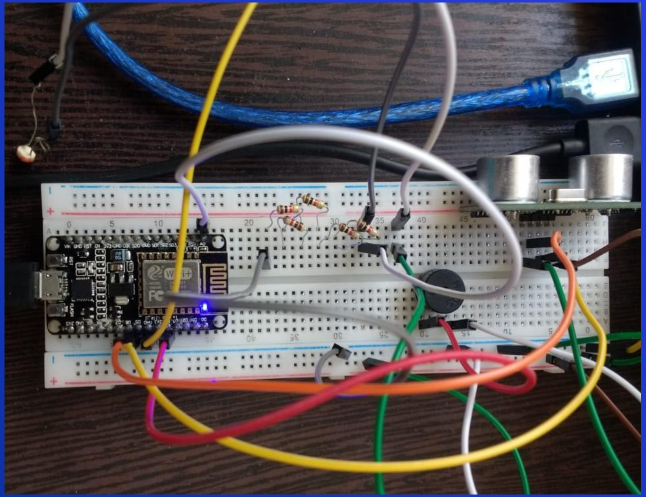
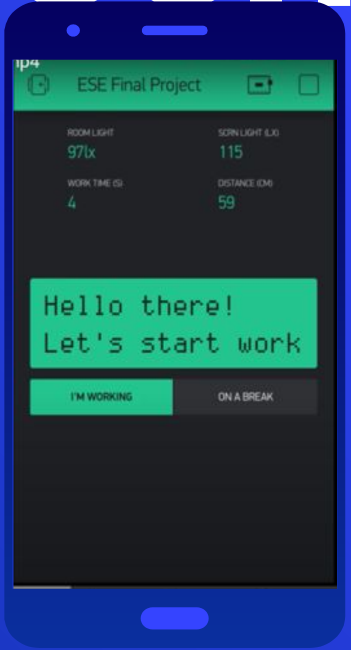
ESE 111 Final Project Group 13

Racheal and Vaishnavi

This handy doctor will make sure you're using your screen right!

Dr. Screen





ESE Final Project

ROOM LIGHT
97lx

SCRN LIGHT (LX)
115

WORK TIME (S)
4

DISTANCE (CM)
59

Hello there!
Let's start work

The Virtual Workplace

The shift to online work and learning environments has dramatically increased **screen usage**

With this comes a **greater risk of health issues** associated to continuous exposure to **blue light**

Dr. Screen - The Virtual Doctor

Your doctor is here! This handy device ensures users' screen usage is healthy as per generally health practices for online screen usage.

Healthy screen usage is measured through

- Distance between user's eye level and screen
- Time spent working in one stretch of time
- Relative light intensity of screen and environment

Parameters

Time spent working in one stretch

Prescription: It's recommended the user takes a break every **20 minutes**

Modification: Set a threshold for working as **2 minutes**. Display time intervals in seconds (1 interval = 2s)

Computation: Timer written in code

Distance

Prescription: It's recommended the distance between the user's eye level and screen is **50 to 100 cm**

Modification: None

Computation: Ping Sensor

Component	I/O/ Device
Ping Sensor	**Input**
Photoresistor	**Input**
Virtual Switch	**Device**
Buzzer	**Output**
Virtual LCD	**Output**

Parameters

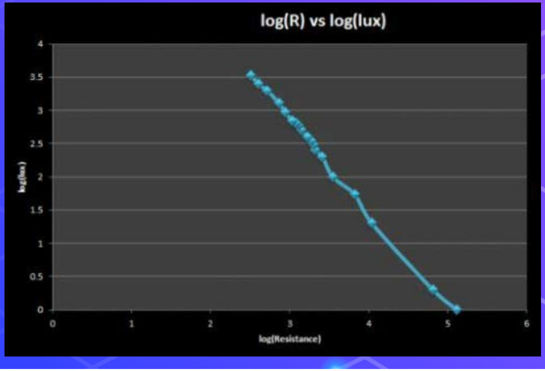
Relative light Intensity

Prescription: It’s recommended the screen’s light intensity **matches** the light intensity from the environment.

Modification: Based on experimentation with our device, we determined an error factor of **75 lx**. Setup adapted to that of exp.

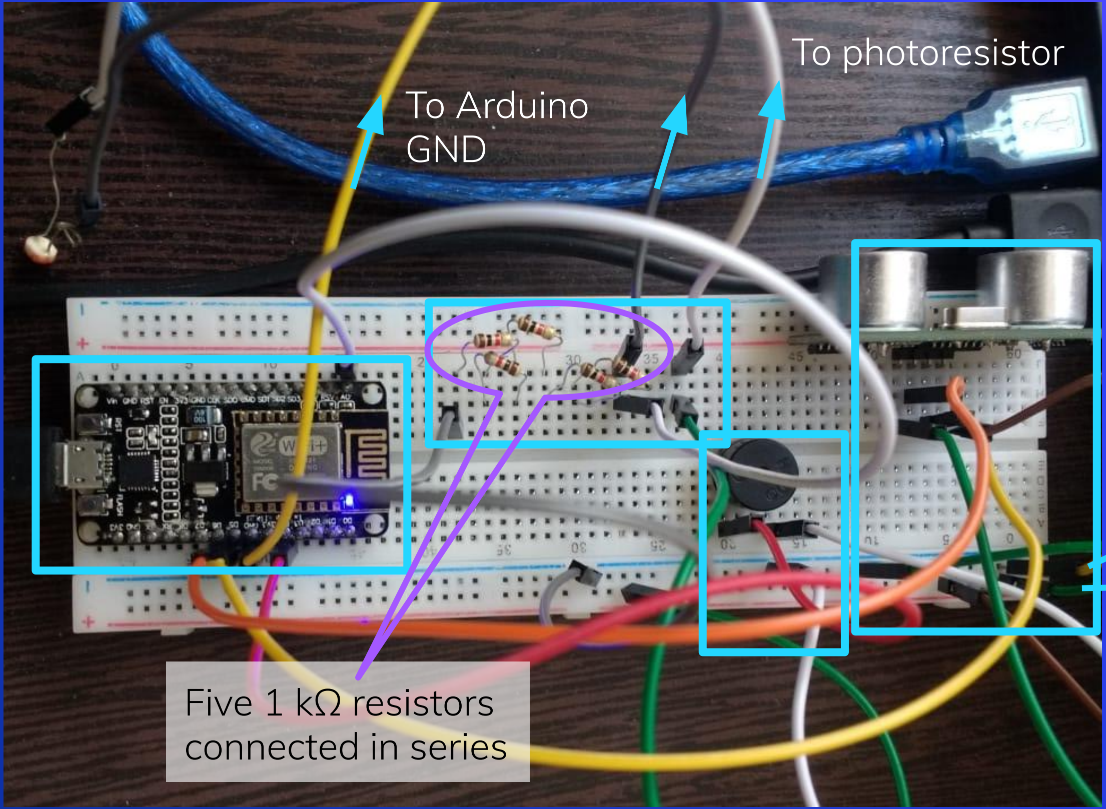
Computation:

1. Read analog value from photoresistor. This measures the voltage drop across the natural resistor connected to it
2. Convert voltage drop across natural resistor to voltage drop across photoresistor by considering total voltage supplied (5V)
3. Convert voltage across photoresistor to photoresistance using formula for resistors in series (natural resistor, photoresistor)
4. Convert photoresistance to light intensity using derived formula from an external experiment with 5V supply and 5kΩ resistor



$$log_{10}(lux) = -1.4 \times log_{10}\,(R) + 7.098$$

Circuit Schematics



To Arduino GND

To photoresistor

Five 1 kΩ resistors connected in series

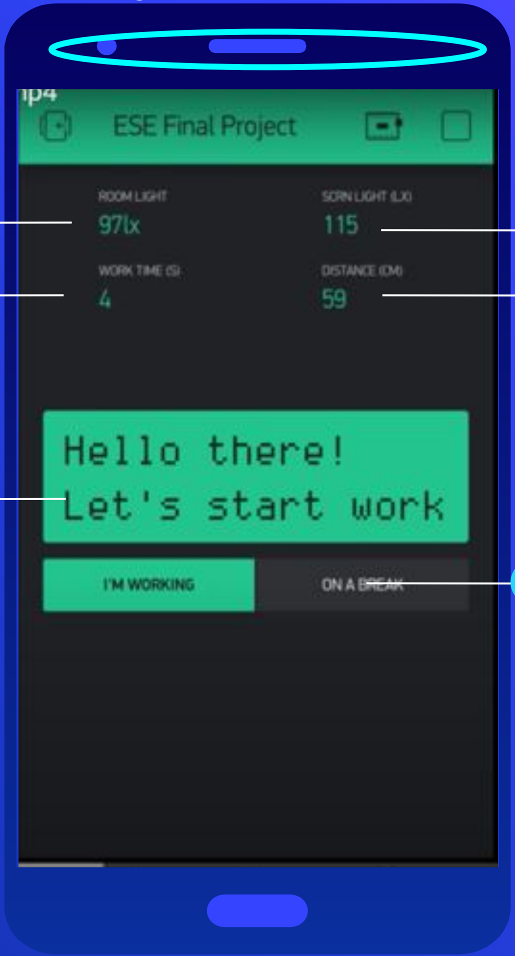
To Arduino GND

To Arduino 5V

Component Terminal	Node MCU Pin
Buzzer (+ve terminal)	D2
Ping Sensor Trigger	D5
Ping Sensor Echo	D6
Photoresistor Analog Pin	A1
GND	GND

Blynk Setup

Smartphone's built-in light sensor (encircled)



ESE Final Project

ROOM LIGHT: 97lx

SCRN LIGHT (LX): 115

WORK TIME (S): 4

DISTANCE (CM): 59

```
Hello there!
Let's start work
```

1 Value display for **ambient light intensity** measured in lux (lx) by the smartphone's built-in light sensor

2 Value display for **screen's light intensity** in lux (lx) calculated from voltage drop across photoresistor

3 Value display for **elapsed time** in one work session in seconds calculated in code

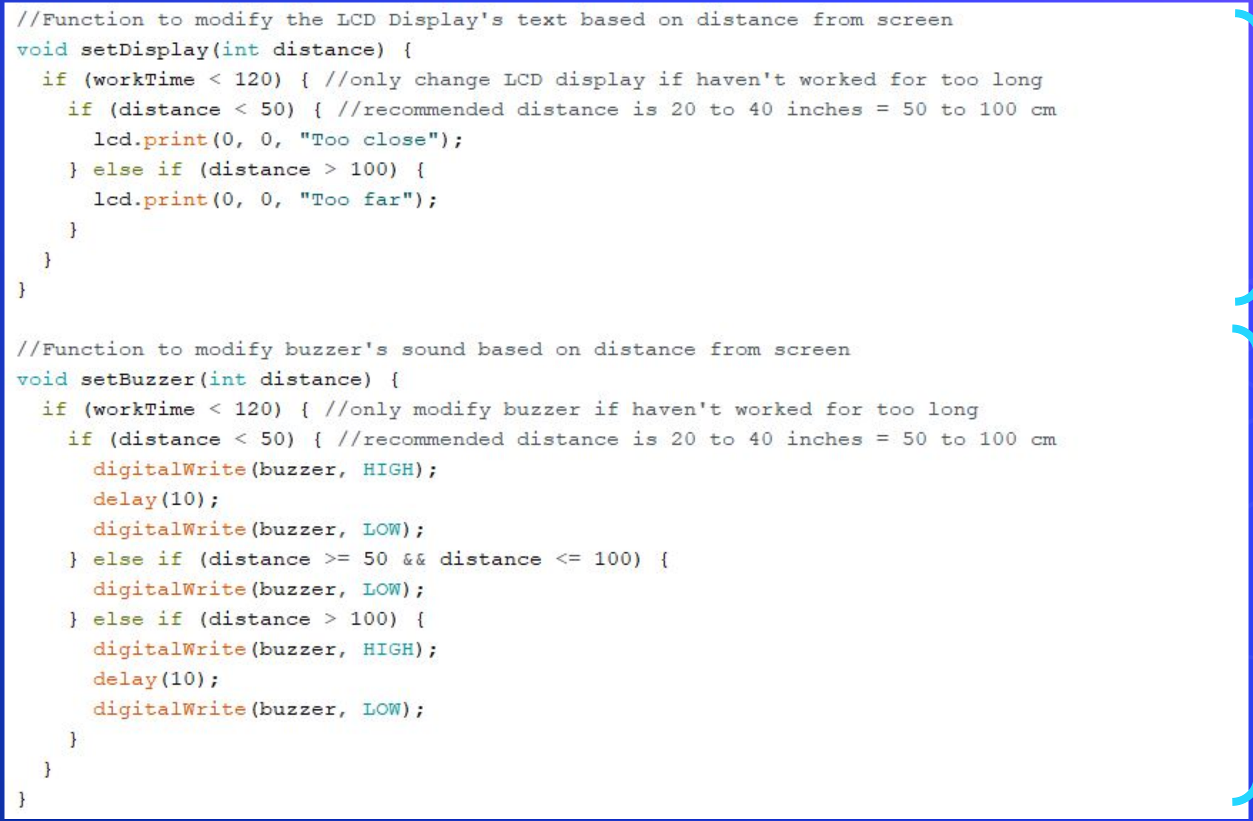
4 Value display for **distance between user and screen** measured in cm by Ping sensor

