Yang Hu

Professor Peter Jansen

ISTA-303: Introduction to Creative Coding

22 October 2018

ISTA 303: Assignment 2

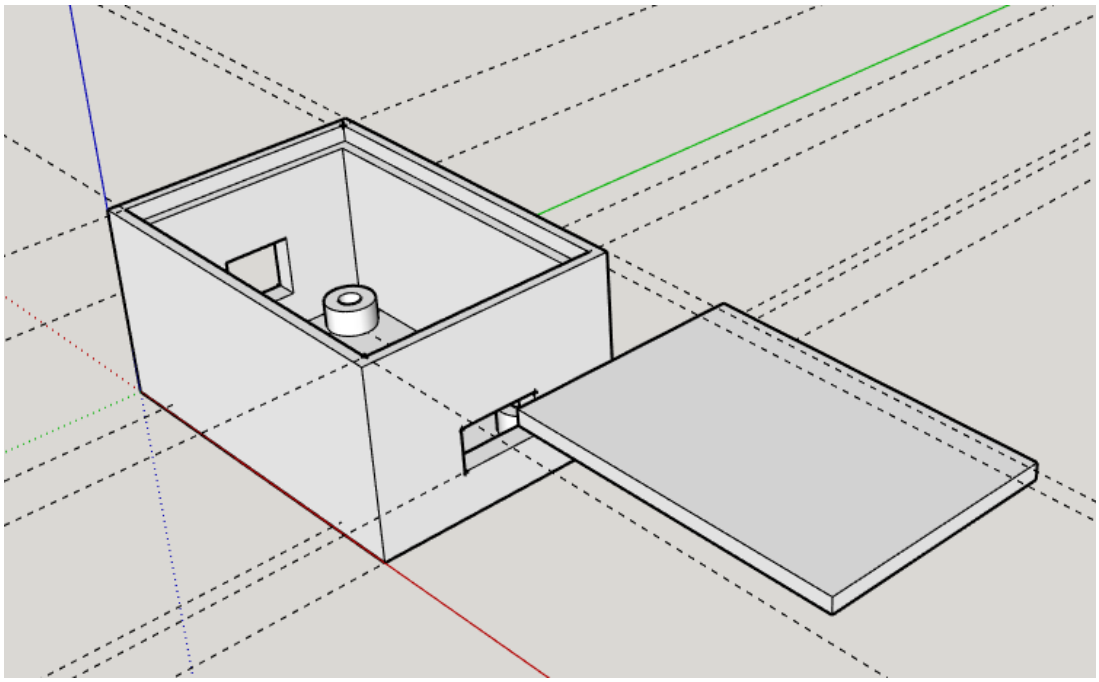
Ultrasonic Sonar with Visualization

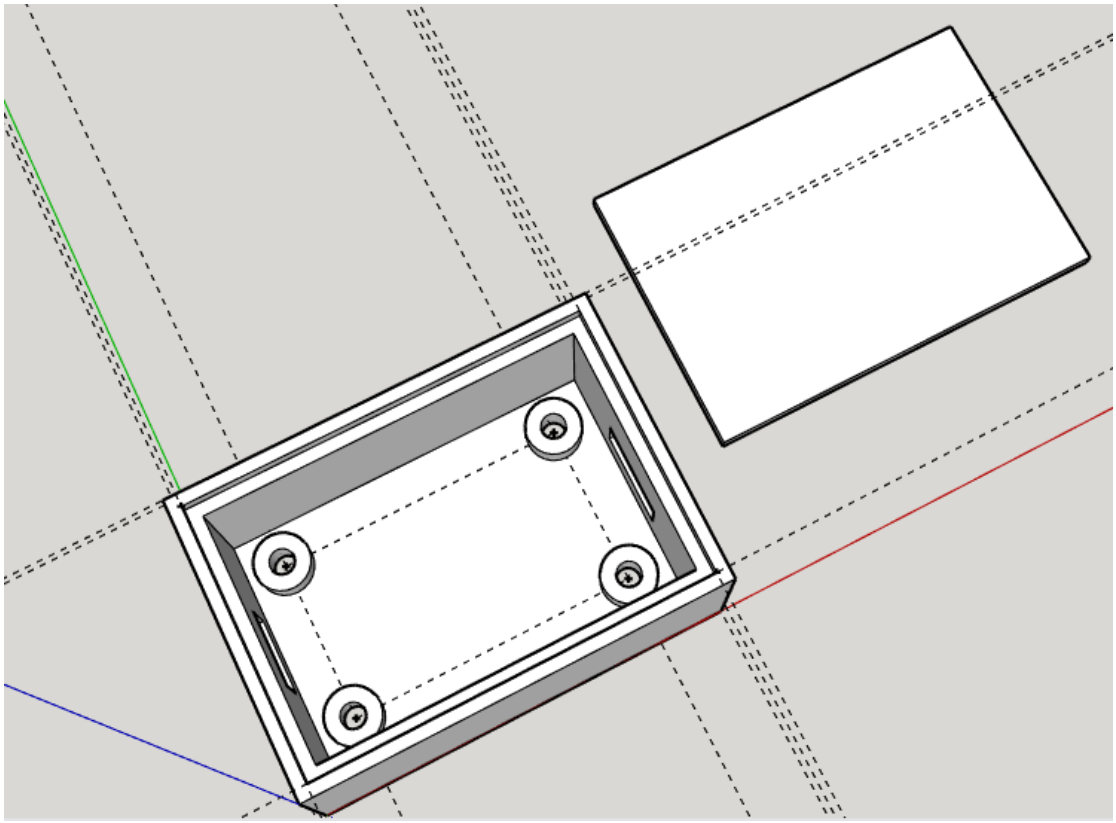
Documentation

Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

Part 1: 3D Foundations





Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

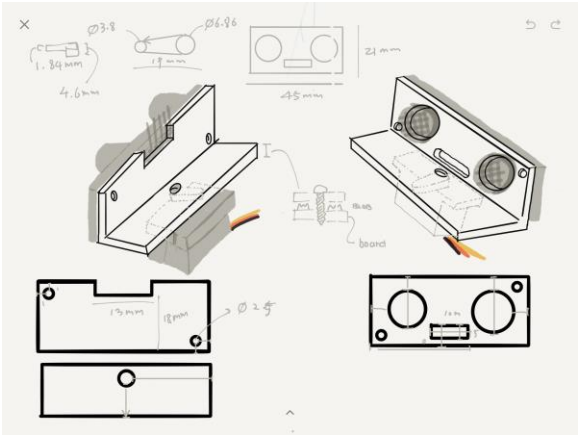
Part 2: 3D/2.5D Bracket

My design steps for each bracket are:

1. Sketching and measuring (on paper)
2. (2.5D) Paper prototype
