Ryan Antcliff
Dr. Mohammed
CSE 321
16 November 2025

1. Project Scope and Current Prototype

The goal of this project is to build a safety-aware automatic door lock that can unlock the door when an authorized user approaches from the outside and can also unlock from the inside when someone walks up to the door. The final system will combine multiple factors, including presence detection (ultrasonic sensors), identity (RFID), door status (magnetic contact sensor), and a servo motor that turns the deadbolt. For Phase 2, I focused on getting a reliable hardware prototype running with the following features:

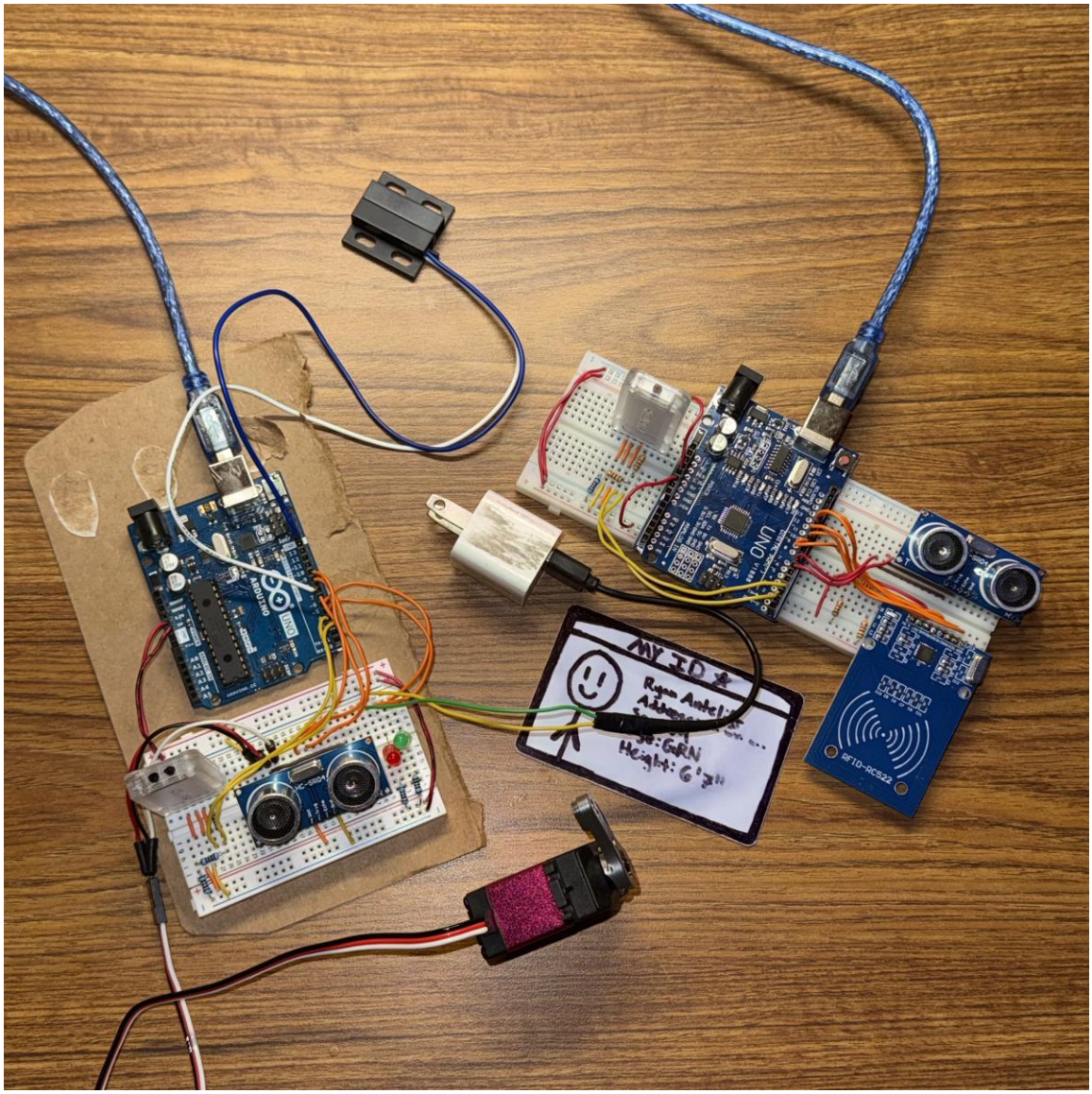
- Outside unit detects a person with an HC-SR04 ultrasonic sensor and scans an RFID card using an RC522 reader.
- If the card UID matches an approved list and a person is detected in front of the door, the outside unit sends an UNLOCK command over Bluetooth.
- The inside unit receives Bluetooth commands and decides when to unlock or relock the servo-driven deadbolt based on presence and the magnetic door-closed sensor.
- Both Arduinos continuously log sensor values and decisions over the serial port for debugging and testing.

