Vending Machine Report:

This report is divided into 6 chapters to explain every single point

* **Ch1 : The Usage of the Vending machine & its importance**

Vending machines are automated devices that dispense a variety of products, typically food, beverages, and other small items, upon receiving payment. They are important and widely used for several reasons:

1. **Convenience:** Vending machines provide quick and easy access to products without the need for human interaction. This is especially useful in locations where traditional stores may not be available, such as offices, schools, hospitals, and public transportation hubs.
2. **24/7 Availability:** Unlike many retail stores, vending machines operate around the clock, offering consumers the ability to purchase items at any time of the day or night.
3. **Space Efficiency:** Vending machines require relatively small spaces, making them ideal for places where a full store would be impractical.
4. **Labor Cost Savings:** They eliminate the need for a cashier, reducing labor costs for the business owner.
5. **Variety:** Modern vending machines can offer a wide range of products, from snacks and drinks to personal hygiene items, electronics, and even fresh food.
6. **Technology Integration:** Advanced vending machines incorporate features such as cashless payment options, touch screens, and even internet connectivity for inventory management and remote monitoring.
7. **Impulse Purchases**: They capitalize on impulse buying behavior, providing quick solutions for immediate needs, whether it's a snack or a forgotten item like a phone charger.
8. **Accessibility:** Vending machines can be placed in various high-traffic areas, making products accessible to a larger audience without the need for a full retail setup.
9. **Hygiene and Safety:** Especially in situations where minimizing human contact is important (e.g., during a pandemic), vending machines offer a more hygienic alternative to traditional retail.
10. **Marketing and Branding:** They provide opportunities for brands to showcase their products in a unique and eye-catching manner, often using the machine itself as an advertising platform.

In summary, vending machines are valuable for their convenience, efficiency, and ability to provide a wide range of products in various locations, catering to the immediate needs of consumers.

* **Ch2 : The components used to create the vending machine**:

-**For the outer structure and look:**

- Wood (for the shelves )  
 - Glass (placed in front of the products)  
 - Hinges (to open and close the door)

- Screws

- Paint

- Cover

- Springs ( # 4 placed on the mg90 servos)

- lock ( hidden from customers )  
**-For the Arduino connections and inner mechanisms:**

-**Arduino Mega (#1) :** The Arduino Mega is a microcontroller board based on the ATmega2560, and it is widely used for various electronics projects due to its greater number of input/output pins and larger memory compared to other Arduino boards like the Arduino Uno , all devices will be connected to it to work and to enable us to do the coding.

-**Breadboard (#1):** used for building and testing electronic circuits without soldering , all the components that need to be connected to the GND and VCC will be connected to it, in addition to the placement of the resistors .

-**RFID (#1) :** RFID systems use radio waves to communicate between a tag attached to an item and a reader device, which captures the data stored on the tag.

-**Keypad (#1) :** Keypads are input devices consisting of a set of buttons arranged in a grid, which are used to input data or commands into a system, here we use it to enter the password , refill the machine , choose the product to be bought and change the price of the products .

-**LCD i2c (#1):** LCD (Liquid Crystal Display) technology is widely used in various applications due to its ability to display clear, sharp images and text with low power consumption, and here it displays the steps to be done to enable the customer to buy a product or the vendor to refill it ,open it or change products prices .

-**LEDs (#6) :** LEDs (Light Emitting Diodes) are versatile, energy-efficient light sources used in a wide variety of applications across different fields such as decoration in our project to light the shelves and to specify if the password is correct or not.

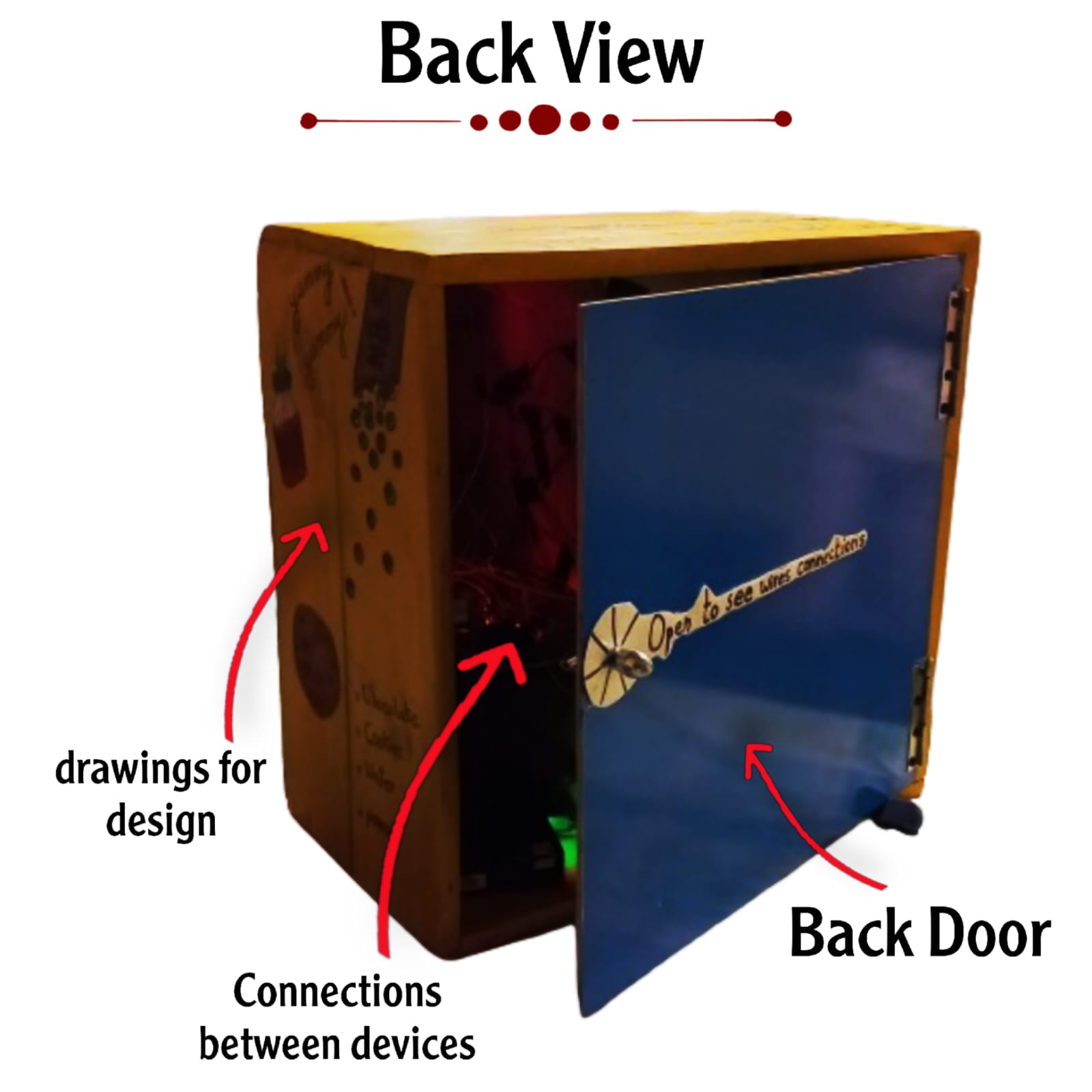
-**sg90 Servo Motor (#1):** The SG90 is a popular, small, and inexpensive servo motor used to control the opening and closing of the lock in our project (it can rotate from 0 to 180 degrees) and is connected to the lock through a metal wire.

