Ryan Antcliff
Dr. Mohammed
CSE 321
7 December 2025

Phase 3: Automatic Door Lock

Overview

The goal of this project is still the same as in Phases 1 and 2: build a safety-aware automatic door lock that can unlock the door when an authorized person approaches from the outside, and also unlock from the inside when someone walks up to the door. The system looks at multiple pieces of information at once, including presence (ultrasonic sensors), identity (RFID), door status (a magnetic contact switch), and then uses a servo motor to actually turn the deadbolt.

By Phase 3 I've taken the prototype from "mostly working on a breadboard" to a cleaner, more stable setup with proper schematics, organized code, and a documented GitHub repository that somebody else could realistically follow. The final design still uses two separate controller units (inside and outside), talking over a pair of HC-05 Bluetooth modules with a very simple command protocol.

GitHub repository (code + documentation + images):

https://github.com/ryantcliff/AutomaticDoorLock

What Changed Since Phase 2

In Phase 2 I had the basic behavior working: the outside board could see a person, read an RFID card, and send an unlock command; the inside board could receive that command and move the servo while checking the door sensor. For Phase 3 I focused on tightening everything up and removing things that were more experimental than useful.

One big clarification: even though some of my sketches include FreeRTOS headers and commented-out tasks, I never had FreeRTOS actually running on the hardware for the previous phases. My outside controller is an Arduino Uno-style ATmega328P clone, and once I added the RFID library, Bluetooth, and ultrasonic code, there just wasn't enough memory left to run FreeRTOS tasks reliably. I experimented with it, but it never made it into a stable build.

For Phase 3 I committed fully to a straightforward loop-based approach on both boards. The final code uses millis() timers, short non-blocking checks, and a simple state

machine instead of a RTOS. That's honestly more appropriate for the amount of work these boards are doing, and it let me clean up a lot of half-finished task code.

Other changes compared to Phase 2:

- I rewrote the main loops so sensor reads, Bluetooth handling, and state updates are clearly separated but still run in one loop() function per board.

- I tightened up the presence detection logic so that invalid distance readings immediately clear the "person detected" flag instead of leaving stale values around.

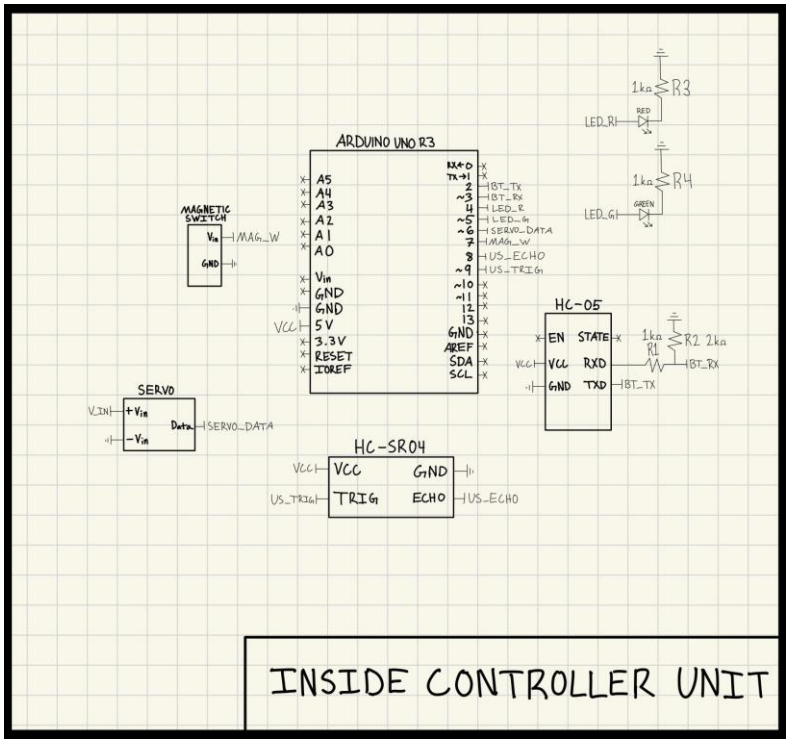
- I made the lock behavior more strict about the door actually being closed before it's allowed to move back to the "locked" state.

