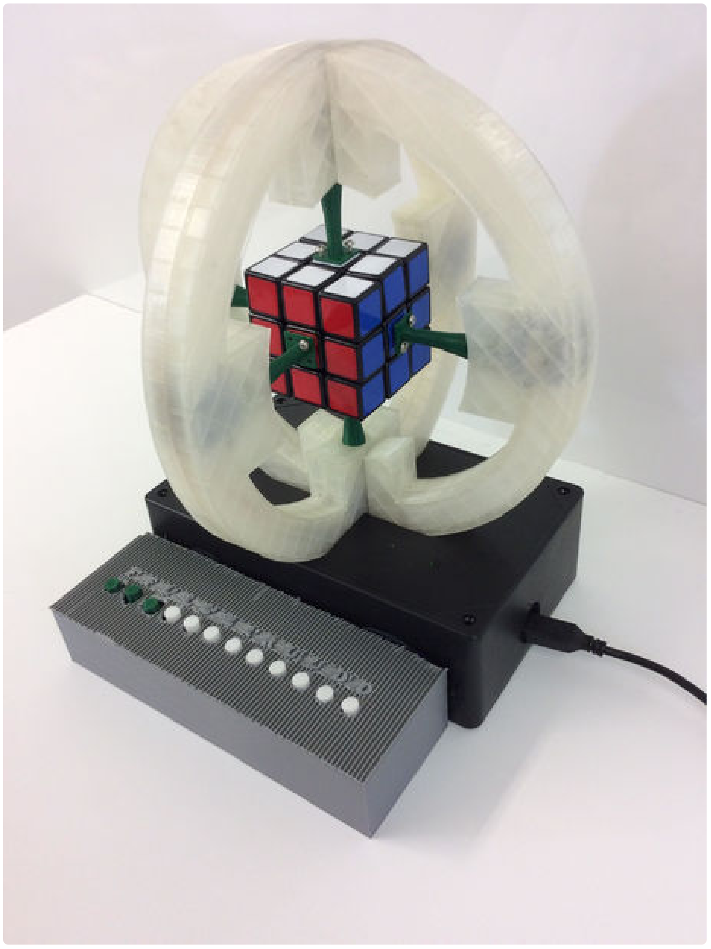
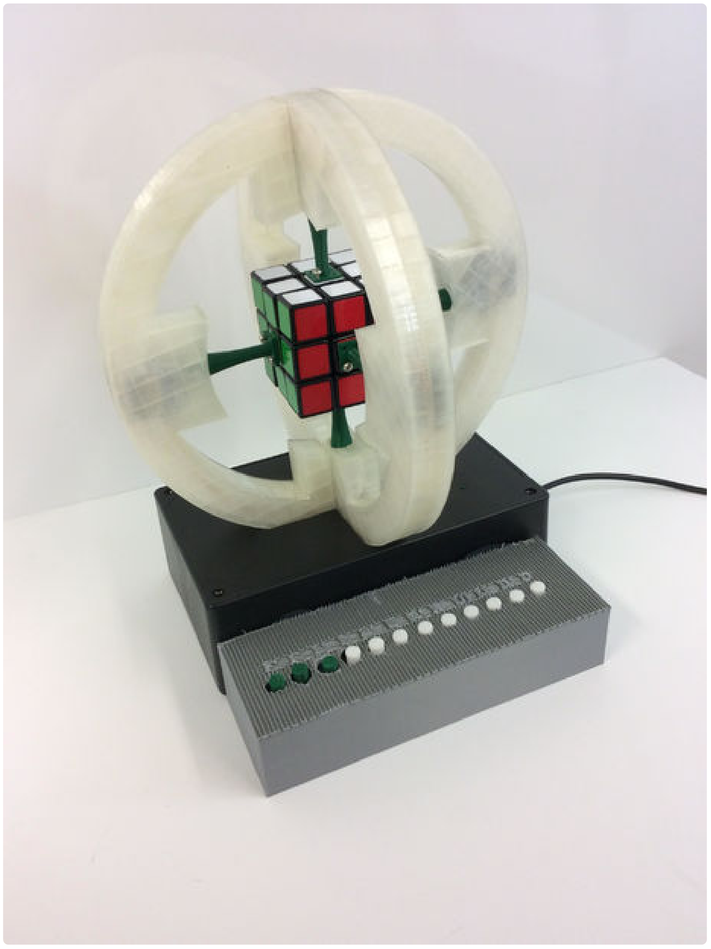
instructables