1.1 Hardware Component Inventory

Core Components Used in Phase 2:

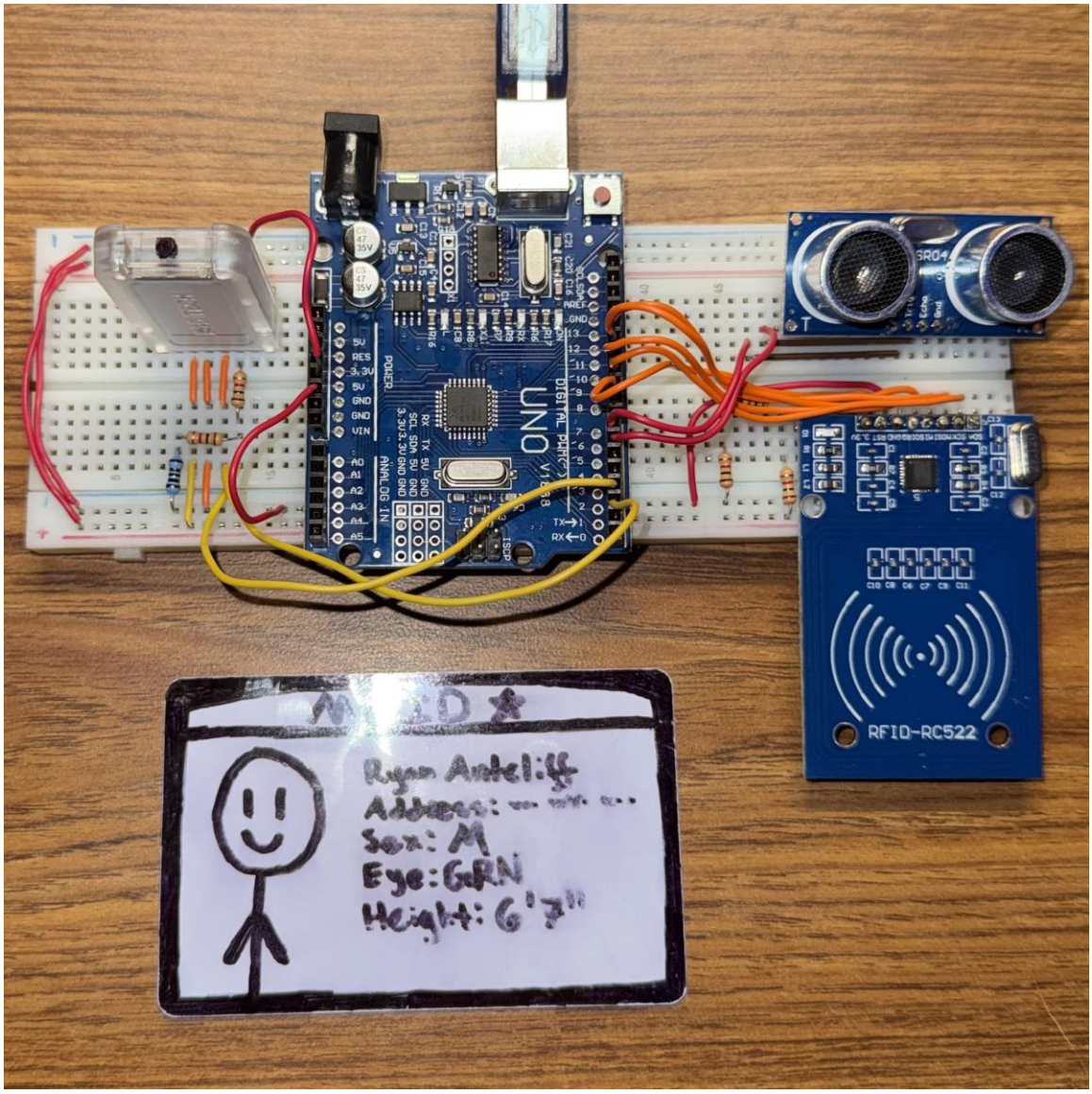
- Ultrasonic Sensor (HC-SR04), quantity 2 - one inside, one outside, currently the outside unit is wired to the master.
- Arduino Uno R3, quantity 2 - one dedicated to the outside unit (RFID + presence), one to the inside unit (servo + safety sensors). (The outside unit Arduino is a clone with MiniCore ATmega328P bootloader.)
- Bluetooth module (HC-05), quantity 2 - configured as master (outside) and slave (inside) for a simple two-device link.
- Servo motor - 20kg DC 4.8~6.8V high-torque hobby servo.
- Magnetic contact sensor - door-frame reed switch used to detect whether the door is fully closed before locking.
- RFID reader (RC522) + RFID card - used instead of Camera with Facial Recognition software because of project plan downscaling due to semester time constraints.

