

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS (DECANA DE AMÉRICA)
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA ESCUELA
PROFESIONAL DE INGENIERÍA DE SOFTWARE



“Mini Casa inteligente con aplicativo por bluetooth”

Curso: Internet de las Cosas

Equipo: 8

Integrantes

- Román Céspedes, Samuel Aarón
- Martinez Santos, Luz Cristina

LIMA - PERÚ

2024

Resumen	3
1. Introducción	4
1.1. Estado del Arte	4
Título: Diseño e Implementación de un Sistema de Automatización para el Hogar Basado en Arduino y Módulo Bluetooth	4
2. Título: Implementación de un Sistema de Seguridad para el Hogar Usando Sensores y Arduino	4
3. Título: Sistema de Monitoreo de Temperatura y Humedad para el Hogar Basado en Arduino y Aplicación Móvil	4
4. Título: Control de Iluminación Inteligente para el Hogar Usando Arduino y Bluetooth	5
1.2. Planteamiento del Problema	5
1.3. ODS enfocado al Proyecto	6
1.4. Objetivos	6
2. Marco teórico	6
2.1. Domótica	6
2.2. Arduino	7
2.3. Base de Datos	8
2.4. Docker	8
2.5. IOT	8
2.6. Sensores	8
3. Dispositivos y Sensores	8
3.1. Arduino UNO R3 con Cable	8
3.2. Sensor DHT11	9
3.3. Luz Led	9
3.4. Resistencia	10
3.5. Teclado Matricial 4x4	10
3.6. Módulo Bluetooth HC-05	11
3.7. Micro Servo SG90	11
3.8. Protoboard	12
4. Implementación del sistema	12
5. Resultados	25
6. Conclusiones	33
7. Bibliografía	33

Resumen

El proyecto trata sobre el diseño e implementación de un sistema de seguridad y domótico para el hogar. Se centra en desarrollar un sistema integral de automatización y seguridad para una casa utilizando tecnologías como Arduino, el módulo Bluetooth HC-05 y una aplicación Android llamada Arduino Bluetooth Controller. Este sistema permite el control remoto de diversas funciones domésticas desde un dispositivo móvil mediante comandos Bluetooth. Las características principales del proyecto incluyen el control remoto de iluminación, un sistema de seguridad para la entrada principal, monitoreo de temperatura para la detección temprana de incendios con alarmas preventivas, gestión de estacionamiento, control de acceso a través de la puerta principal mediante Bluetooth o contraseña, y un sistema de alarma que se activa ante intentos fallidos de acceso o detección de incendios.

1. Introducción

1.1. Estado del Arte

Diseño e Implementación de un Sistema de Automatización para el Hogar Basado en Arduino y Módulo Bluetooth

Autor: Pérez Gómez, J. L. (2019).

Objetivos: Diseñar e implementar un sistema de automatización del hogar utilizando un microcontrolador Arduino y un módulo Bluetooth para permitir el control remoto de dispositivos eléctricos mediante una aplicación móvil.

Funcionamiento: El sistema utiliza un Arduino Uno como controlador central, conectado a un módulo Bluetooth HC-05 para recibir comandos desde una aplicación Android. Los dispositivos eléctricos (luces, ventiladores, etc.) están conectados al Arduino a través de relés, permitiendo su control remoto.

Conclusiones: El sistema desarrollado demostró ser efectivo y fiable, permitiendo el control remoto de dispositivos eléctricos en el hogar. Sin embargo, se recomendó mejorar la seguridad de la comunicación Bluetooth y la interfaz de usuario de la aplicación móvil para una experiencia más intuitiva.

Implementación de un Sistema de Seguridad para el Hogar Usando Sensores y Arduino

Autor: Ramírez Sánchez, M. A. (2018).

Objetivos: Desarrollar un sistema de seguridad para el hogar que utilice sensores de movimiento y puertas, controlado por un Arduino, para monitorear y notificar actividades sospechosas.

Funcionamiento: El sistema incluye sensores de movimiento y puertas conectados a un Arduino. Cuando se detecta una intrusión, el sistema activa una alarma y envía una notificación al usuario a través de un módulo GSM.

Conclusiones: El sistema fue capaz de detectar intrusiones y notificar al usuario en tiempo real, mostrando una alta eficacia en pruebas controladas. Se sugirió mejorar la autonomía del sistema mediante el uso de baterías recargables y optimizar la comunicación GSM para reducir costos.

Sistema de Monitoreo de Temperatura y Humedad para el Hogar Basado en Arduino y Aplicación Móvil

Autor: López Martínez, R. (2020).

Objetivos: Crear un sistema de monitoreo que permita medir y registrar la temperatura y humedad en el hogar, enviando alertas al usuario mediante una aplicación móvil en caso de valores fuera del rango establecido.

Funcionamiento: El sistema utiliza sensores DHT11 conectados a un Arduino para medir temperatura y humedad. Los datos son enviados a una aplicación móvil a través de un módulo WiFi ESP8266, permitiendo al usuario

monitorear las condiciones del hogar en tiempo real.

Conclusiones: El sistema demostró ser preciso en la medición de temperatura y humedad, y la aplicación móvil fue efectiva en la notificación de alertas. Se recomendó integrar más sensores para una cobertura más amplia y mejorar la interfaz de usuario de la aplicación.

Control de Iluminación Inteligente para el Hogar Usando Arduino y Bluetooth

Autor: García Rodríguez, F. (2017).

Objetivos: Implementar un sistema de control de iluminación que permita encender y apagar luces del hogar de manera remota utilizando un Arduino y un módulo Bluetooth.

Funcionamiento: El sistema consiste en un Arduino Uno conectado a un módulo Bluetooth HC-05, que recibe comandos de una aplicación móvil para controlar el encendido y apagado de luces a través de relés.

Conclusiones: El sistema fue efectivo en el control remoto de la iluminación del hogar, proporcionando una solución simple y económica. Se recomendó implementar medidas de seguridad adicionales para proteger la comunicación Bluetooth y mejorar la durabilidad de los componentes utilizados.

1.2. Planteamiento del Problema

En el contexto actual, la dinámica productiva y las necesidades personales enfrentan desafíos crecientes de eficiencia, velocidad y accesibilidad. La capacidad de realizar actividades de manera eficiente mediante herramientas remotas se ha convertido en crucial tanto para la competitividad como para mejorar la calidad de vida. La automatización de espacios, impulsada por tecnologías de la información y las telecomunicaciones, se presenta como una necesidad inminente.

Una de las áreas críticas afectadas por estas dinámicas es el control de acceso en entornos residenciales con altos estándares de seguridad. Este desafío no solo afecta a instituciones, sino también a individuos como adultos mayores que viven solos, personas con discapacidades que buscan independencia y aquellos que buscan optimizar su tiempo en un mundo acelerado.

En la actualidad, tareas cotidianas como abrir y cerrar puertas se convierten en problemas significativos debido a preocupaciones de seguridad y la necesidad de eficiencia. La implementación de sistemas de domótica en el hogar no solo facilita estas tareas, sino que también mejora la seguridad general del entorno residencial.

A pesar de la disponibilidad de numerosas aplicaciones avanzadas en el mercado, la mayoría de ellas están diseñadas para ofrecer una experiencia de lujo y tienen costos elevados, lo que limita su accesibilidad para el público en general. Esto crea una brecha significativa entre las soluciones disponibles y las necesidades prácticas de las personas que buscan simplemente facilitar actividades cotidianas de manera efectiva y asequible.

En este contexto, surge la necesidad de desarrollar soluciones accesibles y eficientes que integren tecnologías como Arduino, Bluetooth y aplicaciones móviles para ofrecer sistemas de automatización y seguridad en el hogar que sean tanto efectivos como accesibles para una amplia gama de usuarios.

1.3. ODS enfocado al Proyecto

ODS 7 - Energía Asequible y No Contaminante: Al implementar sistemas de control de iluminación eficientes, se promueve el uso responsable de la energía.

ODS 11 - Ciudades y Comunidades Sostenibles: Mejorar la seguridad y la eficiencia energética de las viviendas contribuye a la creación de comunidades más seguras y sostenibles.

1.4. Objetivos

General

Diseñar e implementar un sistema de automatización y seguridad para el hogar utilizando Arduino IDE y un módulo Bluetooth HC-05, controlado a través de una aplicación Android llamada Arduino Bluetooth Controller. Este sistema permitirá el control remoto de diversas funciones domésticas desde un dispositivo móvil.

