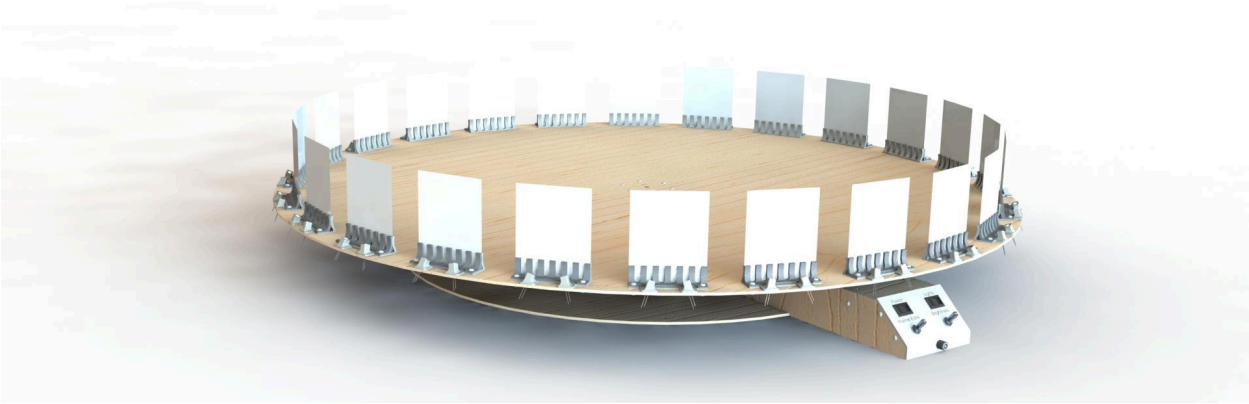
David Van Dyke

DIY Spinning Stroboscope



Overview

This report details how to create a spinning stroboscope and was created as a christmas present for girl scout leader Louisa Ho. A stroboscope is a mechanical device that uses flash strobe lights over multiple images to make it appear as if it were moving. This device has 24 images and spins at a rate of 1 rotation per second while activating a strobe light every 1/24th of a second. To the viewer, each of these images appears for 1/24th of a second and it looks like a video at 24 frames per second (FPS). This device can be used to make GIFs in real life and act as a demonstration of sampling. See Figure 1 below to see the entire device.

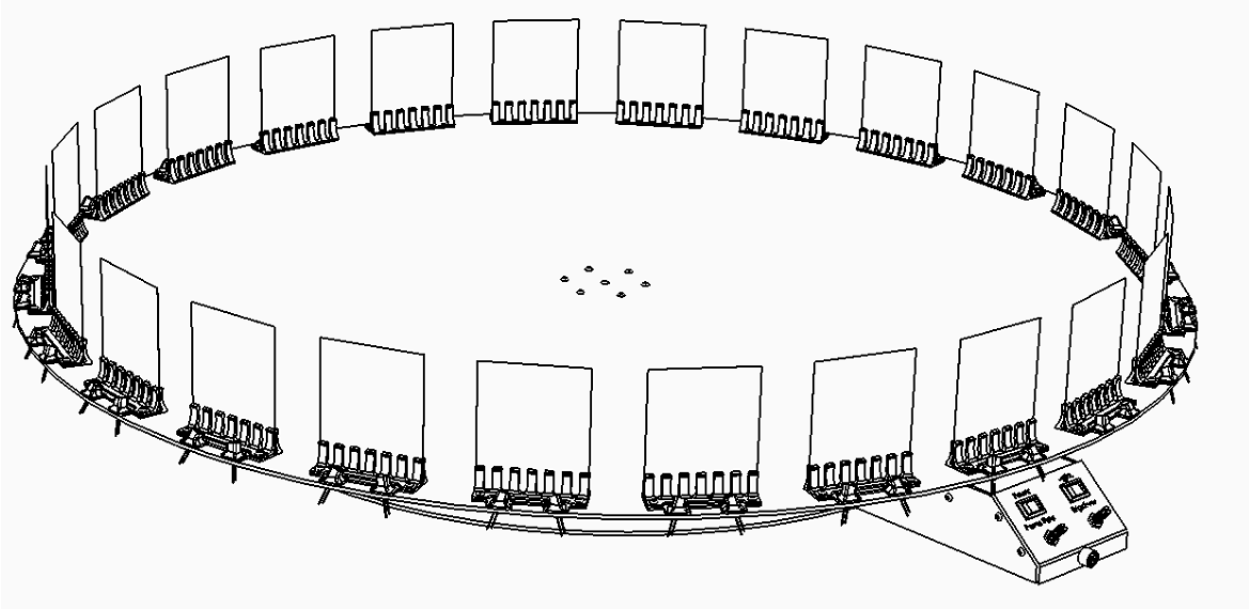


Figure 1. The full system assembly for the stroboscope.

The documentation behind this project (this document): can be found here
The source files and code needed to make it can be found here
A walkthrough of how to make one can be found at www.dhvd.net/stroboscope.html

Materials and Tools

This device was designed to be made using simple tools and parts from Amazon. This was done to keep the budget low and make it accessible people with an ordinary skill in the art. A full list of the parts purchased can be seen in Table 1 below.

The following tools are required to make this device:

Measuring tools: Yardstick, pencil, protractor
Woodworking: Sandpaper, disc sander (recommended), scrollsaw/bandsaw, drill
Electrical: Soldering iron, solder

Table 1. Parts to purchase for the stroboscope. Wire, glue, 3D printing filament, and paper are not included in this list.

Name	Quantity Required	Source (12/20/18)	Cost
⅛" (0.115" Actual) Thickness MDF	4 ft x 8 ft	https://www.homedepot.com/p/Hardboard-Tempered-Panel-Common-1-8-in-4-ft-x-8-ft-Actual-0-115-in-x-47-7-in-x-95-7-in-832777/202189720	8.62
Nema 17 Stepper Motor	1	https://www.amazon.com/gp/product/B074TCM8ZL/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1	21.99
Stepper Motor Driver	1	Included with stepper motor	
M3 x 10mm Socket Button Head Screw	78	https://www.amazon.com/gp/product/B019ZC3MD0/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1	8.67
M3 x 0.5mm Lock Nut	78	https://www.amazon.com/gp/product/B075ZZW7VL/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1	7.99
NPN Power Transistor	1	https://www.amazon.com/gp/product/B01N9N1RZG/ref=oh_aui_detailpage_o01_s00?ie=UTF8&psc=1	7.35
R6-2RS Sealed Ball Bearing	2	https://www.amazon.com/gp/product/B07GT82D7P/ref=oh_aui_detailpage_o01_s01?ie=UTF8&psc=1	9.45
Arduino Uno	1	https://www.amazon.com/gp/product/B01EWOE0UU/ref=oh_aui_detailpage_o01_s02?ie=UTF8&psc=1	10.86
12V Power Supply	1	https://www.amazon.com/gp/product/B01461MOGQ/ref=oh_aui_detailpage_o01_s02?ie=UTF8&psc=1	10.59

12V 5mm LED Light	48	https://www.amazon.com/gp/product/B01MYYUS2R/ref=od_aui_detailpages02?ie=UTF8&psc=1	8.56
SPST Rocker Switch	2	https://www.amazon.com/FBApayipa-Solder-Rocker-Switch-Toggle/dp/B01N2U8PK0/ref=sr_1_3?s=industrial&ie=UTF8&qid=1540956013&sr=1-3&keywords=rocker+switch&dpID=51fr7fljMyL&preST=_SX342_QL70_&dpSrc=srch	6.50
10K Potentiometer	2	https://www.amazon.com/Uxcell-a15040700ux0380-Terminals-Linear-Potentiometer/dp/B019I13X5K/ref=sr_1_4_acs_ac_1?s=industrial&ie=UTF8&qid=1539287508&sr=1-4-acs&keywords=potentiometer#feature-bullets-btf	7.20
		Total	107.78

Design

Electrical

The device is spun at a rate of 1 rotation per second using a stepper motor. The motor driven by a stepper motor driver and controlled by an arduino uno. It receives power from a 12VDC power supply. The strobe lights are 48 12V LED lights wired in parallel, powered by the same 12VDC power supply. A NPN transistor controlled by the arduino turns the circuit on and off to create the strobe effect.

