**Appendix F. Program Sample**

#include <Wire.h>

#include <Adafruit\_PWMServoDriver.h>

#include <LiquidCrystal\_I2C.h> // Include the LiquidCrystal\_I2C library

#include <HX711.h> // Include the HX711 library for the load cell

#include <Keypad.h> // Include the Keypad library

#include "Adafruit\_Thermal.h" // Include the Adafruit Thermal Printer library

#include <RTClib.h> // Include the RTC library

// Pin Definitions

const int trigPin1 = 9;

const int echoPin1 = 10;

const int trigPin2 = 11; // Second ultrasonic sensor trig pin

const int echoPin2 = 12; // Second ultrasonic sensor echo pin

const int trigPin3 = 4; // Third ultrasonic sensor trig pin

const int echoPin3 = 5; // Third ultrasonic sensor echo pin

const int trigPin4 = 6; // Fourth ultrasonic sensor trig pin

const int echoPin4 = 7; // Fourth ultrasonic sensor echo pin

const int trigPin5 = 31; // Fifth ultrasonic sensor trig pin

const int echoPin5 = 32; // Fifth ultrasonic sensor echo pin

const int proximitySensorPin1 = 29; // First proximity sensor pin

const int proximitySensorPin2 = 30; // Second proximity sensor pin

const int buzzerPin = 33; // Buzzer pin

const int ledPin = 34; // LED pin

// Load Cell Pins

const int LOADCELL1\_DOUT\_PIN = 2; // HX711 data pin for load cell 1 (DT)

const int LOADCELL1\_SCK\_PIN = 3; // HX711 clock pin for load cell 1 (SCK)

const int LOADCELL2\_DOUT\_PIN = 8; // HX711 data pin for load cell 2 (DT)

const int LOADCELL2\_SCK\_PIN = 13; // HX711 clock pin for load cell 2 (SCK)

// Keypad Setup

const byte ROWS = 4; // 4 rows

const byte COLS = 3; // 3 columns

char keys[ROWS][COLS] = {

{'1', '2', '3'}, // Row 1

{'4', '5', '6'}, // Row 2

{'7', '8', '9'}, // Row 3

{'\*', '0', '#'} // Row 4

};

byte rowPins[ROWS] = {22, 23, 24, 25}; // Connect to the row pinouts of the keypad

byte colPins[COLS] = {26, 27, 28}; // Connect to the column pinouts of the keypad

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

// Create PWM servo driver instance

Adafruit\_PWMServoDriver pwm = Adafruit\_PWMServoDriver();

// Initialize the LCD with I2C address 0x27, 16 columns, and 2 rows

LiquidCrystal\_I2C lcd(0x27, 16, 2);

// Initialize the HX711 for the load cells

HX711 loadCell1;

HX711 loadCell2;

// Thermal Printer Setup

#include "SoftwareSerial.h"

#define TX\_PIN 18 // Define the TX pin for the thermal printer

#define RX\_PIN 19 // Define the RX pin for the thermal printer

SoftwareSerial printerSerial(RX\_PIN, TX\_PIN); // Create a SoftwareSerial object

Adafruit\_Thermal printer(&printerSerial); // Create a thermal printer object

// RTC Setup

RTC\_DS3231 rtc; // Create an RTC object

// Constants

const int servoMin = 150; // Min pulse length (servo closed)

const int servoMax = 600; // Max pulse length (servo open)

const int distanceThreshold = 5; // Threshold distance (cm)

const float MIN\_WEIGHT\_THRESHOLD\_1 = 5.0; // Minimum weight threshold for load cell 1 in grams

const float MAX\_WEIGHT\_THRESHOLD\_1 = 35.0; // Maximum weight threshold for load cell 1 in grams

const float MIN\_WEIGHT\_THRESHOLD\_2 = 2.0; // Minimum weight threshold for load cell 2 in grams

const float MAX\_WEIGHT\_THRESHOLD\_2 = 10.0; // Maximum weight threshold for load cell 2 in grams

