Smart Locker

Release notes:

Version: 11.05 - 1st assigment version. Work was started on the other milestones as well but it is yet unfinished Done: general details, hardware implementation and most of the software implementation.

Version 21.05 - 2nd milestone - full functionality achieved.

Introduction

Smart Locker is a solution for securing valuable items coming in the form of a lockable safe.

Description and purpose

Smart Locker is a solution for securing valuable items coming in the form of a lockable safe - similar to the EasyBox lockers, which also were the inspiration for this project.

Mainly, this locker is designed to secure various items. Nevertheless, the circuit can be easily adapted (due to it's mechanism) to lock/unlock doors, drawers or so on.

Functionality

Functionality TLDR:

- The locker can be opened/locked by introducing the correct numerical combination or presenting the RFID card.
- If 3 wrong PINs are introduced, the buzzer is triggered, thus entering alarm mode.
- Alarm mode can only be stopped by presenting the Master RFID Card.
- The PIN can be reset by presenting the Master RFID card (not the standard one!) and entering/confirming twice the new PIN
- For each PIN introduction, a status is displayed (“Correct!” / “X tries remaining”)
- A PIR sensor detect human presence close to the keypad.
- If the door is forced, high pressure is read by the weight sensor which prompts entry to alarm mode.
- A motor moves the mechanism for locking/unlocking the door.

General Description

To implement the project in a physical format, a servo motor is used which performs the opening/closing of the door. To actuate the servo motor, and thus open the door, a keypad is added, through which the user can enter the correct password, as well as an RFID card communication module, where, using the correct card, the door can be opened.

The locking mechanism rests against a weight sensor - if the door is forced (i.e high weight reading), the buzzer is triggered. This functionality was later abandoned, since the two load cells used were either too hard to trigger (50kg, 3 wires) or had no pins remaining to be wired to the Arduino board (10kg, 4 wires). Right now, the 50kg is present but not connected, it only is used as part of the locking mechanism.

The alarm system is activated if the door is forced or 3 incorrect passwords have been entered consecutively 3 times, with the user having three attempts to open the door. Once the alarm system is activated, a buzzer will emit a loud audible signal, which can only be stopped by using the correct RFID card.

Hardware Design

Hardware implementation details:

- The code, live state and schematics of the project can be closely monitored on this TinkerCad page: External Link.
- Some of the components are not available in Tinkercad, such as the RC544 RFID or weight sensor module.

Electrical Layout:

Arduino Pinout:

Necessary components:

- SG90 Micro Servo Motor
- 4×4 Matrix Keypad
- RFID-RC522 Communication Module

- LCD with Integrated I2C Module
- Arduino UNO R3 Development Board
- Breadboard Kit with Jumper Wires
- Active buzzer
- PIR sensor HW-201
- 10kg Load Cell

Pinout: Keypad (8 pins)

- Row pins: D9, D8, D7, D6
- Column pins: D5, D4, D3, D2

Servo Motor

- Signal: D10
- Analog Pins

RFID Module (RC522)

- SS (Chip Select): A0
- RST (Reset): A1
- MOSI: D11 (Fixed for SPI)
- MISO: D12 (Fixed for SPI)
- SCK: D13 (Fixed for SPI)

Load Cell

- Signal: A3

Buzzer

- Signal: A2

I2C Bus

- LCD Display
- SDA: A4 (Fixed for I2C)
- SCL: A5 (Fixed for I2C)

Power Connections:

VCC (5V)

- LCD Display
- Servo Motor
- Load Cell
- Keypad

VCC (3.3V)

- RFID Module

GND

- LCD Display
- RFID Module
- Servo Motor
- Load Cell
- Buzzer
- Keypad

Software Design

Code available here

Finite State Machine: The core logic is driven by the currentState variable, which can be one of the following:

- STATE_IDLE: The default state, waiting for PIN input or RFID card.
- STATE_STATUS: Briefly displays status messages (e.g., “Unlocked”, “Locked”).
- STATE_ALERT: Triggered after multiple incorrect PIN attempts, requiring a master RFID card to reset.
- STATE_MASTER_MENU: Activated by the master RFID card, allowing the user to change the PIN.
- STATE_NEW_PIN_FIRST: The first step in changing the PIN, prompting for the new PIN.
- STATE_NEW_PIN_SECOND: The second step, prompting for confirmation of the new PIN.

The loop() function continuously checks for RFID card presence and reads the keypad. Based on the current state and the input received, it transitions to other states and performs corresponding actions.

