Study Timer Developer Guide

ECE 342 Winter 2020

Sean Lee

Table of Contents

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

Parameter	Value
Operating Voltage	3.3 Volts
Maximum Supply Voltage	3.55 Volts
Minimum Supply Voltage	3.15 Volts
Maximum Supply Current	1.5 Amp
Minimum Supply Current	0.5 Amps
Operating Temperature	-40 to 60 Celsius

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

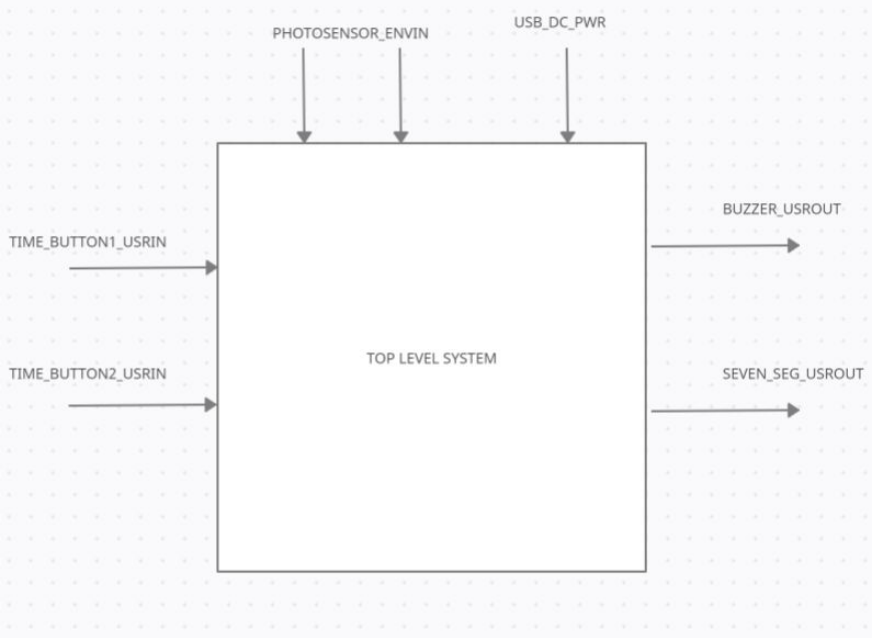

Figure 1: Black Box Diagram of System

This diagram showcases all external inputs and outputs of the system.

