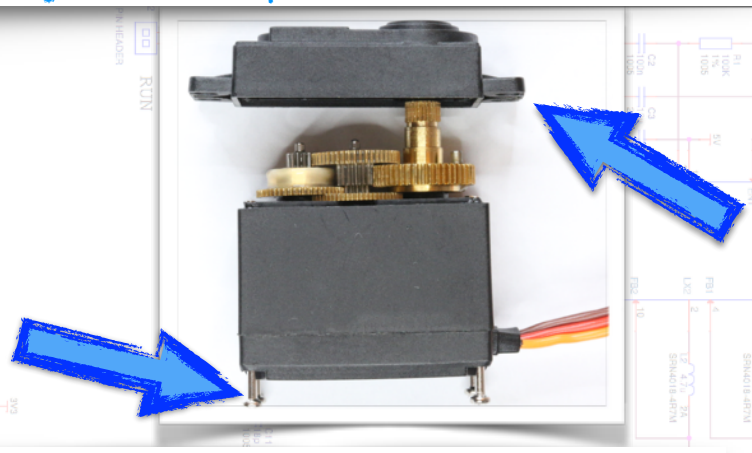
Closed Loop Control - Servo Take Apart

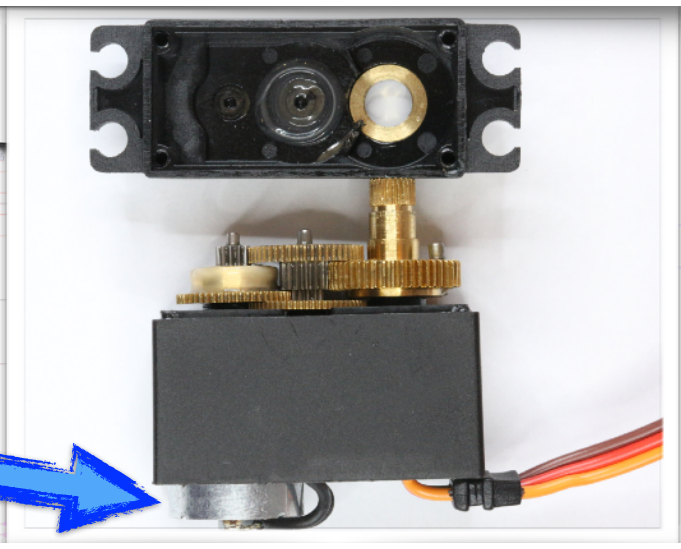
Servos take an input signal like "go to 38 degrees" and move to that position and actively hold that position. You can read more about this signal by googling "PWM servo control". For now just keep in mind that we ask the servo to move to a certain position and hold that position. The question is HOW does it know where it is at any moment? This is idea of knowing where you are in space is an example of Closed Loop Control. We will find a potentiometer (dimmer switch) inside the servo that is attached to the output shaft. The rotational position of that potentiometer is our position signal. Follow the steps below after you have already made a servo work with an Arduino, Microbit (or other micro-controller). Once you have the servo apart hook it back up and run programs for 0°-180° (sweeping), 90° (holding) and 180° (holding). With each program SLOWLY rotate the potentiometer back and fourth and observe the results. Can you explain the behavior of the motor? If you are really confident can you make predictions of this behavior in advance for each of the above cases?

#1



Remove 4 screws, then the top cover.

#2

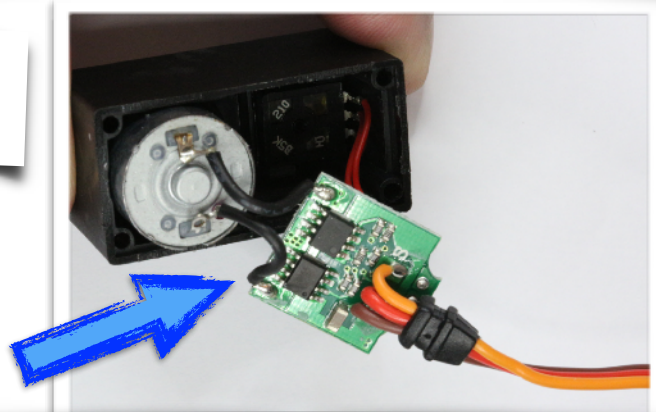


Remove the bottom cover. Locate the motor.

