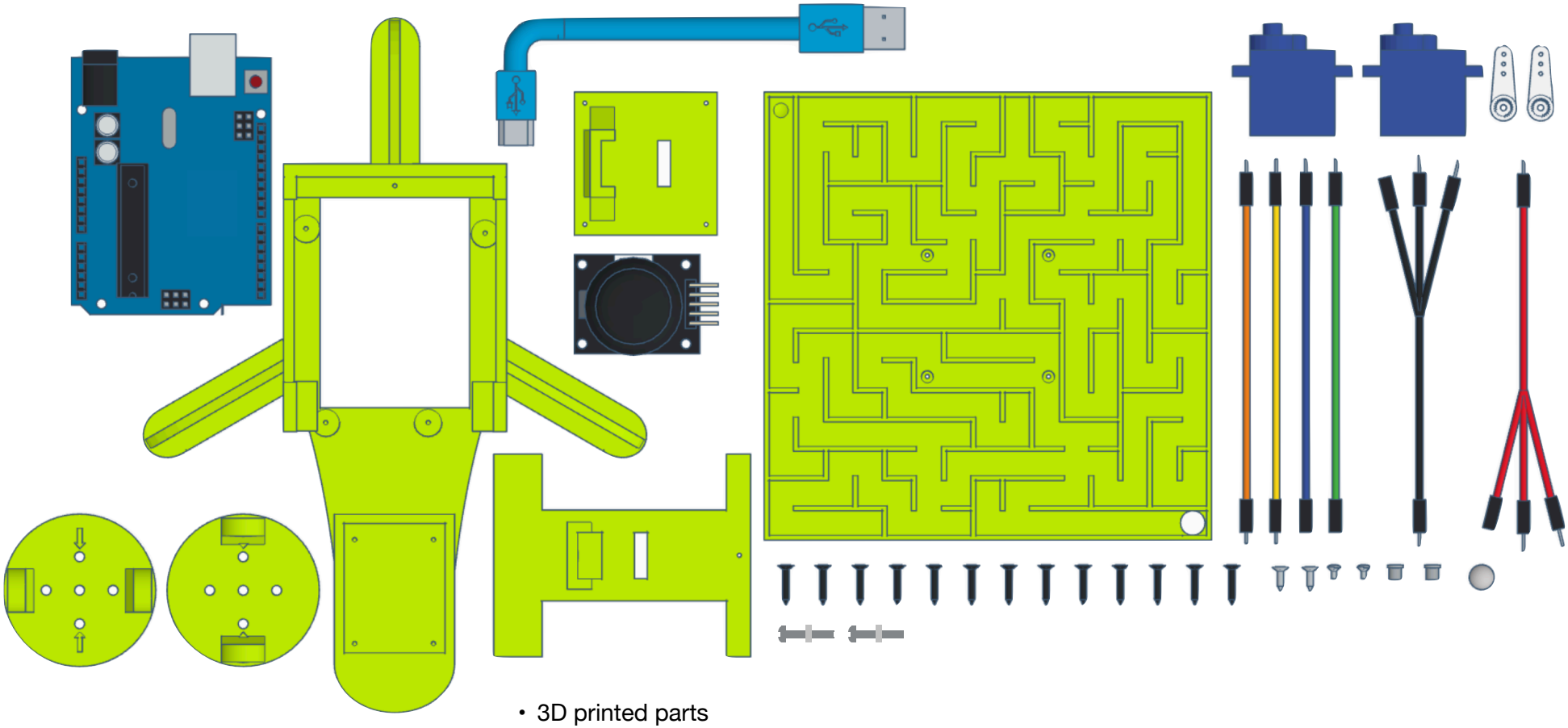
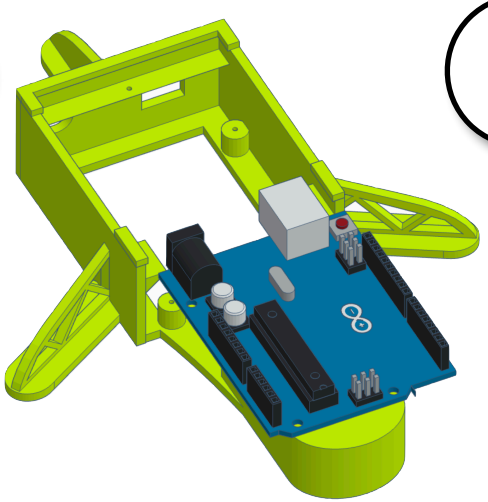
Maze Robot Guide

Materials

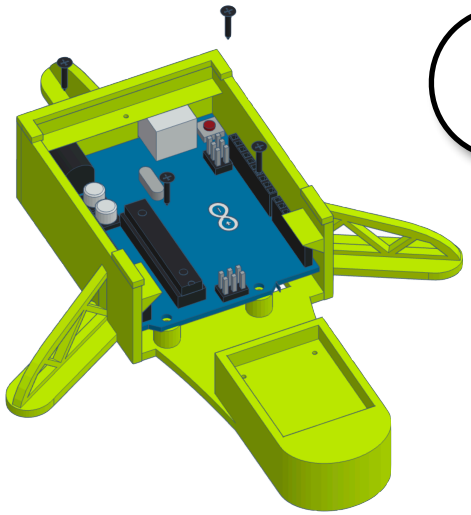


- 3D printed parts
- UNO R3 Board ATmega328P (Arduino Uno R3 equivalent)
- 2 SG90 micro servos w/ servo arms and screws
- KY023 joystick
- 13 Metric M2 Phillips head self tapping screws
- 4 male to male jumper wires
- 4 male to female jumper wires
- 1/4” steel ball
- 2 1/8 x 5/16 rivets (just the head)
- 2x - 4-40 x 1/2" machine screw
- 2x - 4-40 machine screw nut

