Automated Locker

Design Document

Prepared for:
Alden High School

Prepared By:
Tijmen Van Der Beek, Miren Patel, Siquan Wang, Xiaoang Zhang, Dazhou Liu

School of Engineering and Applied Sciences
University at Buffalo, The State University of New York
Spring 2021

Table of Contents

Introduction

The purpose of this project is to design an automated and easily operated locker for Alden High School. According to the staff at Alden High School, the current lockers have frequent locker jams. This results in additional custodial resources which can be prevented by having better designed lockers. These RFID Locker Modules will use Radio Frequency Identification (RFID) for authentication for opening the lockers by both registered students as well as administrative personnel such as the custodial staff. The design will also ensure that the lock mechanism does not have frequent locker jams. For the safety and security of the school, a mechanical override system will also be incorporated to allow for the circumvention of the digital systems. Due to the potential for a large rollout of this system, scalability and reliability have been key considerations at every step of our development. Everything can be found on Github.

Note for RFID and NFC Components

The locking/unlocking mechanism is executed by RFID components in our design. However, the previous locker team working on Alden High School locker chose to use the NFC components. In the Spring 2021 semester, we acquired the NFC components and attempted to implement them. The result was that the NFC components did not work as expected due to improper factory functionalities. As a consequence, we decided to use RFID modules to lock/unlock the locker in Alden High School.

RFID Locker Module

1. **Master tag**: Acts as a master key, allowing opening the lock mechanism on any locker.
2. **Registrar tag**: Puts the locker into registration mode, allowing for registration of a new student tag. Student tag registrations persist until cleared with the clear tag.
3. **Clear tag**: Clears all registered student tags. Only master tags can open the locker until a new student tag is registered.
4. **Reset Button**: When the locker is first booted up you can register the administrative tags after that the reset button is the only way to reset the locker and re-register new ones.

The use of administrative tags in this way eliminates the need for additional input channels (such as buttons), reducing the hardware complexity and cost. We have also attempted to minimize the number of output channels as well, using a single RGB LED to convey the system state.

Software

The provided RFID Locker Module software is written in C++ for deployment on an Arduino Uno (or equivalent). The main source code is LockerCodeAlden.ino. In the source code

(available on our Github repository) you'll find descriptive comments to help with understanding the code's functionality as well as its deployment.

Important Static Variables

- **Init_num:** This is the number it checks to see if it has been initialized yet.
- **tagDelay:** This is the processing delay between each tag read in ms
- **ERRORBlinkRate:** This is the speed at which the led binks in ms
- **LOCKER_TIME:** 30 This is the time the locker has till it turns off in sec when locked
- **UNLOCK_TIME:** 10 This is the time the locker will be unlocked for in sec when unlocked