5 **Virtual LCD Display** to display messages about user's screen usage

6 **Virtual Switch** for user to determine work mode or break mode

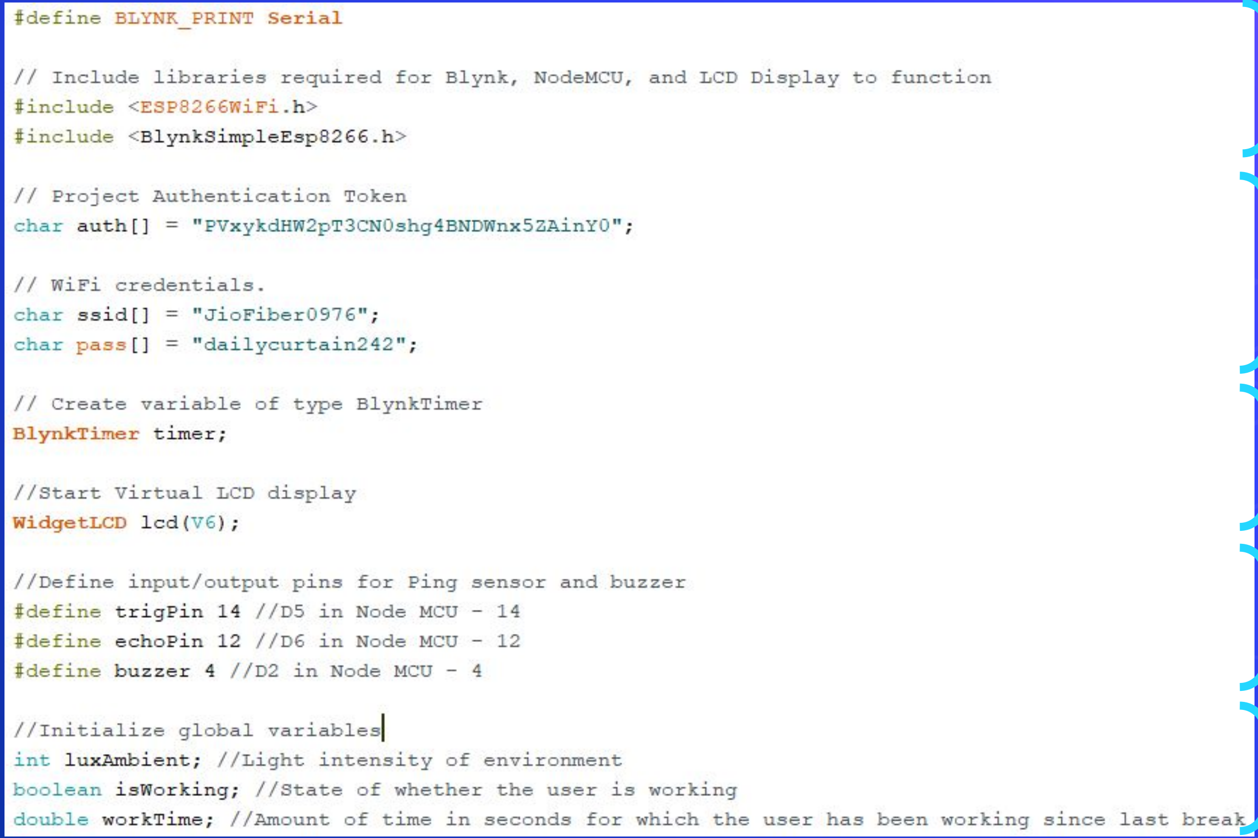
Widget	Virtual Pin
Room Light	V1
Screen Light	V2
Segmented Switch	V3
Work Time	V4
Distance	V5
Virtual LCD (Advanced)	V6

