LABORATORY REPORT

Mini Project: Washing Machine MCTA3202

GROUP B

PROGRAMME: MECHATRONIC ENGINEERING

GROUP MEMBERS:	MATRIC NO:
1. MUHAMMAD AFIQ BIN AHMAD	2115203
2. HANIS BINTI MOHD IZANI	2020590
3. SARAH AISHAH BINTI SA’AID HAZLEY	2117600
4. NUR AYU NAZIERA BINTI ROSLI	2119202
5. NOORUL GHOUSIAH BINTI NOORDEEN SAHIB	2118298

DATE OF SUBMISSION:
Wednesday, 22th January 2023

ABSTRACT

This lab project introduces an automated washing machine system leveraging RFID technology for user authentication. Integrating LCD displays, servo motors, buzzers, and Arduino controllers, the system enhances user experience through personalized RFID card interactions. The Arduino microcontroller orchestrates seamless communication between components, offering a secure, convenient, and intelligent laundry solution. This project showcases the practical application of embedded systems and smart technologies in household appliances, contributing to the evolution of user-centric home automation.

TABLE OF CONTENTS

Table of Contents

INTRODUCTION

Washing machines are machines used to wash, rinse, and spin clothes. For our project, we have designed and integrated a machine that does exactly that, and more. What differs our washing machine from the washing machines available in the market today?

In this report, we will dive into details of the user interface, mechanical process, sensor usage and safety integrated into our washing machine.

For starters, washing machines make your life easier, and we have created a washing machine to make your life even easier. We aim to enhance overall performance, energy efficiency and user experience for the washing machine.

By the end of this report, readers will be able to gain a comprehensive understanding on system integration within a washing machine. In the future, we hope to serve further advancements in home appliances, making them more intelligent for the betterment of humanity.

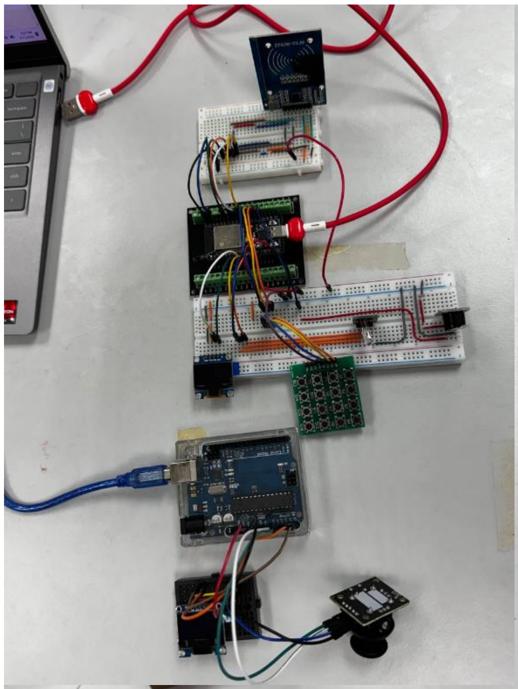
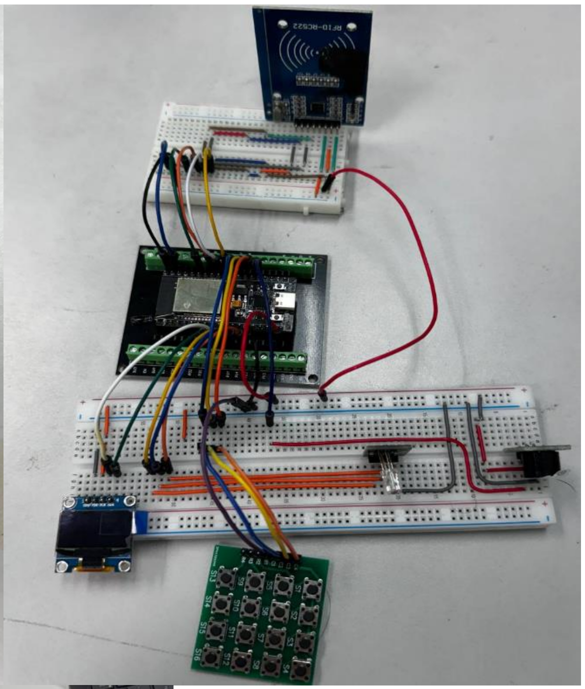
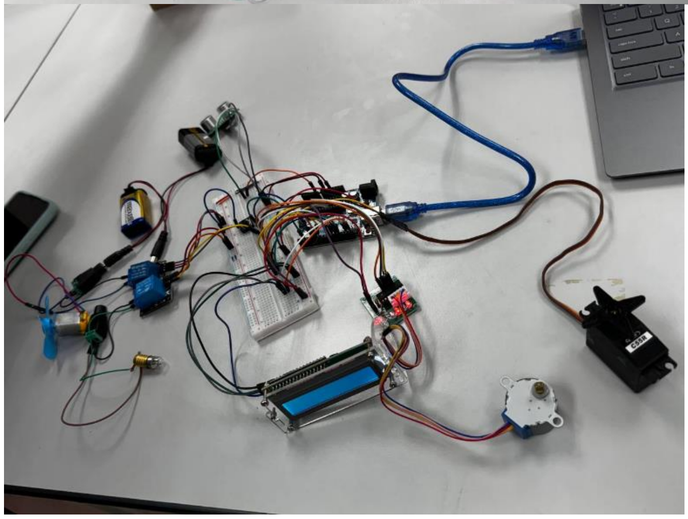
PROCEDURE

