Automated Locker Design Documentation

Zachary Salim, Simran Singh, Sriram Subramanian,
Jacob Bellewich, Thomas Wu, Anand Balakrishnan, Ian Wilson

CSE453 — Hardware/Software Integrated Systems Design
University at Buffalo
Spring 2018

Figure 1: Front of Modified Locker Door

Contents

1 Introduction

Locker 2135 in Alden, NY Middle School was modified with an electronic system that replaces the previous locking mechanism. The system allows a user to unlock the locker door by scanning a pre-registered RFID (Radio-Frequency Identification) tag. After 10 seconds of being unlocked, the locking mechanism automatically locks again. If the door is still opened when the locking mechanism has automatically locked again, the door can be closed, as the latch of the lock is designed with this in mind. RFID tags can be registered and de-registered from the system. A mechanical key override was also installed to allow access to the inside of the locker in case of electronic system failure (e.g., dead battery).

2 Important Note

Refer to the Automated Locker User Manual for detailed diagrams of system components and their layout on the locker door and in the locker, as well as a description of how the system operates and is used from a user's perspective.

3 Parts List

- Lock-Style Solenoid (Rated for 12V)
- 60mm White Arcade Momentary Push Button
- Dewalt 12V Lithium Ion Battery (Model No. DCB127)
- Dewalt Battery Charger (Model No. DCB115)
- Arduino Uno
- MFRC522 RFID Reader and Tags
- Common Anode RGB LED x 2
- Pololu Pushbutton Power Switch (HP)
- 1N4004 Diode
- NPN BJT Transistor TIP120 (high-power) and heatsink
- 220 Ohm Resistors x 3
- 10K Ohm Resistor
- 4-40 bolts x 11
- 4-40 nuts x 11
- 3 mm locking washers x 11
- 6-32 bolts x 4
- 6-32 nylon locking nuts x 4
- 6-32 locking washers x 4