The user interface includes a rocker switch to turn the whole system on and off, a rocker switch to turn the lights on, a rotary potentiometer to modify the frame rate, and a rotary potentiometer to alter the brightness (see Figure 2). The first rocker switch must be set to “1” for the system to operate as this controls the 12VDC power. The second rocker switch is wired in series with every LED so they will not turn on unless this switch is flipped. The first potentiometer is wired into an analog pin on the arduino and will tell the arduino how frequently to active the strobe light (ex. every 1/6th second instead of every 1/24th). The last potentiometer is wired in series with every LED which will reduce the current as the resistance increases, causing the LED lights to decrease in brightness. A full wiring schematic showing how each component must be connected can be seen in Figure 3 and the code can be found in Appendix A1.

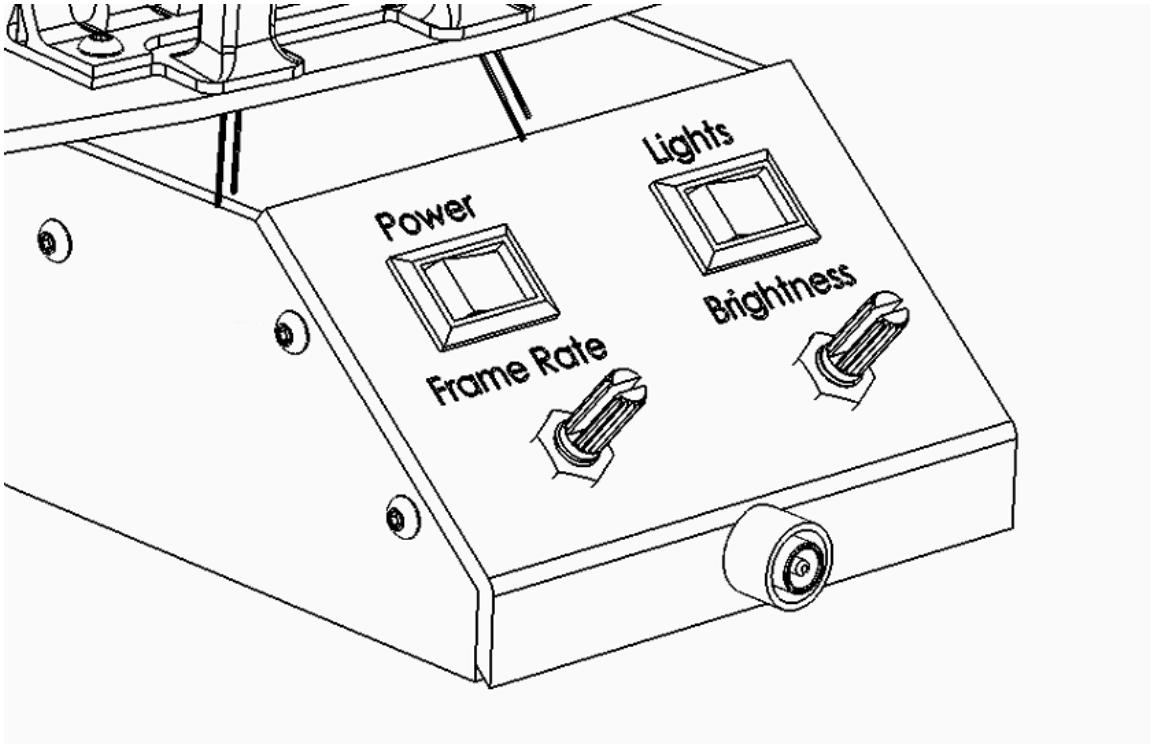


Figure 2. User interface

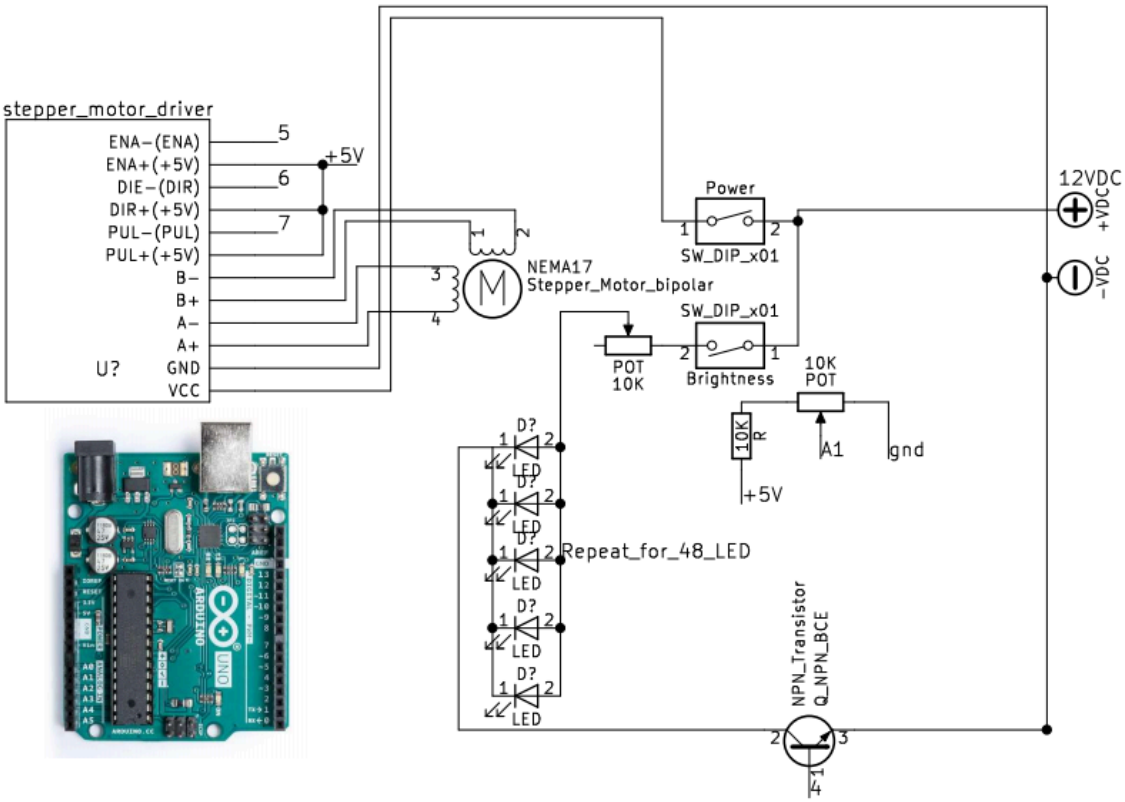


Figure 3. Full electrical wiring schematic

Mechanical

The main body of the stroboscope is constructed of ⅛” thick (0.115” actual) MDF board which is cheap and easy to machine. The engineering drawings used to create each of these parts can be found in Appendix A2. These MDF parts are either screwed together using 3D printed 90 degree tabs or glued together. 24 images must be positioned along the main disc to create the stroboscopic effect. These images are printed on paper and is clamped in by two 3D printed pieces which are also screwed into the base. These 3D printed parts also house the LED lights and point them directly to the center of the image (see Figure 4).

