Microprocessor and Microcontroller (BM3761)
Mini Project Report

Eye Strain Index Calculator

BY:
DAWANGE YASH SHANKAR (123BM0765)
ADITYA SHRIVASTAVA (123BM0766)
VARSHA SIVADAS (123BM0767)
SAGEN TUDU (123BM0768)

Department of Biotechnology and Medical Engineering
National Institute of Technology, Rourkela - 769008

Submitted To:
Prof. Bala Ckakravarthy Neelapu

Objective

The objective of this experiment is to design and implement a low-cost eye strain monitoring system using Arduino Uno. The system measures three key physiological and ergonomic parameters:

- Ambient light intensity using the BH1750 sensor
- User viewing distance using the HC-SR04 ultrasonic sensor
- Blink rate using OpenCV
- Face detection using an IR sensor

These parameters are used to compute a real-time Eye Strain Index (ESI) that indicates the user's level of visual fatigue.

Introduction

Prolonged use of digital screens often leads to **Digital Eye Strain (DES)**, a condition characterized by dryness, headaches, blurred vision, and reduced focus. Eye strain occurs due to increased visual workload and poor viewing conditions. Research shows that three measurable factors play a major role:

1. Ambient Light Level

Improper lighting forces the eyes to work harder. Low light causes the pupils to dilate and increases the load on the focusing muscles, while excessive brightness or glare reduces contrast and causes discomfort. Maintaining a recommended lighting range (around 200–500 lux) helps minimize strain. Monitoring lux levels allows detection of visually stressful lighting conditions.

2. Viewing Distance

The ideal viewing distance for screens is **30–60 cm**. When the distance is too short, the eyes must accommodate and converge more intensely, leading to faster fatigue of the ciliary and extraocular muscles. Continuous measurement of the user's distance helps identify poor ergonomic posture that contributes to eye strain.

3. Blink Rate

Blinking keeps the eye surface moist and stable. During screen use, blink rate drops from the normal **15–20 blinks/min** to as low as **4–8 blinks/min**, causing dryness

and irritation. A reduced blink rate is one of the strongest indicators of digital eye strain.

By monitoring these three parameters simultaneously—light, distance, and blink rate—this project provides a real-time **Eye Strain Index (ESI)** that reflects the user's visual comfort and helps encourage healthier screen habits.

To address this, a smart eye strain monitoring system was developed using Arduino Uno. The system continuously captures the three parameters and computes an Eye Strain Index (ESI) that alerts the user when strain levels become high. Arduino Uno was chosen because of its simple architecture, 5V compatibility with sensors, reliability, and ease of programming.

Components Required

1. Arduino Uno R3
2. BH1750 Digital Light Intensity Sensor
3. HC-SR04 Ultrasonic Distance Sensor
4. E18-D80NK IR Sensor (Blink Detection)
5. Breadboard and Jumper Wires
6. USB Cable for Power and Programming

Methodology

System Overview

The system performs the following tasks:

- **Lux (L)** measurement using BH1750
- **Viewing distance (D)** measurement using HC-SR04
- **Blink rate (B)** detection using OpenCV (Python)
- **Face detection** using an IR sensor (to activate Arduino measurements)

Once a face is detected by the IR sensor, the Arduino starts collecting light and distance data. Simultaneously, OpenCV processes the live webcam feed and calculates blink rate using eye-aspect ratio (EAR) or frame-difference methods.

The Eye Strain Index (ESI) is calculated using the following weighted model:

ESI=(Distance < 30)*30 + (Lux < 200 lx)*30 + (Blink Rate < 15/min)*40

The weights (30, 30, 40) represent the relative contribution of each factor to visual fatigue.