2. Hardware Component Prototype



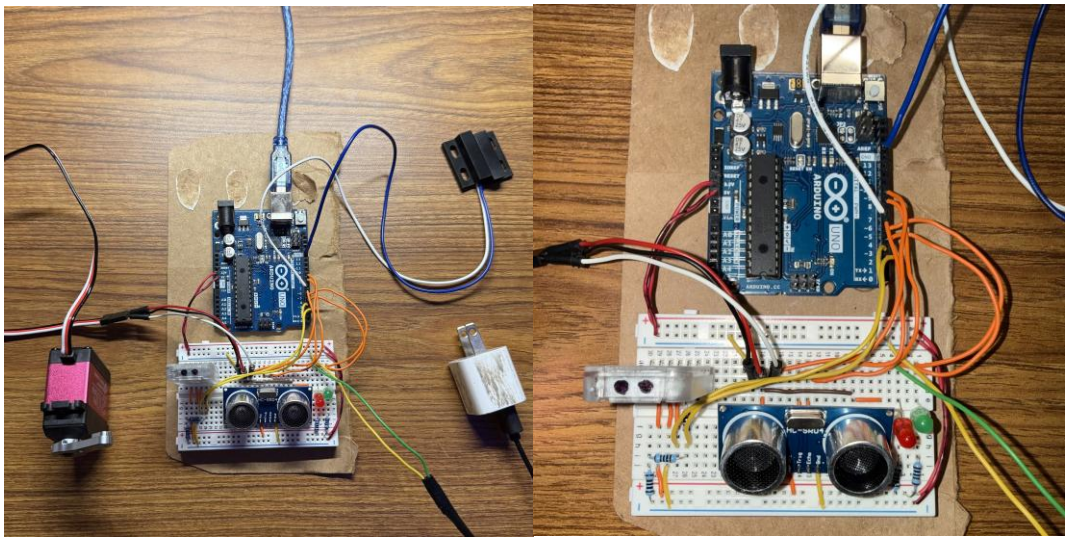
The hardware is split into two physical prototypes: an outside unit responsible for authentication and an inside unit responsible for physically operating the lock while enforcing safety constraints.