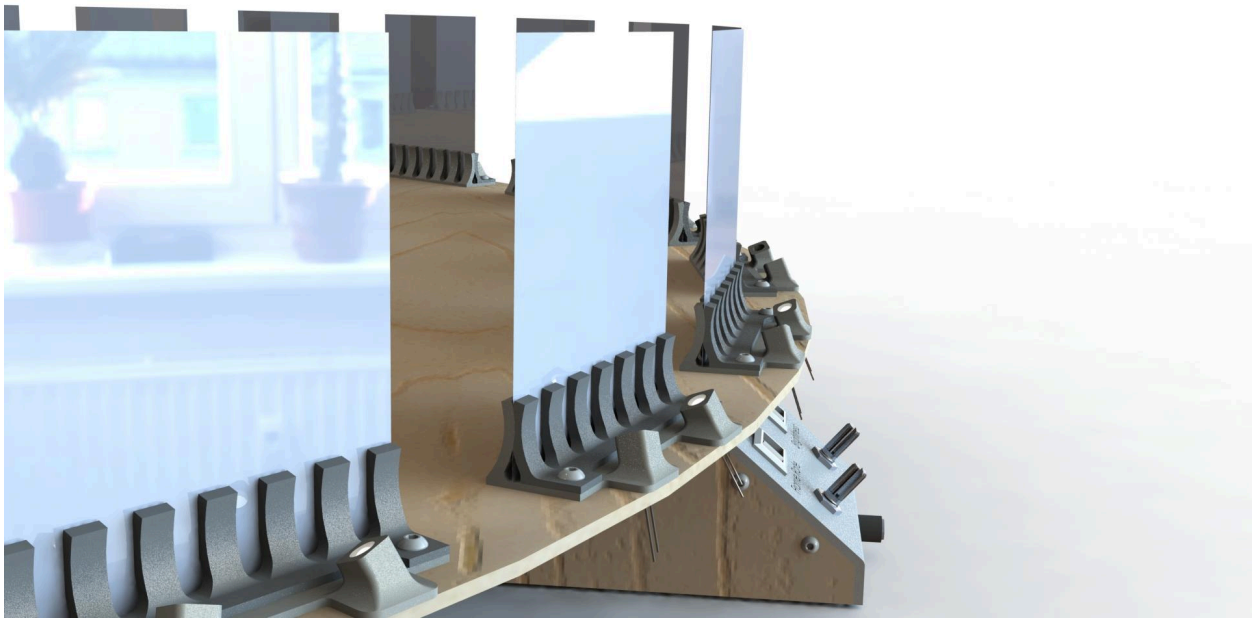


Figure 4. 3D printed clamps used to securely mount each image to the spinning base

A separate 3D printed part is used to connect the spinning disc to the stepper motor shaft. Ball bearings are used to support the disc and wires are clamped to these bearings to transfer power from the 12VDC power supply in the base to the spinning disc. Two wires are clamped to the outside ring of the bearing, which remains stationary, and a seperate two are clamped to the inside ring which rotates with the stepper motor (see Figure 5).

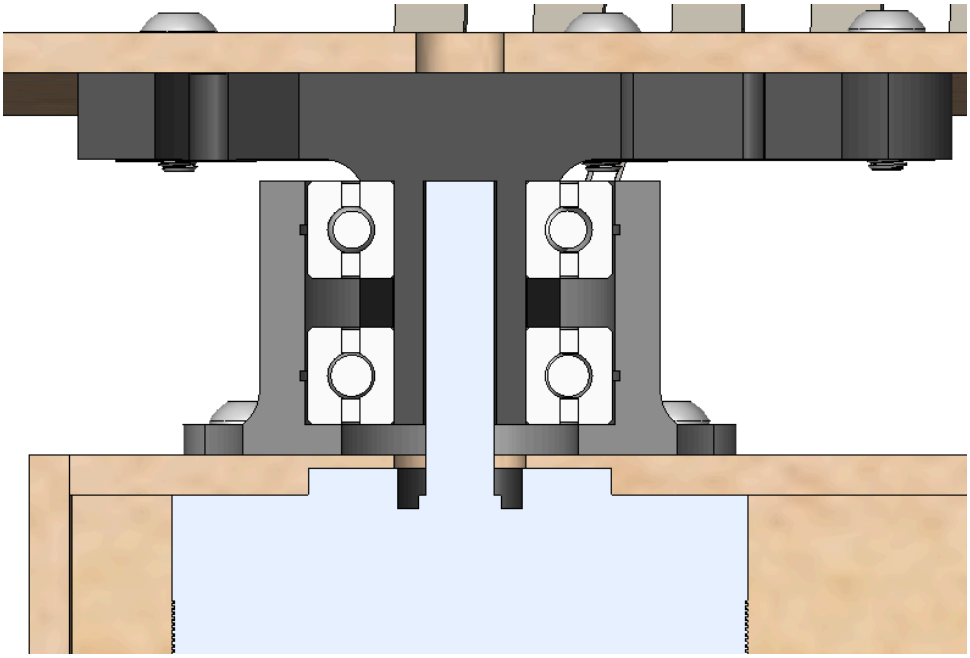


Figure 5. Cross section for the torque transfering system in the stroboscope. The housings for the bearings are 3D printed and have wire routing locations so wires can be routed to the inside and outside faces of the bearing.

Code

The stroboscope uses an Arduino Uno to control the speed of the stepper motor and frequency of the LED strobes. This code can be seen in Appendix A1.

Construction

Figure 6 below shows how all the parts in the stroboscope are connected. See www.dhvd.net/stroboscope.html for a full walkthrough of how to build one.

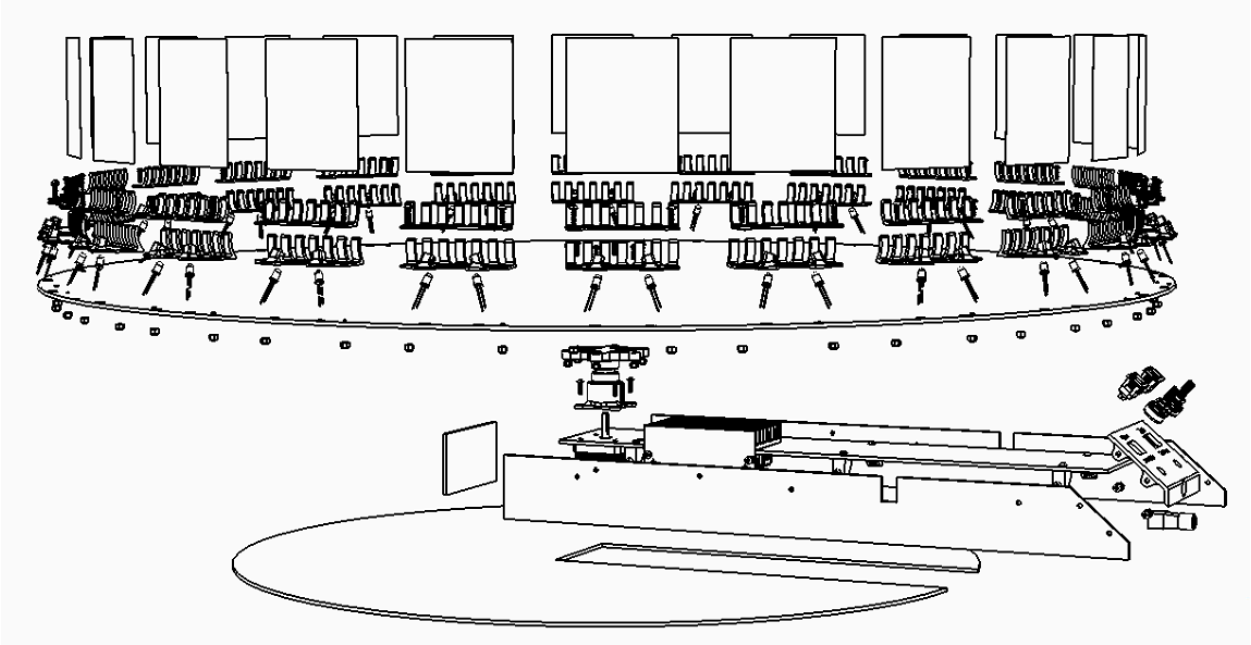


Figure 6. Exploded view of the whole assembly.

