Tokatuker Cabinet Digital Lock Testing

By: Bob Glicksman, Jim Schrempp; 3/28/20; updated 6/29/20

NOTICE: Use of this document is subject to the terms of use described in the document "Terms_of_Use_License_and_Disclaimer" that is included in this release package. This document can be found at:

https://github.com/TeamPracticalProjects/Wireless_IO_Board/blob/master/Terms_of_Use_License_and_Disclaimer.pdf

Contents

BACKGROUND.

The Maker Nexus RFID Access Control System development team was asked to evaluate electronic cabinet locks that could be used in conjunction with the RFID Access Control System. The basic idea is to secure (lock) a cabinet or tool-chest with one of these locks such that the locked device would be unlocked when the MN RFID Access Control System gives the "green light" for access to a particular area or item of equipment. In this manner, tooling and/or physical lock keys could be secured so that only people who are permitted to that location or equipment item can use it.

Jim Schrempp found a candidate device on the Internet:

https://www.amazon.com/Tokatuker-Electronic-Cabinet-Hidden-Drawer/dp/B075QF1VPR

Jim and Bob dissected the device and found a circuit board that provides the RFID token capability and that unlocks the lock. We unplugged the circuit board, exposing a connector with 4 wires: 2 wires supply battery (3 volt) power to the circuit board and the other two wires activate the lock mechanism. We found that by supplying battery power to the lock mechanism wires manually, we could lock and unlock the device.

We subsequently dissected the lock mechanism itself and found that the device was activated by a small 3 volt DC motor. A half turn of the motor releases the mechanism so that a locking ring can be inserted and removed. A spring loaded pusher forces the locking ring out of the mechanism when it is unlocked, thus forcing open the cabinet door or drawer. A reverse half turn of the motor locks the mechanism, allowing the locking ring to be inserted but not removed.

We then developed and tested a Particle Photon based breadboard that allows us to remotely lock and unlock the mechanism. We empirically determined the minimum time to pulse the motor to produce the half turn and which direction locks and which unlocks the mechanism. Unfortunately, we damaged the mechanism in disassembly and reassembly and the test unit does not operate reliably. Jim procured a second, factory fresh unit, and this report documents the results of tests and observations of the factory fresh unit.

OUT OF THE BOX TESTS.

The purpose of these tests is to establish a baseline operation of the unit as it is designed to work with its own, internal, electronics. I unpacked the unit, inserted batteries, programmed the master key card, and then tested the lock mechanism using the master key card. Here are my observations:

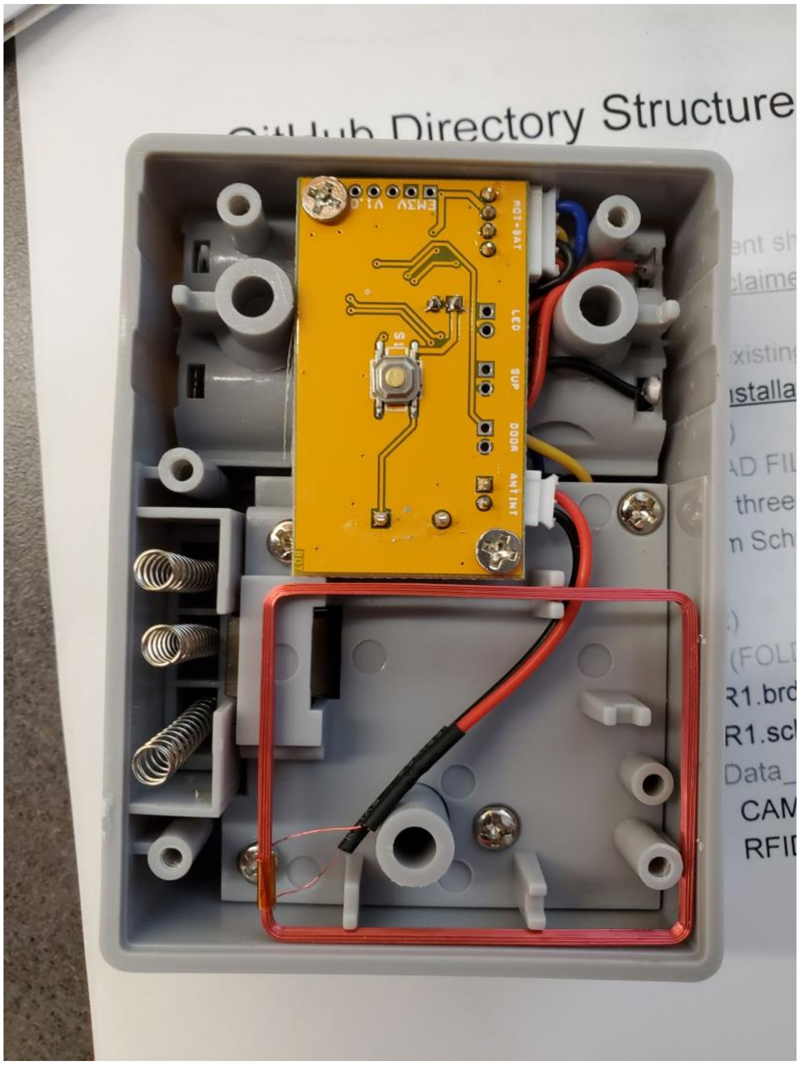
1. When the card is tapped, the device unlocks and stays unlocked for about 10 seconds, It then locks again. I don't know why the unit stays unlocked this long, since the locking ring must be inserted when the device is in the locked state (else it will just be ejected).

