# Table of Contents

- System Overview 3
- Electrical Specification 3
- User Guide 3
- Block Diagram 4
- Electrical Schematic 6
- 3D Model and Dimensions 7
- PCB Layout 9
- Parts List 10
- Arduino Code 10
System Overview

The study timer is used to set a prescribed amount of time where you cannot use your phone in order to conduct study sessions without the distraction of your handheld device. The user may select between two timer options of five or twenty-five minutes through pushbuttons and start the study timer by their phone on top of the timer box. If the phone is removed from its designated area during the timer countdown, a buzzer will go off until the phone has been placed back where it was. Once the timer runs out, the buzzer will go off until the phone has been removed from its designated area and the system will proceed to restart its routine all over again. A seven segment display is used to display the time, and two photo sensors are used for reading the light levels of the environment. One photo sensor is used for detecting the presence of the phone and the other is used to detect the light levels around the study timer in order to determine the brightness setting of the timer display.

Electrical Specification

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Voltage</td>
<td>3.3 Volts</td>
</tr>
<tr>
<td>Maximum Supply Voltage</td>
<td>3.55 Volts</td>
</tr>
<tr>
<td>Minimum Supply Voltage</td>
<td>3.15 Volts</td>
</tr>
<tr>
<td>Maximum Supply Current</td>
<td>1.5 Amp</td>
</tr>
<tr>
<td>Minimum Supply Current</td>
<td>0.5 Amps</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>-40 to 60 Celsius</td>
</tr>
</tbody>
</table>

Table 1: Electrical Specifications

User Guide

Power Device:
To power up the system, insert a USB 2.0 A-Male to Mini-B cable into the USB port located to the left hand side of the timer box labeled “USB.” Once the USB port has been plugged in, the timer display located on the top of the timer box will display a zero.

Setup Timer:
The system provides the selection between two timer options of five or twenty-five minutes. The five minute timer option is selected through pushing the button labeled ‘5’ located at the front of the timer box. The twenty-five minute timer option is selected through pushing the button labeled ‘25’ located at the front of the timer box. The selected timer option should be seen on the timer display.

Start Timer:
The timer will commence once a phone (or any solid, opaque object) has been placed on top of the sensor located on top of the timer box and below the timer display. The timer should then commence.

Time’s Up:
Once the timer reaches zero, the buzzer will go off indicating that the phone can now be removed. The buzzer will stop once the phone has been removed away from the sensor area on top of the timer box. The system will then start all over again, allowing the user to select a new timer option to start the timer again.

Block Diagram

Figure 1: Black Box Diagram of System

This diagram showcases all external inputs and outputs of the system.
Figure 2: Block Diagram of Individual Blocks in System

This diagram showcases the connection between all individual blocks of the system and the interfaces used for communication.

<table>
<thead>
<tr>
<th>Interface Name</th>
<th>Interface Type</th>
<th>Specifics</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME_BUTTON1_USRIN</td>
<td>User input</td>
<td>Input state 1: Logic HIGH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input state 2: Logic LOW</td>
</tr>
<tr>
<td>TIME_BUTTON2_USRIN</td>
<td>User input</td>
<td>Input state 1: Logic HIGH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input state 2: Logic LOW</td>
</tr>
<tr>
<td>BUTTON_DSIG</td>
<td>Digital Signal</td>
<td>Input state 1: Logic HIGH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input state 2: Logic LOW</td>
</tr>
<tr>
<td>PHOTOSENSOR_ENVIN</td>
<td>Environmental Input</td>
<td>Resistance: 1-10k Ohms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Length: 4.46mm/0.18in</td>
</tr>
<tr>
<td>PHOTOSENSOR_ASIG</td>
<td>Analog Signal</td>
<td>Brightness Reading: 0-10,000 lux</td>
</tr>
<tr>
<td>USB_DCPWR</td>
<td>DC power</td>
<td>Vmax: 5V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Imax: 0.5A</td>
</tr>
<tr>
<td>ARDUINO_CODE</td>
<td>Code</td>
<td>C language</td>
</tr>
</tbody>
</table>
Table 2: Interface Data Table

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Output Type</th>
<th>Frequency</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUZZER_USROUT</td>
<td>User output</td>
<td>Frequency: 440 +/- 1 Hz</td>
<td>Voltage: 3-10V</td>
</tr>
<tr>
<td>SEVEN_SEG_USROUT</td>
<td>User output</td>
<td>Wavelength Peak: 565-660nm</td>
<td>Power Dissipation: (Max) 75-105mW</td>
</tr>
</tbody>
</table>