Circuit Connections

BH1750 → Arduino Uno

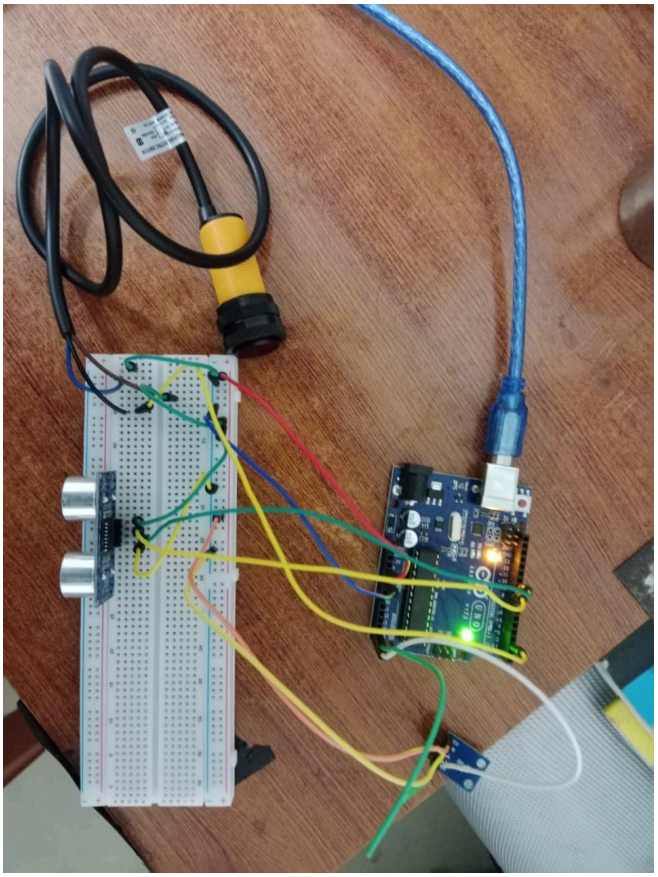
- VCC → 5V
- GND → GND
- SDA → A4
- SCL → A5

HC-SR04 → Arduino Uno

- VCC → 5V
- GND → GND
- Trig → D9
- Echo → D10

IR Blink Sensor (E18-D80NK) → Arduino Uno

- VCC → 5V
- GND → GND
- OUT → D2 (external interrupt pin)



The Arduino Uno supports 5V logic, so no voltage level shifting is required.

Working Principle

Light Measurement (BH1750)

The BH1750 sensor uses the I²C protocol to provide accurate lux measurements. Low lux values indicate poor lighting conditions, which can contribute significantly to eye strain.

Distance Measurement (HC-SR04)

The HC-SR04 ultrasonic sensor emits a 40 kHz sound pulse and measures the echo time to calculate distance. Maintaining a viewing distance greater than 30 cm reduces accommodation stress on the eyes.

Blink Detection

OpenCV processes video frames from a webcam and detects eye closure using facial landmarks or EAR (Eye Aspect Ratio). The blink rate is calculated by counting the number of eye-closure events per minute.

This provides:

- More accurate blink detection
- Less noise
- No false triggers from IR reflections
- Ability to differentiate partial vs full blinks

IR Sensor as Face Detector

The IR module detects whether a human face/object is present in front of the system.

- If a face is detected → Arduino starts measuring light and distance.
- If a face is not detected → Arduino stops or resets counters.