# The Electronic Rubik's Cube

by RyanZ20

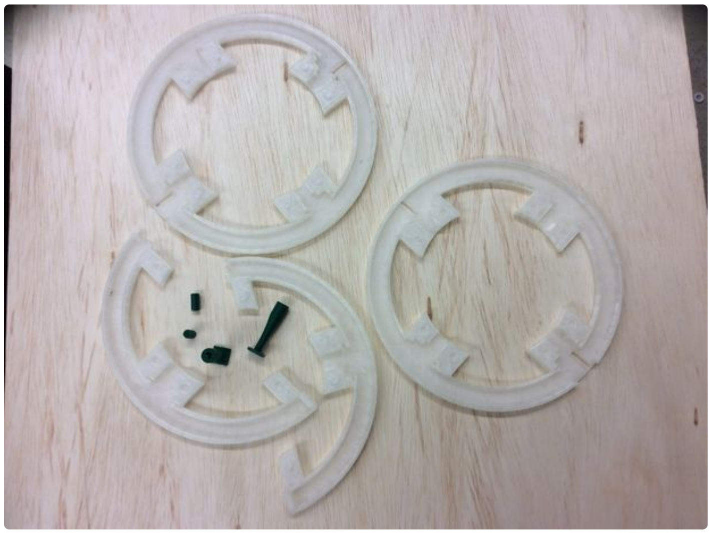
This instructable was created in fulfillment of the project requirement of the Makecourse at the University of South Florida (www.makecourse.com)

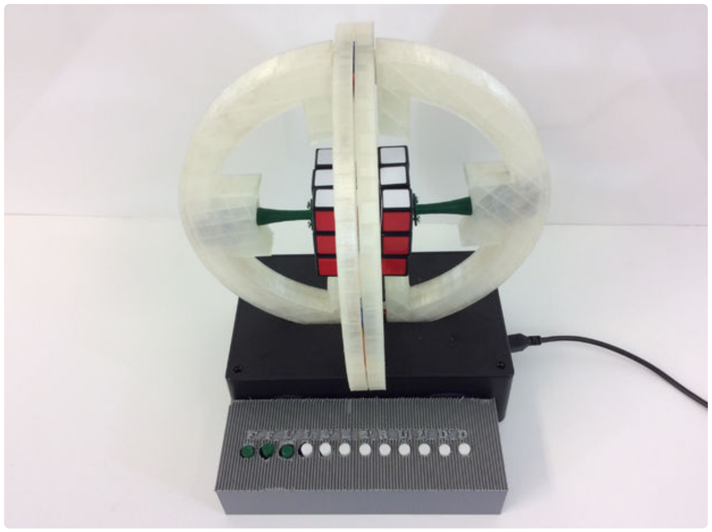
This is an electronic button controlled Rubik's cube. Keep reading to learn how to build your own!

## Step 1: Design and Make the Hardware

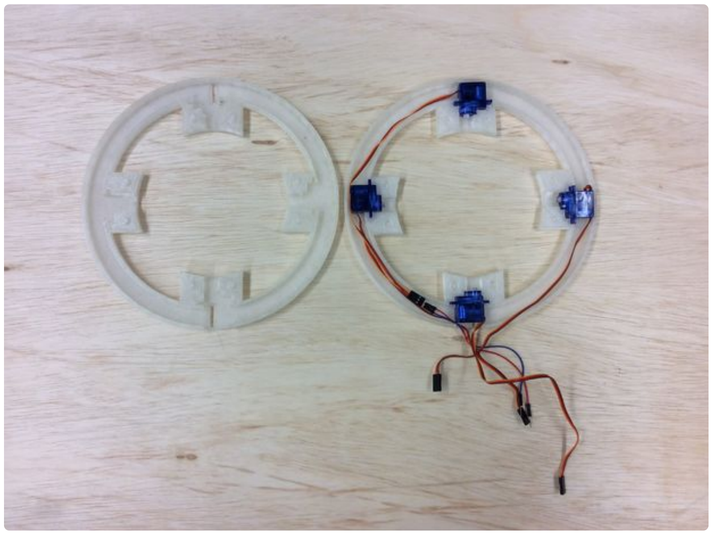
The first step in creating an electronic Rubik's cube is to design the hardware. This can be done through either 3D printing, laser cutting, or many other manufacturing methods. The main criteria for the hardware is that the center tile of each face must have a way to be controlled. I used 6 extensions that were connected to motors using a male/female 90 degree notch. These extensions were then screwed directly into the Rubik's cube tile. Another design criteria is that there must be a way for the electronics to be connected from the motors to the Arduino, and from the Arduino to the buttons. I did this by designing a spherical type structure so that the servos can be held on six sides while still giving a good view of the cube. This was all done by creating models in solidworks and then 3D printing them.