**Electrical Schematic**

The schematic shows the connections made between the arduino nano, seven segment display, and individual components of the system. The arduino nano uses three digital pins, four analog pins, the 3.3V pin, and ground pin.
3D Model and Dimensions

Figure 4: 3D Model of Enclosure (Side View)

Figure 5: 3D Model of Enclosure (Front View)
Figure 6: Dimensions of 3D Enclosure Model

TinkerCad was used for creating 3D models for the enclosure of the system. The container is made to house the arduino and user input components while the lid houses the seven segment display and photosensors. For this project, the 3D printed model was not manufactured and a cardboard enclosure was constructed instead.
Figure 7: Dimensions of Cardboard Enclosure Model

PCB Layout

Figure 8: PCB Layout of System
The PCB layout shows the connections between the individual components, seven segment display, and the arduino nano. The PCB is 60 by 60 mm.

**Parts List**

<table>
<thead>
<tr>
<th>Description</th>
<th>Product Number</th>
<th>Quantity</th>
<th>Cost per Unit</th>
<th>Cost Total</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mini Nano V3.0 Atmega 328P Microcontroller Board</td>
<td>G418920056</td>
<td>1.0</td>
<td>$9.99</td>
<td>$9.99</td>
<td>Elisona</td>
</tr>
<tr>
<td>Adafruit 0.56” 7-Segment FeatherWing Display</td>
<td>3106</td>
<td>1.0</td>
<td>$11.95</td>
<td>$11.95</td>
<td>Adafruit Industries</td>
</tr>
<tr>
<td>Photo cell (CdS Photoresistor)</td>
<td>161</td>
<td>2.0</td>
<td>$0.95</td>
<td>$0.95</td>
<td>Digi-Key Electronics</td>
</tr>
<tr>
<td>Piezo Buzzer &amp; Audio Indicators Round</td>
<td>SD160701</td>
<td>1.0</td>
<td>$1.03</td>
<td>$1.03</td>
<td>TDK</td>
</tr>
<tr>
<td>Through-Hole Resistors - 10k ohm 5%</td>
<td>2780</td>
<td>1.0</td>
<td>$0.03</td>
<td>$0.75</td>
<td>Mouser Electronics</td>
</tr>
</tbody>
</table>

Table 3: Bill of Materials

**Arduino Code**

The code used for the arduino is shown below. The main loop function is used for running the timer and displaying it on the seven segment display. The other functions are used to run different routines based on the state of the phone and light levels of the environment. Areas of improvement for the code is to reduce the number of functions used for the different routines.
as well as simplifying the timer routine by combining the two functions called that check the light levels of the environment.

/∗ Project: Pomodoro Timer
 ∗ Class: ECE 342 Junior Design II
 ∗ Group: Timer 04 (2d)
 ∗ Name: Sean Lee
 ∗/
 #include <Wire.h>
 #include "Adafruit_LEDBackpack.h"
 #include "Adafruit_GFX.h"

 Adafruit_7segment matrix = Adafruit_7segment();

 const int button1 = 4; // button pins from 4-8
 const int button2 = 5;
 const int buzzerPin = 6;

 uint16_t counter_0; // variables for storing the digit value of each counter
 uint16_t counter_1;
 uint16_t counter_3;
 uint16_t counter_4;

 int buttonState1 = 0; // button state variable for keeping track
 int buttonState2 = 0;

 int photocellPin = A0; // pin A0 for photocell
 int photocellPin2 = A1;
 int sensorReading; // sensor reading

 void setup() {
 #ifndef __AVR_ATtiny85__
 Serial.begin(115200); // serial communication
 #endif
 matrix.begin(0x70); // seven segment display matrix
 pinMode(button1, INPUT); // set pins as input
 pinMode(button2, INPUT);
 pinMode(buzzerPin, OUTPUT);
 }