Image Preparation

24 images must be attached across circumference of the disc. These images should be from a GIF or video as they will be animated on the stroboscope. Ideally, the user will extract 24 frames from a 1 second GIF/Video. 24 FPS is standard in the film industry and should be easy to use. Once the frames are extracted, they should be printed on thick paper on a 3”x3” square with a 0.625” blank space below the image to create space for clamping (see Figure 7).

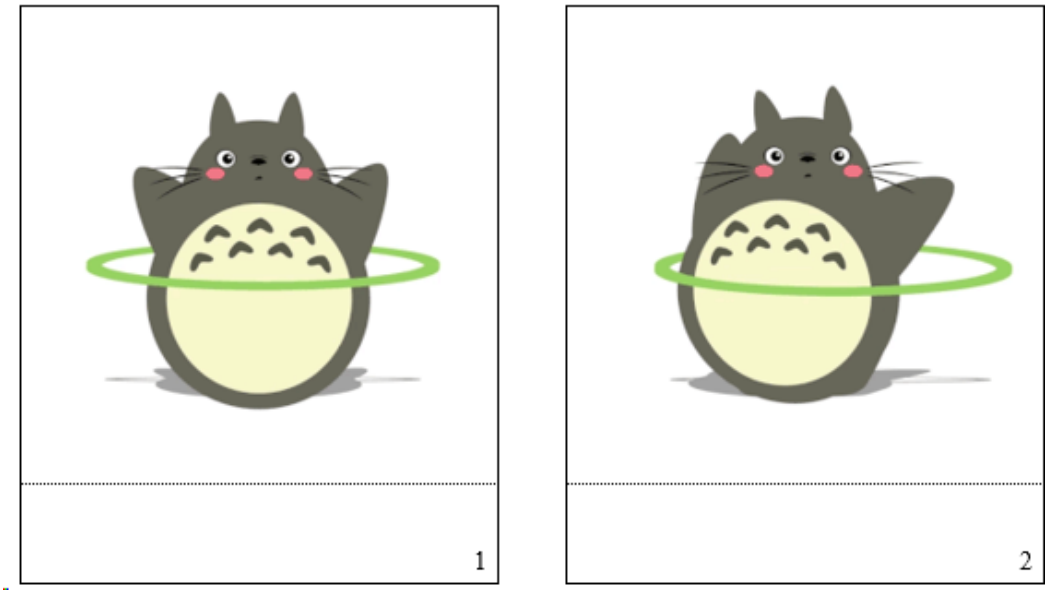


Figure 7. Example of how the images should be printed

Some helpful sources for creating the images:
FInding source GIFs : https://giphy.com/
Editing and extracting frames: https://ezgif.com/split
Image printing and labeling: images → image_template.docx in stroboscope folder

When using the suggested word document for printing the images and labels, the 24 images should be scaled and cropped to a 273x273 px square. This ensures they fit inside the 3” square.

Final Prototype

A stroboscope prototype was created using this design (see Figure 8). It successfully animates a 24 frame gif and it can be seen both in the dark and in the light. The frame rate can be set to 6, 12, or 24 frames per second as an example of how sampling changes a video. This is adjusted by turning the left potentiometer. The brightness of the LED strobes can also be adjusted using the right potentiometer.

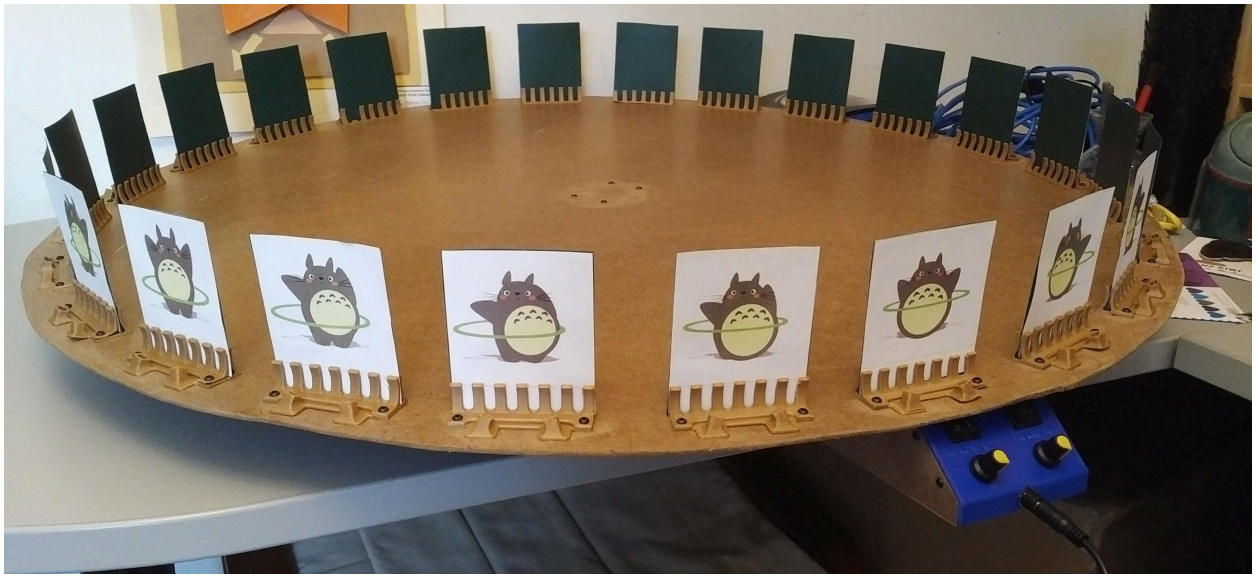


Figure 8. Final assembled prototype. A video of this device in action can be seen at https://www.youtube.com/watch?v=FecR3SHmatk&feature=youtu.be

While this prototype works, it is fairly fragile as the large spinning disk is poorly supported. This makes it very easy to bend the center torque providing shaft while moving the device. I plan to add a large “lazy susan” bearing to the disk to rigidly connect it to the base while allowing it to rotate.