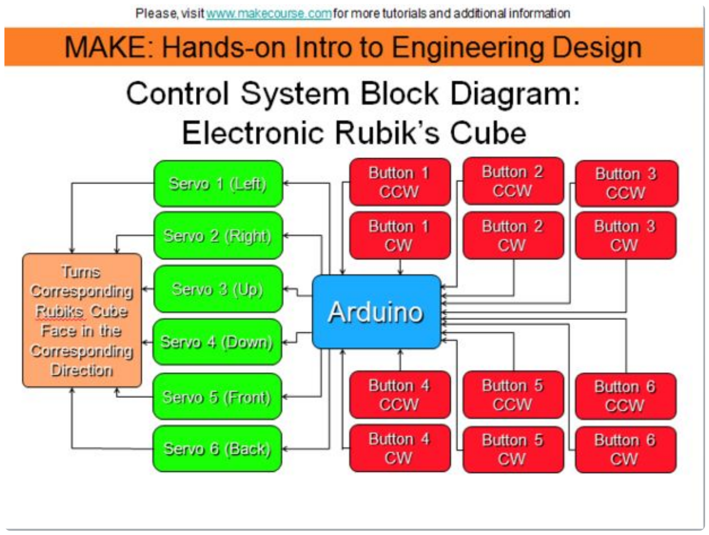




## Step 2: Wire Up the Servo Motors

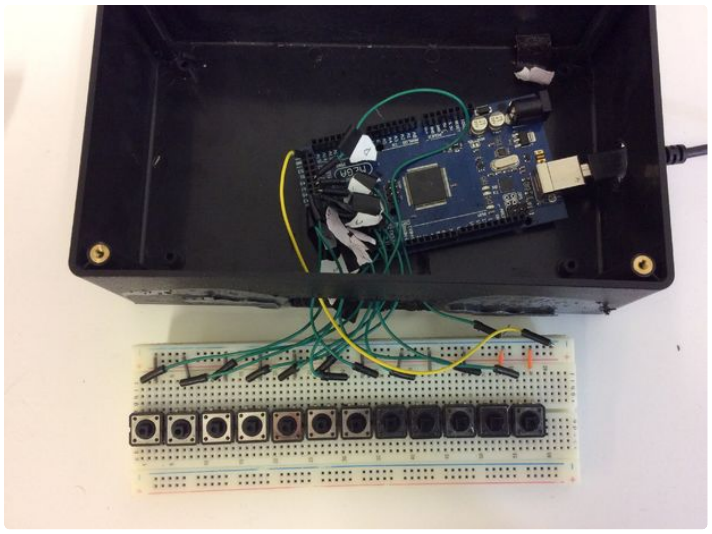
Once the hardware and support structure is created, the next step is to wire the motors to an Arduino Mega. The servos were placed into the support structure and then all the servo wires were fed to a hole at the bottom of the 3D printed structure. This went into a box where the Arduino was housed. The Arduino supplied 5V and ground to a breadboard power strip and then the power and ground wires of the servos were connected to this strip. The data wires of the servos were then plugged directly into the digital ports of the Arduino.