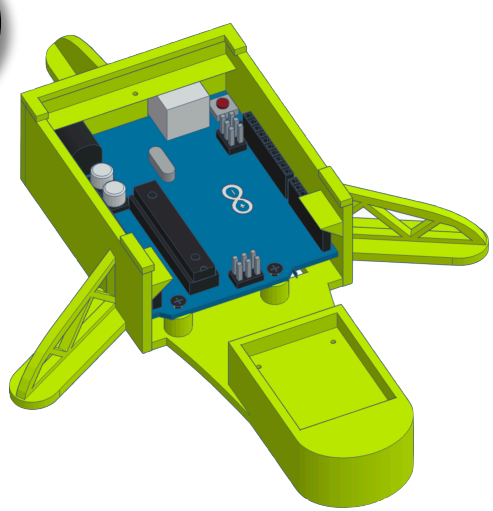
1



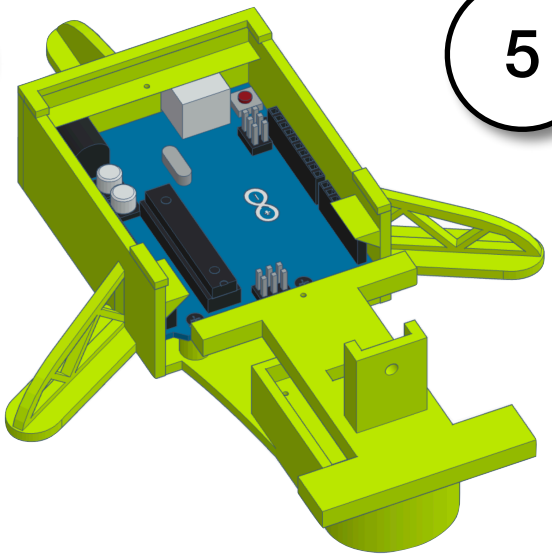
2



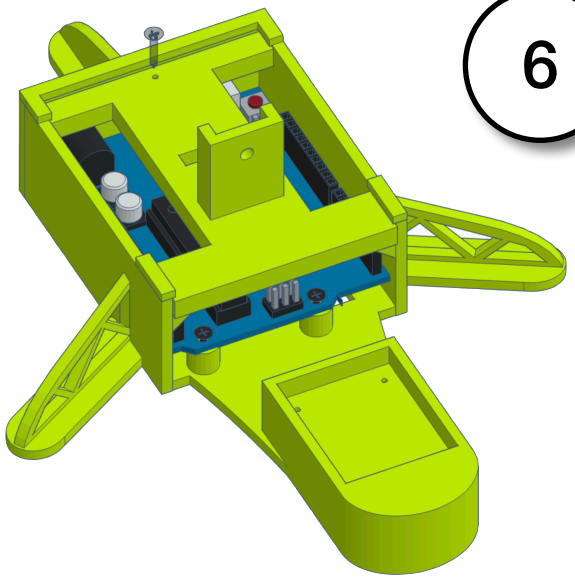
3



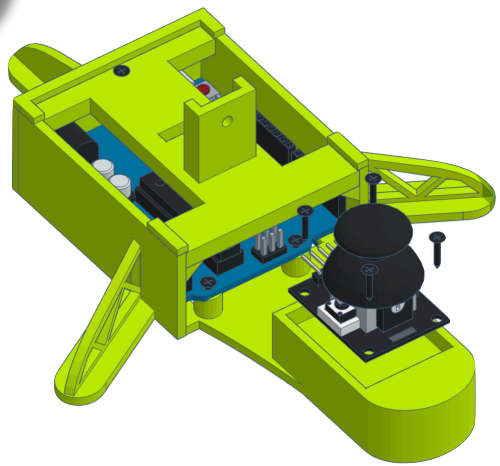
4



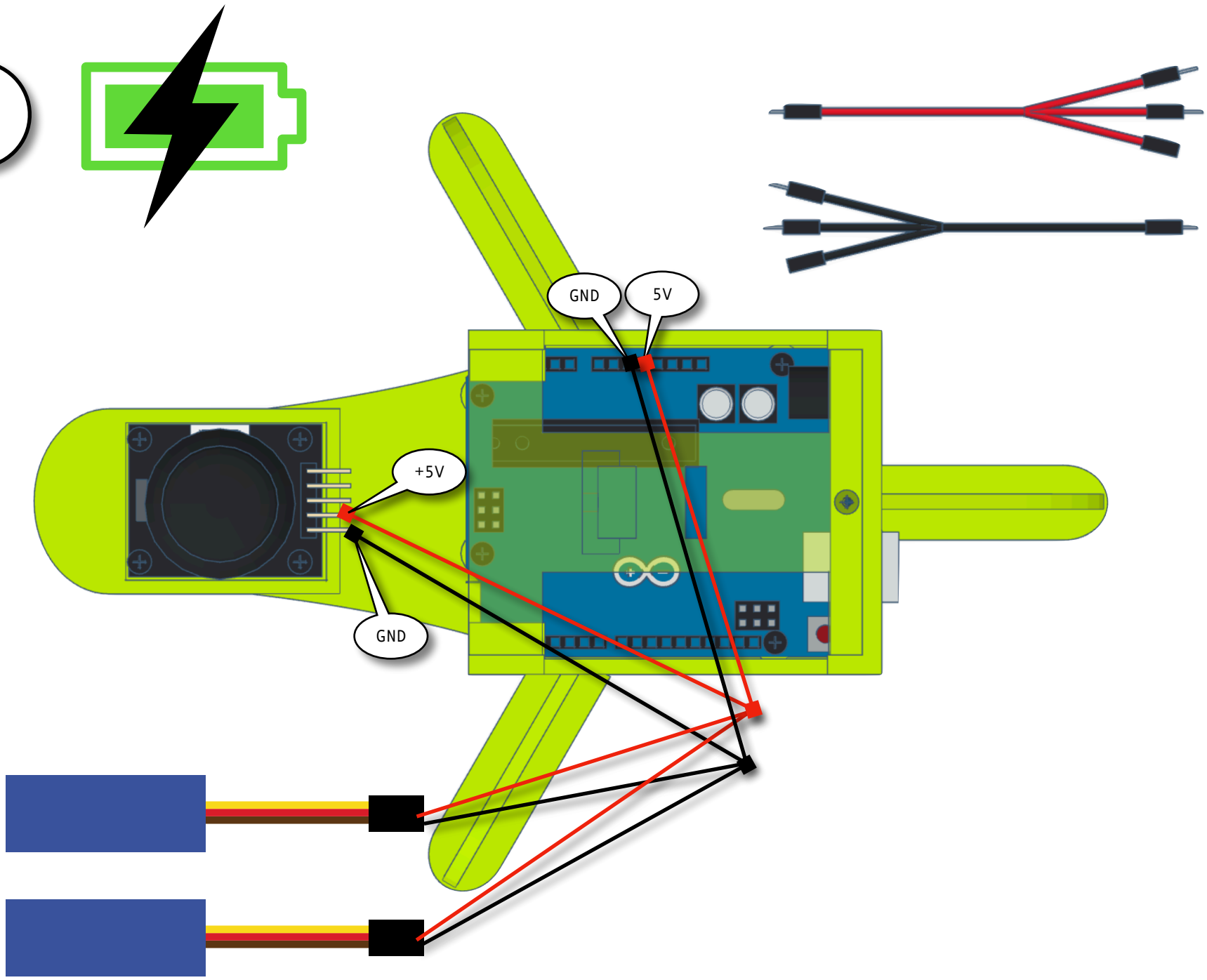
5



6



7



Maze Robot Program Part 1

8

First you need to tell the Arduino how to use servos. You do this by adding the servo library and giving your servos names.