2.1 Outside Unit Prototype



The outside unit Arduino hosts the RC522 RFID reader and an HC-SR04 ultrasonic sensor mounted so that it points away from the door. When the distance drops below the PERSON_THRESHOLD (30 cm in the current code), the unit treats this as a person standing at the door. When a card is tapped, the UID is compared against a list of approved IDs stored in the code. If the ID matches and presence is currently detected, the board sends the character '0' over the HC-05 Bluetooth link to request an unlock.

2.2 Inside Unit Prototype



The inside unit Arduino controls the servo motor that turns the deadbolt, reads the magnetic contact sensor to confirm that the door is fully closed, and uses its own ultrasonic sensor to monitor people approaching from the inside. This unit receives single-character commands from the outside unit over Bluetooth: '0' to unlock and '1' to lock. It only relocks the door when the door is closed and no person is detected nearby, which prevents the lock from engaging on someone who is still in the doorway.

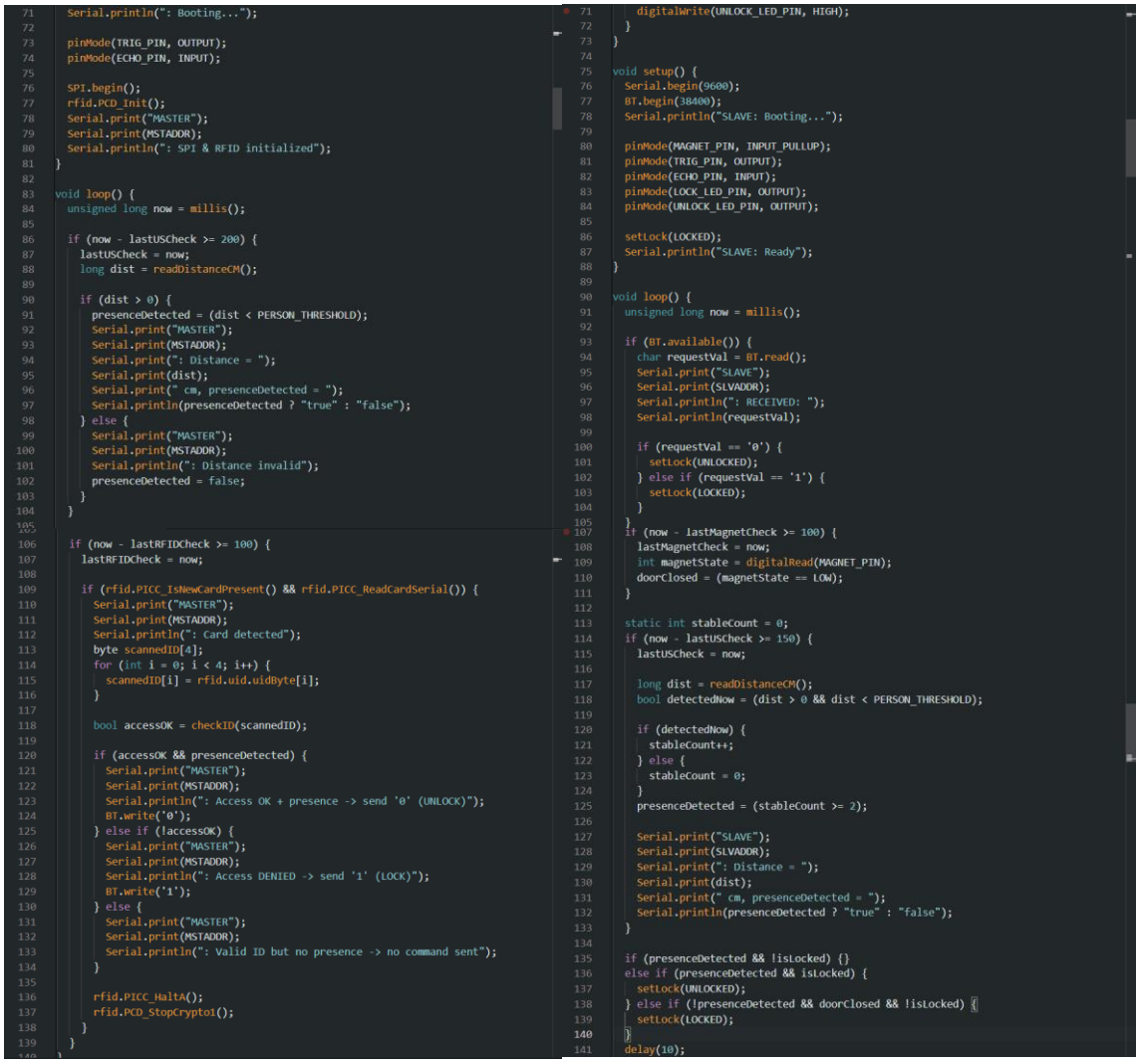
3. Software Component – Codebase

AutomaticDoorLock_Outside.ino

```ino
//Version 3
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <SPI.h>
#include <MFRC522.h>

const int SS_PIN    = 10;
