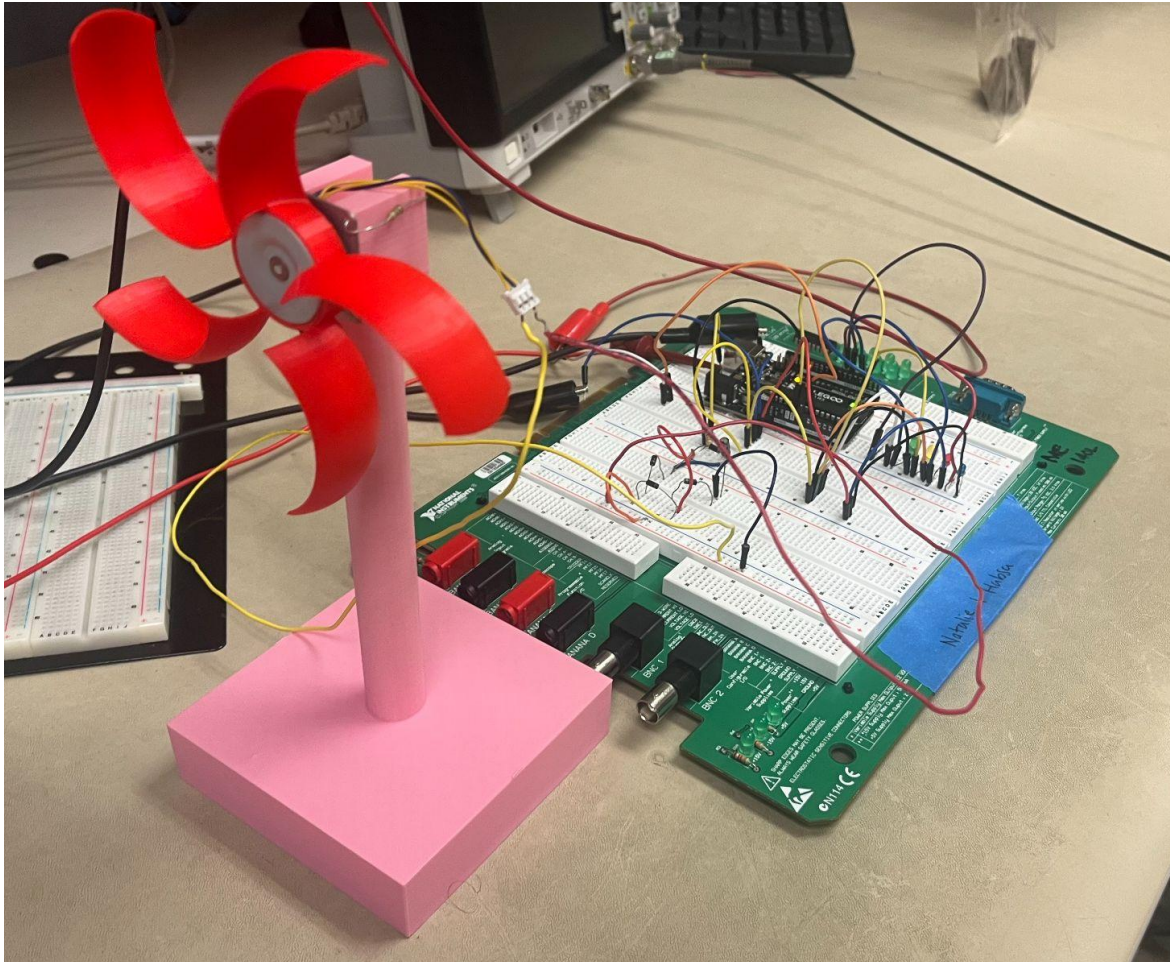
Mechatronics Design Project

By: Natalie Rice & Habsa Aydid



