MINI PROJECT REPORT: WASHING MACHINE

GROUP 4

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION: 12 JUNE 2025

NO	NAME	MATRIC NUMBER
1.	IRDINA NABIHAH BINTI MOHD NAZRI	2214772
2.	IZZAH ZAHIRA BINTI NORAZLEE	2217696
3.	NOR MAISARAH BINTI ISMAIL	2213080

TABLE OF CONTENTS

INTRODUCTION

This report is about the integration of electronics, mechanical systems, and programming has become essential for developing efficient and user-friendly household appliances. This mini project aims to design and implement a prototype of a smart washing machine using fundamental principles of mechatronics system integration. The system is developed using an Arduino Mega 2560 microcontroller, which serves as the central controller interfacing with multiple peripheral components including an RFID reader, Bluetooth module, LCD display, LEDs, a buzzer, and a stepper motor.

This project focuses on the simulation of a functional washing machine capable of user authentication, cycle selection, and credit-based operation. By incorporating RFID for secure user access and Bluetooth communication for input commands, the prototype emulates the key features of a modern laundry device. Furthermore, the system uses a state machine-based control algorithm to manage the sequential execution of washing stages wash, rinse, and spin while providing real-time feedback to the user through visual and audio indicators.

The objective of this project is to demonstrate a comprehensive understanding of component integration, control logic design, and embedded system development. This report outlines the hardware setup, software methodology, experimental results, and conclusions drawn from the implementation process.

ABSTRACT

This mini project presents the design and implementation of a prototype smart washing machine using mechatronic system integration. The system is built around the Arduino Mega 2560 microcontroller and features RFID-based user authentication, Bluetooth-based command input, and a state-driven control logic for executing washing cycles. Key components include an LCD for displaying system status, LEDs for indicating process stages, a stepper motor simulating drum motion, and a buzzer for auditory alerts.

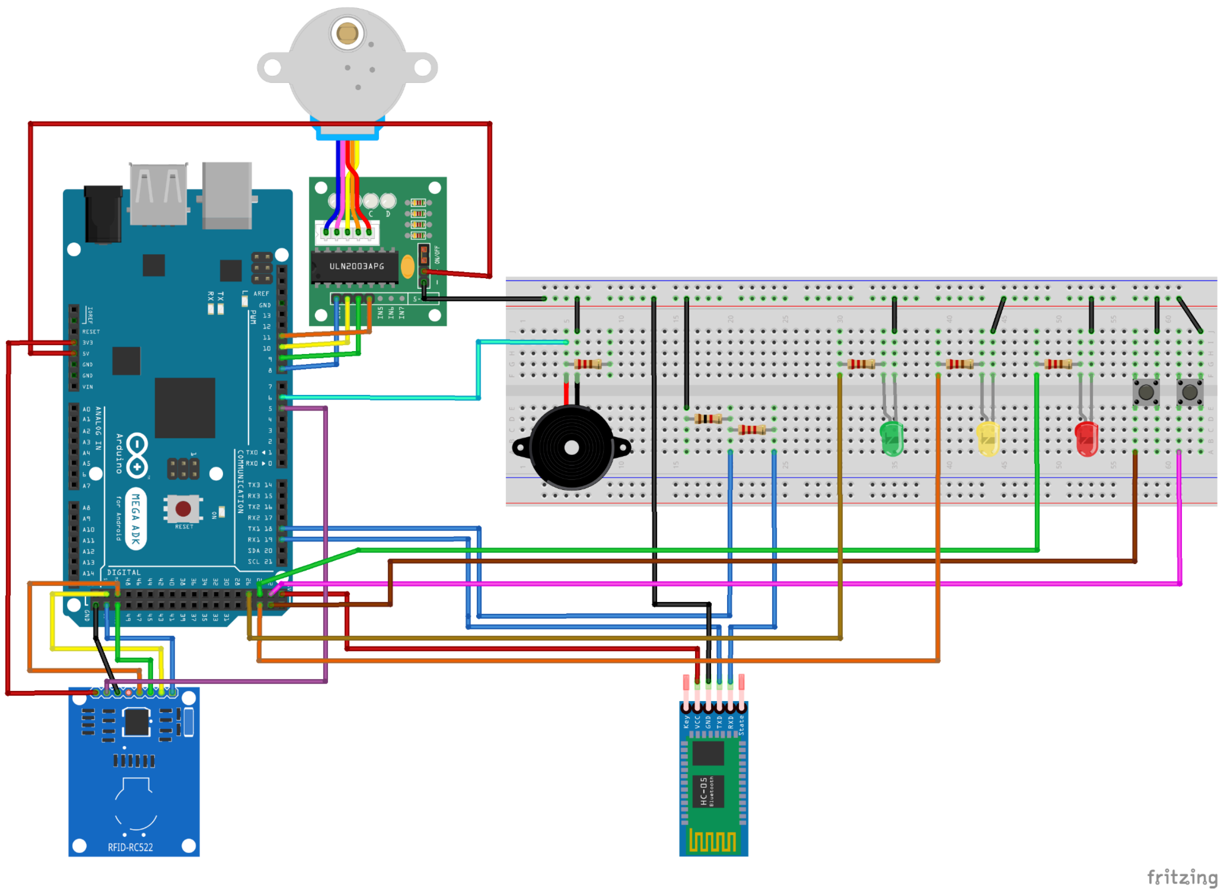
Three washing modes are available, each with different timing configurations for the wash, rinse, and spin stages. Users authenticate via RFID, select a washing cycle through a Bluetooth interface, and initiate the process via physical buttons. The system deducts credits according to the selected mode and provides real-time feedback throughout the cycle.

The project successfully demonstrates fundamental mechatronics principles, including sensor and actuator interfacing, real-time control, and embedded software design. The prototype also establishes a foundation for future development into a fully IoT-enabled smart appliance with remote monitoring and data logging capabilities.

MATERIALS AND EQUIPMENT

- Breadboard
- Jumper wires
- USB cables
- Arduino Mega 2560 microcontroller
- LCD display (16x2 I2C)
- 3x LED (red, yellow and green)
- RFID Module: (MFRC522)
- BT Module: HC-06
- Buzzer
- 2x Push button
- Resistors
- 28BYJ-48 Stepper Motor
- ULN2003 Driver Module

EXPERIMENTAL SETUP



METHODOLOGY

Through this experiment, it is known that there are three stages that create the system;

1. Component Setup

- Microcontroller: Arduino Mega was used as the main controller.
- RFID Module: MFRC522 was interfaced via SPI for user authentication.
- Bluetooth Module: Connected via Serial1 to receive user input (mode selection, reload).
- LCD (20x4): Connected via I2C to display messages and balances.
- Stepper Motor: Simulated drum rotation during washing, rinsing, and spinning.
- LEDs: Indicated each washing stage (Wash, Rinse, Spin).
- Buzzer: Played tones at the beginning and end of the cycle.
- Buttons: Used for Start and Force Stop.