Code



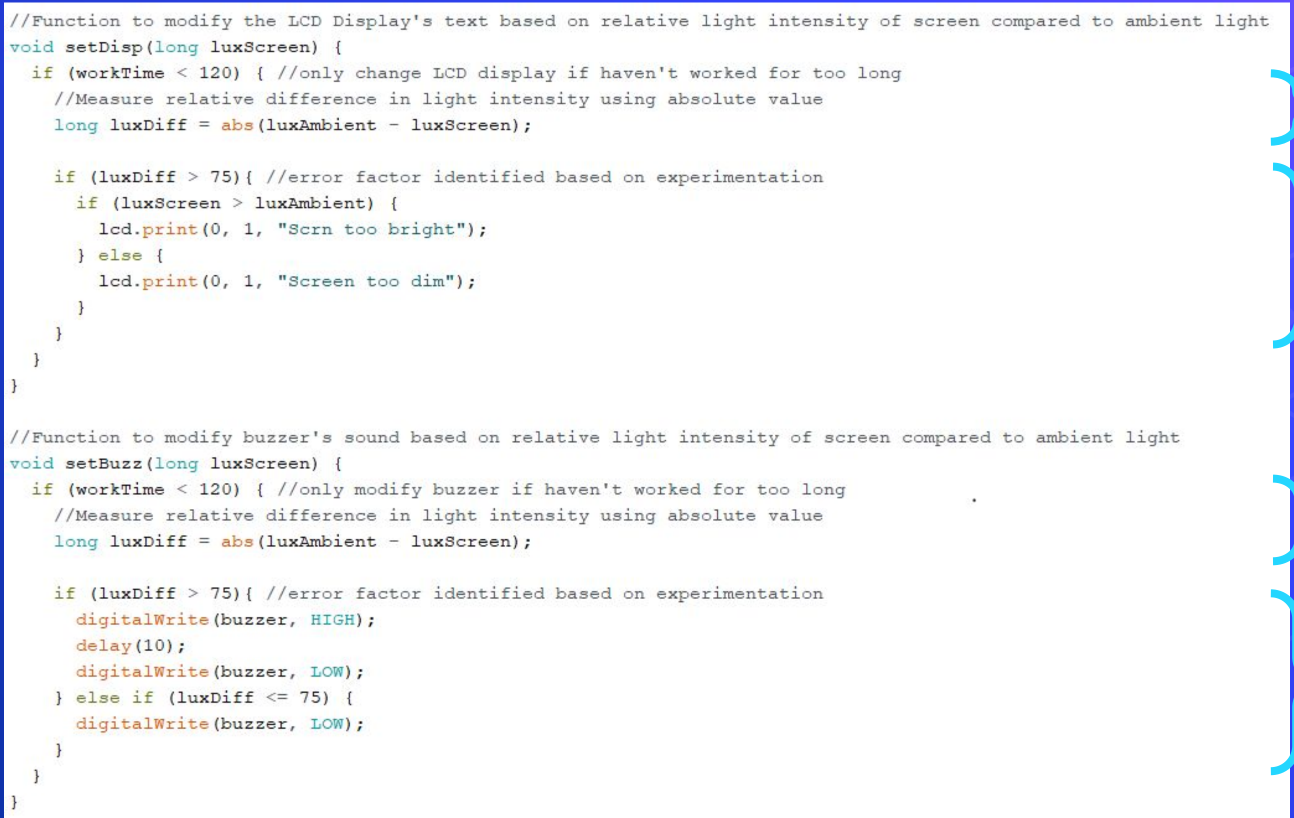
```
//Function to modify the LCD Display's text based on distance from screen
void setDisplay(int distance) {
  if (workTime < 120) { //only change LCD display if haven't worked for too long
    if (distance < 50) { //recommended distance is 20 to 40 inches = 50 to 100 cm
      lcd.print(0, 0, "Too close");
    } else if (distance > 100) {
      lcd.print(0, 0, "Too far");
    }
  }
}

//Function to modify buzzer's sound based on distance from screen
void setBuzzer(int distance) {
  if (workTime < 120) { //only modify buzzer if haven't worked for too long
    if (distance < 50) { //recommended distance is 20 to 40 inches = 50 to 100 cm
      digitalWrite(buzzer, HIGH);
      delay(10);
      digitalWrite(buzzer, LOW);
    } else if (distance >= 50 && distance <= 100) {
      digitalWrite(buzzer, LOW);
    } else if (distance > 100) {
      digitalWrite(buzzer, HIGH);
      delay(10);
      digitalWrite(buzzer, LOW);
    }
  }
}
```

Code



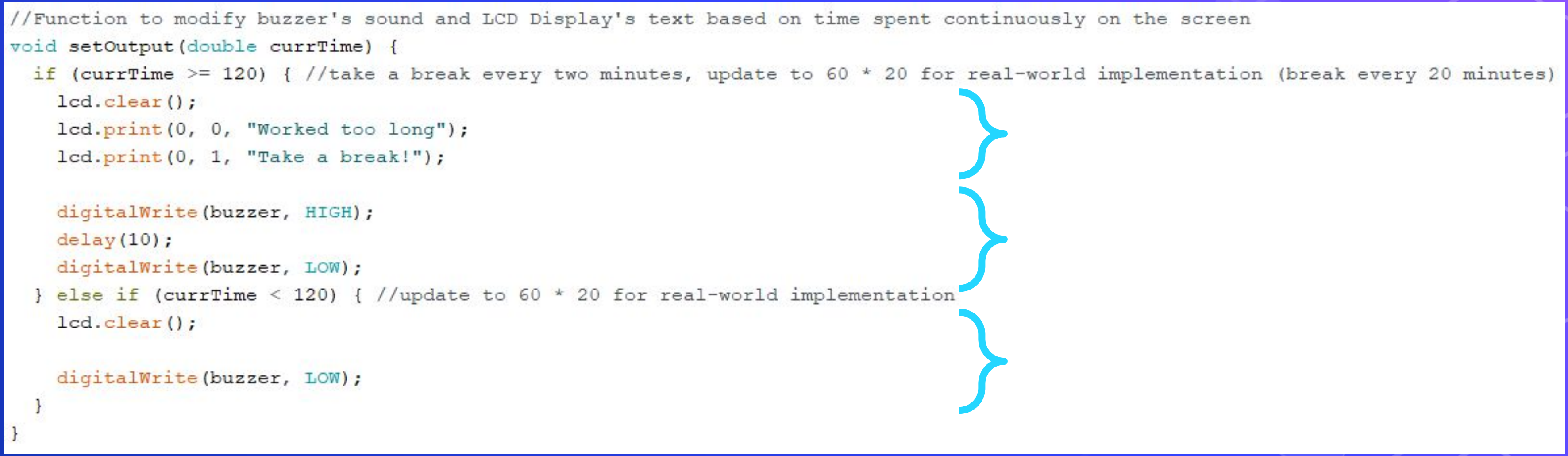
```
#define BLYNK_PRINT Serial

// Include libraries required for Blynk, NodeMCU, and LCD Display to function
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// Project Authentication Token
char auth[] = "PVxykdHW2pT3CN0shg4BNDWnx5ZAinY0";

// WiFi credentials.
char ssid[] = "JioFiber0976";
char pass[] = "dailycurtain242";

// Create variable of type BlynkTimer
BlynkTimer timer;

//Start Virtual LCD display
WidgetLCD lcd(V6);

//Define input/output pins for Ping sensor and buzzer
#define trigPin 14 //D5 in Node MCU - 14
#define echoPin 12 //D6 in Node MCU - 12
#define buzzer 4 //D2 in Node MCU - 4

//Initialize global variables
int luxAmbient; //Light intensity of environment
boolean isWorking; //State of whether the user is working
double workTime; //Amount of time in seconds for which the user has been working since last break
```