Circuit Design Overview

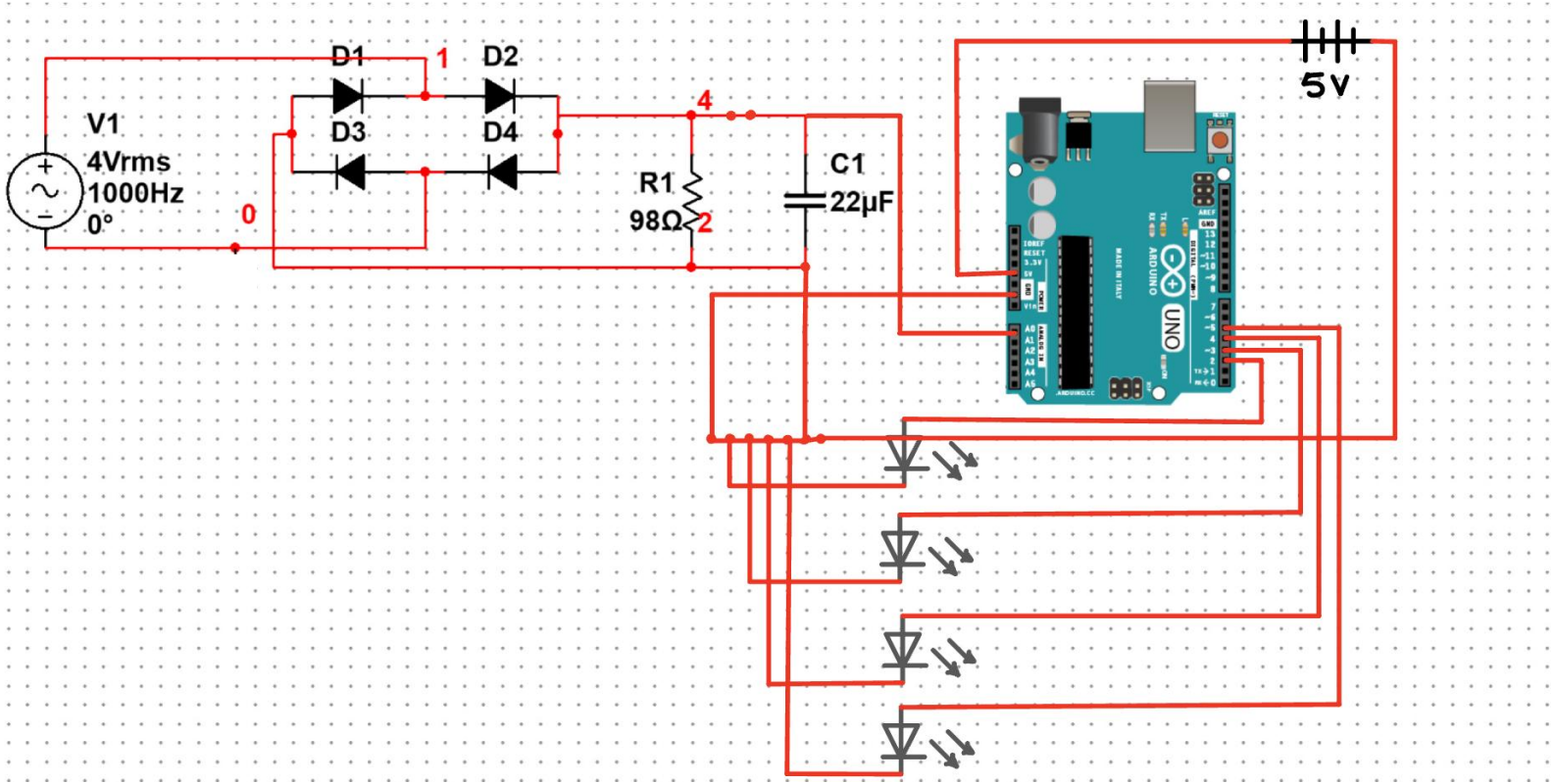
Components Used:

- Electric Fan + Windmill Fan Blades - supply mechanical power to motor
- Motor - converts mechanical power from the fan to AC electrical power
- Four 1N4007 Diodes - full-wave bridge rectifier, converts AC power to DC power
- 0.98kΩ Resistor - control current to LEDs
- 22μF Capacitor - smooths rectified DC voltage
- Arduino - logic controller programed with voltage thresholds
- 5V DC Power Source - connected to Arduino
- 5 LEDs - intensity, too fast, too slow, low acceptable speed, high acceptable speed
- Breadboard
- Jumper wires

Circuit Description:

1) The windmill spins a **motor**. The **Motor** produces an **AC voltage** that increases as the windmill spins faster.
2) Four **diodes** are arranged as a **full-wave bridge rectifier**, which **converts the motor's AC output into a positive DC** signal.
3) The **capacitor** is connected across the rectifier output to **smooth the DC voltage**, reducing ripple so the voltage stays steady instead of pulsing.
4) The **resistor** is used to **control the amount of current flowing to the LEDs**.
5) One **LED** is connected to the smoothed DC output to show the voltage level visually and **indicate the generator's intensity.**
6) The same smoothed DC voltage is sent to the **Arduino's A0** analog input, which **measures the rectified DC voltage.**
7) In the **Arduino's voltage thresholds** are programmed:
    - Below the low threshold **(0 - 0.12 V)**→ **"Too Slow" LED** turns on
    - Above the high threshold **(> 0.91 V)** → **"Too Fast" LED** turns on
    - Between low threshold and mid acceptable speed **(0.12 - 0.5 V)** → the windmill has **low acceptable speed**
    - Between mid acceptable speed and high threshold **(0.5 - 0.91 V)** → the windmill has **high acceptable speed**

Circuit Diagram



Arduino Code

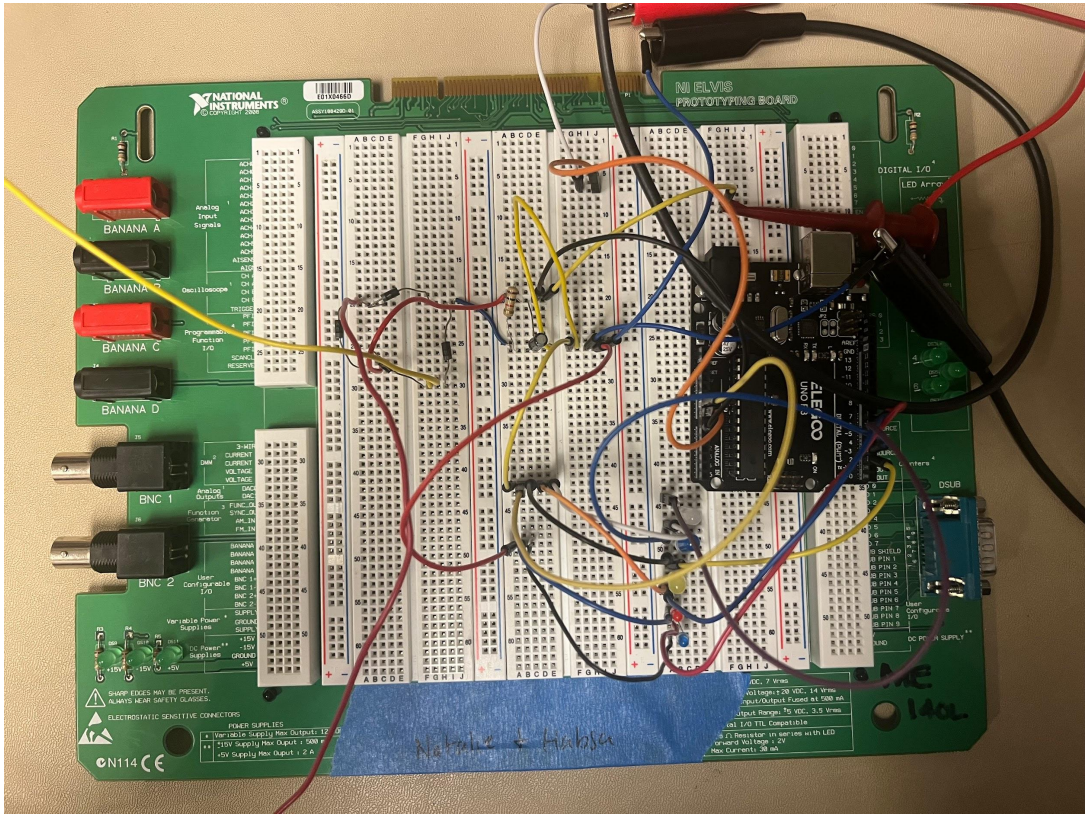
```
// ----- PIN SETUP -----
const int sensorPin      = A0;  // rectifier DC+ (generator output)
const int tooLowLED      = 2;   // "too low" indicator
const int tooHighLED     = 3;   // "too high" indicator
const int speedLED       = 5;   // PWM brightness output

// Acceptable range LEDs
const int lowAcceptLED   = 6;   // RED  : lowest acceptable
const int highAcceptLED  = 7;   // BLUE : highest acceptable

// ---- Voltage Thresholds ----
const float LOW_MAX      = 0.12;   // too low
const float MID_SPLIT    = 0.50;   // divider between low/high acceptable
const float HIGH_MIN     = 0.91;   // too high

// Convert voltage → ADC
int voltToADC(float v) {
  return int((v / 5.0) * 1023.0);
}
```

