TurnIt Timer

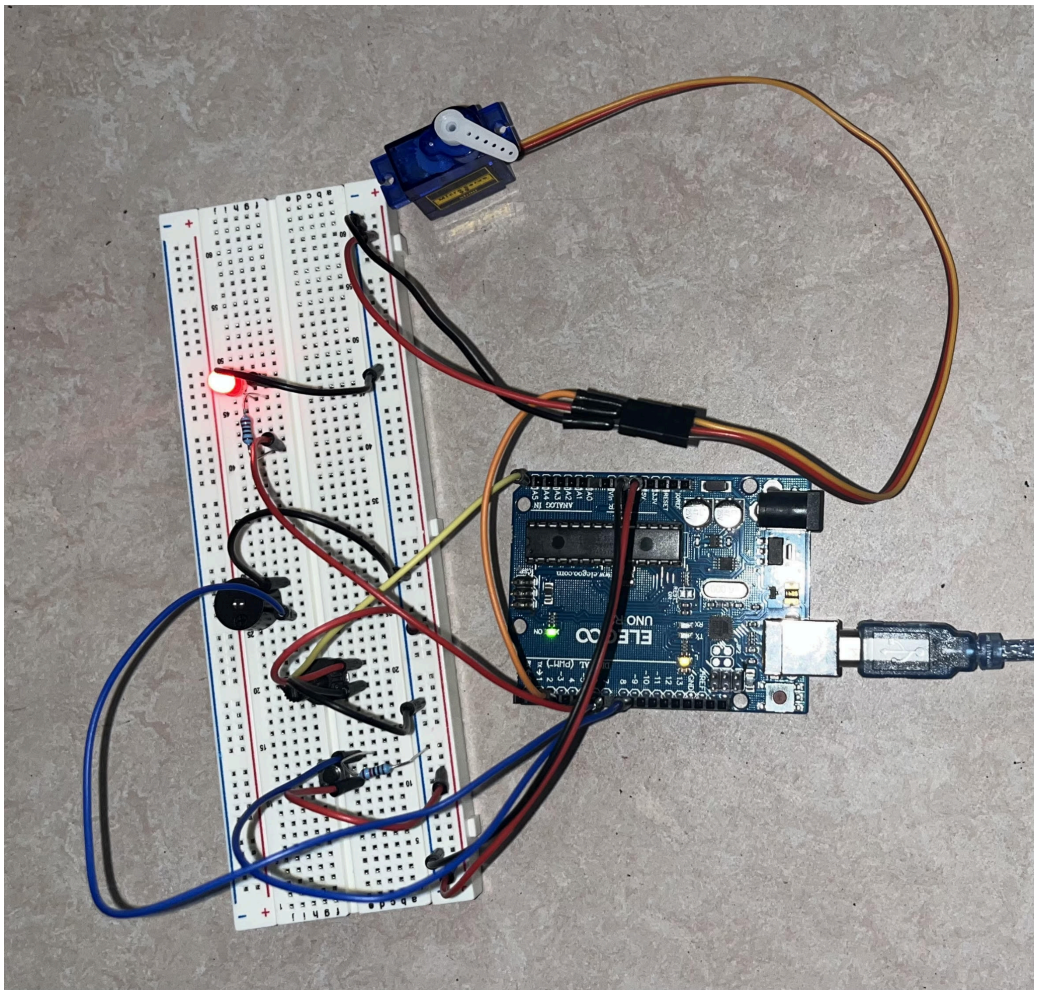
Arduino Design Project

EEL3003 | Spring 2026

Daniel Rosen & Matthew Spillman

University of Florida

Project Overview



Project Description:

The TurnIt Timer is a countdown timer we built on an Arduino Uno. The whole idea is pretty straightforward — you turn the potentiometer dial to pick how long you want the timer to run, hit the button to kick it off, and the servo arm physically sweeps across as the time counts down. Once it hits zero, a red LED turns on and the buzzer goes off.

We built this because it's actually useful in pretty much any situation where you just need a quick timer and don't want to grab your phone. Gym, cooking, study breaks, whatever. The potentiometer lets you dial in the exact duration you want anywhere from 5 to 60 seconds, the servo gives you a live visual so you can see how much time is left at a glance, and the LED and buzzer make sure you don't miss it when time's up. The whole thing runs off the Arduino using five different component types all working together: potentiometer, servo, LED, buzzer, and push button — wired up on a breadboard with clean, well-commented code.