const int RST_PIN   = 9;
const int TRIG_PIN  = 7;
const int ECHO_PIN  = 6;

const int PERSON_THRESHOLD = 30;

const char MSTADDR[]="_._+ADDR:14:3:50b06";

SoftwareSerial BT(2, 3);

MFRC522 rfid(SS_PIN, RST_PIN);

struct approvedID {
  const char* name;
  byte ID[4];
};

approvedID idList[] = {
  {"Ryan Antcliff", {0xB9, 0x95, 0x24, 0x12}}
};
const size_t NUM_IDS = sizeof(idList) / sizeof(idList[0]);

bool checkID(byte scannedID[4]) {
  for (size_t i = 0; i < NUM_IDS; i++) {
    bool matchID = true;
    for (int j = 0; j < 4; j++) {
      if (scannedID[j] != idList[i].ID[j]) {
        matchID = false;
        break;
      }
    }
    if (matchID) {
      Serial.print("Access Granted: ");
      Serial.println(idList[i].name);
      return true;
    }
  }
  Serial.println("Access Denied...");
  return false;
}

long readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(5);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(20);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 100000UL);
  if (duration == 0) return -1;
  long dist = (duration * 0.0344) / 2;
  return dist;
}

bool presenceDetected = false;
unsigned long lastUSCheck   = 0;
unsigned long lastRFIDCheck = 0;

void setup() {
  Serial.begin(9600);
  BT.begin(38400);
  Serial.print("MASTER");
  Serial.print(MSTADDR);
```