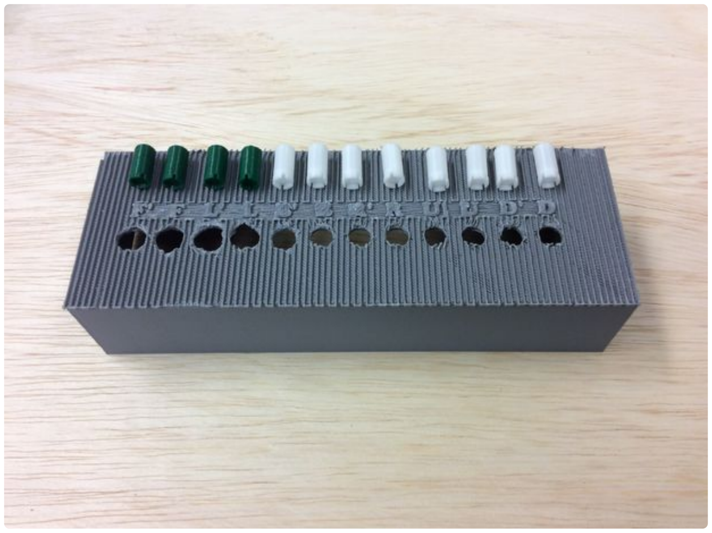


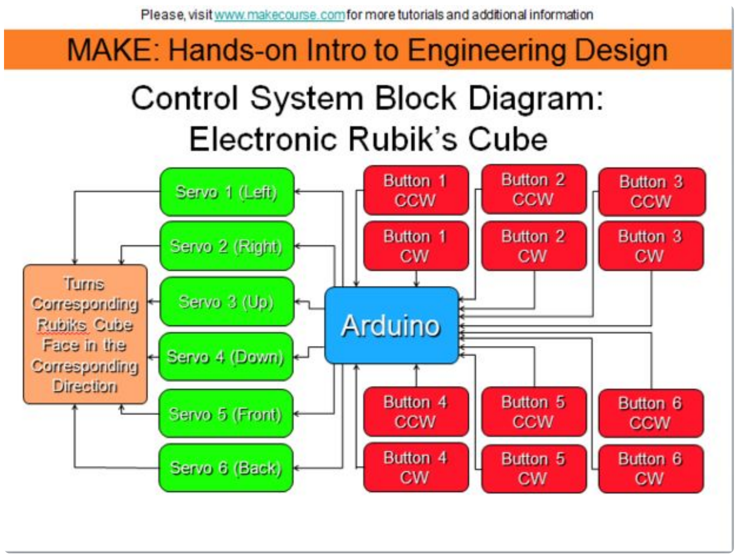


## Step 3: Wire Up the Control Panel

The next step was to create a control panel for the cube.This was done by placing 12 buttons on a large bread board. The Arduino supplied ground to the breadboard and each of the buttons were connected to this power rail. Then the power side of the buttons were wired to digital pins on the Arduino. This circuitry was then placed inside a 3D printed box that labeled each button. I also printed some button extensions. I did this so that the breadboard was hidden and so the buttons were more presentable.