 /∗ Name: loop
 ∗ Function: This is the main loop function that will run the seven segment display routine while calling
* other functions for different scenarios. Once the timer is up, it will call the timer_end
function before returning to the top of the loop to run the routine all over again.
*/
void loop() {
    set_timer();
    matrix.drawColon(true);                                         // display the colon
    delay(1000);                                                    // delay for one second
    for (counter_0; counter_0 < 3; counter_0--) {
        // loop for the first digit on display
        matrix.writeDigitNum(0, counter_0);
        matrix.writeDisplay();
        read_light();
        read_light2();
        for(counter_1 = 0; counter_1 < 10; counter_1++) {             // loop for the second digit on display
            matrix.writeDigitNum(1, counter_1);
            matrix.writeDisplay();
            read_light();
            read_light2();
            for(counter_3 = 5; counter_3 < 6; counter_3++) {
                // loop for the third digit on display
                matrix.writeDigitNum(3, counter_3);
                matrix.writeDisplay();
                read_light();
                read_light2();
                for(counter_4 = 9; counter_4 < 10; counter_4++) {       // loop for the fourth digit on display
                    matrix.writeDigitNum(4, counter_4);
                    matrix.writeDisplay();
                    read_light();
                    read_light2();
                    delay(1000);                                          // delay for one second
                }
            }
        }
        counter_1 = 9;                                                // set second digit restart
        number to 9
    }
} timer_end();
return;
}

/* Name: set_timer
 * Function: Waits for user to first select 5 or 25 minute timer, then waits for user to select
one of the three brightness levels for the display

```cpp
void set_timer() {
    matrix.print(0000); // reset the counter
    matrix.writeDisplay();
    Serial.println(" Please select timer option"); // display options on serial
    Serial.println(" Button [1] for 5 minutes");
    Serial.println(" Button [2] for 25 minutes");
    do{
        buttonState1 = digitalRead(button1); // read button
        if (buttonState1 == HIGH) // button input condition
            {
            matrix.print(500, DEC);
            matrix.drawColon(true); // display the colon
            matrix.writeDisplay();
            Serial.println(" Button 1 pressed");
            counter_0 = 0;
            counter_1 = 4;
            Serial.println(" Timer set for 5 minutes");
            delay(1000);
        }
        buttonState2 = digitalRead(button2); // read button
        if (buttonState2 == HIGH) // button input condition
            {
            matrix.print(2500, DEC);
            matrix.drawColon(true); // display the colon
            matrix.writeDisplay();
            Serial.println(" Button 2 pressed");
            counter_0 = 2; // set appropriate timer values
            counter_1 = 4;
            Serial.println(" Timer set for 25 minutes");
            delay(1000);
        }
    }while((buttonState1 != HIGH) && (buttonState2 != HIGH)); // check when button is pressed
    check_phone();
}
```

/* Name: read_light
   * Function: Reads the sensor value and if it exceeds the given value, call the function phone_removed, otherwise return to loop function
   */
void read_light() {

```c
int sensorReading = analogRead(photocellPin);  // read sensor

Serial.println(sensorReading);                  // print the sensor reading in analog

if (sensorReading < 100) {                     // check if reading is within threshold

    //Serial.println(" Phone Stationary");
}

else if (sensorReading > 200) {                // check if reading exceeds threshold

    //Serial.println(" Phone Removed");
    phone_removed();
}
}

void read_light2() {

    int sensorReading = analogRead(photocellPin2);  // read sensor

    Serial.println(sensorReading);                  // print the sensor reading in analog

    if (sensorReading < 150) {                     // check if reading is within threshold
        matrix.setBrightness(15);                  // set brightness
        matrix.writeDisplay();
        Serial.println(" highest brightness");
    }

    if ((sensorReading > 150) && (sensorReading < 600)) {  // check if reading exceeds threshold
        matrix.setBrightness(5);                    // set brightness
        matrix.writeDisplay();
        Serial.println(" normal brightness");
    }

    else if (sensorReading > 600) {                // check if reading exceeds threshold
        matrix.setBrightness(0);                    // set brightness
        matrix.writeDisplay();
        Serial.println(" lowest brightness");
    }
}
```
void check_phone() {
    int sensorReading = 0;
    do{
        sensorReading = analogRead(photocellPin); // read sensor
        //Serial.println(sensorReading); // print the sensor reading
        in analog
        //Serial.println(" Phone is not placed yet");
        delay(1000);
    }while(sensorReading > 100);
}

void phone_removed() {
    int sensorReading = 0;
    int buzzerReadings = 0;
    do{
        sensorReading = analogRead(photocellPin); // read sensor
        //Serial.println(sensorReading); // print the sensor reading
        in analog
        //Serial.println(" Phone is not placed yet");
        tone(buzzerPin, 440, 100);
        delay(1000);
    }while(sensorReading > 100);
}

void timer_end() {
    int sensorReading = 0;
    do{
        sensorReading = analogRead(photocellPin); // read sensor
        //Serial.println(sensorReading); // print the sensor reading
        in analog
        //Serial.println(" Phone is not removed yet");
        tone(buzzerPin, 440, 100); // play tone through buzzer at
        delay(1000);
    }while(sensorReading < 100);