4 Electronics

4.1 Circuit Schematics

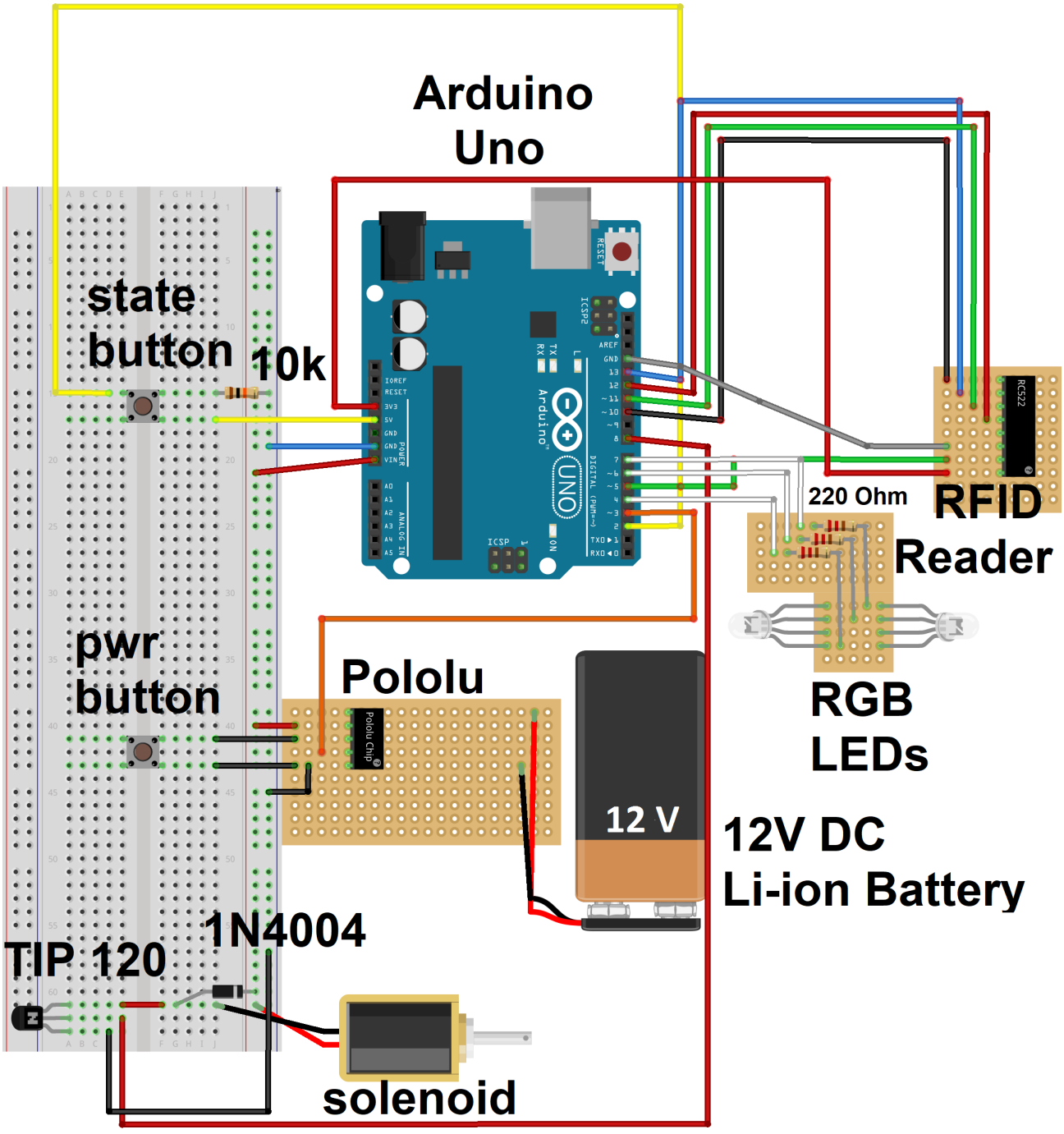

Figure 2: Schematic of the Locker circuit

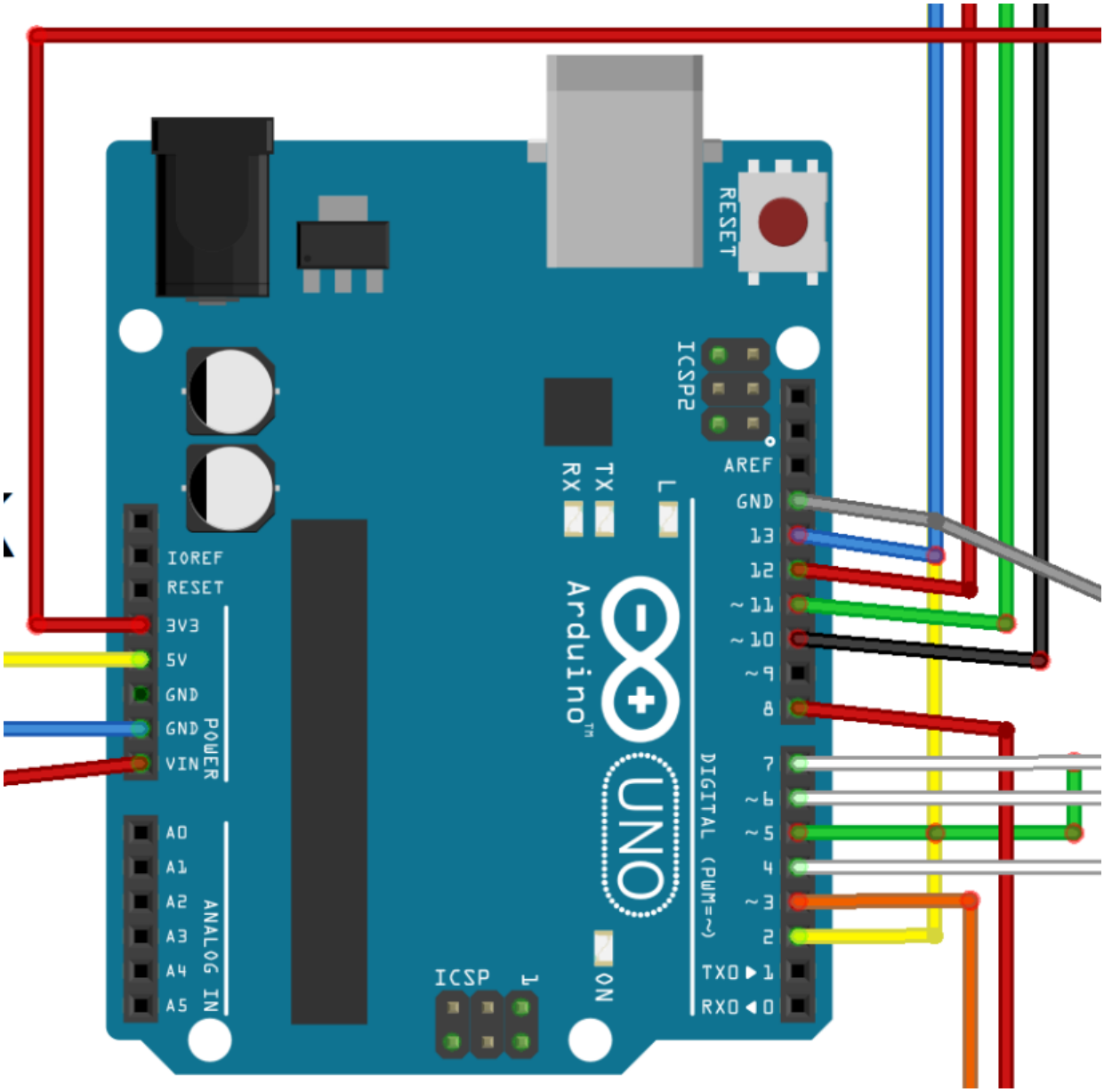

Figure 3: Arduino Detailed View

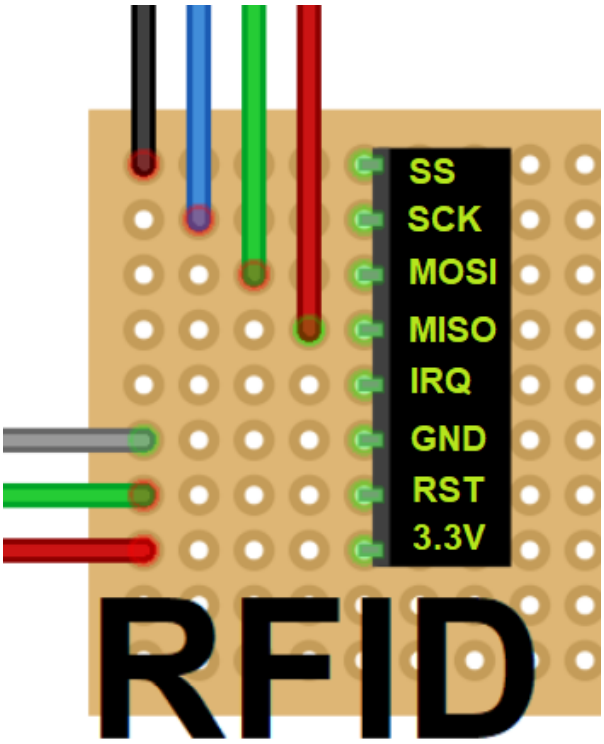

Figure 4: RFID Reader Detailed View

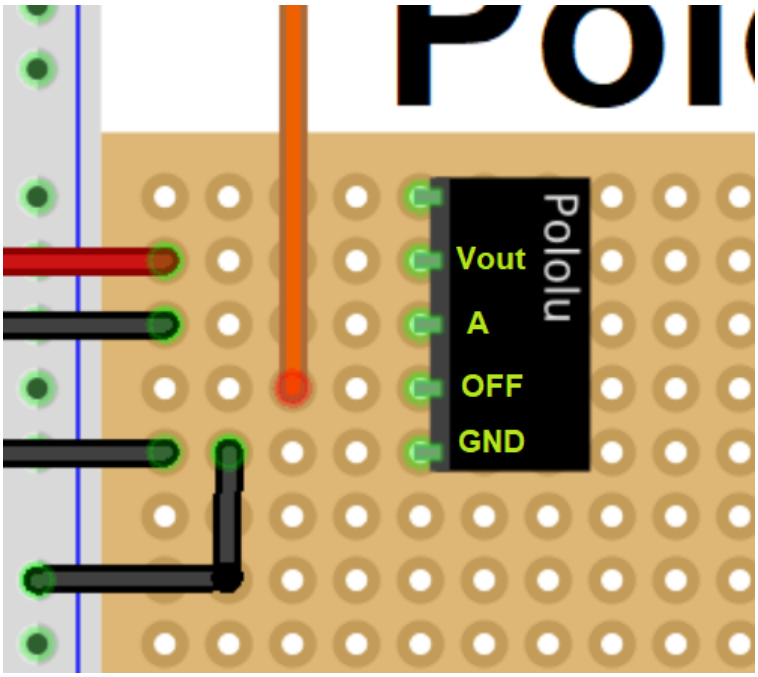

Figure 5: Pololu Chip Detailed View