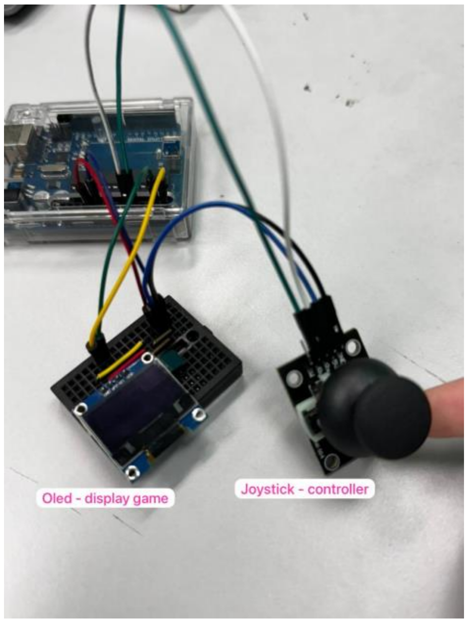
Materials And Equipment

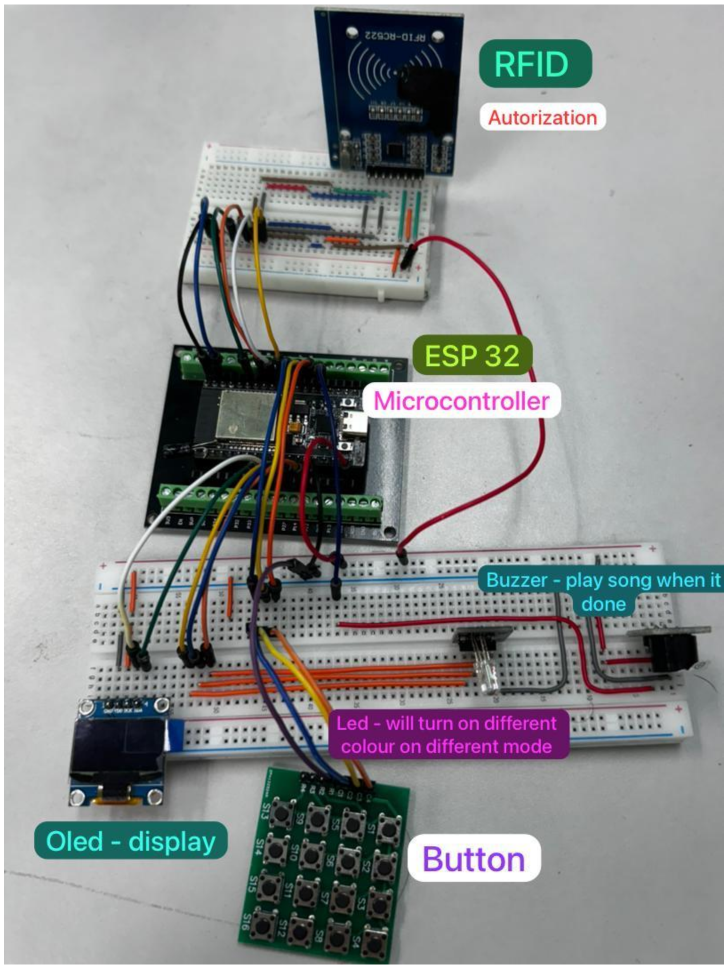
- Breadboard
- Jumper wires
- USB cables
- 2x Arduino Uno microcontroller
- Esp 32 with expansion board
- 2x OLED Display
- 2x LCD display (16*2 I2C)
- Joystick
- LED
- 1x IR Sensor
- RFID Module
- Piezo Buzzer
- Multiple push button
- Servo motor
- 2x 9V Battery
- 9V DC Motor
- 2x Relay Module

- Small lightbulb
- Stepper motor (28BYJ-48)
- ULN2003 driver board
- Ultrasonic sensor US-016

Experimental Setup




RFID
Autorization
ESP 32
Microcontroller
Buzzer - play song when it done
Led - will turn on different colour on different mode
Oled - display
Button

<u>Methodology</u>

1. Hardware Setup

a. setup of technical washing part:

- Prepare an Arduino board.
- Connect the ultrasonic sensor by linking VCC to 5V, OUT to A0, GND to GND, and RANGE to GND.
- For LCD (16x2 I2C), GND to GND, VCC to 5V, SDA to SDA, and SCL to SCL.
- Attach the servomotor’s GND to GND, Vcc to 5V, and DATA to D7.
- Connect Stepper Motor (28BYJ-48) with ULN2003 Driver Board.
- Connect ULN2003 Driver Board to Arduino board by wiring IN1 to D10, IN2 to D11, IN3 to D12, IN4 to D13, Positive to 5V, and Negative to GND.
- For the lightbulb relay, connect GND to GND, VCC to VCC, and IN to D5. Link the fan relay's GND to GND, VCC to VCC, and IN to D6.
- Simulate door closing with a pushbutton by connecting one leg to D2 and the other to GND.

b. setup of user interface part:

- Prepare a separate Arduino board
- Connect two OLED displays to the Arduino using jumper wires, ensuring proper I2C communication. Assign unique addresses if needed.
- Connect the RFID module to the Arduino for user authentication.
- Wire the analog controller to the appropriate pins on the Arduino for game control.
- Connect multiple push buttons for mode selection and other interactive functions.
- Wire the IR sensor to detect the start of the washing process.
- Connect LEDs for lighting, and assign colors based on washing modes.
- Connect to the buzzer to provide audible feedback.

2. Software Implementation

a. Arduino Programming:

- Write code to read RFID cards and grant or deny access based on user permissions.
- Implement the game logic for unauthorized users, taking input from the analog controller.
- Code the logic for displaying washing machine modes on the OLED displays and handling user input through push buttons.
- Develop code to activate the IR sensor, control LED lighting, and change LED colors based on washing modes.
- Write code to trigger the buzzer at the end of each washing cycle.
- Write code for the technical part of the washing machine, that is, controlling stepper motor, servo motor, lcd display, and relay.

b. User Input Handling:

- Push Buttons: Code functions to interpret user input from push buttons for mode selection.
- Analog Controller: Develop code to interpret analog controller input for game control.

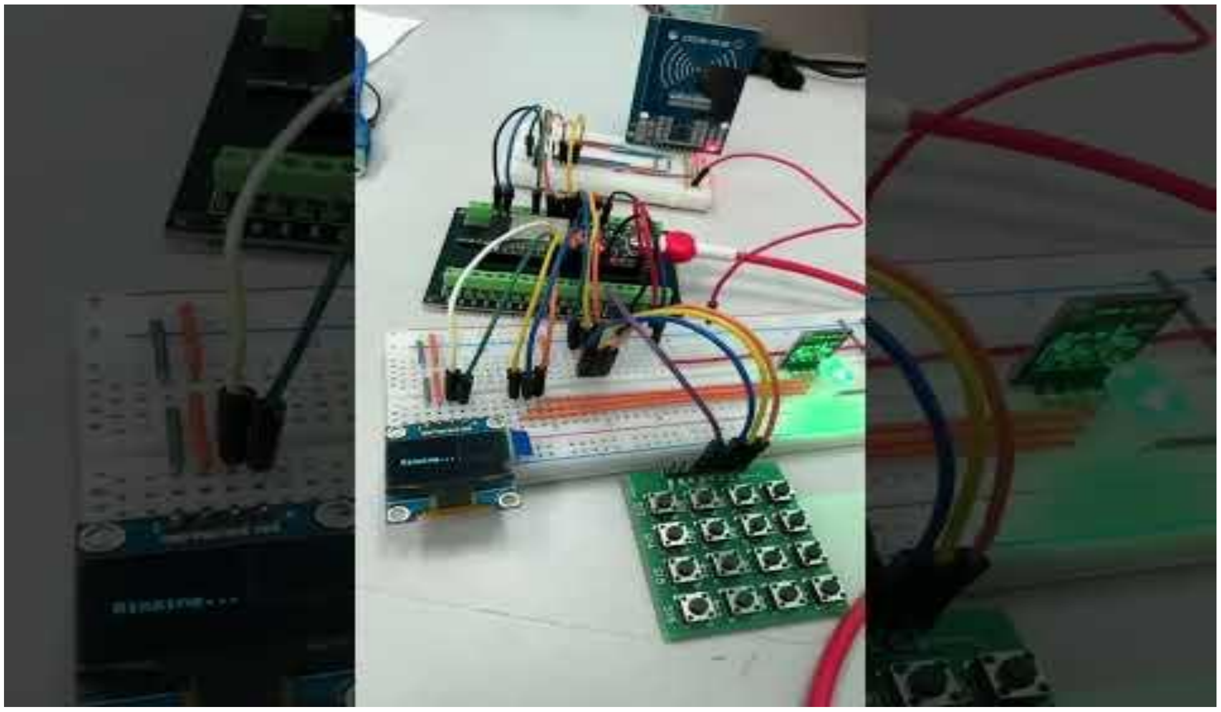
3. Integration:

- Ensure proper communication between components, such as I2C for OLED displays.
- Set unique addresses for I2C devices to avoid conflicts.
- Verify that all components are compatible with the Arduino and have the necessary libraries installed.
- Test the interaction between components, such as verifying that mode selection on the OLED display corresponds correctly with user input from push buttons.
- Optimize the Arduino code for efficiency, clarity, and responsiveness.
- Adjust wiring or component placement based on testing outcomes.

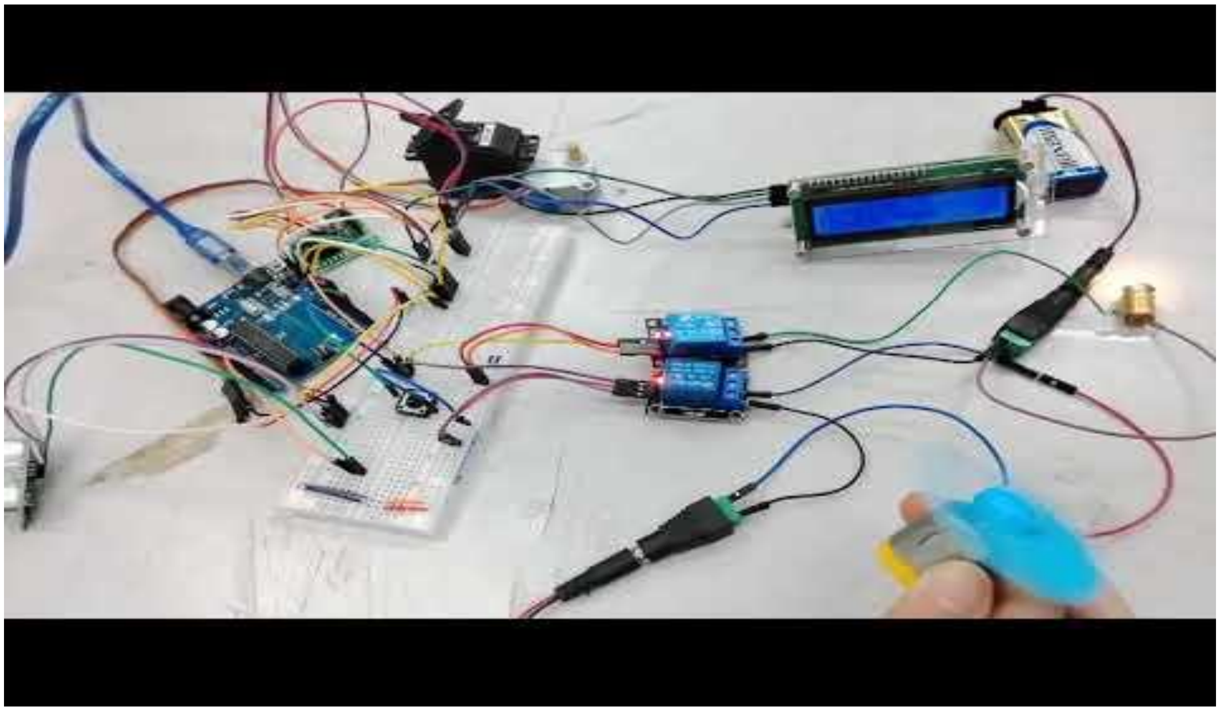
RESULTS

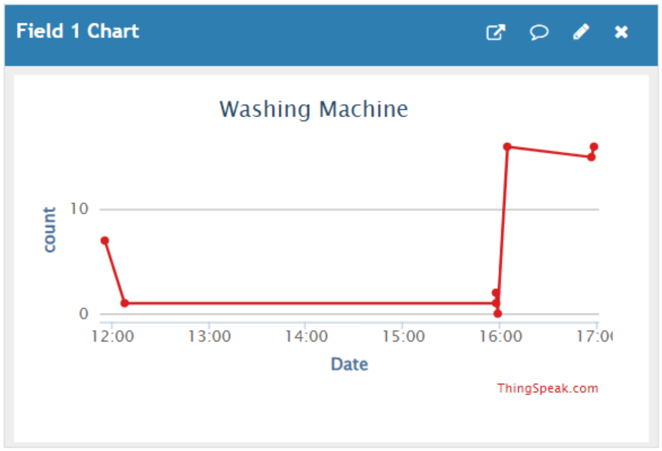
Video:

https://youtu.be/xq3M6l8AeTY?si=l3dPIEE_klKVhA1F



https://youtu.be/q0S5S7AgPpE?si=vDofg0PLuDzu4znN





ThingySpeak live chart

User interface:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

```
#define SCREEN_WIDTH 128  // OLED display width, in pixels
#define SCREEN_HEIGHT 64  // OLED display height, in pixels
#include <SPI.h>
#include <MFRC522.h>
#include "pitches.h"
#define BUZZER_PIN 2


#define SS_PIN 21  // MOSI of RFID
#define RST_PIN 5  // RST of RFID
#define btn_Up 17
#define btn_OK 16
#define btn_DOWN 4

const int WASH_LED_PIN = 27;
const int RINSE_LED_PIN = 14;
const int SPIN_LED_PIN = 12;

bool status_up = false;
bool status_ok = false;
bool status_down = false;

bool last_status_up = false;
bool last_status_ok = false;
bool last_status_down = false;

bool UP = false;
bool OKAY = false;
bool DOWN = false;

int page = 0;
int item = 1;

// Variables to store settings values
int waterTemperature = 30;  // Default value
int washDuration = 5;       // Default value
int spinSpeed = 800;        // Default value

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance.



// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

//song
int melody[] = {
  NOTE_C4, NOTE_C4,
  NOTE_D4, NOTE_C4, NOTE_F4,
  NOTE_E4, NOTE_C4, NOTE_C4,
  NOTE_D4, NOTE_C4, NOTE_G4,
  NOTE_F4, NOTE_C4, NOTE_C4,

  NOTE_C5, NOTE_A4, NOTE_F4,
  NOTE_E4, NOTE_D4, NOTE_AS4, NOTE_AS4,
  NOTE_A4, NOTE_F4, NOTE_G4,
  NOTE_F4
};

int durations[] = {
  4, 8,
  4, 4, 4,
  2, 4, 8,
  4, 4, 4,
  2, 4, 8,

  4, 4, 4,
  4, 4, 4, 8,
  4, 4, 4,
  2
};

//if RFID is correct then:
void authorizedMode() {

  while (true) {
    loopWashingMachine();
  }
}

//display message on Oled
void message(String text) {
  display.clearDisplay();
  display.setCursor(0, 10);
  display.print(text);
  display.display();
  //delay(1000);
}

void loopWashingMachine() {
  lcdSettings();

  status_up = digitalRead(btn_Up);
  status_ok = digitalRead(btn_OK);
  status_down = digitalRead(btn_DOWN);

  btnUpPressed();
  btnOkPressed();
  btnDownPressed();

  // FOR button up
  if (UP && page == 0) {
```

```
  UP = false;
  item--;
  if (item < 1) item = 4;
}
// FOR button down
if (DOWN && page == 0) {
  DOWN = false;
  item++;
  if (item > 4) item = 1;
}

// FOR button ok
if (OKAY) {
  OKAY = false;
  if (page == 0 && item == 1) {
    page = 1;
  } else if (page == 0 && item == 2) {
    page = 2;
  } else if (page == 0 && item == 3) {
    page = 3;
  } else if (page == 0 && item == 4) {
    page = 4;
  } else if (page == 1 && item == 1) {
    page = 0;
  } else if (page == 2 && item == 1) {
    page = 0;
  } else if (page == 3 && item == 1) {
    page = 0;
  } else if (page == 4 && item == 1) {
    page = 0;
  }
}

// Adjust values on settings pages
if (page == 1) {
  adjustSetting(waterTemperature, 10, 90, "Water Temperature");
} else if (page == 2) {
  adjustSetting(washDuration, 1, 10, "Wash Duration");
} else if (page == 3) {
  adjustSetting(spinSpeed, 400, 1200, "Spin Speed");
} else if (page == 4) {
  // Start Washing
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(10, 10);
  display.print("Washing Started");
  display.display();

  while (true) {
    status_ok = digitalRead(btn_OK);
```

```
      if (status_ok == 0) {
        OKAY = true;
        delay(50);
      }

      if (OKAY) {
        OKAY = false;
        Serial.println("S");
        //delay(10000);
        startWash();

        page = 0;
        return;
      }

      delay(100);
    }
  }

  Serial.println(item);
  delay(100);
}

void btnUpPressed() {
  if (status_up != last_status_up) {
    if (status_up == 0) {
      UP = true;
    }
    delay(50);
  }
  last_status_up = status_up;
}

void btnOkPressed() {
  if (status_ok != last_status_ok) {
    if (status_ok == 0) {
      OKAY = true;
    }
    delay(50);
  }
  last_status_ok = status_ok;
}

void btnDownPressed() {
  if (status_down != last_status_down) {
    if (status_down == 0) {
      DOWN = true;
    }
    delay(50);
  }
  last_status_down = status_down;
}

void lcdSettings() {
  if (page == 0) {
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(30, 0);
    display.print("Washing Machine Settings");

    if (item == 1) {
      display.setCursor(5, 17);
      display.setTextColor(WHITE);
      display.print("> Water Temperature");
    } else {
      display.setCursor(5, 17);
      display.setTextColor(WHITE);
      display.print("  Water Temperature");
    }

    if (item == 2) {
      display.setCursor(5, 27);
      display.setTextColor(WHITE);
      display.print("> Wash Duration");
    } else {
      display.setCursor(5, 27);
      display.setTextColor(WHITE);
      display.print("  Wash Duration");
    }

    if (item == 3) {
      display.setCursor(5, 37);
      display.setTextColor(WHITE);
      display.print("> Spin Speed");
    } else {
      display.setCursor(5, 37);
      display.setTextColor(WHITE);
      display.print("  Spin Speed");
    }

    if (item == 4) {
      display.setCursor(5, 47);
      display.setTextColor(WHITE);
      display.print("> Start Washing");

    } else {
      display.setCursor(5, 47);
      display.setTextColor(WHITE);
      display.print("  Start Washing");
    }
  } else if (page == 1) {
    // Water Temperature Settings
    adjustSettingScreen(waterTemperature, 10, 90, "Water Temperature");
  } else if (page == 2) {
    // Wash Duration Settings
    adjustSettingScreen(washDuration, 1, 10, "Wash Duration");
  } else if (page == 3) {
    // Spin Speed Settings
    adjustSettingScreen(spinSpeed, 400, 1200, "Spin Speed");
  } else if (page == 4) {
    // Start Washing
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(10, 10);
    display.print("Start Washing");
    display.setCursor(10, 30);
    display.print("Press OKAY to start");
  }

  display.display();
}

void adjustSetting(int &setting, int minValue, int maxValue, const char *settingName) {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(10, 10);
  display.print(settingName);
  display.setCursor(10, 30);
  display.print("Current: ");
  display.print(setting);
  display.setCursor(10, 50);
  display.print("Press UP/DOWN to adjust");
  display.display();
  delay(100);

  while (true) {
    status_up = digitalRead(btn_Up);
    status_down = digitalRead(btn_DOWN);
    status_ok = digitalRead(btn_OK);

    if (status_up == 0) {
      UP = true;
      delay(50);
    } else if (status_down == 0) {
      DOWN = true;
      delay(50);
    } else if (status_ok == 0) {
      OKAY = true;
      delay(50);
```

```
}

if (UP) {
  UP = false;
  setting++;
  if (setting > maxValue) setting = minValue;
}
if (DOWN) {
  DOWN = false;
  setting--;
  if (setting < minValue) setting = maxValue;
}

display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(10, 10);
display.print(settingName);
display.setCursor(10, 30);
display.print("Current: ");
display.print(setting);
display.setCursor(10, 50);
display.print("OKAY to confirm");

display.display();

if (OKAY) {
  OKAY = false;
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(10, 10);
  display.print(settingName);
  display.setCursor(10, 30);
  display.print("Set to ");
  display.print(setting);
  display.setCursor(10, 50);
  display.print("OKAY to continue");
  display.display();
  while (true) {
    status_ok = digitalRead(btn_OK);
    if (status_ok == 0) {
      OKAY = true;
      delay(50);
    }
    if (OKAY) {
      OKAY = false;
      page = 0;
      return;
    }
  }
```

```
    }
    delay(100);
  }
}

void adjustSettingScreen(int &setting, int minValue, int maxValue, const char *settingName) {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(10, 10);
  display.print(settingName);
  display.setCursor(10, 30);
  display.print("Current: ");
  display.print(setting);
  display.setCursor(10, 50);
  display.print("Press UP/DOWN to adjust");
  display.display();

  while (true) {
    status_up = digitalRead(btn_Up);
    status_down = digitalRead(btn_DOWN);
    status_ok = digitalRead(btn_OK);

    if (status_up == 0) {
      UP = true;
      delay(50);
    } else if (status_down == 0) {
      DOWN = true;
      delay(50);
    }

    if (UP) {
      UP = false;
      setting++;
      if (setting > maxValue) setting = minValue;
    }
    if (DOWN) {
      DOWN = false;
      setting--;
      if (setting < minValue) setting = maxValue;
    }

    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(10, 10);
    display.print(settingName);
    display.setCursor(10, 30);
    display.print("Current: ");
    display.print(setting);
    display.setCursor(10, 50);
    display.print("Press OKAY to confirm");

    display.display();

    if (status_ok == 0) {
      OKAY = true;
      delay(50);
    }

    if (OKAY) {
      OKAY = false;
      display.clearDisplay();
      display.setTextSize(1);
      display.setTextColor(WHITE);
      display.setCursor(10, 10);
      display.print(settingName);
      display.setCursor(10, 30);
      display.print("Set to ");
      display.print(setting);
      display.setCursor(10, 50);
      display.print("Press OKAY to continue");
      display.display();

      while (status_ok == 0) {
        status_ok = digitalRead(btn_OK);
        delay(50);
      }

      page = 0;
      return;
    }

    delay(100);
  }
}
void startWash() {
  Serial.println("S");
  wash();
  rinse();
  spin();
  complete();
}
void wash() {
  Serial.println("S");
  display.clearDisplay();
  display.setCursor(10, 30);
  display.print("Washing...");
  display.display();
  statusLed(1);
  delay(10000);
}
```

```
void rinse() {
  Serial.println("S");
  display.clearDisplay();
  display.setCursor(10, 30);
  display.print("Rinsing...");
  display.display();
  statusLed(2);
  delay(10000);
}
void spin() {
  Serial.println("S");
  display.clearDisplay();
  display.setCursor(10, 30);
  display.print("Spinning...");
  display.display();
  statusLed(3);
  delay(10000);
}
void complete() {
  Serial.println("S");
  display.clearDisplay();
  display.setCursor(10, 30);
  display.print("Washing complete");
  display.display();
  statusLed(4);
  delay(10000);
  statusLed(0);
  playSong();
  delay(10000);
}
void statusLed(int mode) {
  switch (mode) {
    case 0:
      digitalWrite(WASH_LED_PIN, LOW);
      digitalWrite(RINSE_LED_PIN, LOW);
      digitalWrite(SPIN_LED_PIN, LOW);
      Serial.println('P');  // Washing
      break;
    case 1:
      digitalWrite(WASH_LED_PIN, HIGH);
      digitalWrite(RINSE_LED_PIN, LOW);
      digitalWrite(SPIN_LED_PIN, LOW);
      Serial.println('W');  // Washing
      break;
    case 2:
      digitalWrite(WASH_LED_PIN, LOW);
      digitalWrite(RINSE_LED_PIN, HIGH);
      digitalWrite(SPIN_LED_PIN, LOW);
      Serial.println('R');  // Washing
      break;
    case 3:
      digitalWrite(WASH_LED_PIN, LOW);
      digitalWrite(RINSE_LED_PIN, LOW);
      digitalWrite(SPIN_LED_PIN, HIGH);
      Serial.println('S');  // Washing
      break;
    case 4:
      digitalWrite(WASH_LED_PIN, HIGH);
      digitalWrite(RINSE_LED_PIN, HIGH);
      digitalWrite(SPIN_LED_PIN, HIGH);
      Serial.println('C');  // Washing
      break;
  }
}

void playSong() {
  int size = sizeof(durations) / sizeof(int);

  for (int note = 0; note < size; note++) {
    //to calculate the note duration, take one second divided by the note type.
    // e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int duration = 1000 / durations[note];
    tone(BUZZER_PIN, melody[note], duration);

    //to distinguish the notes, set a minimum time between them.
    //the note's duration + 30% seems to work well:
    int pauseBetweenNotes = duration * 1.30;
    delay(pauseBetweenNotes);

    //stop the tone playing:
    noTone(BUZZER_PIN);
  }
}

void setup() {

  Serial.begin(115200);
  Wire.begin(25, 26);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {  // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for (;;)
      ;
  }
  delay(2000);
  pinMode(btn_Up, INPUT_PULLUP);
  pinMode(btn_OK, INPUT_PULLUP);
  pinMode(btn_DOWN, INPUT_PULLUP);
  pinMode(WASH_LED_PIN, OUTPUT);
  pinMode(RINSE_LED_PIN, OUTPUT);
  pinMode(SPIN_LED_PIN, OUTPUT);

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 10);
  message("welcome");

  SPI.begin();         // Initiate SPI bus
  mfrc522.PCD_Init();  // Initiate MFRC522
  Serial.println("Put your card to the reader...");
  //Serial.println();
  delay(3000);
}

void loop() {
  // Look for new cards
  if (!mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  // Select one of the cards
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  // Show UID on the serial monitor
  Serial.print("UID tag :");
  String content = "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    //display.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    //display.print(mfrc522.uid.uidByte[i], HEX );
    //message(String(HEX));
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  //Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();
  if (content.substring(1) == "83 B2 1A 11") {
    Serial.println("Authorized access");
    //Serial.println();
    message("Wash Away~");
    authorizedMode();
  } else {
    //Serial.println("Access denied");
    message("Access denied");
    delay(1000);
    message("Play a game for free wash");
    delay(5000);
    Serial.println('G');
    while (1) {
      if (Serial.available() > 0) {
        char receivedChar = Serial.read();
        if (receivedChar == 'F') {
          authorizedMode();
        }
      }
    }
  }
}
```