3. Modeling in corresponding software
4. (2.5D) iteration by redo individual pieces
5. (3D) iteration by re-printing

3D Design

1. Sketching and measuring (on paper). I included all of the possibly useful data to make the paper design helpful for both 3D and 2.5D design.

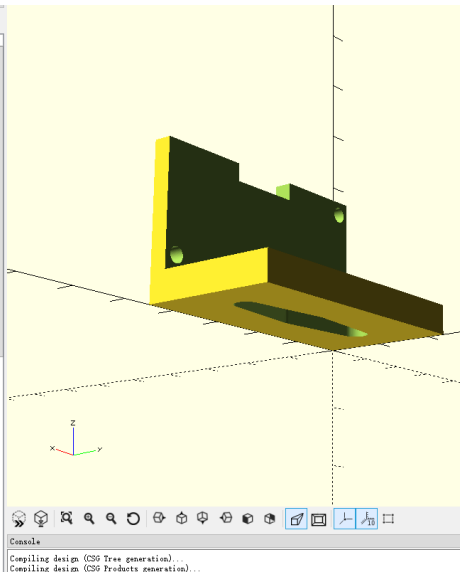


2. Design in openSCAD. The version I chose didn't have the round holes for plugging in the sensor. It is because that 3D printing iteration is more "expensive", and keeping design simple can reduce the risk of re-printing.