Functions

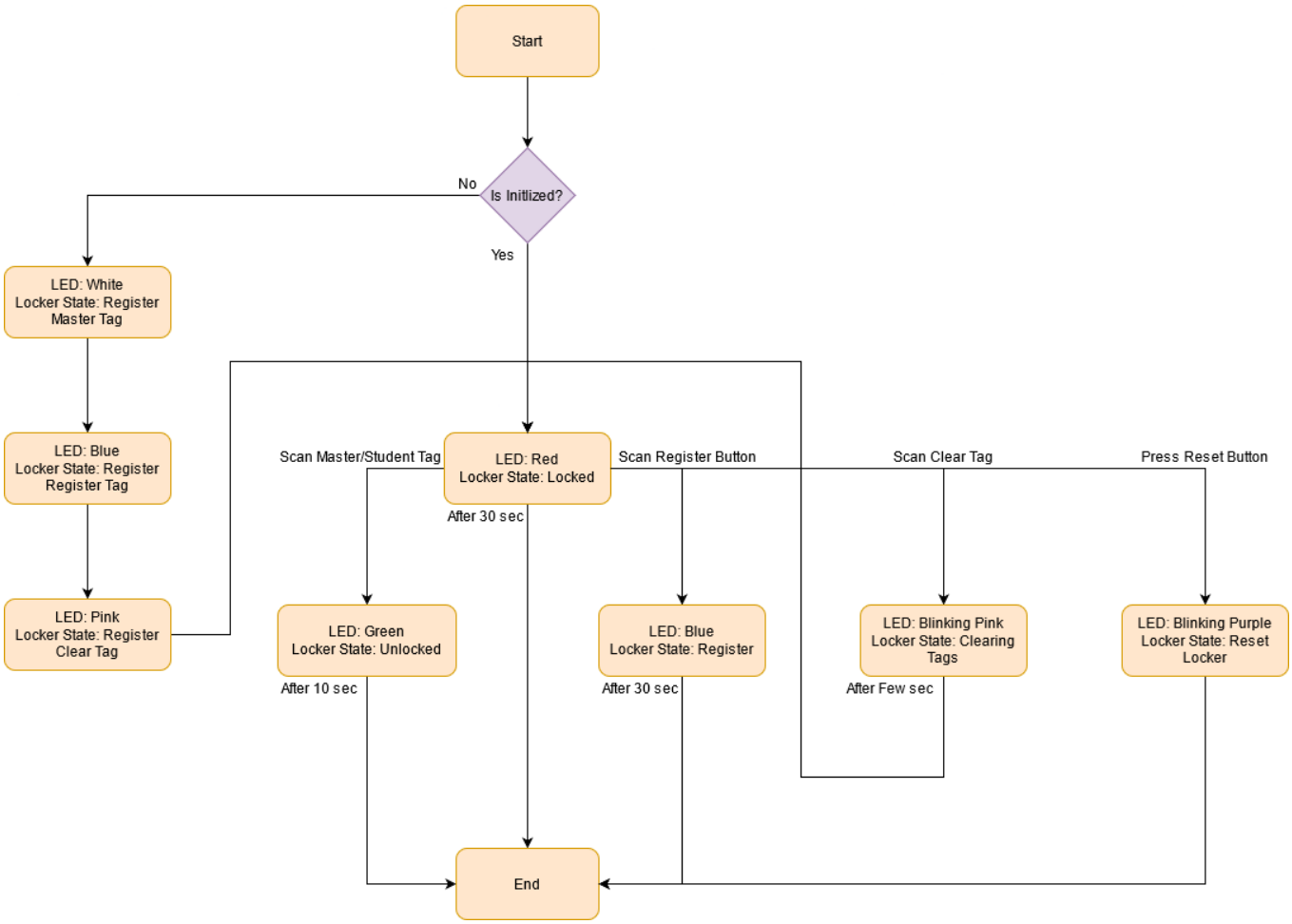
- **void ReadSerial(String &ReadTagString):** This function reads the tag in front of the RFID sensor and passes the string through ReadTagString
- **void clearTags():** This function clears all the student tags in EEPROM
- **void resetLocker():** Resets the locker
- **void loadTags():** This function loads all the tags from the EEPROM into the global tag_list
- **bool writeTag(String data):** This function adds a tag to memory and returns true if successful and false if reached max tags
- **int checkTag(String data):** Checks the given tag with the registered tags if any of them match it returns int for the type of tag.
- **void setLed (int rVal, int gVal, int bVal):** Sets the rgb led color of only the status LED
- **void ledBlink (int rVal, int gVal, int bVal, int halfCycle, int blinkCount):** Blink the RGB LED given the provided RGB value, the time for each half cycle (time off, time on, in ms) and the number of blinks. (Only Status LED)
- **void soundFeedback(int type):** Adds sound effects to the locker for better feedback (0 start, 1 unlock, 2 wrong tag)
- **void initLocker()**: Initializes the locker with faculty tags only runs if not initialized
- **void setBatteryLed():** Sets the battery led color based on the voltage of the battery
- **void enableTimerInterrupt():** Sets the timer interrupt to interrupt every second (We have a counter to determine the time)
- **void resetTimer():** Resets the timer counter back to Locker_Time whenever called

Interrupts

- **void resetISR():** Sets reset to true and sets led to purple
- **ISR(TIMER1_COMPA_vect):** Turns off locker after timer interrupt activates and counter is at 0

Main Flow Chart

This is a flow chart of the logic that the locker should follow.



RGB LED Indicator Key

Status LED (LEFT)

