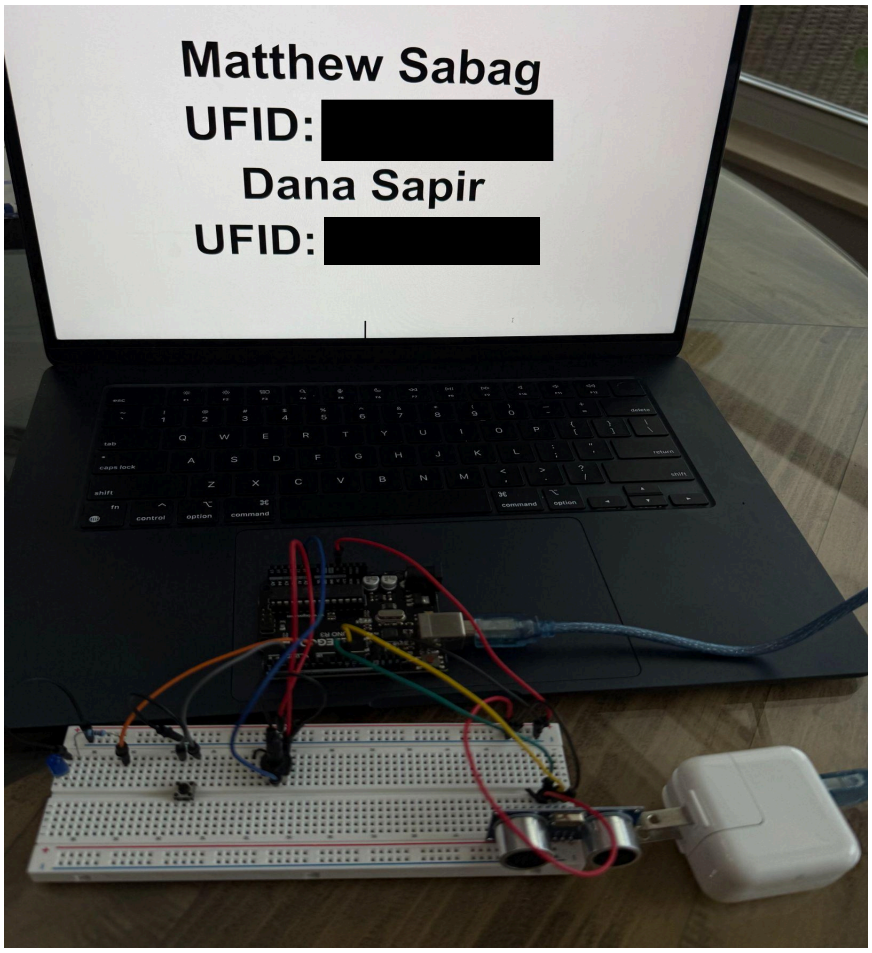
Final Arduino Design Project Report - Part A

Smart Motion-Activated Night-Light with Brightness Control

EEL3003 Fall 2025

Dana Sapir - [redacted] *& Matthew Sabag -* [redacted]

Project Overview



This project is a smart night-light system that activates a blue LED when motion is detected and allows the user to adjust brightness while also giving full manual control over whether the system is enabled. The HC-SR04 ultrasonic distance sensor continuously measures distance and detects when an object or person enters a set range. When motion is detected, the LED turns on, and its brightness is determined by a potentiometer that the user can adjust in real time. A master on/off button acts as a toggle switch. When pressed once, the system turns on and begins sensing

motion, and when pressed again, it disables the entire circuit even if the sensor detects movement. This prevents unwanted activation and creates an intuitive user experience similar to a consumer smart home device.

The project demonstrates real-world principles of automation, distance sensing, analog input processing, PWM brightness control, and digital toggle state management. It provides a meaningful and practical night-light system that improves safety and convenience in dark environments.