Code



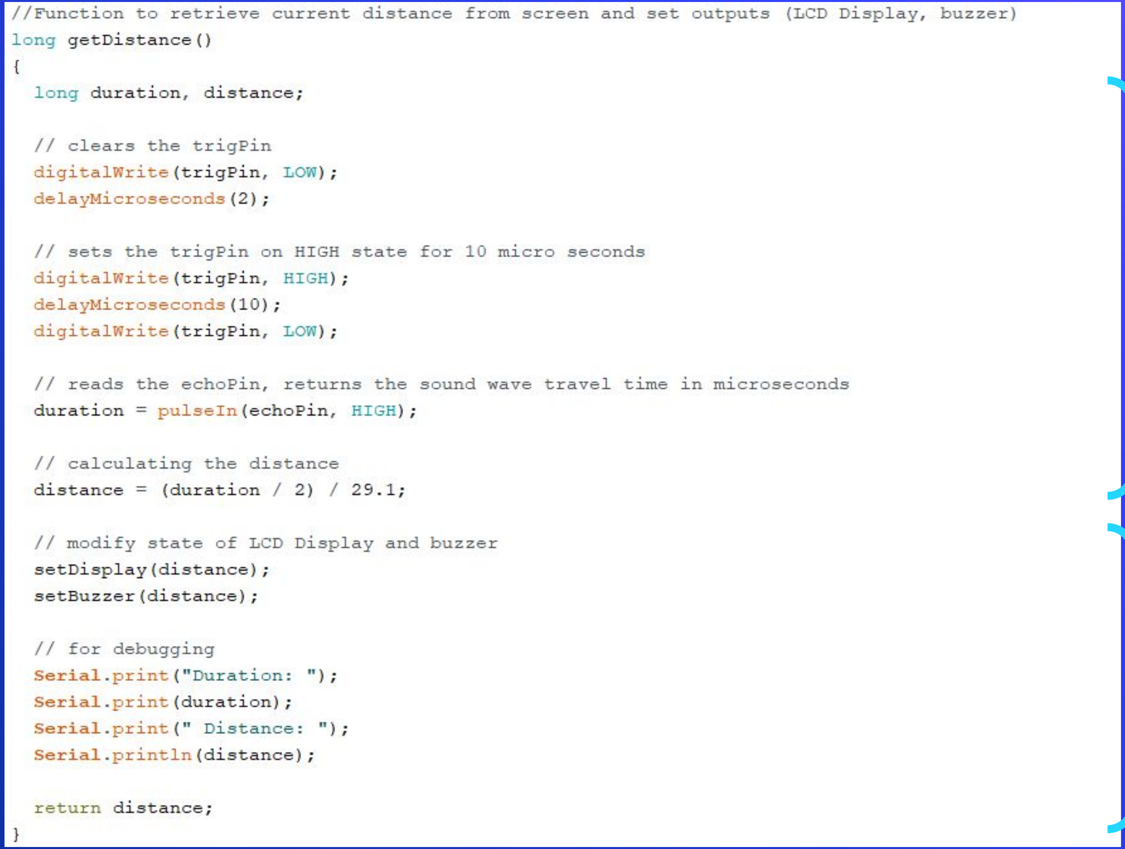
```
//Function to modify the LCD Display's text based on relative light intensity of screen compared to ambient light
void setDisp(long luxScreen) {
  if (workTime < 120) { //only change LCD display if haven't worked for too long
    //Measure relative difference in light intensity using absolute value
    long luxDiff = abs(luxAmbient - luxScreen);

    if (luxDiff > 75){ //error factor identified based on experimentation
      if (luxScreen > luxAmbient) {
        lcd.print(0, 1, "Scrn too bright");
      } else {
        lcd.print(0, 1, "Screen too dim");
      }
    }
  }
}

//Function to modify buzzer's sound based on relative light intensity of screen compared to ambient light
void setBuzz(long luxScreen) {
  if (workTime < 120) { //only modify buzzer if haven't worked for too long
    //Measure relative difference in light intensity using absolute value
    long luxDiff = abs(luxAmbient - luxScreen);

    if (luxDiff > 75){ //error factor identified based on experimentation
      digitalWrite(buzzer, HIGH);
      delay(10);
      digitalWrite(buzzer, LOW);
    } else if (luxDiff <= 75) {
      digitalWrite(buzzer, LOW);
    }
  }
}
```

Code



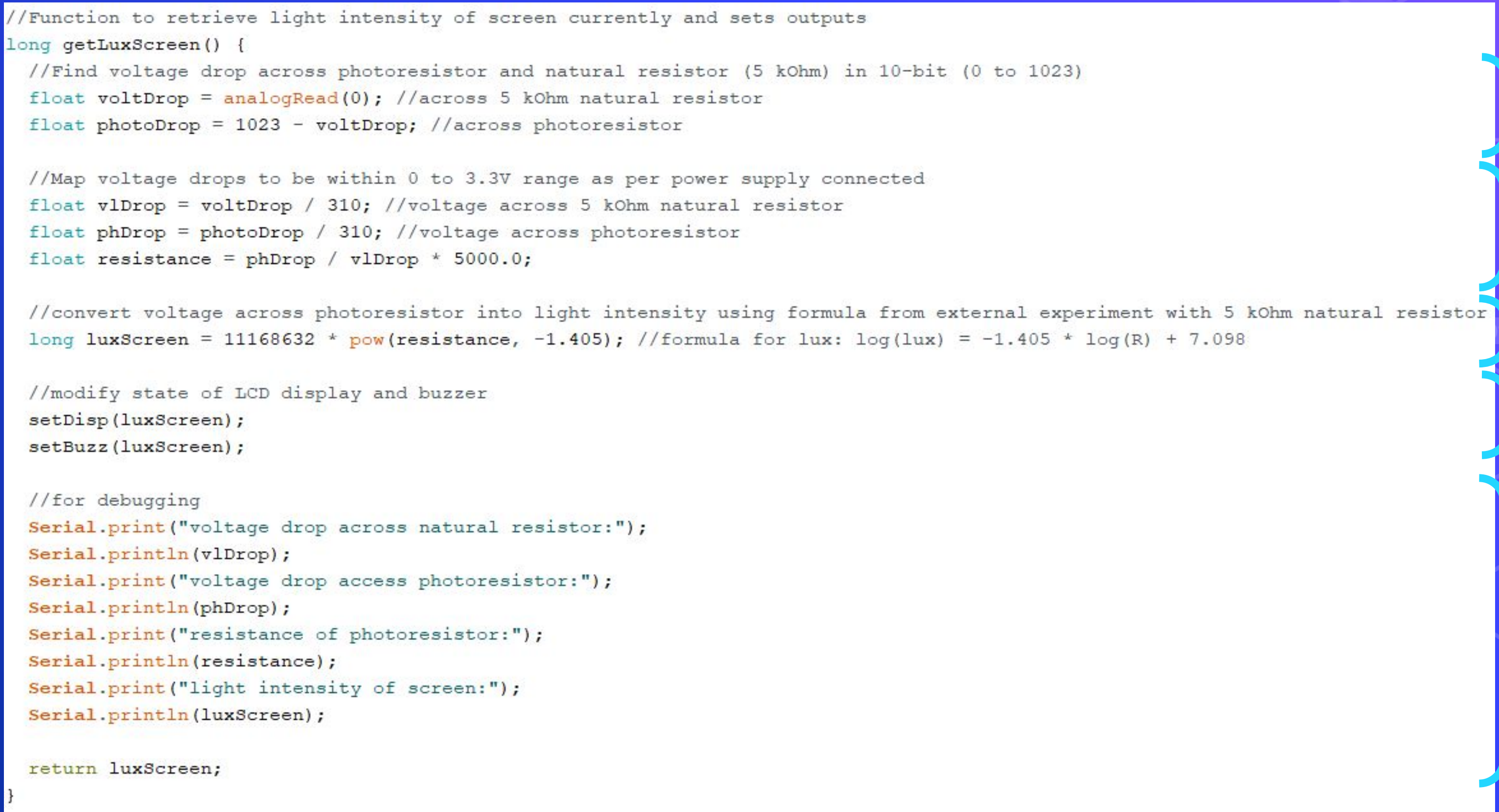
```
//Function to modify buzzer's sound and LCD Display's text based on time spent continuously on the screen
void setOutput(double currTime) {
  if (currTime >= 120) { //take a break every two minutes, update to 60 * 20 for real-world implementation (break every 20 minutes)
    lcd.clear();
    lcd.print(0, 0, "Worked too long");
    lcd.print(0, 1, "Take a break!");

    digitalWrite(buzzer, HIGH);
    delay(10);
    digitalWrite(buzzer, LOW);
  } else if (currTime < 120) { //update to 60 * 20 for real-world implementation
    lcd.clear();

    digitalWrite(buzzer, LOW);
  }
}
```