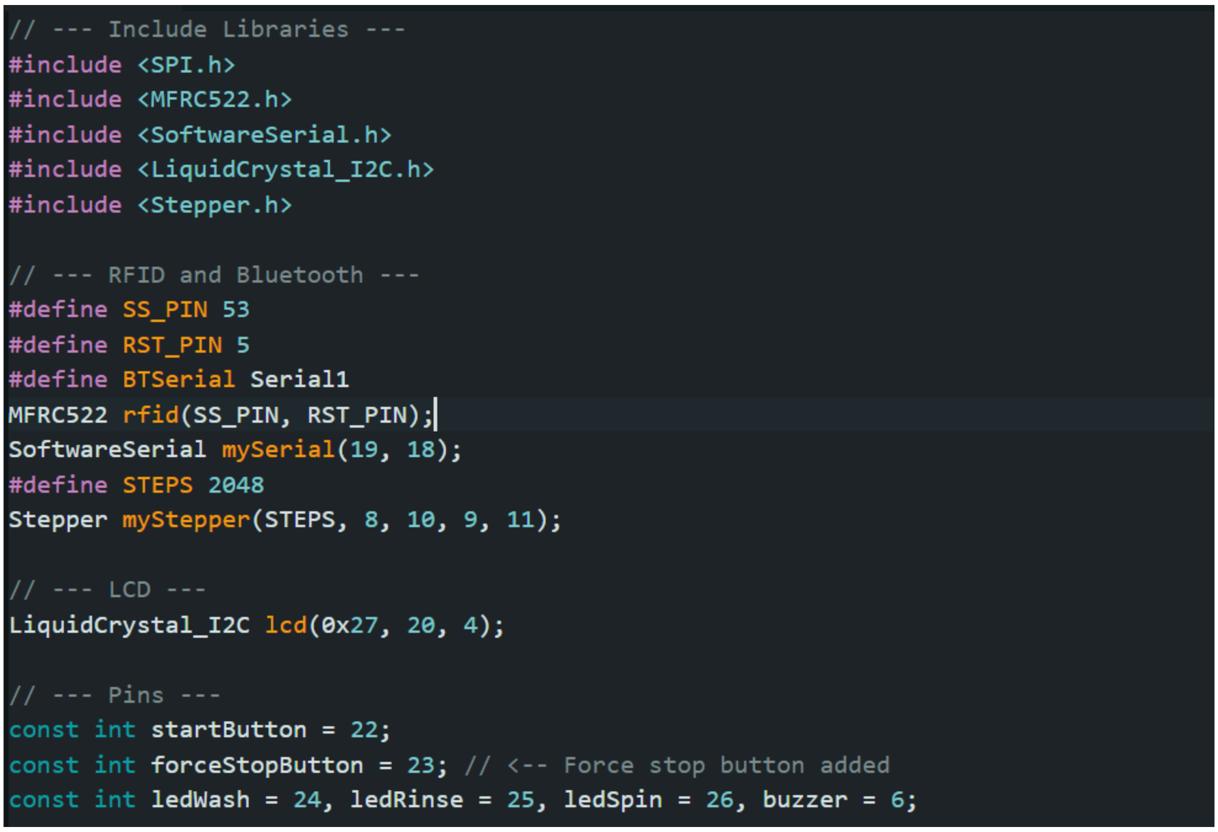
2. Software Design

- A state machine handled the washing cycle with states: WAIT_START → WAIT_CARD → WAIT_COMMAND → RUNNING_CYCLE → COMPLETE
- RFID authentication matched user UID to preloaded IDs and balances.
- Bluetooth input determined the type of cycle (A, B, C) or reloaded credits.
- Credit deduction was implemented based on selected mode.
- The washing cycle ran three steps:
    a. Wash: Stepper motor + LED
    b. Rinse: Stepper motor + LED
    c. Spin: Stepper motor + LED
    d. A force stop button interrupted the process safely at any time.

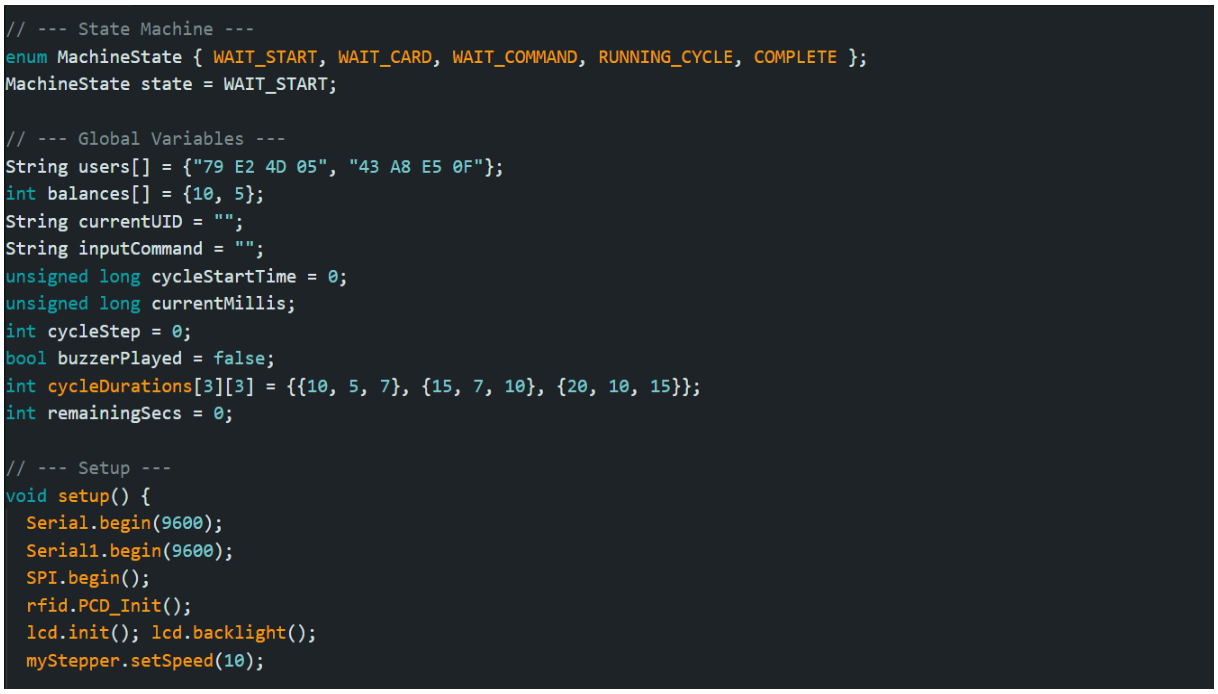
3. Cycle Timing

- Each cycle (A, B, C) had predefined durations for wash, rinse, and spin stages, stored in a 2D array.
- Time tracking was implemented using millis() for non-blocking timing.