Appendix

A1 - Arduino Code

```
//Code to operate the stroboscope
//David Van Dyke- 12/29/18

const int framerate_pin = A1;
int framerate = 24;

const int led_pin = 4;

int rps_speed = 1;

//Initialize stepper motor
const int steps_per_revolution = 1600;
const int ENA = 5; //Enable Pin
const int DIR = 6; //Direction Pin
const int PUL = 7; //Pulse Pin

//Time delay between each step in microseconds
const double timeDelay = ((double)rps_speed)/(double)steps_per_revolution*1000000.0;
const double half_timeDelay = timeDelay/2;


//Current time
//Used to calculate delay
//Warning: May cause errors after 70 min of operation due to overflow
int current_time = 0;
int original_time = 0;

//Locations of frames to light LEDs
int led_locations[24];

//Controls pulse width of the LED
int led_timer = 0;
int led_pulse = 30; //microseconds

void setup() {

  //Start serial, for debugging
  Serial.begin(115200);

  //Initialize motor
  pinMode(ENA, OUTPUT);
```

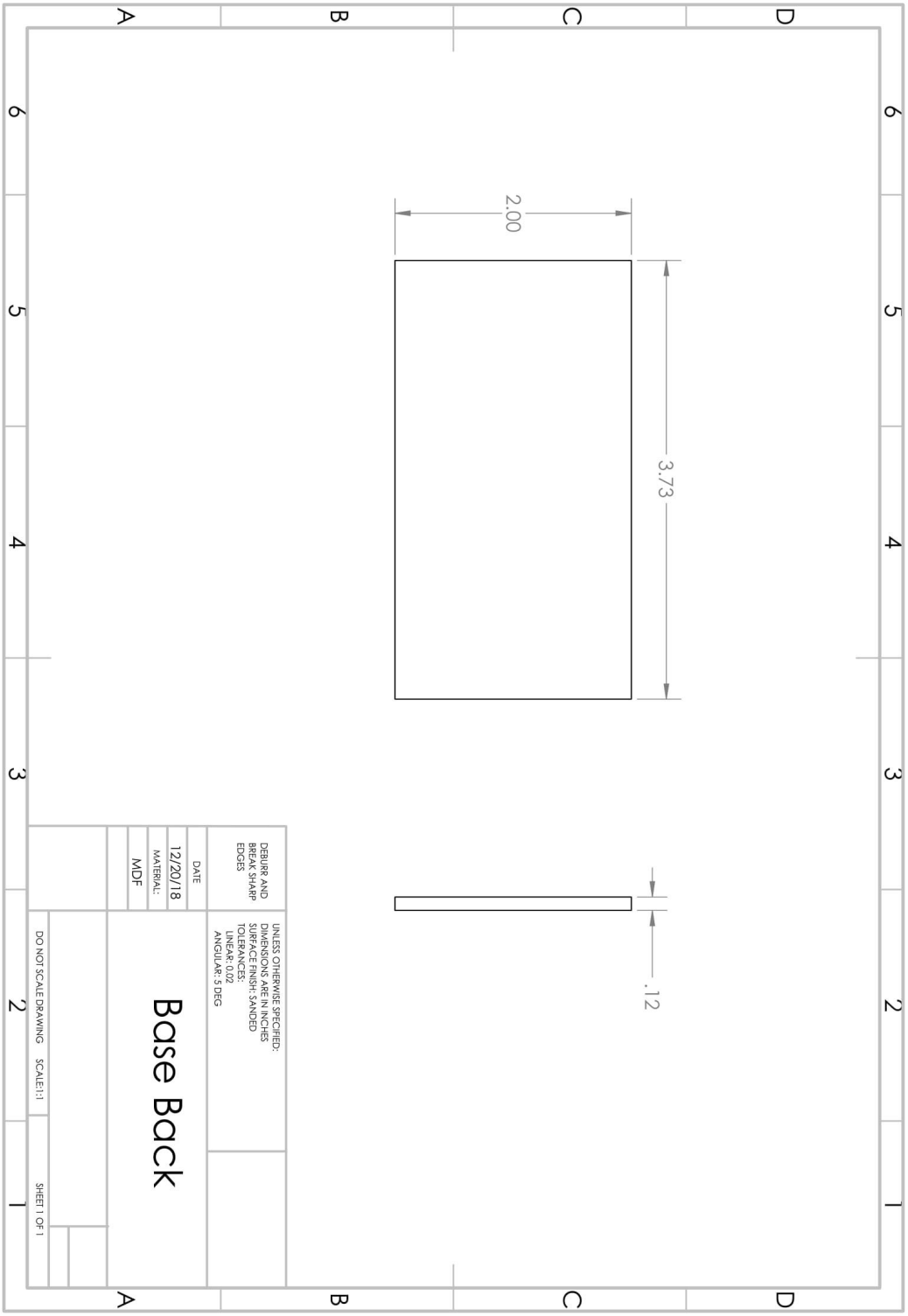
```
  pinMode(DIR, OUTPUT);
  pinMode(PUL, OUTPUT);

  //Set direction, LOW for Clockwise, HIGH for counterclockwise
  digitalWrite(DIR, LOW);
  //Set ENA
  digitalWrite(ENA, HIGH);
  update_framerate();


  //Initialize LED
  pinMode(led_pin, OUTPUT);Serial.print("Time Delay: ");
  Serial.println(timeDelay);
}

void loop() {
  // put your main code here, to run repeatedly:
  //loop every one revolution
  digitalWrite(led_pin, LOW);
  //take every step
  for(int i = 0; i < steps_per_revolution; ++i){
    //Get current and original
    current_time = micros();
    original_time = current_time;

    //update the framerate reading on rhe first step
    if(i == 0)update_framerate();

    //Start the movement of the motor
    digitalWrite(PUL, HIGH);
    //Check if the LED should be lit up and light it up if it should be
    if(is_present(i)){
      digitalWrite(led_pin, HIGH);
      led_timer = micros();
    }
    //Loop until the time has reached the desired delay
    //Used instead of the delay function due to inherint delay other functions
    current_time = micros();
    while(abs(current_time-original_time) < half_timeDelay){
      current_time = micros();

    }
    /*
    if(current_time > led_timer + led_pulse){
       digitalWrite(led_pin, LOW);
    }*/
    //Stop the motor and turn off the LED
    digitalWrite(PUL, LOW);
```