AutomaticDoorLock_Inside.ino

```ino
//Version 2
#include <Arduino.h>
#include <Servo.h>
#include <SoftwareSerial.h>

const int SERVO_PIN       = 6;
const int MAGNET_PIN      = 7;
const int TRIG_PIN        = 9;
const int ECHO_PIN        = 8;
const int LOCK_LED_PIN    = 5;
const int UNLOCK_LED_PIN  = 4;

const int PERSON_THRESHOLD = 30;
const int LOCK_ANGLE       = 10;
const int UNLOCK_ANGLE     = 90;

const char SLVADDR[]="_._+ADDR:14:3:50a37";

#define LOCKED   true
#define UNLOCKED false

SoftwareSerial BT(2, 3);
Servo lockServo;

bool isLocked     = true;
bool doorClosed   = false;
bool presenceDetected   = false;

unsigned long lastUSCheck     = 0;
unsigned long lastMagnetCheck = 0;

long readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(5);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(20);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 100000UL);
  if (duration == 0) return -1;
  long dist = (duration * 0.0344) / 2;
  return dist;
}

void setLock(bool lockState) {
  if (lockState == isLocked) return;

  isLocked = lockState;
  lockServo.attach(SERVO_PIN);

  if (lockState == LOCKED) {
    lockServo.write(LOCK_ANGLE);
  } else {
    lockServo.write(UNLOCK_ANGLE);
  }

  delay(300);
  lockServo.detach();

  if (lockState == LOCKED) {
    Serial.print("SLAVE");
    Serial.print(SLVADDR);
    Serial.println(": locked");
    digitalWrite(LOCK_LED_PIN, HIGH);
    digitalWrite(UNLOCK_LED_PIN, LOW);
  } else {
    Serial.print("SLAVE");
    Serial.print(SLVADDR);
    Serial.println(": unlocked");
    digitalWrite(LOCK_LED_PIN, LOW);
```



```
  Serial.println(": Booting...");

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  SPI.begin();
  rfid.PCD_Init();
  Serial.print("MASTER");
  Serial.print(MSTADDR);
  Serial.println(": SPI & RFID initialized");
}

void loop() {
  unsigned long now = millis();

  if (now - lastUSCheck >= 200) {
    lastUSCheck = now;
    long dist = readDistanceCM();

    if (dist > 0) {
      presenceDetected = (dist < PERSON_THRESHOLD);
      Serial.print("MASTER");
      Serial.print(MSTADDR);
      Serial.print(": Distance = ");
      Serial.print(dist);
      Serial.print(" cm, presenceDetected = ");
      Serial.println(presenceDetected ? "true" : "false");
    } else {
      Serial.print("MASTER");
      Serial.print(MSTADDR);
      Serial.println(": Distance invalid");
      presenceDetected = false;
    }
  }

  if (now - lastRFIDCheck >= 100) {
    lastRFIDCheck = now;

    if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
      Serial.print("MASTER");
      Serial.print(MSTADDR);
      Serial.println(": Card detected");
      byte scannedID[4];
      for (int i = 0; i < 4; i++) {
        scannedID[i] = rfid.uid.uidByte[i];
      }

      bool accessOK = checkID(scannedID);

      if (accessOK && presenceDetected) {
        Serial.print("MASTER");
        Serial.print(MSTADDR);
        Serial.println(": Access OK + presence -> send '0' (UNLOCK)");
        BT.write('0');
      } else if (!accessOK) {
        Serial.print("MASTER");
        Serial.print(MSTADDR);
        Serial.println(": Access DENIED -> send '1' (LOCK)");
        BT.write('1');
      } else {
        Serial.print("MASTER");
        Serial.print(MSTADDR);
        Serial.println(": Valid ID but no presence -> no command sent");
      }

      rfid.PICC_HaltA();
      rfid.PCD_StopCrypto1();
    }
  }
}
```

```
    digitalWrite(UNLOCK_LED_PIN, HIGH);
  }
}

void setup() {
  Serial.begin(9600);
  BT.begin(38400);
  Serial.println("SLAVE: Booting...");

  pinMode(MAGNET_PIN, INPUT_PULLUP);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LOCK_LED_PIN, OUTPUT);
  pinMode(UNLOCK_LED_PIN, OUTPUT);

  setLock(LOCKED);
  Serial.println("SLAVE: Ready");
}

void loop() {
  unsigned long now = millis();

  if (BT.available()) {
    char requestVal = BT.read();
    Serial.print("SLAVE");
    Serial.print(SLVADDR);
    Serial.println(": RECEIVED: ");
    Serial.println(requestVal);

    if (requestVal == '0') {
      setLock(UNLOCKED);
    } else if (requestVal == '1') {
      setLock(LOCKED);
    }
  }
  if (now - lastMagnetCheck >= 100) {
    lastMagnetCheck = now;
    int magnetState = digitalRead(MAGNET_PIN);
    doorClosed = (magnetState == LOW);
  }

  static int stableCount = 0;
  if (now - lastUSCheck >= 150) {
    lastUSCheck = now;

    long dist = readDistanceCM();
    bool detectedNow = (dist > 0 && dist < PERSON_THRESHOLD);

    if (detectedNow) {
      stableCount++;
    } else {
      stableCount = 0;
    }
    presenceDetected = (stableCount >= 2);

    Serial.print("SLAVE");
    Serial.print(SLVADDR);
    Serial.print(": Distance = ");
    Serial.print(dist);
    Serial.print(" cm, presenceDetected = ");
    Serial.println(presenceDetected ? "true" : "false");
  }

  if (presenceDetected && !isLocked) {}
  else if (presenceDetected && isLocked) {
    setLock(UNLOCKED);
  } else if (!presenceDetected && doorClosed && !isLocked) {
    setLock(LOCKED);
  }
  delay(10);
```

