ANALYTICAL PSEUDO CODE

Import necessary libraries
Define pin connections for relays, servos, motor, load cell, and LCD

Initialize LCD with address 0x27, dimensions 16x2
Initialize Load Cell

Define variables: weight, oldweight, newweight

Setup function:
  Start serial communication at 115200 baud
  Initialize and clear LCD
  Display initial weight on LCD

  Set relay, servo, and motor pins as OUTPUT
  Turn off relays (initialize to LOW)

  Close both valves (initialize valve positions)

  Start load cell with specified pins and calibration scale
  Tare (zero) the scale

Loop function (for 5 iterations) :
  Move spout forward
  Open red servo valve
  Run pump for 3.5 seconds
  Measure and display weight on LCD

  Perform precise measurement adjustment:
    Check weight incrementally, pump if weight is less than target
    Stop pumping if target weight is reached

  Put load cell in sleep mode briefly, then wake it
  Update old weight

  Run reverse pump for 4 seconds to remove remaining water
  Close red servo valve
  Move spout back

  Open blue servo valve
  Run pump for 5.5 seconds
  Measure and display weight on LCD

  Perform precise measurement adjustment
  Put load cell in sleep mode briefly, then wake it

Run reverse pump for 4 seconds to remove remaining water
Close blue servo valve

Self-Defined functions:
moveServo(pulseWidth, servoPin):
Generate PWM signal for the servo using pulseWidth

closeValve(pin):
Print message indicating which valve is closing
Sweep servo from 0 to 90 degrees to close the valve

openValve(pin):
Print message indicating which valve is opening
Sweep servo from 90 to 0 degrees to open the valve

pump(duration):
Activate relay for specified duration
Deactivate relay, wait briefly

reversepump(duration):
Activate reverse relay for specified duration
Deactivate relay, wait briefly

precisepump(oldweight,target):
Loop up to 10 times:
Measure weight and calculate weight difference from oldweight
Display weight
If weight difference is less than target, pump briefly
If target is reached, stop pumping
Wait briefly before next check

lcdprint(weight):
Print weight on serial monitor and LCD display

movespout(forward):
Set motor direction based on forward or backward
Rotate motor for a complete revolution