2. I inserted the locking ring and then activated the device a number of times. The ring was ejected each and every time that I activated the device. I tried wiggling the ring around each time I inserted it. In the damaged unit, I was able to wiggle the ring to different positions and cause it to bind up sometimes and not be ejected when the lock is tripped. However, when I wiggled the inserted ring in the new unit, it tended to reposition itself and it ejected every single time that I operated it.

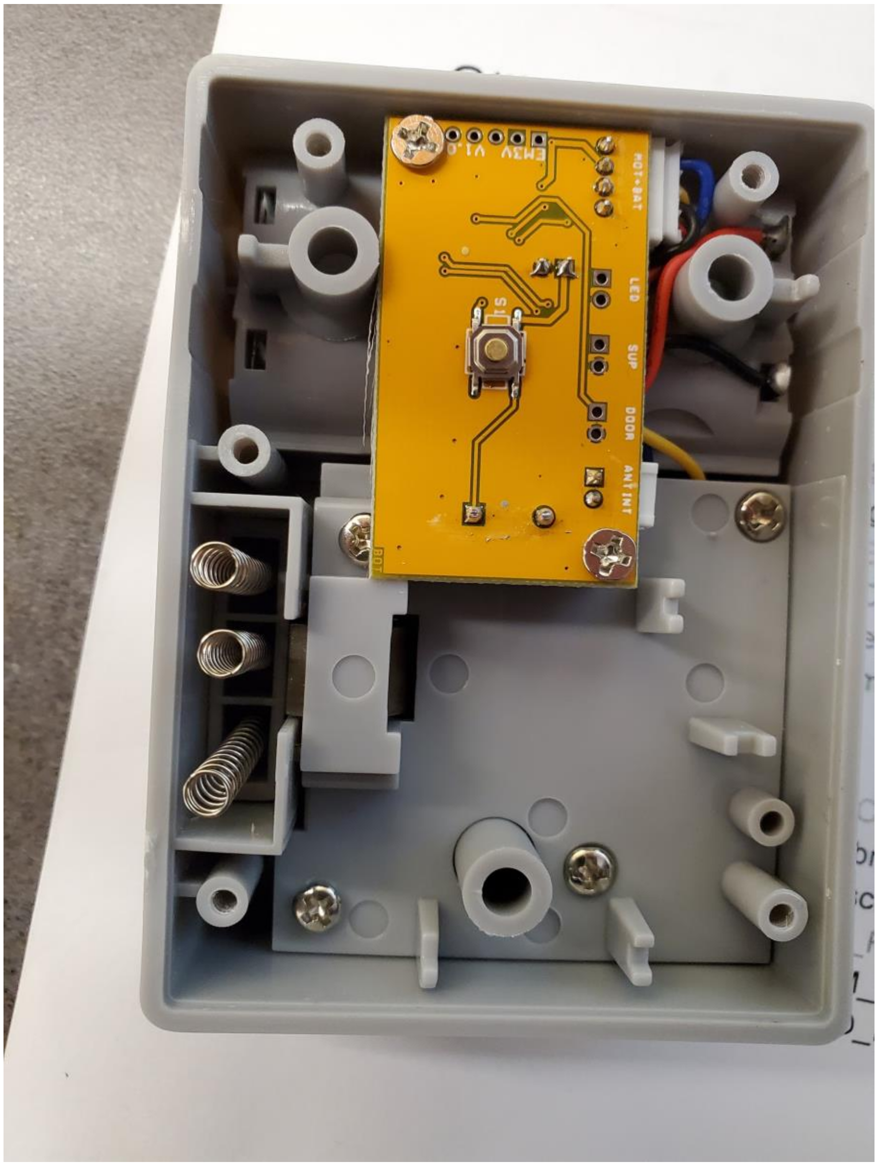
3. The unit beeps repeatedly during the 10 seconds between unlocking and locking. I assume that this is to let you know that you can't close the cabinet or draw during this time (you can close it but it won't lock).

