Engineering Design II

Smart Door Lock

Team Members:

Daggupati Venkata Venu (B23CM1012)

Devesh Labana (B23CS1015)

Dhruv Mishra (B23EE1016)

Maliwal Raghav Kamleshkumar (B23EE1039)

Sunil (B23ME1072)

Introduction

In the ever-evolving landscape of technology, the integration of Internet of Things (IoT) solutions has revolutionized various aspects of our daily lives. The project titled "IoT Based Smart Door Lock" represents a significant advancement in home security and convenience. By harnessing the power of IoT, this project aims to provide users with seamless control over their door locks using their mobile phones.

Utilizing cutting-edge components such as the ESP8266 microcontroller, solenoid lock, relay module, and a 12V adapter, this smart door lock system offers enhanced security features and remote accessibility. With the proliferation of smartphones, the ability to remotely monitor and control household devices has become increasingly sought after.

In this report, we will delve into the design, implementation, and functionality of our IoT-based smart door lock system. We will explore the components used, the architecture of the system, the methods employed for remote access, and the potential benefits and implications of such a solution in modern residential settings. Through this project, we aim to demonstrate the practical application of IoT technology in enhancing both security and convenience in our daily lives.

Hardware

Hardware requirements for this project are as follows:

NodeMCU

NodeMCU is an open-source IoT platform based on the ESP8266 Wi-Fi module. It provides built-in Wi-Fi connectivity and acts as a gateway for sending data to web services. NodeMCU will be used to connect various sensors and components to the internet and send data to the web service.

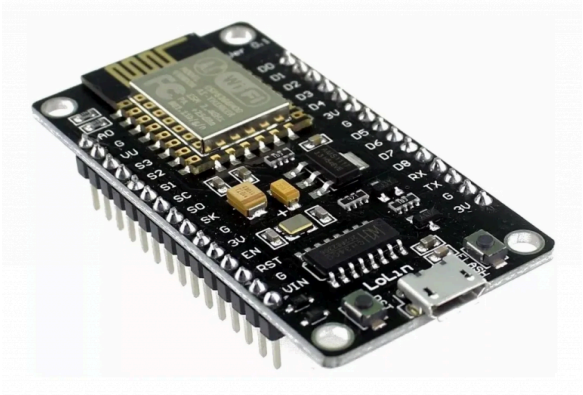


Fig.- NodeMCU ESP8266

Relay Module

A Relay is a device that opens or closes an auxiliary circuit under some predetermined condition in the Main circuit. The object of a Relay is generally to act as a sort of electric magnifier, that is to say, it enables a comparatively weak current to bring into operation on a much stronger current. It also provides complete electrical isolation between the controlling circuit and the controlled circuit. Relays are the switches which aim at closing and opening the circuits electromechanically.

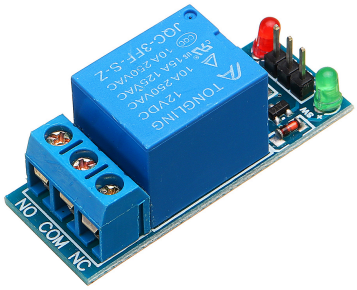


Fig.-Relay Module

Solenoid Lock

The solenoid lock, characterized by its electrical locking mechanism, operates through the application or removal of power, providing a versatile solution for door security systems. In the context of our project, this lock will serve a pivotal role, enabling remote locking and unlocking functionalities in response to user commands. Through seamless integration with our system, it will offer not only convenience but also enhanced security, allowing users to manage access to the door from a distance with ease. This capability ensures that authorized individuals can control entry efficiently while maintaining a robust barrier against unauthorized access.

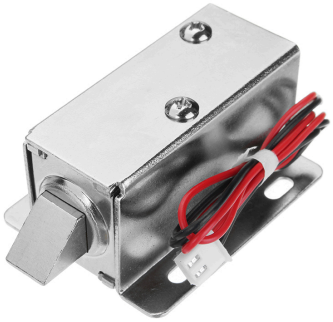


Fig.- Solenoid Lock

Battery

Batteries are portable power sources that provide electrical energy. 11.1V batteries will be used to power the components and ensure their operation even during power outages or when the main power source is unavailable. 11.1 V batteries will be made up from the series combination of three 3.7 v cells.

Jumper Wires

Jumper wires are used to establish connections between different components on a breadboard or circuit. They will be used to connect the solenoid lock, relay module and NodeMCU together in the project.

