Mohamed Noureldin Maya Hussein Youssef Fares

Embedded Project - SafeLock

Introduction

The SafeLock project is a smart door lock system designed to enhance home security by utilizing NFC cards and Bluetooth for authentication. The goal is to replace traditional physical locks with a more secure and convenient solution. The SafeLock system allows users to open a door using pre-registered NFC cards. When a correct card is presented, a green LED lights up for 3 seconds, and the door is unlocked for a short period, simulated by rotating a servo motor. If an incorrect card is used, the system records the attempt and flashes a red LED for 3 seconds. After three incorrect attempts, the system triggers a buzzer to alert nearby security and enters a cooldown period before resetting the attempt count.

Additionally, SafeLock provides an administrative interface accessible via Bluetooth. Administrators must authenticate with a predefined passphrase and can then manage the system, including adding or removing authorized NFC cards and unlocking the door remotely. The system incorporates security measures such as limiting incorrect authentication attempts and requiring re-authentication after disconnection to prevent brute force and deauthentication attacks.

SafeLock uses SPI communication for the NFC card reader and UART for the Bluetooth module. Timers manage LED blinking and cooldown periods, and interrupts detect new card presentations. The system is built on an Arduino Mega board with various components, including a servo motor and an NFC card reader.

While the project has successfully implemented most features, including NFC-based door unlocking, LED indicators, and basic administrative controls via Bluetooth, there are limitations such as the capacity to store only up to 100 cards and the hardcoding of the administrator password. Some desired features, like extended administrative functionalities and more robust security protocols, could not be fully realized within the project timeline.

Hardware

Embedded Board Description

SafeLock is built using an Arduino Mega 2560 board. The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It features:

- **CPU:** ATmega2560 microcontroller with an 8-bit architecture.
- **Clock Speed:** 16 MHz.
- **RAM:** 8 KB of SRAM.
- **Flash Memory:** 256 KB, of which 8 KB is used by the bootloader.
- **EEPROM:** 4 KB.
- **I/O:** 54 digital I/O pins (15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

The Arduino Mega 2560 runs on the Arduino platform, which uses a simplified version of C/C++ known as the Arduino language. The development environment is the Arduino IDE.

Input Device Description

The primary input device for SafeLock is the MFRC522 NFC card reader. This reader communicates with the Arduino Mega 2560 via SPI (Serial Peripheral Interface). The NFC reader reads data from NFC cards, which are used for authentication.

- **Protocol:** SPI communication.
- **Pins Used:** SS_PIN (53), RST_PIN (49), IRQ_PIN (3), MOSI (51), MISO (50), SCK (52).
- **Communication:** The Arduino sends and receives data to/from the NFC reader using the SPI bus. The MFRC522 library is used to interface with the reader, providing functions to initialize the reader, check for card presence, and read card data.

Data Sheet: The MFRC522 datasheet (https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf) provides detailed specifications and operational instructions for the NFC reader.

Output Device Description

The output devices for SafeLock include a servo motor, LEDs, and a buzzer.

- **Servo Motor:** The servo motor is used to simulate the door lock mechanism by rotating to open or lock the door. It is controlled via PWM (Pulse Width Modulation) using the Servo library.
    - **Pin Used:** SERVO_PIN (2).
    - **Operation:** The servo motor rotates to a specified angle based on the input from the Arduino, simulating the locking and unlocking of the door.
- **LEDs:** Two LEDs indicate the status of the system – a green LED for successful authentication and a red LED for incorrect attempts.
    - **Pins Used:** GREEN_LED (10), RED_LED (12).
    - **Operation:** The LEDs are controlled using digital output pins, turning on and off based on the system's state.
- **Buzzer:** The buzzer alerts nearby security in case of multiple incorrect authentication attempts.
    - **Pin Used:** BUZZER_PIN (7).
    - **Operation:** The buzzer is controlled using a digital output pin and is activated when the number of incorrect attempts exceeds the allowed limit.