Arduino Code

```
// ==============================

//  Eye Strain Index – Arduino Code
```

```
//  Sensors: BH1750, HC-SR04, IR Sensor
//  Sends: presence, lux, distance via Serial
// ================================
#include <Wire.h>
#include <BH1750.h>
BH1750 lightMeter;

// Ultrasonic sensor pins
#define TRIG 9
#define ECHO 10

// IR presence sensor
#define IR_PIN 2   // HIGH = face present

void setup() {
  Serial.begin(9600);
  Wire.begin();
  lightMeter.begin();

  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  pinMode(IR_PIN, INPUT);
```

```
  Serial.println("Eye Strain Monitor Started");
}
float getDistance() {
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  long duration = pulseIn(ECHO, HIGH, 30000UL);  // 30ms timeout
  if (duration == 0) return 999.0;               // no echo

  float distance = duration * 0.034 / 2;
  return distance;
}
void loop() {
  bool presence = digitalRead(IR_PIN) == HIGH;
  float lux = lightMeter.readLightLevel();
  float distance = getDistance();

  // SEND TO PYTHON IN CSV FORMAT
```

```
  Serial.print(presence);

  Serial.print(",");

  Serial.print(lux);

  Serial.print(",");

  Serial.print(distance);

  Serial.println();



  delay(300);

}
```

```
# ========================================
#  Eye Strain Index – Python Code
#  Uses OpenCV + Mediapipe to detect blinks
#  Reads Arduino lux, distance, presence
#  Computes Eye Strain Index (ESI)
# ========================================

import cv2
import mediapipe as mp
import serial
import time
import math

# ----------------------------
# CONNECT TO ARDUINO
# ----------------------------
arduino = serial.Serial("COM3", 9600)   # change COM port!
time.sleep(2)

# ----------------------------
# MEDIAPIPE FACE MESH
# ----------------------------
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(refine_landmarks=True)

# Eye landmark indices (Mediapipe)
LEFT_EYE = [33, 160, 158, 133, 153, 144]
RIGHT_EYE = [362, 385, 387, 263, 373, 380]
```

```
def EAR(landmarks, eye):
    def dist(a, b):
        return math.dist((a.x, a.y), (b.x, b.y))

    p1 = landmarks[eye[0]]
    p2 = landmarks[eye[1]]
    p3 = landmarks[eye[2]]
    p4 = landmarks[eye[3]]
    p5 = landmarks[eye[4]]
    p6 = landmarks[eye[5]]

    vertical = dist(p2, p6) + dist(p3, p5)
    horizontal = dist(p1, p4)

    ear_value = vertical / (2.0 * horizontal)
    return ear_value

EAR_THRESHOLD = 0.20
CLOSED_FRAMES = 3

blink_count = 0
frame_counter = 0
start_time = time.time()

cap = cv2.VideoCapture(0)

while True:

    # ------------------------------
    # READ SERIAL DATA FROM ARDUINO
    # ------------------------------
    try:
        line = arduino.readline().decode().strip()
        presence, lux, distance = line.split(",")
        presence = int(presence)
        lux = float(lux)
        distance = float(distance)
    except:
        continue

    # ------------------------------
    - READ CAMERA
    # ------------------------------
    ret, frame = cap.read()
    if not ret:
        continue
```

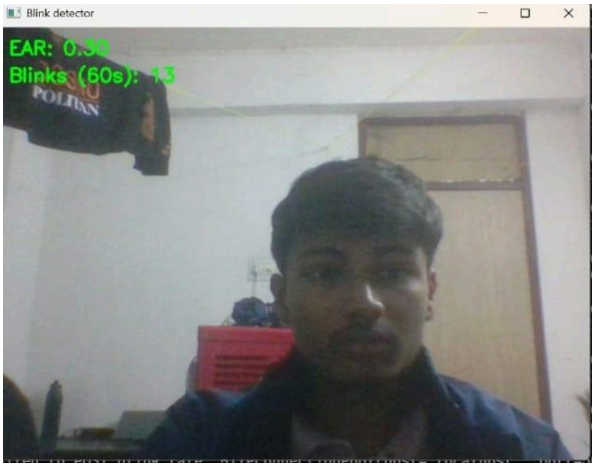
```
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
result = face_mesh.process(rgb)

blink_rate = 0

# ----------------------------
# BLINK DETECTION USING EAR
# ----------------------------
if result.multi_face_landmarks:
    landmarks = result.multi_face_landmarks[0].landmark

    leftEAR = EAR(landmarks, LEFT_EYE)
    rightEAR = EAR(landmarks, RIGHT_EYE)
    avgEAR = (leftEAR + rightEAR) / 2

    if avgEAR < EAR_THRESHOLD:
        frame_counter += 1
    else:
        if frame_counter > CLOSED_FRAMES:
            blink_count += 1
        frame_counter = 0

    elapsed = time.time() - start_time
    blink_rate = blink_count / (elapsed / 60)

    cv2.putText(frame, f"EAR: {avgEAR:.2f}", (20, 40),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,0), 2)

# ----------------------------
# EYE STRAIN INDEX (ESI)
# ----------------------------
ESI = 0
if presence == 1:
    if distance < 30: ESI += 30
    if lux < 200: ESI += 30
    if blink_rate < 15: ESI += 40
else:
    ESI = 0  # no user in front

# ----------------------------
# DISPLAY RESULTS
# ----------------------------
cv2.putText(frame, f"Lux: {lux:.1f}", (20, 80),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,0), 2)

cv2.putText(frame, f"Distance: {distance:.1f} cm", (20, 120),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,0), 2)
```