Específicos

- Implementar el sistema domótico con la configuración hardware necesaria, incluyendo sensores y actuadores compatibles con la plataforma Arduino.
- Garantizar la seguridad del entorno residencial mediante la integración de medidas de control y monitoreo.
- Desarrollar un sistema para la automatización del acceso, permitiendo la apertura y cierre remoto de puertas desde la aplicación móvil.

- Diseñar y desarrollar controles específicos en la aplicación Android, optimizados para la comunicación vía Bluetooth con el sistema Arduino.

Este enfoque busca no solo mejorar la eficiencia en la gestión del hogar, sino también proporcionar una solución accesible y efectiva para la automatización y seguridad residencial.

2. Marco teórico

2.1. Domótica

La palabra "domótica" se origina de la combinación de "domus" (casa en latín) y "tica" (derivado de "automática" en griego, que significa "que funciona por sí sola"). Por lo tanto, el término "domótica" se define como la automatización de una casa o vivienda. La domótica engloba sistemas diseñados para controlar y automatizar diversos aspectos de una residencia, proporcionando servicios como gestión energética, seguridad, confort y comunicación. Entre los elementos que pueden ser controlados en una casa domótica se encuentran la iluminación, la climatización, puertas, ventanas, persianas, toldos, electrodomésticos, así como el suministro de agua, gas y electricidad. En términos generales, prácticamente cualquier dispositivo puede ser gestionado mediante domótica, sin que exista un límite establecido. Los sistemas domóticos típicamente incluyen los siguientes tipos de elementos:

Controladores: Dispositivos responsables de la gestión de la información de otros elementos del sistema. Estos también incluyen las interfaces necesarias para la interacción con el usuario u otras aplicaciones.

Actuadores: Dispositivos que ejecutan las órdenes del controlador, llevando a cabo acciones específicas como el accionamiento de motores de puertas o el encendido y apagado de luces.

Sensores: Dispositivos diseñados para recoger información del entorno y transmitirla al controlador para que este pueda tomar las medidas apropiadas. Estos pueden ser sensores de presencia, temperatura, interruptores, o controles remotos.

2.2. Arduino

Arduino es una plataforma de prototipos electrónicos de código abierto que permite programar pulsos eléctricos y está basada en hardware y software flexibles y fáciles de usar. Diseñada para artistas, diseñadores, aficionados y cualquier persona interesada en crear objetos o entornos interactivos, Arduino puede captar información del entorno mediante sensores y controlar luces, motores y otros dispositivos. El microcontrolador se programa utilizando el “Arduino Programming Language” y el “Arduino Development

Environment". Los proyectos pueden ser autónomos o comunicarse con software en un ordenador. Las placas pueden ser ensambladas manualmente o compradas preensambladas, y el software es gratuito. Además, los diseños de hardware están disponibles bajo licencia open-source, permitiendo su adaptación según las necesidades del usuario. Arduino ha sido reconocida con una mención honorífica en el Ars Electronica Prix en 2006 (David Kushner, 2011).

2.3. Base de Datos

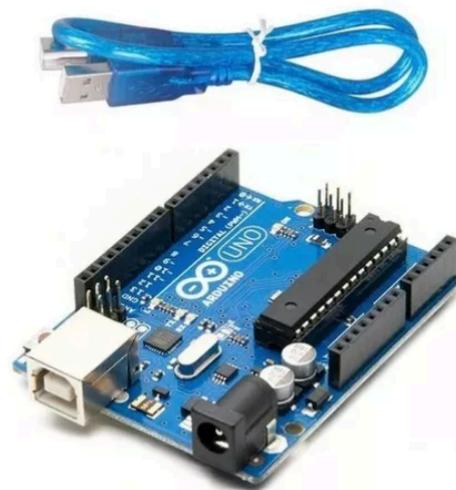
Una base de datos es un sistema organizado para almacenar y gestionar información de manera estructurada, permitiendo su fácil acceso, gestión y actualización. En el contexto de IoT (Internet de las Cosas), las bases de datos juegan un papel fundamental al almacenar y procesar grandes volúmenes de datos generados por dispositivos conectados.

2.4. IOT

oT se refiere a la interconexión de dispositivos físicos (sensores, actuadores, dispositivos móviles, etc.) a través de internet, permitiendo la recopilación y el intercambio de datos. Esta interconexión facilita la automatización de procesos y la mejora de la eficiencia operativa en diversos sectores, desde el hogar inteligente hasta la industria manufacturera.

2.5. Dispositivos y Sensores

2.5.1. Arduino UNO R3 con Cable

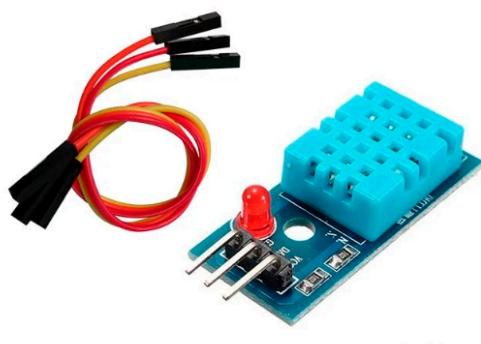


Descripción: Es una placa de microcontrolador basada en el ATmega328P. Tiene 14 pines digitales de entrada/salida (de los cuales 6 pueden usarse como salidas PWM), 6 entradas analógicas, un cristal

oscilador de 16 MHz, una conexión USB, un conector de alimentación, un cabezal ICSP y un botón de reinicio.

Uso: Se conecta a una computadora con un cable USB o se alimenta con un adaptador AC-DC o una batería para comenzar a funcionar. Es ideal para proyectos de electrónica y prototipos rápidos.

2.5.2. Sensor DHT11



Descripción: Es un sensor de temperatura y humedad digital de bajo costo que proporciona datos calibrados de salida digital.

Uso: Ideal para aplicaciones de monitoreo ambiental y de control climático. Se conecta fácilmente a una placa Arduino para leer la temperatura y humedad del ambiente.

2.5.3. Luz Led



Descripción: Un diodo emisor de luz (LED, por sus siglas en inglés) es un componente semiconductor que emite luz cuando se aplica una corriente eléctrica.

Uso: Utilizado en una variedad de aplicaciones, desde indicadores simples hasta iluminación ambiental y pantallas. En proyectos de Arduino, se usa comúnmente para señales visuales

2.5.4. Resistencia



Descripción: Componente electrónico que limita la cantidad de corriente que puede pasar a través de un circuito.

Uso: Protege otros componentes electrónicos, ajusta niveles de voltaje y corriente en circuitos, y se usa en combinación con LEDs y otros componentes.

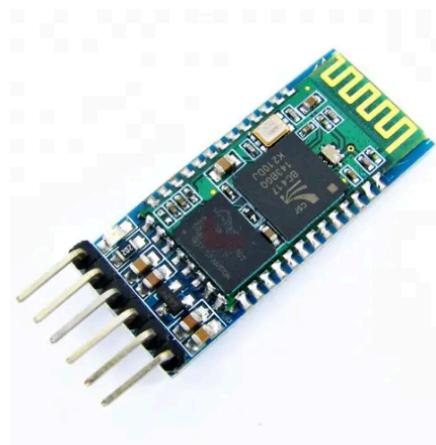
2.5.5. Teclado Matricial 4x4



Descripción: Es una matriz de botones dispuestos en filas y columnas que se usa para introducir números.

Uso en el Programa: Se utiliza para la entrada de datos en proyectos electrónicos. Con Arduino, permite capturar la entrada del usuario, como códigos PIN o menús de selección

2.5.6. Módulo Bluetooth HC-05



Descripción: Es un módulo de comunicación Bluetooth serial, diseñado para proporcionar comunicación inalámbrica de bajo costo entre dispositivos.

Uso: Facilita la comunicación inalámbrica entre Arduino y otros dispositivos como teléfonos móviles o computadoras para transmitir y recibir datos.

2.5.7. Micro Servo SG90



Descripción: Es un pequeño servomotor que puede girar hasta 180 grados. Tiene tres cables: uno para la señal de control, uno para la alimentación y otro para tierra.

Uso: Se puede controlar la posición y movimiento en Arduino. Se puede usar para mover partes de robots o cualquier aplicación que requiera un movimiento preciso y controlado.

2.5.8. Protoboard



Descripción: Una placa de pruebas (protoboard) es una herramienta para crear prototipos de circuitos electrónicos sin la necesidad de soldar.