Power Consumption

SafeLock is designed to be energy efficient, but power consumption is a critical consideration, especially for a system intended to be operational 24/7. The Arduino Mega 2560 operates at 5V with a current consumption of approximately 50mA when idle. The NFC reader, servo motor, LEDs, and buzzer add to the total power consumption:

- **NFC Reader:** Typically consumes around 20mA when idle and up to 30mA during communication.

- **Servo Motor:** Power consumption varies based on the load and movement but generally consumes around 10-20mA when idle and can spike to over 100mA during operation.
- **LEDs:** Each LED consumes about 20mA when lit.
- **Buzzer:** Typically consumes around 30mA when activated.

To optimize power consumption, several strategies can be employed:

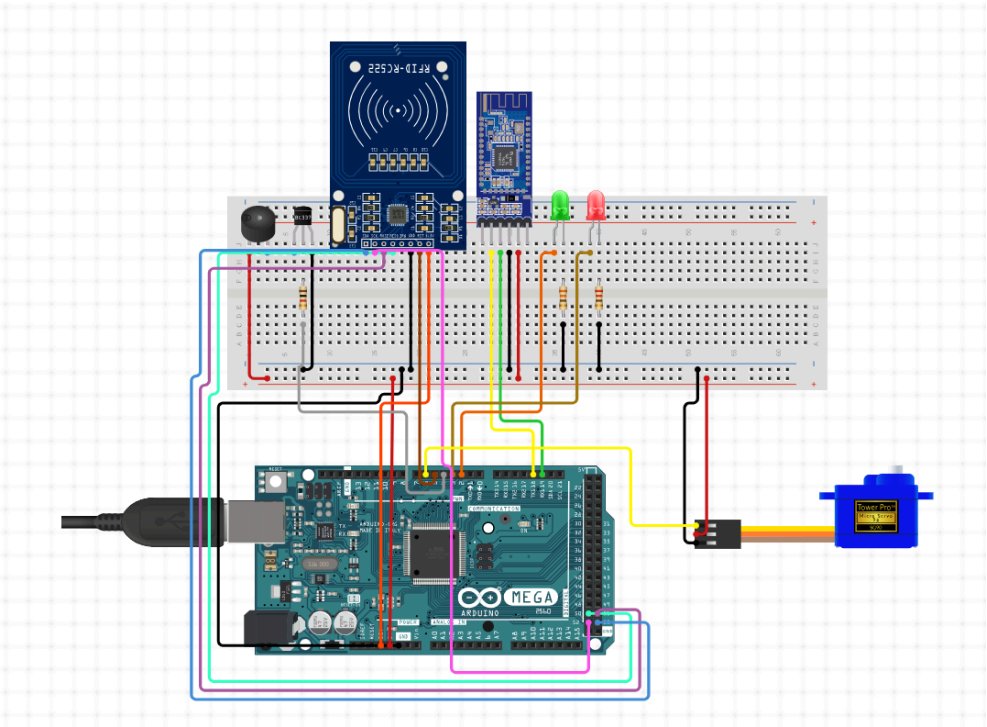
- **Sleep Modes:** Implementing sleep modes for the Arduino during periods of inactivity.
- **Efficient Coding:** Ensuring the code is optimized to reduce unnecessary processing.
- **Hardware Optimization:** Using low-power components and reducing the duty cycle of power-hungry components like the servo motor and buzzer.

Pricing

Parts	Price
Arduino Mega 2560 R3	$38.5
9g Micro Servo	$5.95
Piezo Buzzer	$0.60
RFID - RC522 RF IC Card Sensor Module	$2.51
HM-10 BLE Bluetooth 4.0	$2.85
LED - Basic Green 5mm	$0.33
LED - Basic Red 5mm	$0.36
Breadboard	$8.25
Total	$59.35

Comparing our product with other competitors, other door locks that had similar functionalities ranged from around $ 160 to $ 249.99, which varied based on its features. However, all these are more expensive than our product even though they do offer almost the same main features presented in our product. This gives SafeLock an edge as opposed to other competitors in the market.

