Lab #3 – *Pomodoro Study-bot*

COURSE:	MECH 215 – Programming for Mechanical Engineers
SEMESTER:	Fall 2018
PROFESSOR:	Dr. Brandon W. Gordon
STUDENT(S):	Joseph El-Nouni
STUDENT ID:	40092412
DATE:	28/11/2018

I CERTIFY THAT THIS SUBMISSION IS MY ORIGINAL WORK AND MEETS THE FACULTY'S EXPECTATIONS OF ORIGINALITY.

Table of Contents

Overview

Project Goal

The goal of this project is the creation of a *robotic study-assistant*, which helps students in their study time by automating and enforcing techniques known to improve study efficiency. Mainly, it employs the use of the Pomodoro technique, and it keeps the desk environment study-only.

Unit Features

In order to accomplish its goal, the Study-bot unit comes with the following features:

- **An LCD Character Display**, used to displays timers and options to the user.
- **An Ultrasonic Distance Sensor**, which detects the user's presence at the desk.
- **A Buzzer**, which allows the unit to notify the user of any events or changes.
- **Push-Buttons**, which allow the user to set up the robot and make decisions.

Typical Use-case Scenario

In a typical scenario, the user will begin by turning on the robotic unit.

Once turned on, the robot will give the user a chance to set up the Pomodoro cycle durations (work minutes, then break minutes) using the buttons on the front.

The cycle begins here. Once set, the robot will detect whether the user is seated before beginning its countdown, which will be displayed on the screen.

When the countdown for "work time" ends, the robot will sound an alarm. Then, it will wait until the user leaves before starting the next countdown. The user is not permitted to sit at their work desk for "break time".

Finally, once “break time” has ended, the robot will sound an alarm which must be stopped by the user. From here, **the cycle repeats.**

As a bonus feature, the user may interrupt the robot while it is counting down. (For now, only a placeholder feature. It works; just doesn’t do much).

Hardware

Design

The hardware was designed to include the least amount of electronic circuitry possible. The point was to stray as little as possible from the course material, while still keeping it original.

Implementation

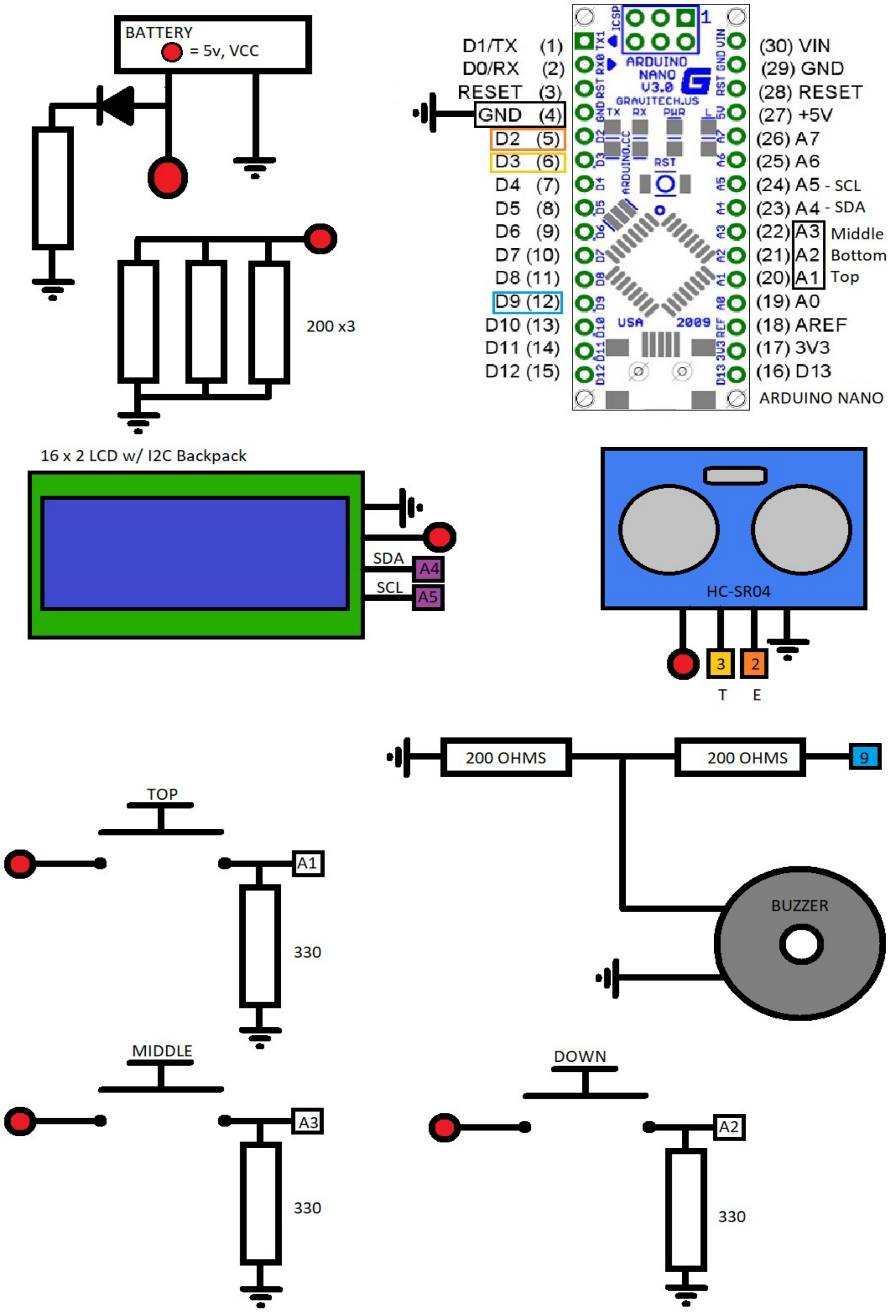
The following parts were used for this project:

Design Element	**Part Name & Model**
LCD Screen	RoboJax 16x2 Character LCD w/ I2C Interface Shield
Distance Sensor	HC-SR04 Ultrasonic Sensor
Brain Board	Arduino Nano board, ATMega328P MCU
Battery	PowerAdd Slim2 (5v, 5000mAh)

Other (more discrete) Parts Included:

Qty.	**Part Type**
1	Generic PWM Buzzer
3	200 Ohm Resistor
1	LED

The parts were assembled according to the following schematic:



Details:

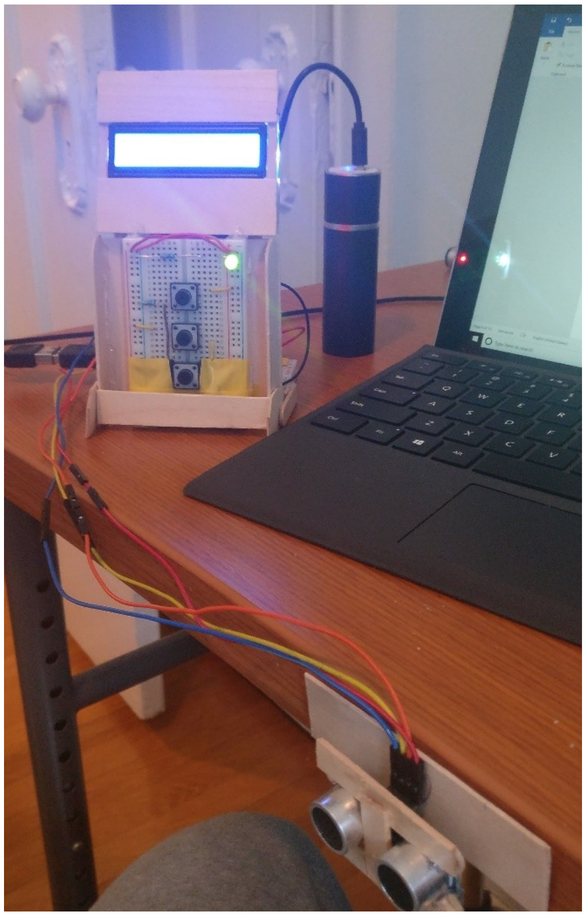
- Resistors were attached in parallel to the battery in order to generate a 70mA load.
  Without it, the battery will turn off when it can no longer detect the circuit's activity.