Code



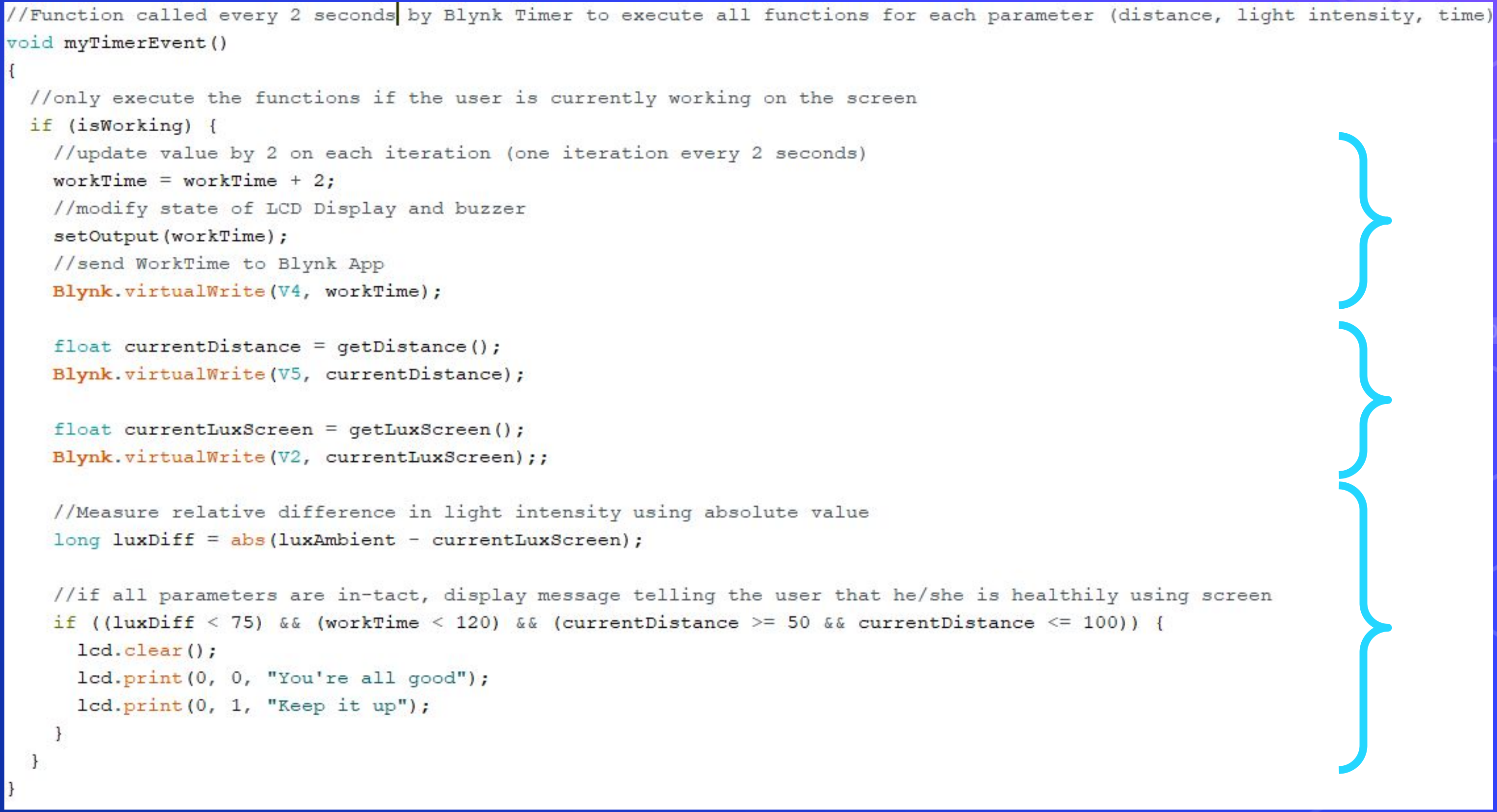
```
//Function to retrieve current distance from screen and set outputs (LCD Display, buzzer)
long getDistance()
{
  long duration, distance;

  // clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // calculating the distance
  distance = (duration / 2) / 29.1;

  // modify state of LCD Display and buzzer
  setDisplay(distance);
  setBuzzer(distance);

  // for debugging
  Serial.print("Duration: ");
  Serial.print(duration);
  Serial.print(" Distance: ");
  Serial.println(distance);

  return distance;
}
```