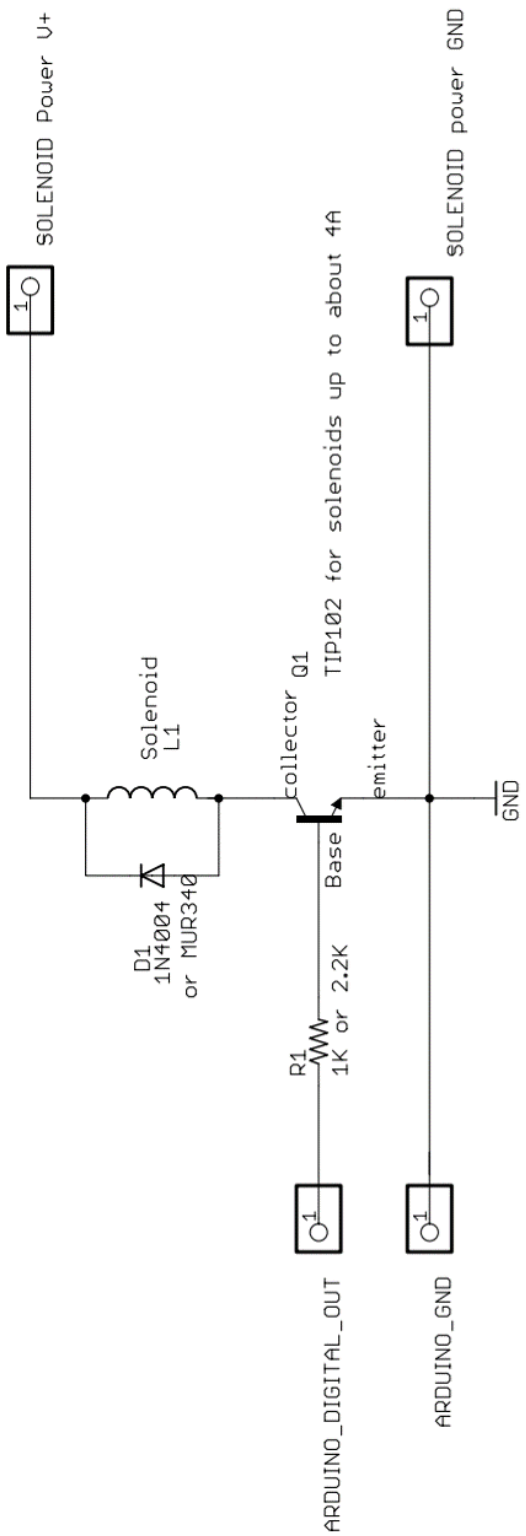

Figure 6: Solenoid Power Circuit

4.2 Wire Color Designations

- **RED**: Active (12V)
- **BLACK**: Ground
- **GREEN**: External Arduino Connections
- **YELLOW**: Internal Arduino Connections
- **BLUE**: Solenoid Ground

4.3 Circuit Description

There are five electronic components connected to the Arduino.

