**Laundry Day: A Stepper Motor-Driven Interactive Project**

**Kristen, Pavan, Caliste, Jacy, Kunstin**

For our group project, we set out to create something both visually engaging and mechanically functional using a **stepper motor** as our primary actuator. The result was **"Laundry Day"** a playful, motorized installation featuring miniature clothing and ribbons that move along a structured frame, powered by **Arduino** and controlled via button inputs. This project combined elements of creative design, electrical engineering, and software development to bring an interactive experience to life.

**Design Concept & Team Contributions**

The initial inspiration stemmed from the idea of **mechanizing a simple, everyday task hanging and moving laundry on a line.** To achieve this, we designed a **rigid frame using wood dowels** that supported a pulley-based movement system. The design featured ribbons and tiny clothing pieces, emphasizing the visual aspect while ensuring the mechanical components functioned smoothly.

Several team members played pivotal roles:

* **Kristen & Pocket** took charge of the **Arduino coding**, ensuring the stepper motor could respond to button inputs and operate smoothly. They structured the program logic to handle different power states and motor speeds effectively.
* **Caliste, Jacy, and Kristen** contributed to the overall **design concept, structural build, and aesthetic elements**, ensuring that the ribbons and clothing pieces aligned well with the movement mechanism along with the write up.
* **The entire team** worked on integrating electrical components efficiently to match the required **voltage and current calculations**, ensuring the system functioned safely within the given power constraints.

**Technical Implementation**

Our **motor control system** is based on an **Arduino microcontroller** that interfaces with the **stepper motor** and receives inputs from **physical buttons** mounted on the frame. The primary electrical components include:

* **A stepper motor** (MS) for precise, controlled movement of the laundry mechanism.
* **Buttons (B1, B2)** for user interaction, allowing control over motor activation and deactivation.
* **A DC fan motor (MF)** to add an additional dynamic element, possibly simulating the effect of wind on the clothes.
* **An on/off switch (S)** to control power flow.
* **Ribbons (R) and mini clothing elements** attached to the structure to enhance the interactive experience.
* **An Arduino board (A)** programmed to control motor behavior based on user inputs.
* **A 9V battery** for power supply, ensuring sufficient voltage for both the motor and Arduino components.

The **Arduino sketch** (program) handles stepper motor control by mapping digital pins **8, 9, 10, and 11** for motor output, while buttons are connected through **analog inputs** to detect user interaction.

**Electrical Considerations & Ohm’s Law**

A critical part of this project was ensuring the stepper motor and fan motor operated within their required **voltage (V), current (I), and resistance (R) parameters**, following **Ohm’s Law**:

V=IRV = IR

Using this equation, we determined how to distribute power efficiently, ensuring the stepper motor did not draw excess current beyond its rated limit. Since stepper motors require **precise current regulation**, we had to make sure our circuit design maintained stable power delivery while **avoiding overheating or excessive load on the battery**.

The **buttons were connected using pull-down resistors**, ensuring they correctly registered user input without erratic behavior due to floating voltage levels.

**Coding & Functionality**

The code, written in **Arduino’s IDE**, allows the stepper motor to respond to button presses. The logic ensures that:

* Pressing one button **activates** the motor (HIGH state).
* Pressing another button **deactivates** the motor (LOW state).
* The system remains stable even with multiple input signals.

The motor’s movement is controlled through **pulse-width modulation (PWM)** signals sent from the Arduino to the motor driver, ensuring precise angular rotation for controlled movement.

Kristen & Pocket were instrumental in structuring the program logic to ensure **smooth transitions between ON and OFF states**, preventing motor jitter and ensuring reliability. **How did you determine the appropriate voltage and current settings for the stepper motor?**

@Pavan - The stepper motor has a listed rating for 5v-12v according to the motor controller board and is driven by 4 inputs on the Arduino.

1. **How does the code manage the transition between “HIGH” (On) and “LOW” (Off) states while maintaining motor stability?**

@Pavan - The code's main loop is constantly checking for button input which determines whether to send the steps() command to the motor using the stepper motor library. If there is no input, no command is sent, and similarly if both buttons are pressed, no command is sent. If the left button is pressed, the motor spins counterclockwise, and if the right button is pressed, the motor is commanded to step clockwise.

@Kunstin - I modified the code so that two buttons control the stepper motor’s clockwise and counterclockwise rotation. Initially, pressing a button would rotate the motor in the corresponding direction, but when no button was pressed, it defaulted to clockwise rotation. I changed the trigger condition for direction switching to detect button state changes—whenever a button transitions from unpressed to pressed, the motor changes direction. This ensures that each button independently controls its respective rotation direction.

1. **What were the biggest challenges in integrating the stepper motor with Arduino, and how did you troubleshoot them?**

@Pavan - The hardest problem was accounting for transient voltage that appeared on the other inputs probably due to induction and back emf from the motor starting and stopping, so the inputs detecting button presses had to be set to read analog voltage with a very high threshold amount of 1000/1024 to ensure there weren't false detections of button input.

**Challenges & Lessons Learned**

1. **Power Management:** Initially, we faced issues with inconsistent motor power due to **voltage drops and insufficient current draw**. Adjusting resistor values and ensuring correct wiring helped stabilize performance.
2. **Button Debouncing:** Physical button presses sometimes caused **erratic behavior** due to unintended multiple signals. We mitigated this by **implementing a debounce function in the code**, ensuring reliable detection.
3. **Structural Alignment:** The pulleys and ribbons had to be carefully **aligned** to avoid excess friction, which could impact movement efficiency. Adjusting tension and ensuring smooth movement helped resolve this.

**Final Thoughts**

This project successfully integrated **electronics, coding, and creative design**, demonstrating how stepper motors can be used in interactive installations. It highlights the importance of **Ohm’s Law, power distribution, and button-controlled motor logic** in a small-scale system.

By combining **aesthetic elements (ribbons, mini clothing) with technical functionality (motor control, Arduino coding)**, our team created an engaging, interactive experience that demonstrates the **fundamentals of electrical engineering in a fun and approachable way**.

Going forward, we could explore further refinements, such as **adding a motor driver for smoother performance**, or incorporating **sensors** to enhance automation.