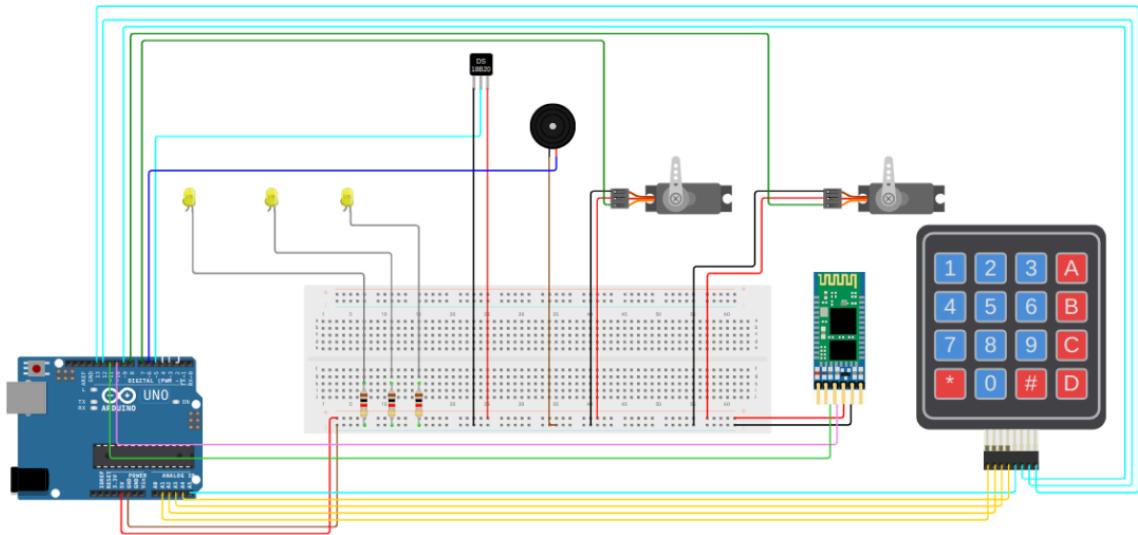
Uso: Permite conectar componentes y hacer modificaciones en los circuitos de manera fácil y rápida. Es una herramienta esencial para experimentar y desarrollar proyectos con Arduino.

3. Implementación del sistema

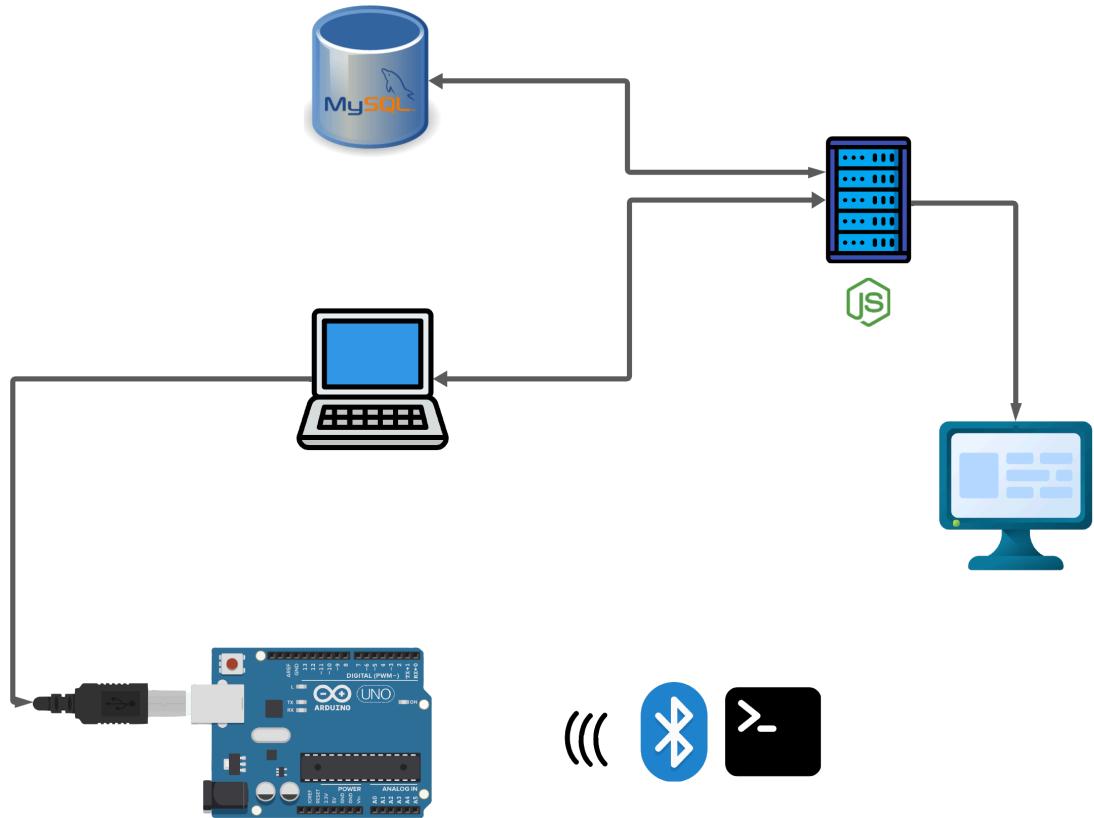
Para la implementación del sistema, utilizamos el entorno de desarrollo Arduino IDE, el cual nos permitió implementar la lógica que subimos a nuestro microcontrolador Arduino. Además, creamos una base de datos MySQL para almacenar la información necesaria para el monitoreo de la casa inteligente. Esta información se guarda a través de un servidor montado en Node.js, que recibe los datos del microcontrolador y luego los registra en la base de datos. Una vez almacenada la información, cualquier interfaz gráfica puede consumir nuestra API en Node.js para obtener y mostrar los datos a los usuarios.

3.1. Diagrama de conexión y estructura

3.1.1. Diagrama de conexión Arduino



3.1.2. Arquitectura de la implementación



3.2. Código arduino

Librerías a utilizar

```
1 #include <SoftwareSerial.h>
2 #include <DHT.h>
3 #include <Keypad.h>
4 #include <Servo.h>
5
```

Utilizamos librerías tanto para el manejo de la recepción y envío de señales bluetooth con nuestro sensor HC-05, para captar la temperatura y humedad, otra para el manejo del teclado para ingresar la contraseña de la puerta y también otra que controla el abrir y cerrar del servomotor.

```
6 #define DHTPIN 5      // Pin donde está conectado el sensor DHT
7 #define DHTTYPE DHT11 // Tipo de sensor DHT (DHT11 o DHT22)
8 #define BUZZER_PIN 6 // Pin donde está conectado el buzzer
9
10 SoftwareSerial BTSerial(10, 11); // RX, TX
11
12 const int ledPin1 = 2; // Pin donde está conectado el primer LED
13 const int ledPin2 = 3; // Pin donde está conectado el segundo LED
14 const int ledPin3 = 4; // Pin donde está conectado el tercer LED
15
16 DHT dht(DHTPIN, DHTTYPE);
17 Servo myServo; // Objeto del servo para la tranquera
18 Servo doorServo; // Objeto del servo para la puerta
19
```

Se configura pines para un sensor DHT11, un buzzer y LEDs, y establece comunicación Bluetooth. También inicializa objetos para controlar un servo de una tranquera y otro de una puerta

```
20 // Configuración del teclado numérico
21 const byte ROWS = 4;
22 const byte COLS = 4;
23 char keys[ROWS][COLS] = {
24     {'1','2','3','A'},
25     {'4','5','6','B'},
26     {'7','8','9','C'},
27     {'*','0','#','D'}
28 };
29 byte rowPins[ROWS] = {A1, A2, A3, A4};
30 byte colPins[COLS] = {A5, 9, 12, 13};
31 Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
32
```

Se configura un teclado numérico de 4x4 en Arduino, especificando los caracteres de las teclas y los pines de conexión para las filas y columnas del teclado.

```
34 // Contraseña esperada
35 String password = "1234";
36 String inputPassword = ""; // Variable para la contraseña ingresada
37 int incorrectAttempts = 0; // Contador de intentos incorrectos
38
39
40 unsigned long lastPrintTime = 0; // Variable para almacenar el último tiempo de impresión
41 const unsigned long printInterval = 4000; // Intervalo de impresión en milisegundos
42
```