// Variables

int detectionCount = 0; // Counter for Ultrasonic 2 detections

bool allowDetection = true; // Flag to allow or block detections

unsigned long lastDetectionTime = 0; // Timestamp of the last detection

const unsigned long detectionDelay = 3000; // 3-second delay between detections

float currentWeight = 0.0; // Variable to store the current weight for load cell 1

float currentWeight2 = 0.0; // Variable to store the current weight for load cell 2

// Student ID Variables

String studentID = ""; // Store the entered Student ID

bool isEnteringID = false; // Flag to indicate if the user is entering an ID

bool isConfirmingID = false; // Flag to indicate if the user is confirming the ID

bool isInsertingBottles = false; // Flag to indicate if the user is inserting bottles

bool isPrintingTicket = false; // Flag to indicate if the user is printing the ticket

// Points Variables

int pointsUltrasonic2 = 0; // Points from Ultrasonic 2 (1 point = 3 minutes)

int pointsUltrasonic4 = 0; // Points from Ultrasonic 4 (1 point = 1 minute)

// Timing Variables

unsigned long lastSensorScanTime = 0; // Timestamp of the last sensor scan

const unsigned long sensorScanInterval = 2000; // 2-second interval for sensor scans

// Keypad Timing Variables

unsigned long lastKeypadCheckTime = 0; // Track the last time the keypad was checked

const unsigned long keypadCheckInterval = 100; // Check the keypad every 100ms

// Variable to store the time when "Not Accepted" is displayed

unsigned long notAllowedDisplayTime = 0;

// Flag to track if a student ID has been entered

bool studentIDEntered = false;

bool isEnterValidIDScreen = false;

// Flag to track if Ultrasonic 5 is detecting an object

bool ultrasonic5Detected = false;

// Function to map angle to pulse length

int angleToPulse(int angle) {

return map(angle, 0, 45, servoMin, servoMax);

}

void setup() {

pinMode(trigPin1, OUTPUT);

pinMode(echoPin1, INPUT);

pinMode(trigPin2, OUTPUT); // Configure second ultrasonic sensor trig pin

pinMode(echoPin2, INPUT); // Configure second ultrasonic sensor echo pin

pinMode(trigPin3, OUTPUT); // Configure third ultrasonic sensor trig pin

pinMode(echoPin3, INPUT); // Configure third ultrasonic sensor echo pin

pinMode(trigPin4, OUTPUT); // Configure fourth ultrasonic sensor trig pin

pinMode(echoPin4, INPUT); // Configure fourth ultrasonic sensor echo pin

pinMode(trigPin5, OUTPUT); // Configure fifth ultrasonic sensor trig pin

pinMode(echoPin5, INPUT); // Configure fifth ultrasonic sensor echo pin

pinMode(proximitySensorPin1, INPUT); // Configure first proximity sensor pin as input

pinMode(proximitySensorPin2, INPUT); // Configure second proximity sensor pin as input

pinMode(buzzerPin, OUTPUT); // Configure buzzer pin as output

pinMode(ledPin, OUTPUT); // Configure LED pin as output

pwm.begin();

pwm.setPWMFreq(60); // Set PWM frequency to 60 Hz

// Initialize the LCD

lcd.begin(16, 2); // Initialize the LCD with 16 columns and 2 rows

lcd.backlight(); // Turn on the backlight

lcd.print(" WELCOME "); // Display welcome message

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.print("Click # to start");

// Initialize the load cells

loadCell1.begin(LOADCELL1\_DOUT\_PIN, LOADCELL1\_SCK\_PIN);

loadCell1.set\_scale(2280.f); // Calibration factor for your load cell (adjust as needed)

loadCell1.tare(); // Reset the scale to zero

loadCell2.begin(LOADCELL2\_DOUT\_PIN, LOADCELL2\_SCK\_PIN);

loadCell2.set\_scale(2280.f); // Calibration factor for your load cell (adjust as needed)

loadCell2.tare(); // Reset the scale to zero

// Initialize the thermal printer

printerSerial.begin(9600); // Start the printer serial communication

printer.begin(); // Initialize the thermal printer

// Initialize the RTC

if (!rtc.begin()) {

Serial.println("Couldn't find RTC");

while (1);