Yang Hu

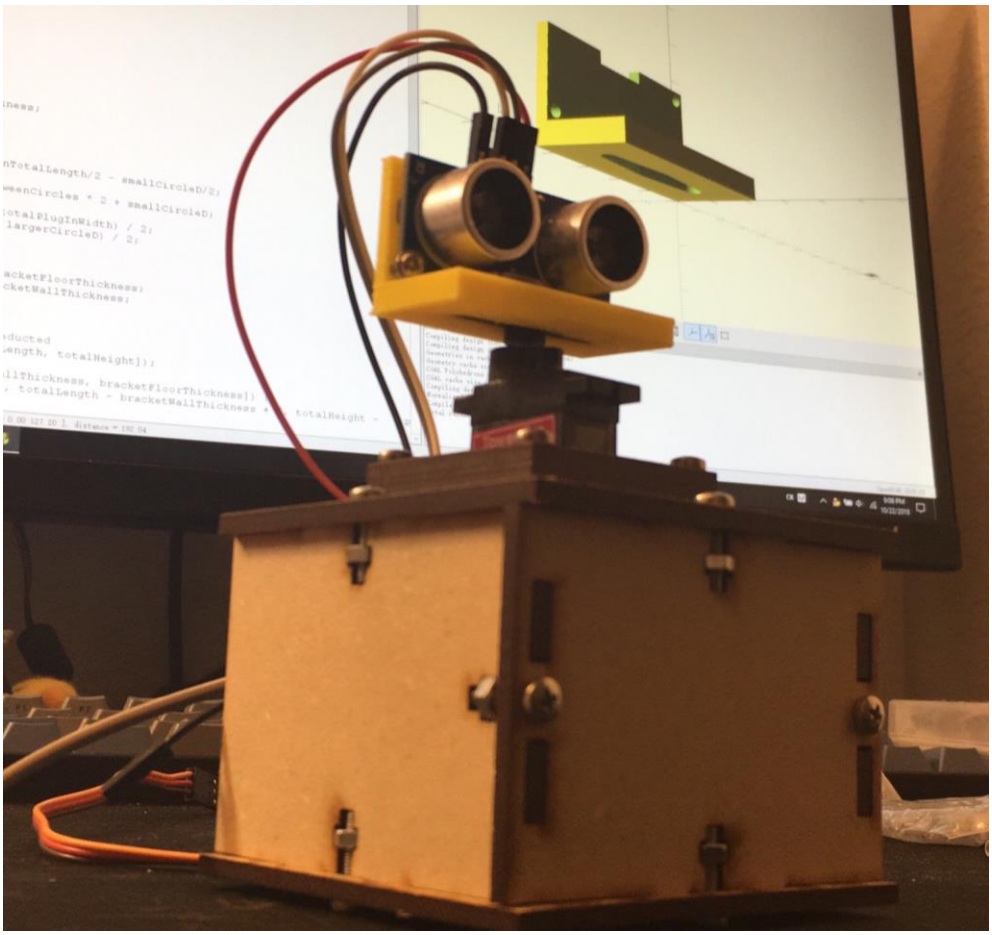
ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

```
bracketWallThickness = 3;
bracketFloorThickness = bracketWallThickness + 2;
// wall
sensorHeight = 21;
sensorWidth = 45;

screwHoleSize = 2.5;
screwHoleOffset = 1 + screwHoleSize/2;

// PinBar
pinBarOffsetX = 16;
pinBarOffsetZ = 18;
pinBarWidth = 13;
pinBarHeight = 3;


// floor
floorWidth = sensorWidth;
floorLength = 20;


// plugInServo
plugInDepth = bracketWallThickness;
smallCircleD = 3.9;
largerCircleD = 6.9;

plugInTotalLength = 31.1;
distanceBetweenCircles = plugInTotalLength/2 - smallCircleD/2;

totalPlugInWidth = distanceBetweenCircles * 2 + smallCircleD;

plugInOffsetX = (floorWidth - totalPlugInWidth) / 2;
plugInOffsetY = (floorLength - largerCircleD) / 2;

// total
totalWidth = sensorWidth;
totalHeight = sensorHeight + bracketFloorThickness;
totalLength = floorLength + bracketWallThickness;
```