Se establece una contraseña esperada, registra intentos incorrectos de ingreso, y gestiona intervalos de impresión en milisegundos.

```
42 void setup() {
43     pinMode(ledPin1, OUTPUT); // Configurar el pin del primer LED como salida
44     pinMode(ledPin2, OUTPUT); // Configurar el pin del segundo LED como salida
45     pinMode(ledPin3, OUTPUT); // Configurar el pin del tercer LED como salida
46     pinMode(BUZZER_PIN, OUTPUT); // Configurar el pin del buzzer como salida
47
48
49     BTSerial.begin(9600); // Inicializar comunicación con el módulo Bluetooth
50     Serial.begin(9600); // Inicializar comunicación serial para depuración
51
52     dht.begin(); // Inicializar el sensor DHT
53     myServo.attach(8); // Pin de control del servo SG90 para la tranquera
54     doorServo.attach(7); // Pin de control del servo SG90 para la puerta
55 }
56
```

En `setup()`, se configuran los pines para controlar LEDs y un buzzer, se inicializa la comunicación serial para Bluetooth y depuración, se inicia el sensor DHT y se adjuntan dos servos para controlar una tranquera y una puerta.

```
151 // Función para abrir la puerta
152 void openDoor() {
153     doorServo.write(90); // Ángulo de 90 grados (posición abierta)
154     Serial.println("Puerta abierta");
155     delay(5000); // Mantener la puerta abierta por 5 segundos (ajusta el tiempo según tus necesidades)
156     closeDoor(); // Cerrar la puerta después del tiempo
157     sendEventToServer("Puerta cerrada.");
158 }
159
```

La función `openDoor()` mueve un servomotor conectado a la puerta a 90 grados para abrirla, imprime "Puerta abierta" por la comunicación serial, espera 5 segundos usando `delay(5000)`, luego cierra la puerta llamando a `closeDoor()` y envía un mensaje al servidor indicando que la puerta se cerró.

```
159 // Función para cerrar la puerta
160 void closeDoor() {
161     doorServo.write(0); // Ángulo de 0 grados (posición cerrada)
162     Serial.println("Puerta cerrada");
163 }
164
165
```

La función `closeDoor()` mueve el servo conectado a la puerta a un ángulo de 0 grados para cerrarla y luego imprime "Puerta cerrada" a través de la comunicación serial.

```
165 // Función para abrir la tranquera
166 void openGate() {
167     myServo.write(90); // Ángulo de 90 grados (posición abierta)
168     Serial.println("Tranquera abierta");
169 }
170
171
```

La función openGate() mueve el servo conectado a la tranquera a un ángulo de 90 grados para abrirlo, y luego imprime "Tranquera abierta" a través de la comunicación serial.

```
171
172 // Función para cerrar la tranquera
173 void closeGate() {
174     myServo.write(0); // Ángulo de 0 grados (posición cerrada)
175     Serial.println("Tranquera cerrada");
176 }
```

La función closeGate() mueve el servo conectado a la tranquera a un ángulo de 0 grados para cerrarlo, y luego imprime "Tranquera cerrada" a través de la comunicación serial.

```
177
178 // Función para enviar eventos al servidor
179 void sendEventToServer(String event) {
180     Serial.println(event);
181 }
```

La función sendEventToServer(String event) recibe el evento que mandaremos a ser procesado en el servidor Node.js

```

127 // Función para verificar la contraseña ingresada
128 void checkPassword() {
129     if (inputPassword == password) {
130         BTSerial.println("Contraseña correcta!");
131         Serial.println("Contraseña correcta!");
132         openDoor(); // Abrir la puerta al ingresar la contraseña correcta
133         sendEventToServer("Contraseña correcta. Puerta abierta.");
134         inputPassword = ""; // Reiniciar la variable de entrada de contraseña
135         incorrectAttempts = 0; // Reiniciar el contador de intentos incorrectos
136     } else {
137         BTSerial.println("Alerta: Contraseña incorrecta! Posible robo detectado.");
138         Serial.println("Alerta: Contraseña incorrecta! Posible robo detectado.");
139         sendEventToServer("Contraseña incorrecta. Posible robo detectado.");
140         incorrectAttempts++; // Incrementar el contador de intentos incorrectos
141         if (incorrectAttempts >= 3) {
142             digitalWrite(BUZZER_PIN, HIGH); // Activar buzzer
143             delay(5000); // Mantener el buzzer activado por 5 segundos
144             digitalWrite(BUZZER_PIN, LOW); // Desactivar buzzer
145             sendEventToServer("Alarma activada por intentos incorrectos.");
146         }
147         inputPassword = ""; // Reiniciar la variable de entrada de contraseña
148     }
149 }
```

La función `checkPassword()` verifica si la contraseña ingresada coincide con la contraseña esperada. Si es correcta, se abre la puerta, se reinician las variables y se envían mensajes indicando el éxito. En caso contrario, se registra un intento incorrecto, se activa una alerta si hay múltiples intentos fallidos y se reinicia la entrada de contraseña.

Contenido del loop

```

58     unsigned long currentTime = millis(); // Obtener el tiempo actual
59
60     // Leer datos del sensor DHT
61     float humidity = dht.readHumidity();
62     float temperature = dht.readTemperature();
63
64     // Comprobar si hay fallo en la lectura
65     if (isnan(humidity) || isnan(temperature)) {
66         Serial.println("Error leyendo del sensor DHT");
67         return;
68     }
```

Se obtiene el tiempo actual, lee la humedad y temperatura del sensor DHT, y verifica si hay errores en la lectura de estos valores. Si encuentra un error, imprime un mensaje de error y sale de la función.

```

69
70     // Comprobar si la temperatura excede el umbral de incendio
71     if (temperature > FIRE_TEMP_THRESHOLD) {
72         BTSerial.println("Alerta: Incendio detectado!");
73         Serial.println("Alerta: Incendio detectado!");
74         digitalWrite(BUZZER_PIN, HIGH); // Activar buzzer
75         sendEventToServer("Alerta: Incendio detectado!"); // Enviar alerta al servidor
76     } else {
77         digitalWrite(BUZZER_PIN, LOW); // Desactivar buzzer
78     }
79 }
```

Se comprueba si la temperatura supera un umbral predefinido para detectar un posible incendio. Si es así, emite alertas por Bluetooth y

comunicación serial, activa un buzzer y envía un evento al servidor. Si no se supera el umbral, desactiva el buzzer.

```
80 // Leer comandos Bluetooth
81 if (BTSerial.available()) { // Verificar si hay datos disponibles desde el Bluetooth
82     char command = BTSerial.read(); // Leer el comando recibido
83     Serial.println(command); // Mostrar el comando para depuración
84
85     if (command == '1') { // Si el comando es '1'
86         digitalWrite(ledPin1, HIGH); // Prender el primer LED
87         sendEventToServer("LED1 ON");
88     } else if (command == '0') { // Si el comando es '0'
89         digitalWrite(ledPin1, LOW); // Apagar el primer LED
90         sendEventToServer("LED1 OFF");
91     } else if (command == '2') { // Si el comando es '2'
92         digitalWrite(ledPin2, HIGH); // Prender el segundo LED
93         sendEventToServer("LED2 ON");
94     } else if (command == '3') { // Si el comando es '3'
95         digitalWrite(ledPin2, LOW); // Apagar el segundo LED
96         sendEventToServer("LED2 OFF");
97     } else if (command == '4') { // Si el comando es '4'
98         digitalWrite(ledPin3, HIGH); // Prender el tercer LED
99         sendEventToServer("LED3 ON");
100    } else if (command == '5') { // Si el comando es '5'
101        digitalWrite(ledPin3, LOW); // Apagar el tercer LED
102        sendEventToServer("LED3 OFF");
103    } else if (command == '6') { // Si el comando es '6'
104        openGate(); // Abrir la tranquera
105        sendEventToServer("Tranquera abierta");
106    } else if (command == '7') { // Si el comando es '7'
107        closeGate(); // Cerrar la tranquera
108        sendEventToServer("Tranquera cerrada");
109    }
110 }
```