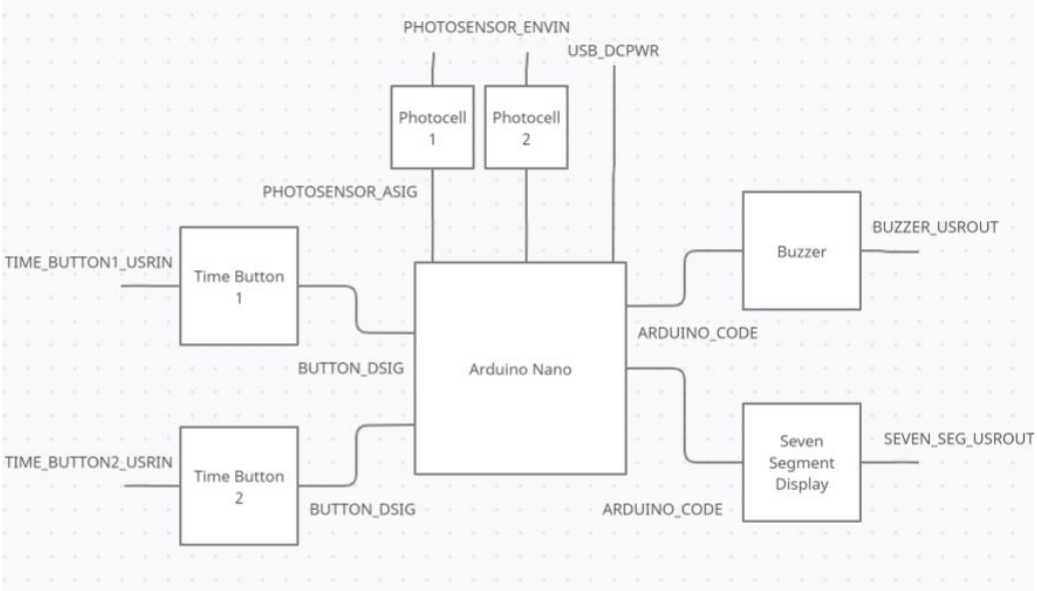

Figure 2: Block Diagram of Individual Blocks in System

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

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

BUZZER_USROUT	User output	Frequency: 440 +/- 1 Hz Voltage: 3-10V
SEVEN_SEG_USROUT	User output	Wavelength Peak: 565-660nm Power Dissipation: (Max) 75-105mW