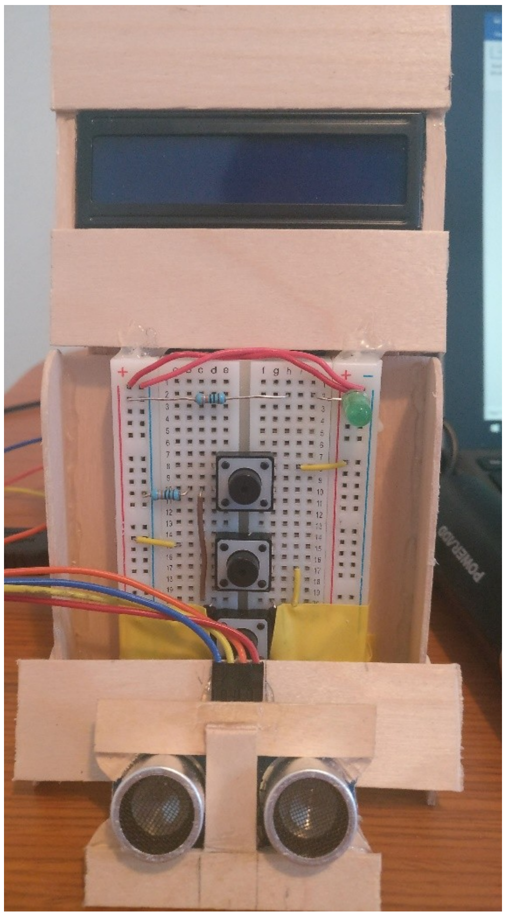
- A voltage divider-circuit was used in order to reduce the volume emitted by the buzzer.

- In order for the pushbuttons to be detected properly, a pull-down resistor was attached to the signal going to the MCU, ensuring that an undressed button will yield a measured value of 0, and not a floating, erratic value.

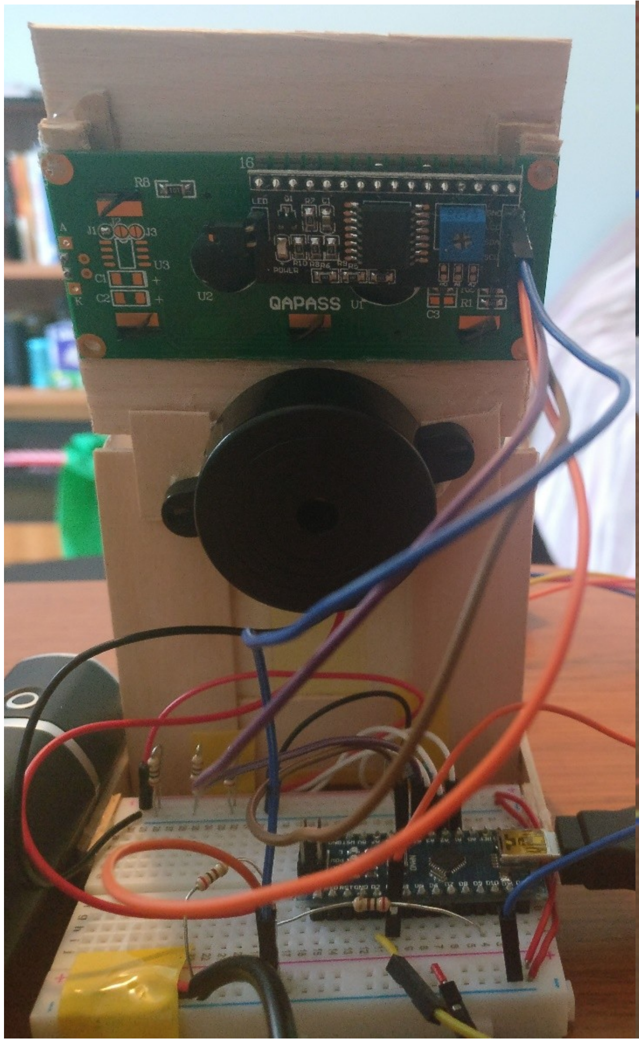
- Finally, the frame that holds it together was made from wooden ice cream sticks (wide AND narrow ones).