}

// Set the RTC to the date & time this sketch was compiled (if not already set)

if (rtc.lostPower()) {

Serial.println("RTC lost power, setting the time!");

rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

}

Serial.begin(9600);

}

void loop() {

int proximitySensorState1 = digitalRead(proximitySensorPin1); // Read the state of the first proximity sensor

int proximitySensorState2 = digitalRead(proximitySensorPin2); // Read the state of the second proximity sensor

static bool welcomeScreenActive = true; // Flag to track if the welcome screen is active

static bool askToInsertBottles = false; // Flag to track if the system is asking to insert bottles

static bool isCountingActive = false; // Flag to track if the counting function is active

static bool isLEDConfirmationActive = false; // Flag to track if the LED confirmation screen is active

// Non-blocking keypad logic

unsigned long currentTime = millis();

if (currentTime - lastKeypadCheckTime >= keypadCheckInterval) {

lastKeypadCheckTime = currentTime; // Update the last keypad check time

char key = keypad.getKey();

if (key) {

if (welcomeScreenActive) {

if (key == '#') { // Transition from welcome screen to LED confirmation

welcomeScreenActive = false; // Disable the welcome screen

isLEDConfirmationActive = true; // Enable LED confirmation screen

lcd.clear();

lcd.print("Is the green LED ");

lcd.setCursor(0, 1);

lcd.print("blinking? \*:Yes #:No");

} else {

// Ignore any other key presses during the welcome screen

Serial.println("Only '#' is allowed during the welcome screen”.);

}

} else if (isLEDConfirmationActive) {

if (key == '#') { // User confirms LED is blinking

isLEDConfirmationActive = false; // Disable LED confirmation screen

lcd.clear();

lcd.print("Enter Student ID: "); // Prompt for Student ID

isEnteringID = true; // Enable Student ID entry

} else if (key == '\*') { // User denies LED is blinking

isLEDConfirmationActive = false; // Disable LED confirmation screen

welcomeScreenActive = true; // Return to welcome screen

lcd.clear();

lcd.print(" WELCOME ");

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.print("Click # to start");

} else {

// Ignore any other key presses during the LED confirmation screen

Serial.println("Only '#' or '\*' are allowed during the LED confirmation screen”.);

}

} else if (isEnteringID && key == '#') { // Confirm Student ID

if (studentID.length() == 10) { // Only allow confirmation if Student ID is not empty

isEnteringID = false;

studentIDEntered = true; // Set the flag to indicate that a student ID has been entered

lcd.clear();

lcd.print("Student ID:");

lcd.setCursor(0, 1);

lcd.print(studentID); // Display the entered Student ID

Serial.print("Student ID Entered: ");

Serial.println(studentID);

// Ask if the user wants to insert bottles

askToInsertBottles = true;

lcd.clear();

lcd.print("Is ID Correct?");

lcd.setCursor(0, 1);

lcd.print("\*: No #: Yes ");

} else {

lcd.clear();

lcd.print("INVALID ID ");

lcd.setCursor(0, 1);

lcd.print("Click \* to Retry");

isEnterValidIDScreen = true;// Prompt for Student ID again

Serial.println("Student ID is empty. Please enter a valid ID”.);

}

} else if (isEnterValidIDScreen) { // Only allow '\*' to be pressed in the "Enter Valid ID" screen

if (key == '\*') { // User presses '\*' to re-enter the ID

isEnterValidIDScreen = false; // Exit the "Enter Valid ID" screen

lcd.clear();

lcd.print("Enter Student ID: "); // Prompt for Student ID again

studentID = ""; // Reset the Student ID

isEnteringID = true; // Enable Student ID entry

} else {

// Ignore any other key presses during the "Enter Valid ID" screen

Serial.println("Only '\*' is allowed during the 'Enter Valid ID' screen”.);