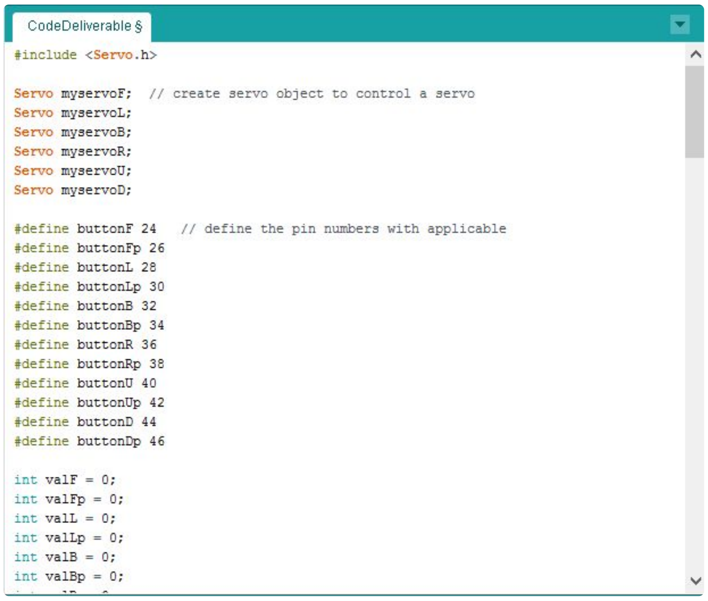




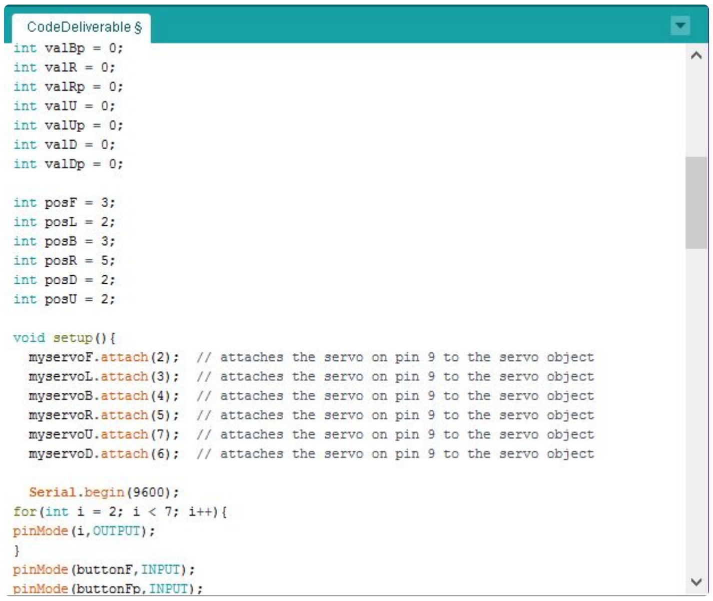

Please, visit www.makecourse.com for more tutorials and additional information
MAKE: Hands-on Intro to Engineering Design
Control System Block Diagram: Electronic Rubik's Cube
Servo 1 (Left)
Servo 2 (Right)
Servo 3 (Up)
Servo 4 (Down)
Servo 5 (Front)
Servo 6 (Back)
Turns Corresponding Rubiks Cube Face in the Corresponding Direction
Arduino
Button 1 CCW
Button 2 CCW
Button 3 CCW
Button 1 CW
Button 2 CW
Button 3 CW
Button 4 CCW
Button 5 CCW
Button 6 CCW
Button 4 CW
Button 5 CW
Button 6 CW

## Step 4: Code Setup

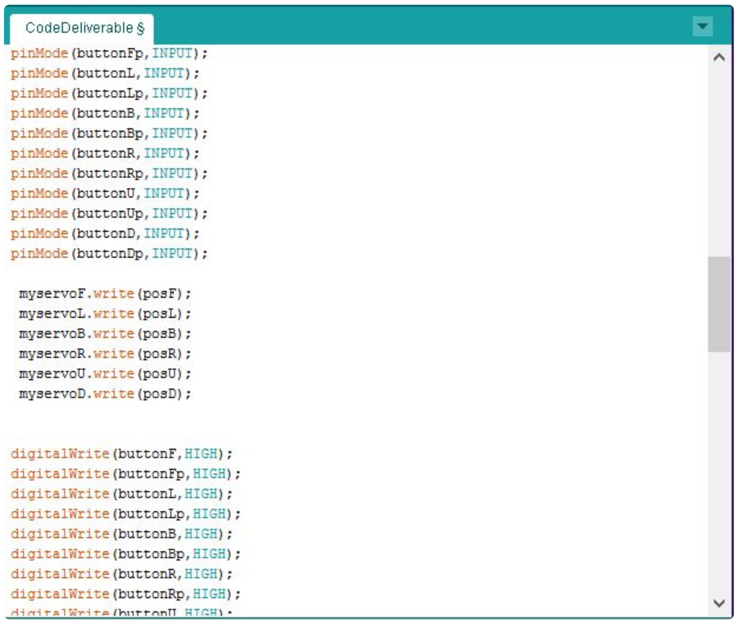
Once all the wiring is complete, it is time to write the code. The first part of the code is to include a servo library. Then you have to create six servo objects, each corresponding to an individual servo. Then the pin numbers where the buttons are wired to are defined as corresponding button names. Next you have to create integers for reading the buttons and for the initial positions of the servos. Once all this is done, you have to attach each of the servo objects to the desired digital pin. Then you have to type "Serial.begin(9600)". The next step is to define all of the used pins as either INPUT or OUTPUT. At this point, you should also write all of the servos to their initial positions and write all of the button pins to HIGH. This concluded the setup of the code.



```
#include <Servo.h>

Servo myservoF;  // create servo object to control a servo
Servo myservoL;
Servo myservoB;
Servo myservoR;
Servo myservoU;
Servo myservoD;

#define buttonF 24   // define the pin numbers with applicable
#define buttonFp 26
#define buttonL 28
#define buttonLp 30
#define buttonB 32
#define buttonBp 34
#define buttonR 36
#define buttonRp 38
#define buttonU 40
#define buttonUp 42
#define buttonD 44
#define buttonDp 46

int valF = 0;
int valFp = 0;
int valL = 0;
int valLp = 0;
int valB = 0;
int valBp = 0;
```