- Pololu Power Switch: Connect the power source to Vin and use Vout to power any other electronic component. Connect a push button from A to GND to enable the button in momentary mode. Connect pin 3 of the Arduino to OFF to enable the circuit to timeout.
- Solenoid: A TIP 120 is used to control solenoid operation by connecting the base to pin 8 of the Arduino.
- RGB LED: Two LEDs are connected in parallel in order to display information to the user. One for the front of the locker and one for the back.
- Push Button: A button is connected to the 5V output and pin 2 of the Arduino. This button is used to let administrators add and delete keys.
- MFRC522 RFID Reader: The RFID module is connected to the Arduino as show in the diagrams above.

4.4 Ordering Additional RFID Tags

https://www.amazon.com/Smart-Keyfobs-Keychain-Mifare-Arduino/dp/B01N06OLMJ

5 Software

The software was written in C++ to run on an Arduino Uno.

5.1 Architecture Overview

The software models a finite state-machine with the following states:

0. **NORMAL**:
   In this state, the Arduino will continuously look for new RFID tags detected by the MFRC522 chip and test it against the registered key list in EEPROM. It will then unlock the locker if the key is registered.

1. **BUTTON PRESSED**:
   This is the *minor state*, where the Arduino keeps track of how long the configuration button has been pressed. While the button is being held, **no operation can be performed until the button has been released**.

2. **DELETE**:
   This is a transient state that is triggered when the Arduino is in the **INTERMEDIATE** state for longer than 5s. In this state, the EEPROM is purged of its key list, that is, once this state is triggered **no MIFARE key will work as none of them are registered**.

3. **ADD**:
   This is a blocking state, i.e., we cannot exit this state unless we press the button again. In this state, any tag that is read will be registered, i.e., the tag's UID will be added to the EEPROM, if it is not already registered.

4. **TRANSITION**:
   This state acts as a transition between the **ADD** state and the **NORMAL** state. In this state, a button release triggers transition to **NORMAL** state.

The finite state-machine, with the transition events, can be seen in figure 7.

5.2 Source Files

Visit https://github.com/CSE-453-AutomatedLocker (our GitHub repository) to view and download all code related to this project. The comments in the code describe the function of important variables and subroutines.

5.2.1 `AutomatedLocker.ino`

This file is the main entry-point for the Arduino code, and contains the definition for the state machine. The two main functions in this file are the `setup` and `loop` functions, which serve the following purposes:

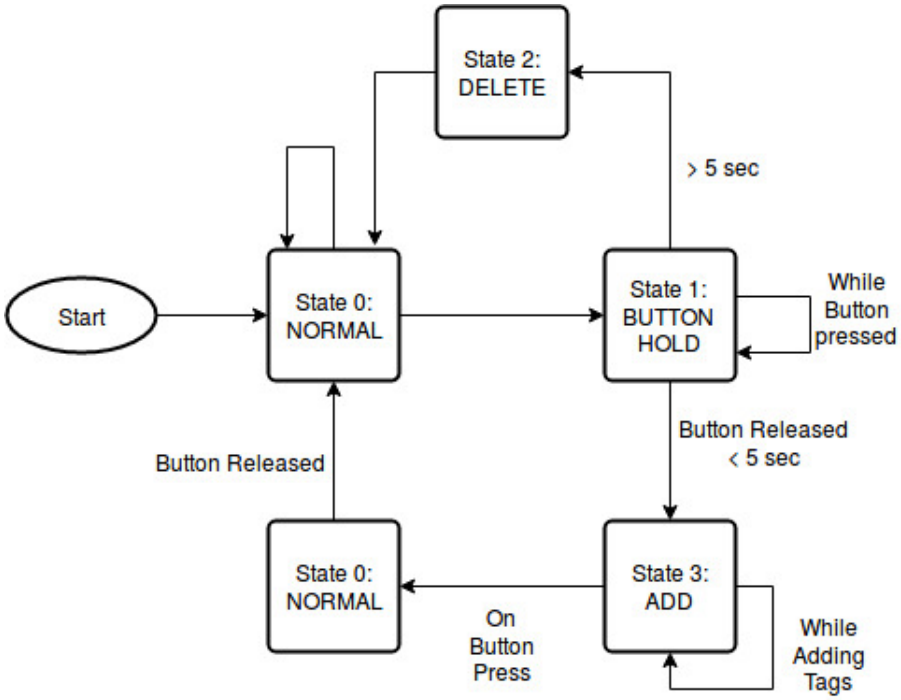

Figure 7: The state machine transitions for the locker software

- `void setup()`: This is called on program startup and is used to initialize GPIO peripherals, like the RFID module, push-buttons, and switch-off trigger.
- `void loop()`: This is called periodically until the Arduino is switched off. This contains the body of the state-machine and calls the functions that handle each state.