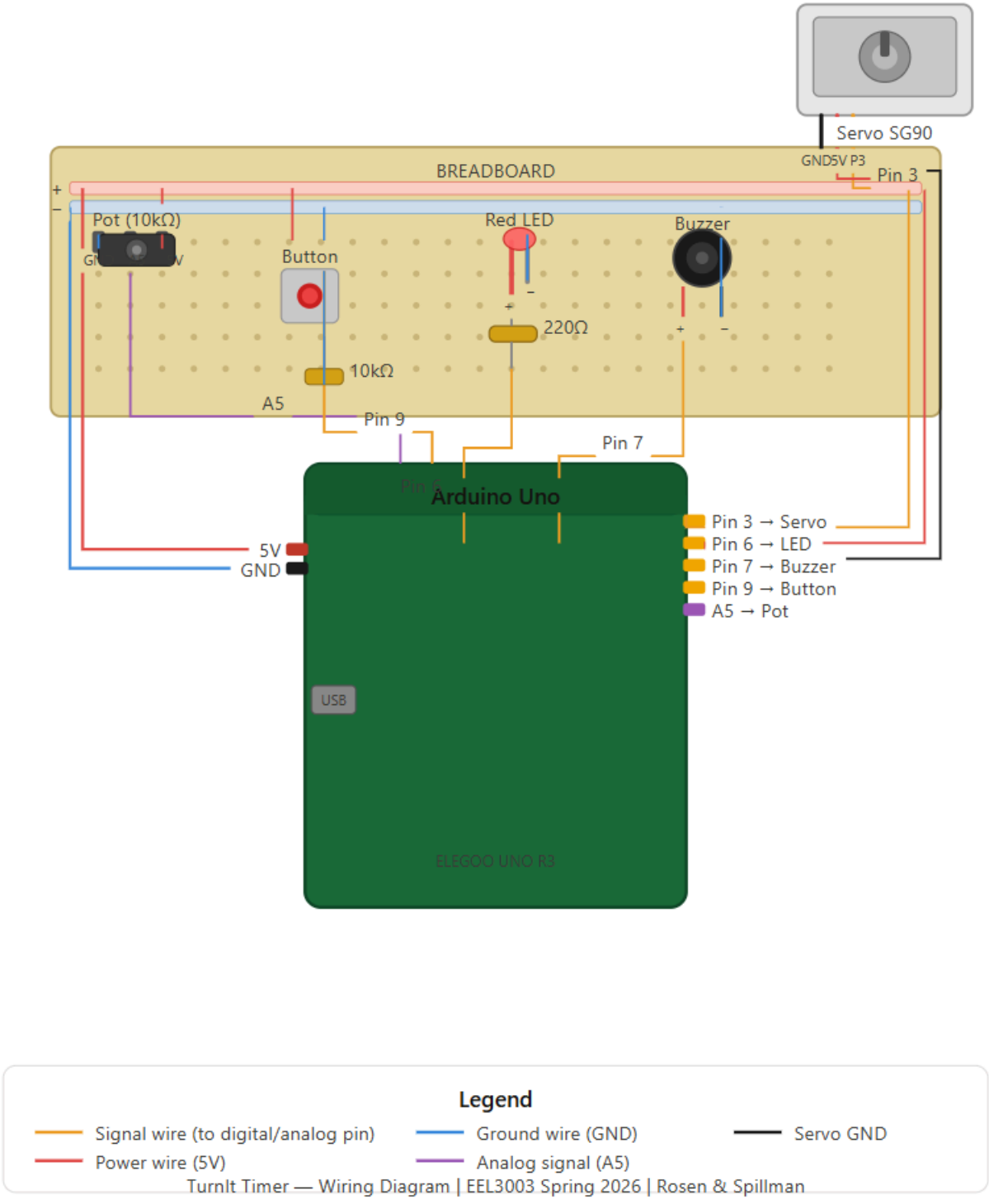
Detailed Parts List

Component	Qty	Notes
Arduino Uno	1	Microcontroller board
Potentiometer (10kΩ)	1	Sets timer duration (analog input)
Servo Motor (SG90)	1	Visual countdown indicator
Red LED	1	Timer done alert
Active Buzzer	1	Audible timer alert
Push Button	1	Starts/resets the timer
220Ω Resistor	1	Current limiting for LED
10kΩ Resistor	1	Pull-down for push button
Breadboard	1	Circuit prototyping
Jumper Wires	~14	Connections between components and/or 5V and ground

All components were sourced from the EEL3003 Arduino kit provided at the start of the semester. No external purchases were required.

Circuit Diagram

The circuit diagram below shows the wiring layout for the TurnIt Timer. It was created using a labeled wiring diagram.