BREADBOARD CONTROL TESTS.

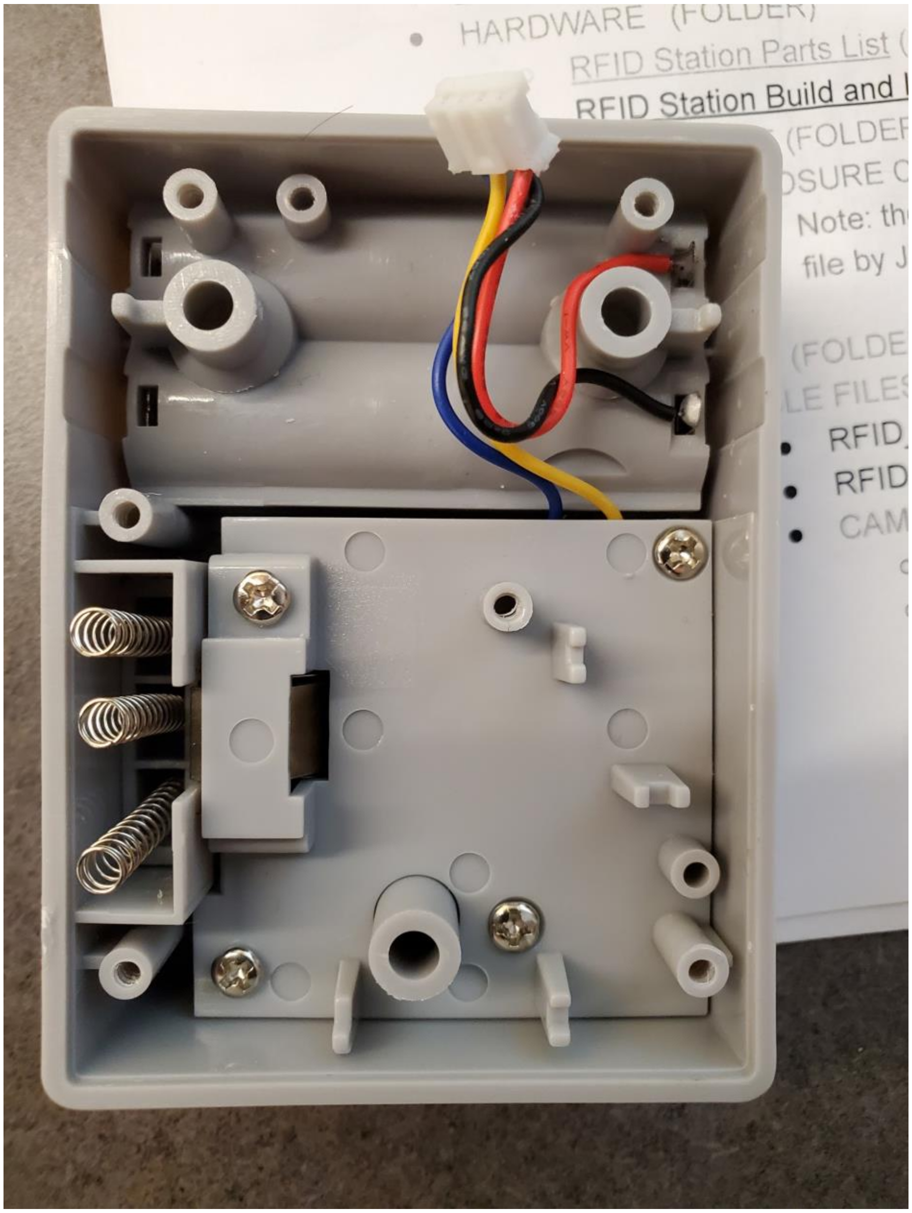
I removed the batteries and the back cover of the new unit, revealing the circuit board and antenna:



Next, I unplugged the RFID antenna from the circuit board and removed it:

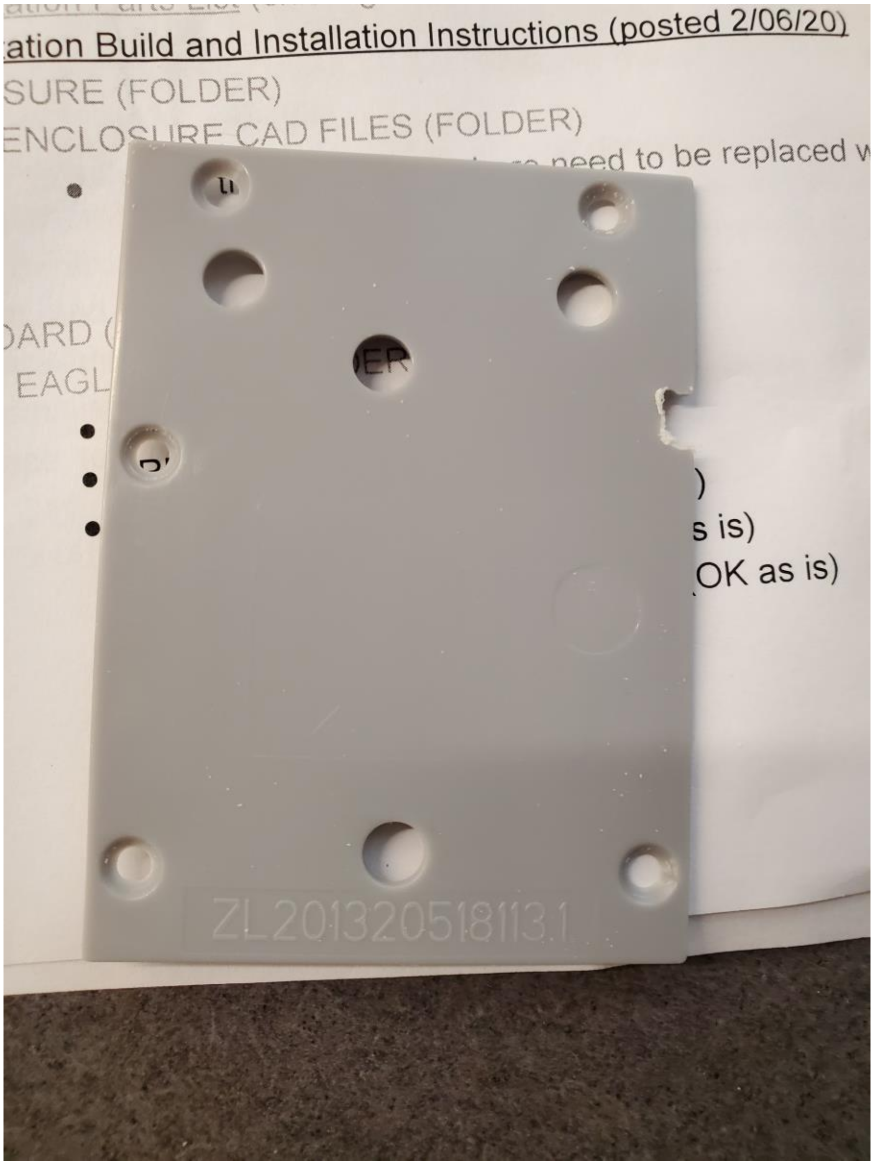


Next, I removed the circuit board and disconnected the remaining connector to it. This connector contains 4 wires - 2 for battery power to the circuit board and 2 for the motor:

- 

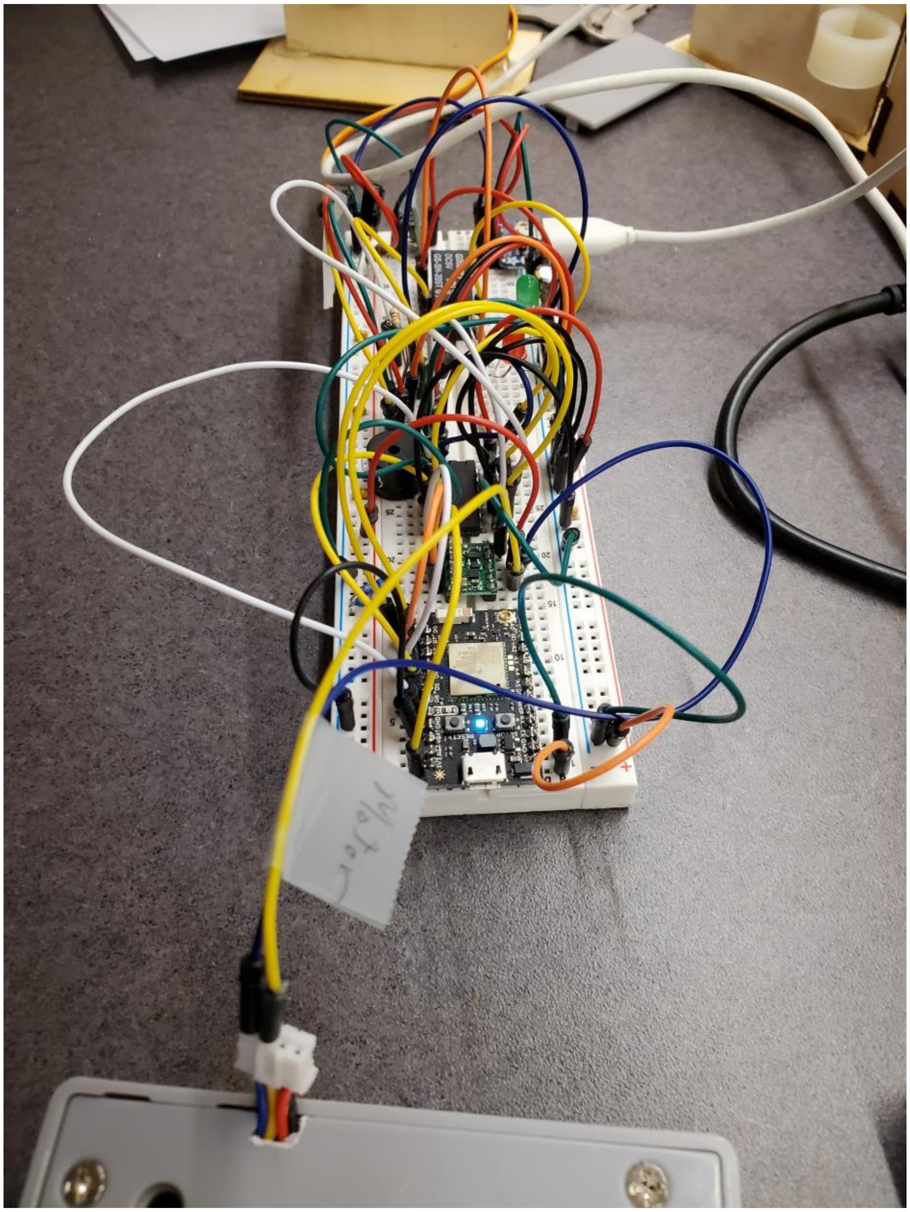
Note the 3 springs in the lower right hand side of the above photo. The back cover has three plastic protrusions that must mate with these three springs and the back cover must be replaced in order to compress these springs. These are the springs that eject the locking ring when the unit is unlocked. These springs, and the plastic piece below them, hold the mechanism open when the motor position is reversed to the lock position so that the locking ring can then be re-inserted and locked.

In order to replace the back cover while exposing the cable with the motor wires on it, I filed a notch into the back cover, as shown in the photo:


ation Build and Installation Instructions (posted 2/06/20)
SURE (FOLDER)
ENCLOSURE CAD FILES (FOLDER)
need to be replaced w
DARD (
EAGL
s is)
OK as is)
ZL201320518113.1

I then replaced the back cover, carefully running the cable wires out through the notch, and carefully aligning the protrusions on the back cover with the three springs:

Finally, I connected the motor driver on the breadboard to the blue and yellow wires on the cable connector. The other two wires are left unconnected.