Schematics



Software

Programming Language

The SafeLock system is programmed using the Arduino language, which is a simplified version of C/C++ tailored for the Arduino microcontroller environment. The choice of Arduino language is due to its ease of use, extensive library support, and strong community backing. These factors significantly speed up development and debugging, especially for handling hardware components like the NFC reader, servo motor, LEDs, and Bluetooth module.

Real-Time Constraints

The SafeLock system has some real-time constraints to ensure timely responses to user interactions and security events. For instance:

- **Card Reading:** The system must promptly detect and read NFC cards presented to it.
- **LED Indicators:** The green and red LEDs must blink for a specified duration to provide visual feedback.
- **Cooldown Periods:** The system must accurately manage cooldown periods after incorrect attempts.
- **Servo Motor:** The servo motor must quickly respond to unlock and lock commands.

If the code encounters unexpectedly large delays, it could lead to security issues (e.g., delays in detecting unauthorized access attempts), poor user experience (e.g., slow unlocking), or even system malfunctions. Therefore, timers and interrupts are used to manage these real-time constraints effectively.

Test cases:

- Test successful unlocking using a valid NFC card.
- Test unsuccessful unlocking using an invalid NFC card.
- Test successful unlocking by an administrator via Bluetooth with the correct passphrase.
- Test unsuccessful unlocking by an administrator via Bluetooth with an incorrect passphrase.
- Test addition of a new NFC card to the system.
- Test successful removal of an existing NFC card from the system.
- Test triggering of the alarm after three unsuccessful unlocking attempts.
- Test that the green LED lights up upon successful unlocking.
- Test that the red LED flashes upon unsuccessful unlocking.

Unit Tests:

- TestNFCUnlockSuccess: Verify that the system unlocks successfully using a valid NFC card.
- TestNFCUnlockFailure: Verify that the system does not unlock using an invalid NFC card.
- TestAdminUnlockSuccess: Verify that the system unlocks successfully via Bluetooth with the correct passphrase.
- TestAdminUnlockFailure: Verify that the system does not unlock via Bluetooth with an incorrect passphrase.
- TestAddCardSuccess: Verify that a new NFC card is successfully added to the system.
- TestRemoveCardSuccess: Verify that an existing NFC card is successfully removed from the system.
- TestRemoveCardFailure: Verify that the system fails to remove a non-existent NFC card from the system.
- TestAlarmTrigger: Verify that the alarm is triggered after three unsuccessful unlocking attempts.
- TestGreenLedOnUnlock: Verify that the green LED lights up upon successful unlocking.
- TestRedLedOnFailure: Verify that the red LED flashes upon unsuccessful unlocking.

Security

Security is a critical aspect of the SafeLock system, as it directly impacts the physical security of a home or facility. Several measures are implemented to enhance security:

- **Limited Attempts:** The system limits the number of incorrect authentication attempts to three. After exceeding this limit, it triggers a buzzer and enters a cooldown period to prevent brute force attacks.
- **Admin Authentication:** Admins must authenticate via Bluetooth with a predefined passphrase to access administrative functions. This process also limits incorrect password attempts to three before a cooldown period.

- **Encryption and Data Integrity:** Although not implemented in the current version, future improvements could include encryption of data transmitted over Bluetooth and integrity checks for NFC communication to prevent interception and tampering.
- **Re-authentication:** Users must re-authenticate after disconnecting from the Bluetooth interface, preventing deauthentication attacks.

Use of External Code

The SafeLock project utilizes several libraries to interface with the hardware components and manage system functions:

- **MFRC522 Library:** Used to interface with the MFRC522 NFC reader. This library provides functions for initializing the reader, checking for card presence, and reading card data.
- **Servo Library:** Used to control the servo motor. This library simplifies PWM control of the motor to rotate it to specific angles.
- **Countimer Library:** Manages timers for cooldown periods and LED blinking intervals.
- **SoftwareSerial and HardwareSerial Libraries:** Facilitate communication with the Bluetooth module via UART.

These libraries handle low-level hardware interactions, allowing us to focus on the application logic. The majority of the code, including the implementation of authentication logic, input/output handling, and security features, is original.

