POMODORO TIMER

DEVELOPER GUIDE

CHRISTOPHER DUEBER

MAY 28 2021

System Overview

The Pomodoro Timer operates on battery power and detects objects laid on top of the bed. The interface allows the user to select from two different time settings, active alarm controls, and a range of brightness. If the alarm is active, the Pomodoro Timer will alert the user through its on-board speaker at 440 Hz if an object has been removed from the bed. The 3D-printed case allows for an assortment of objects to be placed on the bed.

Electrical Specifications

In addition to these design specifications, the customer is also asking for the following:

1. The timer must be less than 1 second off every minute.
2. The system must use connectors for every module used in the timer system, have a power disconnect switch, and not have any exposed conductors. Wires must be organized in split loom or other protective materials. All devices must be rated at least IP43
3. Every switch and potentiometer on the user interface should have a label that can be read from three feet away by at least 2 people other than the project designer.
4. The alarm should be 440 Hz +/- 1 Hz.
5. The LEDs on the timer display should have 3 brightness levels (not 4, not 2, but 3). The brightness level will be selected by a switch, potentiometer, or photosensor.

Execution

The initial planning phase was spent developing an execution plan, creating block diagrams with sub-systems and researching hardware and software possibilities. An execution plan was drafted with an additional design requirement proposal for the customer's approval. My initial proposal was later modified to better fit the difficulty of the project.

My approach to customer requirements began with breaking the system down into isolated sub-systems. I could then build and test each of these to ensure it could be incorporated into the completed system. After revision, I found three distinct sub-systems which included a 4 digit 7-segment display, interface buttons, and a light-sensitive sensor with speaker.

The largest revision occurred when implementing the brightness controls. The initial design was to use the a 7-segment Arduino library that offered brightness controls. For better functionality, the Arduino library was scrapped and replaced with the MAX7219 LED driver which allows for more control over the 7-segment display.

User Guide

The system operates on a 9V battery and will initiate a boot-up sequence. The user can the select the timer duration from the Timer button. If no selection is made, the display will flash prompting the user to select a time duration. The user can then place an object on top of the bed area and then arm the system with the press of the Power button. Finally the Start button is used to begin the timer sequence. At anytime, the user may adjust the brightness levels.

The system will sound an alarm if an object is removed from the top bed area.

Artifacts

1. Block Diagram - Used to create sub-systems and planning phases. Consistently updated with project revisions to meet changing hardware needs and requirements. These are ultimately the starting ground for planning the Pomodoro Timer project.

2. Circuit Layout - Completed in Frizting, this circuit overview indicates how the Pomodoro Timer was completed in the sub-block phase.

3. Schematic - Completed in Fritzing, this schematic shows the schematic for the Pomodoro Timer. Starting from the left, the battery is followed by the user interface buttons. These are then followed by the Arduino Mega 2560 and MAX7219. Finally the far right indicates the output, the speaker and 4 digit 7-segment display.

4. Code - Used the Arduino Mega 2560 with the LedControl.h and Bounce2.h libraries. The LedControl library was used with the MAX7219 to operated the 7-segment display. The Bounce2 library was used to improve the feeling and functionality of the buttons on the user interface.

5. 3D Model - Completed in SolidWorks, these 3D models show both the lid and case for the Pomodoro Timer. These were then sliced using Creality Slicer and finally printed on a Ender 3 printer.