Arduino Code

```
// Pre-converted thresholds
const int ADC_LOW_MAX   = voltToADC(LOW_MAX);     // 0.12 V
const int ADC_MID_SPLIT = voltToADC(MID_SPLIT);   // 0.50 V
const int ADC_HIGH_MIN  = voltToADC(HIGH_MIN);    // 0.91 V

void setup() {
  pinMode(tooLowLED,     OUTPUT);
  pinMode(tooHighLED,    OUTPUT);
  pinMode(speedLED,      OUTPUT);
  pinMode(lowAcceptLED,  OUTPUT);
  pinMode(highAcceptLED, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(sensorPin);
  float voltage = sensorValue * (5.0 / 1023.0);

  // PWM brightness: scale full range to 0–255
  int pwmValue = map(sensorValue, 0, ADC_HIGH_MIN, 0, 255);
  pwmValue = constrain(pwmValue, 0, 255);
  analogWrite(speedLED, pwmValue);
```

```
// Turn all LEDs off before selecting correct range
  digitalWrite(tooLowLED,     LOW);
  digitalWrite(lowAcceptLED,  LOW);
  digitalWrite(highAcceptLED, LOW);
  digitalWrite(tooHighLED,    LOW);

  // ---- RANGE CHECK LOGIC ----

  if (sensorValue < ADC_LOW_MAX) {
    // TOO LOW (0–0.12 V)
    digitalWrite(tooLowLED, HIGH);
  }
  else if (sensorValue < ADC_MID_SPLIT) {
    // LOWEST ACCEPTABLE (0.12–0.50 V)
    digitalWrite(lowAcceptLED, HIGH);
  }
  else if (sensorValue < ADC_HIGH_MIN) {
    // HIGHEST ACCEPTABLE (0.50–0.91 V)
    digitalWrite(highAcceptLED, HIGH);
  }
  else {
    // TOO HIGH (>0.91 V)
    digitalWrite(tooHighLED, HIGH);
  }

  // Debug print
  Serial.print("ADC = ");
  Serial.print(sensorValue);
  Serial.print("   Voltage = ");
  Serial.println(voltage);

  delay(100);
}
```

Circuit:


NATIONAL INSTRUMENTS
NI ELVIS
PROTOTYPING BOARD
BANANA A
BANANA B
BANANA C
BANANA D
BNC 1
BNC 2
DIGITAL I/O
DSUB
SHARP EDGES MAY BE PRESENT.
ALWAYS WEAR SAFETY GLASSES.
ELECTROSTATIC SENSITIVE CONNECTORS
POWER SUPPLIES

Intensity

Too Fast

Too Slow

Lowest Acceptable Speed

Highest Acceptable Speed

Experiment Results

Experiment Data:

Input AC Voltage (Vpp)	Output DC Voltage (V)	Intensity LED Observation	Low Threshold LED Observation (yellow)	High Threshold LED Observation (green)	Low Acceptable Speed LED (red)	High Acceptable Speed LED (blue)
1.3	0.02	off	on	off	off	off
2.2	0.07	dim	on	off	off	off
2.5	0.21	brighter	off	off	on	off
3.3	0.42	brighter	off	off	on	off
3.9	0.63	brighter	off	off	off	on
4.7	0.84	bright	off	off	off	on
5.1	1.02	bright	off	on	off	off

Experimental Evaluation:

- The system works as expected
- The full wave rectifier successfully converts AC power to DC power
- According to the data the LEDs include the following when responding to rectified DC power:
    - **Intensity** LED: gets **brighter** as the fan spins **faster**
    - **“Too Slow”** LED: On between **0 - 0.12V**
    - **“Too Fast”** LED: On between > **0.91V**
    - **Low Acceptable Speed** LED: On between **0.12 - 0.50V**
    - **High Acceptable Speed** LED: On between **0.50 - 0.91V**

These adhere to the programed thresholds defined in the circuit description

Lessons Learned

- How to construct a full-wave rectifier to convert AC power to DC power
- Arduino can be used as a logic controller in place of a comparator
- It is important to use the right capacitance to smooth voltage. Without this component the LEDs will flicker as they receive pulsing voltage.
- How to interpret experimental data to create a system that indicates realistic voltage thresholds.
- It is important to ensure that components are working properly before implementing them into the circuit design

Questions

(a) How did you set the configuration of the RC circuit to smooth the rectified signal? Why choose this R-value and C-value? What will happen when the capacitance value drops?

To smooth the rectified signal, we connected the 22 μF capacitor directly across the DC output of the full-wave bridge. The rectifier makes the AC voltage always positive, but coming out of the rectifier the voltage still pulses. The capacitor charges at peak voltage and releases charge when the voltage dips. This makes the voltage to the LEDs smoother, so the LEDs do not flicker. We used a 0.98 kΩ resistor and a 22 μF capacitor. This resistor was small enough to ensure enough current was being received by the LEDs to create effective thresholds for the lights, but also large enough to ensure the fan did not overpower the circuit. The capacitor was large enough to stop the LEDs from flickering and gave the Arduino stable readings, but it wasn't so large that the circuit reacted too slowly when the windmill sped up or slowed down. If the capacitance drops, the voltage can't stay charged between pulses, so the ripple gets much bigger and the LEDs flicker with unstable voltage supply.

(b) What is the range of DC voltage? How did you get this range? Why did you set the potentiometers to these two values? What are the functions of the two voltage values set by potentiometers? (It is not a good answer to say I tuned the values by eyeball experiment. You have to explain how are these two ref_vols related to the range of DC volt input.)

The range of DC voltage is approximately 0-1.4 V. We got this range by measuring the DC voltage produced when the fan was not blowing and when the fan was blowing its fastest. The thresholds include: **0 - 0.12 V → "Too Slow"** LED turns on, **0.12 - 0.5 V → "low acceptable speed"** LED turns on, **0.5 - 0.91 V → "high acceptable speed"** LED turns on, **> 0.91 V → "Too Fast"** LED turns on**.** We got these thresholds by analyzing the total range of the DC voltage and dividing it up into 4 segments that corresponded with the different fan settings. We did not use potentiometers; instead, we programed our threshold values with an arduino.