Type this:

```
#include <Servo.h>

Servo Xservo;
Servo Yservo;
```

Next you will tell the Arduino all of your variables. Arduino won't know how to do anything unless you first do this.

They will look like this:

```
int var=#;
```

int = integer (a math word that means whole number)

var = variable (a data item that may take on more than one value during the runtime of a program)

Var can be anything you want, but it should make sense to you.

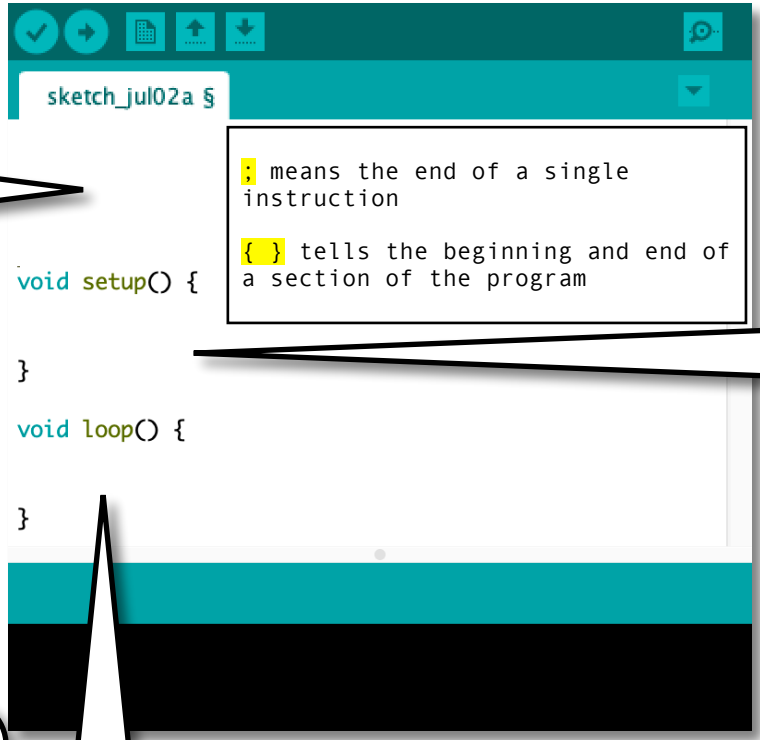
Our first 2 variables will tell our Arduino information about our joystick.

Type this:

```
int VRXpin=A0;
int VRYpin=A1;
```

VRX is the pin on our joystick and A0 is analog input 0 on the Arduino.

VRY is another pin on our joystick and A1 is analog input 1 on the Arduino.



10

void loop () is where we write the instructions that will run over and over until we turn it off.

We will be telling the Arduino to move our servos (OUTPUT) a certain way when we move the joystick (INPUT).

Our first instruction will take a number from the joystick, plug it into a math equation and solve it. Then it will use the answer to that math equation to tell the servo where to move. You will type:

```
Xval=analogRead(VRYpin);
WVx=(Xrange/1023.)*Xval*(-1)+Xrange+adjustX;
Yval=analogRead(VRXpin);
WVy=(Yrange/1023.)*Yval+adjustY;
```

***You might notice that our VRXpin and VRYpin are working with the opposite servo. This is because we are using the joystick backwards to fit into our robot.

9

void setup() is the section where we write the instructions that Arduino will do only once when we first plug it in.

We will first define our pins so the Arduino will know what we connected to it and what it should do with it.

```
pinMode(VRXpin, INPUT);
pinMode(VRYpin, INPUT);
```

VRXpin is the variable we set at the beginning that defines our VRX joystick being connected to A0 on our Arduino. We are now telling the Arduino that this is an input. This tells the Arduino to look for information from our joystick when we move it. Later we will also define outputs so it will know how to talk to our servos to move. VRYpin is for when we move the joystick the other direction.

X = move left to right
Y = move up and down

Maze Robot Program Part 2

```
#include <Servo.h>

Servo Xservo;
Servo Yservo;

int VRXpin=A0;
int VRYpin=A1;

int XServoPin=9;
int YServoPin=10;

int WVx;
int WVy;

int Xval;
int Yval;

int dt=100;

int Xrange=10;
int Yrange=20;

int adjustX= 0 ;
int adjustY=0;
```

; Be sure to end each line with a semicolon. It is just like ending a sentence with a period.

```
void setup() {

pinMode(VRXpin, INPUT);
pinMode(VRYpin, INPUT);

pinMode(XServoPin, OUTPUT);
pinMode(YServoPin, OUTPUT);

Xservo.attach(XServoPin);
Yservo.attach(YServoPin);
}
```

```
void loop() {

Xval=analogRead(VRYpin);
WVx=(Xrange/1023.)*Xval*(-1)+Xrange+adjustX;
Yval=analogRead(VRXpin);
WVy=(Yrange/1023.)*Yval+adjustY;

Xservo.write(WVx);
Yservo.write(WVy);

delay(dt);
}
```

11

These are the rest of your variables.

```
int XServoPin=9;
int YServoPin=10;
```

These variables tell the Arduino where you connected the servos.

```
int WVx;
int WVy;
```

WV stands for “Write Value”. Each of these define the variable used to tell the angle to which the servo will move. These numbers will be the answer to the math equation Arduino is solving using the number given from the joystick.

```
int Xval;
int Yval;
```

These define the variables that will be the numbers coming from the joysticks.

```
int dt=100;
```

dt stands for “delay time”. This variable defines the wait time in the program.

```
int Xrange=10;
int Yrange=20;
```

These define the range of how far the servos will move. Without these the servo would move so far that it would launch the ball.

```
int adjustX= 0 ;
int adjustY=0;
```

These variables are used to adjust our servos so that they point straight up and balance the maze.