5.2.2 `Button.(cpp/h)`

This provides the `Button` class, which allows reading from a momentary push button, and supports reading the amount of time the button has been pressed. In this class the relevant functions are:

- `Button::Button(int pin)`: The constructor for the `Button` takes in the pin the button is connected to as the argument.
- `int Button::read()`: This will read either a `HIGH` or a `LOW` from the button. The class is configured for reading `HIGH` as pressed and `LOW` as released.
- `long Button::getSec()` and `long Button::getMilli()`: These return the number of seconds and milliseconds the button is pressed for respectively.

5.2.3 `RGB_LED.(cpp/h)`

These provide the `RGB_LED` class, which controls the state of the RGD LED connected to the specified pins. The most relevant and used method in this class is:

- `void writeState(int r, int g, int b)`: This writes the specified value to he digital output pins specified in the constructor. This method is agnostic of the design of the RGB LED, i.e., the class doesn't require the LED to be common anode or cathode, and writes the value as is to the digital output.

5.2.4 `access_ctrl.(cpp/h)`

These provide a functional interface to the locker's automated components. This exposes the following functions:

- `int checkNewTag()`: This will check if the RFID module detects a new tag, and if so, checks if the tag is present in the EEPROM.
- `int addNewKey()`: This will, similar to the above function, detect a new tag, and if a new tag is present, add it to the EEPROM without duplication.
- `void lock()`/`void unlock()`: This will lock/unlock the door by actuating the latch solenoid.
- `long unlocked_ms()`: This will return the amount of time the locker has been unlocked, and -1 if it is locked.

5.2.5 `keys.(cpp/h)`

This file is used for functions that add, delete, and validate keys. The first byte stored in the EEPROM is the number of registered tags. All subsequent bytes are framed as 4 byte tags.

- `bool addKey(byte tag[])`: This function adds the 4 byte tag to the Arduino's EEPROM.
- `void addKey()`: This function deletes all tags to the Arduino's EEPROM.
- `bool containsKey(byte tag[])`: This function searches the Arduino's EEPROM for the given 4 byte tag. Returns true if the tag is found, otherwise false.