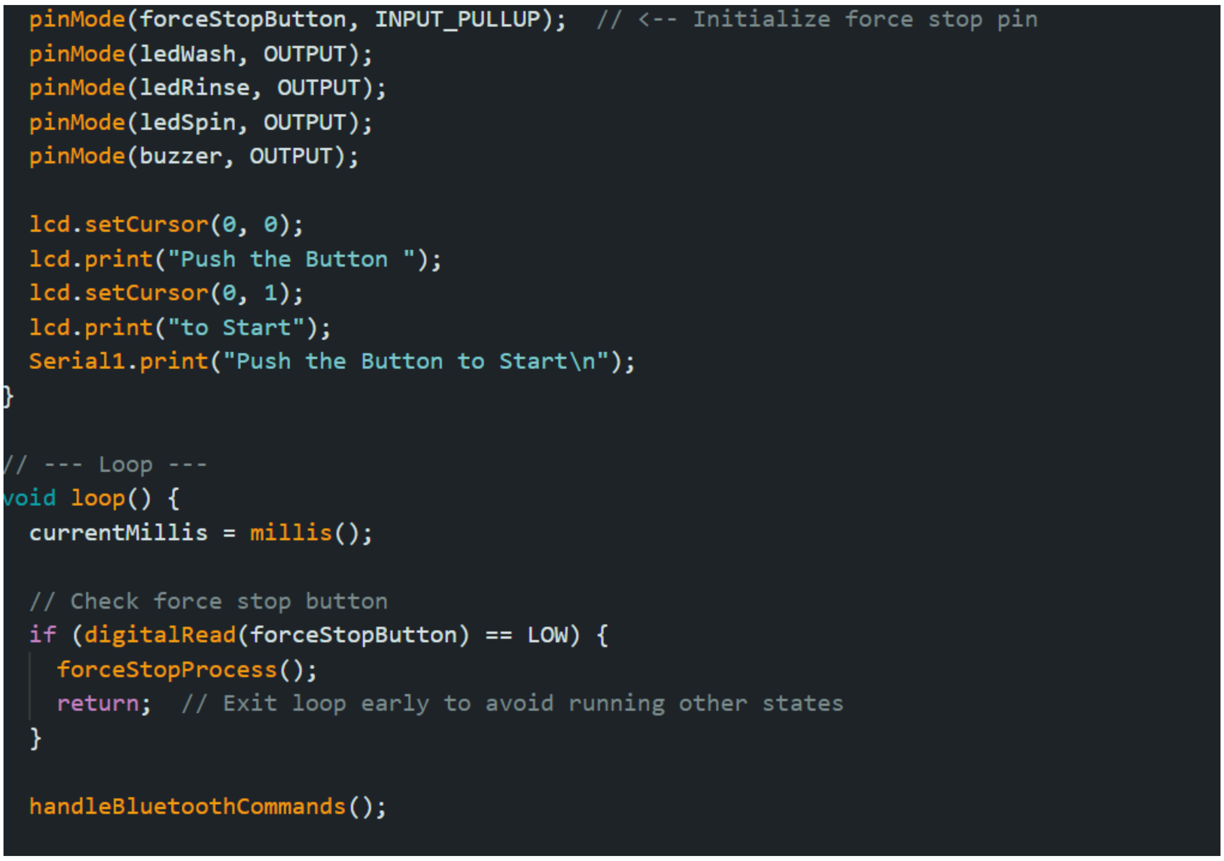
RESULTS



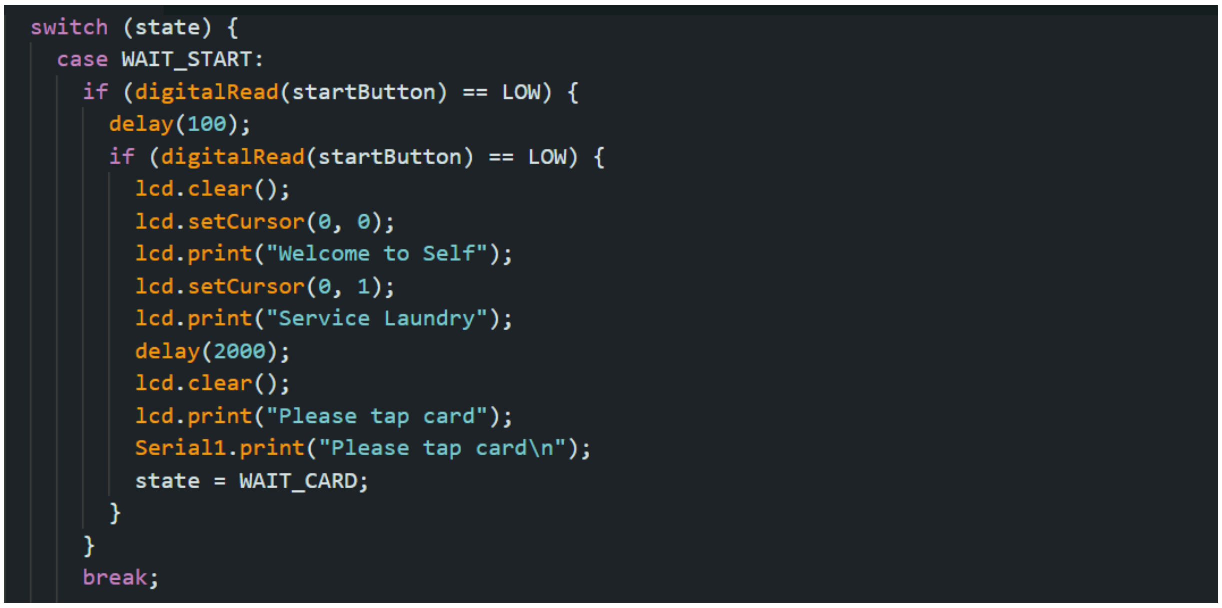
```
// --- Include Libraries ---
#include <SPI.h>
#include <MFRC522.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <Stepper.h>

// --- RFID and Bluetooth ---
#define SS_PIN 53
#define RST_PIN 5
#define BTSerial Serial1
MFRC522 rfid(SS_PIN, RST_PIN);
SoftwareSerial mySerial(19, 18);
#define STEPS 2048
Stepper myStepper(STEPS, 8, 10, 9, 11);

// --- LCD ---
LiquidCrystal_I2C lcd(0x27, 20, 4);

// --- Pins ---
const int startButton = 22;
const int forceStopButton = 23; // <-- Force stop button added
const int ledWash = 24, ledRinse = 25, ledSpin = 26, buzzer = 6;
```



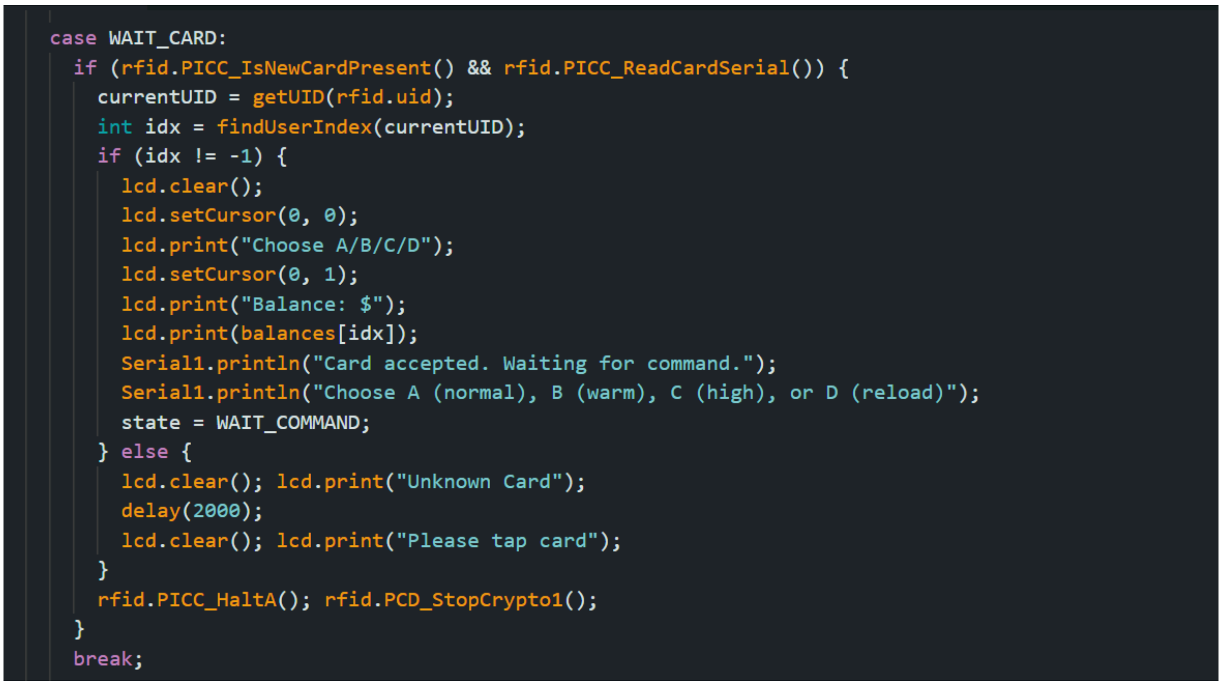
```
// --- State Machine ---
enum MachineState { WAIT_START, WAIT_CARD, WAIT_COMMAND, RUNNING_CYCLE, COMPLETE };
MachineState state = WAIT_START;

// --- Global Variables ---
String users[] = {"79 E2 4D 05", "43 A8 E5 0F"};
int balances[] = {10, 5};
String currentUID = "";
String inputCommand = "";
unsigned long cycleStartTime = 0;
unsigned long currentMillis;
int cycleStep = 0;
bool buzzerPlayed = false;
int cycleDurations[3][3] = {{10, 5, 7}, {15, 7, 10}, {20, 10, 15}};
int remainingSecs = 0;

// --- Setup ---
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  SPI.begin();
  rfid.PCD_Init();
  lcd.init(); lcd.backlight();
  myStepper.setSpeed(10);
```