}

} else if (askToInsertBottles) { // Only allow '#' or '\*' to be pressed

if (key == '#') { // User confirms to insert bottles

askToInsertBottles = false;

isCountingActive = true; // Proceed to counting function

lcd.clear();

lcd.print(" PLEASE INSERT YOUR ");

lcd.print("PLASTIC BOTTLES..”.);

} else if (key == '\*') { // User declines to insert bottles

askToInsertBottles = false;

welcomeScreenActive = true; // Return to welcome screen

lcd.clear();

lcd.print(" WELCOME ");

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.print("Click # to start");

studentID = ""; // Reset the Student ID

studentIDEntered = false; // Reset the flag

} else {

// Ignore any other key presses during the "IS THE ID CORRECT?" screen

Serial.println("Only '#' or '\*' are allowed during the ID confirmation screen”.);

}

} else if (isCountingActive && key == '#') { // Print ticket during counting

if (pointsUltrasonic2 > 0 || pointsUltrasonic4 > 0) { // Check if points are greater than 0

isCountingActive = false; // Stop counting

welcomeScreenActive = true; // Return to welcome screen

// Display "PLEASE WAIT..”. message on the LCD

lcd.clear();

lcd.print("PLEASE WAIT... ");

lcd.print("PRINTING”.);

// Print the ticket

printReceipt(studentID, pointsUltrasonic2, pointsUltrasonic4);

// Reset points and Student ID

pointsUltrasonic2 = 0;

pointsUltrasonic4 = 0;

studentID = "";

studentIDEntered = false; // Reset the flag

// Display welcome message

lcd.clear();

lcd.print(" WELCOME ");

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.print("Click # to start");

} else {

lcd.clear();

lcd.print("No points to print”.);

delay(2000); // Display message for 2 seconds

lcd.clear();

lcd.print(" PLEASE INSERT YOUR ");

lcd.print("PLASTIC BOTTLES..”.);

}

} else if (key == '\*') { // Clear Student ID

studentID = "";

studentIDEntered = false; // Reset the flag

lcd.clear();

lcd.print("Enter Student ID: ");

Serial.println("Student ID Cleared”.);

} else if (isEnteringID) { // Append to Student ID

studentID += key;

lcd.setCursor(studentID.length() - 1, 1); // Move cursor to the next position

lcd.print(key); // Display the pressed key

Serial.print("Key pressed: ");

Serial.println(key);

} else if (isCountingActive) { // Only allow '#' and '\*' during counting

if (key == '#' || key == '\*') {

// Handle '#' or '\*' key press during counting

if (key == '#') { // Print ticket

if (pointsUltrasonic2 > 0 || pointsUltrasonic4 > 0) { // Check if points are greater than 0

isCountingActive = false; // Stop counting

welcomeScreenActive = true; // Return to welcome screen

// Display "PLEASE WAIT..”. message on the LCD

lcd.clear();

lcd.print("PLEASE WAIT... ");

lcd.print("PRINTING”.);

// Print the ticket

printReceipt(studentID, pointsUltrasonic2, pointsUltrasonic4);

// Reset points and Student ID

pointsUltrasonic2 = 0;

pointsUltrasonic4 = 0;

studentID = "";

studentIDEntered = false; // Reset the flag

// Display welcome message

lcd.clear();

lcd.print(" WELCOME ");

lcd.setCursor(0, 1);

lcd.print(" ");

lcd.print("Click # to start");

} else {

lcd.clear();

lcd.print("No points to print”.);

delay(2000); // Display message for 2 seconds

lcd.clear();

lcd.print(" PLEASE INSERT YOUR ");

lcd.print("PLASTIC BOTTLES..”.);

}

} else if (key == '\*') { // Clear Student ID

studentID = "";

studentIDEntered = false; // Reset the flag

lcd.clear();

lcd.print("Enter Student ID: ");

Serial.println("Student ID Cleared”.);

}

} else {

// Ignore any other key presses during counting

Serial.println("Only '#' or '\*' are allowed during counting”.);