- I produced proper inside and outside schematics and added them, along with build photos, to the repository.

Finished Circuit Design

In earlier phases I only had partial wiring sketches. For Phase 3 I built proper circuit diagrams for both units so that someone could reproduce the hardware without guessing.

Inside Controller Unit:



The inside controller is an Arduino Uno R3. It drives the servo, monitors the inside ultrasonic sensor, reads the magnetic door switch, and listens to the HC-05 slave module:

- Servo motor connected to a PWM pin, with 5 V and GND from and external power supply.

- HC-SR04 ultrasonic sensor on dedicated TRIG and ECHO pins to watch for someone standing just inside the door.

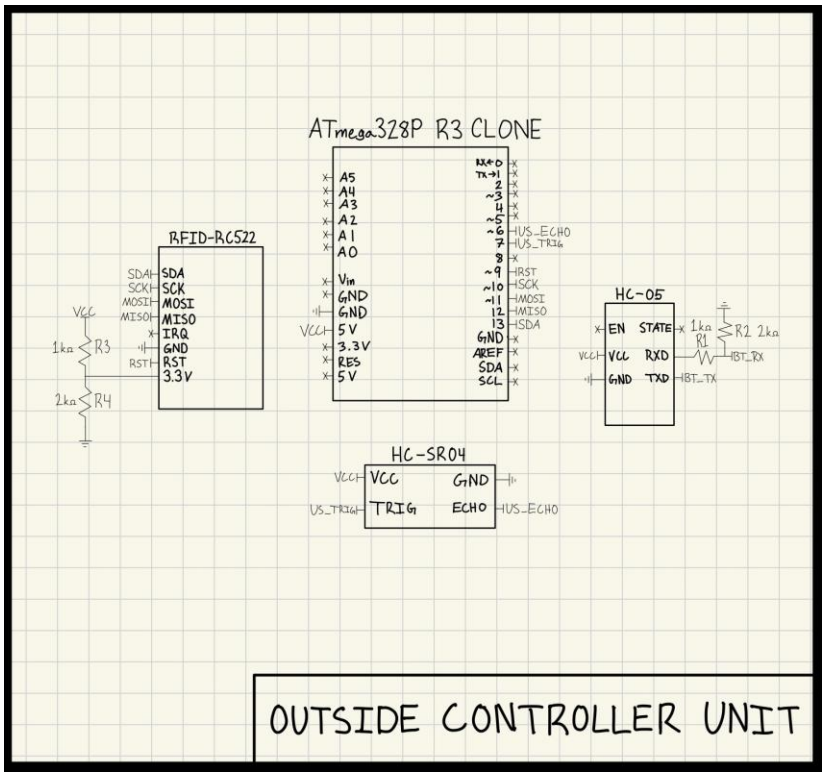
- Magnetic contact switch wired as a digital input; this goes active when the door is physically closed.

- HC-05 Bluetooth module wired to pins 2 and 3 on the Arduino, with a voltage divider on RX.

- Two LEDs (lock/unlock) with resistors, used to show the current lock state.

This unit is responsible for safety. It won't lock if the door is open, and it uses its own ultrasonic reading so it doesn't engage the deadbolt on somebody standing in the doorway.

Outside Controller Unit:



The outside controller is an ATmega328P R3 clone board. It handles authentication and outside presence sensing:

- RC522 RFID reader wired over SPI (MOSI, MISO, SCK, SDA, and RST).

- HC-SR04 ultrasonic sensor mounted facing outward to see people approaching the door.

- HC-05 Bluetooth module configured as the master, paired with the slave on the inside board.

This unit never touches the servo directly. Its only job is to decide whether the situation at the door is “unlock”, “lock”, or “do nothing”, and then send the appropriate single-byte command over Bluetooth.

Final Codebase and Bluetooth Protocol

The repository now has a clear structure:

- AutomaticDoorLock_Outside/AutomaticDoorLock_Outside.ino

- AutomaticDoorLock_Inside/AutomaticDoorLock_Inside.ino

- Phases/Phase1.pdf, Phases/Phase2.pdf, and Phases/Phase3.pdf

- ProjectPictures/ with hardware photos and build process shots

- README.md with setup instructions

The outside controller code does three main things in its loop:

1. It periodically reads the outward-facing ultrasonic sensor and sets a presenceDetected flag when distance is under PERSON_THRESHOLD (30 cm in the current code) and within a valid range.