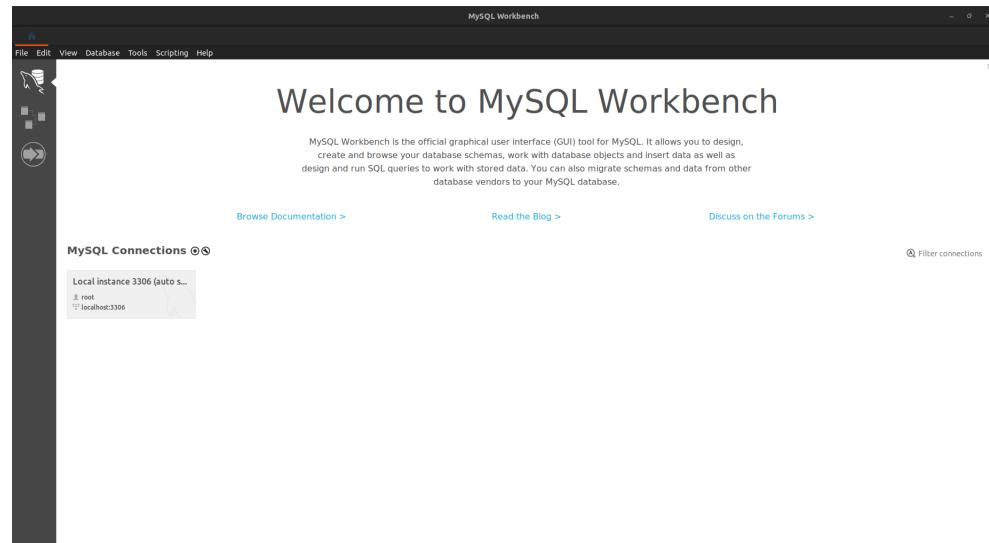
Se verifica si hay datos disponibles desde el módulo Bluetooth. Si se detecta un comando válido ('1' a '7'), se realiza la acción correspondiente: encender o apagar LEDs, abrir o cerrar la tranquera, y se envía un evento al servidor según la acción realizada.

```
112 // Leer entrada del teclado numérico
113 char key = keypad.getKey();
114
115 if (key != NO_KEY) {
116     // Procesar la entrada del teclado
117     if (isDigit(key)) {
118         inputPassword += key; // Concatenar el dígito a la contraseña ingresada
119     } else if (key == '#') {
120         checkPassword(); // Verificar la contraseña al presionar #
121     }
122 }
```

Se lee la entrada del teclado numérico y, si se detecta una tecla válida (dígito o '#'), realiza la acción correspondiente: concatena el dígito a la variable inputPassword o verifica la contraseña llamando a checkPassword().

3.3. Base de datos MySQL

Para la base de datos MySQL utilizamos el gestor Workbench el cuál nos permite realizar las consultas y visualizar nuestros datos de una manera sencilla.



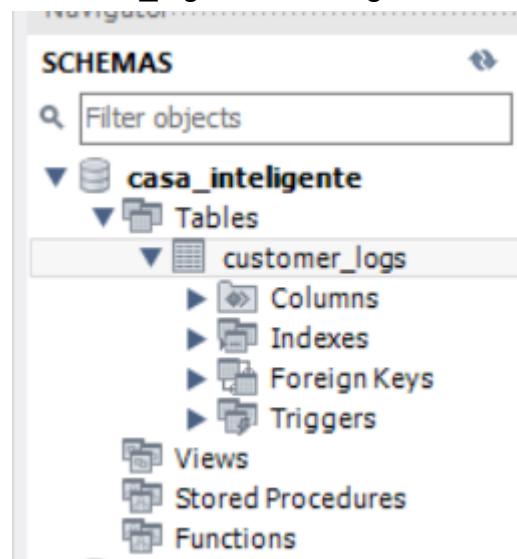
En nuestro gestor Workbench podemos visualizar y/o modificar los datos mediante cualquier consulta SQL, en el siguiente ejemplo se ven los resultados al hacer un select simple a la tabla customer_logs.

A screenshot of the MySQL Workbench interface showing a query results grid. The top navigation bar includes "File", "Edit", "View", "Query", "Database", "Server", "Tools", "Scripting", and "Help". The "Query" tab is active. The central pane shows a query editor with the following SQL code:

```
1 * use casa_inteligente;
2 *
3 * SELECT*FROM customer_logs;
```

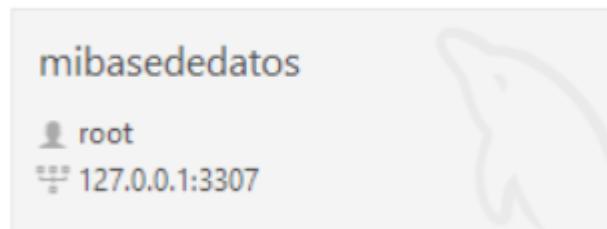
The results grid below displays 20 rows of data from the "customer_logs" table. The columns are "id", "hora" (Time), and "descripcion". The data shows various log entries such as "Transpare cerrada", "Transpare cerrada", "Transpare cerrada", etc. The bottom pane shows the "Action Output" section with a table of log entries corresponding to the SQL execution.

Nuestra base de datos se llamará casa_inteligente y tendrá una tabla customer_logs donde se guarda la información.



Utilizamos asimismo para la conexión la base de datos establecida en el puerto 3307 de nuestra máquina como se muestra en la siguiente imagen.

MySQL Connections ⊕ ✖



3.4. Servidor montado con Node.js

Utilizamos Node.js para montar un servidor local el cual nos permitirá intermediar entre la base de datos con nuestro Arduino, además de servir de API para la consulta de la información registrada.

```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const { SerialPort } = require('serialport');
4 const { ReadlineParser } = require('@serialport/parser-readline');
5 const mysql = require('mysql');
6
```

En este caso usamos los paquetes respectivos tanto para montar el server, permitir parsear la información que usaremos al servir la información, lograr la conexión con el puerto serial el cual nuestro Arduino va a estar conectado y además la conexión con la base de datos.

```
0
7  const cors = require('cors');
8  const app = express();
9  app.use(cors());
10
11 const appPort = 3000; // Cambiado a
12
```

Habilitamos cors para no tener problemas al utilizar la API y además seteamos el puerto para las llamadas en el 3000.

```
12
13 // Configuraci n de la conexi n MySQL
14 const connection = mysql.createConnection({
15   host: 'localhost',
16   user: 'root',
17   password: '',
18   port: 3307,
19   multipleStatements: true // Permite ejecutar m ltiples sentencias
20 });
21
```

Hacemos las configuraciones necesarias para la conexión con la base de datos MySQL.

```
22 // Conectar y asegurar que la base de datos y la tabla existan
23 connection.connect(err => {
24   if (err) {
25     console.error('Error conectando a MySQL: ' + err.stack);
26     return;
27   }
28   console.log('Conectado a MySQL como ID ' + connection.threadId);
29
30 // Crear la base de datos y la tabla si no existen
31 const createDbAndTableQuery = `
32   CREATE DATABASE IF NOT EXISTS casa_inteligente;
33   USE casa_inteligente;
34   CREATE TABLE IF NOT EXISTS customer_logs (
35     id INT AUTO_INCREMENT PRIMARY KEY,
36     hora DATETIME NOT NULL,
37     descripcion VARCHAR(255) NOT NULL
38   );
39 `;
40
41 connection.query(createDbAndTableQuery, (error, results, fields) => {
42   if (error) {
43     console.error('Error al crear la base de datos o la tabla: ' + error.message);
44     return;
45   }
46   console.log('Base de datos y tabla aseguradas.');
47 });
48
49 
```

Procedemos a conectar la base de datos y ejecutar unas queries tanto para crear la base de datos si no existe y también asegurarnos de crear la tabla donde guardaremos los logs de la casa inteligente.

```
51  const portName = 'COM3';
52
53  const serialPort = new SerialPort({
54    path: portName,
55    baudRate: 9600
56  });
57
58  const parser = serialPort.pipe(new ReadlineParser({ delimiter: '\n' }));
59
```

Definimos la configuración para el puerto serial al cual nos conectaremos, el cual es COM3 donde está nuestro Arduino.

```
59
60  parser.on('data', data => {
61    const receivedData = data.trim(); // Convertir datos recibidos a string
62    const currentTime = new Date().toISOString().slice(0, 19).replace('T', ' '); // Hora
63
64    // Insertar datos en la base de datos
65    const insertQuery = 'INSERT INTO customer_logs (hora, descripcion) VALUES (?, ?)';
66    const values = [currentTime, receivedData];
67
68    connection.query(insertQuery, values, (error, results, fields) => {
69      if (error) {
70        console.error('Error al insertar en la base de datos: ' + error.message);
71        return;
72      }
73      console.log('Registro insertado correctamente.');
74    });
75  });
76
```