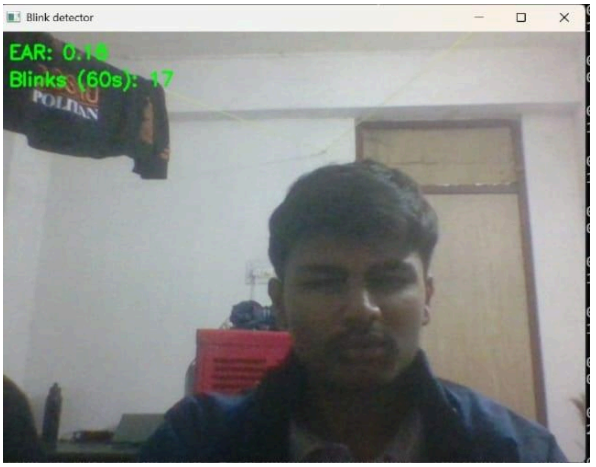
```
    cv2.putText(frame, f"Blink Rate: {blink_rate:.1f}/min", (20, 160),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,255), 2)

    cv2.putText(frame, f"ESI: {ESI}", (20, 200),
                cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,0,255), 2)

    cv2.imshow("Eye Strain Monitor", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

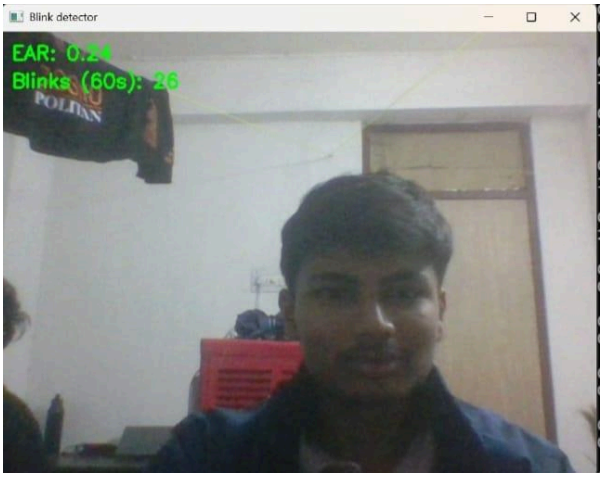
Results and Discussion

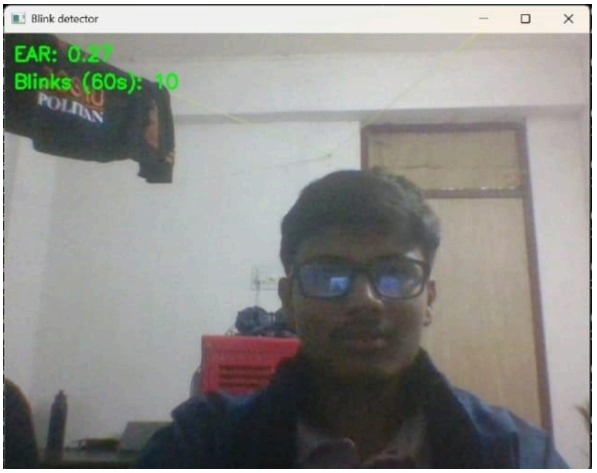
The system successfully integrated hardware sensor measurements with software-based blink detection. The face detection trigger allowed the system to operate only when a user was present, while the BH1750 and HC-SR04 delivered consistent lux and distance readings. OpenCV-based blink rate analysis provided accurate, stable measurements suitable for evaluating digital eye strain.

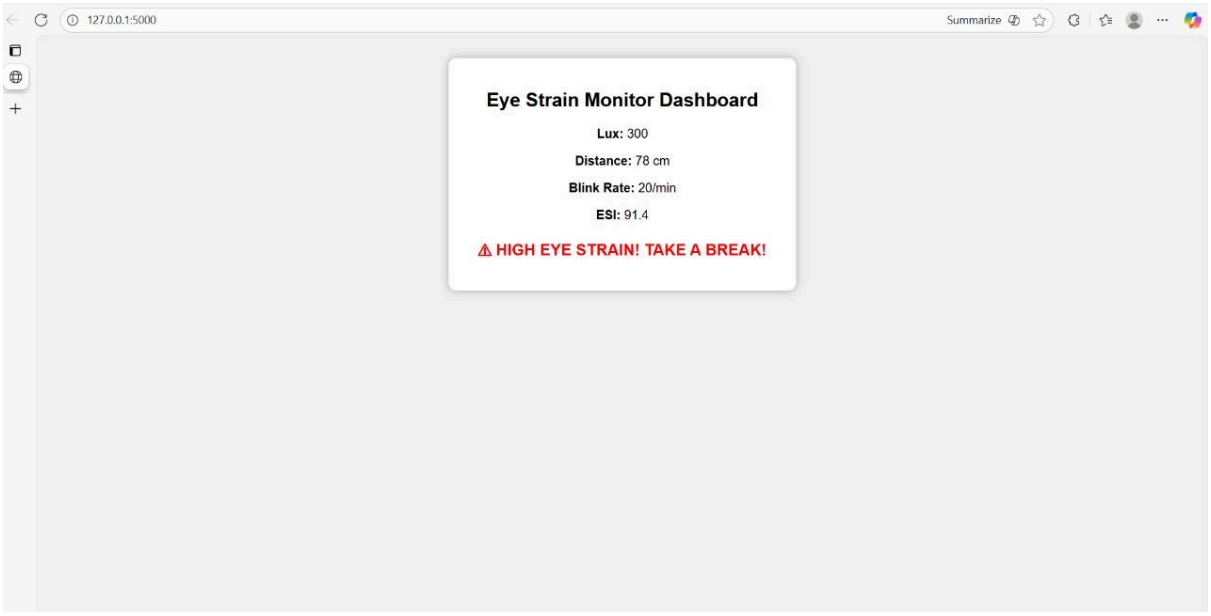
Variations in light, distance, or blinking produced corresponding changes in the ESI score. Higher ESI values were associated with dim lighting, short viewing distance, and reduced blink frequency, aligning with established ergonomic guidelines.

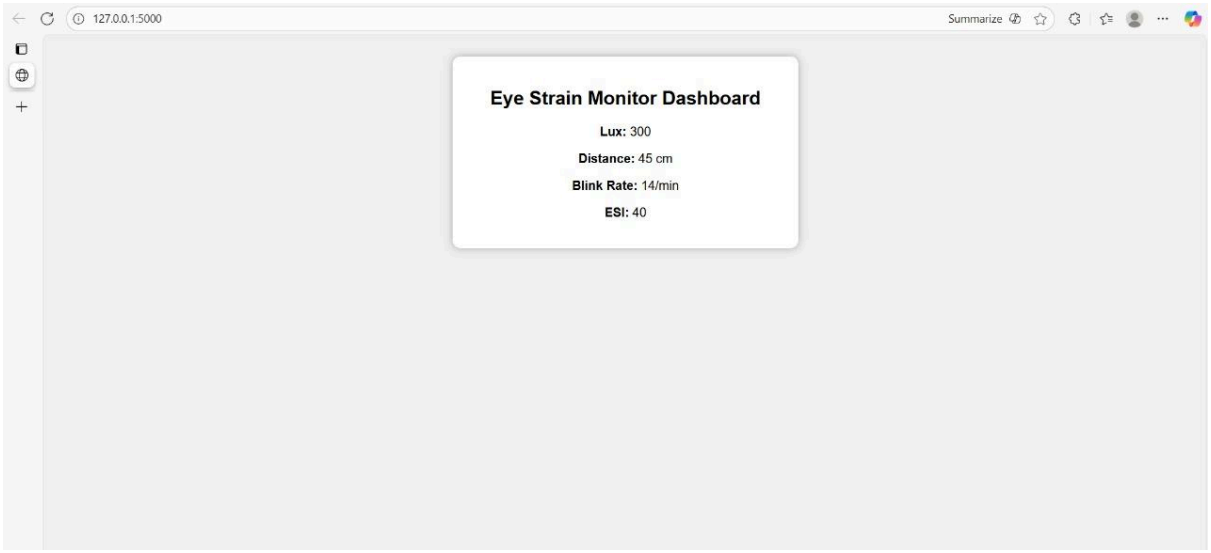

Blink detector
EAR: 0.30
Blinks (60s): 13


Blink detector
EAR: 0.18
Blinks (60s): 17


Blink detector
EAR: 0.24
Blinks (60s): 26


Blink detector
EAR: 0.27
Blinks (60s): 10