Game:

```
#include <U8glib.h>
//#include <Wire.h>

// Initialize the OLED display
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE);

// Joystick pin
int yPin = A1;

// Paddle sizes and positions
int playerPaddleY = 30;
int aiPaddleY = 30;
int paddleHeight = 25;
int paddleWidth = 4;

// Ball position, speed, and size
int ballX = 64, ballY = 32;
int ballSpeedX = 3, ballSpeedY = 3;
int ballSize = 5;

// Scoring and game state
int playerScore = 0;
bool gameOverFlag = true;
char finish = 'H';

void setup() {
  Serial.begin(115200);
  pinMode(yPin, INPUT);
}

void loop() {
  if (Serial.available()){
    char data =Serial.read();
    if (data=='G'&& finish=='H')
    {
      resetGame();
      finish='A';
    }
  }
  Serial.println(finish);
  if (gameOverFlag) {
    displayGameOver();
    if(finish=='A'){
      finish = 'F';
    }
    return;
  }

  int yValue = analogRead(yPin);

  // Move player paddle based on joystick
  if (yValue < 400) playerPaddleY -= 3;
  if (yValue > 600) playerPaddleY += 3;
  playerPaddleY = constrain(playerPaddleY, 0, 64 - paddleHeight);

  // Simple AI for opponent paddle
  aiPaddleY = ballY - paddleHeight / 2;
  aiPaddleY = constrain(aiPaddleY, 0, 64 - paddleHeight);

  // Move ball
  ballX += ballSpeedX;
  ballY += ballSpeedY;

  // Collision with top and bottom
  if (ballY <= 0 || ballY >= 64 - ballSize) ballSpeedY = -ballSpeedY;

  // Collision with paddles
  if (ballX <= paddleWidth && ballY >= playerPaddleY && ballY <= playerPaddleY + paddleHeight)
{
    ballSpeedX = -ballSpeedX;
    playerScore++;
    increaseDifficulty();
  }
  if (ballX >= 128 - paddleWidth - ballSize && ballY >= aiPaddleY && ballY <= aiPaddleY +
paddleHeight) ballSpeedX = -ballSpeedX;

  // Check if the player misses the ball
  if (ballX < 0) {
    gameOverFlag = true;
  }

  // Reset ball if it goes off screen to the right
  if (ballX > 128 - ballSize) {
    ballX = 64;
    ballY = 32;
    ballSpeedX = -ballSpeedX;
  }

  // Drawing
  u8g.firstPage();
  do {
    drawGame();
  } while (u8g.nextPage());
}

void drawGame() {
  // Draw player paddle
  u8g.drawBox(0, playerPaddleY, paddleWidth, paddleHeight);

  // Draw AI paddle
  u8g.drawBox(128 - paddleWidth, aiPaddleY, paddleWidth, paddleHeight);

  // Draw ball
  u8g.drawBox(ballX, ballY, ballSize, ballSize);

  // Draw score
  u8g.setPrintPos(60, 10);
  u8g.print(playerScore);
}

void displayGameOver() {
  u8g.firstPage();
  do {
    u8g.setFont(u8g_font_6x10);
    u8g.drawStr(40, 32, "GAME OVER");
    u8g.setPrintPos(40, 48);
    u8g.print("Score: ");
    u8g.print(playerScore);
  } while (u8g.nextPage());
}

void resetGame() {
  playerPaddleY = 30;
  aiPaddleY = 30;
  ballX = 64;
  ballY = 32;
  ballSpeedX = 3;
  ballSpeedY = 3;
  playerScore = 0;
  gameOverFlag = false;
}

void increaseDifficulty() {
  if (playerScore % 5 == 0) {  // Increase speed every 5 points
    ballSpeedX;
  }
}
```