2. It checks whether a new RFID card is present and, if so, reads the UID into a 4-byte array and compares it against an approved ID list in flash.

3. Based on those two pieces of information, it decides whether to send '0' or '1' over Bluetooth:

- '0' → valid card and presence detected → request unlock

- '1' → access denied → request lock / keep locked

- If the card is valid but no one is in front of the sensor, it logs that case and sends nothing, so the door doesn't unlock for a random card tap from far away.

The inside controller code maintains a simple state machine with states "locked" and "unlocked". In its main loop it:

- Reads the inside ultrasonic sensor and updates whether someone is standing near the door on the inside.

- Reads the magnetic switch to determine doorClosed.

- Listens for Bluetooth commands from the outside unit on SoftwareSerial.

- Updates the lock state and servo position based on both the incoming command and the local safety conditions.

When the inside board receives a '0', it logs the event, moves the servo to the unlocked angle, turns on the appropriate LED, and records lastUnlockRequest using millis(). When it receives a '1', it will only accept that as a lock request if the door is closed; otherwise it prints a message like "LOCK IGNORED - door open" and stays in its current state. There is also a timeout, where after being unlocked for longer than the configured interval, and once the door is closed and nobody is in front of the inside sensor, the state machine transitions back to locked.

Setup, README, and Repository Structure

The README explains required components, how to wire the two boards, and how to pair the HC-05 modules in AT mode as slave and master. It also includes a helper sketch for reading new RFID tag UIDs, and instructions for adding them to the approved list in the main outside sketch.

- Both .ino files are included in their own folders with clear names so they're easy to open in the Arduino IDE.

- I've kept Phase 1, Phase 2, and Phase 3 PDFs in a Phases/ directory as part of the documentation trail.

- A ProjectPictures/ folder contains images of the inside unit, outside unit, the overall system, and an intermediate build stage.

When someone clones the repo, follows the wiring and pairing steps, and uploads the two sketches, they should end up with the same behavior I'm demonstrating in the Phase 3 video.

Testing, Edge Cases, and Limitations

For the final phase I spent more time treating the system like a real product I might actually depend on in my apartment, instead of just an EE lab demo.

I tested:

- Valid vs. invalid cards, both with and without a person in front of the door, to confirm that only the “valid card + presence” case sends an unlock command.

- The situation where the door is open but a lock command is requested. The magnetic switch logic correctly blocks the servo from treating that as a normal lock and logs that the request was ignored.

- The automatic re-lock behavior after an unlock event, making sure that it doesn’t relock right away while someone is still in the doorway.

- Serial output from both boards to confirm that '0' and '1' are received correctly and that distance readings and state transitions match what I expect.

There are still limitations. The Bluetooth link is not encrypted, so this is not something I would trust as a front-door security system out in the wild. The ultrasonic sensors can still be a little noisy depending on the environment, even with tuned thresholds. And the mechanical setup for the servo and deadbolt is good enough for a semester project, but not hardened the way a commercial lock would be.

Future Work

- Adding a way to actually mount the system to a door.

- Adding power supplies to each unit so they’re not just plugged into my laptop.

- Adding at least some basic authentication or encryption on top of the HC-05 traffic.

- Providing a simple way to manage authorized RFID cards without reflashing the microcontroller (for example, an “admin tag” that lets you add or remove IDs).

- Adding a defined “power loss” behavior so the lock doesn’t end up in a random state.

- Replacing or augmenting the ultrasonic sensors with something less susceptible to noise and reflections.

Conclusion

Over the three phases I went from a rough idea about “not forgetting to lock my door” to a working two-board system that can automatically unlock for me when I walk up with an RFID card and relock once I’m inside and the door is closed. Phase 3 was about tightening everything up, with things like finalized schematics, cleaner code, a minimal but reliable communication protocol, and documentation that actually matches what’s running on the hardware.