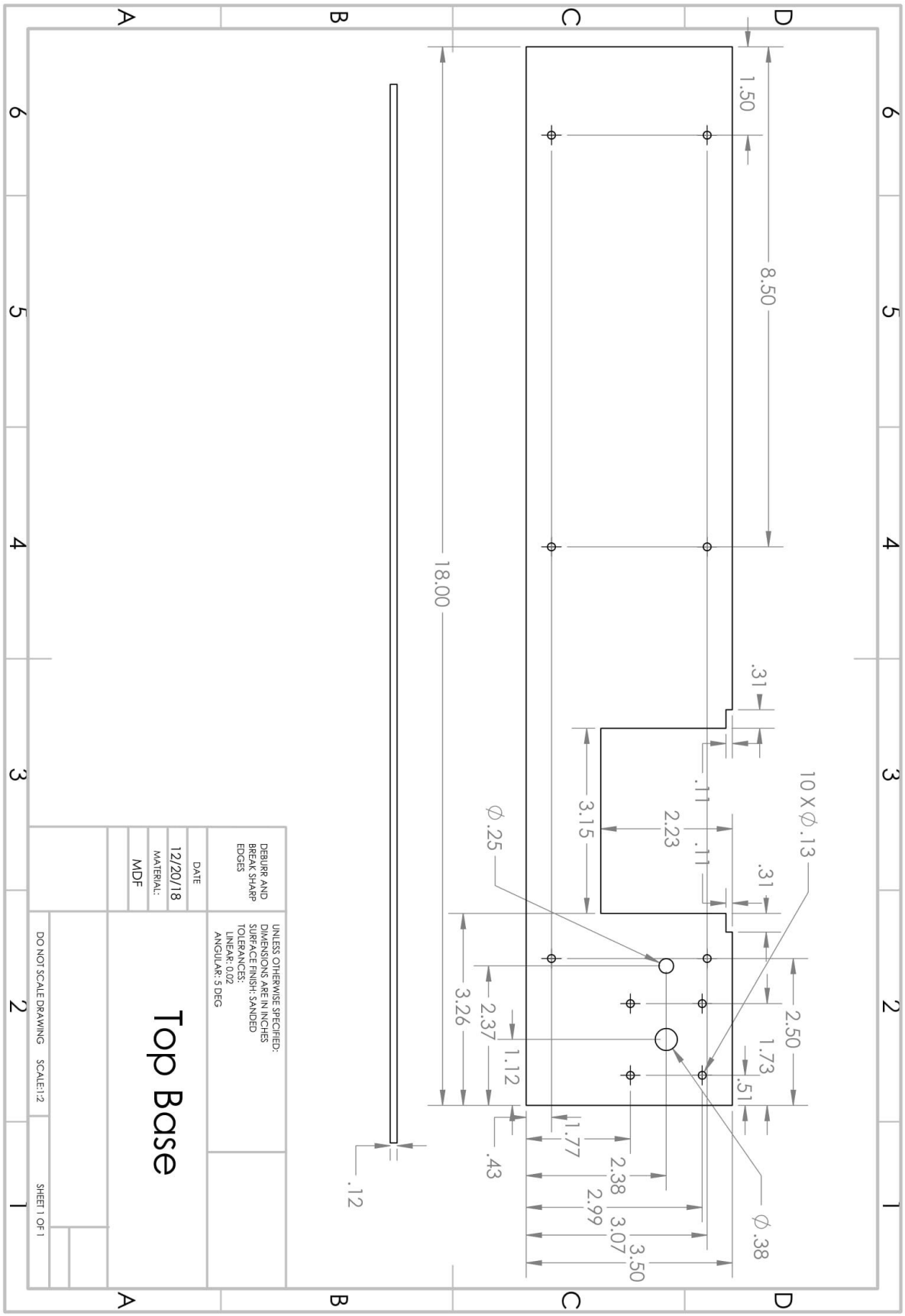
```
    //Loop until the time has reached the desired delay
    //Used instead of the delay function due to inherint delay other functions
    current_time = micros();
    while(abs(current_time-original_time) < timeDelay){
      current_time = micros();
    }
    digitalWrite(led_pin, LOW);
  }
}

//Updates frame and LED keyframes based on input potentiometer
void update_framerate(){
  int input = analogRead(framerate_pin);

  //The input should be between 0-512 assuming a 10K pot and 10K resistor
  if(input < 128){
    framerate = 6;
    led_locations[0] = 0;
    led_locations[1] = (int)(1.0*(double)steps_per_revolution/(double)framerate);
    led_locations[2] = (int)(2.0*(double)steps_per_revolution/(double)framerate);
    led_locations[3] = (int)(3.0*(double)steps_per_revolution/(double)framerate);
    led_locations[4] = (int)(4.0*(double)steps_per_revolution/(double)framerate);
    led_locations[5] = (int)(5.0*(double)steps_per_revolution/(double)framerate);
    led_locations[6] = 0;
    led_locations[7] = 0;
    led_locations[8] = 0;
    led_locations[9] = 0;
    led_locations[10] = 0;
    led_locations[11] = 0;
    led_locations[12] = 0;
    led_locations[13] = 0;
    led_locations[14] = 0;
    led_locations[15] = 0;
    led_locations[16] = 0;
    led_locations[17] = 0;
    led_locations[18] = 0;
    led_locations[19] = 0;
    led_locations[20] = 0;
    led_locations[21] = 0;
    led_locations[22] = 0;
    led_locations[23] = 0;
  }
  else if(input < 256){
    framerate = 12;
    led_locations[0] = 0;
    led_locations[1] = (int)(1.0*(double)steps_per_revolution/(double)framerate);
    led_locations[2] = (int)(2.0*(double)steps_per_revolution/(double)framerate);
```

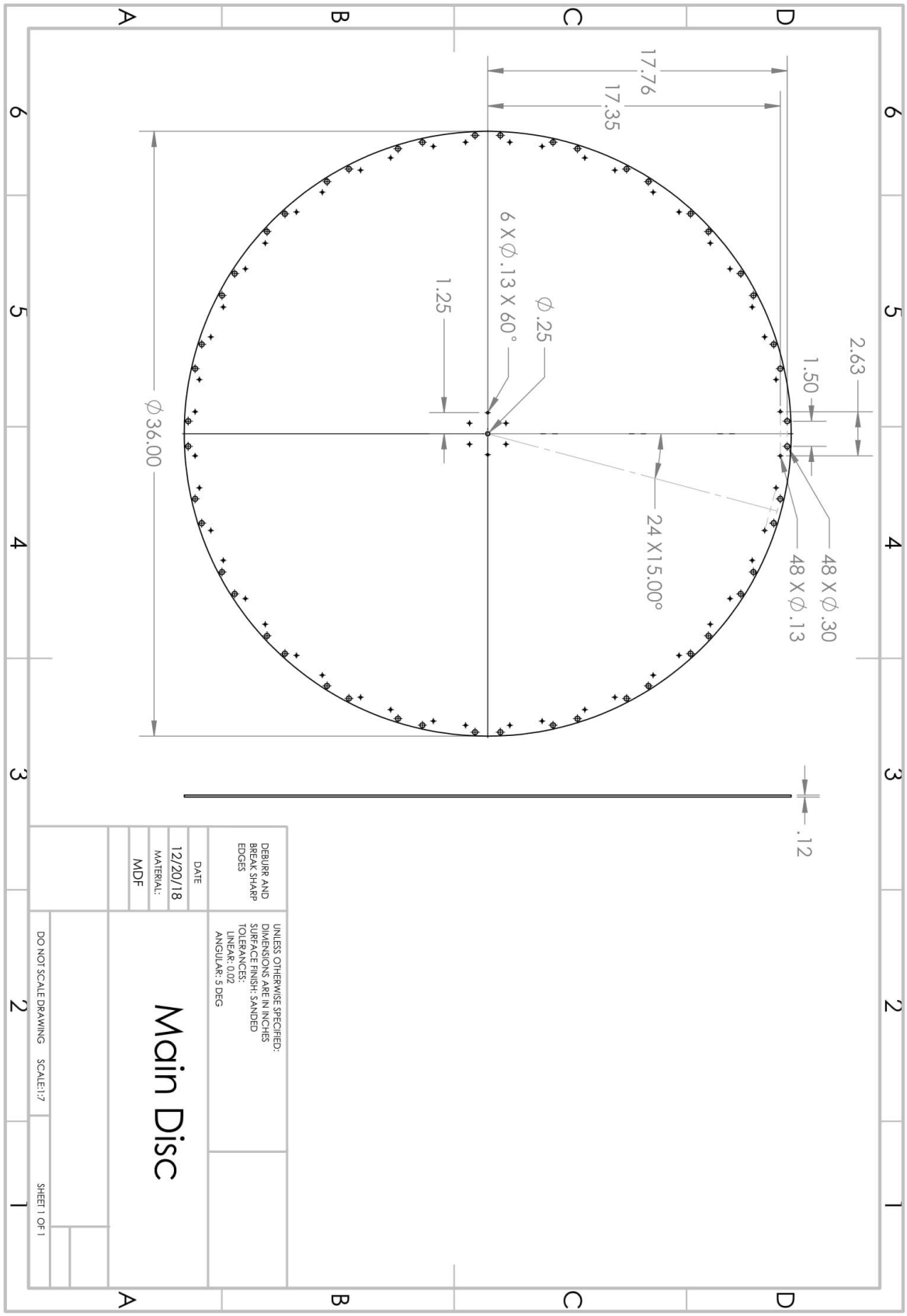
```
  led_locations[3] = (int)(3.0*(double)steps_per_revolution/(double)framerate);
  led_locations[4] = (int)(4.0*(double)steps_per_revolution/(double)framerate);
  led_locations[5] = (int)(5.0*(double)steps_per_revolution/(double)framerate);
  led_locations[6] = (int)(6.0*(double)steps_per_revolution/(double)framerate);
  led_locations[7] = (int)(7.0*(double)steps_per_revolution/(double)framerate);
  led_locations[8] = (int)(8.0*(double)steps_per_revolution/(double)framerate);
  led_locations[9] = (int)(9.0*(double)steps_per_revolution/(double)framerate);
  led_locations[10] = (int)(10.0*(double)steps_per_revolution/(double)framerate);
  led_locations[11] = (int)(11.0*(double)steps_per_revolution/(double)framerate);
  led_locations[12] = 0;
  led_locations[13] = 0;
  led_locations[14] = 0;
  led_locations[15] = 0;
  led_locations[16] = 0;
  led_locations[17] = 0;
  led_locations[18] = 0;
  led_locations[19] = 0;
  led_locations[20] = 0;
  led_locations[21] = 0;
  led_locations[22] = 0;
  led_locations[23] = 0;
}
else if(input < 384){
  framerate = 18;
  led_locations[0] = 0;
  led_locations[1] = (int)(1.0*(double)steps_per_revolution/(double)framerate);
  led_locations[2] = (int)(2.0*(double)steps_per_revolution/(double)framerate);
  led_locations[3] = (int)(3.0*(double)steps_per_revolution/(double)framerate);
  led_locations[4] = (int)(4.0*(double)steps_per_revolution/(double)framerate);
  led_locations[5] = (int)(5.0*(double)steps_per_revolution/(double)framerate);
  led_locations[6] = (int)(6.0*(double)steps_per_revolution/(double)framerate);
  led_locations[7] = (int)(7.0*(double)steps_per_revolution/(double)framerate);
  led_locations[8] = (int)(8.0*(double)steps_per_revolution/(double)framerate);
  led_locations[9] = (int)(9.0*(double)steps_per_revolution/(double)framerate);
  led_locations[10] = (int)(10.0*(double)steps_per_revolution/(double)framerate);
  led_locations[11] = (int)(11.0*(double)steps_per_revolution/(double)framerate);
  led_locations[12] = (int)(12.0*(double)steps_per_revolution/(double)framerate);
  led_locations[13] = (int)(13.0*(double)steps_per_revolution/(double)framerate);
  led_locations[14] = (int)(14.0*(double)steps_per_revolution/(double)framerate);
  led_locations[15] = (int)(15.0*(double)steps_per_revolution/(double)framerate);
  led_locations[16] = (int)(16.0*(double)steps_per_revolution/(double)framerate);
  led_locations[17] = (int)(17.0*(double)steps_per_revolution/(double)framerate);
  led_locations[18] = 0;
  led_locations[19] = 0;
  led_locations[20] = 0;
  led_locations[21] = 0;
  led_locations[22] = 0;
  led_locations[23] = 0;
```