Washing:

```
#include <Stepper.h>
#include <Servo.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

const int stepsPerRevolution = 2048;
Stepper stepper(stepsPerRevolution, 10, 11, 12, 13);

Servo myservo;
int pos = 0;

LiquidCrystal_I2C lcd(0x3F, 16, 2); // I2C address 0x27, 16 column and 2 rows

const int limitSwitchPin = 2;
const int RELAY_PINL = 5;
const int RELAY_PINK = 6;
int fix=15;

//ultrasonic sensor
const int ultrasonics = A0;
unsigned int ADCValue;



void setup() {
  Serial.begin(115200);
  myservo.attach(7);

  lcd.begin(16, 2);
  lcd.print("Washing Machine");

  pinMode(limitSwitchPin, INPUT_PULLUP);
  pinMode(RELAY_PINL, OUTPUT);
  pinMode(RELAY_PINK, OUTPUT);

}


void loop() {
    Serial.println(fix);
    ADCValue = analogRead(ultrasonics);
    // ADCValue *=3;
    //Serial.print("Present Length is: ");
    //Serial.print(ADCValue, DEC);
    //Serial.println("mm");


  //ultrasonic
  if(ADCValue < 80){
      digitalWrite(RELAY_PINL, HIGH);
      digitalWrite(RELAY_PINK, HIGH);
  }else{
      digitalWrite(RELAY_PINL, LOW);
      digitalWrite(RELAY_PINK, LOW);
  }

  int limitSwitchState = digitalRead(limitSwitchPin);
  //Serial.println(limitSwitchState);

  if (limitSwitchState == LOW) {
  lcdPrint("Door closed :)", "ready soon!");
  int washDuration = (1000);
  int rinseDuration = (1000);
  int spinDuration =(1000);

  wash(washDuration);
  rinse(rinseDuration);
  spin(spinDuration);

  lcdPrint("Process", "Complete!");
  complete();
  } else{
    lcdPrint("door open", "-_-");
    }

}

void wash(int duration) {
  lcdPrint("Washing", "1 sec");
  lcdPrint("Washing","In progress");
  waterpipe();
  lcdPrint("Washing", "In Progress");

  stepper.setSpeed(10);
  for (int i = 0; i < duration; i++) {
    stepper.step(1);
    delay(10);
  }

  lcdPrint("Washing", "Complete");
}

void rinse(int duration) {
  lcdPrint("Rinsing", "1 sec");
  lcdPrint("Rinsing","In progress");
  waterpipe();
  lcdPrint("Rinsing", "In Progress");

  stepper.setSpeed(10);
  for (int i = 0; i < duration; i++) {
    stepper.step(1);
    delay(10);
  }

  lcdPrint("Rinsing", "Complete");
}

void spin(int duration) {
  lcdPrint("Spinning", "1 sec");
  lcdPrint("Spinning", "In progress");
  lcdPrint("Spinning", "In Progress");

  stepper.setSpeed(50);
  for (int i = 0; i < duration; i++) {
    stepper.step(1);
    delay(5);
  }

  lcdPrint("Spinning", "Complete");
}

void complete() {
  fix++;
  lcdPrint("clothes", "Complete");
  //Serial.print(fix);

}


void lcdPrint(String line1, String line2) {
  lcd.init();                      // initialize the lcd
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
}

void waterpipe() {
  lcdPrint("water valve", "open");
  for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos);
    delay(15);
  }
  lcdPrint("water valve", "close");
  for (pos = 180; pos >= 0; pos -= 1) {
  myservo.write(pos);
  delay(15);}
```

```
}
```