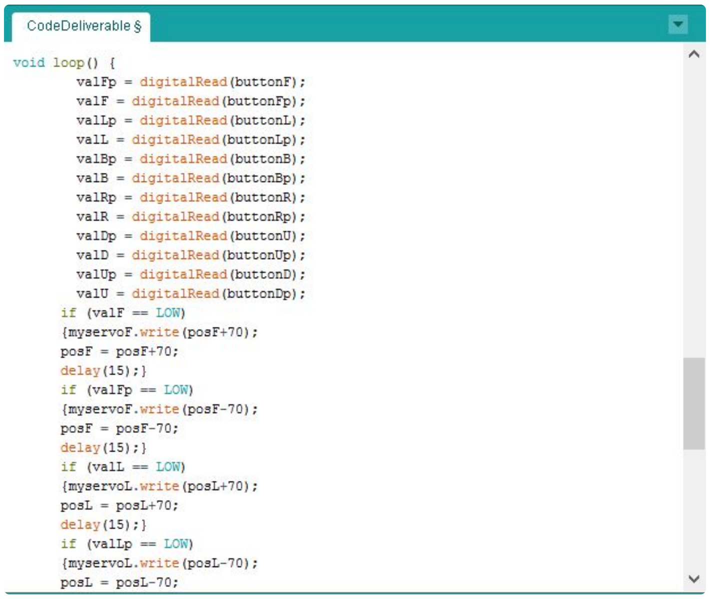
```
int valBp = 0;
int valR = 0;
int valRp = 0;
int valU = 0;
int valUp = 0;
int valD = 0;
int valDp = 0;

int posF = 3;
int posL = 2;
int posB = 3;
int posR = 5;
int posD = 2;
int posU = 2;

void setup(){
  myservoF.attach(2);  // attaches the servo on pin 9 to the servo object
  myservoL.attach(3);  // attaches the servo on pin 9 to the servo object
  myservoB.attach(4);  // attaches the servo on pin 9 to the servo object
  myservoR.attach(5);  // attaches the servo on pin 9 to the servo object
  myservoU.attach(7);  // attaches the servo on pin 9 to the servo object
  myservoD.attach(6);  // attaches the servo on pin 9 to the servo object

  Serial.begin(9600);
for(int i = 2; i < 7; i++){
pinMode(i,OUTPUT);
}
pinMode(buttonF,INPUT);
pinMode(buttonFp,INPUT);
```



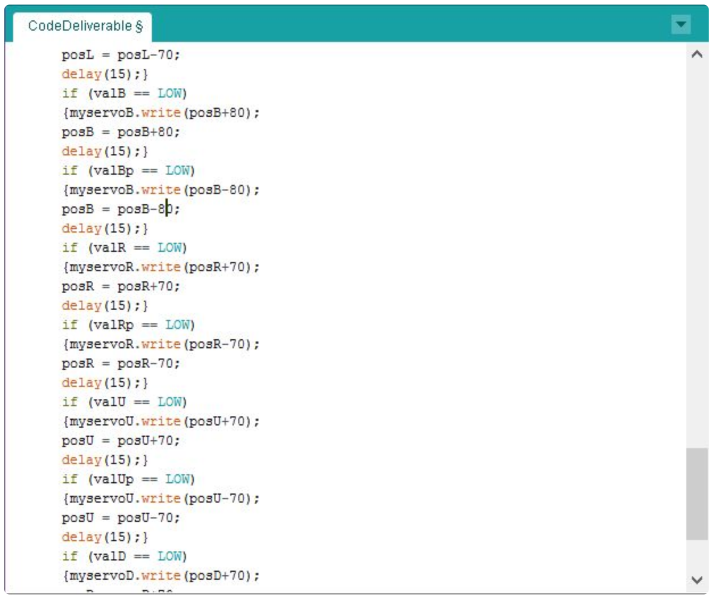
```
pinMode(buttonFp,INPUT);
pinMode(buttonL,INPUT);
pinMode(buttonLp,INPUT);
pinMode(buttonB,INPUT);
pinMode(buttonBp,INPUT);
pinMode(buttonR,INPUT);
pinMode(buttonRp,INPUT);
pinMode(buttonU,INPUT);
pinMode(buttonUp,INPUT);
pinMode(buttonD,INPUT);
pinMode(buttonDp,INPUT);

 myservoF.write(posF);
 myservoL.write(posL);
 myservoB.write(posB);
 myservoR.write(posR);
 myservoU.write(posU);
 myservoD.write(posD);


digitalWrite(buttonF,HIGH);
digitalWrite(buttonFp,HIGH);
digitalWrite(buttonL,HIGH);
digitalWrite(buttonLp,HIGH);
digitalWrite(buttonB,HIGH);
digitalWrite(buttonBp,HIGH);
digitalWrite(buttonR,HIGH);
digitalWrite(buttonRp,HIGH);
digitalWrite(buttonU,HIGH);
```