5.3 Additional Software Flowcharts

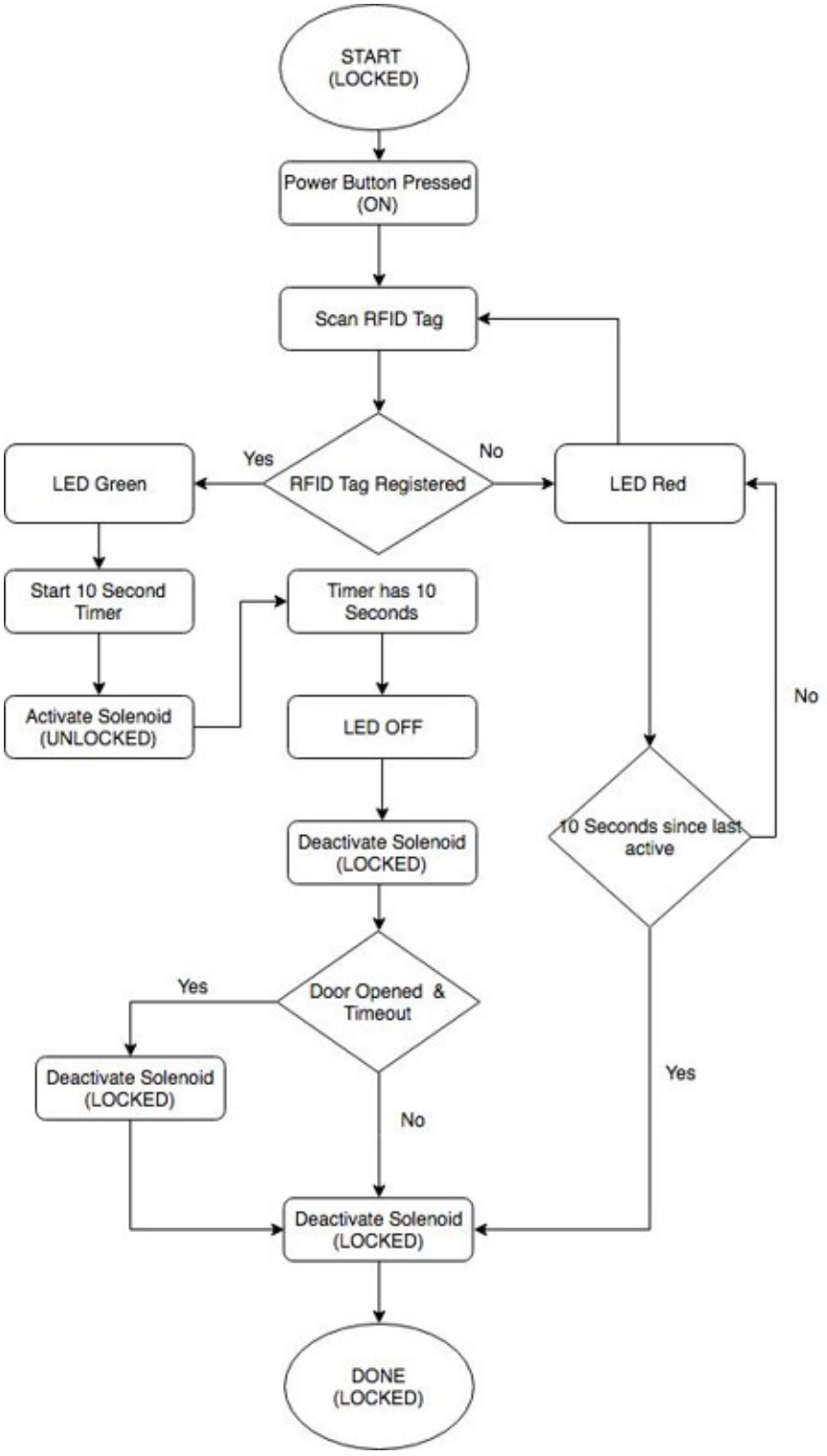

Figure 8: Normal Operation (Awaiting RFID Tag Scan to Unlock, LED Blue)

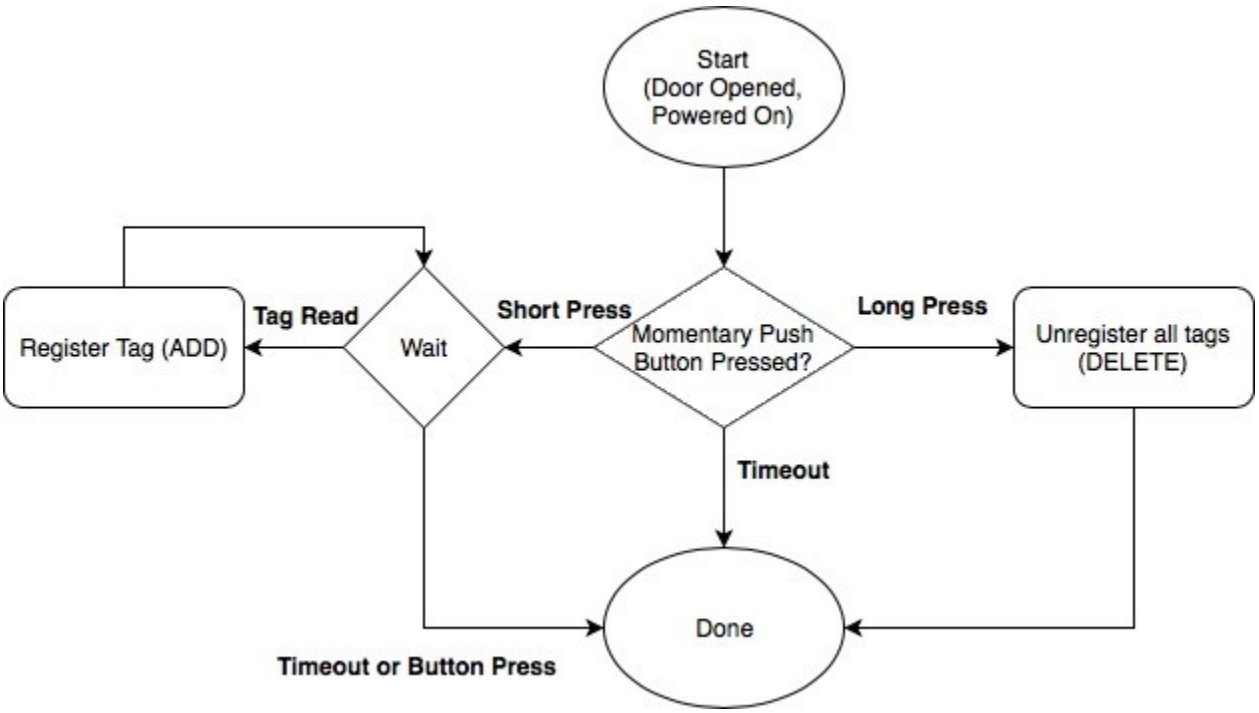

Figure 9: Registering/De-Registering RFID Tags (Long Press = 5 sec.)

6 Mechanical Design

6.1 Physical Modifications to the Locker

- A hole in the middle of the locker door is roughly 2 and 1/8 inches in diameter to accommodate the 60 mm Arcade Push Button.
- The side of the locker door has holes and bolts (6-32 x 4) that hold the 3D-printed mount for the lock-style solenoid.
- There are 4 holes on the face of the locker door which holds (using 4-40 bolts x 4) the 3D-printed housing which houses the RFID Reader and LED. An even larger hole accompanies these four which is for the wires that are routed to the RFID Reader and RGB LED.
- Four more holes (to accommodate 4-40 bolts x 4) are in the middle of the locker for the enclosure for the yellow button/RGB LED in the back of the locker door.
- A hole was drilled on the left side of the locker door (when facing the front) for the mechanical override key hole (19 mm diameter).
- A small cut (see Figure 10) (roughly 3/4 inch x 1/2 inch) to the tabs that are protruding from the side of the locker door was made to allow the solenoid and solenoid housing to move up and down when the silver handle on the outside of the locker is moved up and down. Note that this movement is dependent on the mechanical key override being in the unlocked state (to allow for the movement).
- Two 1/2 inch diameter Neodymium magnets were super glued to the locker frame. This was done to keep the door more tightly closed to prevent the solenoid latch from contacting the metal of the locker door, which had caused too much friction for the solenoid to function properly.