Serial Communication:

```
import serial
import threading
import tkinter as tk
import requests

stop = False
espdata = ""
gamedata = ""
washdata = ""
baud_rate = 115200


def user():
    global stop, espdata
    esp32_serial_port = "COM12"
    esp32_ser = serial.Serial(esp32_serial_port, baud_rate, timeout=1)
    while not stop:
        got_or_not = esp32_ser.readline().decode('utf-8').strip()
        if got_or_not != "":
            espdata = got_or_not
        data = gamedata
        esp32_ser.write(data.encode())
    esp32_ser.close()


def game():
    global stop, gamedata, espdata
    game_serial_port = "COM9"
    game_ser = serial.Serial(game_serial_port, baud_rate, timeout=1)
    while not stop:
        got_or_not = game_ser.readline().decode('utf-8').strip()
        if got_or_not != "":
            gamedata = got_or_not
        game_ser.write(espdata.encode())
    game_ser.close()


def wash():
    global stop, washdata, espdata
    wash_serial_port = "COM17"
    wash_ser = serial.Serial(wash_serial_port, baud_rate, timeout=1)
    while not stop:
        got_or_not = wash_ser.readline().decode('utf-8').strip()
        if got_or_not != "":
            washdata = got_or_not
        data = espdata + '\n'
        wash_ser.write('S'.encode())
    wash_ser.close()

def fixed():
    global washdata
    fix=4
    while not stop:
        if washdata != fix:
            fix=washdata
            requests.get("https://api.thingspeak.com/update?api_key=JWT4DD2CA5ETIWDL&field1=
"+fix)



def update_data_label():
    data_label.config(
        text=f'ESP32: {espdata}\nGame: {gamedata}\nWash: {washdata}')
    root.after(100, update_data_label)


def on_closing(event=None):
    global stop
    stop = True
    root.destroy()


def start_reading():
    global stop
    stop = False
    espthread = threading.Thread(target=user)
    espthread.start()
    gamethread = threading.Thread(target=game)
    gamethread.start()
    washthread = threading.Thread(target=wash)
    washthread.start()
    fixedthread= threading.Thread(target=fixed)
    fixedthread.start()
    update_data_label()


root = tk.Tk()
root.title("Serial Data Monitor")
root.geometry("800x200")

frame_esp = tk.Frame(root, bg="#FFD700", padx=10, pady=10)
frame_esp.grid(row=0, column=0, padx=10, pady=10)

frame_game = tk.Frame(root, bg="#90EE90", padx=10, pady=10)
frame_game.grid(row=0, column=1, padx=10, pady=10)
```

```
frame_wash = tk.Frame(root, bg="#FF6347", padx=10, pady=10)
frame_wash.grid(row=0, column=2, padx=10, pady=10)

data_label = tk.Label(root, text="", font=("Helvetica", 12), justify="left")
data_label.grid(row=1, column=0, columnspan=3, padx=10, pady=10)

button_frame = tk.Frame(root, padx=10, pady=10, bg="#F0F8FF")
button_frame.grid(row=2, column=0, columnspan=3)

start_button = tk.Button(
    button_frame, text="Start Reading", command=start_reading)
start_button.grid(row=0, column=0, pady=10, padx=10)

stop_button = tk.Button(button_frame, text="Stop", command=on_closing)
stop_button.grid(row=0, column=1, pady=10, padx=10)

root.bind('q', on_closing)
root.mainloop()
```

DISCUSSION

The automated washing machine system, as described in the project, represents a commendable integration of advanced technologies to transform a commonplace household appliance into a smart and user-centric solution. By incorporating RFID technology for user authentication, the project ensures secure and personalized interaction, enhancing the overall user experience. The utilization of LCD displays, servo motors, buzzers, and the Arduino microcontroller further demonstrates a sophisticated orchestration of various components. The inclusion of personalized RFID card interactions not only contributes to the system's security but also showcases an innovative approach to user authentication in the context of home appliances. The Arduino microcontroller serves as the central intelligence, facilitating seamless communication between diverse components. Moreover, this project's emphasis on practical applications of embedded systems and smart technologies in household appliances aligns with the broader trend of advancing home automation. This not only underscores the project's technical prowess but also positions it within the evolving landscape of user-centric smart home solutions.

User interfacing:

The user interfacing part serves as a crucial aspect of the overall project, contributing significantly to the user interface and interaction mechanisms of the automated washing machine system. The inclusion of libraries such as Wire, Adafruit_GFX, and Adafruit_SSD1306 facilitates the integration of OLED displays and RFID functionality. RFID technology is employed for user authentication, enhancing security through personalized RFID card interactions. The OLED displays are intelligently utilized to present a user-friendly interface, allowing users to navigate through washing machine settings and configurations. Buttons for UP, OK, and DOWN are defined to handle user inputs, providing a tactile means of interaction. The interaction between the user, the OLED displays, and the physical buttons can be seen as it continuously monitors the button states, adjusts settings based on user inputs, and updates the display accordingly. The function of LCD is particularly noteworthy, dynamically displaying the current washing machine settings and guiding users through configuration options. Additionally, we use buzzer for auditory feedback during specific events, enhancing the overall user experience. We implement various functions for adjusting settings, handling button presses, and executing washing machine cycles. RFID is used at the beginning of the washing machine process. When RFID authentication is successful, it allows the user to proceed with the washing process. But when unauthorized access is detected, prompting the user to play a game for free wash. Overall, this user interfacing plays a pivotal role in creating a seamless and interactive experience for users interacting with the automated washing machine system. We can effectively combine visual and auditory feedback with physical button inputs, showing a thoughtful approach to user interface design in the context of smart home appliances.

Process:

Leveraging a stepper motor, servo, and an ultrasonic sensor, the system orchestrates these processes efficiently. The ultrasonic sensor is used to detect the distance of an object accordingly. If the ultrasonic sensor detects a person or the user, it will automatically light up the bulb and which on the fan. In this project, we use servo motor to control the water valve, opening and closing it during the water supply phase. This demonstrates a realistic approach to simulating the water input and drainage aspects of the washing machine operation. The washing process involves the stepper motor agitating the clothes, simulating a typical washing cycle. Similarly, the rinse and spin functions utilize the stepper motor to perform their respective actions, emulating the corresponding stages in a conventional washing machine. The LCD screen is effectively utilized to provide real-time status updates to the user, such as the current process being executed and completion notifications. Furthermore, the code incorporates safety features by checking if the washing machine door is closed before initiating the washing cycle. In summary, we effectively translate the washing, rinsing, and spinning processes into a programmatic representation suitable for an automated washing machine system. The integration of various components, along with the clear status updates on the LCD display, enhances the user's understanding of the ongoing laundry processes.