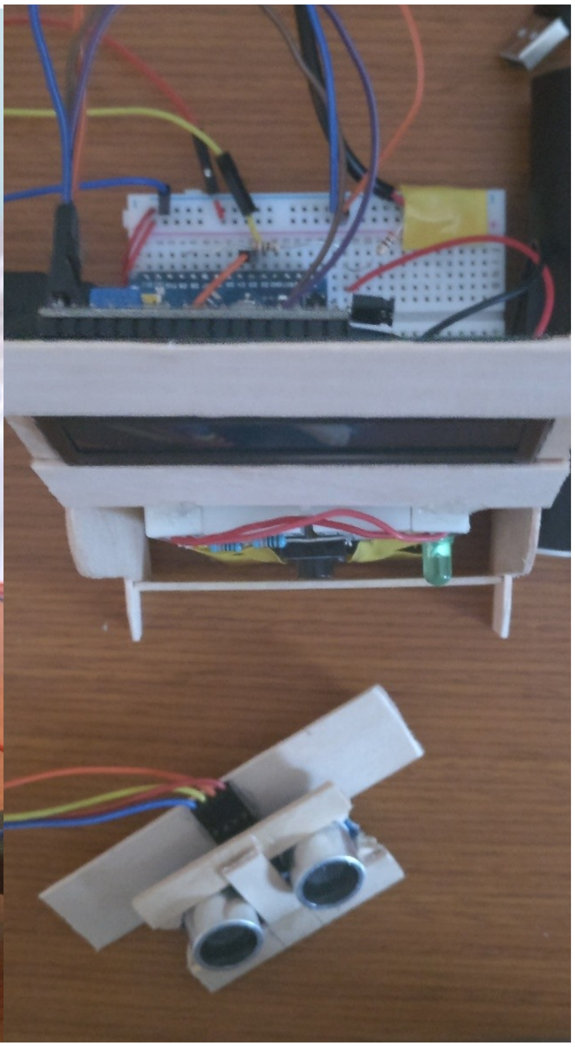
Pictures:





Typical Usage **Front View**





Back View

Top view

The robot stands on its own over a desk with ease.

While the wires seem excessive, they could be easily covered up.

