

System Overview: The egg timer has the ability to choose between two different timer settings, 4 minutes and 10 minutes, has 3 discrete brightness settings and has the ability to add a minute in the middle of the countdown. The select time function and the brightness function are both controlled with knobs. The knobs are rotary potentiometers. These function similarly to the radio tuner or volume control on your car stereo. The timer can be started with the press of a button. The user can add a minute with the press of another button. All these inputs go straight to the Arduino Nano, which is the brain of the system. The Arduino Nano controls all the outputs and takes in all the inputs. A C script running on it polls (periodically checks in a loop) for inputs. The JD Power supply from Junior Design I steps down the wall power to 5V, small enough for the Nano.

Electrical Specifications:

Max voltage supplied to Arduino: 6V
Max voltage supplied from wall plug: 12V
Minimum Voltage: -0.5V
Max current: 800 mA
Nominal current: 40 mA
Max temperature: 125 C
Normal temperature: 21 C
Min temperature:-55 C

User Guide:

1. Open up the enclosure and make sure all the pins are hooked up correctly. Use the schematic provided below to check.
2. Make sure the external wall jack is connected to the back of the timer. Plug in the 12 V DC converter into a type A or B outlet.
3. If the display doesn't turn on still, that means that emergency power switch has been flicked off. Flip on the switch and the seven segment display will light up.
4. Now the timer should be on and fully functional. Use the 'Select Time' knob to choose between 4 and 10 minutes.
5. Use the 'Dimmer' knob to choose between three brightness levels. Once you select your time, hit the 'Start/Stop' button to begin the count down.
6. If you want to add an extra minute as the timer is counting down, press the 'Add Minute' button. Do not continuously press down on this button. Just press it and then release quickly.
7. To reset the time, turn off and on the power switch.
8. When the speaker goes after after the count down is finish, press 'Start/Stop' to get it to stop and to reset the time.
9. You can also reset the time in the middle of the countdown by flicking off and on the emergency power switch.

Design artifacts:

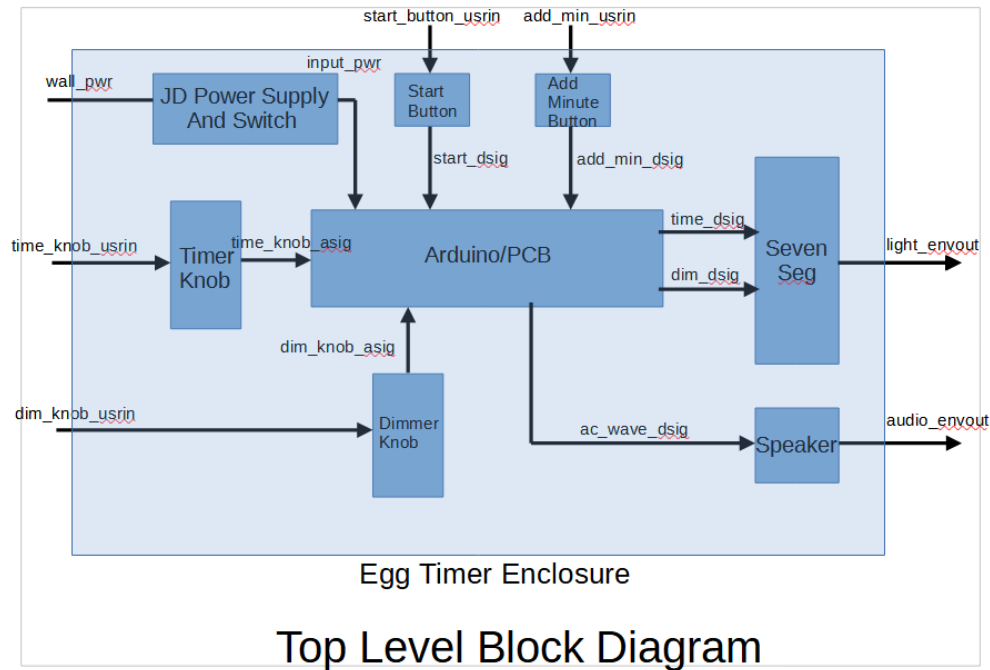


Figure 1: This is a diagram to understand the inputs and outputs of the system. It is broken down into blocks so that even a non ECE student can understand how everything flows. Power input comes from the wall and gets stepped down before it reaches the Arduino. User inputs can come from four different places They all travel to the Arduino/PCB. These two things are combined into a single block given that the PCB only serves as an easy way for the Arduino to read the incoming ADC values. Finally, there are three outputs directed to two different places. The seven segment display takes the dimming signal and the write matrix signal, which the speaker just takes in a PWM wave (a digital wave that goes between only 0 and 1). The table below has the list of interfaces (light_envout, start_dsig, etc, which you can see in block diagram above) and their properties.