3. Iteration. This simpler design helped me to finish the mount by single attempt.

   The first version fits in everything perfectly

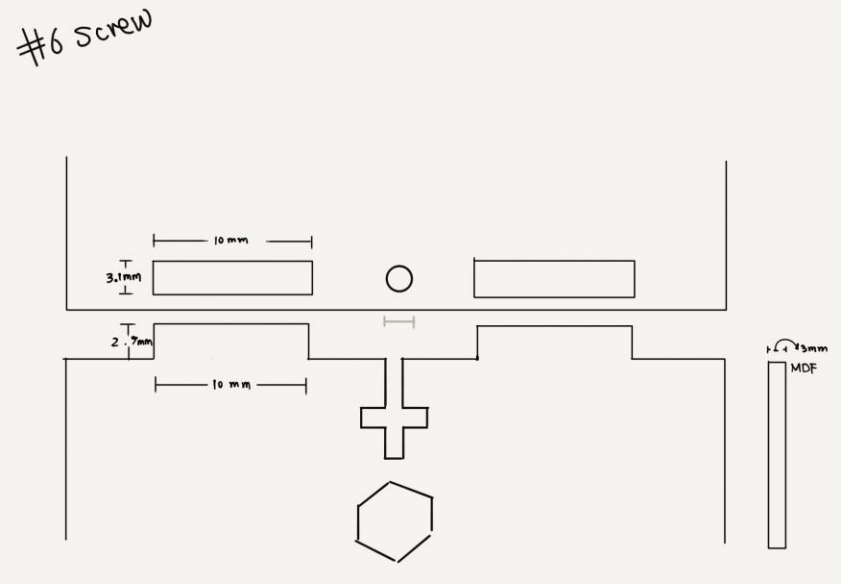


Yang Hu

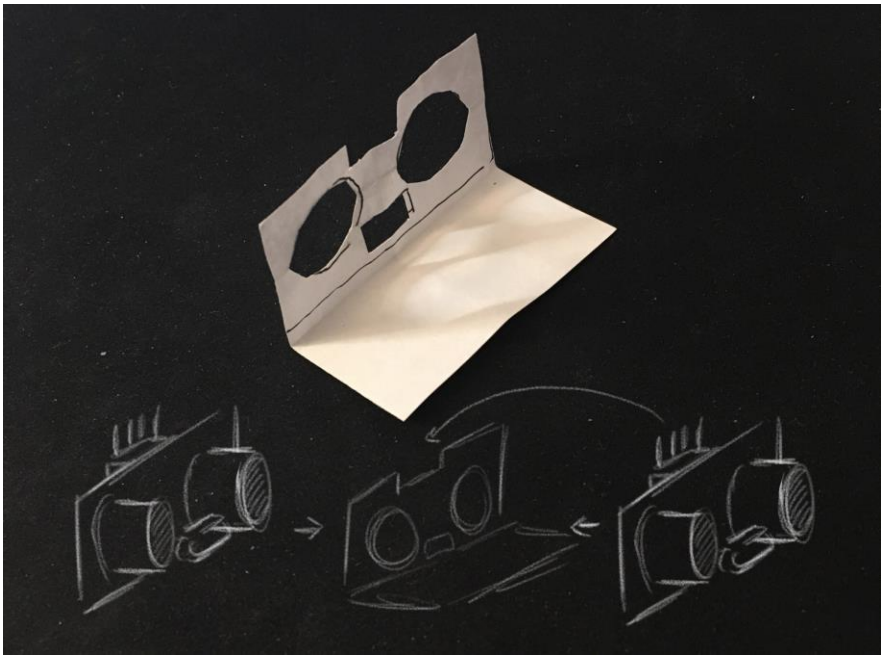
ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

2.5D Design.

1. Sketching and measuring (on paper). This step shares data with 3D. The only new data is the joint part.



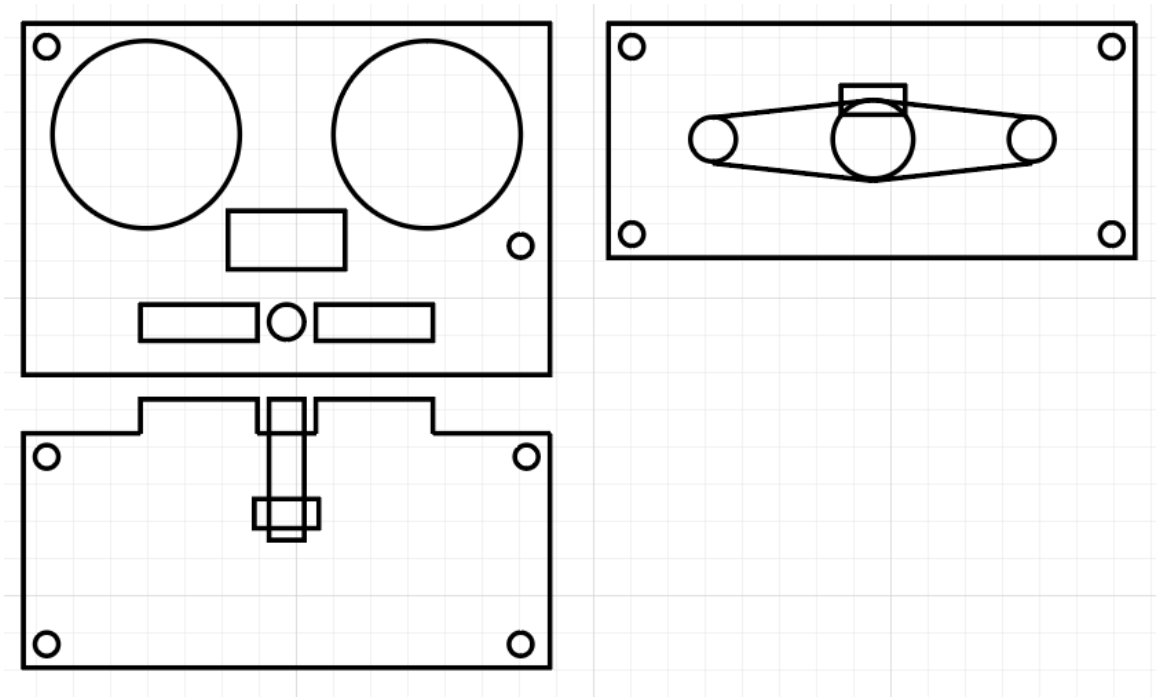
2. Paper prototype. I did some simple paper prototypes before designing in Illustrator. Basically, I used paper prototype to determine where and how should design my joint.



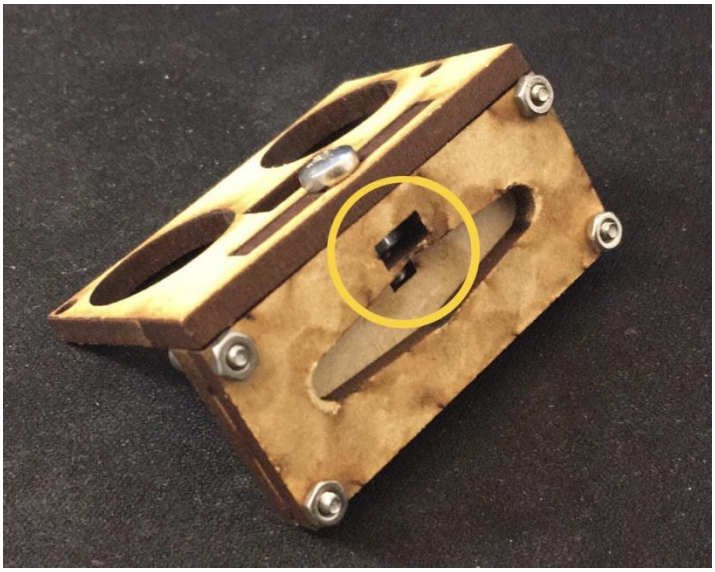
Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