Definimos el evento que triggereá al momento de recibir cualquier información en el puerto serial. Esta información enviada por nuestro Arduino es luego almacenada en nuestra base de datos MySQL.

```

75
76
77 app.use(bodyParser.urlencoded({ extended: true }));
78
79 app.post('/insertLog', (req, res) => {
80   const { hora, descripcion } = req.body;
81
82   // Insertar datos en la base de datos a través de la ruta POST
83   const insertQuery = 'INSERT INTO customer_logs (hora, descripcion) VALUES (?, ?)';
84   const values = [hora, descripcion];
85
86   connection.query(insertQuery, values, (error, results, fields) => {
87     if (error) {
88       console.error('Error al insertar en la base de datos: ' + error.message);
89       return;
90     }
91     console.log('Registro insertado correctamente.');
92     res.send('Registro insertado correctamente.');
93   });
94 });
95
96 // Ruta GET para obtener todos los registros de la tabla customer_logs
97 app.get('/getLogs', (req, res) => {
98   const selectQuery = 'SELECT * FROM customer_logs';
99
100  connection.query(selectQuery, (error, results, fields) => {
101    if (error) {
102      console.error('Error al obtener los registros de la base de datos: ' + error.message);
103      res.status(500).send('Error al obtener los registros de la base de datos');
104      return;
105    }
106    res.json(results);
107  });
108 });
109
110 app.listen(appPort, () => {
111   console.log(`Servidor Node.js escuchando en http://localhost:${appPort}`);
112 });
113

```

Seteamos los endpoints que nos servirán para consultar la información desde cualquier cliente (en nuestro caso, una interfaz web).

3.5. Interfaz gráfica para visualización de información

Para una visualización de los datos más agradable, se brinda una interfaz web que procederá a consumir el API que construimos, de esta manera se pueden usar los datos en la renderización de la UI.

```

1  var http = require('http');
2  var fs = require('fs');
3  var path = require('path');
4
5  http.createServer(function (request, response) {
6      console.log('request ', request.url);
7
8      var filePath = '.' + request.url;
9
10     filePath = './index.html';
11
12
13     var extname = String(path.extname(filePath)).toLowerCase();
14     var mimeTypes = {
15         '.html': 'text/html',
16         '.js': 'text/javascript',
17         '.css': 'text/css',
18         '.json': 'application/json',
19         '.png': 'image/png',
20         '.jpg': 'image/jpg',
21         '.gif': 'image/gif',
22         '.svg': 'image/svg+xml',
23         '.wav': 'audio/wav',
24         '.mp4': 'video/mp4',
25         '.woff': 'application/font-woff',
26         '.ttf': 'application/font-ttf',
27         '.eot': 'application/vnd.ms-fontobject',
28         '.otf': 'application/font-otf',
29         '.wasm': 'application/wasm'
30     };
31
32     var contentType = mimeTypes[extname] || 'application/octet-stream';
33
34     fs.readFile(filePath, function(error, content) {
35         if (error) {
36             if(error.code == 'ENOENT') {
37                 fs.readFile('./404.html', function(error, content) {
38                     response.writeHead(404, { 'Content-Type': 'text/html' });
39                     response.end(content, 'utf-8');
40                 });
41             }
42             else {
43                 response.writeHead(500);
44                 response.end('Sorry, check with the site admin for error: '+error.code+'\n');
45             }
46         }
47         else {
48             response.writeHead(200, { 'Content-Type': contentType });
49             response.end(content, 'utf-8');
50         }
51     });
52
53 }).listen(9587);
54 console.log('Server running at http://127.0.0.1:9587/');
55

```

Montamos un servidor simple en javascript con Node.js para renderizar nuestro código html en el navegador en el puerto 9587.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Customer Logs</title>
7    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;500&display=swap" rel="stylesheet">
8  <style>
9    body {
10      font-family: 'Roboto', sans-serif;
11      margin: 0;
12      padding: 0;
13      background: linear-gradient(to bottom right, #dff1e2, #a9c4d9); /* Fondo degradado */
14      color: #333;
15      display: flex;
16      flex-direction: column;
17      align-items: center;
18      justify-content: center;
19      min-height: 100vh;
20    }
21    h1 {
22      color: #333;
23      margin: 20px 0;
24    }
25    table {
26      width: 80%;
27      border-collapse: collapse;
28      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
29      margin: 20px 0;
30      background-color: #fff;
31      border-radius: 8px;
32      overflow: hidden;
33    }
34    th, td {
35      border: 1px solid #ddd;
36      padding: 12px 15px;
37      text-align: left;
38    }
39    th {
40      background-color: #a9c4d9;
41      color: #333;
42      font-weight: 500;
43    }
44    tr:nth-child(even) {
45      background-color: #f9f9f9;
46    }
47    tr:hover {
48      background-color: #f1f1f1;
49    }
50    tbody tr td:first-child {
51      font-weight: bold;
52    }
53    @media (max-width: 768px) {
54      table {
55        width: 100%;
56      }

```

Código html para la página, se aprecian estilos CSS para darle un mejor aspecto a la interfaz.