Interface	Interface Code	Properties
12 V wall power	wall_pwr	- DC voltage of 12 V. - Max Voltage 12 V. - Min Voltage 5 V. - Max Current: 800 mA.
5 V Input Power	input_pwr	- Connects to Arduino Nano. - Max Voltage: 5 V. - Can be flipped off with an emergency power switch. - Max Current: 800 mA
Timer Knob	time_knob_usrin	- Turns 300 degrees. - A potentiometer of resistance 10 kOhms. - Controls the timer value (2 discrete options for time).
Dimmer Knob	dim_knob_usrin	- Turns 300 degrees. - A potentiometer of resistance 10 kOhms. - Controls the brightness with 3

		distinct levels.
Start Button	start_button_usrin	• Small push button which releases a logic 0 when not pushed and logic 1 when pushed (could also be active low). • Button made of plastic and made to fit the enclosure. Should be around 1 cm in diameter or less.
Add Minute Button	add_min_usrin	• Small push button which releases a logic 0 when not pushed and logic 1 when pushed (could also be active low). • Button made of plastic and made to fit the enclosure. Should be around 1 cm in diameter or less.
Timing Signal	time_knob_asig	• An analog voltage signal between 0V and 5V. • Below 2.5 V will be interpreted as a logic 0 by the Arduino. • Above 2.5 V will be interpreted as a logic 1 • Signal blocked once timer is running.
Dimming Signal	dim_knob_asig	• An analog voltage signal between 0V and 5V. • Below 1.67 V will be interpreted as a 00, dimmest level. • Below 3.33 V will be interpreted as a 01, middle brightness. • Above 3.33 V will be interpreted as a 10, highest brightness.
Start Signal	start_dsig	• Digital Signal; Negative edge triggered. • Starts timer. • Input blocked once timer is running.
Add Minute Signal	add_min_dsig	• Digital Signal; Negative edge triggered. • Input is blocked if timer is not running. • Input is taken if timer is running. • A minute is added to the timer when negative edge received.
Seven Segment Time Signal	time_dsig	• Digital Signal • Four separate signals, each one connected to a seven seg block. • Each signal controls one block.
Seven Segment Dimming Signal	dim_dsig	• Digital Signal • There are three possible values for the signal 00, 01 and 10 which correspond to the brightness levels.

		This signal goes to all the seven seg blocks and dims them equally.
Speaker Wave	ac_wave_dsig	- Digital Signal coming out of the Arduino Pin 8. - Must have a frequency of 440 +/- 1 Hz.
Speaker Audio	audio_envout	The frequency must have an A tone (440 +/- 1 Hz).
Seven Segment Number Display	light_envout	- Has three different and distinct brightness. - Outputs yellow light. (subject to change) - The outputted light should tell the user how much time is remaining.