Wiring Summary

- Potentiometer: left pin to GND, right pin to 5V, middle (wiper) pin to A5
- Servo Motor: red to 5V, brown to GND, orange (signal) to Pin 3
- Push Button: one leg to Pin 9, opposite leg to 5V, 10kΩ pull-down resistor from Pin 9 to GND
- Red LED: anode through 220Ω resistor to Pin 6, cathode to GND
- Passive Buzzer: positive leg to Pin 7, negative leg to GND

Arduino Code

The full Arduino code for the TurnIt Timer is provided below. Code structure was adapted from the Elegoo Super Starter Kit Tutorial (Lessons 5, 7, and 9). All timer logic and component integration was developed with assistance from Claude (Anthropic).

```
#include <Servo.h>


// Pin Definitions

// These define which Arduino pins each component is connected to

#define POT_PIN    A5  // Potentiometer wiper connected to analog pin A5

#define BUTTON_PIN 9   // Push button connected to digital pin 9

#define SERVO_PIN  3   // Servo signal wire connected to digital pin 3

#define LED_PIN    6   // Red LED connected to digital pin 6

#define BUZZER_PIN 7   // Passive buzzer connected to digital pin 7


Servo myServo; // Create a Servo object to control the servo motor


// Timer State Variables

bool running = false;          // Tracks whether the timer is currently active

unsigned long startTime = 0;    // Stores the time in ms when the timer started

unsigned long duration = 0;     // Stores the selected timer duration in ms


void setup() {

  myServo.attach(SERVO_PIN);        // Attach servo to its signal pin

  pinMode(BUTTON_PIN, INPUT);       // Set button pin as input

  pinMode(LED_PIN, OUTPUT);         // Set LED pin as output
```

```
  pinMode(BUZZER_PIN, OUTPUT);      // Set buzzer pin as output

  // Initialize all outputs off at startup
  myServo.write(0);                 // Servo starts at 0 degrees
  digitalWrite(LED_PIN, LOW);       // LED off
  noTone(BUZZER_PIN);               // Buzzer off
}

void loop() {

  // Check if button is pressed
  if (digitalRead(BUTTON_PIN) == HIGH) {
    delay(50); // Debounce delay to avoid false triggers
    if (digitalRead(BUTTON_PIN) == HIGH) {

      if (!running) {
        // Start the timer
        // Read potentiometer value (0 to 1023)
        int potVal = analogRead(POT_PIN);

        // Map potentiometer reading to a duration between 5 and 60 seconds
        duration = map(potVal, 0, 1023, 5000, 60000); // in milliseconds

        startTime = millis(); // Record the start time
```

```
      running = true;       // Set timer to active

      // Reset outputs before starting
      digitalWrite(LED_PIN, LOW);
      noTone(BUZZER_PIN);
      myServo.write(0);

    } else {
      // Reset the timer if already running
      running = false;
      myServo.write(0);
      digitalWrite(LED_PIN, LOW);
      noTone(BUZZER_PIN);
    }

    // Wait for button to be released before continuing
    while (digitalRead(BUTTON_PIN) == HIGH) delay(10);
  }
}

// Timer countdown logic - only runs when timer is active
if (running) {
  unsigned long elapsed = millis() - startTime; // How much time has passed
```

```
    if (elapsed >= duration) {
      // Timer is done
      running = false;
      myServo.write(180);          // Servo sweeps to 180 degrees
      digitalWrite(LED_PIN, HIGH); // Turn on red LED

      // Buzz 3 times to signal timer complete
      for (int i = 0; i < 3; i++) {
        tone(BUZZER_PIN, 1000, 300); // 1000Hz tone for 300ms
        delay(500);                  // Wait 500ms between beeps
      }

    } else {
      // Map elapsed time to servo angle between 0 and 180 degrees
      // Servo sweeps as countdown progresses
      int angle = map(elapsed, 0, duration, 0, 180);
      myServo.write(angle); // Update servo position
    }
  }
}
```

References & Troubleshooting Log

References

1. Arduino IDE: https://www.arduino.cc/en/software
2. Elegoo Super Starter Kit for UNO V1.0 Tutorial — Lesson 5 (Digital Inputs), Lesson 7 (Passive Buzzer), Lesson 9 (Servo): https://www.elegoo.com
3. Code integration, timer logic, and component coordination developed with assistance from Claude (Anthropic): https://claude.ai

Troubleshooting Log

Challenge	What We Tried	How We Fixed It
Servo was jittering even when timer wasn't running	Checked wiring and power connections	Added delay() debounce on the button and confirmed servo only updates when timerRunning is true
Button registering multiple presses on a single click	Tried adjusting sensitivity	Added a 50ms debounce delay and a while loop to wait for button release before proceeding
Potentiometer range felt too large	Tested various ranges	Narrowed map() range to 5-60 seconds so the dial feels responsive and practical

Our biggest problem in the creation of this was that there was no indication that our code/breadboard was working through several iterations and edits. To get a breadboard that worked, we decided to start over, adding each component one at a time, testing using simple code that tested each component as we added it. Once we added all of the components used in our design, we went back to our original code and the circuit started to work as intended.