12

These are the rest of your setup instructions that your program will only run once.

```
pinMode(XServoPin, OUTPUT);
pinMode(YServoPin, OUTPUT);
```

These instructions tell the Arduino pins 9 and 10 are outputs going to the 2 servos.

```
Xservo.attach(XServoPin);
Yservo.attach(YServoPin);
}
```

These instructions attach our servos so that they can be controlled as a servo.

Be sure this sections ends with a bracket }

13

These are the rest of your instructions.

```
Xservo.write(WVx);
Yservo.write(WVy);
```

Remember WVx and WVy are the answers to the math equations solved by Arduino using the numbers inputed by the joystick. These instructions use WVx and WVy and write those values to the servos to make them move. Without these instructions, your servos wouldn’t do anything.

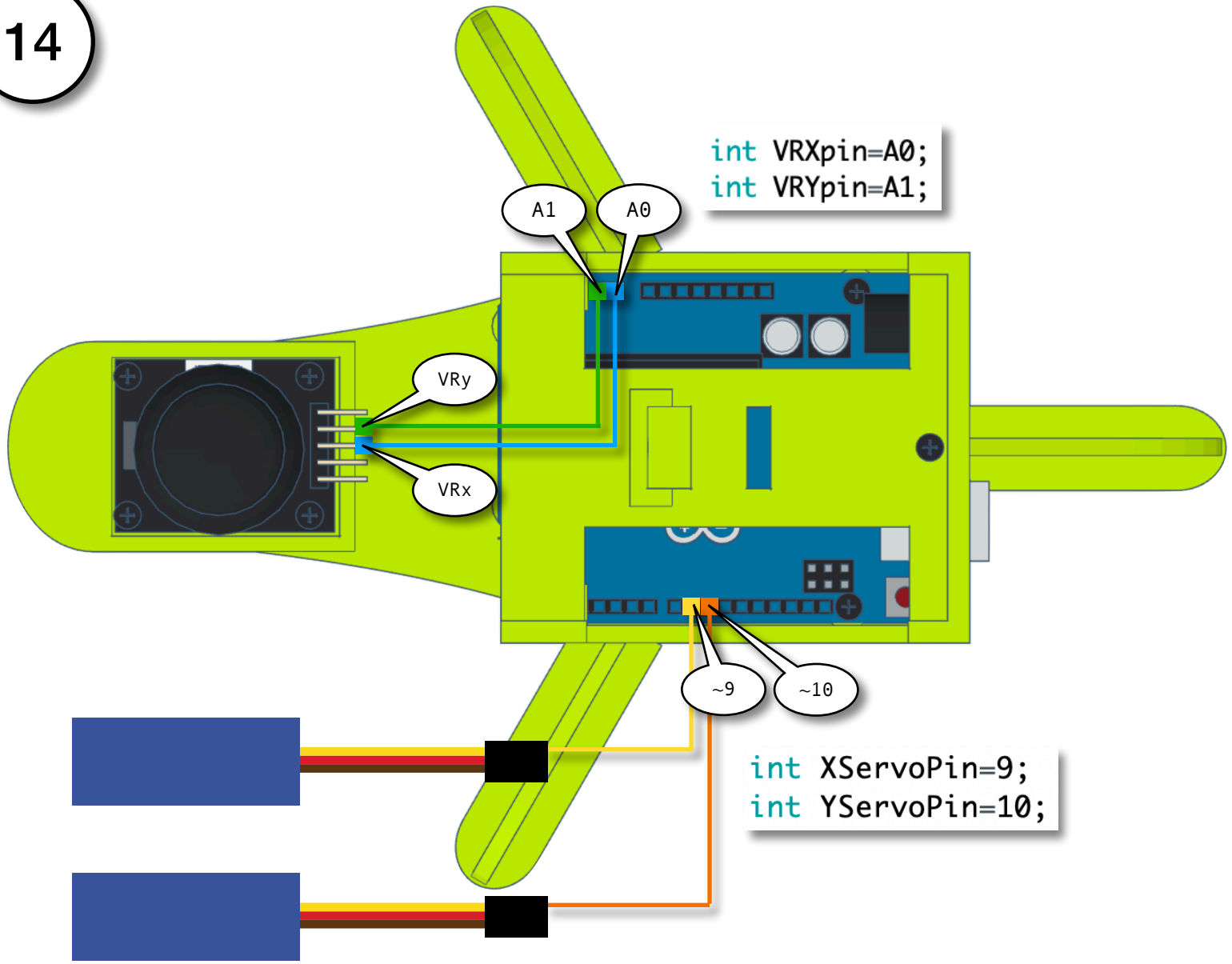
```
delay(dt);
}
```

This tells the Arduino to wait just a little bit before it repeats these instructions. Remember that everything in “void loop()” will repeat over and over until you turn it off.

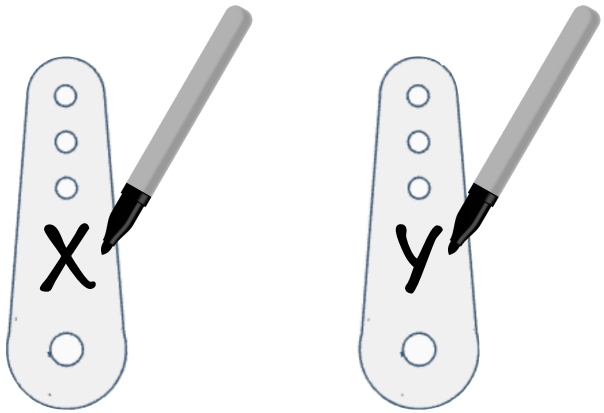
Notice that the section ends with a }.

If you forget your ; and }, your robot won’t work.