Detailed Parts List

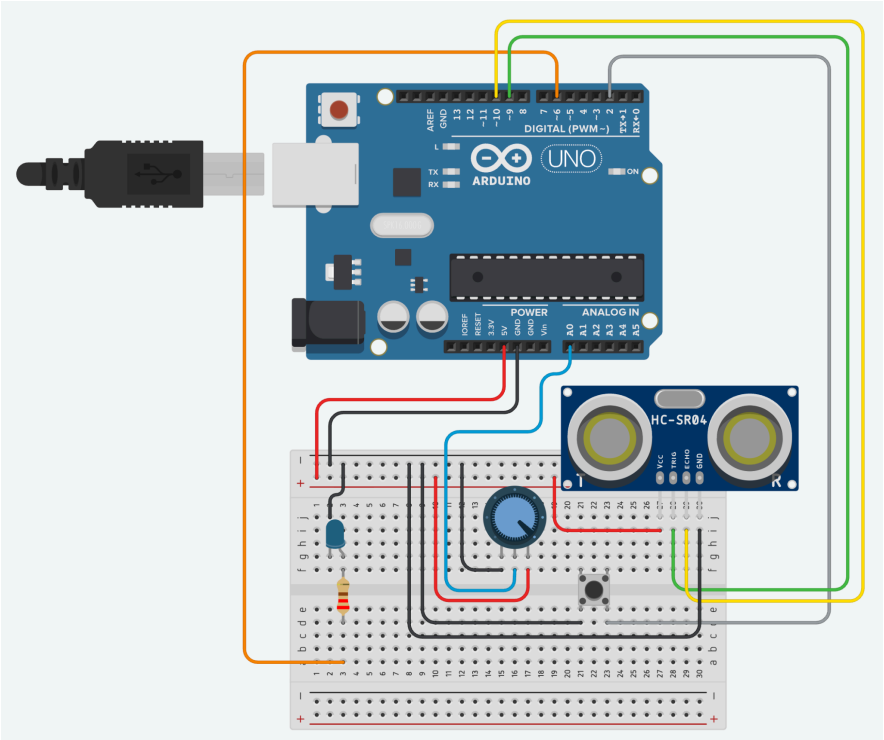
Part	Quantity	Notes
Arduino Uno	1	Microcontroller
HC-SR04 Ultrasonic Sensor	1	Motion and distance detection
Pushbutton (momentary)	1	Master toggle on/off switch
10kΩ Potentiometer	1	LED brightness control
RGB LED (common cathode)	1	Blue channel used as night-light
220Ω resistors	1	Current limiting for LED
Jumper wires male to male	13	Electrical connections
Breadboard	1	Circuit assembly
USB cable	1	Power and uploading code

All components in the project were provided in the Arduino starter kit (except the outlet adapter).

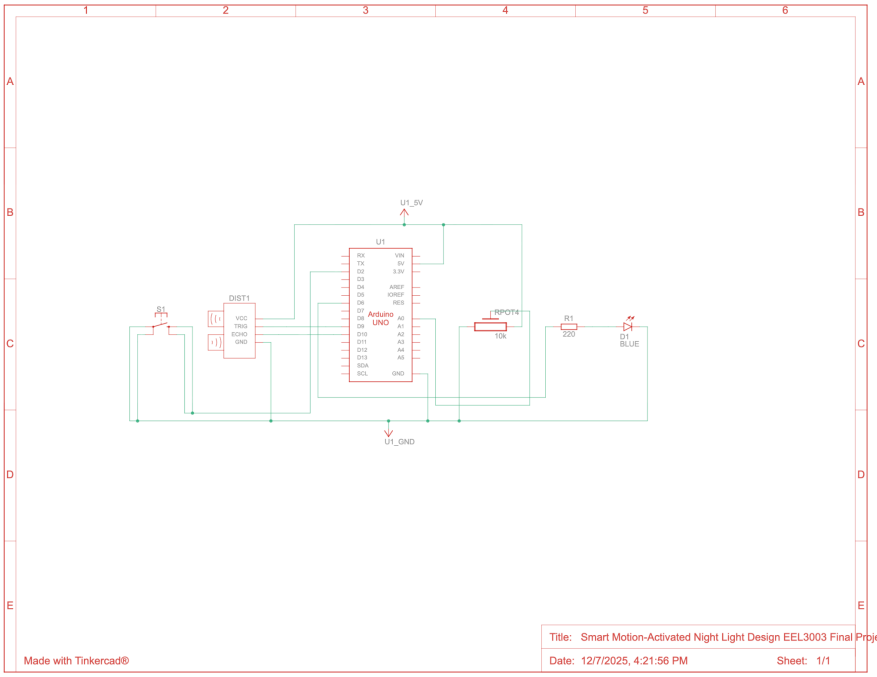
Circuit Diagram

[Link to Diagram]