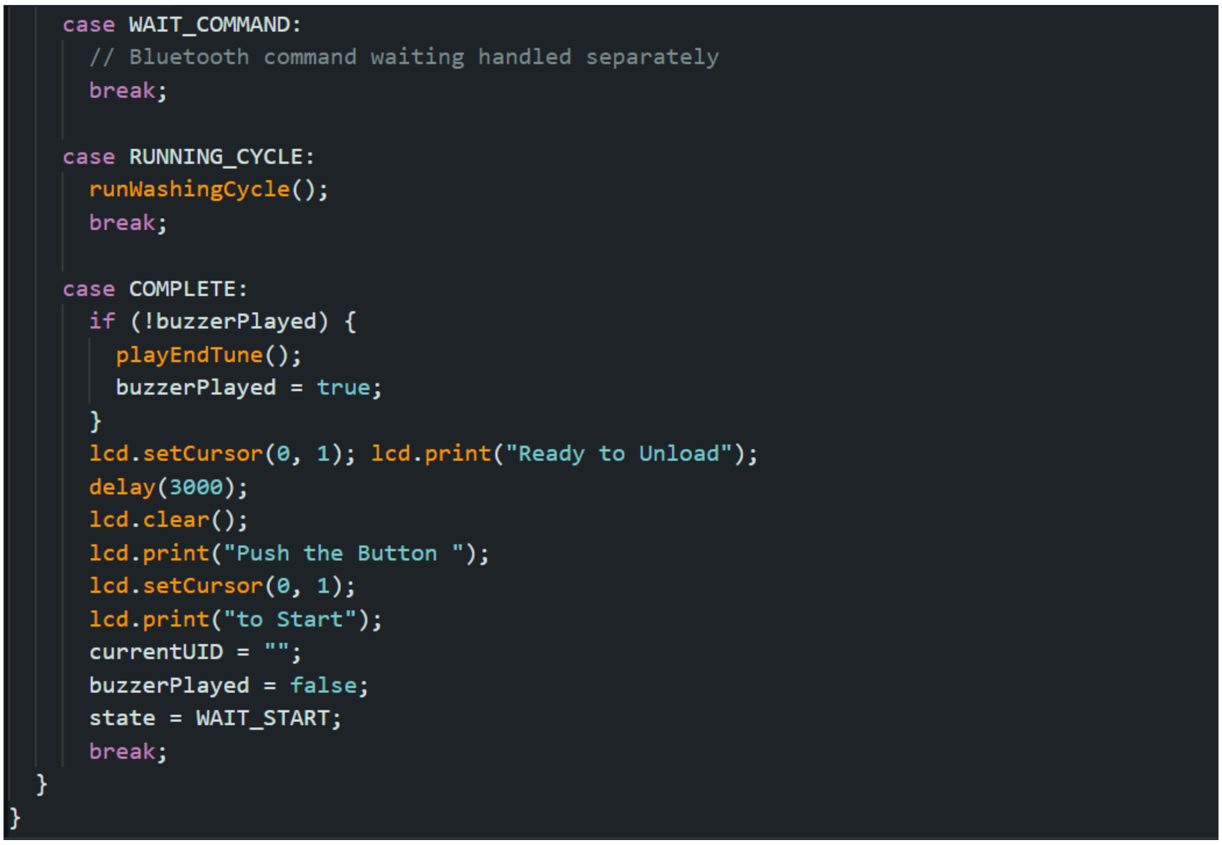
```
  pinMode(forceStopButton, INPUT_PULLUP);  // <-- Initialize force stop pin
  pinMode(ledWash, OUTPUT);
  pinMode(ledRinse, OUTPUT);
  pinMode(ledSpin, OUTPUT);
  pinMode(buzzer, OUTPUT);

  lcd.setCursor(0, 0);
  lcd.print("Push the Button ");
  lcd.setCursor(0, 1);
  lcd.print("to Start");
  Serial1.print("Push the Button to Start\n");
}

// --- Loop ---
void loop() {
  currentMillis = millis();

  // Check force stop button
  if (digitalRead(forceStopButton) == LOW) {
    forceStopProcess();
    return;  // Exit loop early to avoid running other states
  }

  handleBluetoothCommands();
```



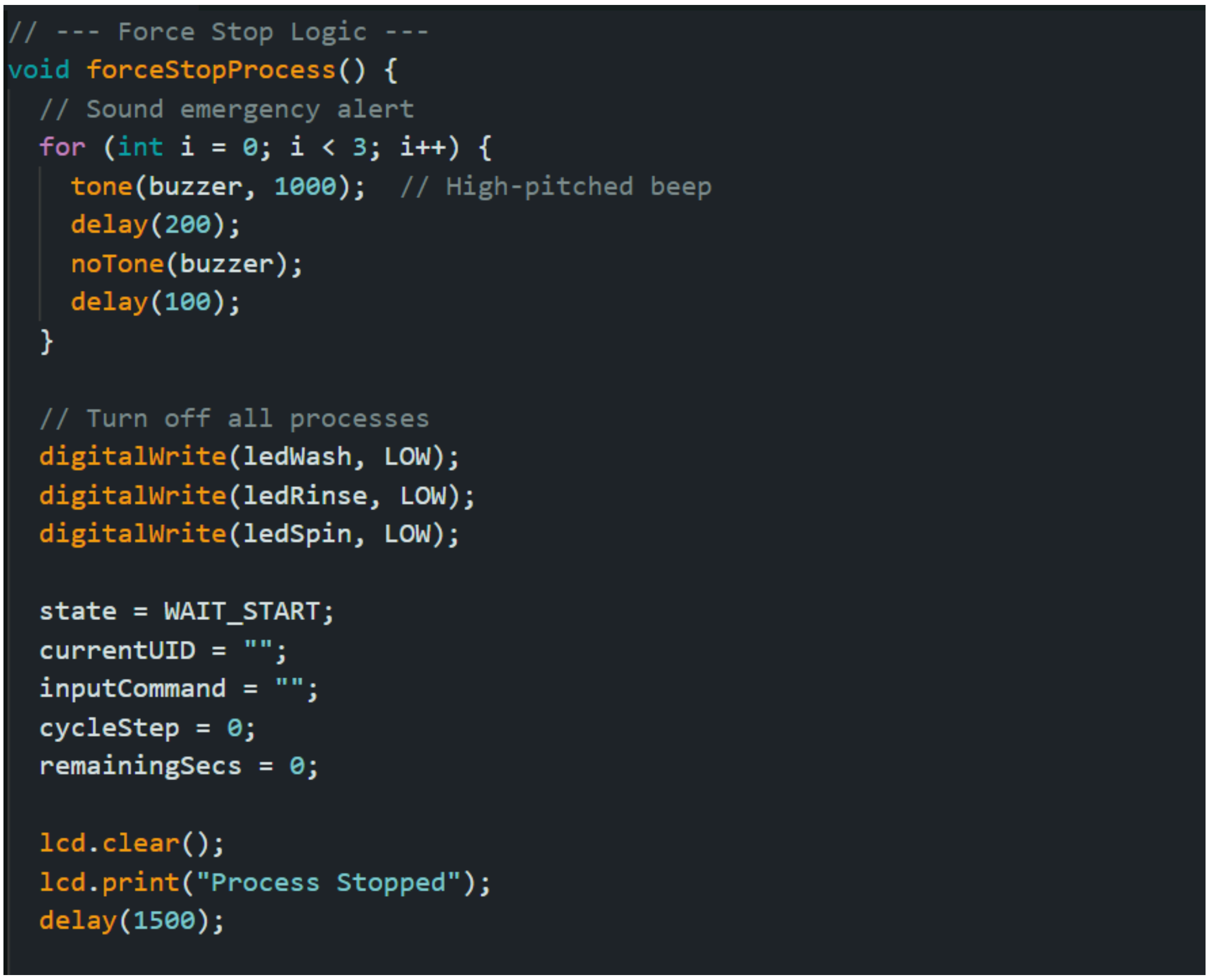
```
  switch (state) {
    case WAIT_START:
      if (digitalRead(startButton) == LOW) {
        delay(100);
        if (digitalRead(startButton) == LOW) {
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Welcome to Self");
          lcd.setCursor(0, 1);
          lcd.print("Service Laundry");
          delay(2000);
          lcd.clear();
          lcd.print("Please tap card");
          Serial1.print("Please tap card\n");
          state = WAIT_CARD;
        }
      }
      break;
```