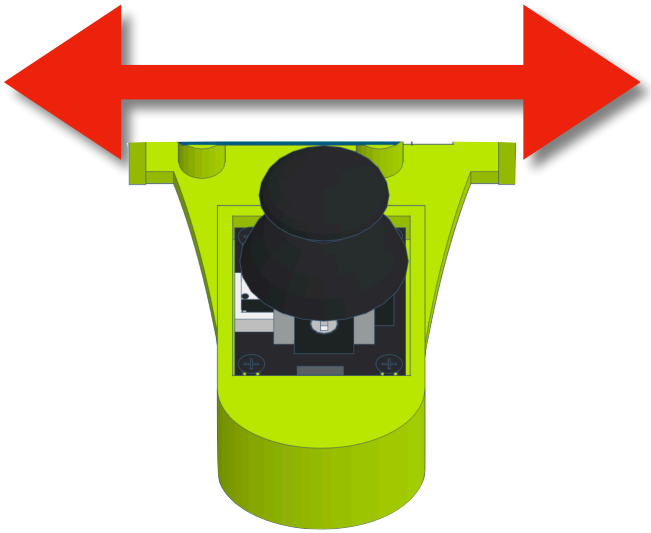
14


int VRXpin=A0;
int VRYpin=A1;
A1
A0
VRy
VRx
~9
~10
int XServoPin=9;
int YServoPin=10;

15

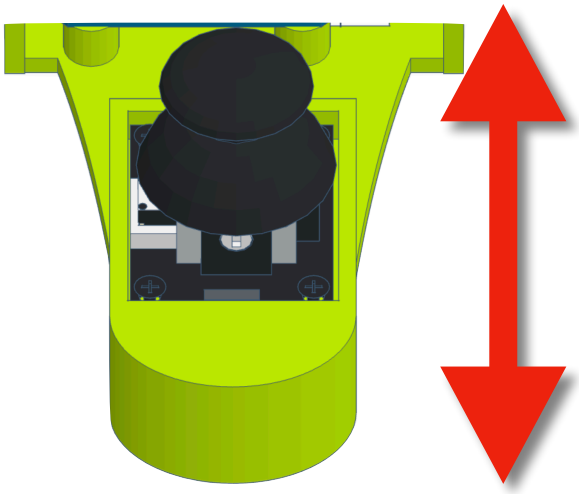


16



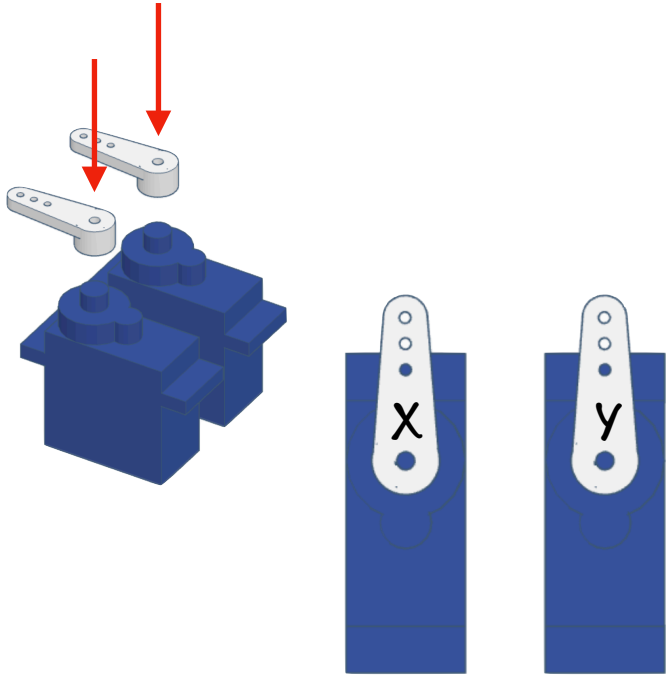
Look to see which servo moves when you move the joystick left and right. The servo that moves is the X servo.

17

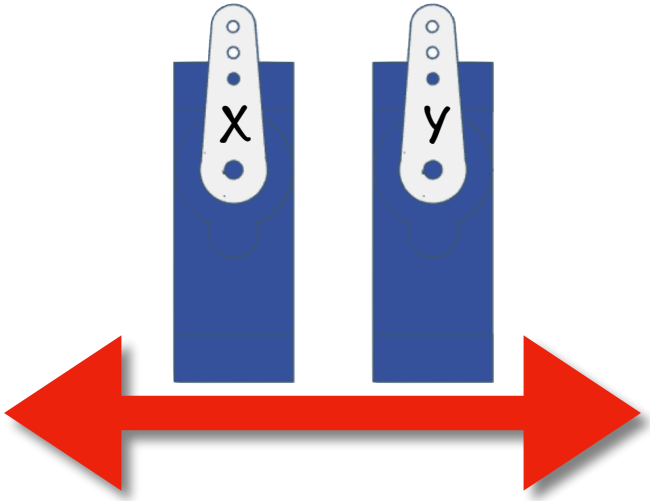


Move the joystick forward and backward. The servo that moves is the Y servo.

18

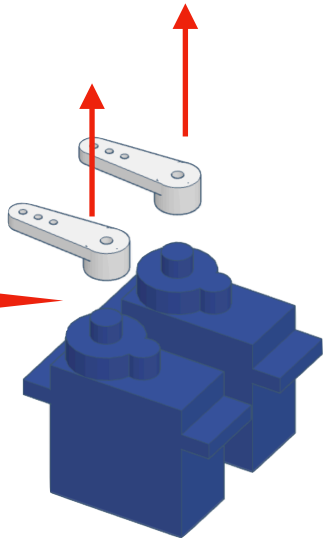


19



20

MOVING THE SERVO ARM BACK AND FORTH BY HAND WILL BREAK IT!!!



Move the joystick left and right and up and down. When the joystick is in the center, both servo arms should be pointing straight up. If they aren't you have to adjust the code in your program.

Go to the top section of your code and look for

```
int adjustX=0;
int adjustY=0;
```

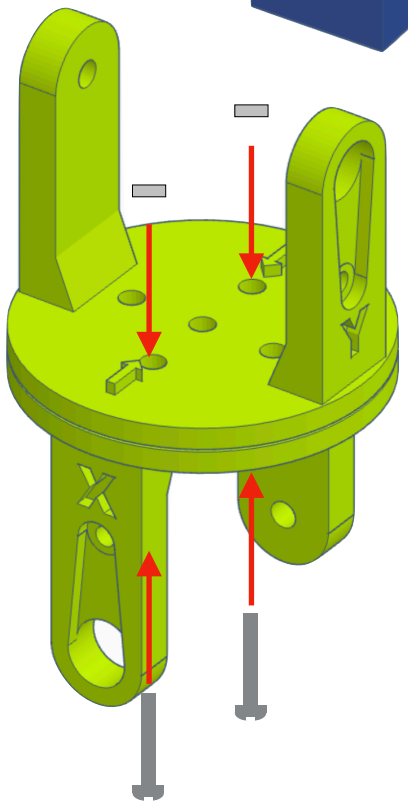
Go up or down 1 number at a time. adjustX is probably between 20 and 25. adjustY is probably between 4 and 10.