6. PCB Design - Completed in Fritzing, the following shows how the system could be implemented on a customized PCB.

Code

```
// ************************************************
// Author: Christopher Dueber
// Project: Pomorodoro Timer
// Class: Junior Design II
// Date: May 14 2021
// ************************************************

#include "LedControl.h" // Segment control library
#include "Bounce2.h" // Push button control library


#define Max7219DIN 6 // (DataIN)
#define Max7219CLK 2 // (Clock)
#define Max7219CS 4 // (CSS)
#define Buzzer 8

LedControl lc=LedControl(Max7219DIN,Max7219CLK,Max7219CS,1); // 1 MAX7219 device

// ARDUINO DIGITAL PIN ASSIGNMENT
//  1                    2                    3                4
//  5                    6                    7                8
//  9                   10                   11               12
// 13                   14                   15               16
// 17                   18                   19               20
// 21                   22 - st_Clock        23               24
// 25 - select_Clock    26                   27               28
// 29                   30 - sensor_BTN      31 - Speaker     32 - pwr_State
// 33                   34                   35               36
// 37                   38                   39               40
// 41                   42                   43               44


// ARDUINO ANALOG PIN ASSIGNMENT
// 0 - brightPin        1 - sensorPin        2                3
```

```
// 4                    5                    6                7



//********************** INTERFACE INIT  ****************************

 const int brightPin = A0; // Clock Brightness analog pin
 const int select_Clock = 25; // Clock select pin (25 / 5 minutes)
 const int st_Clock = 22; // Status output
 boolean st_Clock_status = false; // Status Flag
 boolean select_Clock_status = false; // Select Flag
 int BuzzTrigger=0;

 Bounce SW1 = Bounce(); // START BUTTON
 Bounce SW2 = Bounce(); // CLOCK SELECT BUTTON

//*************************************************************************

//********************** SPEAKER INIT  ****************************

int alarmRead = 0; // Alarm value
float frequency = 0; // Alarm map

//*************************************************************************


//********************** SENSOR INIT  ****************************
const int pwr_State = 32; // Power sensor
const int sensor_BTN = 30; // Active sensor button
const int sensorPin = A1; // Sensor analog signal
int init_Light;
int tck_Light;
boolean old_State = LOW;
boolean init_State = LOW;
boolean LED_State = LOW;
```

```
//************************************************************************



//***********************  TIMER INIT   ****************************
// Digit place value for segment display
int  firstnum=0;
int  secondnum=0;
int  thirdnum=0;
int  fournum=0;
int  fivenum=0;
int  sixnum=0;
int  sevennum=0;
int  eightnum=0;

long int countnumber; // Clock time value (MM:SSSS)

//************************************************************************


//***********************  SETUP BEGIN     ****************************
//************************************************************************



int set_Clock(boolean mode){ // Clock select function
    if (mode == false){
      countnumber = 250000;
      return countnumber;
    }
    if (mode == true){
      countnumber = 50000;
      return countnumber;
    }
  }
```

```
void set_Default(){ // Bootup screen

for (int i=0; i<4; i++){
  int RowBits=B10000000;
      for (int j=0;j<8;j++){
        lc.setRow(0,i, RowBits);
        RowBits = RowBits >>1;
        delay(450);
        lc.clearDisplay(0);
      }
    }

}

void setup(){

//*************************** BOOTUP   *********************************
  Serial.begin(9600);

  lc.shutdown(0,false); // Activate MAX7219
  lc.setIntensity(0,1); // Set initial brightness to 1 (Low)
  lc.clearDisplay(0); // Clear segment display
  set_Default(); // Run bootup screen
  pinMode(Buzzer, OUTPUT); // NOT IMPLEMENTED
  digitalWrite(Buzzer, LOW); // NOT IMPLEMENTED

//*********************** INTERFACE SETUP  *****************************


  pinMode(brightPin, OUTPUT); // Brightness adjust output

  pinMode(st_Clock, INPUT_PULLUP); // Status clock (START/STOP)
  SW1.attach(st_Clock); // Set START BUTTON
  SW1.interval(3); // Delay 5ms
```

```
  pinMode(select_Clock, INPUT); // Select clock (25 / 5 minutes)