3. Design in Illustrator



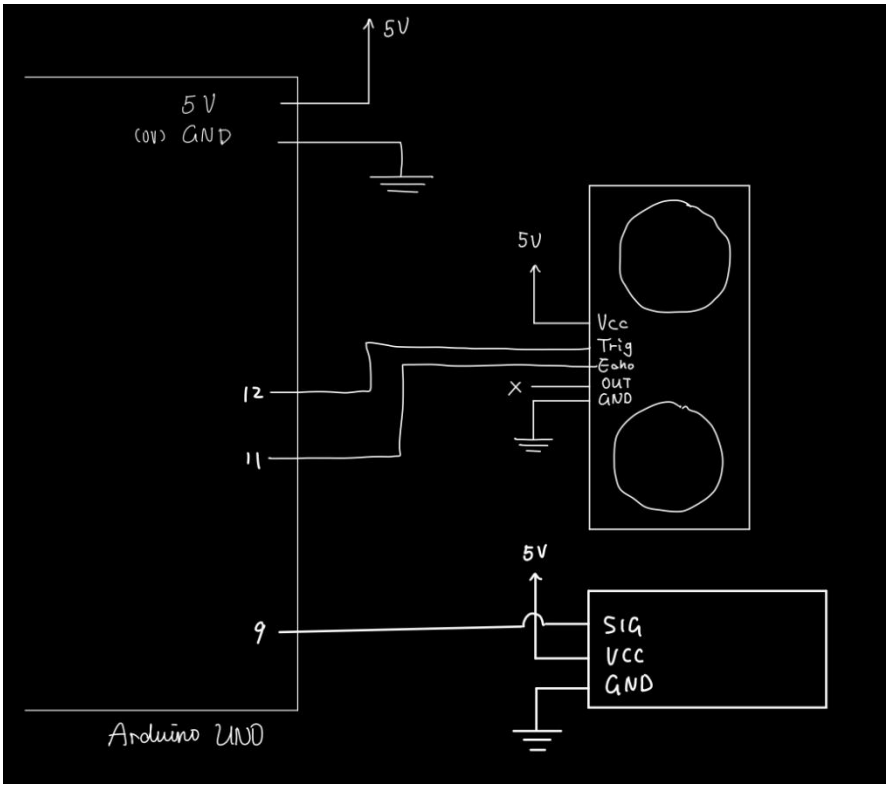
4. Iteration by redo individual pieces. One great aspect about 2.5D design is that we can redo single parts instead of the whole structure during iteration. For example, when designing the bottom two-layer structure, I met with a problem that the joint screw nut stuck out, and I cannot make one layer sit tightly on another. Therefore I redid the bottom layer, which added a small hole for the screw nut.



Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

Part 3: Circuit Schematic



Part 4: Arduino + Sensor

The basic logic of my sensor code is

1. Rotate the servo into an angle
2. Trigger the sensor, and get the time/distance result
3. (optional) calculate the result

During my coding, I found two useful way (different from the original) to achieve the step 2: pulseIn(ECHO_PIN, HIGH, MAX_DIST) function and <NewPing.h> library.

Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

```
float measureDistanceCM() {
  // Hold the trigger pin high for at least 10 us
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  /* pulseIn(pin, value, timeout): Reads a pulse (either HIGH or LOW)
   * on a pin. returns 0 if out of range
   * in this case, it starts timing when pin goes from LOW to HIGH,
   * then stops timing when the pin goes back to LOW
   * it returns the time interval in microseconds
   */
  long duration = pulseIn(ECHO_PIN, HIGH, MAX_DIST);
  /* 0.034cm/microseconds is the sound speed
   * distance = time * speed
   * divide by 2 because the ultra sound goes back and forth
   */
  return duration * 0.034 / 2;
```

The NewPing library provides simpler solution.

#include <NewPing.h>

NewPing sonar(TRIG_PIN, ECHO_PIN, MAX_DIST);