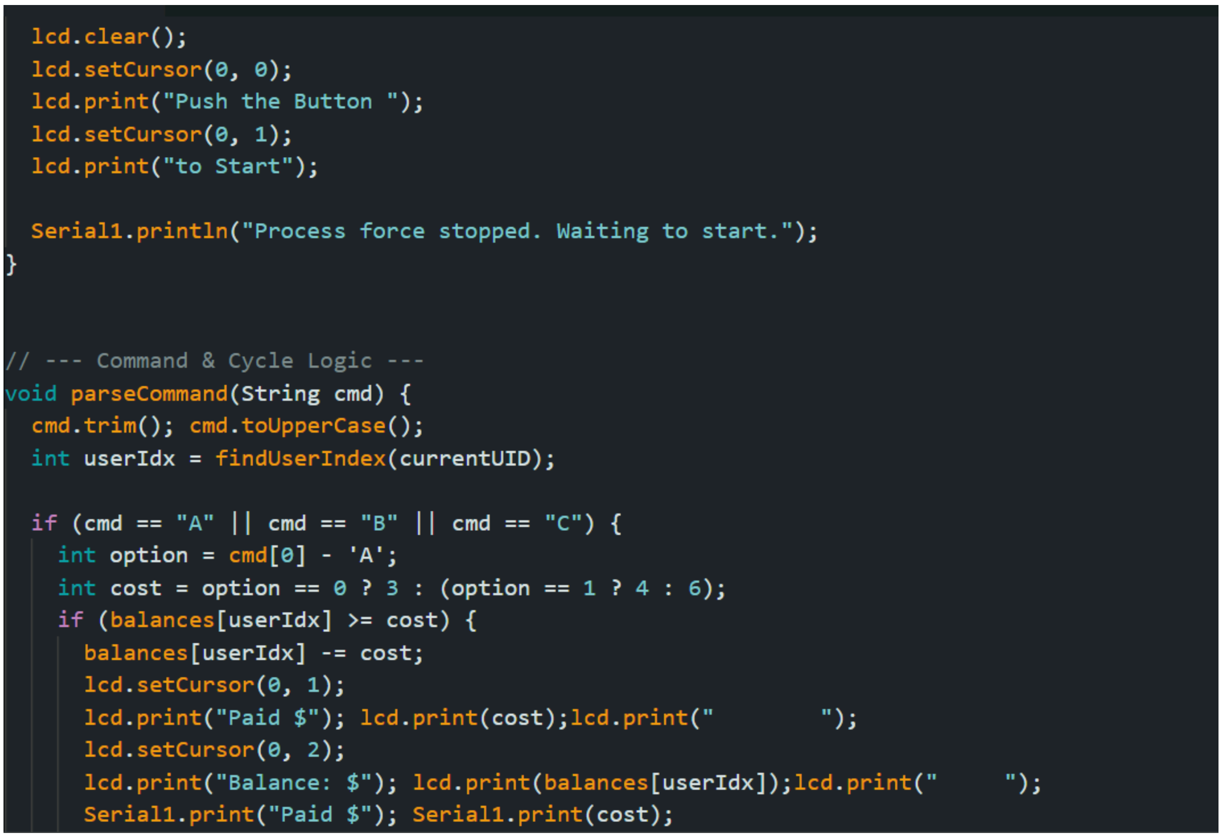
```
    case WAIT_CARD:
      if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
        currentUID = getUID(rfid.uid);
        int idx = findUserIndex(currentUID);
        if (idx != -1) {
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Choose A/B/C/D");
          lcd.setCursor(0, 1);
          lcd.print("Balance: $");
          lcd.print(balances[idx]);
          Serial1.println("Card accepted. Waiting for command.");
          Serial1.println("Choose A (normal), B (warm), C (high), or D (reload)");
          state = WAIT_COMMAND;
        } else {
          lcd.clear(); lcd.print("Unknown Card");
          delay(2000);
          lcd.clear(); lcd.print("Please tap card");
        }
        rfid.PICC_HaltA(); rfid.PCD_StopCrypto1();
      }
      break;
```



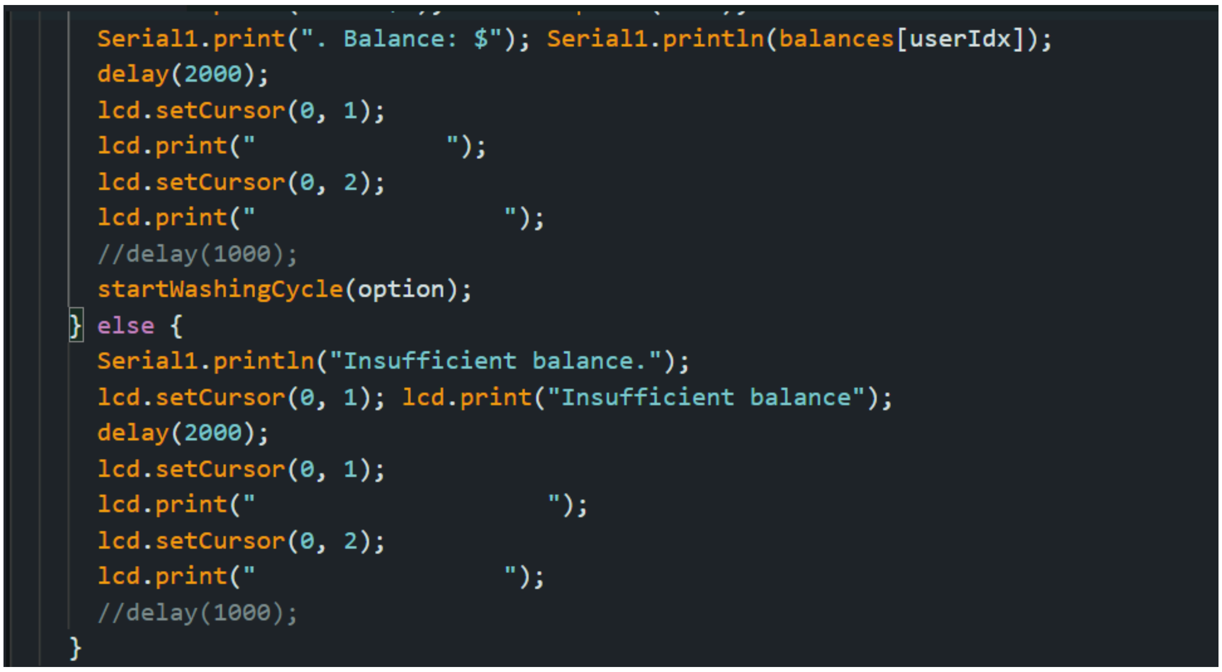
```
    case WAIT_COMMAND:
      // Bluetooth command waiting handled separately
      break;

    case RUNNING_CYCLE:
      runWashingCycle();
      break;

    case COMPLETE:
      if (!buzzerPlayed) {
        playEndTune();
        buzzerPlayed = true;
      }
      lcd.setCursor(0, 1); lcd.print("Ready to Unload");
      delay(3000);
      lcd.clear();
      lcd.print("Push the Button ");
      lcd.setCursor(0, 1);
      lcd.print("to Start");
      currentUID = "";
      buzzerPlayed = false;
      state = WAIT_START;
      break;
  }
}
```