Table 2: Interface Data Table

Electrical Schematic

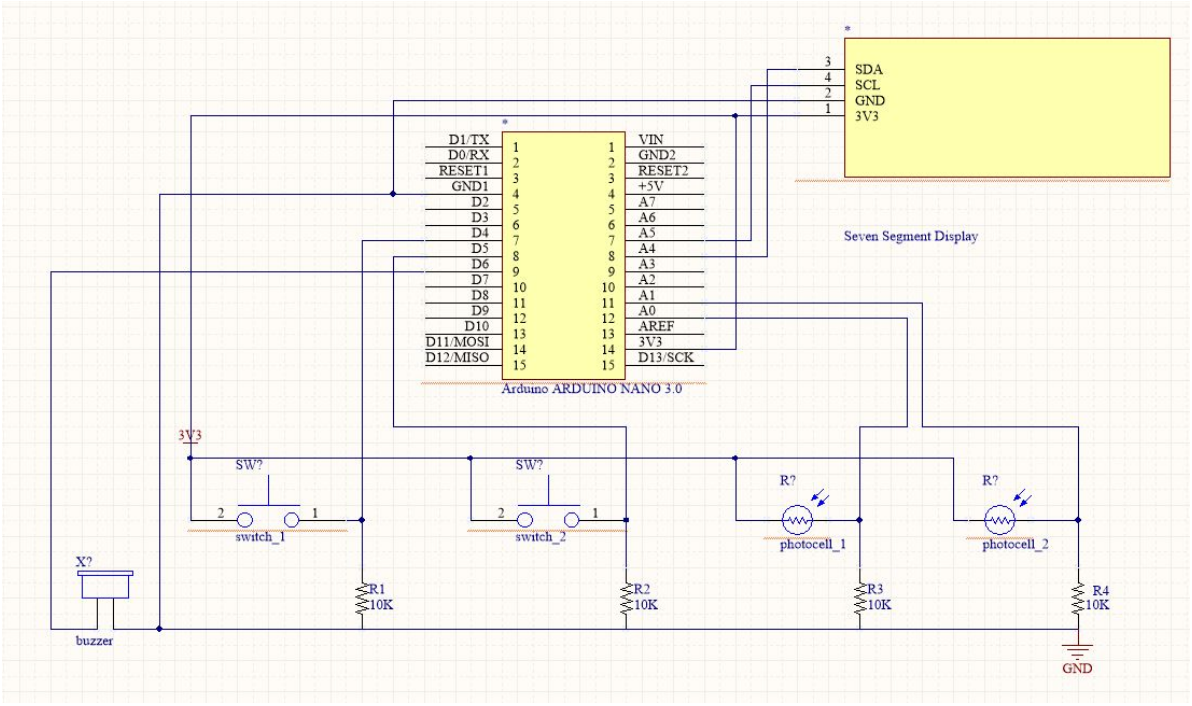

Figure 3: Electrical Schematic of System

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

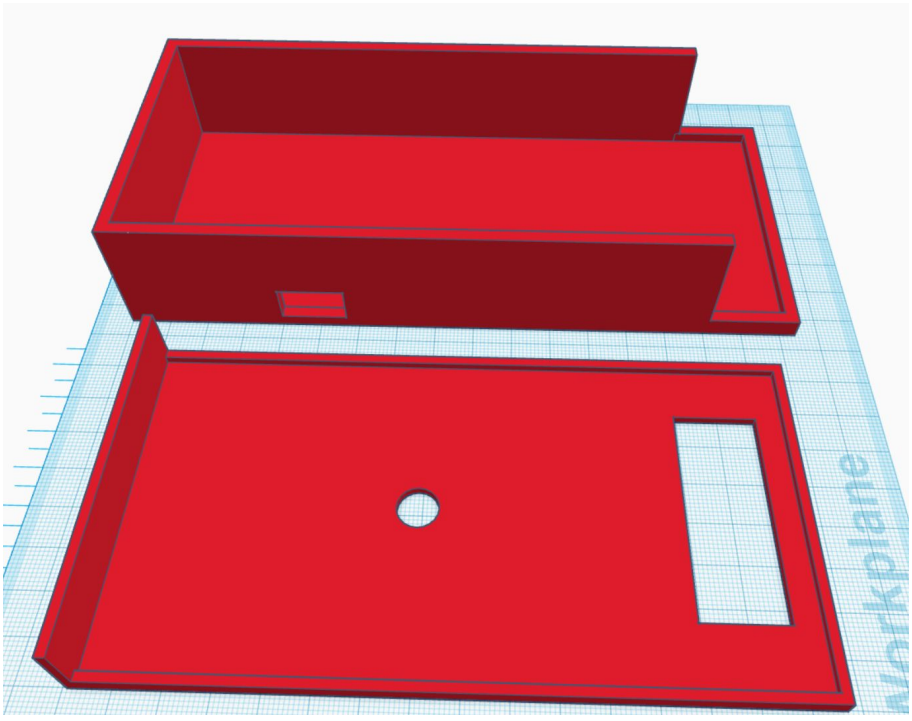

Figure 4: 3D Model of Enclosure (Side View)

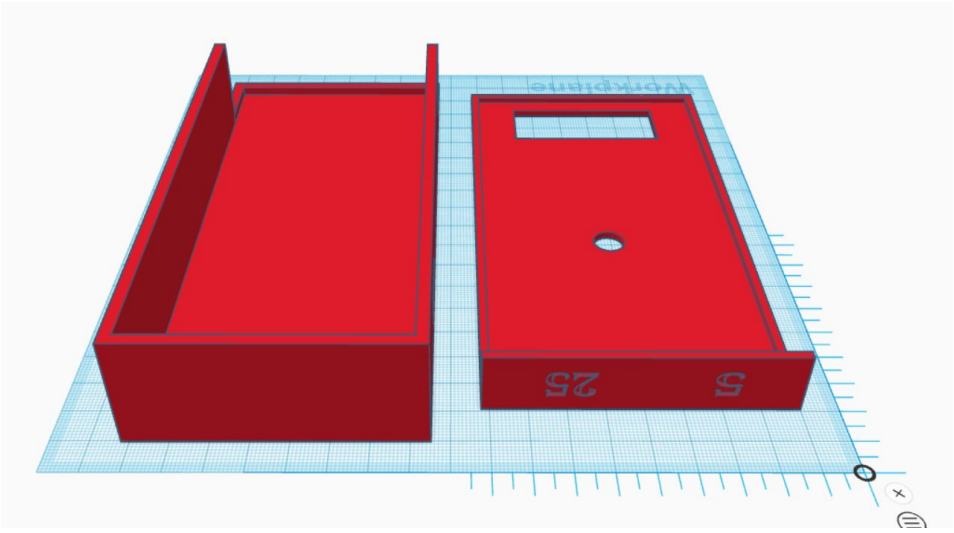

Figure 5: 3D Model of Enclosure (Front View)