  SW2.attach(select_Clock); // Set SELECT BUTTON
  SW2.interval(3); // Delay 5ms




//*************************************************************************



//************************ SENSOR SETUP *********************************

  pinMode(pwr_State, OUTPUT); // Power sensor
  digitalWrite(pwr_State, LOW); // Init power to LOW
  pinMode(sensor_BTN, INPUT); // Light sensor
  init_Light = analogRead(sensorPin);

//*************************************************************************

}


void loop() {

SW2.update(); // LOOP BUTTON UPDATE (CHECK FOR PRESSES)
  if (SW2.fell()){
    select_Clock_status = !select_Clock_status;
    if (select_Clock_status == false){
        set_Clock(false); // IF BUTTON IS PRESSED, CHANGE TIME (25/5 minutes)
       }
    if (select_Clock_status == true){
        set_Clock(true);
    }
  }

SW1.update(); // START SWITCH
```

```
    if (SW1.fell()){
      st_Clock_status = !st_Clock_status;
    }



//************************ TIMER CODE    ******************************
if(st_Clock_status == true)
{
  for (countnumber; countnumber != -1; countnumber --)
  {
      String mystring = String(countnumber); // Transform Counter Int to String
          for manipulation

      // Convert number to a time value
      for (int z = 0; z < 6; z++){

        if ( mystring.substring(z) == "999999" ) {
          countnumber = (countnumber - 400000);
        }

        if ( mystring.substring(z) == "9999" ) {
          countnumber = (countnumber - 4000);
        }
      }

//  Display number on Display depending on number of digits remaining
  if (countnumber > 9999999) {
    firstnum = ((countnumber/10000000)%10);
    secondnum = countnumber/1000000%10;
    thirdnum = countnumber/100000%10;
    fournum = countnumber/10000%10;
    fivenum = countnumber/1000%10;
    sixnum = countnumber/100%10;
    sevennum = countnumber/10%10;
    eightnum = countnumber%10;
```

```
    lc.setDigit(0,7,firstnum,false);
    lc.setDigit(0,6,secondnum,false);
    lc.setDigit(0,5,thirdnum,false);
    lc.setDigit(0,4,fournum,false);
    lc.setDigit(0,3,fivenum,false);
    lc.setDigit(0,2,sixnum,false);
    lc.setDigit(0,1,sevennum,false);
    lc.setDigit(0,0,eightnum,false);
  }
    else {
      if (countnumber > 999999) {
//      firstnum = ((countnumber/10000000)%10);
        secondnum = countnumber/1000000%10;
        thirdnum = countnumber/100000%10;
        fournum = countnumber/10000%10;
        fivenum = countnumber/1000%10;
        sixnum = countnumber/100%10;
        sevennum = countnumber/10%10;
        eightnum = countnumber%10;

        lc.setChar(0,7,'-',false);
        lc.setDigit(0,6,secondnum,false);
        lc.setDigit(0,5,thirdnum,false);
        lc.setDigit(0,4,fournum,false);
        lc.setDigit(0,3,fivenum,false);
        lc.setDigit(0,2,sixnum,false);
        lc.setDigit(0,1,sevennum,false);
        lc.setDigit(0,0,eightnum,false);
      }
        else {
          if (countnumber > 99999) {
//          firstnum = ((countnumber/10000000)%10);
//          secondnum = countnumber/1000000%10;
            thirdnum = countnumber/100000%10;
            fournum = countnumber/10000%10;
            fivenum = countnumber/1000%10;
            sixnum = countnumber/100%10;
```

```
            sevennum = countnumber/10%10;
            eightnum = countnumber%10;

            lc.setChar(0,7,'-',false);
            lc.setChar(0,6,'-',false);
            lc.setDigit(0,0,thirdnum,false);
            lc.setDigit(0,1,fournum,false);
            lc.setDigit(0,2,fivenum,false);
            lc.setDigit(0,3,sixnum,false);
            lc.setDigit(0,4,sevennum,false);
            lc.setDigit(0,5,eightnum,false);
          }
            else {
              if (countnumber > 9999) {
//              firstnum = ((countnumber/10000000)%10);
//              secondnum = countnumber/1000000%10;
//              thirdnum = countnumber/100000%10;
                fournum = countnumber/10000%10;
                fivenum = countnumber/1000%10;
                sixnum = countnumber/100%10;
                sevennum = countnumber/10%10;
                eightnum = countnumber%10;

                lc.setChar(0,7,'-',false);
                lc.setChar(0,6,'-',false);
                lc.setChar(0,5,'-',false);
                lc.setDigit(0,0,fournum,false);
                lc.setDigit(0,1,fivenum,false);
                lc.setDigit(0,2,sixnum,false);