}

} else { // Start entering Student ID

isEnteringID = true;

lcd.clear();

lcd.print("Enter Student ID: ");

lcd.print(" ");

lcd.print("Max: 10 char");

studentID += key;

lcd.setCursor(0, 1);

lcd.print(key); // Display the first key

Serial.print("Key pressed: ");

Serial.println(key);

}

}

}

// Non-blocking sensor scan (counting function)

if (isCountingActive && currentTime - lastSensorScanTime >= sensorScanInterval) {

lastSensorScanTime = currentTime; // Update the last scan time

// Measure distance for fifth ultrasonic sensor

long duration5 = getUltrasonicDistance(trigPin5, echoPin5);

int distance5 = duration5 \* 0.034 / 2;

// Check if Ultrasonic 5 detects an object

if (distance5 <= 10 && distance5 > 0) {

ultrasonic5Detected = true; // Set the flag to true

digitalWrite(buzzerPin, HIGH); // Turn on the buzzer

digitalWrite(ledPin, HIGH); // Turn on the LED

lcd.clear(); // Clear the LCD

lcd.print("DON'T DEPOSIT BOTTLE");

lcd.print("BIN IS FULL"); // Display "The bin is full" message

delay(2000); // Keep buzzer, LED, and message on for 2 seconds

digitalWrite(buzzerPin, LOW); // Turn off the buzzer

digitalWrite(ledPin, LOW); // Turn off the LED

delay(3000); // Wait for 3 seconds before next cycle (total 5 seconds)

} else {

ultrasonic5Detected = false; // Set the flag to false

}

// Measure distance for first ultrasonic sensor (only if Ultrasonic 5 is not detecting an object)

if (!ultrasonic5Detected) {

long duration1 = getUltrasonicDistance(trigPin1, echoPin1);

int distance1 = duration1 \* 0.034 / 2;

Serial.print("Distance 1: ");

Serial.println(distance1);

// Measure distance for third ultrasonic sensor (only if Ultrasonic 5 is not detecting an object)

long duration3 = getUltrasonicDistance(trigPin3, echoPin3);

int distance3 = duration3 \* 0.034 / 2;

Serial.print("Distance 3: ");

Serial.println(distance3);

// Logic for servo 0 movement (Ultrasonic 1)

if (studentIDEntered) { // Only execute if a student ID has been entered

if (proximitySensorState1 == LOW || proximitySensorState2 == LOW) { // Either proximity sensor detects an object

pwm.setPWM(0, 0, angleToPulse(0)); // Keep servo 0 closed

Serial.println("Proximity sensor active. Servo 0 remains closed”.);

lcd.clear();

lcd.print("Not Allowed Metal, Glass) (Covered Bottle”,);

delay(1000); // Display "Not Allowed" on the LCD

notAllowedDisplayTime = millis(); // Record the time when "Not Allowed" is displayed

} else if (distance1 <= distanceThreshold && distance1 > 0) {

if (currentWeight < MIN\_WEIGHT\_THRESHOLD\_1 || currentWeight > MAX\_WEIGHT\_THRESHOLD\_1) { // Weight out of range

pwm.setPWM(0, 0, angleToPulse(0)); // Keep servo 0 closed

Serial.println("Weight out of range. Servo 0 remains closed”.);

lcd.clear();

lcd.print("");// Display "Not Allowed" on the LCD

notAllowedDisplayTime = millis(); // Record the time when "Not Allowed" is displayed

} else { // Weight is within range

Serial.println("Ultrasonic 1 detected within range. Waiting 2 seconds before opening servo 0”.);

// Wait for 5 seconds

pwm.setPWM(0, 0, angleToPulse(30)); // Open servo 0 to 45 degrees

Serial.println("Servo 0 opened to 45 degrees”.);

lcd.clear();

lcd.print("Accepted");

}

} else {

pwm.setPWM(0, 0, angleToPulse(0)); // Return servo 0 to closed position

Serial.println("No detection or weight out of range. Servo 0 remains closed”.);