**-mg90 Servo Motors (#4):**  The MG90 is bigger than the sg90 servo motor and can adjust to heavier things too. We will use it to rotate the springs placed on it to make the treats move to the front and fall (these servo motors can rotate 360 degrees).

-**Buzzer (#1) :** Used to make a buzzing sound if the password was written incorrectly .

* **Ch3: The Outer Design and Sequence of Events :**

****

****

* **The Sequence of Events :**

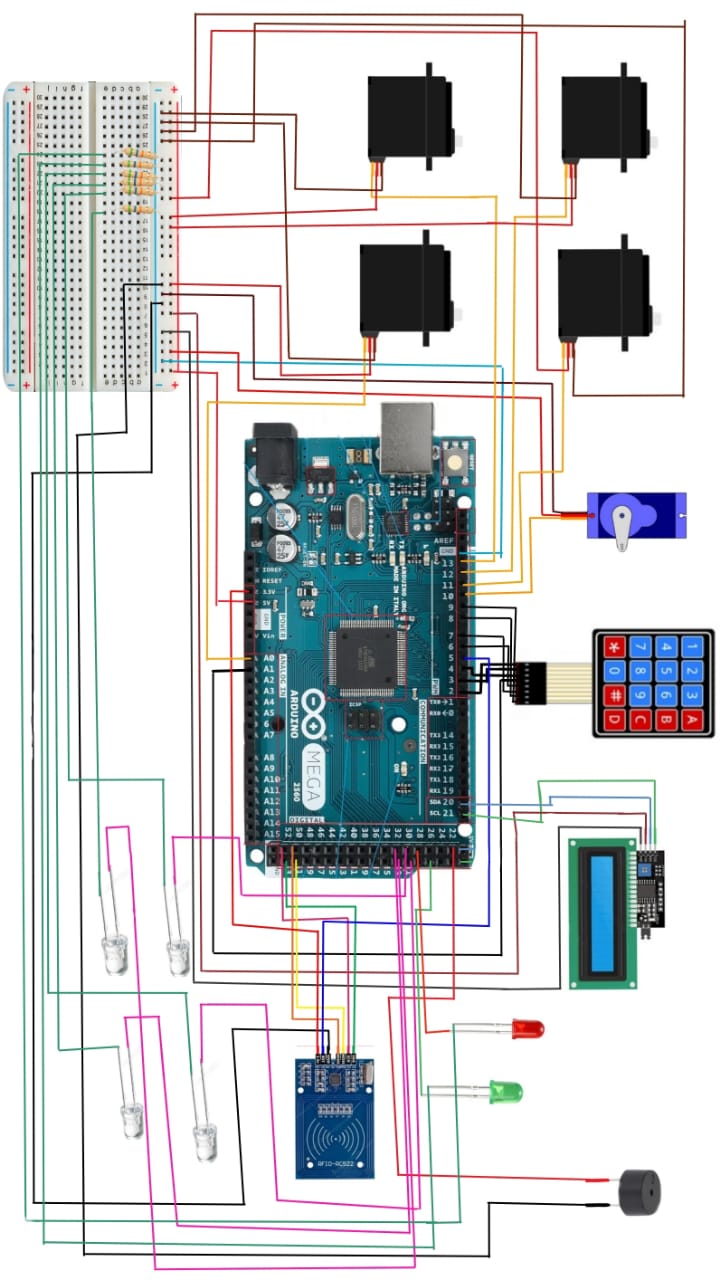
Concerning Customer:

- The customer can buy the product through the rfid card that contains a balance , he should follow the instructions needed to choose the wanted product.  
The system will test if there is a sufficient balance in the card, if yes the customer will be able to buy and the servos will rotate according to what item is chosen.  
Of course his balance will be decreased and presented too.  
Finally the customer will take his treat from the lower door.