#3

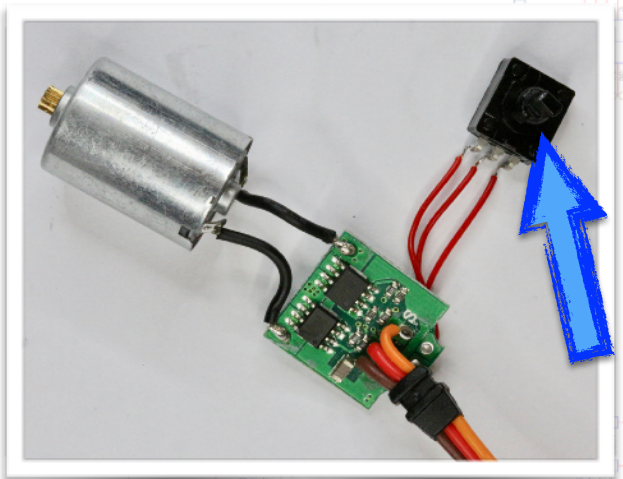
Bonus question: Can you approximate the gear ratio?

#4



Gently remove the circuit board to reveal the potentiometer underneath

#5



Remove the motor, control board and servo still intact with soldered wires. Hook the servo back up and run the program. Rotate the potentiometer SLOWLY in both directions. Observe the behavior of the closed loop when it is now an open loop!

#6

sketch_mar31a §

```
#include <Servo.h>

Servo motor;

void setup() {
motor.attach(11);
}

void loop() {
motor.write(0);
delay(1000);
motor.write(180);
delay(1000);
}
```

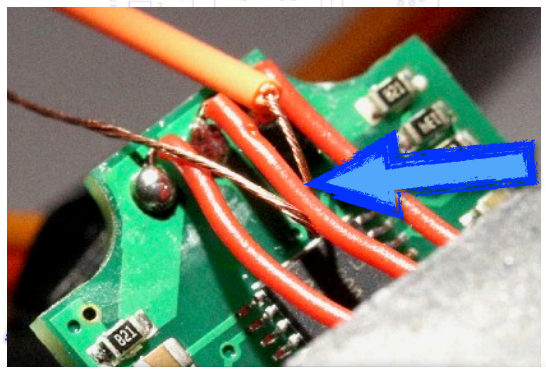
Try different values and observe the effect on the motor.

The servo as an output

The next step after taking the servo apart is to do a hardware hack on it as before you reassemble. We are going to tap the center wire on the potentiometer (POT). The center tap reads the changing value of the potentiometer. Strip a long section of jumper wire (about 1.5") and follow the directions below to tap into the sweeper. It is not necessary to solder the wire to test this out.

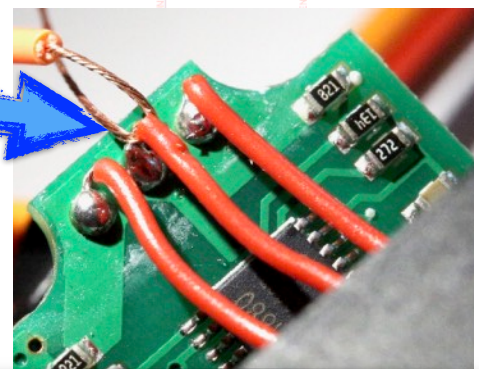
We will use this signal as an analog input in a program that first listens for a signal and records it. We will then use that signal as the base to generate a PWM signal back to the servo with that same pattern.

#1



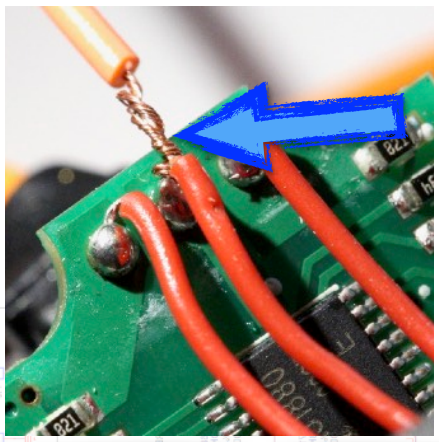
Loop a section of jumper hooks around the center tap on the POT

#2



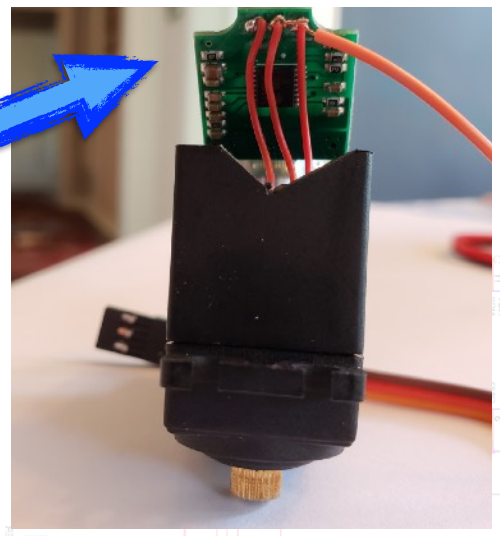
Loop the wire around it's self at the solder joint

#3



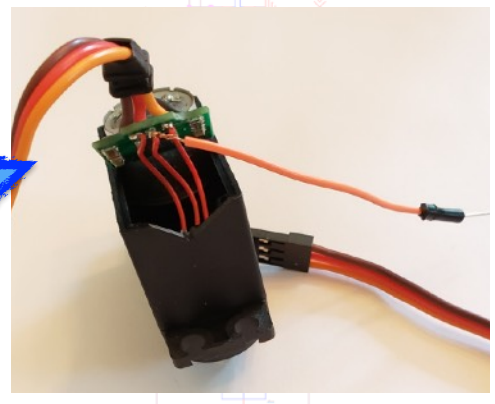
Make three loops total and trim the end of the wire.