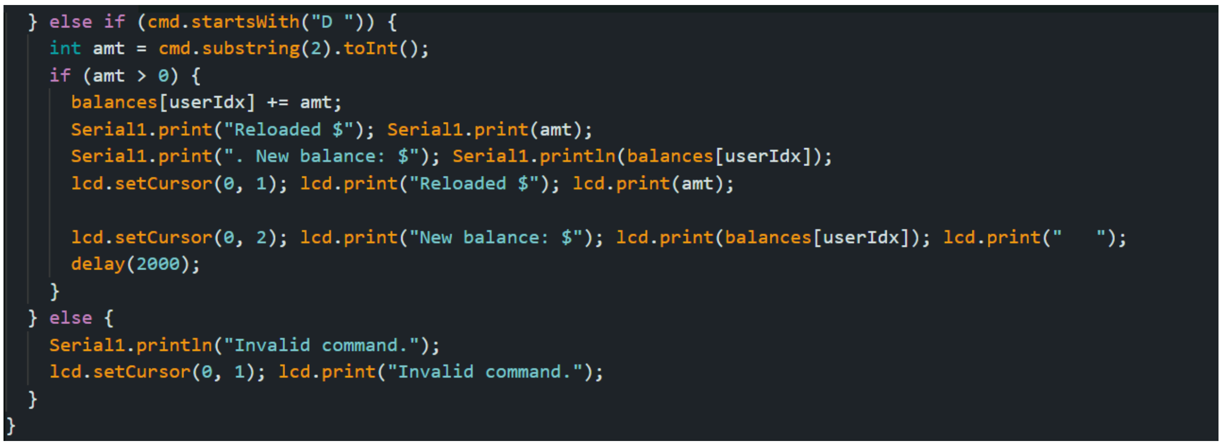
```
// --- Force Stop Logic ---
void forceStopProcess() {
  // Sound emergency alert
  for (int i = 0; i < 3; i++) {
    tone(buzzer, 1000);  // High-pitched beep
    delay(200);
    noTone(buzzer);
    delay(100);
  }

  // Turn off all processes
  digitalWrite(ledWash, LOW);
  digitalWrite(ledRinse, LOW);
  digitalWrite(ledSpin, LOW);

  state = WAIT_START;
  currentUID = "";
  inputCommand = "";
  cycleStep = 0;
  remainingSecs = 0;

  lcd.clear();
  lcd.print("Process Stopped");
  delay(1500);
```



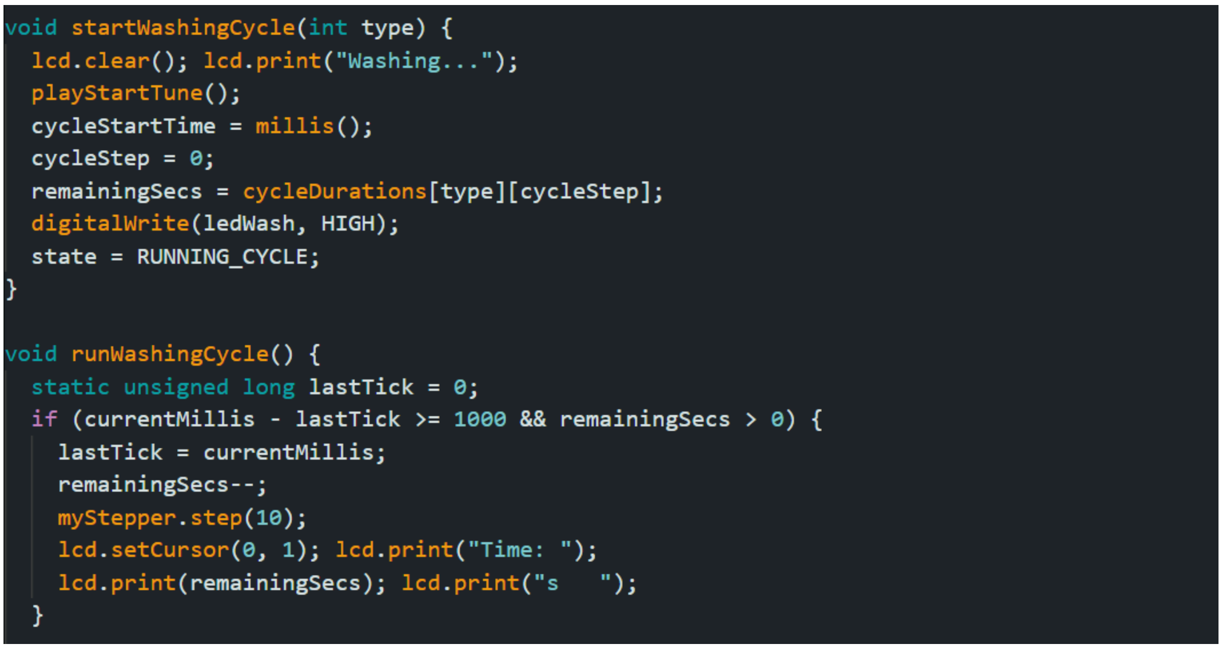
```
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Push the Button ");
  lcd.setCursor(0, 1);
  lcd.print("to Start");

  Serial1.println("Process force stopped. Waiting to start.");
}


// --- Command & Cycle Logic ---
void parseCommand(String cmd) {
  cmd.trim(); cmd.toUpperCase();
  int userIdx = findUserIndex(currentUID);

  if (cmd == "A" || cmd == "B" || cmd == "C") {
    int option = cmd[0] - 'A';
    int cost = option == 0 ? 3 : (option == 1 ? 4 : 6);
    if (balances[userIdx] >= cost) {
      balances[userIdx] -= cost;
      lcd.setCursor(0, 1);
      lcd.print("Paid $"); lcd.print(cost);lcd.print("       ");
      lcd.setCursor(0, 2);
      lcd.print("Balance: $"); lcd.print(balances[userIdx]);lcd.print("    ");
      Serial1.print("Paid $"); Serial1.print(cost);
```