Concerning the vendor or worker:

-He will be able to open the machine using two methods. Either by entering a password he knows through the keypad or by using the rfid tag.  
When the machines door is unlocked he can refill the machine by rotating the Springs backwards.  
Of course later on he can lock the door again.  
  
If he used the tag, he will have an extra option which is the CHANGE PRICE he could choose the item to be change its price and the instructions will be very clear.  
  
If any problems related to wires occurred, the programmer can open the backdoor and fix the connections too.

NOTE : A power bank provider is used to supply power to our vending machine so that the project looks more professional

* **Ch4: The Connections between the used devices + CODE explanation :**

**-Mg90 Servos (#4) :** we have four mg90 servo motors that are connected to pins :

The other two pins in each of the four mg90 servos are connected like so:  
  
- red wire : to 5v (vcc)

-brown wire to: GND

- A0

- 11

- 12

- 13

On each mg90 servo place a spring that will be used to place the products in between its loops.

**-Sg90 Servos (#1) :** we have one sg90 servo that has the following connections:  
 - Orange wire on pin 10  
 - Brown on GND

- Red on VCC

On its plastic wing place a metal wire that will be connected on its other end to the lock hand. When the motor rotates for 90 degrees the metal wire will move backward pulling the hand of the lock backwards too.

**-Keypad (#1) :** The Keypad has 8 pins that are connected in this way :

- R1 -> pin 2

- R2 -> pin 3

- R3 -> pin 4

- R4 -> pin A1

- C1 -> pin 6

- C2 -> pin 7

- C3 -> pin 8

- C4 -> pin 9

**- LCD i2c (#1) :**  The connections of the LCD are the following:

- VCC -> 5v (to the breadboard)

-GND -> GND (to the breadboard)

-SCL -> pin 21 (SCL)

-SDA -> pin 20 (SDA)

* + **Buzzer (#1) :**The Buzzer has the following connections :  
     - ( + ) -> pin 22

- ( - ) -> GND

**- RFID (#1) :** The RFID connections are the following :

- RST -> pin 5

- SDA -> pin 53

- MOSI -> pin 51

- MISO -> pin 50

- SCK -> pin 52

- 3.3v -> pin 3.3v

- GND -> GND

-**Breadboard (#1) :** connect the 5v from arduino mega to the (+) in breadboard and the GND from arduino Mega to the (-) in the breadboard to increase the number of GND and VCC pins.

**- LEDs (#6) :** Each Led has needs two wires connections. Every long leg of every LED will be connected to a pin relative to it of the arduino and every short leg will be connected to a resistor placed on the breadboard. (The resistor will have a side connected the short leg of the Led and the other side to the GND of the breadboard)

Led pins:

-Pin 27 -pin 31

-Pin 28 -pin 32

-Pin 30 -pin 33

**CODE + Explanation:**#include <Wire.h>

Definition of all the needed libraries respectively for the wires , LCD, RFID, Servos and finally the keypad

#include <LiquidCrystal\_I2C.h>

#include <SPI.h>

#include <MFRC522.h>

#include <Servo.h>

#include <Keypad.h>

Define the pins for the Buzzer, and all the Leds in addition to the RFID pins

#define buzzer 22

#define led1 30

#define led2 31

#define led3 32

#define led4 33

#define greenLed 27

#define redLed 28

// RFID

#define SS\_PIN 53

#define RST\_PIN 5

MFRC522 mfrc522(SS\_PIN, RST\_PIN);

// LCD

LiquidCrystal\_I2C lcd(0x27, 16, 2);  // Adjust address as necessary

Define the Liquid Crystal I2C and the pins and declare four instances of the servos

// Servos

Servo servo1;

Servo servo2;

Servo servo3;

Servo servo4;

Servo lockServo;

const byte ROWS = 4; // Four rows

const byte COLS = 4; // Four columns

Specify the number of Columns and Rows for the Keypad , then define the KeyMap, after that define the pins relative to each row and column. Finally create the keypad object.

// Define the keymap

char keys[ROWS][COLS] = {

  {'1', '2', '3', 'A'},

  {'4', '5', '6', 'B'},

  {'7', '8', '9', 'C'},

  {'\*', '0', '#', 'D'}

};

//Connect the keypad ROW0, ROW1, ROW2, ROW3 to these Arduino pins

byte rowPins[ROWS] = {2, 3, 4, A1};

//Connect the keypad COL0, COL1, COL2, COL3 to these Arduino pins

byte colPins[COLS] = {6, 7, 8, 9};

// Create the Keypad object

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// Variables

int cardBalance = 1000; // Initial balance for demonstration purposes

int item1Price = 100;

Initialize the cardBalance, the price of each item, and the new Price.

Put an initial password and define the (password\_entered) to be compared to the initial password for password verification and testing.

int item2Price = 125;

int item3Price = 75;

int item4Price = 150;

int newPrice = 0;

String password = "123456A";

String password\_entered;

String symbol\_entered;

void setup() {

  // Initialize serial communication for debugging

  Serial.begin(9600);

Initialize the serial communication, in addition to the RFID setup

  Serial.println("Initializing...");

  // RFID setup

  SPI.begin();

  mfrc522.PCD\_Init(SS\_PIN, RST\_PIN);

  Serial.println("RFID initialized");

  // Servo setup

Indicate the pin relative to each servo motor by the .attach(pin) and initialize the LCD too

  servo1.attach(A0);

  servo2.attach(11);

  servo3.attach(12);

  servo4.attach(13);

  lockServo.attach(10);

  Serial.println("Servos initialized");

  // LCD setup

  lcd.init();

  lcd.backlight();

  Serial.println("LCD initialized");

Specify the output devices using the already defined pins

  pinMode(buzzer, OUTPUT);

  pinMode(led1, OUTPUT);

  pinMode(led2, OUTPUT);

  pinMode(led3, OUTPUT);

  pinMode(led4, OUTPUT);

  pinMode(greenLed, OUTPUT);

  pinMode(redLed, OUTPUT);

The inner LEDs will always be turned on so we set them to High to keep them lit.  
  