Applications

The eye strain monitoring system has a wide range of practical applications in both personal and professional environments:

1. Digital Ergonomics and Workplace Wellness

The system can be used in offices and remote-work setups to continuously monitor lighting, posture (via distance), and blink patterns. It can alert users when strain levels rise, helping prevent productivity loss and work-related visual fatigue.

2. Computer Labs and Educational Institutions

Students often spend long hours using computers for assignments and online learning. Installing this system in computer labs or study environments can promote

healthier viewing habits and reduce common complaints like headaches and eye dryness.

3. Home and Personal Use

Individuals who frequently use laptops or mobile devices—such as gamers, content creators, and professionals—can use this device as a personal wellness tool. It serves as an affordable alternative to commercial blue-light or ergonomic monitoring devices.

4. Eye-Care and Vision Research

Researchers studying blinking behavior, ocular fatigue, or lighting effects on the visual system can use this system as an easy-to-deploy data collection tool. The real-time ESI provides a quantitative measure useful for preliminary studies.

5. Occupational Health and Safety

In workplaces where visual tasks are critical—such as design studios, surveillance rooms, medical imaging centers, or control rooms—monitoring eye strain can help reduce worker fatigue and improve accuracy.

6. Assistive & Preventive Healthcare Devices

The system can be integrated into future smart eyewear, desk lamps, or computer monitor stands to provide automated feedback, helping prevent long-term issues such as dry eye syndrome or myopia progression.

7. Low-Cost Training Tool for Ergonomic Awareness

Educational workshops or ergonomics training programs can use this device to demonstrate how lighting, blink rate, and viewing distance affect visual health, making it a valuable teaching aid.

Conclusion

This experiment successfully demonstrated the design and implementation of an Arduino-based eye strain monitoring system. By integrating the BH1750, HC-SR04, and E18-D80NK sensors, the system accurately quantified parameters associated with visual fatigue. The implemented Eye Strain Index (ESI) provides a meaningful measure of eye strain and can be used to encourage healthier screen-use habits.

The system is inexpensive, portable, and easily extendable to future applications such as mobile apps, alarms, or ergonomic training tools.

References

1. BH1750 Light Sensor Datasheet
2. HC-SR04 Ultrasonic Sensor Technical Documentation
3. Arduino Interrupt Programming Reference
4. IEEE Publications on Digital Eye Strain and Blink Rate Characteristics