float range = sonar.ping_cm(); // This directly returns the centimeter result

Part 5: Extensions

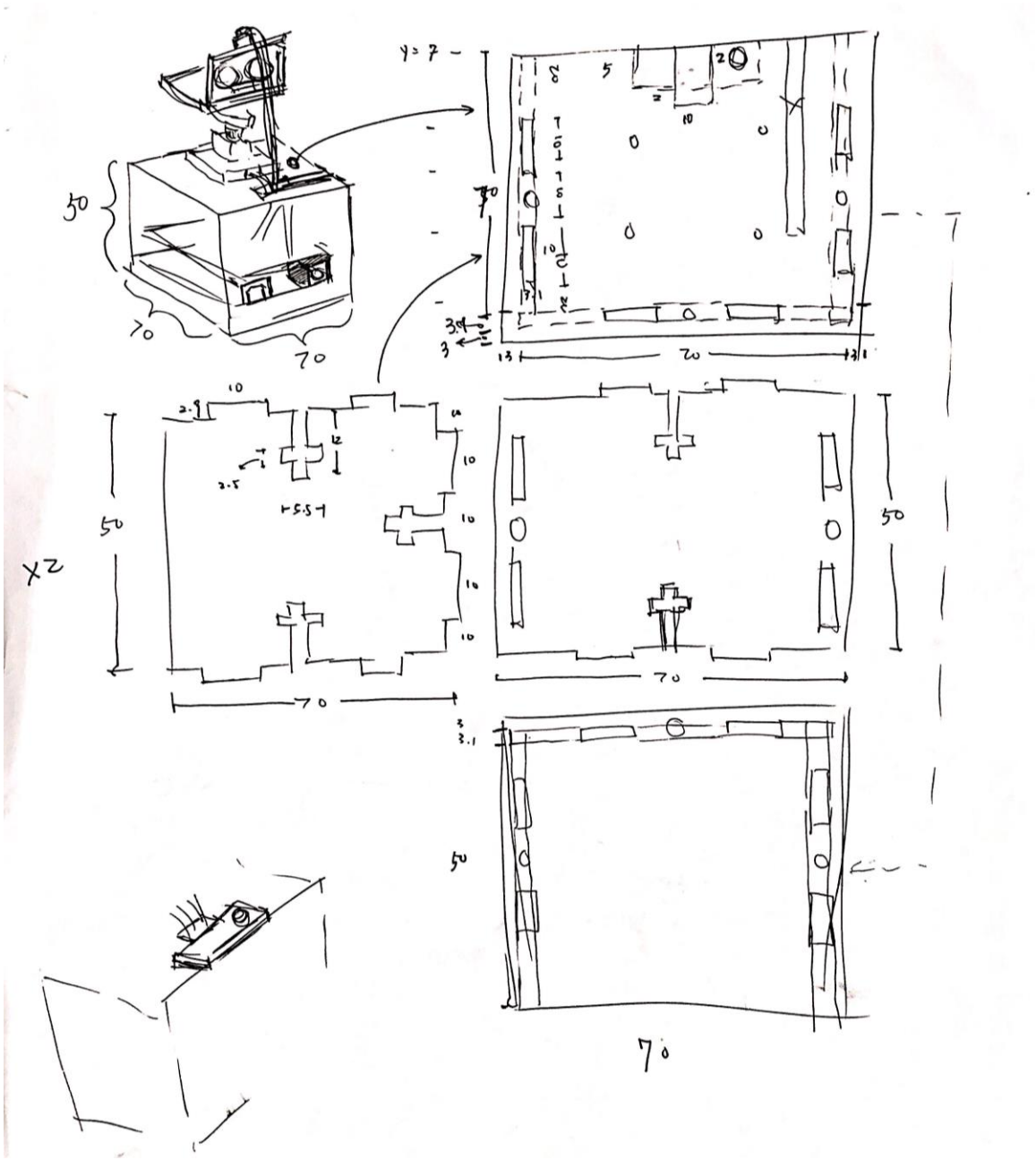
1. Servo/Arduino Holder

Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

This design follows my 2.5D steps mentioned early in this documentation. Which are sketching and measuring (on paper), modeling in corresponding software and iterating by redo individual pieces