The greenLed and RedLed used to specify the state of the entered password if correct or not are set to Low and lit when needed.

  // Turn on the always-on LEDs

  digitalWrite(led1, HIGH);

  digitalWrite(led2, HIGH);

  digitalWrite(led3, HIGH);

  digitalWrite(led4, HIGH);

  digitalWrite(greenLed, LOW);

  digitalWrite(redLed, LOW);

  Serial.println("Setup completed");

  // Initial display

  resetDisplay();

}  
void loop() {

First we have to get the pressed key, then in the If condition we are stating that if a key is pressed and it is a “ \* ” then call the function startDispensing(); else if the pressed key is “ # “ then call the function refillProcess();

  char key = keypad.getKey();

  if (key != NO\_KEY) {

    if (key == '\*') {

      startDispensing();

    } else if (key == '#') {

      refillProcess();

    }

  }

}

void startDispensing() {

This function is called to make the customer buy the product , first it displays on the screen “Scan your card” and “D-> Exit”

  lcd.clear();

  lcd.print("Scan your card");

  lcd.setCursor(0, 1);

  lcd.print("D-> Exit");

  while (true) {

    char key = keypad.getKey();

Then we see if the pressed key is “D” then the function resetDisplay();is called else if we call the function checkRFID() and see if the card was read successfully and prints it .

    if (key == 'D') {

      resetDisplay();

      return;

    }

    if (checkRFID()) {

      break;

    }

  }

  lcd.clear();

  lcd.print("Card read");

  lcd.setCursor(0, 1);

  lcd.print("Successfully");

  delay(3000);

  // Display initial balance

After the card is read successfully its balance is displayed on the screen to see if it contains a balance and is sufficient for usage.  
Then if its usage is sufficient the customer will choose between the 4 items the one he wants .So the key he presses is read to see which item is chosen and accordingly its detached servo will rotate calling the processItemDispense(servo#, pin, item#Price);function that will later be explained  
The servo in the function will be placed according to what item is chosen.

  lcd.clear();

  lcd.print("Balance:");

  lcd.print(cardBalance);

  lcd.print(" lbp ");

  delay(3000);

  while (true) {

    lcd.clear();

    lcd.print("Select item:");

    lcd.setCursor(0, 1);

    lcd.print("1 2 3 4  D:Exit");

    char key = keypad.getKey();

    if (key != NO\_KEY) {

      switch (key) {

        case '1':

          processItemDispense(servo1, A0, item1Price);

          return; // Return to main loop to reset the display

        case '2':

          processItemDispense(servo2, 11, item2Price);

          return; // Return to main loop to reset the display

        case '3':

          processItemDispense(servo3, 12, item3Price);

          return; // Return to main loop to reset the display

        case '4':

          processItemDispense(servo4, 13, item4Price);

          return; // Return to main loop to reset the display

        case 'D':

          resetDisplay();

          return;

      }

    }

  }

}  
void processItemDispense(Servo &servo, int pin, int itemPrice) {

This function is called to specify the price of the item, by displaying everything on the LCD. Then we call the function controlServo(servo, pin);and the servo parameter in it is taken from the &servo that specifies which servo should rotate.

  if (cardBalance >= itemPrice) {

    lcd.clear();

    lcd.print("price: ");

    lcd.print(itemPrice);

    lcd.print(" lbp ");

    cardBalance -= itemPrice;

    lcd.setCursor(0, 1);

    lcd.print("Balance: ");

    lcd.print(cardBalance);

    lcd.print(" lbp ");

    delay(3000);

    controlServo(servo, pin); // Use helper function to control the servo

    lcd.clear();

After that the LCd displays “Receive your” and the first line and “Item” on the second line with a delay of 5 seconds to enable the user to take his treat . Finally “Sa7ten ☺” is displayed.

    lcd.setCursor(0, 0);

    lcd.print("Receive your");

    lcd.setCursor(1, 1);

    lcd.print("item.");

    delay(5000);

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print("SA7TEN :)");

    delay(3000);

“Insufficient balance” is displayed on the screen if the balance in the card is 0 lbp.  
Then resetDisplay();function is called to return to the initial state

  } else {

    lcd.clear();

    lcd.print("Insufficient");

    lcd.setCursor(0, 1);

    lcd.print("balance");

    delay(3000);

  }

  resetDisplay(); // Always reset display at the end

}

This function is used to rotate the specified servo using the &servo and its corresponding pin. First we should make sure the servo is attached then rotate it clock wise and push the item forward to fall.

void controlServo(Servo &servo, int pin) {

  if (!servo.attached()) {

    servo.attach(pin);

  }  
// Rotate servo to 180 degrees

  servo.write(-180);  delay(2000);

  servo.detach();

}

This function Checks if a new RFID card is within the reader's range. If no new card is present, it prints "No new card present" to the serial monitor, waits for 50 milliseconds, and returns false.

bool checkRFID() {

  if (!mfrc522.PICC\_IsNewCardPresent()) {

    Serial.println("No new card present");

    delay(50);

    return false;

  }

  if (!mfrc522.PICC\_ReadCardSerial()) {

    Serial.println("Error reading card serial");

    delay(50);

    return false;

  }

Attempts to read the UID of the detected card.