}

}

// Logic for servo 2 movement (controlled by ultrasonic 3 and proximity sensor)

if (studentIDEntered) { // Only execute if a student ID has been entered

if (distance3 <= distanceThreshold && distance3 > 0 && proximitySensorState1 == HIGH && proximitySensorState2 == HIGH) {

if (currentWeight2 >= MIN\_WEIGHT\_THRESHOLD\_2 && currentWeight2 <= MAX\_WEIGHT\_THRESHOLD\_2) {

// Ultrasonic 3 detects within threshold and both proximity sensors do not detect an object

Serial.println("Ultrasonic 3 detected within range. Opening servo 2”.);

pwm.setPWM(2, 0, angleToPulse(30)); // Open servo 2 to 45 degrees

Serial.println("Servo 2 opened to 45 degrees”.);

lcd.clear();

lcd.print("Accepted");

} else { // Weight out of range

pwm.setPWM(2, 0, angleToPulse(0)); // Keep servo 2 closed

Serial.println("Weight out of range. Servo 2 remains closed”.);

lcd.clear();

lcd.print(""); // Display "Not Allowed" on the LCD

notAllowedDisplayTime = millis(); // Record the time when "Not Allowed" is displayed

}

} else {

pwm.setPWM(2, 0, angleToPulse(0)); // Return servo 2 to closed position

Serial.println("No detection or proximity sensor active. Servo 2 remains closed”.);

}

}

}

// Measure distance for second ultrasonic sensor

long duration2 = getUltrasonicDistance(trigPin2, echoPin2);

int distance2 = duration2 \* 0.034 / 2;

Serial.print("Distance 2: ");

Serial.println(distance2);

// Measure distance for fourth ultrasonic sensor

long duration4 = getUltrasonicDistance(trigPin4, echoPin4);

int distance4 = duration4 \* 0.034 / 2;

Serial.print("Distance 4: ");

Serial.println(distance4);

// Read proximity sensor states

int proximitySensorState1 = digitalRead(proximitySensorPin1); // First proximity sensor

int proximitySensorState2 = digitalRead(proximitySensorPin2); // Second proximity sensor

// Read the current weight from the load cells

currentWeight = loadCell1.get\_units(5); // Get the average of 5 readings

if (currentWeight < 0) {

currentWeight = 0.0; // Ensure weight is not negative

}

Serial.print("Weight 1: ");

Serial.print(currentWeight);

Serial.println(" g");

currentWeight2 = loadCell2.get\_units(5); // Get the average of 5 readings

if (currentWeight2 < 0) {

currentWeight2 = 0.0; // Ensure weight is not negative

}

Serial.print("Weight 2: ");

Serial.print(currentWeight2);

Serial.println(" g");

// Logic for servo 1 movement (Ultrasonic 2: 1 point = 3 minutes)

if (distance2 <= distanceThreshold && distance2 > 0 && allowDetection) {

pointsUltrasonic2 += 1; // Add 1 point (equivalent to 3 minutes)

updateLCDWithPointsAndTime(); // Update the LCD with points and converted time

Serial.println("Ultrasonic 2 detected within range. Added 1 point (3 minutes)”.);

allowDetection = false; // Block further detections

lastDetectionTime = millis(); // Record the time of the last detection

pwm.setPWM(1, 0, angleToPulse(30)); // Open servo 1 to 45 degrees

Serial.println("Servo 1 opened to 45 degrees”.);

} else {

pwm.setPWM(1, 0, angleToPulse(0)); // Return servo 1 to closed position

}

// Logic for servo 3 movement (Ultrasonic 4: 1 point = 1 minute)

if (distance4 <= distanceThreshold && distance4 > 0 && allowDetection) {

pointsUltrasonic4 += 1; // Add 1 point (equivalent to 1 minute)

updateLCDWithPointsAndTime(); // Update the LCD with points and converted time

Serial.println("Ultrasonic 4 detected within range. Added 1 point (1 minute)”.);

allowDetection = false; // Block further detections

lastDetectionTime = millis();

pwm.setPWM(3, 0, angleToPulse(50)); // Open servo 3 to 45 degrees

Serial.println("Servo 3 opened to 45 degrees”.);