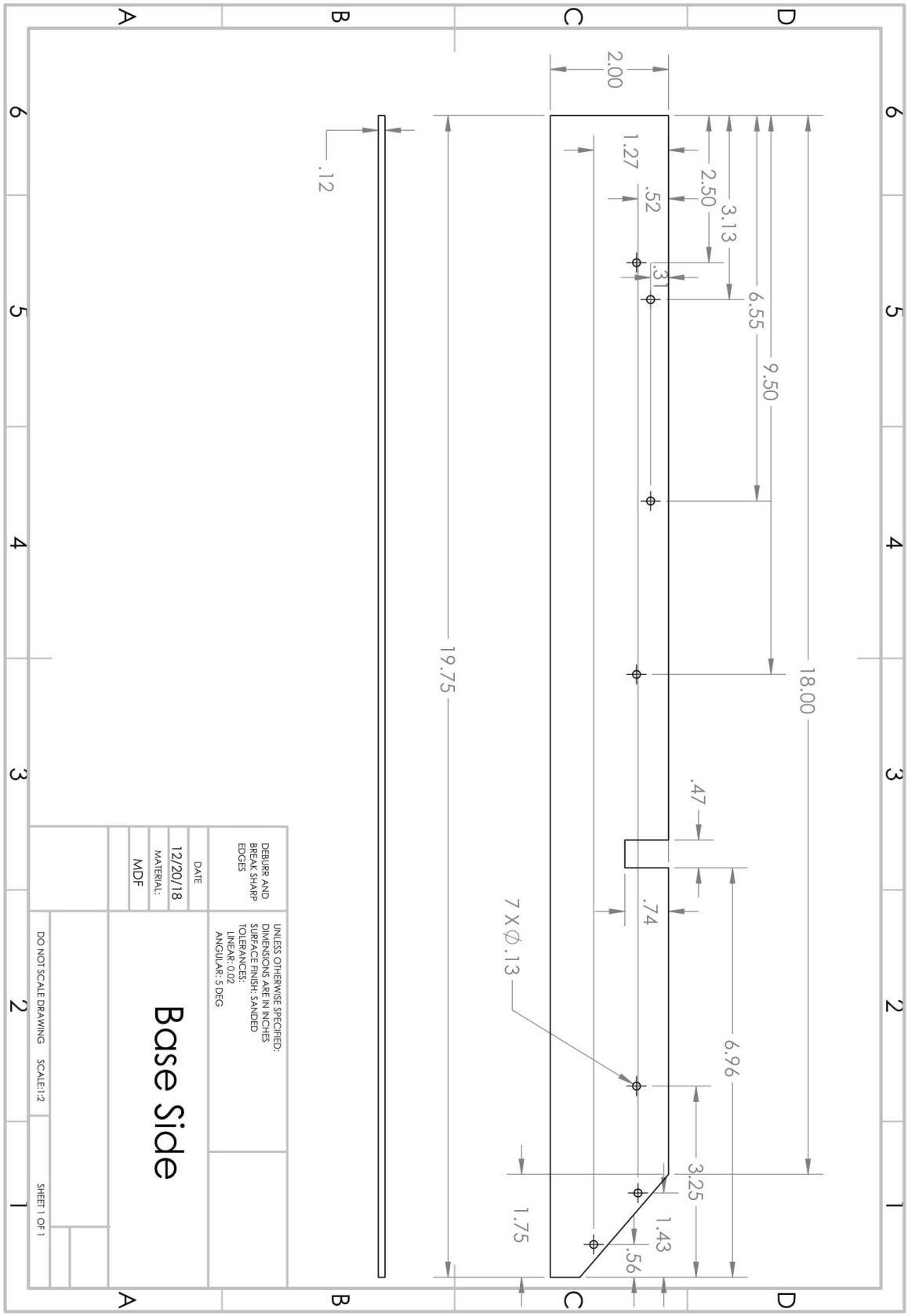
```
  }
  else{
    framerate = 24;
    led_locations[0] = 0;
    led_locations[1] = (int)(1.0*(double)steps_per_revolution/(double)framerate);
    led_locations[2] = (int)(2.0*(double)steps_per_revolution/(double)framerate);
    led_locations[3] = (int)(3.0*(double)steps_per_revolution/(double)framerate);
    led_locations[4] = (int)(4.0*(double)steps_per_revolution/(double)framerate);
    led_locations[5] = (int)(5.0*(double)steps_per_revolution/(double)framerate);
    led_locations[6] = (int)(6.0*(double)steps_per_revolution/(double)framerate);
    led_locations[7] = (int)(7.0*(double)steps_per_revolution/(double)framerate);
    led_locations[8] = (int)(8.0*(double)steps_per_revolution/(double)framerate);
    led_locations[9] = (int)(9.0*(double)steps_per_revolution/(double)framerate);
    led_locations[10] = (int)(10.0*(double)steps_per_revolution/(double)framerate);
    led_locations[11] = (int)(11.0*(double)steps_per_revolution/(double)framerate);
    led_locations[12] = (int)(12.0*(double)steps_per_revolution/(double)framerate);
    led_locations[13] = (int)(13.0*(double)steps_per_revolution/(double)framerate);
    led_locations[14] = (int)(14.0*(double)steps_per_revolution/(double)framerate);
    led_locations[15] = (int)(15.0*(double)steps_per_revolution/(double)framerate);
    led_locations[16] = (int)(16.0*(double)steps_per_revolution/(double)framerate);
    led_locations[17] = (int)(17.0*(double)steps_per_revolution/(double)framerate);
    led_locations[18] = (int)(18.0*(double)steps_per_revolution/(double)framerate);
    led_locations[19] = (int)(19.0*(double)steps_per_revolution/(double)framerate);
    led_locations[20] = (int)(20.0*(double)steps_per_revolution/(double)framerate);
    led_locations[21] = (int)(21.0*(double)steps_per_revolution/(double)framerate);
    led_locations[22] = (int)(22.0*(double)steps_per_revolution/(double)framerate);
    led_locations[23] = (int)(23.0*(double)steps_per_revolution/(double)framerate);
   }
}

//checks if a value is present in the led_locations array
bool is_present(int led_loc){
  for(int i=0; i<24; ++i){
    if(led_loc == led_locations[i]){
      return true;
    }
  }
  return false;
}
```