## Step 5: Code Loop

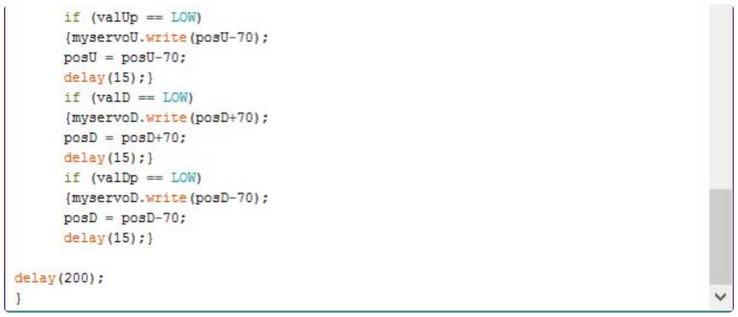
The loop is the part of the code that continuously runs while the Arduino is powered on. The first step of the loop is to read all of the button pins and assign that value to the button integers defined in the previous step. The rest of the code is comprised of if statements. These statements compare all of the button integers to LOW. If this condition is met, the servo is turned a desired amount depending on which button was pressed. The initial position of the servo must then be updated to be equal to the increase made by the button press. The only thing left to do with the code is to put a 200 ms delay on the loop so that the loop doesn't read a button press more than one time.



```
void loop() {
        valFp = digitalRead(buttonF);
        valF = digitalRead(buttonFp);
        valLp = digitalRead(buttonL);
        valL = digitalRead(buttonLp);
        valBp = digitalRead(buttonB);
        valB = digitalRead(buttonBp);
        valRp = digitalRead(buttonR);
        valR = digitalRead(buttonRp);
        valDp = digitalRead(buttonU);
        valD = digitalRead(buttonUp);
        valUp = digitalRead(buttonD);
        valU = digitalRead(buttonDp);
      if (valF == LOW)
      {myservoF.write(posF+70);
      posF = posF+70;
      delay(15);}
      if (valFp == LOW)
      {myservoF.write(posF-70);
      posF = posF-70;
      delay(15);}
      if (valL == LOW)
      {myservoL.write(posL+70);
      posL = posL+70;
      delay(15);}
      if (valLp == LOW)
      {myservoL.write(posL-70);
      posL = posL-70;
```



```
      posL = posL-70;
      delay(15);}
      if (valB == LOW)
      {myservoB.write(posB+80);
      posB = posB+80;
      delay(15);}
      if (valBp == LOW)
      {myservoB.write(posB-80);
      posB = posB-80;
      delay(15);}
      if (valR == LOW)
      {myservoR.write(posR+70);
      posR = posR+70;
      delay(15);}
      if (valRp == LOW)
      {myservoR.write(posR-70);
      posR = posR-70;
      delay(15);}
      if (valU == LOW)
      {myservoU.write(posU+70);
      posU = posU+70;
      delay(15);}
      if (valUp == LOW)
      {myservoU.write(posU-70);
      posU = posU-70;
      delay(15);}
      if (valD == LOW)
      {myservoD.write(posD+70);
```



```
      if (valUp == LOW)
      {myservoU.write(posU-70);
      posU = posU-70;
      delay(15);}
      if (valD == LOW)
      {myservoD.write(posD+70);
      posD = posD+70;
      delay(15);}
      if (valDp == LOW)
      {myservoD.write(posD-70);
      posD = posD-70;
      delay(15);}

delay(200);
}
```

## Step 6: Enjoy!

This is everything required to create an awesome electronics Rubik’s. Good luck and most importantly, have fun!

Download (https://cdn.instructables.com/ORIG/FZA/XEP0/J2AVST2P/FZAXEP0J2AVST2P.mp4)

**http://www.instructable…**

(https://cdn.instructables.com/ORIG/FZA/XEP0/J2AVST2P/FZAXEP0J2AVST2P.mp4)