If it fails to read the card's serial number, it prints "Error reading card serial" to the serial monitor, waits for 50 milliseconds, and returns false.

  String cardUID = "";

  for (byte i = 0; i < mfrc522.uid.size; i++) {

    cardUID.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : ""));

    cardUID.concat(String(mfrc522.uid.uidByte[i], HEX));

  }

  Serial.print("Card UID: ");

  Serial.println(cardUID);

Initializes an empty string cardUID. Iterates through each byte of the card's UID (mfrc522.uid.uidByte[i]).

If a byte is less than 0x10, it concatenates a leading zero (0) to maintain a two-digit hexadecimal format. Converts each byte to a hexadecimal string and concatenates it to cardUID. Prints the constructed cardUID to the serial monitor.

  mfrc522.PICC\_HaltA();

  mfrc522.PCD\_StopCrypto1();

  return true;

}

 mfrc522.PICC\_HaltA(): Halts communication with the RFID card.

 mfrc522.PCD\_StopCrypto1(): Stops encryption on the PCD (Proximity Coupling Device).

void resetDisplay() {

  lcd.clear();

This function returns to the initial state where two options are presented “start” or “refill / Price”

  lcd.setCursor(0, 0);

  lcd.print("\*-Start");

  lcd.setCursor(0, 1);

  lcd.print("#-Refill /Price");

}

void refillProcess() {

  enum RefillState { SHOW\_OPTIONS, ENTER\_PASSWORD, CHANGE\_PASSWORD, RFID\_SCAN, RFID\_MENU, SCAN\_RFID,CHANGE\_PRICES };

  RefillState currentState = SHOW\_OPTIONS;

  RefillState previousState = SHOW\_OPTIONS;

This function is called when the user presses “#” from the initial state.   
Then the cases are divided according to what the user chooses.

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("By Pass:A");

  lcd.setCursor(0, 1);

  lcd.print("By Tag:C");

  lcd.setCursor(10, 1);

  lcd.print("D:Exit");

If the user presses “A” then the case is “ENTER\_PASSWORD” OR “CHANGE\_PASSWORD” else if he presses “C” then the case is “RFID\_MENU”

  while (true) {

    char key = keypad.getKey();

    if (key != NO\_KEY) {

      switch (currentState) {

        case SHOW\_OPTIONS:

          if (key == 'D') {

            resetDisplay();

If the user presses “D” then resetDisplay();is called else if “A” is pressed we have two cases either changing the password or entering the password.

And change the previous state to the current state and the current state to this state because the events are sequential.

            return;

          } else if (key == 'A') {

            lcd.clear();

            lcd.setCursor(0, 0);

            lcd.print("Enter pass->#");

            lcd.setCursor(0, 1);

            lcd.print("Change->\*");

            lcd.setCursor(10, 1);

            lcd.print("B:Back");

            previousState = SHOW\_OPTIONS;

            currentState = ENTER\_PASSWORD;

          }

else if (key == 'C') {

            lcd.clear();

If the user pressed “C” then he can use the TAG not enter the password manually.  
he has two options either “Refill” or “change the prices”.  
Also update the previous state and the initial state

            lcd.setCursor(0, 0);

            lcd.print("Press each key");

            lcd.setCursor(0, 1);

            lcd.print("3 times");

            delay(3000);

            lcd.clear();

            lcd.setCursor(0, 0);

            lcd.print("1-Refill");

            lcd.setCursor(0, 1);

            lcd.print("2-Change Prices");

            previousState = SHOW\_OPTIONS;

            currentState = RFID\_MENU;

          }

          break;

If he pressed “#” he en*t*ers to the ENTER\_PASSWORD case then chooses if he wants to enter the password manually by pressing “A” or tag by pressing “C”.  
If he presses “A” then we see the 7 digits he enters and compare them to the initial password.  
“\*” to delete if the user mistaken.  
  
If he presses “B” then he returns to the previous state.

    case ENTER\_PASSWORD:

    if (key == 'B') {

        lcd.clear();

        lcd.setCursor(0, 0);

        lcd.print("By Pass:A");

        lcd.setCursor(0, 1);

        lcd.print("By Tag:C");

        lcd.setCursor(10, 1);

        lcd.print("D:Exit");

        currentState = SHOW\_OPTIONS;

    } else if (key == '#') {

        lcd.clear();

        lcd.setCursor(0, 0);

        lcd.print("7 digit: ");

        lcd.setCursor(10, 0);

        lcd.print(" \*:Del ");

When the number of entered digits is seven or below the numbers are displayed on the LCD .  
if “\*” is pressed remove the last entered key

        password\_entered = "";

        while (password\_entered.length() < 7) {

    char key = keypad.getKey();

    if (key != NO\_KEY) {

        if (key == '\*' && password\_entered.length() > 0) {

            password\_entered.remove(password\_entered.length() - 1);

            lcd.setCursor(password\_entered.length(), 1);

            lcd.print(' ');

            lcd.setCursor(password\_entered.length(), 1);

        } else if (key != '\*') {

            password\_entered += key;

            lcd.setCursor(password\_entered.length() - 1, 1);

            lcd.print(key);

        }

Then compare the entered password to the initially specified password, if they are equal turn on the green led , else turn on the red led and in both cases let the buzzer start buzzing.