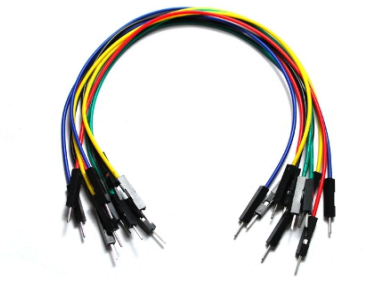


Fig.- Jumper Wires

Software

Software requirements for this project are as follows:

Arduino IDE

The software requirement for this research project includes the use of the open-source Arduino IDE (Integrated Development Environment). Arduino IDE is a software platform that provides a convenient and user-friendly environment for writing, compiling, and uploading code to Arduino microcontroller boards. Arduino IDE is widely used in the field of electronics and embedded systems development. It supports a variety of Arduino boards and provides a simplified programming language based on Wiring, making it accessible even to beginners. The open-source nature of Arduino IDE allows for community contributions and continuous improvement of the software. With Arduino IDE, we can write code in a high-level language, which is then compiled and uploaded to the NodeMCU. The IDE offers features such as syntax highlighting, code suggestions, and a serial monitor for debugging and interacting with the board. It also provides a library manager, allowing users to easily add additional functionality to their projects through third-party libraries. The IDE simplifies the development process, making it easier to write and test code for controlling sensors, actuators, and other components connected to the Arduino boards.

Implementation

1. Problem Statement

Sometimes, house occupants forget to lock the door due to hurry when leaving the house or may doubt whether they have locked the door or not. This poses a threat to home security.

The traditional methods of securing residential properties with mechanical door locks present limitations in terms of convenience and security monitoring. With the increasing reliance on smartphones and the need for enhanced security measures, there arises a demand for innovative solutions that integrate Internet of Things (IoT) technology.

The project aims to address the following challenges:

1. Lack of remote accessibility: Current door lock systems do not offer the ability to remotely monitor and control access to residential properties, leading to inconvenience and security concerns for homeowners.
2. Limited security features: Mechanical door locks lack advanced security features such as activity logs, real-time notifications, and remote locking/unlocking capabilities, leaving properties vulnerable to unauthorized access.
3. Complexity in installation and operation: Existing IoT-based door lock solutions may involve complex installation procedures and operational complexities, making them less accessible to the average homeowner.

The objective of this project is to develop an IoT-based smart door lock system that overcomes these challenges by providing seamless remote access, advanced security features, and user-friendly operation through integration with mobile devices.

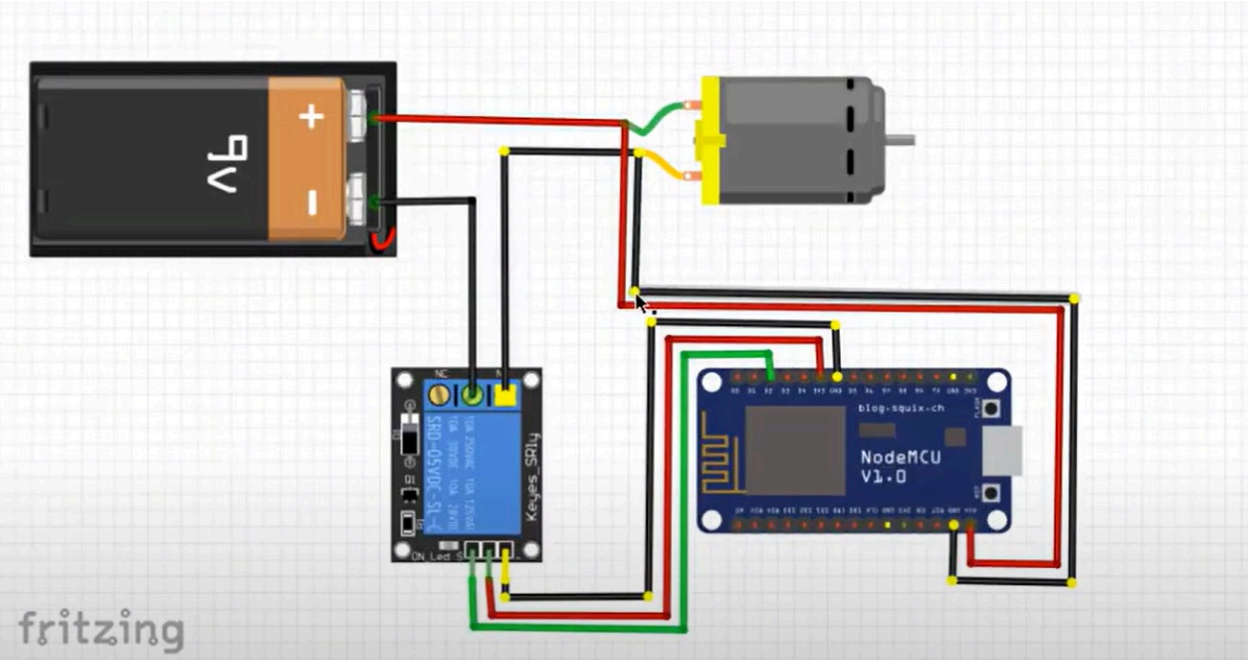
2. Workflow

Workflow of IoT-Based Smart Door Lock System:

1. User Interaction:
    - The user interacts with the system through a single HTML page user interface (UI) hosted on a local server or accessed directly from the microcontroller.
2. Mobile Application Communication:
    - The user accesses the HTML page UI from their smartphone's web browser.
    - The HTML page UI provides controls for locking and unlocking the smart door lock.
3. ESP8266 Microcontroller:
    - The ESP8266 microcontroller hosts the HTML page UI and serves it to users accessing the local server.
    - It is responsible for controlling the operation of the door lock based on user input.
4. Operation:
    - When the user interacts with the HTML page UI to lock or unlock the door, the corresponding command is sent to the ESP8266 microcontroller.
5. Door Lock Mechanism:
    - The ESP8266 microcontroller interfaces with the solenoid lock and relay module to control the locking and unlocking of the door.
    - Upon receiving a lock or unlock command, the microcontroller actuates the solenoid lock through the relay module.
6. Remote Access:
    - Users can access the HTML page UI remotely from their smartphones by connecting to the local server hosted by the ESP8266 microcontroller.
    - This allows users to remotely lock or unlock the door from anywhere within the local network range.
7. Monitoring:
    - Real-time feedback on the status of the door lock (locked or unlocked) may be provided to users through the HTML page UI.
    - The HTML page UI may display status indicators or messages indicating the current state of the door lock.

This simplified workflow demonstrates the local hosting of the user interface and the direct communication between the user's device and the ESP8266 microcontroller without the need for cloud services.

Circuit Diagram



Explanation of the Code

Certainly! Let's break down the functionality of the code in more detail without writing the actual code:

1. WiFi Configuration: The code defines the name (SSID) and password of the WiFi network that the ESP8266 will connect to. This allows the ESP8266 to establish a connection to the local WiFi network, enabling communication with devices on the same network.
2. Server Setup: The code initializes a WiFi server on port 80, which is the default port for HTTP communication. This server listens for incoming connections from clients (such as web browsers) and handles their requests.
3. Setup Function:
    - Serial Communication: Serial communication is initiated for debugging purposes, allowing messages to be printed to the serial monitor for monitoring and troubleshooting.
    - Pin Configuration: The GPIO pin connected to the solenoid lock is configured as an output pin. This pin will be used to control the locking and unlocking of the door.

- WiFi Connection: The ESP8266 connects to the WiFi network using the provided SSID and password. This step is essential for enabling communication over the local network.
- Server Start: The WiFi server is started, allowing it to begin listening for incoming connections from clients.
- IP Address Printing: The local IP address of the ESP8266 is printed to the serial monitor. This IP address is used by clients to connect to the ESP8266 server.

4. Loop Function:
    - Client Connection Handling: The loop continuously checks for client connections to the server. When a client connects, it is accepted, and the server waits for the client to send data.
    - Request Processing: Once data is received from the client, the server reads the client's request. This request typically includes information about what action the client wants to perform, such as locking or unlocking the door.
    - Action Execution: Based on the client's request, the server performs the necessary action, such as setting the GPIO pin connected to the solenoid lock to HIGH to unlock the door or LOW to lock it.
    - Response Generation: After performing the action, the server generates a response to send back to the client. This response typically includes information about the current state of the door lock and options for controlling it, such as HTML buttons for locking and unlocking the door.
    - Client Disconnection: Finally, once the response is sent, the server waits for the client to disconnect before moving on to the next iteration of the loop.

By breaking down the code's functionality in this way, you can understand how it enables communication between the ESP8266 and clients over a local WiFi network, allowing users to control the smart door lock remotely through a web interface.

```
#include <ESP8266WiFi.h>

const char* ssid = "Wifi network's name";
const char* password = "password";

int LOCK = 4; // GPIO4 (D2)
WiFiServer server(80);

void setup() {
  Serial.begin(9600);
  delay(10);

  pinMode(LOCK, OUTPUT);
  digitalWrite(LOCK, LOW);

  // Connect to WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the server
  server.begin();
  Serial.println("Server started");
  Serial.println("");
  Serial.println("****WiFi doorlock****");

  // Print IP address
  Serial.print("Use this URL to connect: ");
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");
}
```

```
void loop() {
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()) {
    delay(1);
  }

  // Read the first line of the request
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();

  // Match the request
  int value = LOW;
  if (request.indexOf("/LOCK=ON") != -1) {
    digitalWrite(LOCK, HIGH);
    value = HIGH;
  }
  if (request.indexOf("/LOCK=OFF") != -1) {
    digitalWrite(LOCK, LOW);
    value = LOW;
  }

  // Set LOCK according to the request

  // Return the response
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println(); // do not forget this one
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");

  client.print("Door is now: ");

  if (value == HIGH) {
    client.print("Open");
  } else {
    client.print("Closed");
  }
  client.println("<br><br>");
  client.println("<a href=\"/LOCK=ON\"><button><h1>Turn On</h1></button></a>");
  client.println("<a href=\"/LOCK=OFF\"><button><h1>Turn Off</h1></button></a><br/>");
  client.println("</html>");

  delay(1);
  Serial.println("Client disconnected");
  Serial.println("");
```

Future Uses of the Project

The future uses of our smart door lock system could include:

1. Integration with home automation for comprehensive control.
2. Access control solutions for businesses and rental properties.
3. Enhancing security and convenience in residential communities.
4. Streamlining access in healthcare facilities and educational institutions.
5. Customization and integration with other IoT devices for tailored solutions.