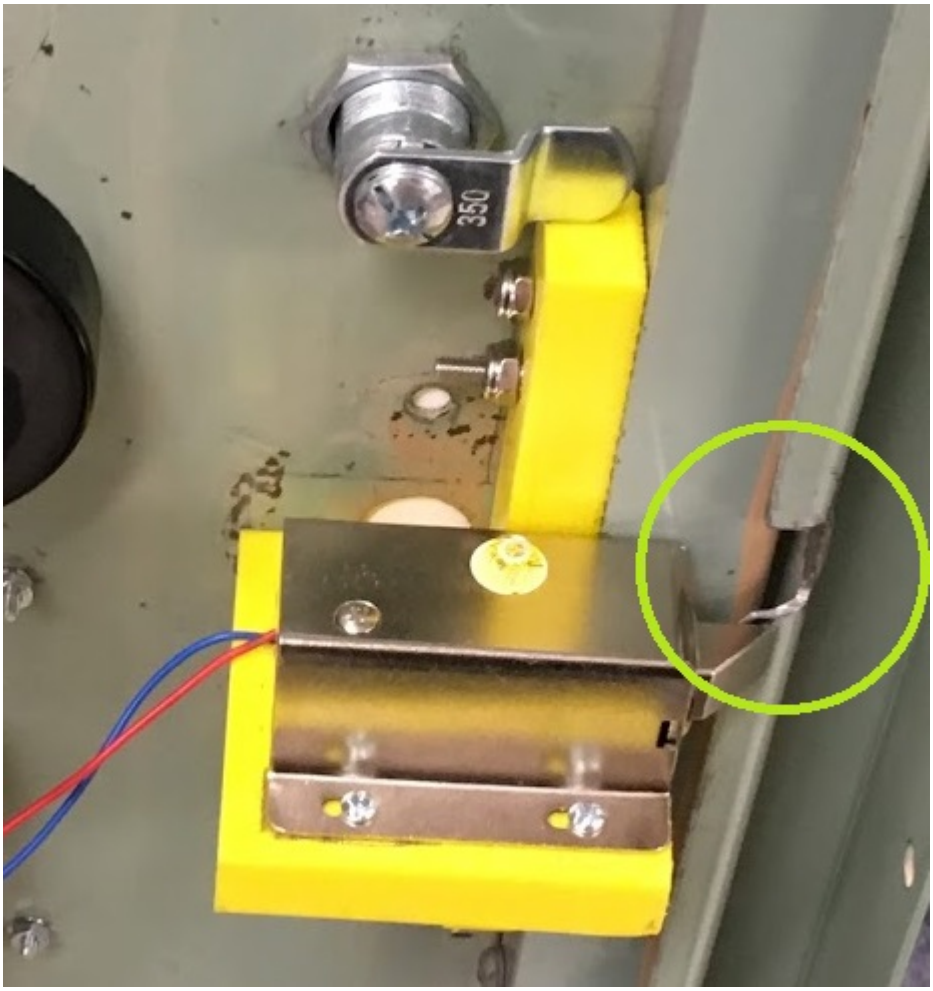

Figure 10: Small Cut Modification for Solenoid

6.2 Other Modifications

- Three holes were drilled into the solenoid housing to accommodate 4-40 bolts x 3 that are used to hold the solenoid in place on the housing.

6.3 Battery Cradle (Inside Locker)

CAUTION: Note that the battery cradle is NOT the same as the battery charger used to charge the Dewalt 12V Li-ion batteries! Adhere to the caution statement printed on the battery cradle at all times.

The battery cradle is a modified Dewalt battery charger (model no. DCB115). All pre-existing electronics from the battery charger were removed and discarded with the exception of the battery connector pins, which were kept. The battery cradle acts as a dock for the Dewalt 12V DC Li-ion rechargeable battery (model no. DCB127) that powers the system. On the inside of the cradle resides the Arduino Uno and a circuit board containing the Pololu circuit board, the TIP120 high-power BJT transistor with heatsink, 3 x 220 Ohm resistors, 1 x 10k Ohm resistor, the 1N4004 Diode, and all applicable wired connections as shown in figure 2.