Table 1: Interface Definitions

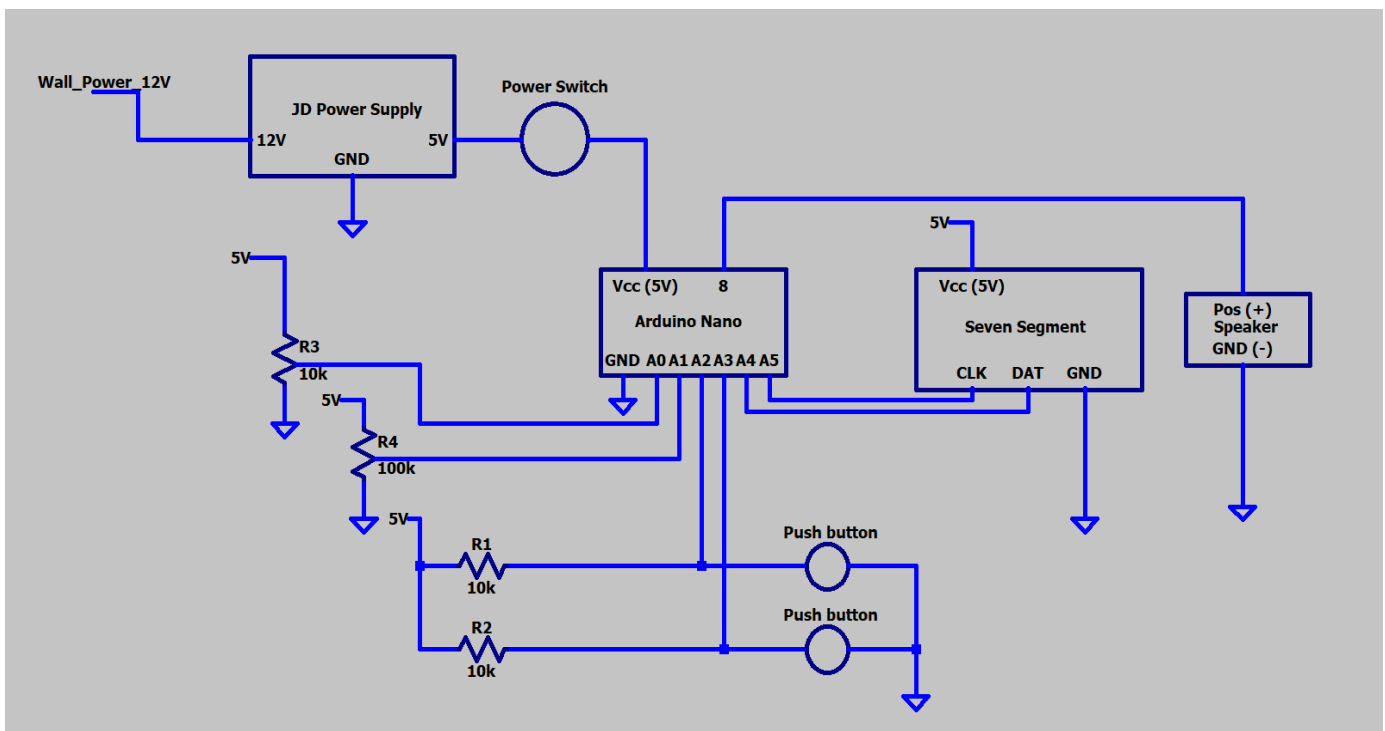


Figure 2: The schematic is similar to the block diagram in that it shows much of the same blocks. However, it shows another level of complexity. There are the individual pins on the blocks (so it's now clear where each signal is going), specific resistor values, and specific voltage values. R3 and R4 are variable resistors with three terminals. One terminal going to 5V, the middle terminal reading off a voltage to a connection (in this case a pin) and the last terminal going to 0V. One thing this diagram doesn't show is that all the 5V sources are tied together and all GNDs (which means 0V and are represented by the triangle symbols) are tied together too.

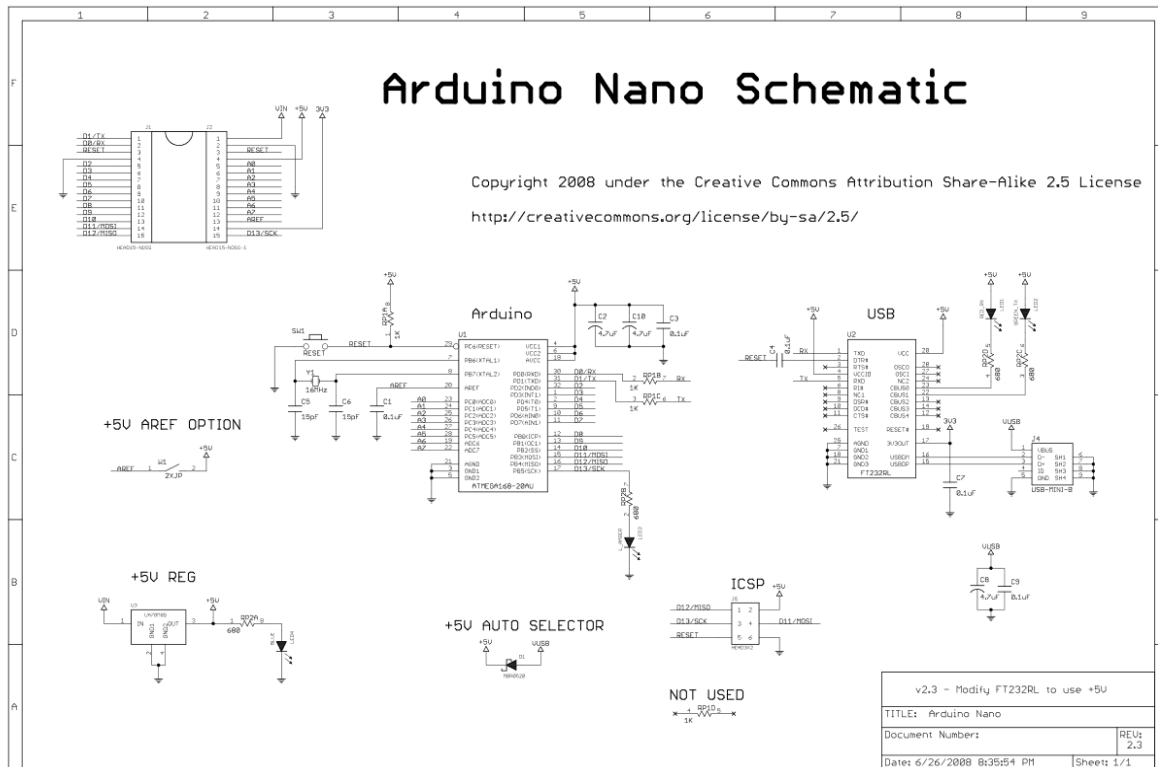


Figure 3: Arduino Nano Schematic.