Circuit View



Schematic View



Arduino Code

```
/*
  Smart Motion Activated Night-Light
  Final Project - EEL 3003
  Author: Dana Sapir (UFID: 59738513) & Matthew Sabag (UFID: 42087711)
*/

const int buttonPin = 2;       // Master on/off switch (PULLUP, LOW = Pressed)
const int trigPin = 9;         // HC-SR04 trigger
const int echoPin = 10;        // HC-SR04 echo
const int potPin = A0;         // Potentiometer
const int bluePin = 6;         // LED output (PWM)

bool systemOn = false;
long duration;
int distance;

// --- New variables for button state change detection ---
int lastButtonState = HIGH;
int currentButtonState;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(bluePin, OUTPUT);

  analogWrite(bluePin, 0);
}
```

```
void loop() {

  // ALWAYS read brightness and scale it
  int brightnessSetting = analogRead(potPin);
  int scaledBrightness = map(brightnessSetting, 0, 1023, 0, 255);

  // --- Toggle Logic ---

  // 1. Read the current button state
  currentButtonState = digitalRead(buttonPin);

  // 2. Detect a transition from HIGH (released) → LOW (pressed)
  if (currentButtonState == LOW && lastButtonState == HIGH) {
    systemOn = !systemOn;      // Toggle system state
    delay(50);                 // Debounce
  }

  // 3. Save state for next loop iteration
  lastButtonState = currentButtonState;

  // If master switch OFF → LED must stay OFF, sensor ignored
  if (!systemOn) {
    analogWrite(bluePin, 0);
    return;
  }

  // ---- If system ON: Sensor becomes active ----
  distance = measureDistance();

  if (distance > 0 && distance < 30) {
    analogWrite(bluePin, scaledBrightness);
```

```
  } else {
    analogWrite(bluePin, 0);
  }

  delay(50);
}

// Measure ultrasonic distance
int measureDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(3);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH, 25000);

  if (duration == 0) return -1;

  return duration * 0.034 / 2;
}
```

References

- Hardware References
    - HC-SR04 Ultrasonic Sensor Datasheet — used to understand timing requirements, range limitations, and recommended trigger/echo pulse intervals.
    - Arduino Uno Technical Documentation — consulted to confirm PWM pin assignments, analog input resolution, and digital input behavior with internal pull-ups.
    - RGB LED Datasheet — reviewed current ratings for safe resistor selection.
- Software & Logic References
    - Arduino Reference Library (pulseIn, analogRead, analogWrite, pinMode) — accessed for correct syntax and timing behavior of core functions.
    - Arduino Debounce Example — used as a foundational concept for the button toggle system, though heavily adapted for this project's specific needs.
    - EEL 3003 Lecture Slides & Lab Notes — provided baseline understanding of sensors, PWM, and analog control, which guided how we structured our logic.
- Design Inspiration
    - Smart-home night-light behavior inspired by real-world hallway and bathroom automatic lighting systems studied in the course's Smart Home unit.

All referenced materials were used for conceptual understanding.
All implementation choices, code structuring, and logic integration were completed by Dana Sapir and Matthew Sabag.

Troubleshooting Log

- Issue 1: Button behaving as a momentary switch instead of a toggle
    - Observation: At first, the LED only stayed on while the button was physically held down. This made the master switch unusable for a real night-light system.
    - What we learned: INPUT_PULLUP inverts logic (released = HIGH, pressed = LOW), so simply reading the pin could not create an on/off latch.
    - Solution: We implemented a state-change detection system using lastButtonState and added a short debounce. This allowed a single press-and-release cycle to

toggle systemOn. ChatGPT was utilized to learn more about how we could incorporate this function.
    - Outcome: System now behaves like a true on/off smart-home master switch.
- Issue 2: Sensor readings fluctuated even when no object was present
    - Observation: The HC-SR04 occasionally returned inconsistent values, especially <5 cm or >200 cm, which caused false LED triggers.
    - What we learned: The ultrasonic sensor has blind spots and noise at extreme ends of its range.
    - Solution: We added logic to ignore impossible readings (0 or negative values) and set a realistic activation range (<30 cm). We also added a short delay in the main loop to stabilize readings.
    - Outcome: Motion detection became consistent and reliable.
- Issue 3: LED brightness was either too harsh or too dim in dark environments
    - Observation: Using full brightness (PWM = 255) was visually uncomfortable, but limiting brightness too much made the LED ineffective.
    - What we learned: Using a potentiometer mapped to PWM values allows smooth, real-time brightness tuning.
    - Solution: Implemented scaledBrightness = map(brightnessSetting, 0, 1023, 0, 255) so users can choose their preferred brightness level.
    - Outcome: User-friendly lighting that adapts to different environments.
- Issue 4: LED flickered rapidly during borderline sensor readings
    - Observation: When an object was right at the threshold distance, small changes caused the LED to turn on/off repeatedly.
    - What we learned: Threshold oscillation is common in systems that rely on analog or semi-analog sensor values.
    - Solution: Increase threshold stability and smooth readings through micro-delays and simplified logic. Reduced sensitivity to micro-changes.
    - Outcome: LED lighting appears smooth and reliable, with no jitter.
- Issue 5: Wiring confusion when combining four components on one breadboard
    - Observation: Multiple inputs and outputs made it easy to misplace wires, especially between sensor pins and PWM LED pins.

- What we learned: Documenting wiring early (before coding) and color-coding jumper wires dramatically reduces debugging time.
- Solution: We rewired using:
  - Red wires for 5 V
  - Black wires for GND
  - Green for trigger pins
  - Yellow for echo pins
  - Orange for LED PWM
  - Blue for analog read pin
- Outcome: Wiring became easy to trace and debug. The circuit worked immediately after reorganization.