        Serial.print("Key pressed: ");

        Serial.println(key);

    }

}

if (password\_entered == password) {

 digitalWrite(greenLed, HIGH); // Turn on the green LED for success

  for (int i = 0; i < 15; i++) { // Flash the buzzer for 3 seconds

    tone(buzzer, 2000);

  delay(100);

  noTone(buzzer);

  delay(100);

  } digitalWrite(greenLed, LOW); // Turn off the green LED

  lcd.setCursor(0, 0);

  lcd.print("CORRECT PASSWORD");

Write “correct password” if the password was correct and door opened because we will open the door too by rotating the lockservo 180 degrees.  
  
Then present two options that are “to refill” and “ lock again” .

  lcd.setCursor(0, 1);

  lcd.print("DOOR OPENED");

  lockServo.write(180);

  delay(3000);

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("1->4 To refill");

  lcd.setCursor(0,1);

  lcd.print("###->lock again");

while (true) {

Acoording to what is pressed by the keypad the function controlServoReverse(servo1, A0);will be called (previously explained

        char key = keypad.getKey();

        if (key != NO\_KEY) {

            if (key == '1') {

                controlServoReverse(servo1, A0);

            } else if (key == '2') {

                controlServoReverse(servo2, 11);

            } else if (key == '3') {

                controlServoReverse(servo3, 12);

            } else if (key == '4') {

                controlServoReverse(servo4, 13);

            } else if (key == '#') {

                // Check if the user wants to lock the door again

                symbol\_entered = "";

                while (symbol\_entered.length() < 2) {

                    char key = keypad.getKey();

                    if (key != NO\_KEY) {

                        symbol\_entered += key;

                    }

                }

                if (symbol\_entered == "##") {

                    lockServo.write(90);

If the vendor or worker entered “###” the door will be locked again.  
and then we will return to the initial state using the resetDisplay();

                    resetDisplay();

                    return;

                }

            }

        }

    }

}

else {

            digitalWrite(redLed, HIGH); // Turn on the red LED for

failure

          for (int i = 0; i < 6; i++) { // Flash the buzzer and red

LED simultaneously for 3 seconds

            tone(buzzer, 2000);

            delay(250);

            noTone(buzzer);

            delay(250);

            }

          digitalWrite(redLed, LOW); // Turn off the red LED

            lcd.setCursor(0, 0);

            lcd.print("WRONG PASSWORD!");

            lcd.setCursor(0, 1);

If the entered password was incorrect write “wrong password” “please try again” and return to the state before it.

            lcd.print("PLEASE TRY AGAIN");

            delay(3000);

            lcd.clear();

            lcd.setCursor(0, 0);

            lcd.print("Enter pass->#");

            lcd.setCursor(0, 1);

            lcd.print("Change->\*");

            lcd.setCursor(10, 1);

            lcd.print("B:Back");

If the user entered “\*” then he wants to change the password .  
if in the another state which is the new one he presses “\*” then it is used to erase a mistake.

        }

    } else if (key == '\*') {

        lcd.clear();

        lcd.setCursor(0, 0);

        lcd.print("old pass:");

        lcd.setCursor(0, 1);

        lcd.print(" \*->Del ");

        lcd.setCursor(9, 1);

        lcd.print("B->Back");

        currentState = CHANGE\_PASSWORD;

    }

    break;

If he pressed back “B” then he will return to the previous state

case CHANGE\_PASSWORD:

    if (key == 'B') {

        lcd.clear();

        lcd.setCursor(0, 0);

        lcd.print("Enter pass->#");

        lcd.setCursor(0, 1);

        lcd.print("Change->\*");

        lcd.setCursor(10, 1);

        lcd.print("B:Back");

        currentState = ENTER\_PASSWORD;

    } else if (key != NO\_KEY) {

        password\_entered = "";

        int pos = 9;  // Initial position after "old pass: "

        while (password\_entered.length() < 7) {

            char key = keypad.getKey();

            if (key != NO\_KEY) {

               if (key == '\*' && password\_entered.length() > 0) {

                password\_entered.remove(password\_entered.length()-1);

                    lcd.setCursor(pos - 1, 0);

                    lcd.print(' ');

This part of the code is responsible for changing the old password to a new one.  
  
Then verifying it by comparing the new password to the password entered and saved in a variable .

                    pos--;

                    lcd.setCursor(pos, 0);

                } else if (key != '\*') {

                    password\_entered += key;

                    lcd.setCursor(pos++, 0);

                    lcd.print(key);

                }

            }

        }

        if (password\_entered == password) {

            lcd.clear();

If the pressed key is “ \*” then erase else add it to the entered password

            lcd.setCursor(0, 0);

            lcd.print("Enter new pass:");

            lcd.setCursor(9, 1);

            lcd.print(" \*:Del ");

            password\_entered = "";

            while (password\_entered.length() < 7) {

    char key = keypad.getKey();

    if (key != NO\_KEY) {

        if (key == '\*' && password\_entered.length() > 0) {

            password\_entered.remove(password\_entered.length() - 1);

            lcd.setCursor(password\_entered.length(), 1);

            lcd.print(' ');

            lcd.setCursor(password\_entered.length(), 1);

        } else if (key != '\*') {

            password\_entered += key;

            lcd.setCursor(password\_entered.length() - 1, 1);

            lcd.print(key);

        }

        Serial.print("Key pressed: ");

This step is the verification step that tells the user to enter the new password again to verify it and change the password accordingly.