} else {

pwm.setPWM(3, 0, angleToPulse(0)); // Return servo 3 to closed position

Serial.println("No detection. Servo 3 remains closed”.);

}

// Check if 3 seconds have passed since the last detection

if (!allowDetection && (millis() - lastDetectionTime >= detectionDelay)) {

allowDetection = true; // Allow the next detection

Serial.println("3 seconds passed. Ready for next detection”.);

}

// Check if 2 seconds have passed since "Not Allowed" was displayed

if (notAllowedDisplayTime > 0 && (millis() - notAllowedDisplayTime >= 2000)) {

lcd.clear(); // Clear the LCD screen

updateLCDWithPointsAndTime(); // Return to the counting display

notAllowedDisplayTime = 0; // Reset the timer

}

}

delay(10); // Small delay to reduce CPU usage

}

// Function to get ultrasonic distance

long getUltrasonicDistance(int trigPin, int echoPin) {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Measure the duration of the echo signal

return pulseIn(echoPin, HIGH);

}

// Function to update the LCD with points and converted time

void updateLCDWithPointsAndTime() {

lcd.clear();

lcd.print("Points(Big): ");

lcd.print(pointsUltrasonic2); // Display points from Ultrasonic 2

lcd.setCursor(0, 1);

lcd.print("Points(Small): ");

lcd.print(pointsUltrasonic4); // Display points from Ultrasonic 4

// Convert points to minutes

int minutesUltrasonic2 = pointsUltrasonic2 \* 3; // 1 point = 3 minutes for Ultrasonic 2

int minutesUltrasonic4 = pointsUltrasonic4 \* 1; // 1 point = 1 minute for Ultrasonic 4

int totalMinutes = minutesUltrasonic2 + minutesUltrasonic4; // Sum of converted minutes

lcd.setCursor(0, 2);

lcd.print("Total Mins:");

lcd.print(totalMinutes);

lcd.setCursor(20,3);

lcd.print("Press # to PRINT"); // Display the total converted minutes

}

// Function to print the receipt on the thermal printer

void printReceipt(String studentID, int pointsUltrasonic2, int pointsUltrasonic4) {

DateTime now = rtc.now(); // Get the current date and time from the RTC

printer.println("ARDUINO-BASED PLASTIC BOTTLE");

printer.println("CONVERSION INTO POINTS TO");

printer.println("DUTY HOURS IN CITY COLLEGE");

printer.println("OF TAGAYTAY");

printer.println(" ");

printer.println("Student ID: " + studentID); // Print Student ID

printer.println("Points(Big): " + String(pointsUltrasonic2)); // Print Points from Ultrasonic 2

printer.println("Points(Small): " + String(pointsUltrasonic4)); // Print Points from Ultrasonic 4

// Convert points to minutes

int minutesUltrasonic2 = pointsUltrasonic2 \* 3; // 1 point = 3 minutes for Ultrasonic 2

int minutesUltrasonic4 = pointsUltrasonic4 \* 1; // 1 point = 1 minute for Ultrasonic 4

int totalMinutes = minutesUltrasonic2 + minutesUltrasonic4; // Sum of converted minutes

printer.println("Converted Time: " + String(totalMinutes) + " mins"); // Print total converted minutes

printer.println("Date: " + String(now.year()) + "-" + String(now.month()) + "-" + String(now.day())); // Print Date

printer.println("Time: " + String(now.hour()) + ":" + String(now.minute()) + ":" + String(now.second())); // Print Time

printer.println(" ");

printer.println(" ");

printer.println(" ");

printer.println(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ");

printer.println(" OSES FACULTY ");

printer.println(" ");

printer.println("-----------------------------");

printer.println("PLEASE PROCEED TO THE OSES");

printer.println("TO CONFIRM YOUR DUTY HOURS"); // Print Converted Hours

printer.println("-----------------------------"); // Print a separator line

printer.feed(2); // Feed 2 lines

}