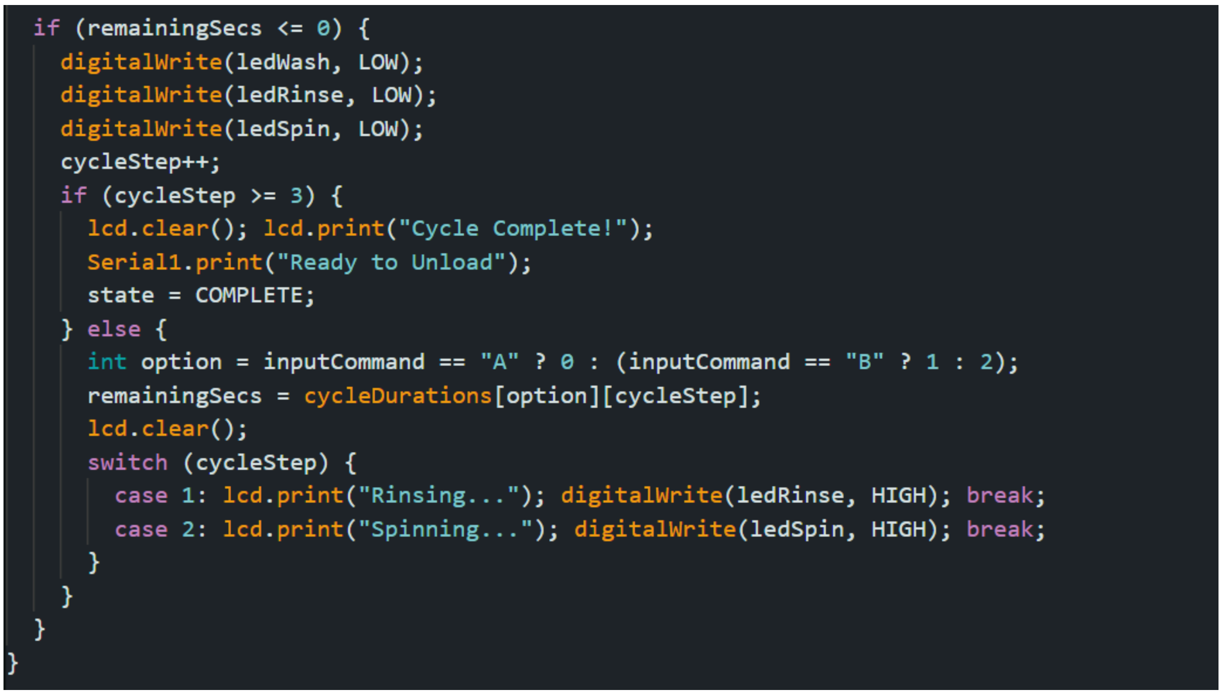
```
      Serial1.print(". Balance: $"); Serial1.println(balances[userIdx]);
      delay(2000);
      lcd.setCursor(0, 1);
      lcd.print("             ");
      lcd.setCursor(0, 2);
      lcd.print("                 ");
      //delay(1000);
      startWashingCycle(option);
    } else {
      Serial1.println("Insufficient balance.");
      lcd.setCursor(0, 1); lcd.print("Insufficient balance");
      delay(2000);
      lcd.setCursor(0, 1);
      lcd.print("                    ");
      lcd.setCursor(0, 2);
      lcd.print("                 ");
      //delay(1000);
    }
```



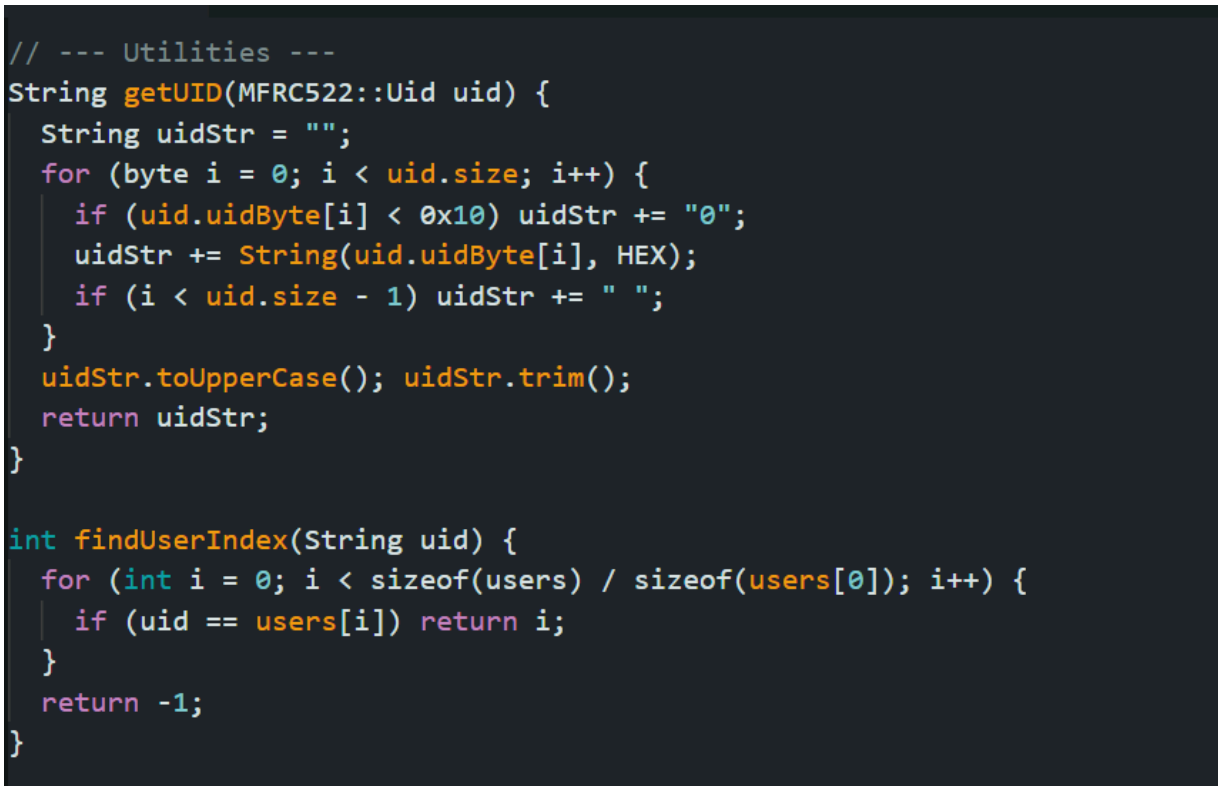
```
  } else if (cmd.startsWith("D ")) {
    int amt = cmd.substring(2).toInt();
    if (amt > 0) {
      balances[userIdx] += amt;
      Serial1.print("Reloaded $"); Serial1.print(amt);
      Serial1.print(". New balance: $"); Serial1.println(balances[userIdx]);
      lcd.setCursor(0, 1); lcd.print("Reloaded $"); lcd.print(amt);

      lcd.setCursor(0, 2); lcd.print("New balance: $"); lcd.print(balances[userIdx]); lcd.print("   ");
      delay(2000);
    }
  } else {
    Serial1.println("Invalid command.");
    lcd.setCursor(0, 1); lcd.print("Invalid command.");
  }
}
```