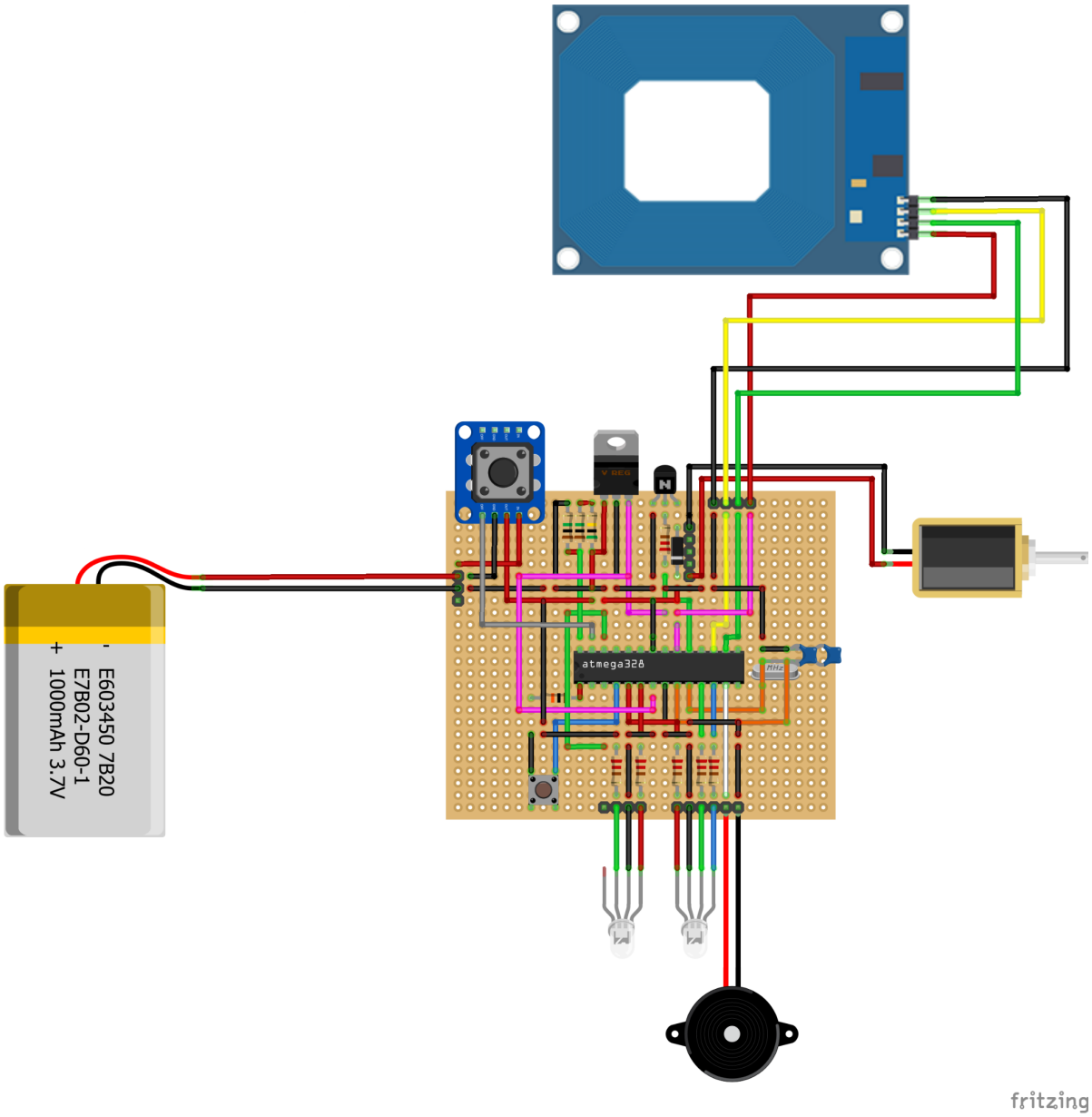
State	LED Color
Initial State	OFF
Locked	RED
Unlocked	Green
Processing	Yellow
Register Mode	Blue
If the tag is wrong	Blinks RED
If the tag is correct	Blinks GREEN
Successful Register	Blinks Blue Twice
Already Registered	Blinks Blue Three Times
Unsuccessful Register	Blinks Red

Already Registered Faculty Tag	Blinks Red and color of tag type.
Resets Locker	Blinks Purple turns off
Master Tag Registration Mode	White
Register Tag Registration Mode (Only after Master Tag Reg)	Blue
Clear Tag Registration Mode	Pink
Successful Faculty Tag Registration	Blinks Color of tag type

Battery LED (RIGHT)

State	**LED 2 Color (Battery Indicator LED)**
Initial State	OFF
When the push button is pressed, it turns on the circuit and is ready to read the tags.	**GREEN:** if the battery level is sufficient (battery is greater than 10 V) **RED:** if the battery is at low level (the battery is less than 8 V but greater than 7 V) **Blinking RED:** if the battery is low (the battery is less than 7 V). At this point the circuit won't turn on, battery recharge is necessary
OFF State	OFF

Schematic



Description

- We are using the Atmega328p Chip that we program with the Arduino and put on our circuit.
- The battery is a 12 V 3Ah DeWalt battery. The one in the schematic is a placeholder image.
- Adafruit Power Switch Button: This is the power button for the locker with an off pin on it allowing us to turn off the locker after being idle for a certain amount of time.