I can now lock and unlock the device by using the Particle console to trip the mechanism. When the Particle cloud function tripLock() is called (with any data as the argument, even a null string), the unit unlocks for a preset time and then locks. The function that operates this unit is:

```
const int pulseTime = 30;   // motor activation time (ms)
const int unlockTime = 2000;    // unlock time for door to swing open

int tripLock(String command) {  // cloud function to trip the lock
    unlock();
    delay(unlockTime);
    lock();
}   // end of tripLock()

void unlock() { // unlock the latch
    digitalWrite(directionPin, HIGH);   // motor direction is unlock
    digitalWrite(runPin, HIGH);
    delay(pulseTime);
    digitalWrite(runPin, LOW);
}   // end of unlock()

void lock() { // lock the latch
    digitalWrite(directionPin, LOW);   // motor direction is lock
    digitalWrite(runPin, HIGH);
    delay(pulseTime);
    digitalWrite(runPin, LOW);
}   // end of lock()
```

The complete test code (locktest.ino) can be found at:

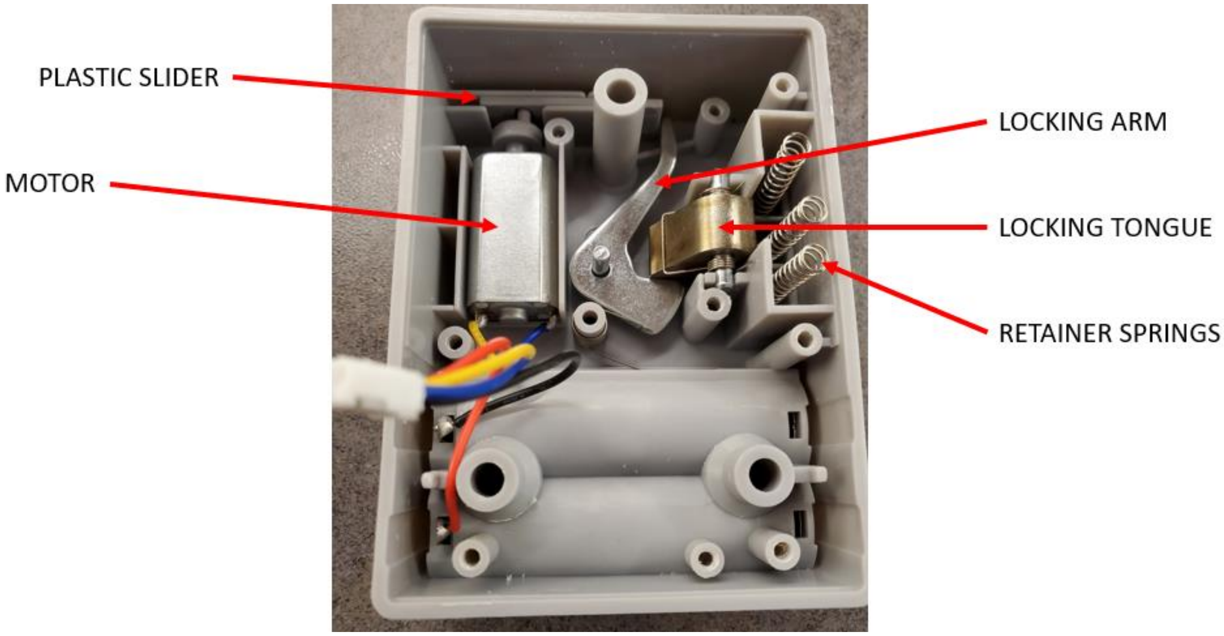
https://github.com/TeamPracticalProjects/Wireless_IO_Board/tree/master/Test_Firmware

Note that the 30 ms time to pulse the motor to run a half turn was empirically determined. As a follow-on exercise, I pulsed the motor for 20 ms and found that it did not travel a complete 180 degrees. Pulsing the motor for 25 ms barely made the motor turn 180 degrees; in fact, it seemed a bit short. Therefore, it seems that 30 ms is the correct motor pulse timing.

The 2 second (2000 ms) time between unlock() and lock() was arbitrary on my part. The factory unit's operation leaves 10 seconds between unlock and lock, but I don't know why it has to be so long.

LOCK MECHANISM STUDY AND PULSE TIME TESTING.

