```cpp
//*****************************//
// Libraries
//*****************************//
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//*****************************//
// Global variables
//*****************************//
LiquidCrystal_I2C lcd(0x27, 16, 2);

const int dirPin = 9;
const int stepPin = 8;
const int alarmPin = 12;

// Core variables
float hopperWeight = 0;
float calibrationWeight = 0;
int calibrationSteps = 5000;
float pourWeight = 0;
long pourSteps = 0;
int secondsBetweenPours = 0;
unsigned long lastPourMillis = 0;
long totalStepsPoured = 0;
long stepsBeforeRefill = 0;
float maxHoldWeight = 100;

// System state
String inputString = "";
bool stringComplete = false;
bool systemRunning = false;
bool refillNeeded = false;

// Alarm variables
unsigned long previousAlarmMillis = 0;
bool alarmOn = false;
const unsigned long alarmOnDuration = 200;
const unsigned long alarmOffDuration = 1000;
unsigned long alarmInterval = alarmOffDuration;

//*****************************//
// Functions
//*****************************//
void stepMotor(int steps) {
  digitalWrite(dirPin, HIGH);
  for (int i = 0; i < steps; i++) {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(750);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(750);
  }
}

void calculatePourSteps() {
  if (calibrationWeight > 0) pourSteps = ((float)calibrationSteps / calibrationWeight) * pourWeight;
  else pourSteps = 0;
}

void calculateRefillSteps() {
  if (calibrationWeight > 0) stepsBeforeRefill = ((float)calibrationSteps / calibrationWeight) * hopperWeight;
  else stepsBeforeRefill = 0;
}

void pourPellets() {
  lcd.clear();
  lcd.print("Pouring...");
  Serial.println("Pouring pellets");

  // Pre-pour beeps
  for (int i = 0; i < 3; i++) {
    tone(alarmPin, 800, 200);
    delay(250);
  }
  delay(500);

  calculatePourSteps();
  stepMotor(pourSteps);
  totalStepsPoured += pourSteps;

  calculateRefillSteps();
  if (totalStepsPoured >= stepsBeforeRefill) {
    refillNeeded = true;
    systemRunning = false;
    Serial.println("Refill needed");
    lcd.clear();
    lcd.print("Refill Needed!");
  } else {
    Serial.println("Pour complete");
  }
}

void playAlarm() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousAlarmMillis >= alarmInterval) {
    previousAlarmMillis = currentMillis;
    if (alarmOn) {
      noTone(alarmPin);
      alarmOn = false;
      alarmInterval = alarmOffDuration;
    } else {
      tone(alarmPin, 1000);
      alarmOn = true;
      alarmInterval = alarmOnDuration;
    }
  }
}

void updateLCD() {
  lcd.clear();
  lcd.print("Auto-Feeder");
  lcd.setCursor(0, 1);

  if (systemRunning) {
    unsigned long currentMillis = millis();
    unsigned long elapsedMillis = (currentMillis >= lastPourMillis) ? (currentMillis - lastPourMillis) : (0xFFFFFFFF - lastPourMillis) + currentMillis + 1;

    int timeRemaining = secondsBetweenPours - (elapsedMillis / 1000);
    if (timeRemaining < 0) timeRemaining = 0;

    lcd.print("Next: ");
    lcd.print(timeRemaining);
    lcd.print("s");
  } else if (refillNeeded) {
    lcd.print("Refill Required");
  } else {
    lcd.print("Ready");
  }
}

void processCommand(String cmd) {
  cmd.trim();

  if (cmd.startsWith("cal ")) {
    calibrationWeight = cmd.substring(4).toFloat();
    Serial.print("Cal weight: ");
    Serial.println(calibrationWeight);

  } else if (cmd.startsWith("pour ")) {
    pourWeight = cmd.substring(5).toFloat();
    Serial.print("Pour weight: ");
    Serial.println(pourWeight);

  } else if (cmd.startsWith("time ")) {
    secondsBetweenPours = cmd.substring(5).toInt();