TECHNICAL PSEUDO CODE

```
// Import libraries and define pin connections/constants

START FUNCTION setup:
   INITIALIZE Serial at 115200 baud
   INITIALIZE LCD with 16x2 display
   DISPLAY "Weight" on LCD

   SET relayPin, revrelayPin, servobluePin, servoredPin, stepPin, dirPin as OUTPUT
   TURN OFF relays by setting relayPin, revrelayPin to LOW

   INITIALIZE both valves to closed by CALLING closeValve(servoredPin) and
closeValve(servobluePin)

   INITIALIZE load cell with pins LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN
   CALIBRATE scale with set_scale value, tare to zero

   CALL five_looper()  // Start main dispensing sequence
   DISPLAY "Task Completed" on LCD
END FUNCTION

START FUNCTION loop:
   // Empty main loop, all logic handled in setup
END FUNCTION

START FUNCTION five_looper:
   FOR i FROM 0 TO 4:
      CALL movespout(true)        // Move spout forward
      CALL openValve(servoredPin)  // Open red valve
      CALL pump(3500)             // Run pump for 3.5 seconds
      SET weight = get_weight()    // Measure weight
      CALL lcdprint(weight)       // Display weight

      CALL precisepump(oldweight, 5.0)  // Dispense precisely to reach target weight
      CALL scale.power_down()     // Power down load cell
      WAIT 1000 milliseconds
      CALL scale.power_up()       // Power up load cell

      SET oldweight = weight
      CALL reversepump(4000)      // Remove remaining fluid
      CALL closeValve(servoredPin) // Close red valve
      CALL movespout(false)       // Move spout back
      CALL openValve(servobluePin) // Open blue valve
      CALL pump(5500)             // Run pump for 5.5 seconds
      SET weight = get_weight()
```

```
      CALL lcdprint(weight)

      CALL precisepump(oldweight, 10.0) // Dispense precisely for blue valve
      CALL scale.power_down()
      WAIT 1000 milliseconds
      CALL scale.power_up()

      SET oldweight = weight
      CALL reversepump(4000)      // Remove remaining fluid
      CALL closeValve(servobluePin)
   END FOR
END FUNCTION

START FUNCTION moveServo(pulseWidth, servoPin):
   SET servoPin HIGH
   WAIT pulseWidth microseconds
   SET servoPin LOW
   WAIT (20 - (pulseWidth / 1000)) milliseconds // Maintain 20ms period
END FUNCTION

START FUNCTION closeValve(pin):
   IF pin == servoredPin THEN:
      DISPLAY "Closing Red valve"
   ELSE:
      DISPLAY "Closing Blue valve"
   END IF
   FOR angle FROM 0 TO 90:
      SET pulseWidth = map(angle, 0, 180, 500, 2500)
      CALL moveServo(pulseWidth, pin)
      WAIT 15 milliseconds
   END FOR
END FUNCTION

START FUNCTION openValve(pin):
   IF pin == servoredPin THEN:
      DISPLAY "Opening Red valve"
   ELSE:
      DISPLAY "Opening Blue valve"
   END IF
   FOR angle FROM 90 TO 0:
      SET pulseWidth = map(angle, 0, 180, 500, 2500)
      CALL moveServo(pulseWidth, pin)
      WAIT 15 milliseconds
   END FOR
END FUNCTION

START FUNCTION pump(dur):
```

```
    DISPLAY "Pump ON"
    SET relayPin HIGH
    WAIT dur milliseconds
    DISPLAY "Pump OFF"
    SET relayPin LOW
    WAIT 2000 milliseconds
END FUNCTION

START FUNCTION reversepump(dur):
    DISPLAY "Reverse Pump ON"
    SET revrelayPin HIGH
    WAIT dur milliseconds
    DISPLAY "Reverse Pump OFF"
    SET revrelayPin LOW
    WAIT 2000 milliseconds
END FUNCTION

START FUNCTION precisepump(oldweight, target):
    FOR i FROM 0 TO 9:
        SET weight = get_weight()
        SET newweight = weight - oldweight
        CALL lcdprint(weight)

        IF newweight < target THEN:
            CALL pump(800)
        ELSE:
            DISPLAY "Target Weight dispensed"
            CALL lcd.clear()
            CALL lcd.setCursor(0, 0)
            lcd.print("Fluid Released: ")
            CALL lcd.setCursor(0, 1)
            lcd.print(newweight / 10.0, 1)
            WAIT 1000 milliseconds
            BREAK
        END IF

        CALL lcdprint(weight)
    END FOR
END FUNCTION

START FUNCTION lcdprint(weight):
    DISPLAY "Weight: ", weight
    CALL lcd.setCursor(0, 0)
    lcd.print("Weight: ")
    CALL lcd.setCursor(0, 1)
    lcd.print(weight, 1)
END FUNCTION
```

```
START FUNCTION movespout(forw):
   IF forw THEN:
      SET dirPin HIGH
      DISPLAY "Moving Spout Forward"
   ELSE:
      SET dirPin LOW
      DISPLAY "Moving Spout Backward"
   END IF

   FOR x FROM 0 TO stepsPerRevolution:
      SET stepPin HIGH
      WAIT 2000 microseconds
      SET stepPin LOW
      WAIT 2000 microseconds
   END FOR

   WAIT 2000 milliseconds
END FUNCTION
```