Important Functions:

- setup(): Initializes all the peripherals (LCD, servo, buzzer, keypad, RFID) and sets the initial state of the door lock.
- loop(): The main execution loop that handles state transitions, reads sensors and input devices, and performs actions based on the current state.
- handleIdleState(): Manages the PIN entry process via the keypad, checks the entered PIN against the stored password, and handles locking/unlocking or triggering the alert state.
- unlockDoor(): Controls the servo motor to the unlocked position.
- lockDoor(): Controls the servo motor to the locked position.
- isMasterCard(): Checks if the scanned RFID card's UID matches the masterUID.
- isUserCard(): Checks if the scanned RFID card's UID matches the userUID.

What are the used libraries and why?

- <Servo.h>: This library provides functionalities to control servo motors. It's used here to move the servo that controls the door lock.
- <Wire.h>: The Wire library allows communication with I2C devices. It's essential for communicating with the I2C-based Liquid Crystal Display (LCD).
- <LiquidCrystal_I2C.h>: This library provides a convenient way to control I2C-based LCDs, making it easy to display text and information to the user.

- <Keypad.h>: This library simplifies reading input from a matrix keypad. It handles the scanning of rows and columns to detect which key is pressed.
- <SPI.h>: The SPI (Serial Peripheral Interface) library is used for communication with devices that use the SPI protocol, such as the MFRC522 RFID module.
- <MFRC522.h>: This library provides functions to interact with the MFRC522 RFID reader/writer module, allowing the system to read RFID card UIDs.

These libraries abstract away the low-level details of hardware communication, making it easier to develop the application logic.

Results, Conclusions

The intended functionality was achieved (except the load cell feature) - a demo can be found on the following YouTube link or embedded in the GitHub page.

Download

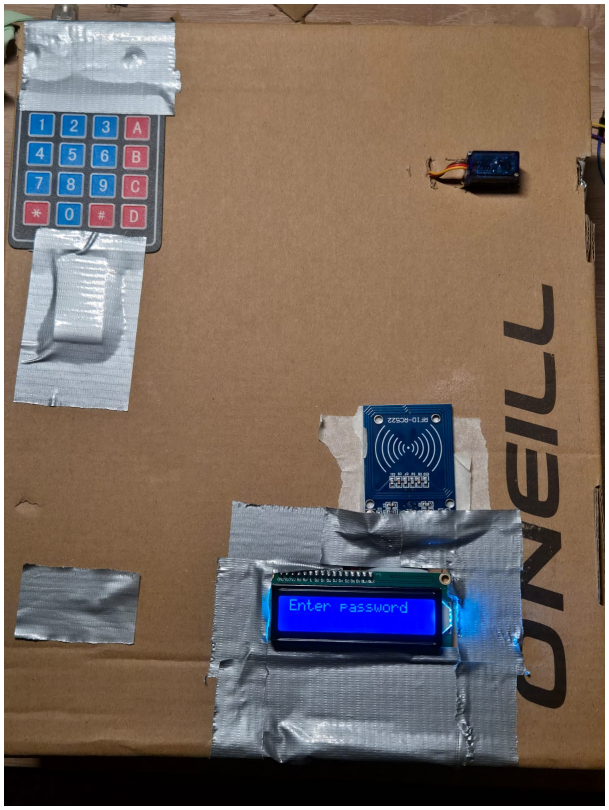
In order to try out the project for yourselves, use the same pinout as described in the project and download the project repository: git clone https://github.com/MirceaOvidiu/SmartLocker

Jurnal

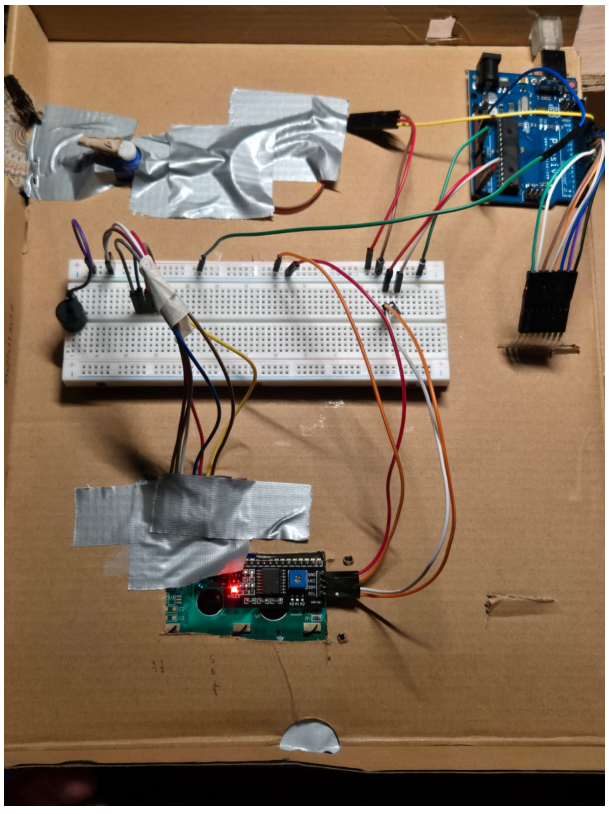
11th of May - started initial wiring of the components to check basic functionalities.