    Serial.print("Time: ");
    Serial.println(secondsBetweenPours);

  } else if (cmd.startsWith("hopper ")) {
    hopperWeight = cmd.substring(7).toFloat();
    totalStepsPoured = 0;
    refillNeeded = false;
    calculateRefillSteps();
    Serial.print("Hopper: ");
    Serial.print(hopperWeight);
    Serial.print("g, refill at: ");
    Serial.println(stepsBeforeRefill);

  } else if (cmd == "start") {
    if (calibrationWeight > 0 && pourWeight > 0 && secondsBetweenPours > 0) {
      systemRunning = true;
      lastPourMillis = millis();
      Serial.println("System started");
    } else {
      Serial.println("Set all parameters first");
    }

  } else if (cmd == "stop") {
    systemRunning = false;
    Serial.println("System stopped");

  } else if (cmd == "alarm_on") {
    tone(alarmPin, 1000);
    Serial.println("Alarm on");

  } else if (cmd == "alarm_off") {
    noTone(alarmPin);
    Serial.println("Alarm off");

  } else if (cmd == "status") {
    Serial.print("Running:");
    Serial.print(systemRunning ? "Y" : "N");
    Serial.print(" Refill:");
    Serial.print(refillNeeded ? "Y" : "N");
    Serial.print(" Cal:");
    Serial.print(calibrationWeight);
    Serial.print(" Pour:");
    Serial.print(pourWeight);
    Serial.print(" Time:");
    Serial.print(secondsBetweenPours);
    Serial.print(" Hopper:");
    Serial.println(hopperWeight);

  } else if (cmd == "manual") {
    if (calibrationWeight > 0 && pourWeight > 0) {
      pourPellets();
    } else {
      Serial.println("Set cal and pour weight first");
    }

  } else if (cmd == "calibrate") {
    Serial.println("Calibrating...");
    lcd.clear();
    lcd.print("CALIBRATING...");
    stepMotor(calibrationSteps);
    Serial.println("Weigh pellets and set cal weight");
  } else if (cmd == "help") {
    Serial.println("Commands:");
    Serial.println("cal <weight> - set calibration");
    Serial.println("pour <weight> - set pour amount");
    Serial.println("time <seconds> - set time between");
    Serial.println("hopper <weight> - set hopper weight");
    Serial.println("start/stop - control system");
    Serial.println("manual - manual pour");
    Serial.println("calibrate - run calibration");
    Serial.println("status - show status");
    Serial.println("alarm_on/alarm_off");

  } else {
    Serial.println("Unknown command - type 'help'");
  }
}

//*****************************//
// Setup and Loop
//*****************************//
void setup() {
  pinMode(dirPin, OUTPUT);
  pinMode(stepPin, OUTPUT);

  lcd.init();
  lcd.backlight();

  Serial.begin(9600);
  while (!Serial) {
    ;  // Wait for serial port to connect
  }
  Serial.println("Pellet dispenser ready");

  updateLCD();
  inputString.reserve(50);
}

void loop() {
  if (stringComplete) {
    processCommand(inputString);
    inputString = "";
    stringComplete = false;
  }

  if (systemRunning && !refillNeeded) {
    unsigned long currentMillis = millis();
    unsigned long elapsedMillis = (currentMillis >= lastPourMillis) ? (currentMillis - lastPourMillis) : (0xFFFFFFFF - lastPourMillis) + currentMillis + 1;

    if (elapsedMillis >= (unsigned long)secondsBetweenPours * 1000UL) {
      pourPellets();
      lastPourMillis = millis();
    }
  }

  if (refillNeeded) {
    playAlarm();
  }

  static unsigned long lastLCDUpdate = 0;
  if (millis() - lastLCDUpdate > 1000) {
    updateLCD();
    lastLCDUpdate = millis();
  }

  delay(10);
}

void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    if (inChar == '\n') {
      stringComplete = true;
    } else {
      inputString += inChar;
    }
  }
}
```