Firmware

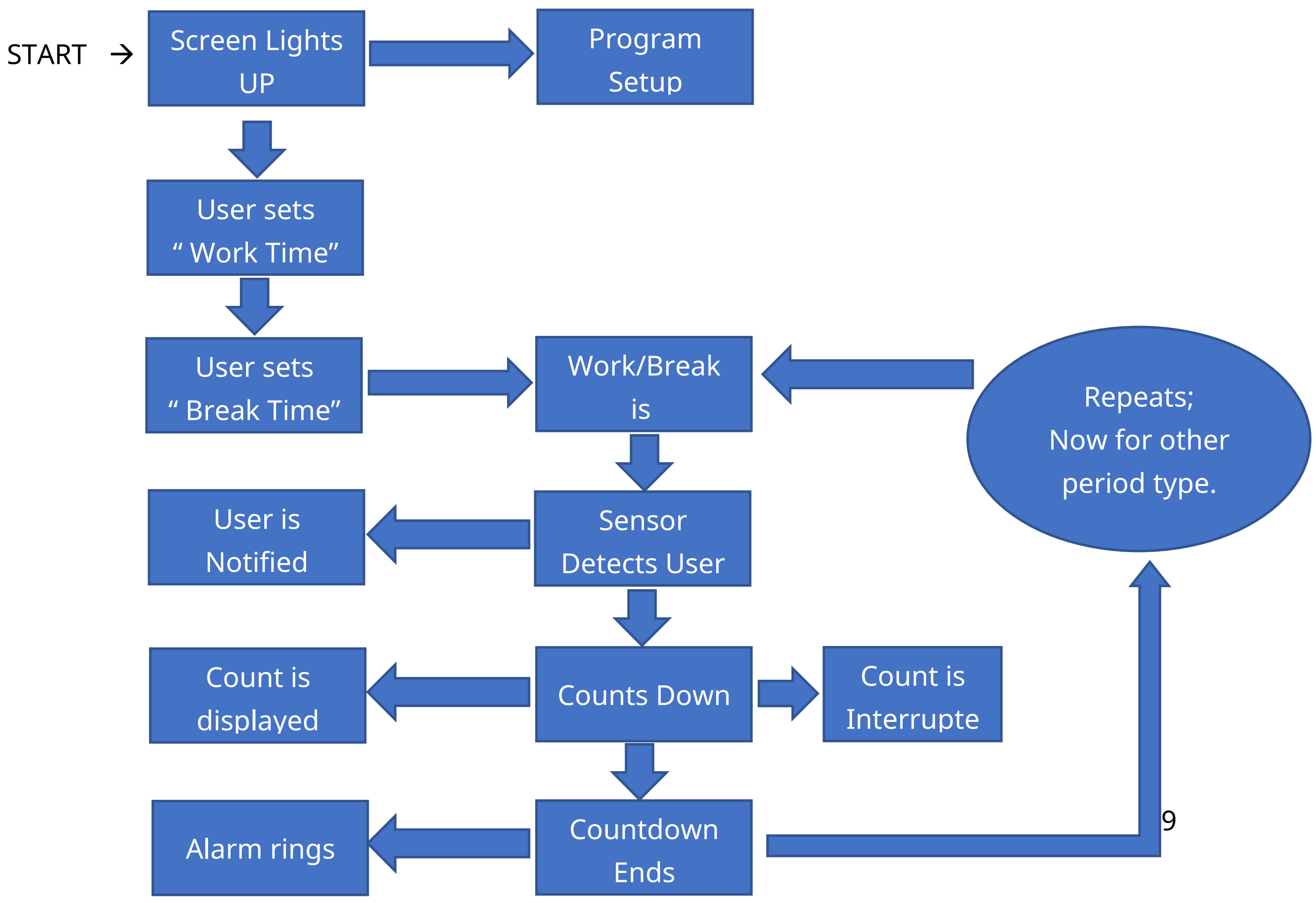
Design

The software was designed with the following intentions:

- To allow the user to set their own Work/Break times;
- To allow the robot to sense whether the user is seated;
- To count down the specified number of minutes for the specific period type (work or break);
- To notify the user of the end of every period;
- To allow the user to interrupt the aforementioned countdown;
- To allow the above to be displayed on the LCD screen throughout.

As such, it followed the model depicted in this sequence diagram:

Front End (What the user can see/touch)	**Software Layer 1**	**Software Layer 2**



Implementation

Programming Methodology:

In order to finish programming this multi-part project, I used a miniaturized SCRUM list which allowed to integrate and troubleshoot bit of my program in an orderly fashion.

Here’s the order I followed:

1. Get the screen to show a message.
2. Get the screen to detect button presses.
3. Get the buzzer to beep with every button press.
4. Get the screen to show which button I am pressing.
5. Get the screen to show a distance measurement.
6. Integrate a function to select the number of minutes for Work/Break.
7. Integrate a button selection routine in the Loop with the LCD.
8. Integrate a Integrate a countdown routine in the Loop.
9. Get the screen to display the countdown.
10. Integrate interruptions into the countdown routine.
11. Integrate an alarm at the end of the countdown.
12. Integrate a blocking-polling distance check before the countdown routine.
13. Troubleshoot number related problems (delays, negative time values, etc.)

I had the Loop() function handle the most important tasks which required real-time inputs from (or results displayed to) the user.

This included the Time selection, Seat Detection and the Countdowns.