14th of May - finished wiring and torture of the sacrificed shoebox.

1. Frontal view.

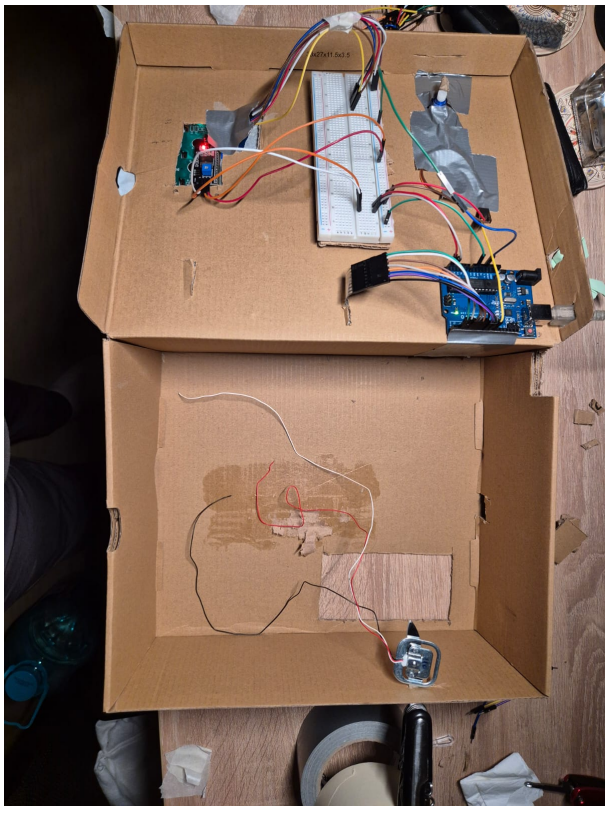


2. Interior view

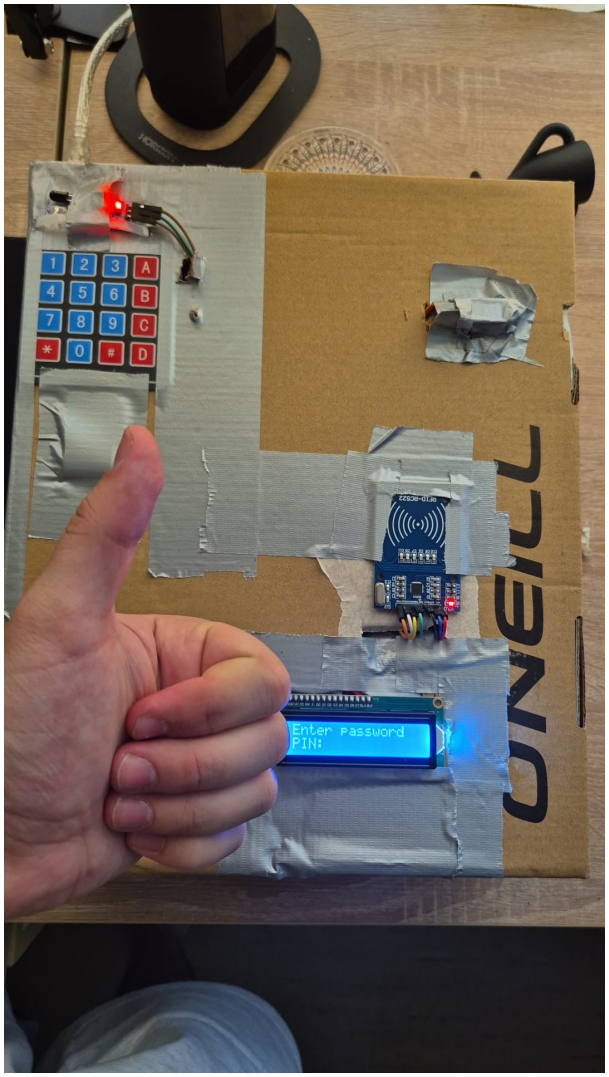


3. This is the mechanism that prevents the use of brute force - an weight sensor is placed above the locking mechanism.

4. Interior overview.



5. Final version - 21st of May



Bibliography

Resources used:

RFID guide

Servo Motor guide

Keypad configuring

Load cell and HX711 module tutorial

I2C display configuration

Export to PDF