Like this:

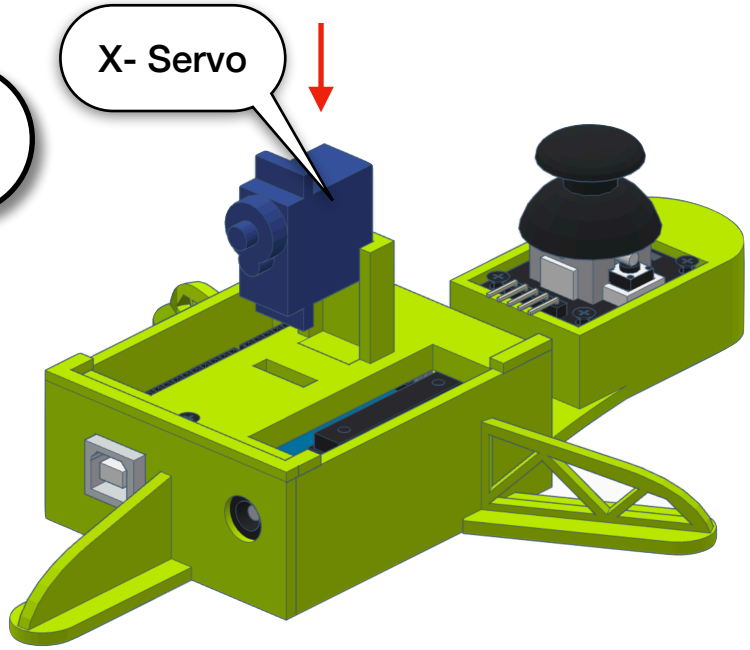
```
int adjustX=23;
int adjustY=7;
```

When you are finished, unplug the robot from the computer.