The following are the other functions that were included, with a brief description for each:

bool check_button();   //checks if a button is pressed. Updates "CURRENT_BTN"

bool set_work(char btn);   //increments work minutes according to button press. True when middle button is pressed

```
bool set_break(char btn);     //increments breakMinutes. Returns 1 when middle button is pressed

bool wselected = false;             //1 = work time has been selected

bool bselected = false;             //1 = breakTime has been selected

void display_sel_time(int t);  //Displays time selection as it increments

void display_countdown();    //Displays time left in countdown

void options();                     //Displays options. Very simplified placeholder.
```

Results

Performance:

Now complete, the robot performs adequately. While the program that it now runs (as well as its hardware, admittedly) is much simpler than what had first been intended, it is still a perfectly usable, functioning piece of engineering.

As it starts, it allows you to select the Work time and the Break time, after which it will request that you sit and will start counting down the "work time". During that time, any button press will interrupt the timer with the placeholder "Options function", which hints at a multitude of potential improvements. Once the timer ends, an alarm sounds. The robot waits for the user to leave, and then proceeds with the break timer. Finally, once the break time is over, the alarm sounds once more, and the cycle repeats.

To me, this is a more than adequate level of performance, considering we've really only worked on this project for about a month at the best (not to mention that it isn't a full-time endeavor, either).

Troubleshooting:

I did run into a few problems:

Problem	Solution
The USB Battery would turn off after a few seconds.	This was because of the program sometimes pausing and thus not using up enough current. The battery thus turns off to conserve power. Placing three resistors in parallel to the source allowed me to draw enough current to keep it on at all times.
The Pushbuttons were not being detected.	Hardware: They were placed upside down. The fix was brief. Software: I needed to make sure my global variables were being reset so as to not instantly break certain loops upon the functions repeating.
Text displayed but with Jitter	This is due to two print commands running at the same time at very fast speeds (when reading the buttons). Delays (debouncing) were added so the problem would stop.

References

[1] https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-nano.html

[2] http://startingelectronics.org/articles/arduino/connecting-buzzer-arduino-uno/

[3] https://www.reddit.com/r/arduino/comments/2jdzto/arduino_with_a_usb_battery_pack_arduino_turns_off/

