Semi-Automatic Rotary Dispenser

Objective

To design and implement a semi-automatic rotary dispensing system that allows for both manual and automatic control of the motor through sensors and buttons. The system will serve as a base for feeding rodents or for other controlled behavior experiments.

Materials

Component	Quantity	Remarks
Infrared Sensor TCRT5000	1	Detects proximity within the system
IR Break Beam Sensor	1	Detects interruption when passing an object
Arduino UNO or compatible	1	Main microcontroller
28BYJ-48 Stepper Motor	1	Rotator motor with gearbox
ULN2003 module for the engine	1	Driver to control the engine
Breadboard	1	For temporary connections
Push buttons	2	To enable manual rotation or reset
LEDs (red and green)	2	Visual indicators
Resistors 220Ω	2	For LED protection
Jumpers and cables	Several	For connections

CAD Design

The system was designed in Fusion 360 and SolidWorks, with the following features in mind:

- A central compartment where the 28BYJ-48 engine is housed.
- Rotating disc where the motor will be placed

.f3d file available for printing and simulation.

Print design

The parts are designed for FDM printers:

- Recommended Material: PLA
- Assembly tolerances of 0.3 mm for a good fit between parts.
- Includes:
  - Base box
  - Rotary arm (spoon type)
  - Slot lid
  - Sensor support
  - Shaft for coupling the 28BYJ-48 motor

Implementation

1. **Semi-automatic mode:**
   - When the infrared sensor detects proximity, the red LED lights up.
   - If the Beam sensor is interrupted (e.g. the mouse sticks out its paw), an automatic motor sequence is activated.
   - The motor rotates a certain number of steps and stops, dispensing a portion.
2. Manual Mode:
   - The first button manually activates the stepper motor for one rotation.
   - The second button restarts the system or performs an additional function, such as a second dispense.
3. Indicators:
   - Red LED: Proximity detected.
   - Blue LED: Interrupt in the beam sensor

Basic connections:

- TCRT5000 (IR Sensor):
  - OUT → pin 4 (digital)
- Beam sensor:
  - OUT → pin 5 (digital)

- Bellboy:
    - BTN1 → pin 6 (with INPUT_PULLUP)
    - BTN2 → pin 7
- Leds:
    - LED1 → pin 2 (with 220Ω resistor)
    - LED2 → pin 3
- Engine:
    - Connected to module ULN2003 → pins 8, 9, 10, 11

Code

```
#include <Stepper.h>

Engine
const int stepsPerRevolution = 2048;
Stepper myStepper(stepsPerRevolution, 8,
10, 9, 11);

Bellboy
const int btnForward = 6;
const int btnBackward = 7;
bool lastBtnFwd = HIGH;
bool lastBtnBwd = HIGH;

// Sensor TCRT5000
const int sensorA0 = A0;   Analog sensor pin
const int ledPin = 5;       LED indicator
const int thresholdProximity = 500;  Adjust
this value according to your sensor

void setup() {
  Engine
  myStepper.setSpeed(10);

  Bellboy
  pinMode(btnForward, INPUT_PULLUP);
  pinMode(btnBackward, INPUT_PULLUP);

  Sensor
  pinMode(ledPin, OUTPUT);

  Serial
  Serial.begin(9600);
  Serial.println("Combined system ready.
Press buttons to move the engine.");
}
```

```
void loop() {
  ---SENSOR---
  int read = analogRead(sensorA0);
  Serial.print("A0 Sensor: ");
  Serial.print(read);

  if (reading > thresholdCloseness) {
    Serial.println(" --> 🟢 FREE");
    digitalWrite(ledPin, LOW);
  } else {
    Serial.println(" --> 🔴 CLOSE");
    digitalWrite(ledPin, HIGH);
  }

  --- BUTTONS WITH FALLING EDGE ---
  bool currentBtnFwd =
digitalRead(btnForward);
  bool currentBtnBwd =
digitalRead(btnBackward);

  if (lastBtnFwd == HIGH && currentBtnFwd
== LOW) {
 Serial.println(" ➡️ Moving forward 1/16 of a
turn");
    myStepper.step(stepsPerRevolution/16);
  }

  if (lastBtnBwd == HIGH && currentBtnBwd
== LOW) {
 Serial.println(" ⬅️ Going back 1/16 of a
turn");
    myStepper.step(-stepsPerRevolution / 16);
  }
  lastBtnFwd = currentBtnFwd;
  lastBtnBwd = currentBtnBwd;

  delay(500);  More stable reading
}
```

Evidence