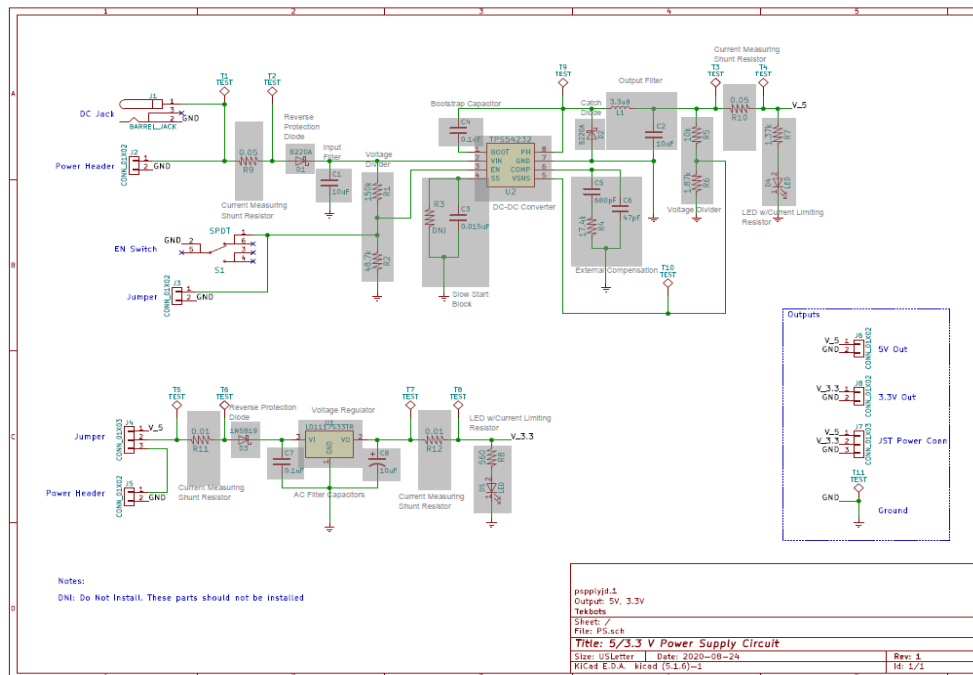


Figure 4: JD Power Supply Schematic. It shows the individual chips, resistors and capacitors. There are sub-blocks labeled in the diagram.

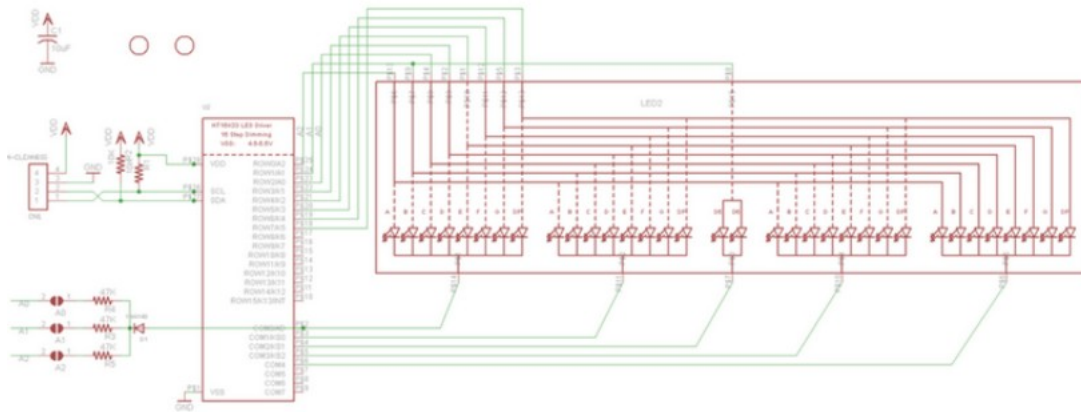


Figure 5: Seven Segment Display Schematic (sorry it's a little hard to read)

Taking things to one more level of complexity, these are the schematics for the big blocks like the Arduino, Power Supply and Matrix Display. At this level of detail, one can see individual resistors, chips, registers, etc that make up these blocks.

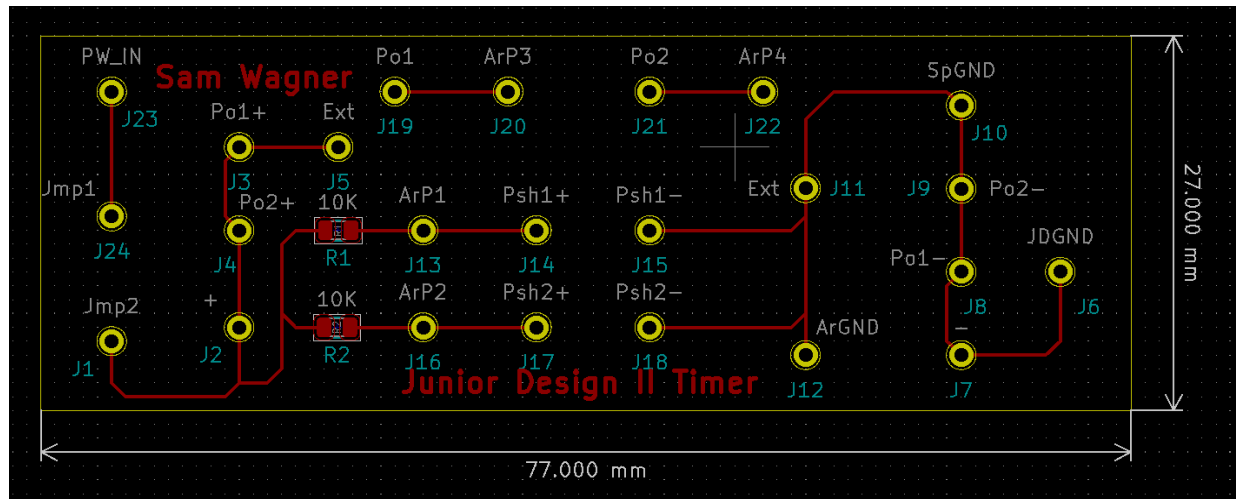


Figure 4: (PCB Layout) The PCB is 27x77mm which is around 3 square inches. It was printed by oshPark. There are two sides to the design, the 5V side and the GND side. The PCB was essentially designed so that it would be a replacement breadboard. However, it can accommodate 2 push buttons with spots for 2 pullup resistors (R1 and R2). There are 24 pins, most of which were used. J11 and J10 turned out to be extra. Because of a design flaw, J19-J22 are also not used. It was discovered these pins were not needed. The PCB still works just fine with these small errors. This is the only important layer, since the backside is just full of through holes. Below is a list of pins and their uses:

- J1: Jumper to supply 5V to the circuit. Lower terminal of emergency switch connected here.
- J2: Goes to 5V on the seven segment display.
- J3: Goes to 5V of the dimmer potentiometer.
- J4: Goes to 5V of the timer select potentiometer.
- J5: Extra. Used in this case to supply 5V to the Arduino Nano.
- J6: Connect to JD Power Supply GND.
- J7: Goes to GND of the seven segment display.
- J8: Goes to GND of dimmer potentiometer.
- J9: Goes to GND of the timer select potentiometer.
- J10: Not used. Goes to lower terminal of speaker to serve as GND. Arduino has extra

GND.
J11: Not used. Extra.
J12: Hook Arduino GND to the other GNDs.
J13: Goes to Arduino pin A2. Connected upstream from the push button.
J14: Push button 1 upper terminal.
J15: Push button 1 lower terminal.
J16: Goes to Arduino pin A3. Connected upstream from the push button.
J17: Push button 2 upper terminal.
J18: Push button 2 lower terminal.
J19-J22: Not used. Originally connected to the middle terminals of the potentiometers.
J23: Connected to the JD Power Supply 5V pin.
J24: Connected to the upper terminal of the emergency power switch.

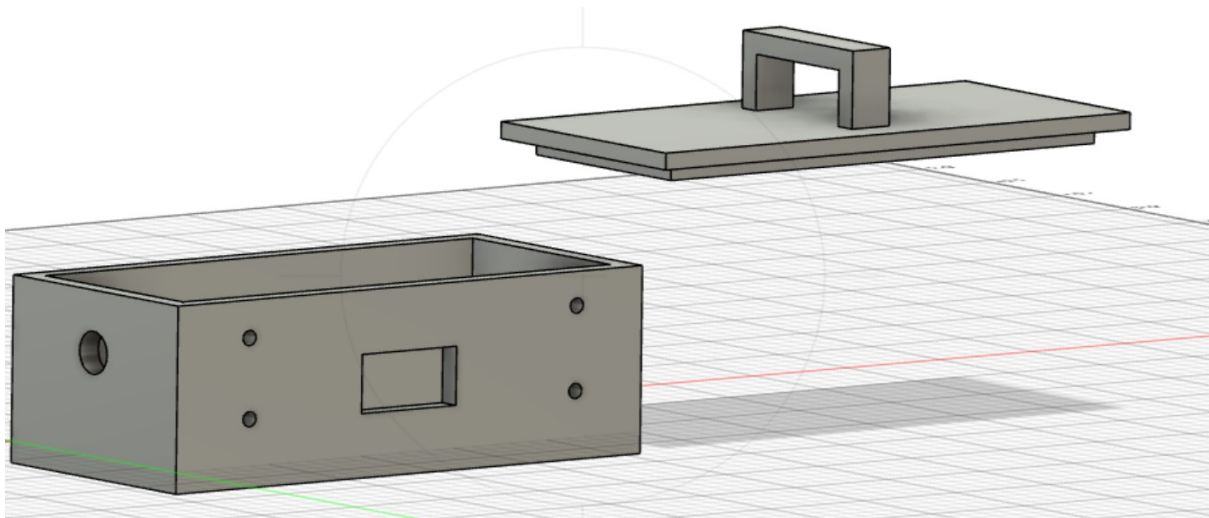


Figure 5: (The 3D enclosure) A box some 250mm x 120 mm x 87 mm was designed with holes to fit the seven segment display, the rotary potentiometers (variable resistors used to vary the voltage), push buttons, the speaker, the emergency switch and the power jack. The lid is designed to snap onto the top of the box. The inner part of the lid, fits snugly into the box, which allows it to snap on. The box will be made of lightweight plastic that is sturdy and durable. It will have a thickness of 10 mm all around. The components are placed in the holes and the remaining gaps are filled in with silicone (waterproofing and adhesive agent) to make it waterproof.

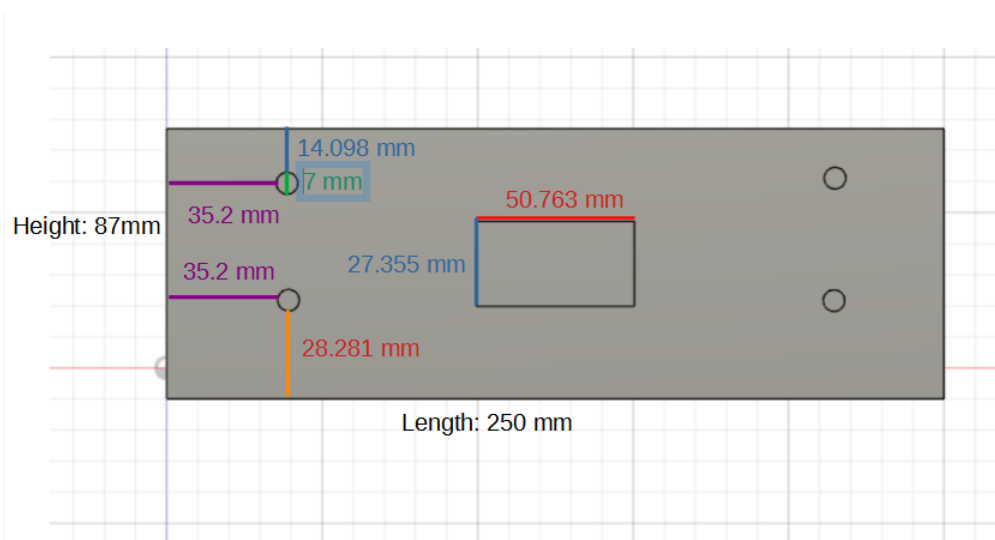


Figure 6: Dimensions for the front of the box. All circular holes in the diagram are 7mm in diameter. The box is mirrored across the middle, which means the dimensions on the left are the same as on the right.