        Serial.println(key);

    }

}

            lcd.clear();

            lcd.setCursor(0, 0);

            lcd.print("Re-enter new pass:");

            lcd.setCursor(9, 1);

            lcd.print(" \*:Del ");

            String new\_password\_confirm = "";

while (new\_password\_confirm.length() < 7) {

    char key = keypad.getKey();

    if (key != NO\_KEY) {

        if (key == '\*' && new\_password\_confirm.length() > 0) {

            // Remove the last character from the confirmation

password

       new\_password\_confirm.remove(new\_password\_confirm.length() - 1);

            // Move cursor back and print space to 'erase' character

from LCD

            lcd.setCursor(new\_password\_confirm.length(), 1);

            lcd.print(' ');

            // Move cursor back again to current position

            lcd.setCursor(new\_password\_confirm.length(), 1);

        } else if (key != '\*') {

            // Add the key to the confirmation password

            new\_password\_confirm += key;

            // Print the key to the LCD

            lcd.setCursor(new\_password\_confirm.length() - 1, 1);

            lcd.print(key);

        }

    }

}

            if (password\_entered == new\_password\_confirm) {

                password = new\_password\_confirm;

                lcd.clear();

                lcd.setCursor(0, 0);

                lcd.print("Password changed");

                delay(3000);

If the new password was equal to the password before , then the password will be updated successfully.  
  
change the currentState to “ENTER\_PASSWORD”

                lcd.clear();

                lcd.setCursor(0, 0);

                lcd.print("Enter pass->#");

                lcd.setCursor(0, 1);

                lcd.print("Change->\*");

                lcd.setCursor(10, 1);

                lcd.print("B:Back");

                currentState = ENTER\_PASSWORD;

            } else {

                lcd.clear();

                lcd.setCursor(0, 0);

                lcd.print("Passwords don't");

If in the verification step the re entered password was not equal to the new entered password then it will display “The passwords don’t match”

                lcd.setCursor(0, 1);

                lcd.print("match!!");

                delay(3000);

                lcd.clear();

                lcd.setCursor(0, 0);

                lcd.print("Enter pass->#");

                lcd.setCursor(0, 1);

                lcd.print("Change->\*");

                lcd.setCursor(10, 1);

                lcd.print("B:Back");

                currentState = ENTER\_PASSWORD;

            }

        } else {

            lcd.clear();

            lcd.setCursor(0, 0);

            lcd.print("Incorrect pass");

            delay(3000);

Here we are not changing the password but entering the usual old password and seeing if it is correct or not.  
  
currentState = ENTER\_PASSWORD

            lcd.clear();

            lcd.setCursor(0, 0);

            lcd.print("Enter pass->#");

            lcd.setCursor(0, 1);

            lcd.print("Change->\*");

            lcd.setCursor(10, 1);

            lcd.print("B:Back");

            currentState = ENTER\_PASSWORD;

        }

    }

    break;

If the Case was RFID\_MENU then the user could do two things, either the refill by pressing 111 or the “Change price “ by pressing 222  
if he pressed 111 => RFID\_SCAN , if 222 => SCAN\_RFID;

    case RFID\_MENU:

    char key;

    while (true) {

        key = keypad.getKey(); // Get the key press

        if (key != NO\_KEY) {

          Serial.print("Key pressed: ");

            Serial.println(key); // Check if a key was pressed

            if (key == '1') {

                currentState = RFID\_SCAN;

                break; // Exit the loop after state transition

            } else if (key == '2') {

                currentState = SCAN\_RFID;

                break; // Exit the loop after state transition

            }

        }

    }

    break;

This is the case for changing the prices ,first we have to scan the card and make sure it is detected then we will move to a ne state which is CHANGE\_PRICE

    case SCAN\_RFID:

       lcd.clear();

          lcd.setCursor(0, 0);

          lcd.print("Scan Card");

          lcd.setCursor(0, 1);

          lcd.print("B->Back");

           previousState = SCAN\_RFID; // Save the previous state

           currentState = SCAN\_RFID; // Continue scanning

           while (true) {

        char key = keypad.getKey();

        if (key == 'B') {

            // Go back

            refillProcess();

            return;

        }

        if (checkRFID()) {

           lcd.clear();

            lcd.print("Card Detected");

            delay(2000);

            lcd.clear();

            lcd.setCursor(0, 0);

            lcd.print("Press any key");

            lcd.setCursor(0, 1);

            lcd.print("to change price");

          currentState = CHANGE\_PRICES;

          break;

        }

           }

           break;

case CHANGE\_PRICES:

Here the vendor chooses the item after we get the value of the pressed key from the keypad, if the pressed key was “\*” we will call the refillProcess();function and change the previous state to the current state to save the sequence

    while (true) {

        lcd.clear();

        lcd.print("Choose item(1->4)");

        lcd.setCursor(0, 1);

        lcd.print("\*: Back");

        previousState = SCAN\_RFID;

        currentState = CHANGE\_PRICES;

        while (true) {

            char key = keypad.getKey();

            if (key != NO\_KEY) {

                if (key == '\*') {

                    refillProcess();

                    currentState = previousState;

Then we choose the item number that accordingly we will change the price

                    return;

                }

                if (key >= '1' && key <= '4') {

                    int itemNum = key - '0';

                    lcd.clear();

                    lcd.print("Set price:");

                    lcd.setCursor(0, 1);

                    lcd.print("--- #=Done \*:Del");

                    String newPriceStr = ""; // Start with an empty

string for dynamic length

int cursorPos = 0; // Position of the cursor on

LCD

Here we will get the newPrice according to what was entered by the keypad (itemNb)

                    while (true) {

                        char key = keypad.getKey();

                        if (key != NO\_KEY) {

                            if (key == '#') {

                                int newPrice = newPriceStr.toInt();

                                switch (itemNum) {

                                    case 1:

                                        item1Price = newPrice;

                                        break;

                                    case 2:

                                        item2Price = newPrice;

Everything here is a repetition of things we previously explained

                                        break;

                                    case 3:

                                        item3Price = newPrice;

                                        break;