```

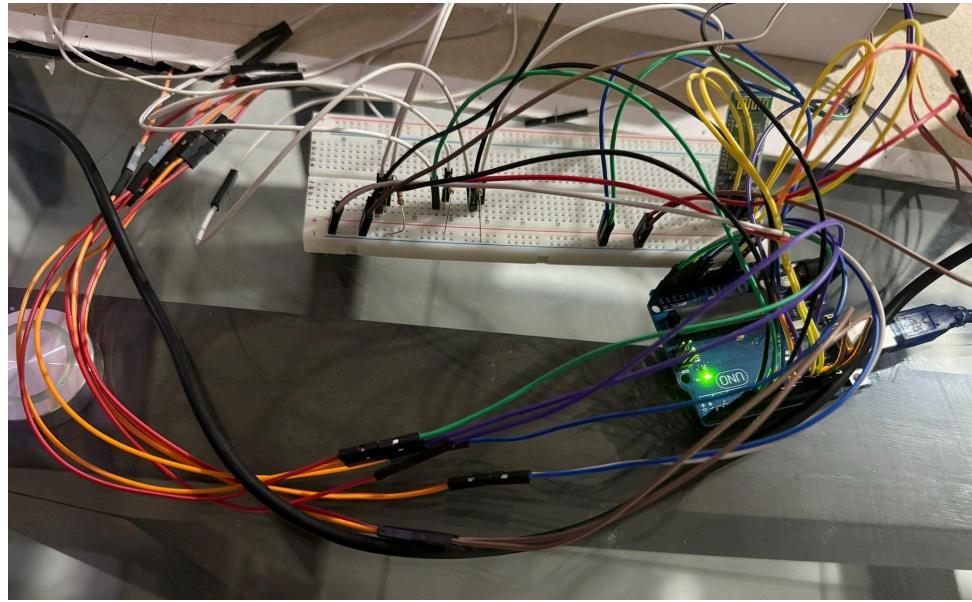
59  </head>
60  <body>
61    <h1>Customer Logs</h1>
62    <table id="logsTable">
63      <thead>
64        <tr>
65          <th>ID</th>
66          <th>Hora</th>
67          <th>Descripción</th>
68        </tr>
69      </thead>
70      <tbody>
71        | <!-- Los datos se insertarán aquí -->
72      </tbody>
73    </table>
74
75    <script>
76      document.addEventListener('DOMContentLoaded', function() {
77        fetch('http://localhost:3000/getLogs')
78          .then(response => response.json())
79          .then(data => {
80            const tableBody = document.querySelector('#logsTable tbody');
81            data.forEach(log => {
82              // Filtra los registros que tienen log.descripcion vacía
83              if (!log.descripcion) {
84                const row = document.createElement('tr');
85                const idCell = document.createElement('td');
86                const horaCell = document.createElement('td');
87                const descripcionCell = document.createElement('td');
88
89                idCell.textContent = log.id;
90                const date = new Date(log.hora);
91                const formattedDate = date.toLocaleDateString();
92                const formattedTime = date.toLocaleTimeString();
93                horaCell.textContent = `${formattedDate} ${formattedTime}`;
94                descripcionCell.textContent = log.descripcion;
95
96                row.appendChild(idCell);
97                row.appendChild(horaCell);
98                row.appendChild(descripcionCell);
99                tableBody.appendChild(row);
100            }
101          });
102        })
103        .catch(error => console.error('Error al obtener los registros:', error));
104      });
105    </script>
106  </body>
107 </html>

```

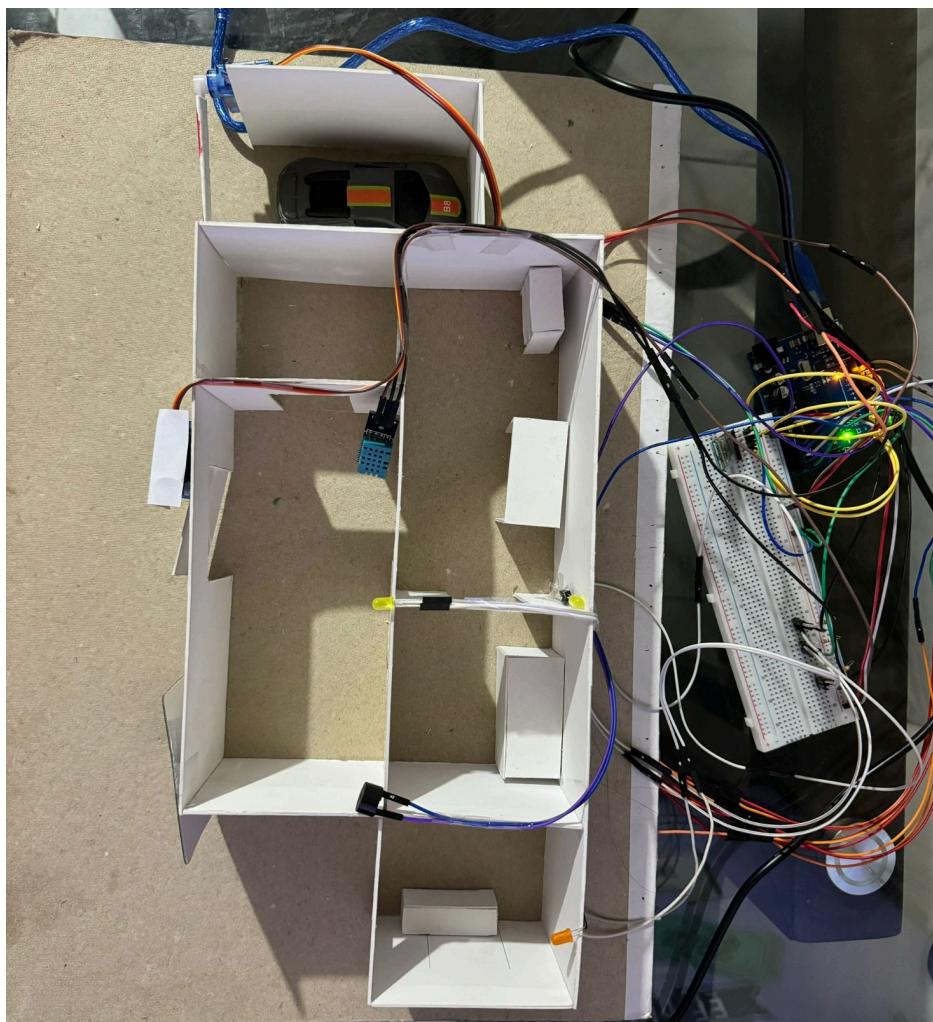
Utilizamos JavaScript para pegarle al endpoint de nuestro API y utilizar la response con los datos para así mostrarlo en la interfaz.

4. Resultados

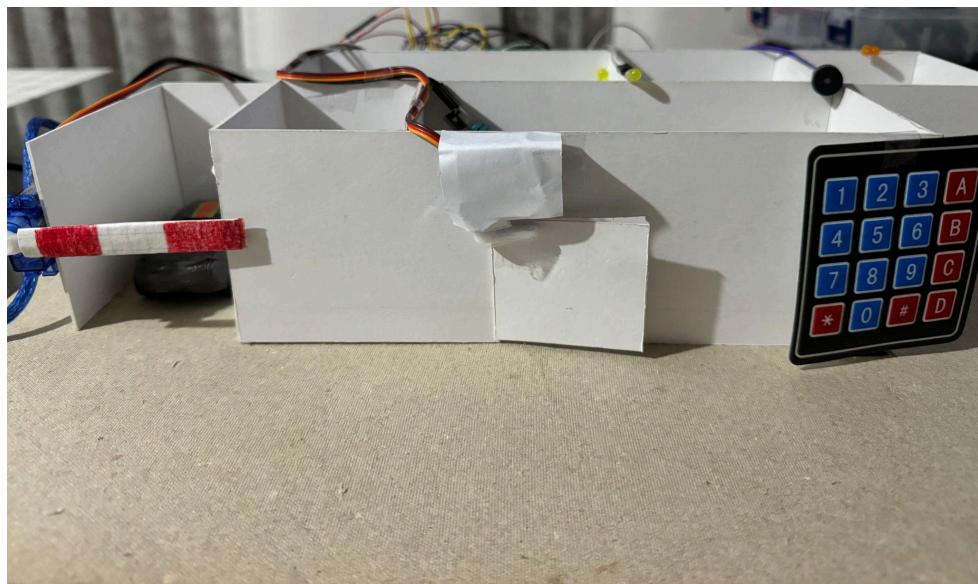
Circuito Arduino con todas las conexiones para la casa inteligente.



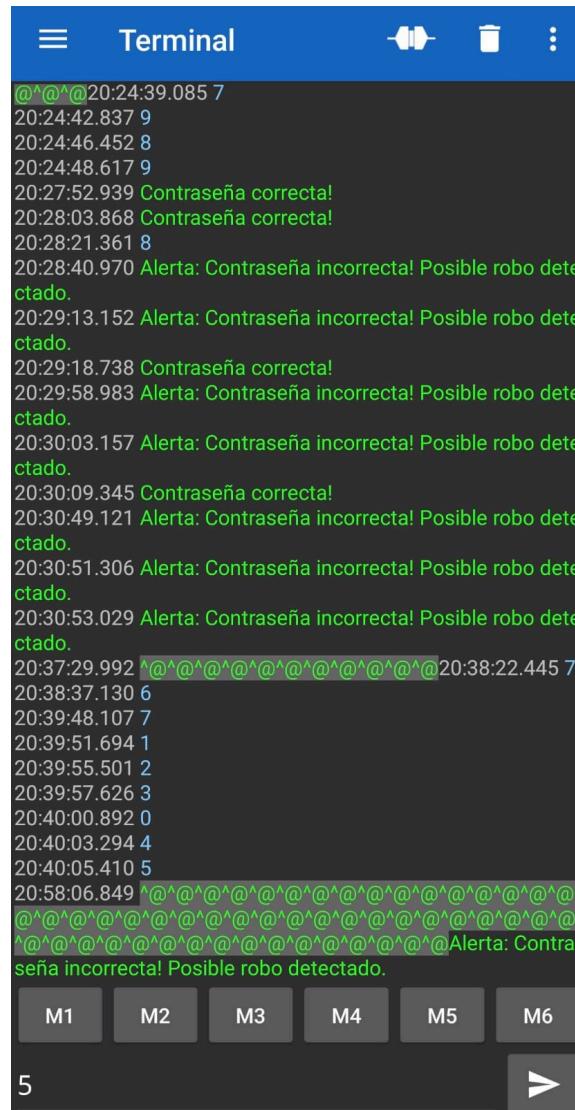
Maqueta de casa inteligente con todos los sensores a la vista que utilizamos.



Vista delantera de la casa inteligente



Uso de terminal serial bluetooth para la comunicación con Arduino



Ejemplo de logging en Monitor Serial de Arduino IDE, en este caso monitoreo de temperatura.

Output Serial Monitor X

Message (Enter to send message to 'Arduino Uno' on 'COM3')

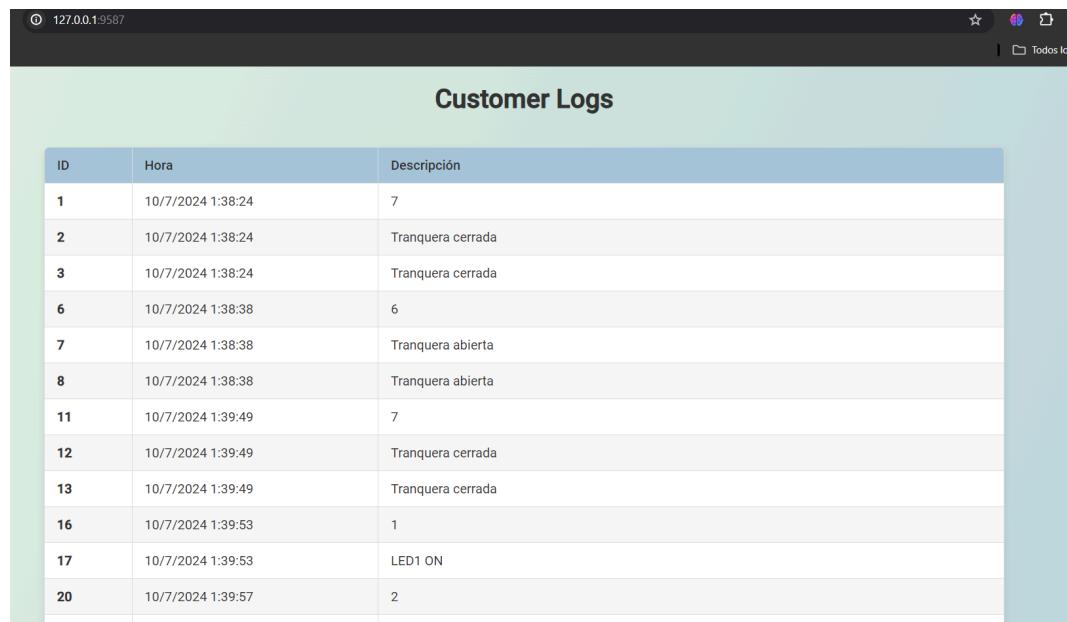
Servidor node.js corriendo, se visualiza en la terminal con éxito las operaciones respectivas a la conexión serial y las inserciones a la base de datos.

Tabla con los registros del monitoreo de información en la casa inteligente.

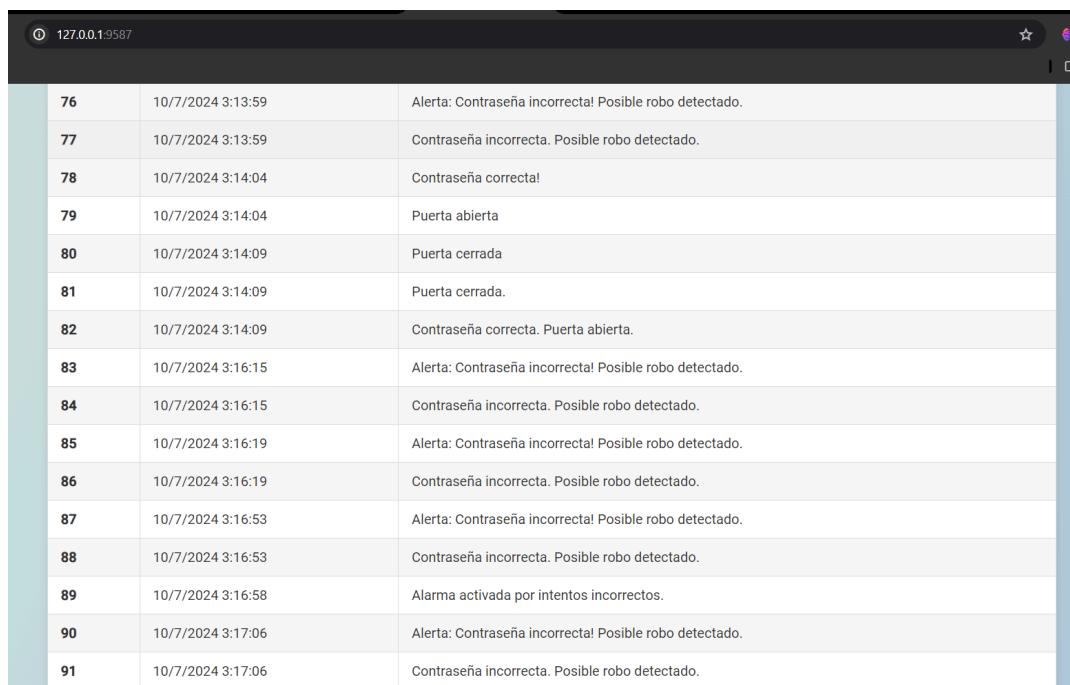
id	hora	descripcion
1	2024-07-10 01:38:24	7
2	2024-07-10 01:38:24	Tranquera cerrada
3	2024-07-10 01:38:24	Tranquera cerrada
6	2024-07-10 01:38:38	6
7	2024-07-10 01:38:38	Tranquera abierta
8	2024-07-10 01:38:38	Tranquera abierta
11	2024-07-10 01:39:49	7
12	2024-07-10 01:39:49	Tranquera cerrada
13	2024-07-10 01:39:49	Tranquera cerrada