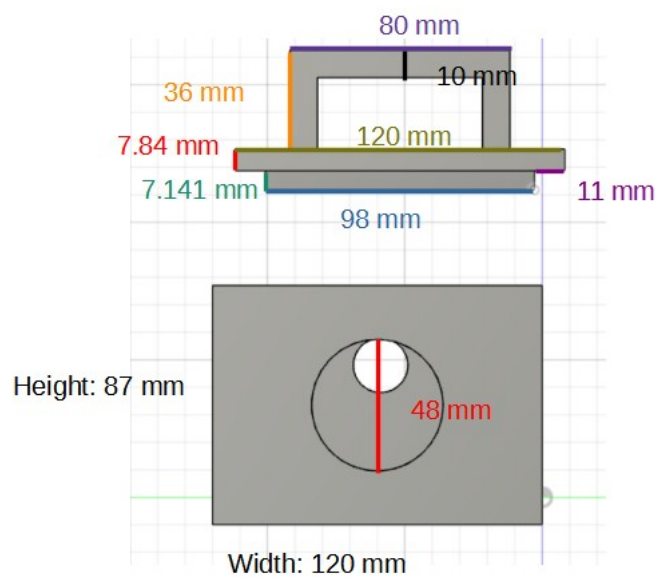


Figure 7: Dimensions for the right side of the box and the lid. The large hole is for the speaker. The lid has a handle, which makes it easy for the user to pull up and take it off.

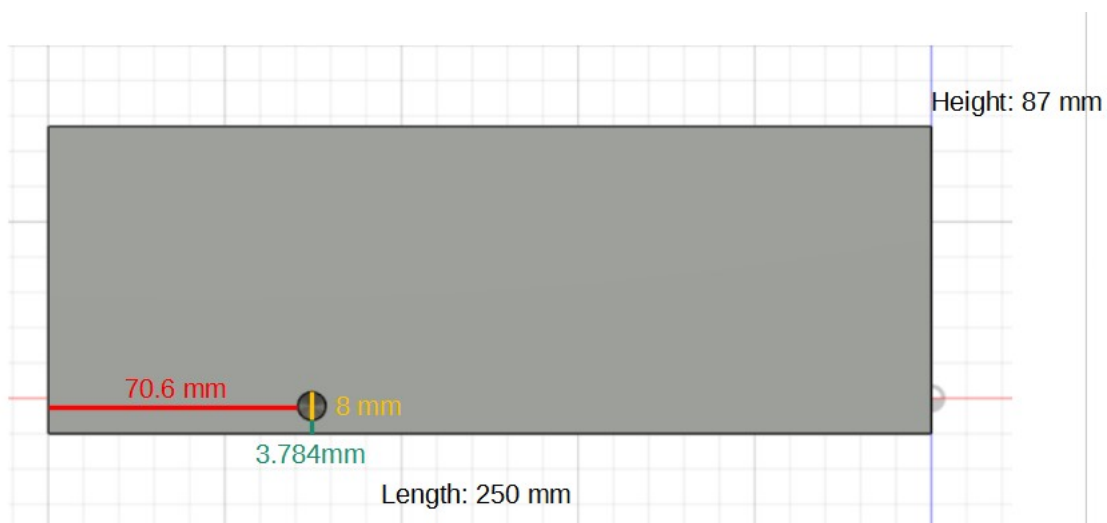


Figure 8: Dimensions for the back side of the box. The little hole near the bottom is used for the external power jack. The JD Power plug should be attached on the side of the hole with some adhesive to hold it in place.

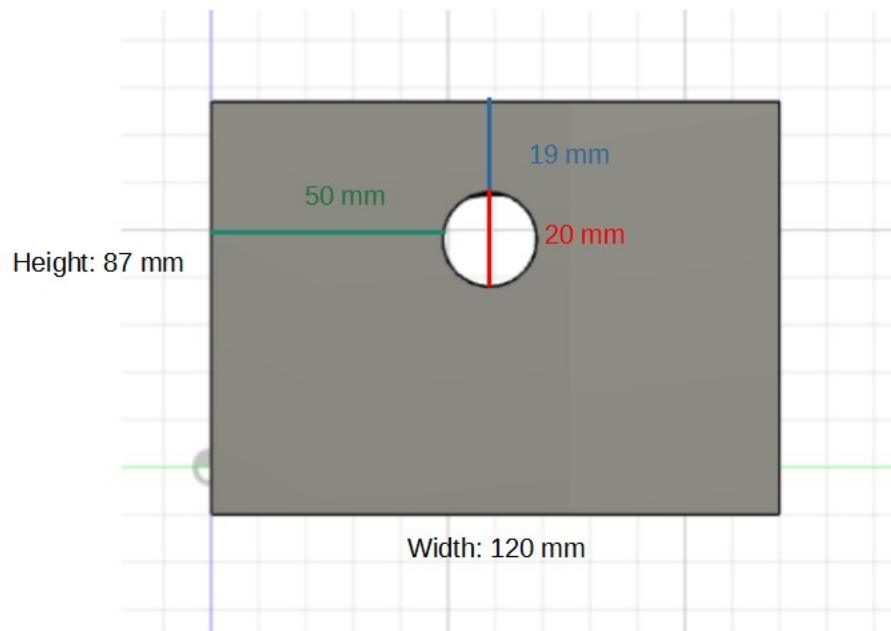


Figure 9: This is the left side of the box. The emergency power switch should fit inside this hole. Make sure there is enough silicone to hold the switch in place.