Code



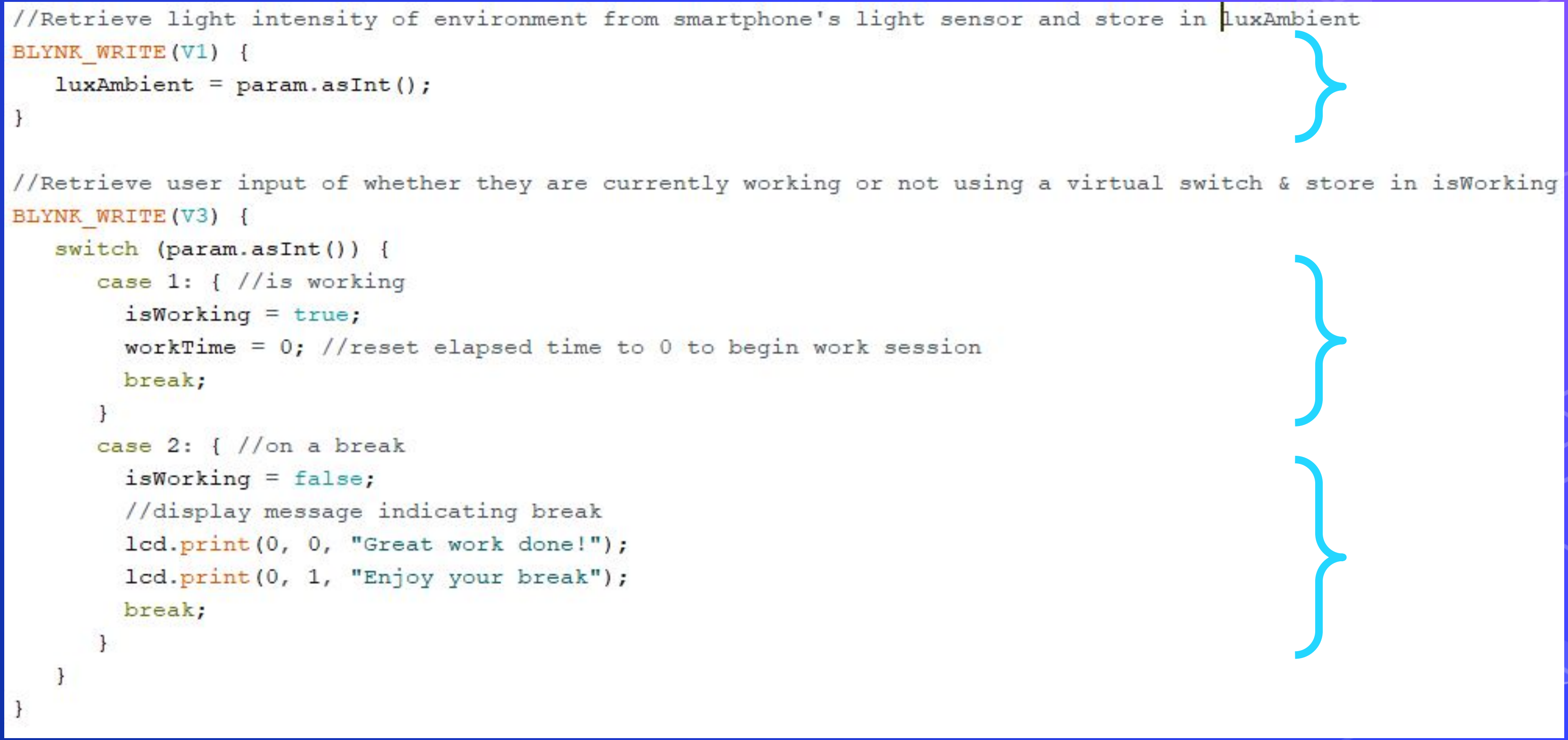
```
//Function to retrieve light intensity of screen currently and sets outputs
long getLuxScreen() {
  //Find voltage drop across photoresistor and natural resistor (5 kOhm) in 10-bit (0 to 1023)
  float voltDrop = analogRead(0); //across 5 kOhm natural resistor
  float photoDrop = 1023 - voltDrop; //across photoresistor

  //Map voltage drops to be within 0 to 3.3V range as per power supply connected
  float vlDrop = voltDrop / 310; //voltage across 5 kOhm natural resistor
  float phDrop = photoDrop / 310; //voltage across photoresistor
  float resistance = phDrop / vlDrop * 5000.0;

  //convert voltage across photoresistor into light intensity using formula from external experiment with 5 kOhm natural resistor
  long luxScreen = 11168632 * pow(resistance, -1.405); //formula for lux: log(lux) = -1.405 * log(R) + 7.098

  //modify state of LCD display and buzzer
  setDisp(luxScreen);
  setBuzz(luxScreen);

  //for debugging
  Serial.print("voltage drop across natural resistor:");
  Serial.println(vlDrop);
  Serial.print("voltage drop access photoresistor:");
  Serial.println(phDrop);
  Serial.print("resistance of photoresistor:");
  Serial.println(resistance);
  Serial.print("light intensity of screen:");
  Serial.println(luxScreen);

  return luxScreen;
}
```