Figure 11: Battery Cradle (Located Inside Locker)

- **10**: Battery Cradle
- **11**: Battery Connectors

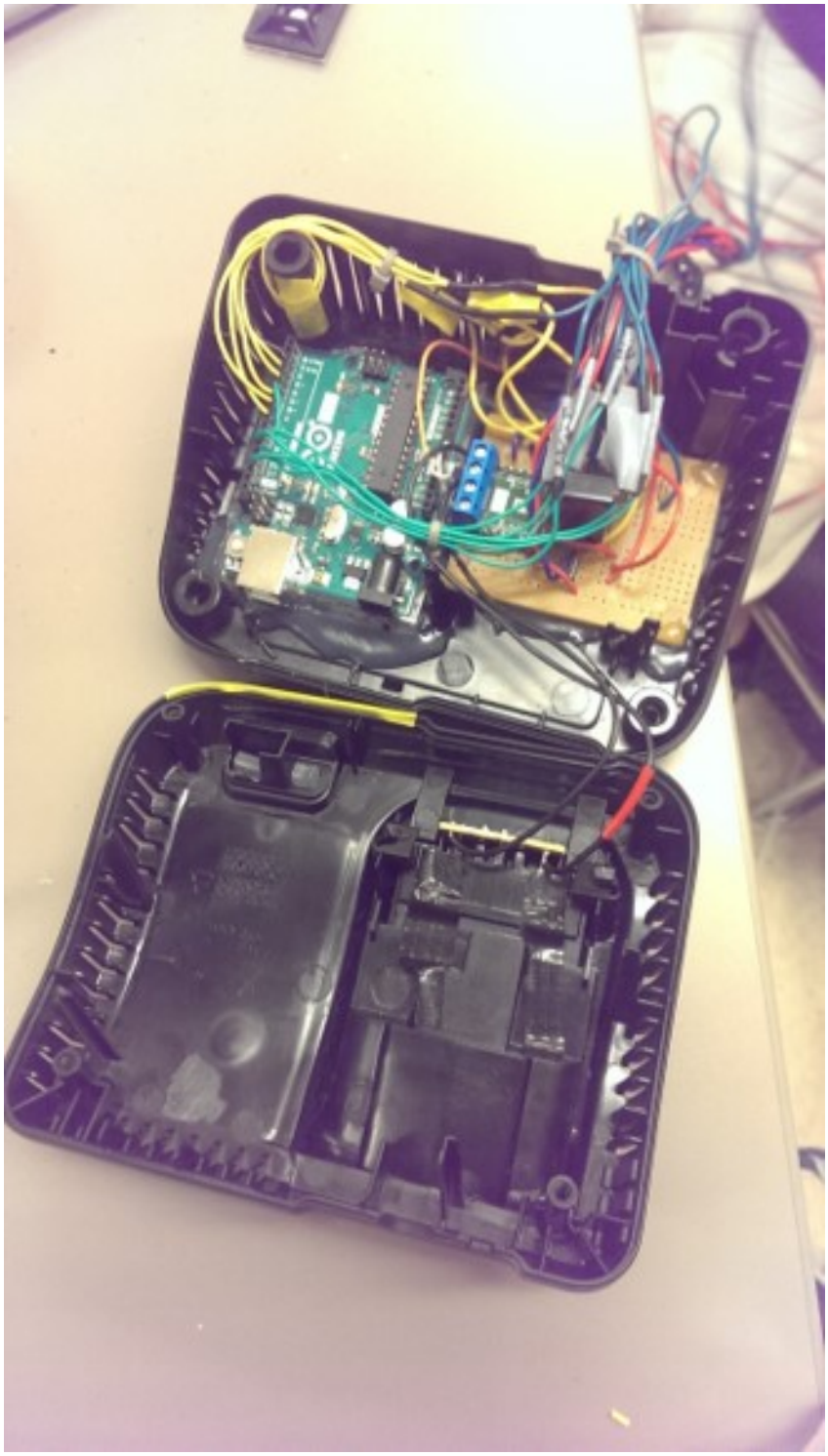

Figure 12: View Inside Battery Cradle

6.4 3D-Printed Parts

Visit https://github.com/CSE-453-AutomatedLocker (our GitHub repository) for CAD files of all 3D Designs.

- Large enclosure base and lid to house RFID reader and RGB LED
- Small enclosure base and lid to house yellow push button and RGB LED
- Solenoid housing
- Screen protector for Automated Locker 2017 team's design in Alden, NY high school

7 Troubleshooting

1. If you press the power button and the RGB LED does not illuminate within 10 seconds, it either means that the battery is dead, the Pololu Pushbutton Power Switch Chip is compromised, the Arduino is damaged, or the RGB LED is broken.

2. If the internal push button does no function properly, the Arduino or the button itself is damaged.

3. If scanning a registered tag does not unlock the locker, check to see if the RGB LED is changing colors. If the RGB LED flashes green when scanned, the solenoid is causing the problem. If the RGB LED flashes red, the tag is not registered. If the RGB LED does not change at all then the RFID reader is broken.

8 Safety & Caution Statements

PRODUCT SAFETY NOTICE: The use of a substitute replacement component which does not meet the same characteristics as the recommended replacement one, shown in the parts list in the Design Documentation, may create shock, fire, or other hazards. Do not allow any liquids to get inside the charger, as electric shock may result. If a component does get wet, don't turn the system on. Only use approved DeWalt Batteries of the exact model number listed in the Design Documentation with the provided DeWalt battery charger, as other types of batteries may burst causing personal injury and damage. Refrain from damaging the white box on the front of the locker door; equipment residing inside the box may also be damaged as a result. Do NOT attempt to make any modifications or repairs to the system without consulting a trained engineer.