Complete Code for ESP32

```
//Importing Libraries
#include "HX711.h"
#include <LiquidCrystal_I2C.h>

// Defining pin connections & motor's steps per revolution
int relayPin = 26;
int revrelayPin = 27;
int servoredPin = 18;
int servobluePin = 19;
const int dirPin = 17;
const int stepPin = 23;
const int stepsPerRevolution = 900;
const int LOADCELL_DOUT_PIN = 2;
const int LOADCELL_SCK_PIN = 4;

// Setting the LCD address & size
LiquidCrystal_I2C lcd(0x27, 16, 2);

HX711 load_cell;
HX711 scale;

float weight = 0.0;
float oldweight = 0.0;
float newweight = 0.0;

void setup() {

  // Initializing the serial communication
  Serial.begin(115200);

  // Initialize the LCD
  lcd.init();
  lcd.clear();
  lcd.backlight();

  //Display Weight
```

```
  lcdprint(weight);

  // Setting the pins as OUTPUT
  pinMode(relayPin, OUTPUT);
  pinMode(revrelayPin, OUTPUT);
  pinMode(servobluePin, OUTPUT);
  pinMode(servoredPin, OUTPUT);
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);

  // Initialize the relay to be off (LOW turns off the relay)
  digitalWrite(relayPin, LOW);
  digitalWrite(revrelayPin, LOW);

  //Initializing valve position
  closeValve(servoredPin);
  closeValve(servobluePin);

  //Starting the load cell
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale(3257.28);
  scale.tare();

  //running the main code
  five_looper();

  //end
  delay(1000);
  lcd.setCursor(0, 0);
  lcd.clear();
  lcd.print("      Task");
  lcd.setCursor(0, 1);
  lcd.print("   Completed");
}

void loop(){

}

void five_looper() {
```

```
for (int i = 0; i < 5; i++) {
  //moving spout forward
  movespout(true);

  //red servo open
  openValve(servoredPin);

  //pump for 4 sec
  pump(4000);

  //weight
  weight = scale.get_units(10);

  //Display Weight
  lcdprint(weight);

  //Precise Measurement
  precisepump(oldweight, 5.0);

  // put the ADC in sleep mode
  scale.power_down();
  delay(1000);
  scale.power_up();

  oldweight = weight;

  //removing remaining water
  reversepump(4000);

  //red servo close
  closeValve(servoredPin);

  //moving spout back
  movespout(false);

  //blue servo open
  openValve(servobluePin);

  //pump for 5.5 sec
  pump(5500);
```

```
    //weight
    weight = scale.get_units(10);

    //Display Weight
    lcdprint(weight);

    //Precise Measurement
    precisepump(oldweight, 10.0);

    // put the ADC in sleep mode
    scale.power_down();
    delay(1000);
    scale.power_up();

    oldweight = weight;

    //removing remaining water
    reversepump(3500);

    //blueservo close
    closeValve(servobluePin);
  }
}

void moveServo(int pulseWidth, int servoPin) {
  // Manually generate the PWM signal for the servo
  digitalWrite(servoPin, HIGH);
  delayMicroseconds(pulseWidth);
  digitalWrite(servoPin, LOW);

  delay(20 - (pulseWidth / 1000));  // 20ms period minus the pulse width in milliseconds
}

void closeValve(int pin) {
  if (pin == servoredPin) {
    Serial.println("Closing Red valve");
  } else {
    Serial.println("Closing Blue valve");
  }
```

```
  // Sweep from 0 to 90 degrees
  for (int angle = 0; angle <= 90; angle++) {
    int pulseWidth = map(angle, 0, 180, 500, 2500);  // Mapping angle to pulse width (500µs to 2500µs)
    moveServo(pulseWidth, pin);                      // Moving servo to current angle
    delay(15);
  }
}

void openValve(int pin) {
  if (pin == servoredPin) {
    Serial.println("Opening Red valve");
  } else {
    Serial.println("Opening Blue valve");
  }
  // Sweep back from 90 to 0 degrees
  for (int angle = 90; angle >= 0; angle--) {
    int pulseWidth = map(angle, 0, 180, 500, 2500);  // Mapping angle to pulse width
    moveServo(pulseWidth, pin);                      // Moving servo to current angle
    delay(15);
  }
}

void pump(int dur) {
  // Turn the relay ON
  Serial.println("Pump ON");
  digitalWrite(relayPin, HIGH);  // Activating the relay
  delay(dur);

  // Turn the relay OFF
  Serial.println("Pump OFF");
  digitalWrite(relayPin, LOW);  // Deactivating the relay
  delay(2000);
}

void reversepump(int dur) {
  // Turn the relay ON
  Serial.println("Reverse Pump ON");
  digitalWrite(revrelayPin, HIGH);  // Activating the relay
  delay(dur);
```

```
  // Turn the relay OFF
  Serial.println("Reverse Pump OFF");
  digitalWrite(revrelayPin, LOW);  // Deactivating the relay
  delay(2000);
}

void precisepump(float oldweight, float target) {
  for (int i = 0; i < 10; i++) {
    weight = scale.get_units(10);
    newweight = weight - oldweight;

    //Display weight
    lcdprint(weight);

    if (newweight < target) {
      //pump for 0.8 seconds
      pump(800);
    } else {
      Serial.println("Target Weight dispensed");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Fluid Released: ");
      lcd.setCursor(0, 1);
      lcd.print(newweight / 10.0, 1);
      delay(1000);
      break;
    }
    //Display weight
    lcdprint(weight);
  }
}

  void lcdprint(float weight) {

    //Printing on serial monitor
    Serial.print("Weight: ");
    Serial.println(weight, 1);

    // Display text
    lcd.setCursor(0, 0);
```

```
  lcd.print("Weight: ");
  lcd.setCursor(0, 1);
  lcd.print(weight, 1);
}

void movespout(bool forw) {
  if (forw) {
    // Set motor direction clockwise
    digitalWrite(dirPin, HIGH);
    Serial.println("Moving Spout Forward");
  } else {
    // set motor direction anticlockwise
    digitalWrite(dirPin, LOW);
    Serial.println("Moving Spout Backward");
  }

  // Spin motor slowly
  for (int x = 0; x < stepsPerRevolution; x++) {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(2000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(2000);
  }

  delay(2000);
}
```

Team Id: TW-LAM-630