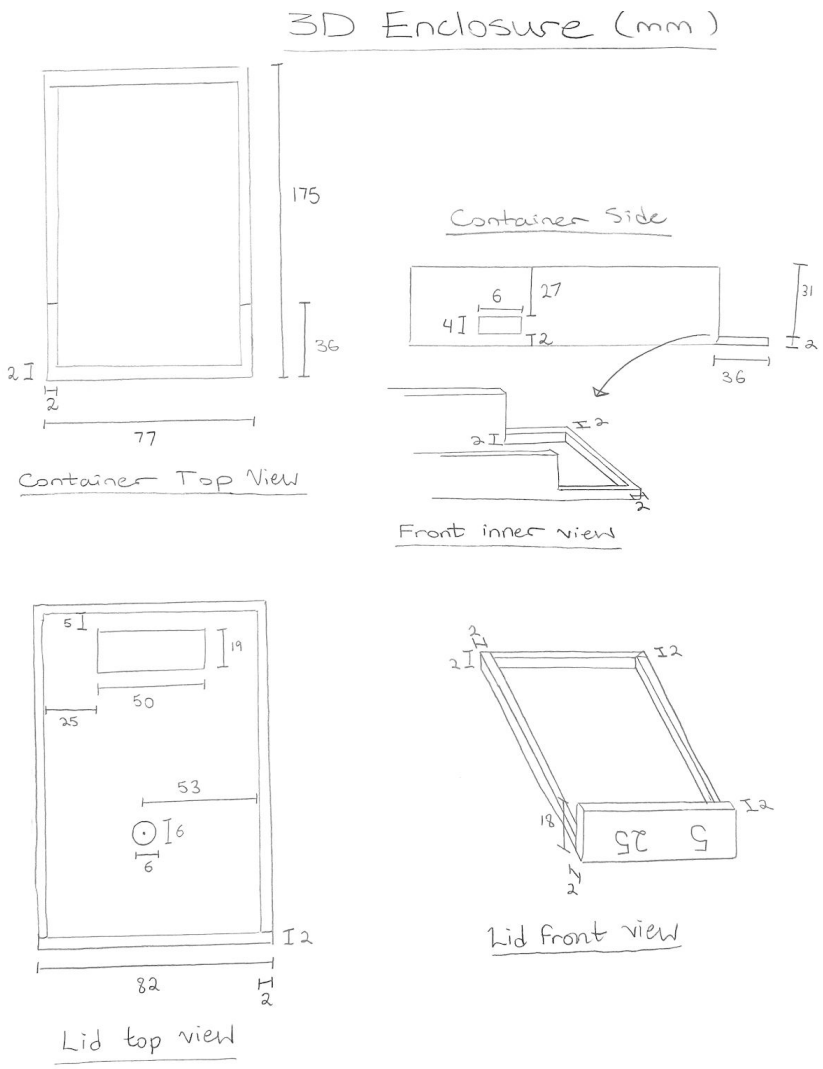

Figure 6: Dimensions of 3D Enclosure Model

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.