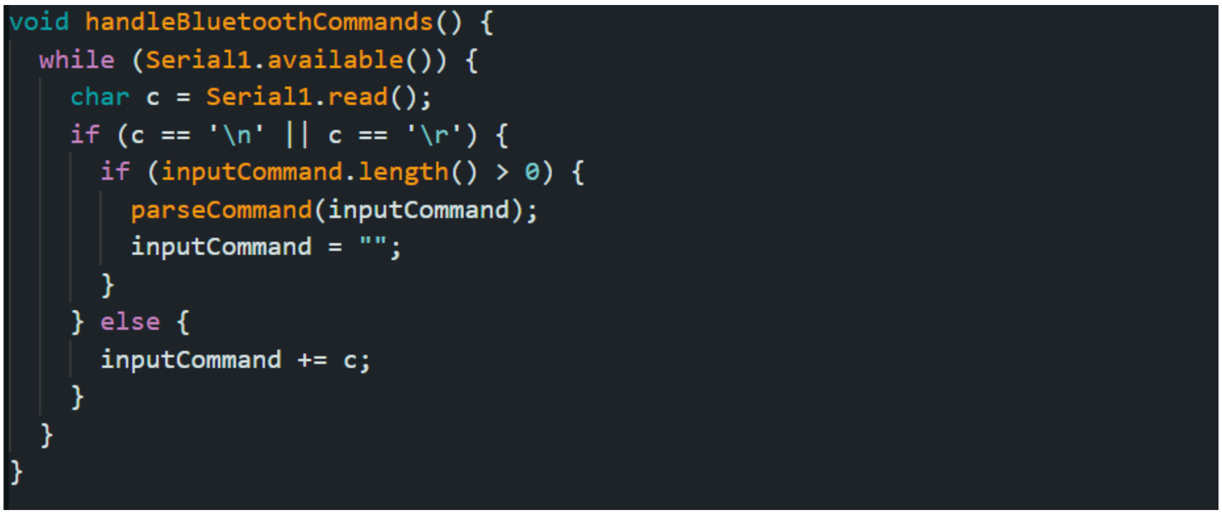
```
void startWashingCycle(int type) {
  lcd.clear(); lcd.print("Washing...");
  playStartTune();
  cycleStartTime = millis();
  cycleStep = 0;
  remainingSecs = cycleDurations[type][cycleStep];
  digitalWrite(ledWash, HIGH);
  state = RUNNING_CYCLE;
}

void runWashingCycle() {
  static unsigned long lastTick = 0;
  if (currentMillis - lastTick >= 1000 && remainingSecs > 0) {
    lastTick = currentMillis;
    remainingSecs--;
    myStepper.step(10);
    lcd.setCursor(0, 1); lcd.print("Time: ");
    lcd.print(remainingSecs); lcd.print("s   ");
  }
```



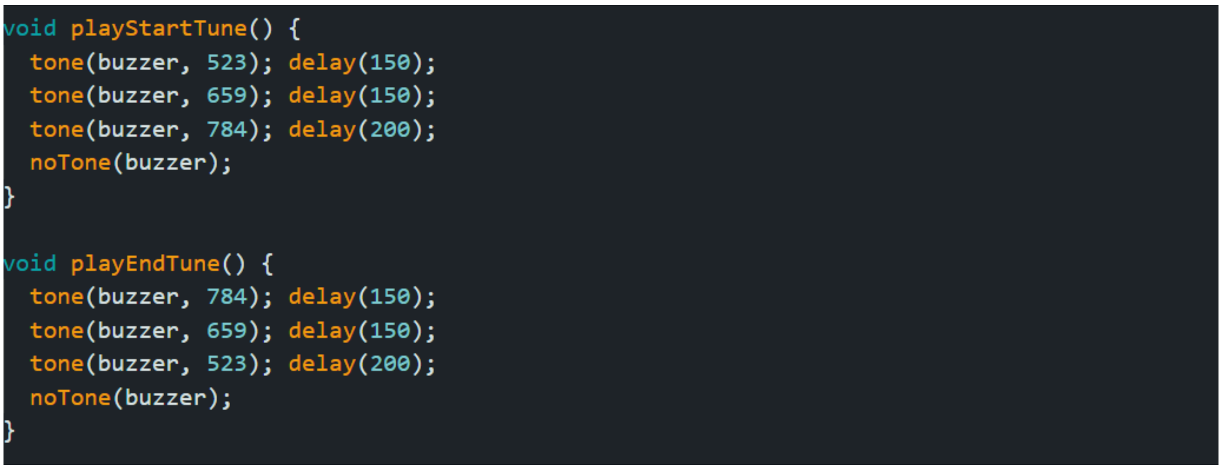
```
  if (remainingSecs <= 0) {
    digitalWrite(ledWash, LOW);
    digitalWrite(ledRinse, LOW);
    digitalWrite(ledSpin, LOW);
    cycleStep++;
    if (cycleStep >= 3) {
      lcd.clear(); lcd.print("Cycle Complete!");
      Serial1.print("Ready to Unload");
      state = COMPLETE;
    } else {
      int option = inputCommand == "A" ? 0 : (inputCommand == "B" ? 1 : 2);
      remainingSecs = cycleDurations[option][cycleStep];
      lcd.clear();
      switch (cycleStep) {
        case 1: lcd.print("Rinsing..."); digitalWrite(ledRinse, HIGH); break;
        case 2: lcd.print("Spinning..."); digitalWrite(ledSpin, HIGH); break;
      }
    }
  }
}
```



```
// --- Utilities ---
String getUID(MFRC522::Uid uid) {
  String uidStr = "";
  for (byte i = 0; i < uid.size; i++) {
    if (uid.uidByte[i] < 0x10) uidStr += "0";
    uidStr += String(uid.uidByte[i], HEX);
    if (i < uid.size - 1) uidStr += " ";
  }
  uidStr.toUpperCase(); uidStr.trim();
  return uidStr;
}

int findUserIndex(String uid) {
  for (int i = 0; i < sizeof(users) / sizeof(users[0]); i++) {
    if (uid == users[i]) return i;
  }
  return -1;
}
```



```
void handleBluetoothCommands() {
  while (Serial1.available()) {
    char c = Serial1.read();
    if (c == '\n' || c == '\r') {
      if (inputCommand.length() > 0) {
        parseCommand(inputCommand);
        inputCommand = "";
      }
    } else {
      inputCommand += c;
    }
  }
}
```



```
void playStartTune() {
  tone(buzzer, 523); delay(150);
  tone(buzzer, 659); delay(150);
  tone(buzzer, 784); delay(200);
  noTone(buzzer);
}

void playEndTune() {
  tone(buzzer, 784); delay(150);
  tone(buzzer, 659); delay(150);
  tone(buzzer, 523); delay(200);
  noTone(buzzer);
}
```

CONCLUSION

In this lab, we successfully demonstrated the practical integration of software and hardware in an embedded system through the development of a basic washing machine prototype.Firstly, the Arduino-based system provided user authentication via RFID, command control via Bluetooth, and user feedback through an LCD, LEDs, and a buzzer.

Moreover, a structured state machine allowed efficient process management and real-time control of the washing cycle. Hence, this project highlighted the keys of mechatronics principles such as interfacing, control logic, and component integration.

More importantly, the prototype forms a strong foundation for future enhancement into a full IoT-enabled washing device or system by using cloud sensitivity, remote monitoring and also data analytics.

ACKNOWLEDGEMENTS

A special thanks goes out to Dr. Wahju Sediono and Dr. Zulkifli Bin Zainal Abidin, my teaching assistant, and my peers for their invaluable help and support in finishing this report. Their advice, feedback, and experience have greatly influenced the level of quality and understanding of this work. Their time, patience, and commitment to supporting my academic success are greatly appreciated.

STUDENT'S DECLARATION

Certificate of Originality and Authenticity

This is to certify that we are **responsible** for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.

We hereby certify that this report has **not been done by only one individual** and **all of us have contributed to the report**. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have **read** and **understand** the content of the total report and no further improvement on the reports is needed from any of the individual's contributors to the report.

We therefore, agreed unanimously that this report shall be submitted for **marking** and this **final printed report** has been **verified by us**.

Signature: [illegible] Read [✓]
Name: IRDINA NABIHAH BINTI MOHD NAZRI Understand [✓]
Matric Number: 2214772 Agree [✓]

Signature: [illegible] Read [✓]
Name: IZZAH ZAHIRA BINTI NORAZLEE Understand [✓]
Matric Number: 2217696 Agree [✓]

Signature: [illegible] Read [✓]
Name: NOR MAISARAH BINTI ISMAIL Understand [✓]
Matric Number: 2213080 Agree [✓]