Code



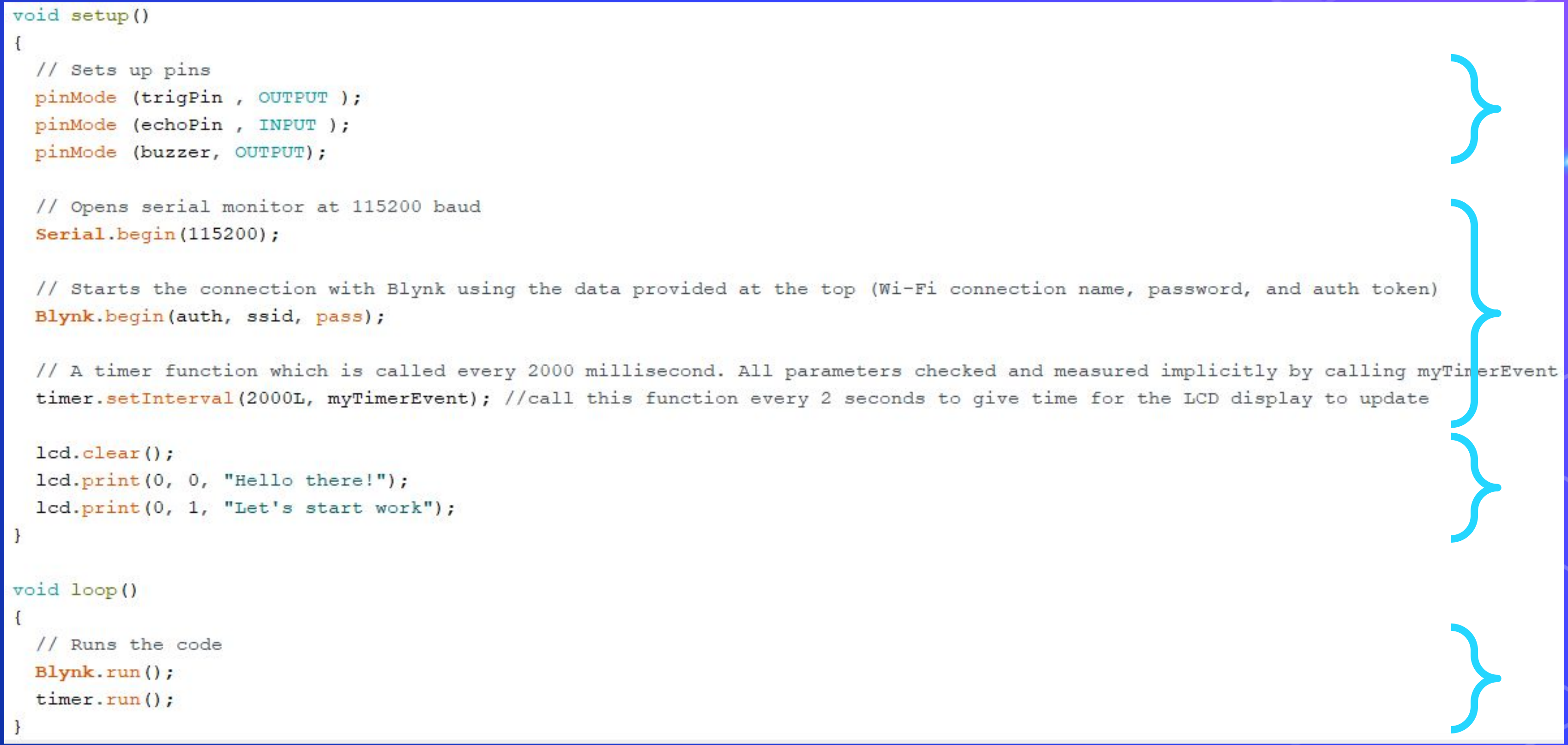
```
//Function called every 2 seconds by Blynk Timer to execute all functions for each parameter (distance, light intensity, time)
void myTimerEvent()
{
  //only execute the functions if the user is currently working on the screen
  if (isWorking) {
    //update value by 2 on each iteration (one iteration every 2 seconds)
    workTime = workTime + 2;
    //modify state of LCD Display and buzzer
    setOutput(workTime);
    //send WorkTime to Blynk App
    Blynk.virtualWrite(V4, workTime);

    float currentDistance = getDistance();
    Blynk.virtualWrite(V5, currentDistance);

    float currentLuxScreen = getLuxScreen();
    Blynk.virtualWrite(V2, currentLuxScreen);;

    //Measure relative difference in light intensity using absolute value
    long luxDiff = abs(luxAmbient - currentLuxScreen);

    //if all parameters are in-tact, display message telling the user that he/she is healthily using screen
    if ((luxDiff < 75) && (workTime < 120) && (currentDistance >= 50 && currentDistance <= 100)) {
      lcd.clear();
      lcd.print(0, 0, "You're all good");
      lcd.print(0, 1, "Keep it up");
    }
  }
}
```

Code



```
//Retrieve light intensity of environment from smartphone's light sensor and store in luxAmbient
BLYNK_WRITE(V1) {
   luxAmbient = param.asInt();
}

//Retrieve user input of whether they are currently working or not using a virtual switch & store in isWorking
BLYNK_WRITE(V3) {
   switch (param.asInt()) {
      case 1: { //is working
        isWorking = true;
        workTime = 0; //reset elapsed time to 0 to begin work session
        break;
      }
      case 2: { //on a break
        isWorking = false;
        //display message indicating break
        lcd.print(0, 0, "Great work done!");
        lcd.print(0, 1, "Enjoy your break");
        break;
      }
   }
}
```

Code



```
void setup()
{
  // Sets up pins
  pinMode (trigPin , OUTPUT );
  pinMode (echoPin , INPUT );
  pinMode (buzzer, OUTPUT);

  // Opens serial monitor at 115200 baud
  Serial.begin(115200);

  // Starts the connection with Blynk using the data provided at the top (Wi-Fi connection name, password, and auth token)
  Blynk.begin(auth, ssid, pass);

  // A timer function which is called every 2000 millisecond. All parameters checked and measured implicitly by calling myTimerEvent
  timer.setInterval(2000L, myTimerEvent); //call this function every 2 seconds to give time for the LCD display to update

  lcd.clear();
  lcd.print(0, 0, "Hello there!");
  lcd.print(0, 1, "Let's start work");
}

void loop()
{
  // Runs the code
  Blynk.run();
  timer.run();
}
```

Challenges and Adjustments

- Measuring screen light intensity
  - Blocking out ambient light
  - Converting voltage drop to light intensity
- Measuring relative light intensity
  - Limited analog pins on Node MCU
  - Identifying error factor
- Integrating LCD Display into the circuit
- Enabling buzzer to function with code

Implications

Strengths

- Measures three key screen-use parameters at once and with priority (screen time > distance and screen time > intensity)
- Alerts the user in multiple ways (audio and visual) to ensure they correct their setup
- Very user-friendly setup and operation
- Ensures productivity and healthy screen usage at once (20 min work sessions ~ Pomodoros)
- Applicable to any kind of work using screens: teleworking, online learning, social networking, server monitoring, etc.

Limitations

- Design is quite bulky so only suitable for laptops/PCs
- Cap is not custom-manufactured so some ambient light can still pass through
- Recommended parameters are for adults so not applicable for children/elderly
- Direct eye contact is required for accurate reading on Ping sensor
- Sensors are fixed to computer screen (not detachable)

Future Development

- Use more compact sensors, breadboard, and microprocessor to suit phone screens
- Design a smaller cap using precision tools to more accurately restrict ambient light
- Include child and elderly modes into device code with their respective parameters
- Use a more advanced sensor to accommodate for an acute viewing angle
- Incorporate a clip-on feature to enable sensors to be detachable from the screen

Bibliography

https://www.digikey.com/en/maker/projects/design-a-luxmeter-with-an-ldr-and-an-arduino/623aeee0f93e427bb57e02c4592567d1

https://www.wired.com/2013/09/flux-eyestrain/

https://www.osha.gov/SLTC/etools/computerworkstations/components_monitors.htm

https://www.healthline.com/health/eye-health/20-20-20-rule#symptoms

https://invootech.blogspot.com/2017/06/how-to-convert-ldr-dependent-resistor.html

Thanks!

Any questions?