Related Work

Previous Similar Projects

Several projects and commercial products have aimed to enhance home security using smart lock technologies similar to SafeLock. Here are a few notable examples:

- **Smart Locks by Companies:**

    - **August Smart Lock:** This commercial product allows users to lock and unlock doors using a smartphone app, with additional features like remote access, guest keys, and integration with smart home systems.
    - **Yale Assure Lock:** Another popular commercial smart lock, it offers features such as keyless entry, voice assistance integration, and various access methods including keypads, biometric sensors, and mobile apps.
- **DIY Smart Lock Projects:**
    - **Arduino-Based Smart Locks:** Numerous DIY projects have been shared on platforms like Instructables and GitHub, where hobbyists build smart locks using Arduino boards, RFID/NFC readers, and Bluetooth modules. These projects typically focus on providing keyless entry and basic access control.

Comparison With SafeLock

While SafeLock shares similarities with both commercial products and DIY projects, it also has distinct features and improvements:

- **NFC and Bluetooth Integration:** SafeLock combines both NFC and Bluetooth technologies for authentication and administrative control, whereas many DIY projects focus on either one. This dual approach enhances security and user convenience.
- **Security Measures:** SafeLock implements multiple layers of security, including limiting incorrect attempts, using cooldown periods, and requiring re-authentication after disconnections. These measures make it more resilient against common attack vectors like brute force and replay attacks, compared to many DIY projects that often overlook these aspects.
- **Administrative Interface:** The Bluetooth-based administrative interface in SafeLock allows for remote management of NFC cards and system settings, providing greater flexibility and control. This feature is more advanced compared to most DIY projects, which typically require physical access for configuration changes.
- **User Feedback and Alerts:** SafeLock uses visual (LEDs) and auditory (buzzer) feedback to inform users of the system's status and security events, enhancing user interaction and situational awareness.

- **Scalability and Capacity:** While SafeLock has a limitation of storing up to 100 NFC cards, this capacity is often sufficient for home or small office use. Some commercial products may offer higher capacities, but for a DIY project, this is a reasonable balance between functionality and complexity.

Future Enhancements

To further improve SafeLock and make it more competitive with existing solutions, the following enhancements could be considered:

- **Encryption:** Implementing encryption for data transmission between the NFC reader, Bluetooth module, and Arduino to enhance security.
- **Biometric Integration:** Adding biometric authentication methods such as fingerprint or facial recognition for an additional layer of security.
- **Cloud Connectivity:** Enabling remote access and management via cloud services for enhanced flexibility and real-time monitoring.

Conclusion

Team Contributions

The SafeLock project was a collaborative effort by our team, consisting of Mohamed Noureldin, Maya Makram, and Youssef Fares. As a team we worked together during the spring break to implement every component and its integration. We also sat together to make decision choices including the timer selection, interrupts, and the implementation of the card management system.

Challenges

Throughout the development of SafeLock, our team faced several challenges:

- **Hardware Integration:** Ensuring reliable communication between the various hardware components (NFC reader, servo motor, LEDs, buzzer) and the Arduino Mega board required careful wiring and debugging.

- **Real-Time Constraints:** Managing real-time constraints such as accurate cooldown periods and responsive card reading posed challenges. We used timers and interrupts to address these issues effectively.
- **Security Implementation:** Implementing robust security measures to prevent brute force attacks and unauthorized access was crucial. This required careful planning and coding to ensure the system's resilience.
- **Bluetooth Communication:** Developing a reliable and secure Bluetooth interface for the administrative functions involved overcoming issues related to data integrity and connection stability.

Future Work

While SafeLock successfully achieved its primary objectives, there are several areas for future improvement:

- **Increased Capacity:** Expanding the system's capacity to store more than 100 NFC cards would make it suitable for larger installations.
- **Enhanced Security:** Implementing encryption for data transmission and incorporating biometric authentication methods (e.g., fingerprint or facial recognition) would further enhance security.
- **Cloud Connectivity:** Adding cloud connectivity for remote access and management could provide users with greater flexibility and real-time monitoring capabilities.
- **User Interface:** Developing a user-friendly mobile application for both regular users and administrators could improve the overall user experience.