                                    case 4:

                                        item4Price = newPrice;

                                        break;

                                }

                                lcd.clear();

                                lcd.print("Price updated");

                                delay(2000);

                                currentState = SCAN\_RFID;

                                break;  // Exit the inner while loop

after updating price

                            } else if (key == '0' &&

newPriceStr.toInt() == 0 && cursorPos == 0) {

                                // Skip leading zeros if entered

(optional)

                                continue;

                            } else if (key == '\*' && cursorPos > 0) {

                                // Backspace if there are characters

to delete

                                lcd.setCursor(--cursorPos, 1);

                                lcd.print(' ');

newPriceStr.remove(newPriceStr.length() - 1);

// Remove last character

                            } else if (cursorPos < 3) {

                                // Allow up to 3 digits

                                lcd.setCursor(cursorPos++, 1);

                                lcd.print(key);

                                newPriceStr += key;

// Append entered key to string

                            }

                        }

                    }

                    break;

                }

            }

        }

    }

    break;

Everything here is a repetition of things we previously explained

  case RFID\_SCAN:

    lcd.clear();

    lcd.print("Scan refill card");

    lcd.setCursor(0, 1);

    lcd.print("B-> Back");

    previousState = RFID\_SCAN; // Save the previous state

    currentState = RFID\_SCAN; // Continue scanning

    while (true) {

        char key = keypad.getKey();

        if (key == 'B') {

            // Go back

            refillProcess();;

            return;

        }

        if (checkRFID()) {

            // If an RFID card is scanned

            lcd.clear();

            lcd.print("DOOR OPENED");

            lockServo.write(180); // Open the door

            delay(3000);

            lcd.clear();

            lcd.setCursor(0,0);

            lcd.print("1->4 To refill");

            lcd.setCursor(0,1);

            lcd.print("Scan to lock");

            // Loop to allow the user to refill items

            while (true) {

                char key = keypad.getKey();

                if (key != NO\_KEY) {

                    if (key == '1') {

                        controlServoReverse(servo1, A0);

                    } else if (key == '2') {

                        controlServoReverse(servo2, 11);

                    } else if (key == '3') {

                        controlServoReverse(servo3, 12);

                    } else if (key == '4') {

                        controlServoReverse(servo4, 13);

                    }

                }

                // Check if RFID card is scanned again to lock the door

                if (checkRFID()) {

                    lockServo.write(90); // Lock the door

                    lcd.clear();

                    lcd.print("DOOR LOCKED");

                    delay(2000);

                    resetDisplay();

If the rfid tag or card were scanned again then the door will be locked again.  
  
Make the default case the initial state

                    return;

                }

            }

        }

    }

    break;

default:

This function is used to indicate which servo should rotate anti\_clockwise to refill the products, in other words to make the products move backwards

    resetDisplay();

    return;

}

    }

  }

}

void controlServoReverse(Servo &servo, int pin) {

  if (!servo.attached()) {

    servo.attach(pin);

  }

  servo.write(180); // Rotate servo to 0 degrees

  delay(2000);

  servo.detach();

}

* **CH5 : The RESULT :**

As a result you will have a vending machine that has the capability of making customers buys products using the rfid card payment method.

The vendor will be able to unlock his machine through two methods, either by using the tag and the rfid, or by entering a specific password. Concerning the password the vendor will be able to change it after some verification steps too.  
  
After the vendor opened the machine door, now he can fill in the products again, and suppose he changed them he can also change their prices accordingly.  
  
Of course every step will be shown on the Lcd screen clearly.

* **CH6 : Recommendations :**

**User Interface Enhancements**

1. **Touch screen LCD**: Upgrade to a touch screen LCD to make navigation and input more intuitive.
2. **LED Indicators**: Add LED indicators to show machine status, such as "ready," "processing," or "error."

**Security Enhancements**

1. **Two-Factor Authentication**: Implement two-factor authentication for the vendor using a combination of RFID and a password.
2. **Encrypted Communication**: Ensure that all data, especially passwords and transactions, are encrypted to enhance security.

**Payment System**

1. **Multiple Payment Methods**: Integrate additional payment methods such as mobile payments (NFC), QR code scanning, or even traditional coin and bill acceptors.
2. **Transaction Logging**: Implement a system to log transactions for auditing and inventory management purposes.

**Vendor Interface**

1. **Mobile App Integration**: Develop a mobile app to allow the vendor to monitor and manage the vending machine remotely.
2. **Voice Command**: Add voice command capabilities for a hands-free experience.

**Inventory Management**

1. **Stock Sensors**: Install sensors to automatically detect stock levels and notify the vendor when supplies are low.
2. **Automated Restocking**: Create a system that can automatically reorder products when stock levels are low.

**User Experience**

1. **Personalization**: Offer personalized recommendations or promotions based on user purchase history (if tracking purchases).
2. **Feedback Mechanism**: Provide a way for customers to leave feedback or rate their experience.

**Technical Improvements**

1. **Modular Design**: Use a modular design for easier maintenance and upgrades. For example, make it easy to swap out the RFID reader or LCD screen.
2. **Energy Efficiency**: Optimize the system for low power consumption to extend the machine’s operational life on a single power source.

**Documentation and Support**

1. **User Manual**: Create a comprehensive user manual for both customers and vendors.
2. **Support System**: Set up a support system, such as a website or a hotline, for troubleshooting and assistance.

**Additional Features**

1. **Temperature Control**: Add temperature control for products that need refrigeration or heating.
2. **Remote Monitoring**: Implement a remote monitoring system to track machine status, sales, and stock levels in real-time.

Thank you for your time

Hope Everything was clear