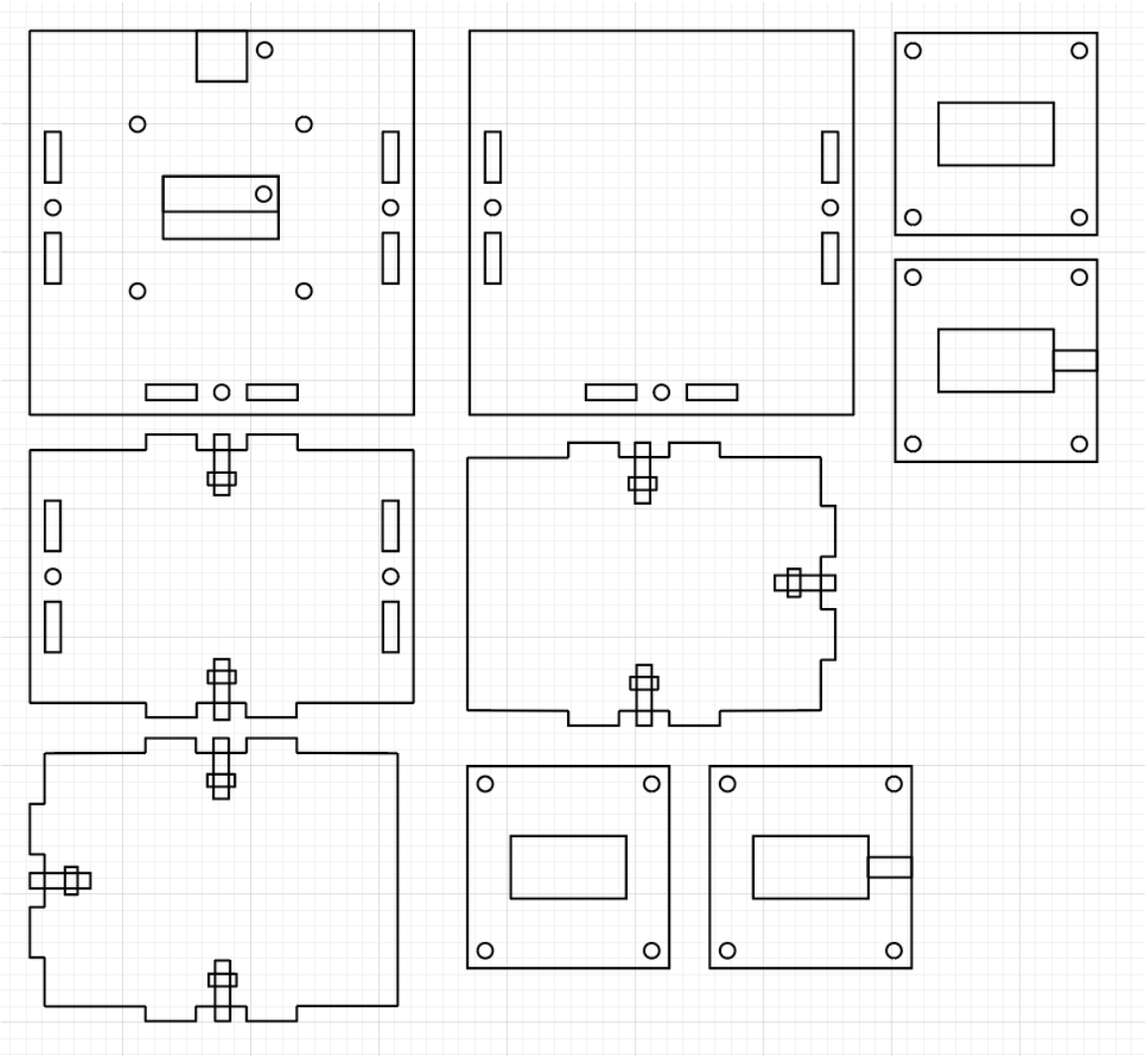
a. Design and measuring.



Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

b. Illustrator design



c. Design choices

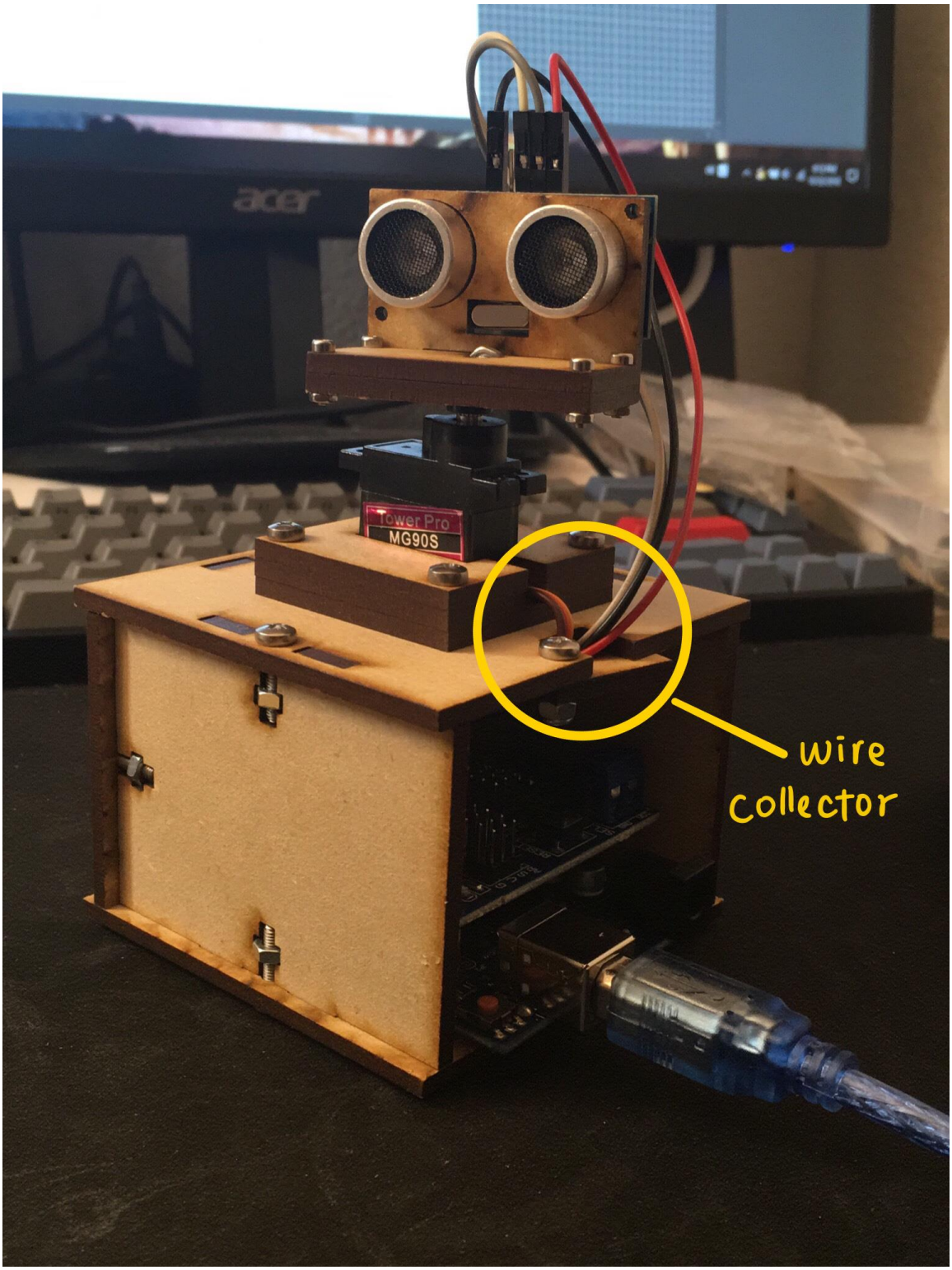
a. Half-open box. I didn't let the box encapsulate the Arduino completely for iteration and reuse concerns. The half-open structure allows me to take out everything at any time while provides solid stand for the whole structure.