16	2024-07-10 01:39:53	1
17	2024-07-10 01:39:53	LED1 ON
20	2024-07-10 01:39:57	2

Servidor para renderizar la interfaz corriendo.

Visualización de interfaz web consumiendo y mostrando la información de la base de datos.



ID	Hora	Descripción
1	10/7/2024 1:38:24	7
2	10/7/2024 1:38:24	Tranquera cerrada
3	10/7/2024 1:38:24	Tranquera cerrada
6	10/7/2024 1:38:38	6
7	10/7/2024 1:38:38	Tranquera abierta
8	10/7/2024 1:38:38	Tranquera abierta
11	10/7/2024 1:39:49	7
12	10/7/2024 1:39:49	Tranquera cerrada
13	10/7/2024 1:39:49	Tranquera cerrada
16	10/7/2024 1:39:53	1
17	10/7/2024 1:39:53	LED1 ON
20	10/7/2024 1:39:57	2



76	10/7/2024 3:13:59	Alerta: Contraseña incorrecta! Posible robo detectado.
77	10/7/2024 3:13:59	Contraseña incorrecta. Posible robo detectado.
78	10/7/2024 3:14:04	Contraseña correcta!
79	10/7/2024 3:14:04	Puerta abierta
80	10/7/2024 3:14:09	Puerta cerrada
81	10/7/2024 3:14:09	Puerta cerrada.
82	10/7/2024 3:14:09	Contraseña correcta. Puerta abierta.
83	10/7/2024 3:16:15	Alerta: Contraseña incorrecta! Posible robo detectado.
84	10/7/2024 3:16:15	Contraseña incorrecta. Posible robo detectado.
85	10/7/2024 3:16:19	Alerta: Contraseña incorrecta! Posible robo detectado.
86	10/7/2024 3:16:19	Contraseña incorrecta. Posible robo detectado.
87	10/7/2024 3:16:53	Alerta: Contraseña incorrecta! Posible robo detectado.
88	10/7/2024 3:16:53	Contraseña incorrecta. Posible robo detectado.
89	10/7/2024 3:16:58	Alarma activada por intentos incorrectos.
90	10/7/2024 3:17:06	Alerta: Contraseña incorrecta! Posible robo detectado.
91	10/7/2024 3:17:06	Contraseña incorrecta. Posible robo detectado.

5. Conclusiones

- El proyecto demuestra una integración efectiva de Arduino, Node.js, Bluetooth, y MySQL para automatizar y monitorear una casa inteligente. La combinación de sensores, actuadores y una base de datos asegura un sistema robusto y eficiente.
- La implementación de comunicación Bluetooth permite la supervisión y el control remoto de la casa inteligente en tiempo real. Esto facilita la interacción con el sistema desde dispositivos móviles, mejorando la conveniencia y accesibilidad para los usuarios, y permitiendo respuestas rápidas a eventos críticos, como detección de incendios o accesos no autorizados.

6. Bibliografía

Pérez Gómez, J. L. (2019). Diseño e Implementación de un Sistema de Automatización para el Hogar Basado en Arduino y Módulo Bluetooth. [Tesis de Licenciatura, Universidad Nacional Autónoma de México].

Ramírez Sánchez, M. A. (2018). Implementación de un Sistema de Seguridad para el Hogar Usando Sensores y Arduino. [Tesis de Licenciatura, Universidad de Guadalajara].

López Martínez, R. (2020). Sistema de Monitoreo de Temperatura y Humedad para el Hogar Basado en Arduino y Aplicación Móvil. [Tesis de Licenciatura, Universidad Politécnica de Madrid].

García Rodríguez, F. (2017). Control de Iluminación Inteligente para el Hogar Usando Arduino y Bluetooth. [Tesis de Licenciatura, Universidad Técnica Federico Santa María].

Barberan Villacampa, F.(2019) “Control Domótico de una Vivienda” Proyecto de la Facultad de Ingeniería. Universidad ROVIRA I VIRGILI De Barcelona.

RUIZ, José Manuel, (2020), Manual de Programación Arduino, México.