I opened the covering of the mechanical parts of the old unit and took a photo of the mechanism:



There is a small 3 volt DC motor in the upper left side of the unit. The motor has a cam that linearly moves a plastic slider piece, shown above the motor. The plastic slider pushes the shiny metal locking arm in the center of the photo. When the slider is pushed to the right via the motor, it allows the brass locking tongue to spring back into the locking arm's cavity. When the motor pushes the slider back to the left (½ turn counter-clockwise), a spring on the locking arm wants to rotate the locking arm counter-clockwise so that it prevents the locking tongue from being pushed back, thus locking the mechanism.

The three springs shown in the upper right of the photo push a plastic retainer (under the springs) down toward the front of the unit. When the motor/slider is moved to the lock position, the locking arm wants to rotate counter-clockwise, but it can't because the retainer prevents the locking tongue from springing to the right. However, when the locking ring is inserted into the slot in the front of the unit, it pushes this retainer to the back (compressing the three springs). Since the locking ring is hollow in the center, the locking tongue is now free to spring to the right and the locking arm can then rotate counter-clockwise and thus prevent the tongue from being pushed back to unlock the ring.

When the motor is activated to make a ½ turn clockwise, it pushes the plastic slider to the right which then pushes the locking arm to rotate clockwise. The spring loaded retainer has, all the while, been trying to force the locking ring out of the unit, but it couldn't because the locking tongue could not move back to the left. But now that the motor has moved the plastic slider to the right, rotating the locking arm clockwise, the ring can be ejected from the unit pushing the locking tongue back into the cavity of the locking arm.

This very clever mechanism requires only a small current to the small DC motor (in the right direction) to operate a much stronger mechanism. The stronger forces of the mechanism come from the various springs in the unit; specifically, the three large springs behind the plastic ring retainer. Energy to compress these springs comes not from the motor or the batteries but rather, from the user closing the cabinet door, thus forcing the locking ring into the retainer and compressing the springs.

Takeaways:

1. The unit locks via turning the motor ½ turn counter-clockwise, so that the plastic slider is all of the way to the left.
2. The unit unlocks via turning the motor ½ turn clockwise, so that the plastic slider is all of the way to the right.
3. When the unit is unlocked, the locking ring is forced out of the unit via the retainer and three large springs and the door/draw is pushed open. The plastic retainer then keeps the unit open by holding the tongue back into the cavity of the locking arm.
4. The locking ring, and thus the door/draw, cannot be closed until the motor is rotated back to the lock position.
5. Once the motor is rotated back to the lock position, the locking ring may be inserted into the unit and it will lock, until the motor is again activated to unlock the unit.

Major takeaway: the motor must be pulsed in the clockwise direction long enough to make the ½ turn (180 degrees) counter-clockwise to open the unit. The motor power is only on during this pulse time. After a short period of time, the motor must be pulsed again, this time with opposite DC polarity, to move it ½ turn counter-clockwise; back to the lock position. The motor is again unpowered and the cabinet door/draw may be closed and will automatically lock.

CONTROLLER DESIGN.

We now know that the motor must be pulsed with 3 volt DC power for 30 ms -- pulsed counter-clockwise to open the lock and eject the locking ring, and then back clockwise to lock the mechanism and enable the locking ring to be inserted and stay locked into the unit. The batteries and the internal RFID circuit board and antenna are removed and discarded. One white connector in the unit's PC board is for the antenna, which is removed. The other connector is 4 pin connector, as follows:

- Orange: battery + to the PC board
- Black: battery - to the PC Board
- Yellow: motor + from the PC board
- Blue: motor - from the PC Board.

Only the yellow and blue connections are activated with a 3 volt, 30ms pulse -- +/- for unlock and -/+ for lock.

The controller uses a Particle Photon, a 5 volt to 3.3 volt low dropout voltage regulator (LD1117-3.3), and a low voltage H-bridge motor controller (Pololu DRV8838). The schematic can be found at:

https://github.com/TeamPracticalProjects/Wireless_IO_Board/blob/master/Hardware/PCB/Wireless_IO_Board_Rev_1_PCB_sch.pdf

Note: this schematic is for a general purpose I/O board that supports this project as well as other I/O types (buzzer, LED, relay, etc.)

Two I/O pins from the Photon control the motor controller: One for the direction (DIR - Photon pin D2) and one is for the motor power pulse (RUN - Photon pin D3; to pulse the motor with power for 30 ms). The motor power comes from the 3.3 volt power supply, while the motor controller logic is powered via the Photon’s 3.3 volt power output. Particle code fragment to operate the motor was presented above.