b. The wire collector "gate". Similar reasoning as choice a. The gate can collect all wires to hold them tightly and make them look better.

Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

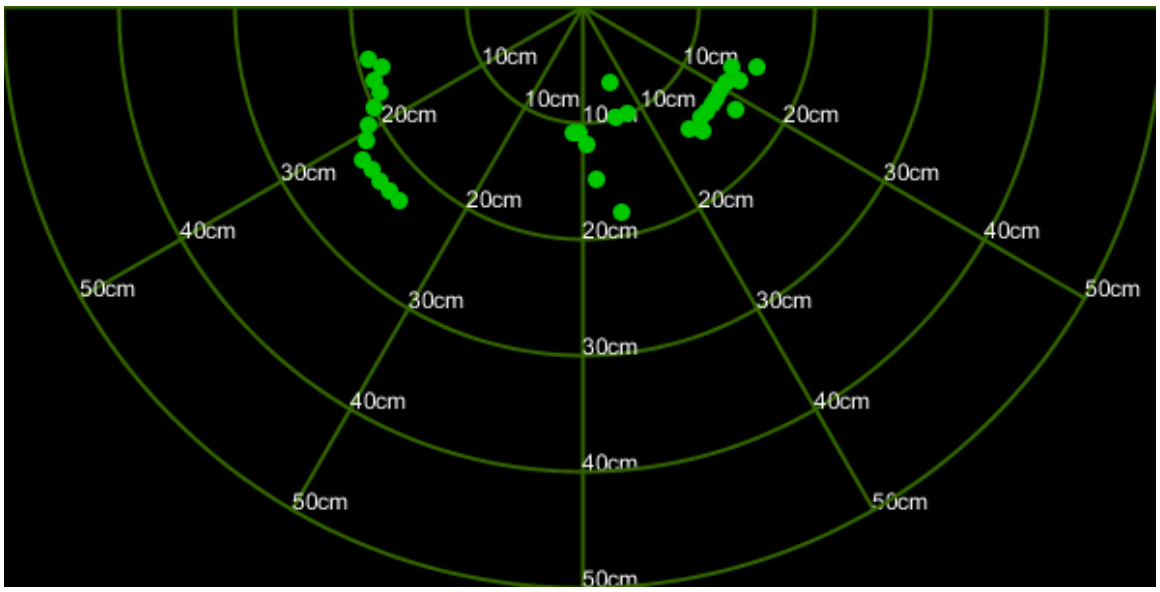
d. Outcome after the iterations.



Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

2. Processing Visualization



I made a radar-like interface to visualize sensor results. It basically works in these steps, in each draw():

a. Draw background. This cleans the canvas.

b. Draw radar arcs, text and lines. This is a function call which contains several for-loops to draw similar arcs, text and lines.

c. Draw all dots (stored in a 2D array [index (related to angle)][x (angle), y (distance in cm)]). The 2D array get updated ”back-and-forth”, for example, when the radar starts from 0, data are stored in normal way (index 0, 1, 2 …). However, when it reaches 180 degree, which is index 59 , the next data will be in index 59, and then 58, 57, … This makes the effect that the radar is actually “updated”, because in each draw (except the very first loop), the new data is replacing the old one.

Yang Hu

ISTA-303 Assignment 2: Ultrasonic Sonar with Visualization

Part 6: References

NewPing library. *Arduino*, https://playground.arduino.cc/Code/NewPing