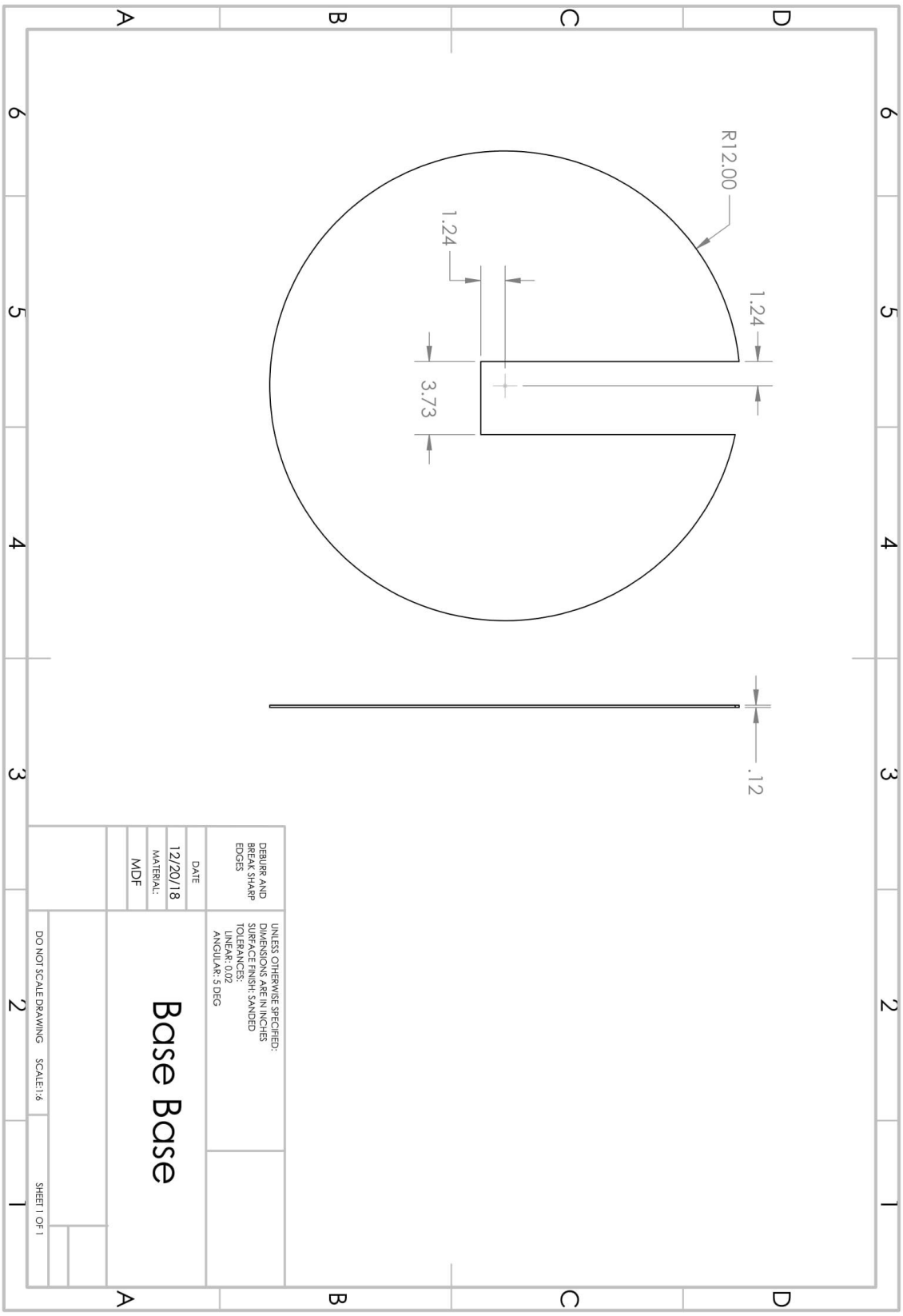
A2 - Engineering Drawings




1.50
8.50
18.00
.31
.11
.11
2.23
3.15
10 X Ø .13
Ø .25
.31
2.50
1.73
.51
2.37
3.26
1.12
.43
1.77
2.38
2.99
3.07
3.50
Ø .38
.12
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
SURFACE FINISH: SANDED
TOLERANCES:
LINEAR: 0.02
ANGULAR: 5 DEG
DEBURR AND
BREAK SHARP
EDGES
DATE
12/20/18
MATERIAL:
MDF
Top Base
DO NOT SCALE DRAWING
SCALE:1:2
SHEET 1 OF 1


17.76
17.35
2.63
1.50
48 X ∅ .30
48 X ∅ .13
24 X15.00°
∅ .25
6 X ∅ .13 X 60°
1.25
∅ 36.00
.12
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
SURFACE FINISH: SANDED
TOLERANCES:
LINEAR: 0.02
ANGULAR: 5 DEG
DEBURR AND BREAK SHARP EDGES
DATE
12/20/18
MATERIAL:
MDF
Main Disc
DO NOT SCALE DRAWING
SCALE:1:7
SHEET 1 OF 1


Base Side
DEBURR AND BREAK SHARP EDGES
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
SURFACE FINISH: SANDED
TOLERANCES:
LINEAR: 0.02
ANGULAR: 5 DEG
DATE
12/20/18
MATERIAL:
MDF
DO NOT SCALE DRAWING
SCALE:1:2
SHEET 1 OF 1
2.00
1.27
.52
.31
2.50
3.13
6.55
9.50
18.00
.12
19.75
.47
.74
7 X Ø .13
6.96
3.25
1.43
.56
1.75


R12.00
1.24
1.24
3.73
.12
DEBURR AND BREAK SHARP EDGES
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
SURFACE FINISH: SANDED
TOLERANCES:
LINEAR: 0.02
ANGULAR: 5 DEG
DATE
12/20/18
MATERIAL:
MDF
Base Base
DO NOT SCALE DRAWING
SCALE:1:6
SHEET 1 OF 1