The current Phase 2 codebase is organized as two Arduino sketches, AutomaticDoorLock_Outside.ino and AutomaticDoorLock_Inside.ino. The outside sketch configures SPI and the RC522 reader, sets up the ultrasonic sensor pins, and opens a SoftwareSerial port to the HC-05 Bluetooth module. In the main loop it reads distance, updates a presenceDetected flag, checks for a new RFID card, and when appropriate, sends '0' or '1' over Bluetooth.

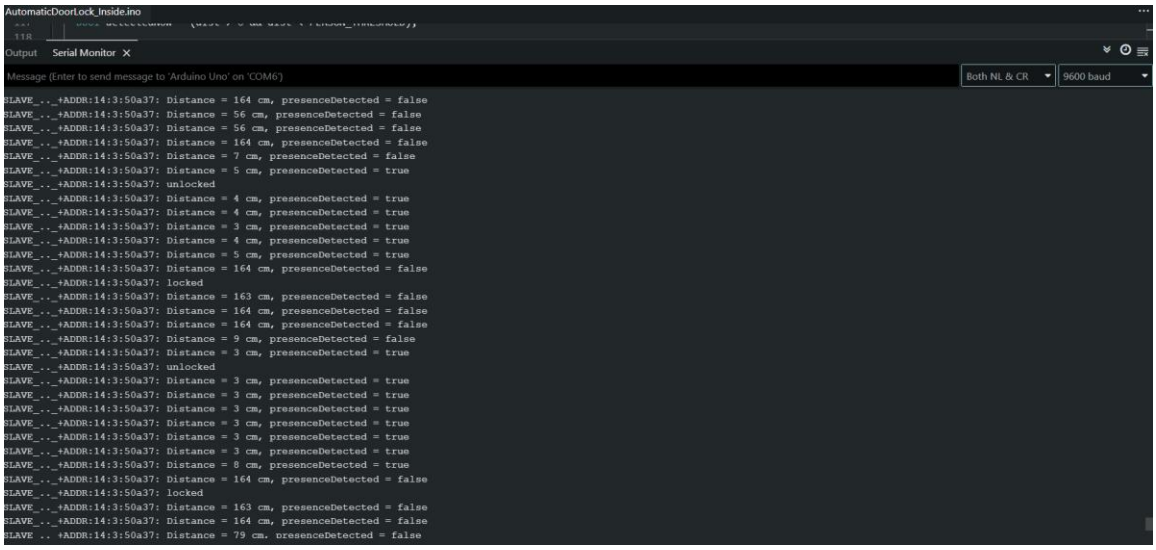
The inside sketch listens for Bluetooth characters while periodically sampling the ultrasonic sensor and the magnetic contact switch using millis()-based timing. The helper function setLock() wraps all servo updates and status LED changes so that the rest of the program only requests logical lock states.