                lc.setDigit(0,3,sevennum,false);
                lc.setDigit(0,4,eightnum,false);
              }
              else {
                if (countnumber > 999) {
 //               firstnum = ((countnumber/10000000)%10);
 //               secondnum = countnumber/1000000%10;
 //               thirdnum = countnumber/100000%10;
```

```
    //              fournum = countnumber/10000%10;
                    fivenum = countnumber/1000%10;
                    sixnum = countnumber/100%10;
                    sevennum = countnumber/10%10;
                    eightnum = countnumber%10;

                    lc.setChar(0,7,'-',false);
                    lc.setChar(0,6,'-',false);
                    lc.setChar(0,5,'-',false);
                    lc.setChar(0,4,'-',false);
                    lc.setDigit(0,0,fivenum,false);
                    lc.setDigit(0,1,sixnum,false);
                    lc.setDigit(0,2,sevennum,false);
                    lc.setDigit(0,3,eightnum,false);
                  }
                  else {
  //                firstnum = ((countnumber/10000000)%10);
  //                secondnum = countnumber/1000000%10;
  //                thirdnum = countnumber/100000%10;
  //                fournum = countnumber/10000%10;
  //                fivenum = countnumber/1000%10;
                    sixnum = countnumber/100%10;
                    sevennum = countnumber/10%10;
                    eightnum = countnumber%10;

                    lc.setChar(0,7,'-',false);
                    lc.setChar(0,6,'-',false);
                    lc.setChar(0,5,'-',false);
                    lc.setChar(0,4,'-',false);
                    lc.setChar(0,3,'-',false);
                    lc.setDigit(0,0,sixnum,false);
                    lc.setDigit(0,1,sevennum,false);
                    lc.setDigit(0,2,eightnum,false);
               }

             }
           }
```

```
      }
    }
// BUZZER NOT IMPLEMENTED ** TOO ANNOYING
// DELAYS USED TO ENSURE CORRECT TIME ~ 1 SEC
// If one second as gone by sound buzzer
    if (BuzzTrigger == 99){
        //digitalWrite (Buzzer, HIGH) ;// Buzzer On
        delay (9) ;// Delay 2ms
        //digitalWrite (Buzzer, LOW) ;// Buzzer Off
    BuzzTrigger = 0; // Trigger for countdown sound
    }
    else {
      delay (5.9);
      BuzzTrigger = BuzzTrigger + 1;
    }

// If countdown at zero sound alarm and flash display
    if (countnumber == 0) {
      for (int y = 0; y < 8; y++){
        for (int x = 0; x < 8; x++)
        {
          lc.setDigit(0,x,0,false);
        }
        delay (100) ;// Delay 1ms
        for (int x = 0; x < 8; x++)
        {
          lc.setChar(0,x,'-',false);
        }
        delay (300) ;// delay 1ms
      }
    }



//**************************************************************************

//********************** DIMMER CODE  ******************************
```

```
{
int pot_Value = analogRead(A0);
int bright_Level = map(pot_Value, 0, 1023, 0, 15); // Brightness range from 0 -
    15
lc.setIntensity(0, bright_Level);
}

//*************************************************************************

//*********************** SPEAKER CODE  ****************************
{
  alarmRead = analogRead(sensorPin); // ANALOG SIGNAL FOR SPEAKER
  frequency = map(alarmRead, 1, 25, 440, 440); // MAP VALUES 1 - 25, PLAY 440
      Hz if > 15
  if(alarmRead > 15){
    tone(31, frequency);
    }
  if(alarmRead < 15){
    noTone(31);
  }
}
//*************************************************************************


//************************** SENSOR CODE   ****************************

{
  {
   init_State = digitalRead(sensor_BTN); // SENSOR BUTTON
    if (init_State != old_State) // FLAG STATE
    {
       if ( init_State == HIGH )
       {
           if (LED_State == LOW) {
             digitalWrite(pwr_State, HIGH); // TURN ON POWER TO ALARM
             LED_State = HIGH;
            }
```

```
          else {
            digitalWrite(pwr_State, LOW); // TURN OFF POWER TO ALARM
            LED_State = LOW;
            }
       }
       old_State = init_State; // FLAG STATE
    }
  }

  tck_Light = analogRead(sensorPin); // ANALOG SIGNAL FOR ALARM MAP
  Serial.println(tck_Light);

  if (tck_Light < init_Light - 50){ // RANGE FOR LIGHT LIMIT FOR ALARM
    digitalWrite(9, HIGH);
    }
  else {
      digitalWrite(9, LOW);
    }
   }
  }
 }
}
//*************************************************************************
```