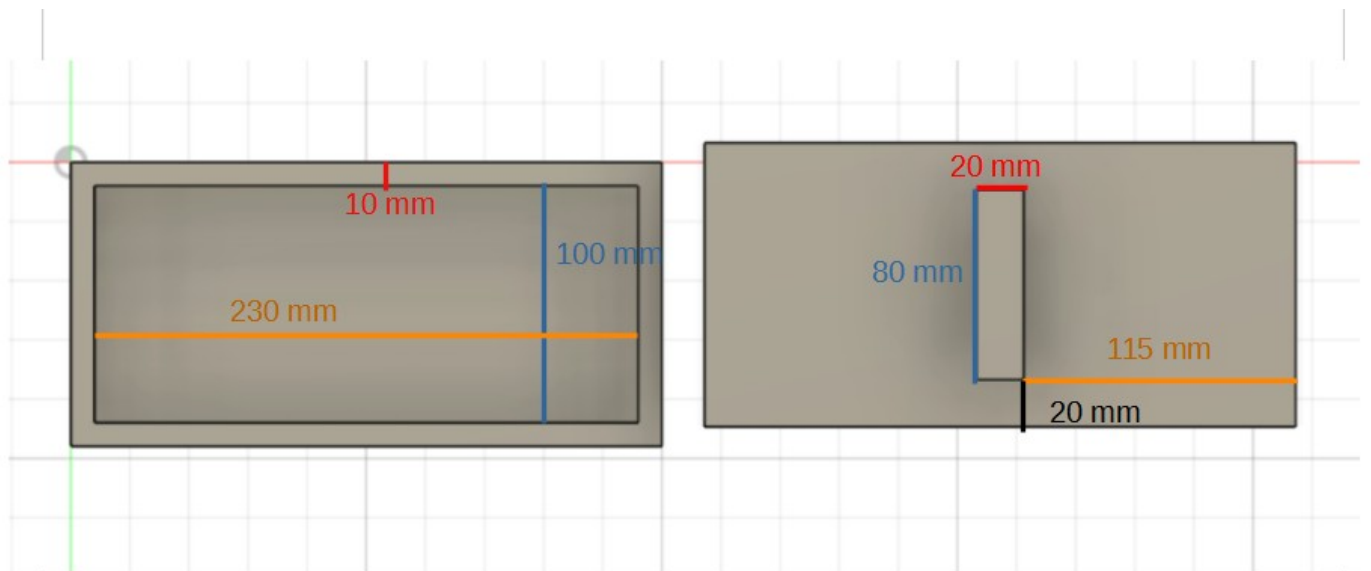


Figure 10: This a view of the box and the lid from above. The box is on the left and the lid is on the right. The thickness of the box is 10mm all the way around.

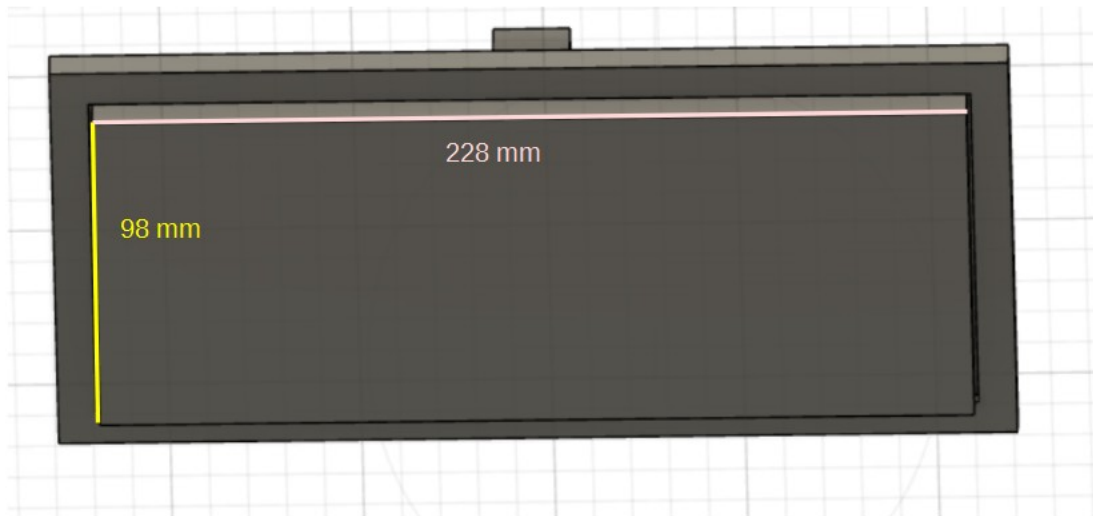


Figure 11: This is the bottom the lid. The extrusion on the bottom is a little smaller than the length and width of the inner perimeter of the box. This is because if they had the exact same dimensions, the lid wouldn't fit (or at least it would be a struggle to get it on). This small offset allows the lid to be 'snapped' on.