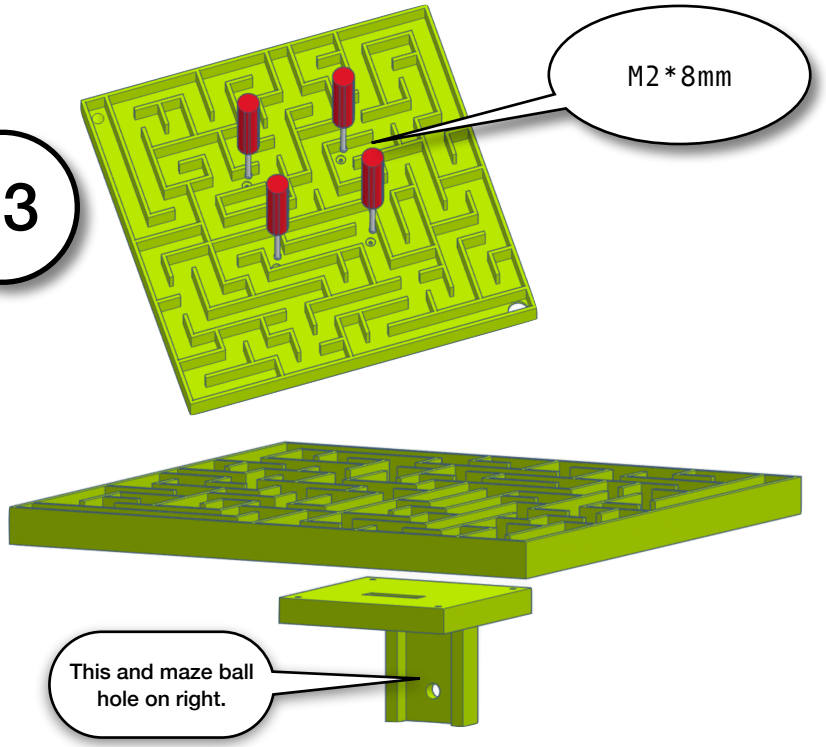
21



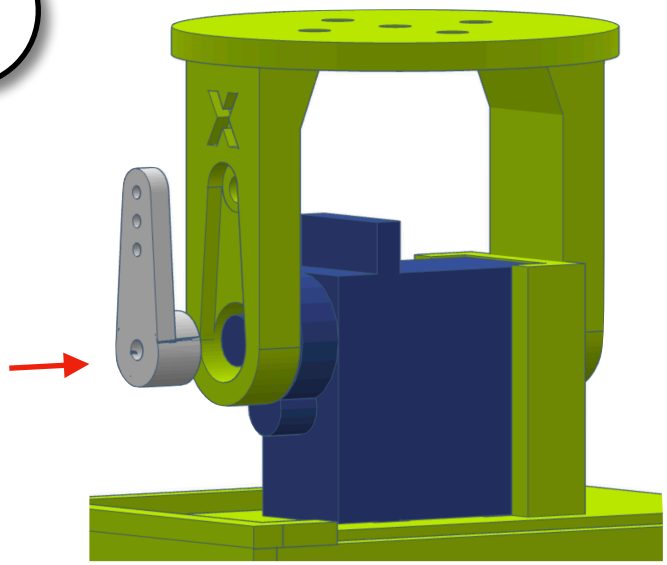
22



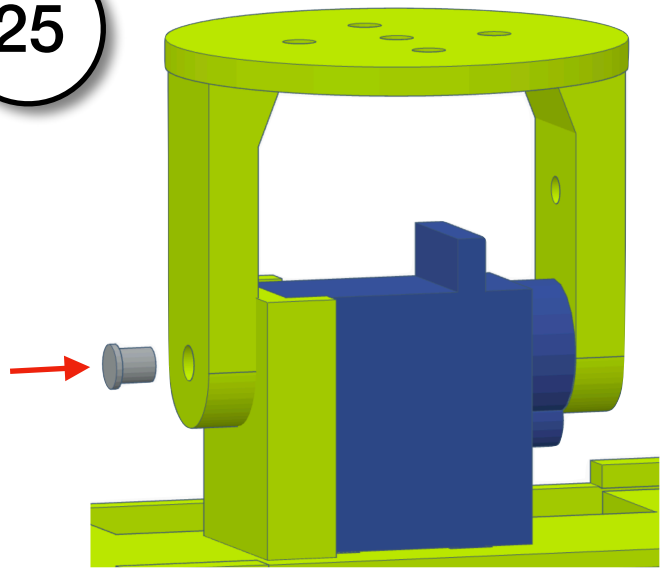
23



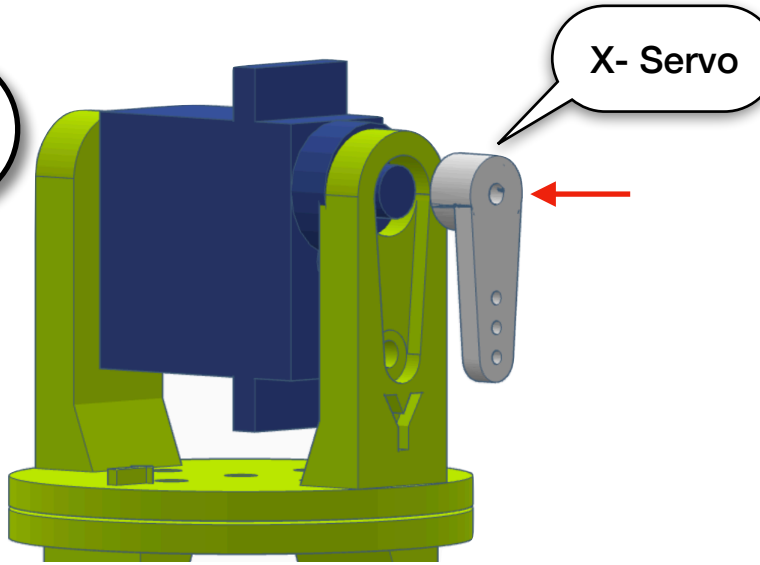
24



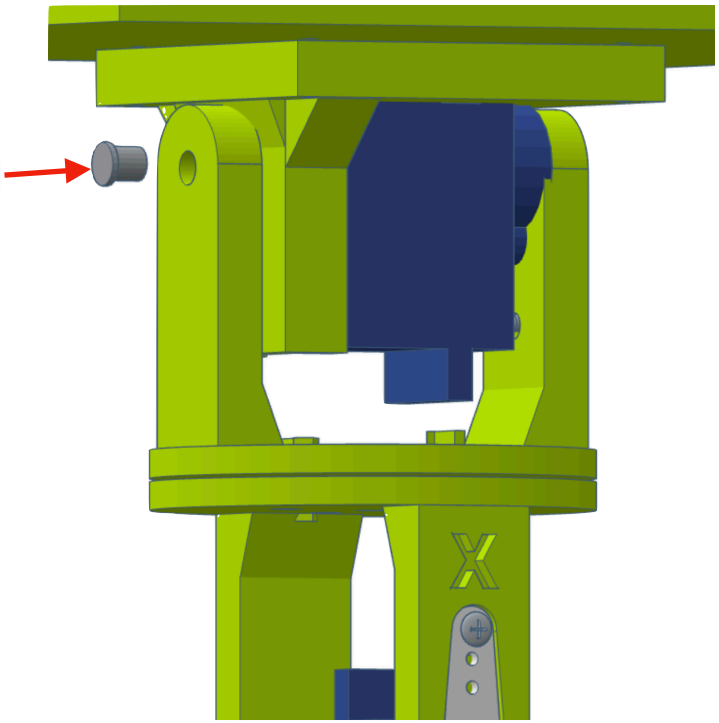
25



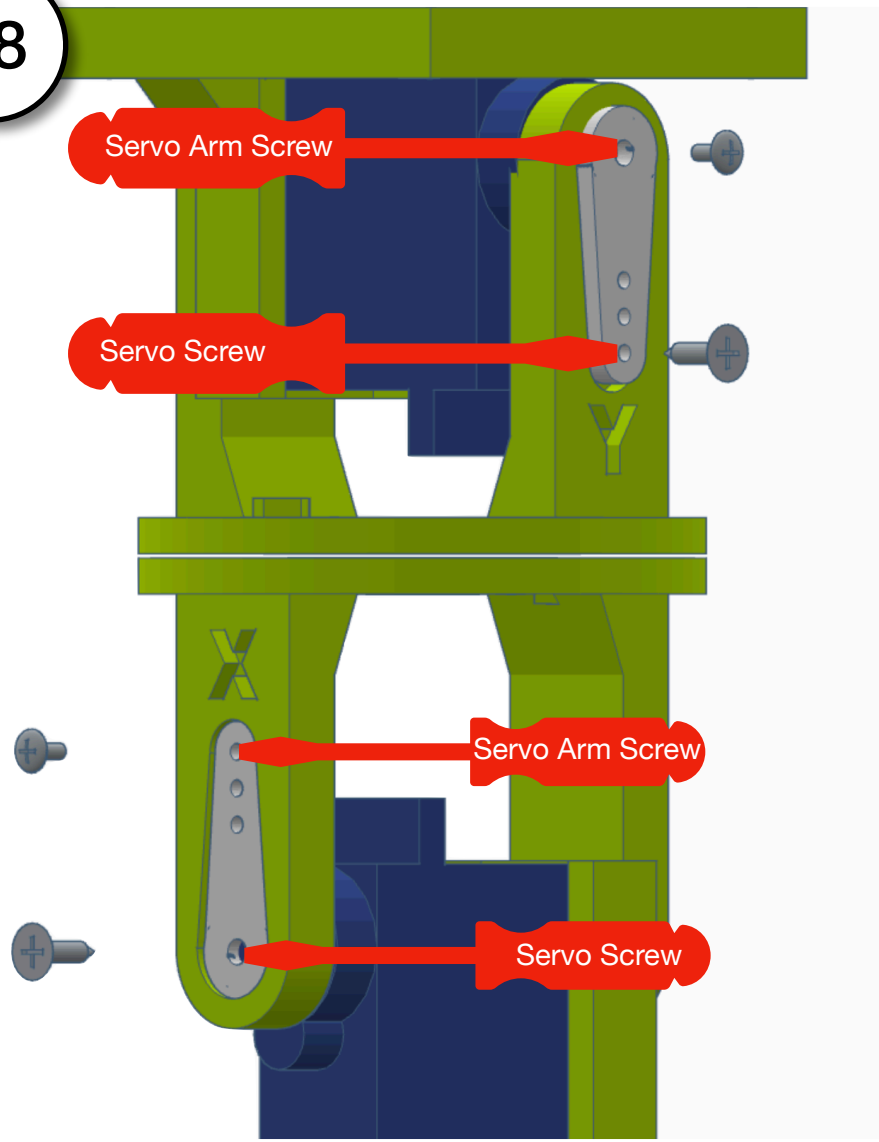
26


X- Servo
Y

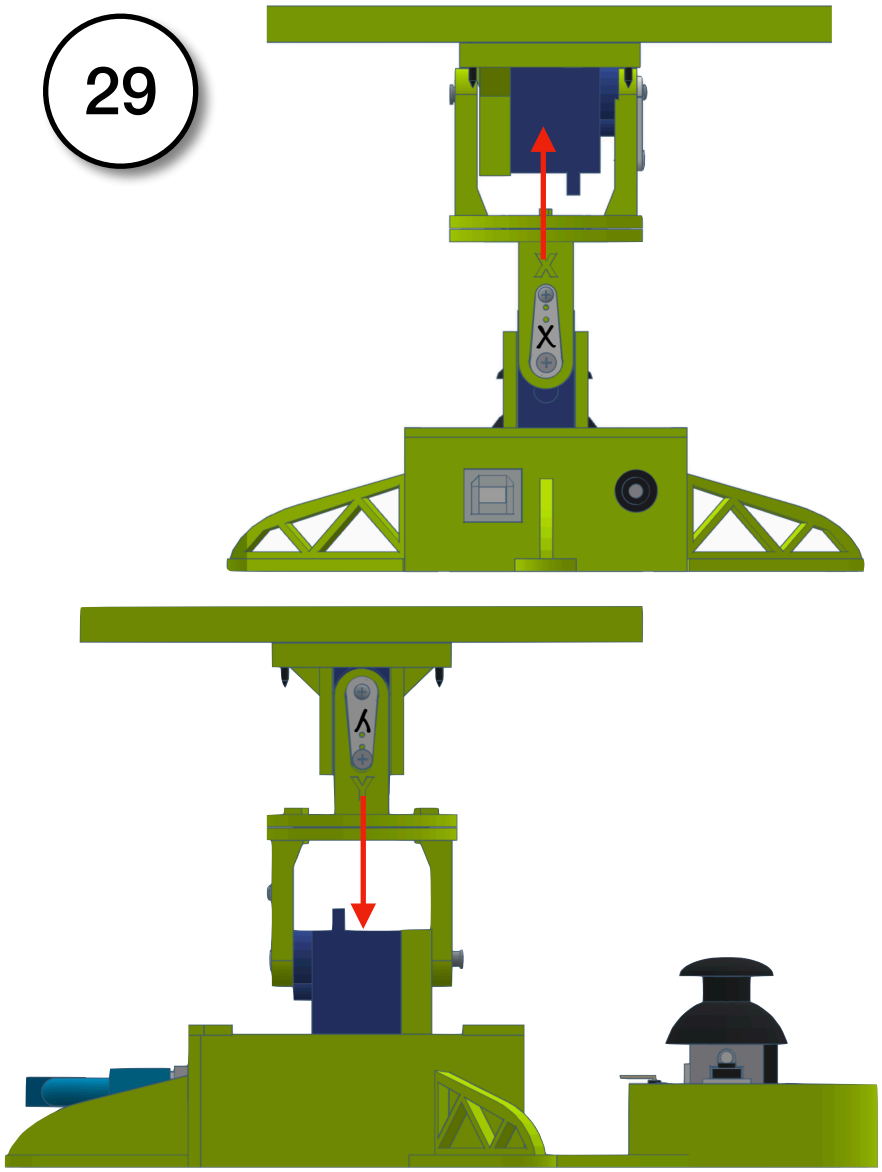
27


X

28


Servo Arm Screw
Servo Screw
Y
X
Servo Arm Screw
Servo Screw

29



You want to make sure your maze is balanced.

Go back to the top section of your code and look again for

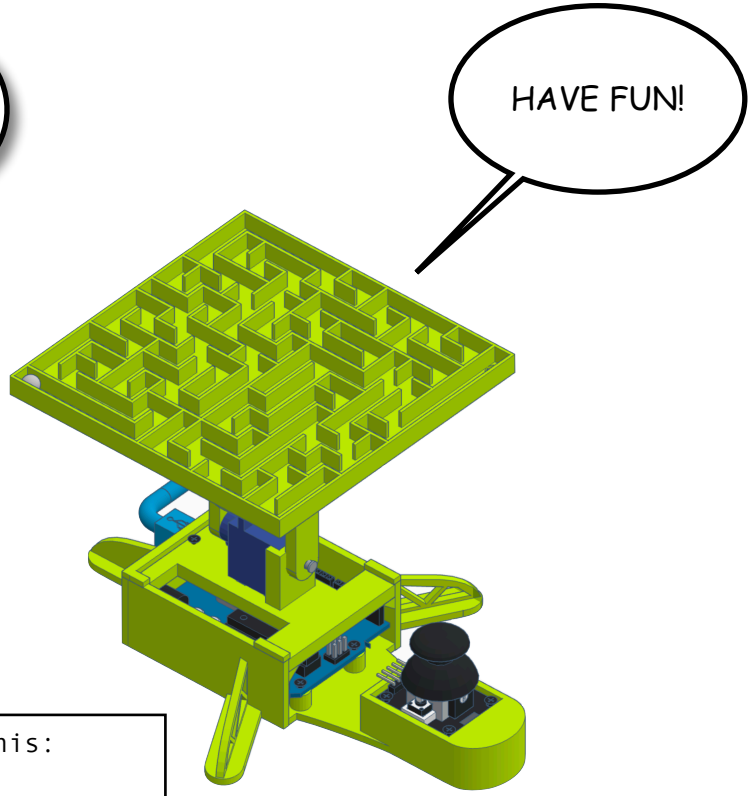
```
int adjustX=0;
int adjustY=0;
```

Go up or down 1 number at a time. adjustX is probably between 20 and 25. adjustY is probably between 4 and 10.

Like this:

```
int adjustX=23;
int adjustY=7;
```

30



You did it! Now sign your code like this:

```
// Coded by John Doe
```

Coding Cheat Sheet

```
#include <Servo.h>

Servo Xservo;
Servo Yservo;

int VRXpin=A0;
int VRYpin=A1;

int XServoPin=9;
int YServoPin=10;

int WVx;
int WVy;

int Xval;
int Yval;

int dt=100;

int Xrange=10;
int Yrange=20;

int adjustX=23;
int adjustY=7;
```

```
void setup() {

pinMode(VRXpin, INPUT);
pinMode(VRYpin, INPUT);

pinMode(XServoPin, OUTPUT);
pinMode(YServoPin, OUTPUT);

Xservo.attach(XServoPin);
Yservo.attach(YServoPin);
}

void loop() {

Xval=analogRead(VRYpin);
WVx=(Xrange/1023.)*Xval*(-1)+Xrange+adjustX;
Yval=analogRead(VRXpin);
WVy=(Yrange/1023.)*Yval+adjustY;

Xservo.write(WVx);
Yservo.write(WVy);

delay(dt);
}

// Coded by John Doe
```