[4] https://www.arduino.cc/reference/en/

[5] http://haneefputtur.com/controlling-lcd-from-push-button-using-arduino.html

[6] https://github.com/duinoWitchery/hd44780

Appendix A: Program Listing

```
#include <HCSR04.h>                         //Ultrasonic Sensor Library
#include <Wire.h>
#include <hd44780.h>                        // HD4480 allows interfacing for various LCD shields
#include <hd44780ioClass/hd44780_I2Cexp.h> // i2c expander i/o class header

/*DEFINES*/
#define UP      0x1
#define DOWN    0x2
#define MID     0x3

int echo = 2;
int trig = 3;
int up_btn = 15;
int mid_btn = 17;
int down_btn = 16;
int buzz = 9;       //PIN value declarations. Different from other variables.
/*********/


/*PROTOTYPES*/
bool check_button();            //checks if a button is pressed. Updates "CURRENT_BTN"
bool set_work(char btn);        //increments work minutes according to button press. Returns 1 when middle button is pressed
bool set_break(char btn);       //increments breakMinutes according to button press. Returns 1 when middle button is pressed
bool wselected = false;         //1 = work time has been selected
bool bselected = false;         //1 = breakTime has been selected
void display_sel_time(int t);   //Displays time selection as it increments
void display_countdown();       //Displays time left in countdown
void options();                 //Displays option. Very simplified placeholder.
/************/


/*VARIABLES*/
const int LCD_COLS = 16;
const int LCD_ROWS = 2;         //Rows and Cols used to initialize LCD object.
char CURRENT_BTN;               //contains value of "last pressed button".
bool pressed;                   //True if a button press has been detected by check_button()
bool proceed = false;           //will be used to break loops.
hd44780_I2Cexp lcd;             //lcd object
UltraSonicDistanceSensor sensor(trig, echo);  //HC-SR04 sensor
int work_mins;                  //selected work/break minutes
int break_mins;
int minutes;                    //minutes to count down
int ten_secs;                    //tens of seconds to count down
int seconds;                     //unit seconds to count down
/***********/



void setup() {

  /*PINS*/
  pinMode(up_btn, INPUT);   //UP button
  pinMode(mid_btn, INPUT);  //CENTER button
  pinMode(down_btn, INPUT); //DOWN button
  pinMode(echo, INPUT);     //echo pin
  pinMode(trig, OUTPUT);    //trig pin
  pinMode(buzz, OUTPUT);    //buzzer
  /******/

  int status;

  status = lcd.begin(LCD_COLS, LCD_ROWS);   // initialize LCD with number of columns and rows: returns error code
                                            // LCD pins are SDA and SCL, initialized from the library
  if (status)                               // non zero status means it was unsuccesful
  {
    status = -status;                       // convert negative status value to positive number
                                            // begin() failed so blink error code using the onboard LED if possible
    hd44780::fatalError(status);            // does not return
  }

  lcd.print("Greetings!");                  //initializing message
  delay(1000);
  lcd.clear();





}

void loop() {

  //Obtaining "work time":
  lcd.setCursor(0, 0);
  lcd.print("Work Time:");            //Notifies user
  display_sel_time(0);
  while (wselected == false)
  {
    if (check_button() == 1)
    {
      wselected = set_work(CURRENT_BTN);  //breaks loop if MID is pressed.

      delay(500);                         //So it doesnt increment 50 times per press.

      display_sel_time(work_mins);
    }
  }

  //Obtaining "break time": See above
        lcd.setCursor(0, 0);
        lcd.print("Break Time:");
        display_sel_time(0);
        while (bselected == false)
        {
          if (check_button() == 1)
          {
            bselected = set_break(CURRENT_BTN);

            delay(500);

            display_sel_time(break_mins);
          }
        }

        //Work Countdown starts:

        minutes = work_mins;
        if (minutes = < 0)   //Minimum is 1 minute, otherwise it will count down into the negatives.
          minutes = 1;
        ten_secs = 0;
        seconds = 0;

        while (1) // Cycle should continuously repeat.
        {
          //Check that user is SEATED:
          do {
            lcd.clear();
            tone(buzz, 2000);
            delay(1000);
            noTone(buzz);
            lcd.print("Please Sit!");               //Notifies
            delay(5000);
          } while (sensor.measureDistanceCm() > 30);//If further than 30cm away from table, will wait.

          lcd.clear();
          //Countdown:
          while (proceed == false)
          {
```

```
      display_countdown();
      delay(333);                           //This portion repeats 3 times per second
                                            //This makes sure it never misses button press.
      if (check_button() == true) {
        options();                          //Checking the Options PAUSES the countdown.
        delay(100);
      }
      delay(333);

      if (check_button() == true) {
        options();
        delay(100);
      }
      delay(333);

      if (check_button() == true) {
        options();
        delay(100);
      }

      seconds--;                           //After 1s, unit seconds go down by 1.

      if (seconds < 0)                     //If they go to -1, +=10 to get it back to 9.
      {
        seconds += 10;
        ten_secs--;                        //In that case, the "tens of seconds" lose 1.
      }

      if (ten_secs < 0)                    //Same thing as with the unit seconds, only now the limit is 5, not 9.
      {
        ten_secs += 6;
        minutes--;
      }


      if (minutes == 0)
      {
        if (ten_secs == 0)
        {
          if (seconds == 0)
            proceed = true;
        }
      }
    }