Code for the Arduino Nano:

```
#include <Wire.h> // Enable this line if using Arduino Uno, Mega, etc.
#include <Adafruit_GFX.h>
#include "Adafruit_LEDBackpack.h"
```

```
Adafruit_7segment matrix = Adafruit_7segment();
const int analogInPin = A0; //dimmer knob input
const int analogInPin1 = A1; //select time knob
const int analogInPin2 = A2; //start/finish button
const int analogInPin3 = A3; //add minute button
int sensorValue = 0; // value read from the potentiometer
int sensorValue1 = 0; // value read from another potentiometer
int sensorValue2 = 0; // value read from a button
int sensorValue3 = 0; // value read from another button
int minutes = 0;
int seconds = 0;
void setup() {
#ifdef __AVR_ATtiny85__
  Serial.begin(9600);
  Serial.println("7 Segment Backpack Test");
#endif
  matrix.begin(0x70);
}
```

```
void loop() {
  // print with print/println
  sensorValue2 = analogRead(analogInPin2);
  while(sensorValue2 > 800){
    //adjust brightness
    sensorValue = analogRead(analogInPin);
    delay(1);
    if(sensorValue < 341){ //if voltage below 1.67 V
      matrix.setBrightness(0); //set brightness to lowest level
    }
  }
}
```

```

    delay(1);
    if((sensorValue >= 341) && (sensorValue < 682)){ //if voltage between 1.67
V and 3.33 V
        matrix.setBrightness(3); //set brightness to medium level
    }
    delay(1);
    if(sensorValue >= 682){ //if voltage greater than 3.33 V
        matrix.setBrightness(15); //set brightness to highest level
    }
    sensorValue1 = analogRead(analogInPin1); //read for timer value
    delay(1);
    if(sensorValue1 >= 512){
        minutes = 1000;
        seconds = 0;
    }
    delay(1);
    if(sensorValue1 < 512){
        minutes = 400;
        seconds = 0;
    }
    matrix.println(minutes+seconds); //print ADC value
    matrix.writeDisplay(); //display ADC value on matrix
    sensorValue2 = analogRead(analogInPin2);
    delay(1);
    if(sensorValue2 < 900){
        break;
    }
}

```

```

sensorValue = analogRead(analogInPin); //take in analog ADC value
while(minutes+seconds >= 0){
    while(seconds > 0){
        for(int i = 403; i > 0; i--){
            delay(1);
            sensorValue = analogRead(analogInPin);
            if(sensorValue < 341){ //if voltage below 1.67 V
                matrix.setBrightness(0); //set brightness to lowest level
            }
            if((sensorValue >= 341) && (sensorValue < 682)){ //if voltage between 1.67
V and 3.33 V
                matrix.setBrightness(3); //set brightness to medium level
            }
            if(sensorValue >= 682){ //if voltage greater than 3.33 V
                matrix.setBrightness(15); //set brightness to highest level
            }
            delay(1);
            sensorValue3 = analogRead(analogInPin3);
            if(sensorValue3 < 100){
                minutes = minutes + 100;
                matrix.println(minutes+seconds); //print ADC value
                delay(300); //wait for the button to stop bouncing
                sensorValue3 = analogRead(analogInPin3);
            }
        }
        //total delay (2ms+ a little extra) * 400 loops = 1 second delay in total
        seconds = seconds - 1;
        matrix.println(minutes+seconds); //print ADC value
        matrix.writeDisplay(); //display ADC value on matrix
    }
    minutes = minutes - 100;
}

```

```

        seconds = seconds + 60;
    }
    sensorValue2 = analogRead(analogInPin2);
    delay(1);
    while(sensorValue2 > 800){
        tone(8, 440, 5);
        sensorValue2 = analogRead(analogInPin2);
        if(sensorValue2 < 100){
            break;
        }
        delay(1);
    }
    delay(200);
}

```

Component	Model no.	Specifications
Jameco 10k Potentiometer	24N-10K-15R-R	300 degree rotation – 125V – 0.05 W
Arduino Nano ATmega168P Board	P1564414455	8 bit AVR – 16 KB memory
0.56" Adafruit Seven Segment Backpack	Chip no: HT16K33	5V 1.05"x1.97" Yellow LED Display
JD Power Supply	N/A	12 V in – 5/3.3V out
Kobitone Speaker	25SP105	Freq Response of 350 Hz to 4KHz. Rated Power: 0.3 W Max Power: 0.6 W Impedance: 8 Ohm +/- 15%
Panasonic 10k SMD resistor from Digi-key	P10KACT-ND	Resistance: 10k, 1.25x2.0x0.6mm Rated power: 0.125 W Temp: -55 to 155 C
0.1" long male connector pins	2012-254-1140-RG	0.6 mm in diameter
Twidex SPST mini pushbuttons	JND-PBS-110-X6C	Rated for 1A 250 V AC, 6.9 mm in diameter
TWTADE 2 pin Rocker Switch	KCD1-X-Y	6A/250V, 21x15mm

Table 2: Part Information



Figure 12: Final product display. Timer in the mock enclosure.