebas de sensores:

- Verifica que los sensores infrarrojos estén correctamente conectados y configurados.
- Coloca objetos o superficies opacas frente a cada sensor para simular la presencia de vehículos y asegurarte de que los sensores detecten correctamente estos cambios.
- Realiza pruebas tanto con los espacios de estacionamiento vacíos como ocupados para asegurarte de que los sensores brinden los resultados esperados en ambos casos.

2. Pruebas de funcionalidad:

- Verifica que la pantalla LCD muestre correctamente la disponibilidad de espacios y el estado de cada uno.
- Realiza pruebas de entrada y salida de vehículos para asegurarte de que se actualice correctamente el número de espacios disponibles.
- Asegúrate de que el servo motor se mueva adecuadamente al recibir señales para permitir la entrada y salida de vehículos.

3. Pruebas de límites:

- Evalúa el comportamiento del programa en situaciones límite, como cuando todos los espacios de estacionamiento están ocupados o cuando no hay espacios disponibles.
- Verifica que el programa maneje adecuadamente estas situaciones límite y brinde mensajes claros y precisos en la pantalla LCD.

4. Análisis de rendimiento:

- Realiza un análisis del rendimiento del código para asegurarte de que no haya retrasos o bloqueos inesperados.
- Verifica que el código sea eficiente y que no haya secciones que consuman demasiados recursos o provoquen retrasos significativos en la ejecución.
- Realiza pruebas de carga para simular diferentes escenarios y asegurarte de que el programa funcione correctamente incluso en situaciones de alta demanda.

5. Depuración y ajustes:

- Utiliza herramientas de depuración para identificar posibles errores o problemas en el código.
- Realiza ajustes y modificaciones en el código según sea necesario para corregir errores, mejorar la eficiencia o agregar nuevas funcionalidades.

Las pruebas exhaustivas y el análisis del rendimiento son fundamentales para garantizar el correcto funcionamiento del código de control del sistema de estacionamiento. Mediante la realización de pruebas de sensores, funcionalidad y límites, y el análisis del rendimiento, se puede detectar y corregir posibles problemas o limitaciones del código, mejorando así la confiabilidad y eficiencia del sistema en general.

CONCLUSIÓN

24

Los principales hallazgos del proyecto y las lecciones aprendidas. Evalúa el éxito del sistema de estacionamiento en términos de cumplimiento de los objetivos establecidos. También puedes mencionar posibles mejoras o áreas de desarrollo futuro.

En conclusión, el proyecto de implementación del sistema de estacionamiento automatizado utilizando Arduino ha sido un proceso apasionante y gratificante. A lo largo de este proyecto, se han abordado diferentes aspectos clave relacionados con la detección de vehículos, el control de la barrera y la visualización de la disponibilidad de espacios de estacionamiento.

Mediante el uso de componentes electrónicos como sensores infrarrojos, un servomotor y una pantalla LCD, se ha logrado desarrollar un sistema funcional y eficiente que optimiza el control y la gestión de los espacios de estacionamiento.

Durante la implementación, se han enfrentado desafíos significativos que han requerido un enfoque creativo y soluciones innovadoras. Uno de los desafíos más comunes ha sido el ajuste de la sensibilidad de los sensores. La correcta calibración de los sensores es crucial para garantizar una detección precisa de la presencia de vehículos. Se han realizado pruebas y ajustes exhaustivos para lograr una sensibilidad óptima y evitar falsas detecciones o no detecciones.

Además, los problemas de conexión y cableado también han sido una preocupación importante en el desarrollo del proyecto. Las conexiones sueltas o los cables defectuosos pueden afectar la comunicación entre los componentes del sistema y generar fallas en su funcionamiento. Se ha dedicado tiempo y esfuerzo a verificar y asegurar todas las conexiones, así como a utilizar cables de calidad para garantizar una conexión sólida y confiable.

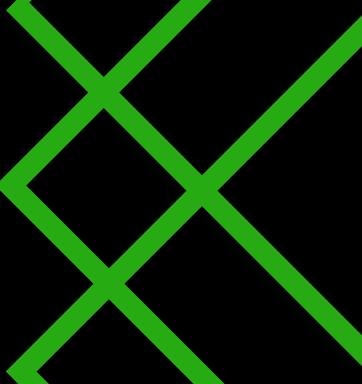
A medida que el proyecto avanzaba, se han realizado pruebas rigurosas para verificar el rendimiento del sistema en diferentes escenarios y condiciones. Estas pruebas han permitido identificar y resolver problemas adicionales, como los errores en el procesamiento de datos. Mediante ajustes en la programación y la implementación de técnicas de filtrado y validación de datos, se ha mejorado la confiabilidad y precisión del sistema.

En última instancia, este proyecto demuestra cómo la combinación de componentes electrónicos, programación y pruebas rigurosas puede conducir al desarrollo de un sistema de estacionamiento automatizado eficiente y confiable. El sistema resultante ofrece beneficios significativos, como una mejor gestión de los espacios de estacionamiento, una mayor comodidad para los usuarios y una mayor eficiencia en la asignación de los espacios disponibles.

Aunque este proyecto ha logrado crear un sistema funcional, siempre hay espacio para la mejora continua. En futuras iteraciones, se podrían explorar mejoras adicionales, como la integración de tecnologías avanzadas de detección, como cámaras o sensores ultrasónicos, para mejorar aún más la precisión y confiabilidad del sistema.

En resumen, el proyecto de implementación del sistema de estacionamiento automatizado utilizando Arduino ha sido un proceso desafiante pero gratificante. A través de la resolución de problemas, la creatividad y el enfoque en la calidad, se ha logrado desarrollar un sistema que muestra el potencial de la tecnología para mejorar la gestión de los espacios de estacionamiento. Este proyecto sienta las bases para futuros proyectos en el campo de los sistemas y ofrece una experiencia enriquecedora en términos de aprendizaje y aplicación de conceptos técnicos.

REFERENCIAS Y FUENTES DE INFORMACION



Incluye las referencias correspondientes para dar crédito a las fuentes consultadas.

- <https://arduino.cl>
- https://es.wikipedia.org/wiki/Placa_de_pruebas
- <https://aprendiendoarduino.wordpress.com/2015/03/26/lenguaje-de-programacion-de-arduino-estructura-de-un-programa/>
- <https://arduino-spain.site/lenguaje-arduino/>
- <https://urany.net/blog/conoce-el-funcionamiento-de-los-servomotores>
- https://www.youtube.com/watch?v=Kmkt_QPxDp4
- <https://zather94.files.wordpress.com/2015/04/estacionamiento-automatizado-con-arduino.pdf>
- <https://www.youtube.com/watch?v=N08mgQHrbXw>