- The voltage divider takes the 12 volt power line and reduces it so the chip can read it and set the Power Led appropriately. The voltage divider consists of two 1M Ohms, one 470k Ohm and the output voltage shouldn't exceed 5 volts.
- The Voltage regulator brings down the 12 volt to 5 volt to power the chip and RFID reader.
- The 2N2222 Transistor is used to power the solenoid with the chip.
- Status RGB LED:
    - Red On, but locked
    - Green→ Unlocked
    - Blue→ Register mode
    - Blinking pink→ Clearing all tags
    - Blinking blue→ Successful tag registration
- Power RGB LED:
    - Red Low Battery
    - Green High Battery
    - Blinking Red Out of Power
- We are using headers on the circuit board so parts can be easily replaced.
- We are using two 22pF capacitors and a 16Mz Clock to run the chip.
- We have a 10k Ohm resistor on the reset pin on the chip.
- The resistors on the LEDs and 2N2222 transistor are 220 Ohms.

Parts List

1. Arduino UNO R3
2. Atmega328 chips (Remove from Arduino board)
3. Adafruit Push-button Power Switch Breakout
4. RFID Card Reader - Serial
5. RFID Tags x5
6. Diode Rectifier 1N4001 x3
7. Voltage Regulator
8. 2N2222A Transistor x3
9. LED RGB Diffused x6
10. 220ohm Resistor x20
11. 1M ohm Resistor x2
12. 470k ohm Resistor x
13. 10k Resistor
14. 22pf Capacitors x2
15. 16Mz Clock
16. DeWalt 12v Lithium Ion Battery

17. DeWalt 12v-20v Lithium Ion Battery Charger
18. 5V 2 Terminals Buzzer x2
19. Key Lock
20. Solenoid
21. PCBs
22. White Filament (Used for LED caps)
23. Breadboard, wiring, washers/nuts/screws
24. Push Button (Activation)
25. Push Button (Reset)
26. Magnetic Push Latches (Optional)

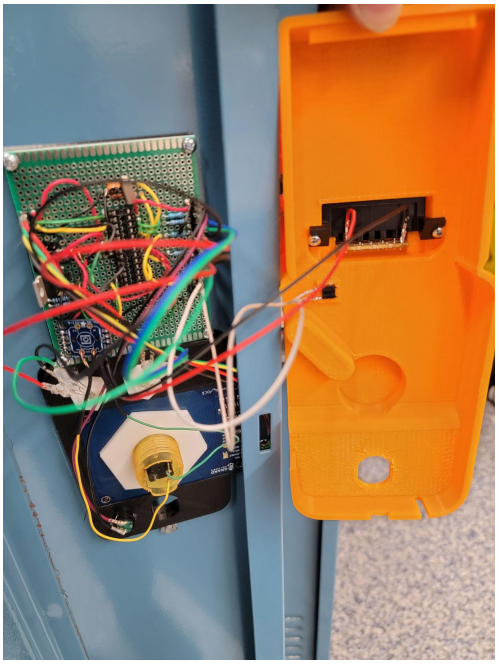
Case

Front

The front of the case has two LEDs, the power button and the lock for the mechanical override.

Inside Front

The inside shows the RFID reader in the center with the button going through it. We glued the speaker in the bottom corner. We have two LED caps screwed in with nuts above the RFID reader. On the top is where we screwed the circuit board. On the bottom is where the lock core goes.

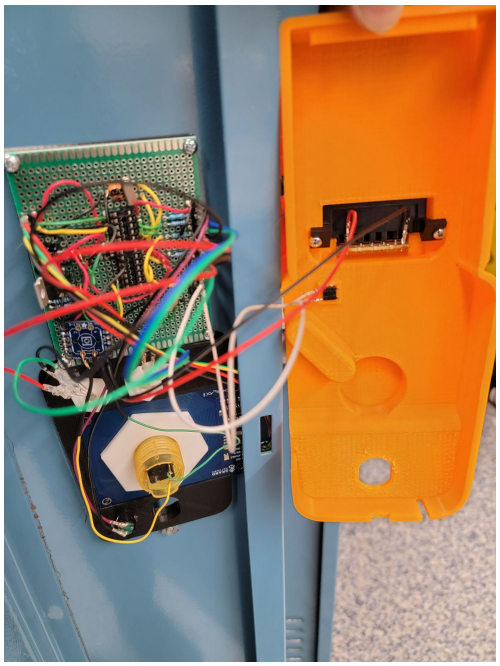


Back

We have the spot for the DeWalt battery and the register/clear button below that. The register/clear button is recessed so that it cannot be accidentally pressed and must be pushed with something sharp like a pencil.