Serial communication:

Besides, we managed to establish a serial communication interface between three different devices, represented by COM ports for ESP32, game and washing machine. The code employs the `serial` library for serial communication and utilizes the `threading` module to create separate threads for reading data from each device concurrently. The data from the ESP32, game, and washing machine is stored in global variables which are `espdata`, `gamedata`, and `washdata`, respectively. It is to facilitate communication between threads. Additionally, a fixed thread is implemented to periodically send data to the ThingSpeak IoT platform using the `requests` library. This is because, if the use of the washing machine reaches a certain level, it will need to be serviced. Other than that, the Tkinter GUI application provides a user-friendly interface to monitor the real-time data from the three devices. The GUI consists of three frames, each representing one device and labeled with distinct colors. The main data label in the GUI is updated in real-time to display the information received from the ESP32, game, and washing machine. In summary, we manage to demonstrate an effective implementation of concurrent serial communication, providing a robust interface for monitoring and interacting with data from multiple devices. The use of threading enhances the efficiency of data reading, ensuring that each device's data can be processed simultaneously, and the Tkinter GUI enhances the user experience by visualizing real-time data updates.

In summary, the automated washing machine system stands as a compelling example of how technology can be harnessed to elevate the functionality and user experience of everyday appliances, making strides toward more intelligent and interconnected home environments.

SAFETY

Our washing machine took safety into great consideration. For starters, we made sure to think about the environment of where the washing machine would be located as a whole. For example, when you walk into the launderette, ultrasonic sensor will sense you and automatically turn on the lights and fan. This will ensure the security of the customers, as we do not want them to do their laundry in the dark.

Next, the usage of card authorization. The implementation of card authorization allows only certain people to access the washing machine, although people without the card could play a game to access it. The card authorization makes it easier and faster to use the washing machine.

On top of that, we made sure that the washing machine would only start when the customer presses the start button. This will prevent the washing machine starting without the customer giving input.

Lastly, we implemented future maintenance in our washing machine. Washing machines are not self-sustaining; they would need servicing sometimes. We implemented using thingyspeak, to accumulate the number of times the washing machine has been used. Once the number goes up to 100, the washing machine will be due for servicing. If servicing does not take place, washing machine might break down and worst-case scenario, have the washing machine catches fire due to excess lint or fluff build up over the years.

That said, we believe our washing machine has taken note of the safety and security of the customers and the maintenance of it.

CONCLUSION

In conclusion, our team has successfully built a functional washing machine prototype. This washing machine includes elements of the Internet of Things (IoT) and various communication protocols, along with the interfacing of multiple sensors and actuators, all integrated into a single functioning system. The washing machine prototype has many opportunities for improvement, serving as the foundational design for a more advanced and improved future model.

The significance of our project lies in its contribution to the evolution of household appliances. The incorporation of IoT not only enhances user convenience but also aligns with the growing trend of interconnected devices in modern homes. The various communication protocols used showcase our commitment to creating a well-integrated system.

Throughout the project, we encountered challenges that required innovative solutions, reinforcing our problem-solving skills. These experiences have not only added depth to our learning but also prepared us for future endeavors in mechatronic engineering.

As we acknowledge the opportunities for improvement in our prototype, we recognize the valuable lessons learned and unexpected findings that have further enriched our understanding of system integration. Moving forward, this project not only stands as a testament to our technical capabilities but also as a steppingstone towards more advanced and impactful innovations in the realm of smart home technology.

RECOMMENDATIONS

There are several improvements that can be implemented in this washing machine prototype. Firstly, the overall design of the washing machine can be enhanced by giving more consideration to the functioning of the washing system, similar to the attention given to the user interface. For example, door locking mechanisms can be incorporated into the machine. Additionally, making the machine smarter by adding various types of sensors would allow it to choose appropriate washing cycles based on the types and weights of clothes.

Secondly, appropriate components should be used in the electrical circuit to represent the working of the washing machine. For instance, a water valve can be employed for water supply, and a DC motor can be utilized for the rotation of the washing machine drum.

Thirdly, the washing machine prototype can be built to resemble an actual washing machine by creating a body and housing for the electrical components. This will make the prototype more convincing and understandable.

Last but not least, communication between different Arduino boards can be improved. This can be achieved through the coding of Arduino sketches. Different communication methods can be analyzed to choose the most appropriate method of communication to build a well-integrated system.

REFERENCES

ArduinoGetStarted.com (n.d.). *Arduino - Relay*. Arduino Get Started. Retrieved January 14, 2024, from https://arduinogetstarted.com/tutorials/arduino-relay

Benne de Bakker (n.d.). *28BYJ-48 Stepper Motor with ULN2003 Driver and Arduino Tutorial*. Makerguides.com. Retrieved January 15, 2024, from https://www.makerguides.com/28byj-48-stepper-motor-arduino-tutorial/

Instructables (n.d.). *Tutorial: How to Use Analog Ultrasonic Distance Sensor US-016 With Arduino UNO*. Autodesk Instructables. Retrieved January 15, 2024, from https://www.instructables.com/Tutorial-How-to-Use-Analoge-Ultrasonic-US-016/

APPENDICES

ACKNOWLEDGEMENT

We extend our sincere gratitude to Dr. Nadzril Bin Sulaiman, Dr. Wahju Sediono, and Assoc. Prof. Dr. Zulkifli Bin Zainal Abidin, our esteemed lecturers for the mechatronic system integration 1 (MCTA 3203) course. Their guidance was invaluable during the execution of the experiment, and we greatly appreciate their support and insights.

Special thanks are also due to the lab technician for their assistance in preparing the components, contributing significantly to the successful completion of the washing machine project.

These individuals have played a crucial role in our learning journey, imparting knowledge on component interfacing, Arduino programming, data acquisition, and IoT applications. We sincerely appreciate their dedication to our education and the invaluable contributions they've made to our understanding of mechatronic system integration.

STUDENT'S DECLARATION

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 person.

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 contributor to the report.

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

Signature: Ghousiah Name: Noorul Ghousiah Binti Noordeen Sahib Matric Number: 2118298	Read	✓
	Understand	✓

Contribution: Recommendation, Conclusion	Agree	/
Signature: hanis Name: hanis Matric Number: 2020590 Contribution:	Read	/
	Understand	/
	Agree	/
Signature: Afiq Name: Muhammad Afiq bin Ahmad Matric Number: 2115203 Contribution: abstract, procedure, results	Read	/
	Understand	/
	Agree	/
Signature: ayunaziera Name: Nur Ayu Naziera Binti Rosli Matric Number: 2119202 Contribution: discussion	Read	/
	Understand	/
	Agree	/
Signature:Sarah Name: Sarah Aishah binti Sa’aid Hazley Matric Number: 2117600 Contribution: introduction,safety	Read	/
	Understand	/
	Agree	/