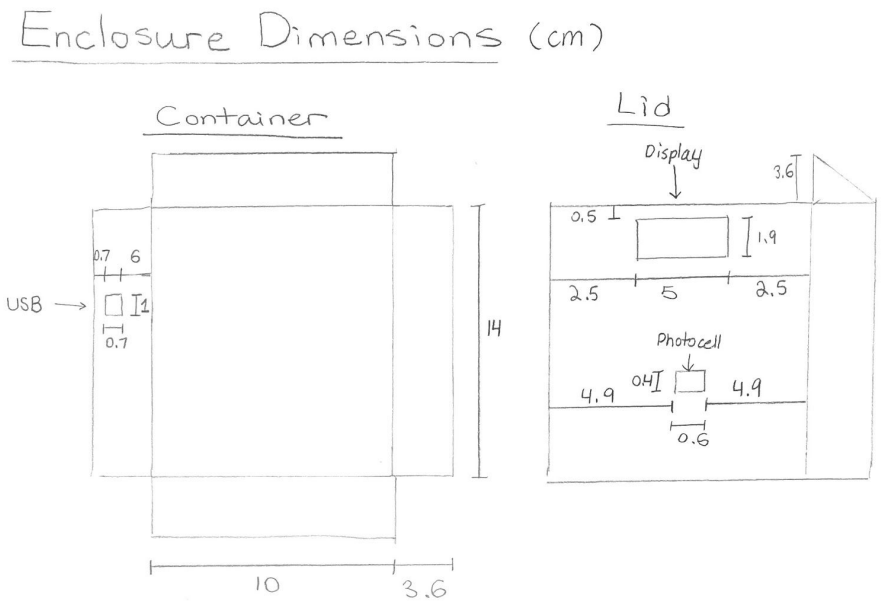

Figure 7: Dimensions of Cardboard Enclosure Model

PCB Layout

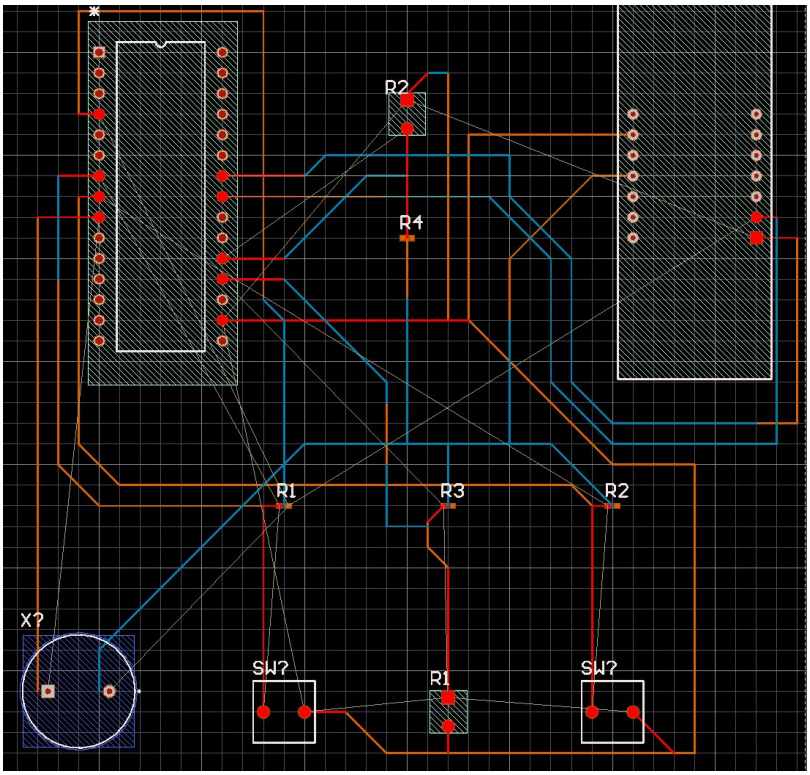

Figure 8: PCB Layout of System

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

Description	Product Number	Quantity	Cost per Unit	Cost Total	Manufacturer
Mini Nano V3.0 Atmega 328P Microcontroller Board	G418920056	1.0	$9.99	$9.99	Elisona
Adafruit 0.56” 7-Segment FeatherWing Display	3106	1.0	$11.95	$11.95	Adafruit Industries
Photo cell (CdS Photoresistor)	161	2.0	$0.95	$0.95	Digi-Key Electronics
Piezo Buzzer & Audio Indicators Round 16mmx7mm 4096Hz Vin = 1-5V	SD160701	1.0	$1.03	$1.03	TDK
Through-Hole Resistors - 10k ohm 5% 1/4W (Pack of 25)	2780	1.0	$0.03	$0.75	Mouser Electronics

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; 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
 */
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;                                       // set appropriate timer values
        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() {
```

```
  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");
  }
}
```

```
/*  Name: check_phone
 *  Function: Reads the sensor value until it is less than 300.
 */
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);
}

/*  Name: phone_removed
 *  Function: Reads sensor and plays a buzzer tone until the reading value is less than 300
 */
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);
}

/*  Name: timer_end
 *  Function: Reads sensor value and plays a buzzer tone until the reading value is greater
than 200
 */
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
440Hz
  delay(1000);
  }while(sensorReading < 100);
```

```
}
```