```

```
        tone(buzz, 2500);                   //Buzzer alarm sequence
        delay(1000);
        noTone(buzz);
        delay(200);
        tone(buzz, 2500);
        delay(1000);
        noTone(buzz);
        delay(200);
        tone(buzz, 2500);
        delay(1000);
        noTone(buzz);
        delay(200);

        proceed = false;      //so it can be reused.


        minutes = break_mins; //It's a repeat of he above, only now we are using the break minutes.
        if (minutes = < 0)
          minutes = 1;

        ten_secs = 0;
        seconds = 0;


        do {
          lcd.clear();
          tone(buzz, 2000);
          delay(1000);
          noTone(buzz);
          lcd.print("Break Time!");
          delay(5000);
        } while (sensor.measureDistanceCm() < 30);   //Another difference is the need to be AWAY from the desk now.

        lcd.clear();


        while (proceed == false)
        {
          display_countdown();
          delay(333);

          if (check_button() == true) {
            options();
            // execute_opt(option);
            delay(100);
          }
          delay(333);

          if (check_button() == true) {
            options();
            // execute_opt(option);
            delay(100);
          }
          delay(333);

          if (check_button() == true) {
            options();
            // execute_opt(option);
            delay(100);
          }

          seconds--;

          if (seconds < 0)
          {
            seconds += 10;
            ten_secs--;
          }

          if (ten_secs < 0)
          {
            ten_secs += 6;
            minutes--;
          }


          if (minutes == 0)
          {
            if (ten_secs == 0)
            {
              if (seconds == 0)
                proceed = true;
            }
          }
        }

        tone(buzz, 2500);
        delay(1000);
        noTone(buzz);
        delay(200);
        tone(buzz, 2500);
        delay(1000);
        noTone(buzz);
        delay(200);
        tone(buzz, 2500);
        delay(1000);
        noTone(buzz);
        delay(200);


      }




      lcd.clear(); //Just in case the loop breaks somehow.
      while (1);

    }


    bool check_button()
    {
      pressed = false;

      if (digitalRead(up_btn) == 1)
      {
        delay(150);
        if (digitalRead(up_btn) == 1)
        {
          tone(buzz, 2500);
          delay(100);
          noTone(buzz);
          CURRENT_BTN = UP;
          pressed = true;
          return pressed;
        }
      }

      if (digitalRead(down_btn) == 1)
      {
        delay(150);
        if (digitalRead(down_btn) == 1)
        {
          tone(buzz, 2500);
          delay(100);
          noTone(buzz);
          CURRENT_BTN = DOWN;
          pressed = true;
          return pressed;
        }
      }

      if (digitalRead(mid_btn) == 1)
      {
        delay(150);
        if (digitalRead(mid_btn) == 1)
        {
          tone(buzz, 2500);
          delay(75);
          noTone(buzz);
          delay(75);
          tone(buzz, 2500);
          delay(75);
          noTone(buzz);
          CURRENT_BTN = MID;
          pressed = true;
          return pressed;
        }
      }
      return pressed;
    }




    bool set_work(char btn)
    {
      bool middle = false;

      switch (btn)
      {
        case UP:
          work_mins += 5;

          if (work_mins > 50)
            work_mins = 50;

          break;
        case DOWN:
          work_mins -= 5;
          if (work_mins < 0)
            work_mins = 0;

          break;
        case MID:
          middle = true;
          break;

        default:
          break;
      }

      return middle;
    }

    bool set_break(char btn)
    {
      bool middle = false;

      switch (btn)
      {
        case UP:
          break_mins += 5;
          if (break_mins > 50)
            break_mins = 50;
```

```

            break;
          case DOWN:
            break_mins -= 5;
            if (break_mins < 0)
              break_mins = 0;

            break;
          case MID:
            middle = true;
            break;

          default:
            break;
        }

        return middle;
      }

      void display_sel_time(int t)
      {
        lcd.setCursor(0, 1);
        lcd.print(t);
        lcd.print(" mins   ");
      }


      void display_countdown()
      {
        lcd.clear();

        if (minutes < 10)
          lcd.setCursor(6, 0);
        else
          lcd.setCursor(5, 0);

        lcd.print(minutes);
        lcd.print(":");
        lcd.print(ten_secs);
        lcd.print(seconds);
      }



      void options()
      {
        /* pressed == false;
          CURRENT_BTN = 0x0;
          bool mid = false;
          lcd.clear();
          lcd.print("OPTIONS:");
          display_option(1);

          while (mid == false);
          {

           if (check_button() == 1) {
             mid = set_opt(CURRENT_BTN);
             delay(500);
             display_option(option);

           }
          }       */
        lcd.clear();
        lcd.print("OPTIONS ~");
        delay(2000);
      }
```