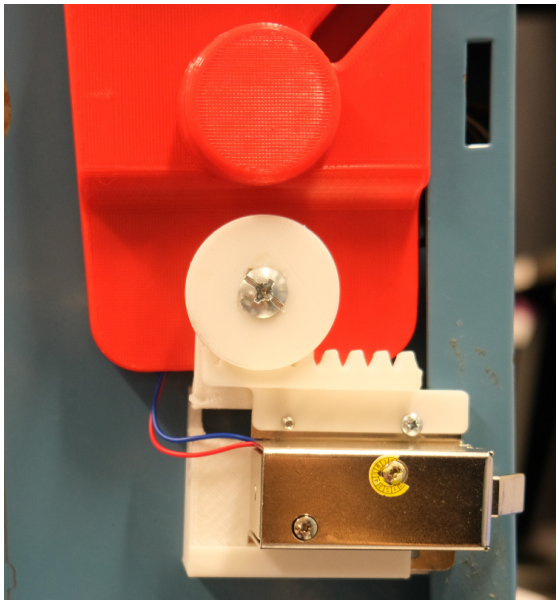
Inside Back

We used the charging insert from the DeWalt charger as our battery mount and screwed it into the locker case. We glued the register/clear button to the back. The prints can be found on Github.



Mechanical Design

The mechanical override utilizes a gear and a rack to horizontally slide the solenoid. The rack is a jagged rectangle plate that is mounted to the solenoid with screws. The gear is screwed to the locker core and is attached to the rack.
When a key is inserted into the keyhole and turned, the gear inside is turned around. As a result, the rack with the solenoid screwed to it is moved back and forth. When the solenoid is moved far enough inside, the solenoid no longer acts as a door latch resulting in the locker to be unlocked.If the magnetic push-to-open latch is installed, make sure to push the locker door in completely before turning the key. This is because the spring is pushing onto the door as well as the solenoid. There will be friction on the solenoid if it is not pushed in. It can potentially damage the solenoid bracket and even the solenoid in extreme cases. All the 3D parts can be found on our Github.



Upgrades

- Improving the mechanical override with compact gear built with better materials to improve smoothness of unlocking process. We can even change the whole design to a superior one where the solenoid can be properly mounted.
- Custom manufactured circuit board instead of hand soldering through-hole board to improve stability and durability of the lock and speed up the soldering process.
- Some minor improvements to the case and overall design of the locker to make the assembly process easier.
- Designing a better case to incorporate better with the school locker with the school's latch design.
- Central power distribution system to multiple lockers and put lockers in parallel.
- Incorporate a larger battery like the 20 V 10Ah DeWalt battery.
- Better energy efficiency.
- Disable the access of the reset button for normal users and only let the administrators have access to it.
- See what issues the current prototype comes up with during testing and improve from there.

Troubleshooting

- The circuit will not turn on properly, if at all, without the RFID reader attached to it.
- If the circuit is not working properly or not turning on, try replacing the chip.

- If the circuit turns on but the startup sound sounds off, clear all the tags with the button on the back. (This will most likely happen with a new chip that was just programmed)
- If the circuit turns on at 7 V input instead of 12 V input or is not turning on at all, the voltage regulator might be damaged. Swap a new voltage regulator to fix the problem.
- If the status LED is blinking RED while being powered by 12 V, the voltage divider might be broken. Desoldering the 3 resistors and resoldering on 3 new resistors with the same resistance should fix this issue.
- If solenoid is not working, this could be either the solenoid or the npn transistor. Swap the solenoid first and test the functionality. If it is still not working, swap the npn transistor on the board.
- If any plastic part is broken, you can reprint the part and replace the broken part.
- If the locker does not read the tags, clear all the tag registrations and try to re-register one tag. If it is still not functioning, check the RFID wire connection with the board and retry. If still having trouble, replace the RFID with a new one. If the problem still persists after these steps, replace the processing chip on the board with a new one.
- If the buzzer does not work, check the wire connection with the board first and retry. If you are still having trouble, replace the buzzer with a new one.

References

We inherited the previous two locker teams' ideas then improved and expanded from them. This is their Github1 Github2. Our team's Github is linked here.