4. Test Scripts and Current Verification

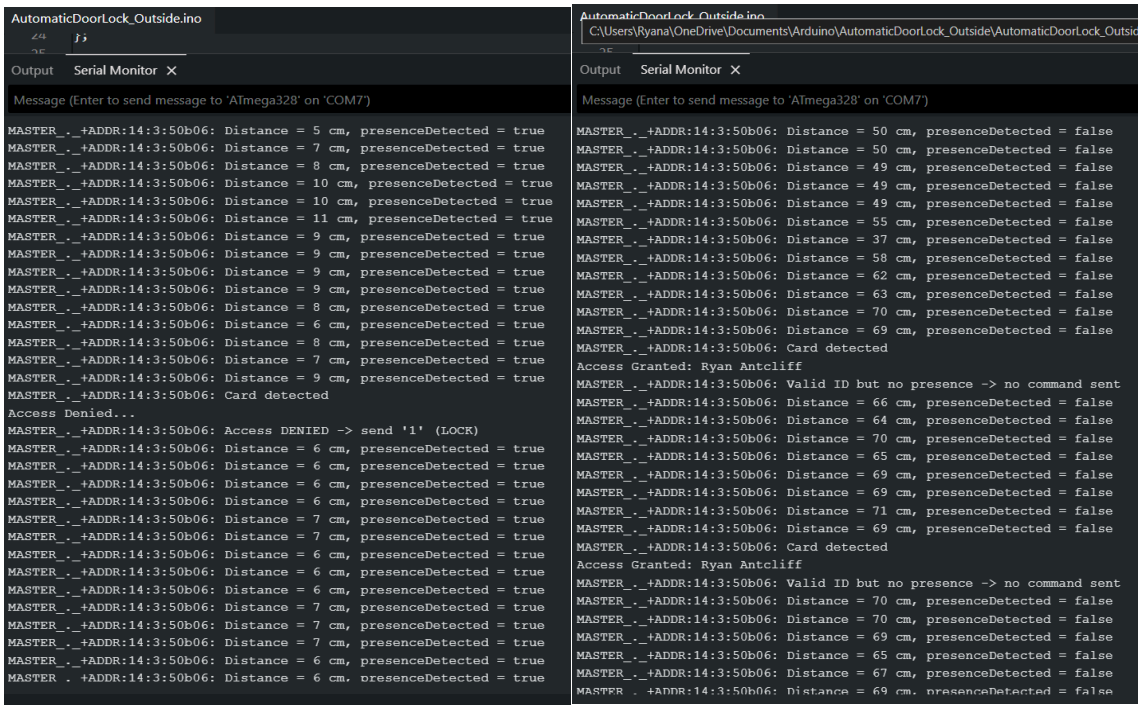
To verify each sensor and actuator before full integration, I wrote smaller demo sketches and captured serial output. For the ultrasonic sensors, I printed the measured distance and whether presence was detected. For the servo, I logged “locked” and “unlocked” messages whenever the

state changed. For Bluetooth, I printed every character received to make sure '0' and '1' were not corrupted.

Representative serial tests:



Inside unit serial logs show distance in centimeters, presenceDetected state, and resulting locked/unlocked states.



Outside unit logs show RFID UIDs, Access Granted / Access Denied messages, and which command ('0' or '1') was sent over Bluetooth. Additional logs confirm that when a valid ID is scanned but no one is in front of the ultrasonic sensor, the system prints that no command is sent.

4.1 Future Hardware and Software Testing Plan

Planned tests for the final system include measuring servo torque on the real deadbolt, mounting the magnetic sensor on an actual door, and tuning inside/outside distance thresholds, stress-testing the Bluetooth link.

5. Design Artifacts and Future Architecture

Although the current prototype is implemented as two Arduino sketches, I have already drafted the long-term architecture using diagrams and CRC cards. These will guide the refactor to a FreeRTOS-based design with clear modules for sensing, communication, safety checks, and administration.

Key design-level components include:

- InsideUnitController and OutsideUnitController for coordinating each side of the door.
- BluetoothComm for encapsulating the HC-05 link.
- ServoMotorLock for abstracting servo position and lock states.
- SafetyManager and ErrorHandler for enforcing safety constraints and handling failure cases.
- UltrasonicSensorInside/Outside and MagneticSensor for presence and door-state inputs.

6. Progress and Observations

Compared to Phase 1, this phase moves the project from paper design into a working physical prototype. Now I have a complete sensing and actuation chain where the outside unit can recognize my ID card, check that I am actually standing at the door, and request an unlock; the inside unit receives that request, verifies the door is closed, and turns the servo while logging everything to the serial monitor. Some key lessons from Phase 2 include the importance of consistent baud rates on the HC-05 modules, careful placement of ultrasonic sensors to avoid false triggers, and designing the state machine so that the door never locks on a person in the doorway. In Phase 3, I plan to migrate the sketches to a FreeRTOS-style structure. Overall, the Phase 2 prototype demonstrates that the automatic door lock can be built safely on top of the current hardware and software foundation.