#4



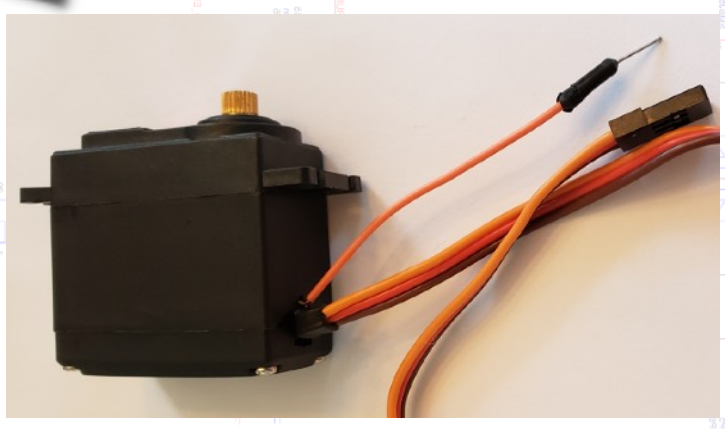
Cut a notch in the side of the servo.

#5



Take great care in closing the servo case. Be sure to not add any strain to the existing solder joint on the POT.

#6



The servo with the POT tapped goes to the analog input A1.

The Arduino Sketch

```
/*based on the work by peterbiglab 2020*/

#include <Servo.h>

static const unsigned int arrayDimension = 500;
int recordingPin = A1;
unsigned int recordedPositions[500];

Servo yourServo;
int servoPin = 7;
int deg0PotentiometerValue = 0;
int deg180PotentiometerValue = 0;

void setup() {
  Serial.begin(115200);
  Serial.println("Getting 0 and 180 degrees values from the potentiometer inside the servo...");
  yourServo.attach(servoPin);
  yourServo.write(0);
  delay(2000);
  deg0PotentiometerValue = analogRead(recordingPin);
  delay(50);
  Serial.print("Got ");
  Serial.print(deg0PotentiometerValue);
  Serial.println(" for 0 degrees.");
  yourServo.write(180);
  delay(2000);
  deg180PotentiometerValue = analogRead(recordingPin);
  delay(50);
  Serial.print("Got ");
  Serial.print(deg180PotentiometerValue);
  Serial.println(" for 0 degrees.");
  Serial.println("Going to 90 degrees...");
  yourServo.write(90);
  delay(2000);
  yourServo.detach();
}

void loop() {
  Serial.println(F("----------"));
  Serial.println(F("Give me a resolution (how many frames do you want to record / second). Then press enter to start recording."));
  unsigned int resolution = 1000 / getDataFromSerial().toInt();
  Serial.println(F("Recording started..."));
  unsigned long timeBeforeRecording = millis();
  record(recordedPositions, arrayDimension, resolution, recordingPin);
  unsigned long recordingTime = millis() - timeBeforeRecording;
  Serial.print(F("Recording ended. Recording time is "));
  Serial.print(recordingTime);
  Serial.println(F(" ms."));
  Serial.println(F("Type any key (and enter) to replay and to start a new recording session."));
  getDataFromSerial();
  Serial.println(F("Replaying started..."));
  yourServo.attach(servoPin);
  replay(recordedPositions, arrayDimension, resolution, yourServo);
}

void record(int *recordingArray, int arrayDimension, int recordingResolution, int recordingPin) {
  for (int i = 0; i < arrayDimension; i++) {
    recordingArray[i] = map(analogRead(recordingPin), deg0PotentiometerValue, deg180PotentiometerValue, 0, 180);
    delay(recordingResolution);
  }
}

void replay(int *recordingArray, int arrayDimension, int recordingResolution, Servo servoToMove) {
  for (int i = 0; i < arrayDimension; i++) {
    servoToMove.write(recordingArray[i]);
    delay(recordingResolution);
  }

  servoToMove.detach();
}

String getDataFromSerial() {
  waitForDataFromSerial(false);
  String toReturn = Serial.readString();
  Serial.flush();
  return toReturn;
}

void waitForDataFromSerial(boolean doFlush) {
  while (!Serial.available()) {
    //Wait for user input.
  }

  if (doFlush) {
    Serial.flush();
  }
}
```