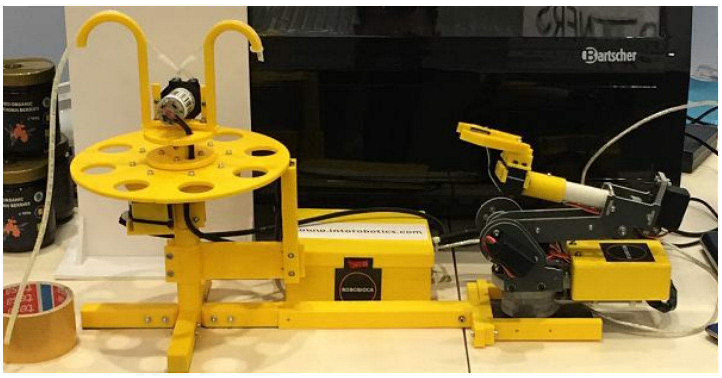
· March 1, 2019 · 4 comments

How to Control Servo Motors With Arduino (no noise, no vibration)

In this tutorial, I provide enough level of depth to be informed about how to control servo motors with Arduino. I use PWM software and hardware resources because this is the right way to control one or more servo motors. After reading this article, you should be able to control your hobby servos without noise or vibration. In addition, I write a section at the end of the article where I describe how I fix several MG996R servos and a conclusion.

Before going into the details, I want to tell you something. I know how hard is to produce great work and then put it out into the world. But no matter how much was written on forums and blogs about how to control hobby servo motors like MG996R, almost all these tutorials provide information to control it while working freely, without applying torque. The servo changes its behavior under applying torque. If it is not controlled in the right way, the servos vibrate, make noise and rotate randomly. That is how I came to write this tutorial. I want to describe you in details the solutions tried to control the servos, as well as the solution to smoothly control many hobby servos like MG996R while applying torque – in my case, the servo actuates a robotic arm.

The story:


RoboBioca Robot Arm

The MG996R servo is the main servo motor used to move the SainSmart 6-axis robot arm. From all of the 6 axes of the arm, four of them are actuated by the MG996R. Only two axes use the SG90 servo to move the end effector.

I used this robot arm to build a robot waiter able to grab and handle a small plastic cup. Because I need something different for my project, I cannot use the default configuration of the arm. I replaced the SG90 servo motors with only one servo motor with metallic gearbox and claws. In this way, I reduced the number of degrees of freedom of the arm from 6 to 5.

In this project, I used the Blynk application to send commands to the robot. The communication between the Android tablet running Blynk and the robot is done wirelessly using a Bluetooth connection.

And because I would not have had enough problems to solve, the Bluetooth connection interferes with the servo motors. These are making noise and running randomly while is working the Bluetooth connection. So an extra problem to be solved.

The power supply

A stable power supply of 6V as required for powering the servo motors is mandatory. I used an

TWITTER UPDATE

Dragos Calin 21 May 2020

New Project: How To Build a DIY Robot Chassis: http://bit.ly/2TmOFMW #Robotics

LATEST POSTS

How To Build a DIY Robot Chassis May 20, 2020

LIDAR sensors for Robots May 8, 2020

How to Use sensor_msgs/Range (ROS) For Multiple Sensors with rosserial May 5, 2020

Getting started with the Garmin (Qwiic) LIDAR-Lite v4 LED May 1, 2020

How To Build an Obstacle Avoiding Robot April 25, 2020

SUPPORT THIS BLOG

Support this blog by purchasing from Amazon through this link

adjustable LTC3780 DC buck/boost converter module. Also, I have a power adapter (12V – 3A) to feed the converter. I used a digital multimeter to set an output voltage as close as 6V for the converter.

First try

The first attempt was to control the servo motors with the Arduino Sensor Shield V5. The biggest problem with Arduino UNO is that I have only two PWM pins (pin 9 and pin 10) that can be used with PWM while running the Servo2 library. The PWM pins are used for the control signal of servo motors. Unlike DC motors, the PWM control is required for servos to determine the position rather than the speed of the servo shaft.

> *This is the Servo library distributed with Arduino 0016 and earlier. It can drive up to two servos using pins 9 and 10 on a standard board or 11 and 12 on a Mega. Other pins wont work.*

I had two servo motors working well when I control them with PWM and that's all. I leave them and make another try.

Second try

With five servos and only two pins with PWM that I can use, I tried another way to control the servo motors. I gave up the Servo2 library and I use capacitors and the Servo library. This attempt turned out more successful than the previous one. From the solutions found on the Internet, I used 470uF 50V electrolytic capacitor for each servo. It works, but not as I want it. Shaking and vibrations don't occur in most of the servo positions, but the random rotations still continued.

Third try

The third attempt was successful. I used the Adafruit 16-Channel PWM/Servo driver to control all the five servos. The driver uses an i2c-controlled PWM driver with a built-in clock and 12-bit resolution for each servo, that means about 4us resolution at 60Hz update rate.

I used the Adafruit_PWMServoDriver library and the inspiration to write the code was coming from examples provides by the library.

This is a demo with the robot arm controlled with PWM using the Adafruit's driver.

```cpp
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();


#define MIN_PULSE_WIDTH 135
#define MAX_PULSE_WIDTH 470
#define DEFAULT_PULSE_WIDTH 150
#define FREQUENCY 50


const uint8_t servoA = 0;
const uint8_t servoB = 1;
const uint8_t servoC = 2;
const uint8_t servoD = 3;
const uint8_t servoE = 4;

const uint8_t delayServo = 25;
```

```


void controlServo0(uint8_t oldPos, uint8_t newPos)
{
  if (oldPos <= newPos) switchServo = UP; else switchServo = DOWN;

  switch (switchServo) {
    case UP: {
        for (uint8_t pos = oldPos; pos <= newPos; pos += 1) {
          delay(delayServo);
          pwm.setPWM(0, 0, angleToPulse(pos));
        }
        break;
      }
    case DOWN: {
        for (uint8_t pos = oldPos; pos >= newPos; pos -= 1) {
          delay(delayServo);
          pwm.setPWM(0, 0, angleToPulse(pos));
        }
        break;
      }
  }
}


void controlServo1(uint8_t oldPos, uint8_t newPos)
{
  if (oldPos <= newPos) switchServo = UP; else switchServo = DOWN;

  switch (switchServo) {
    case UP: {
        for (uint8_t pos = oldPos; pos <= newPos; pos += 1) {
          delay(delayServo);
          pwm.setPWM(1, 0, angleToPulse(pos));
        }
        break;
      }
    case DOWN: {
        for (uint8_t pos = oldPos; pos >= newPos; pos -= 1) {
          delay(delayServo);
          pwm.setPWM(1, 0, angleToPulse(pos));
        }
        break;
      }
  }
}


void controlServo2(uint8_t oldPos, uint8_t newPos)
{
  if (oldPos <= newPos) switchServo = UP; else switchServo = DOWN;

  switch (switchServo) {
    case UP: {
        for (uint8_t pos = oldPos; pos <= newPos; pos += 1) {
          delay(delayServo);
          pwm.setPWM(2, 0, angleToPulse(pos));
        }
        break;
      }
    case DOWN: {
        for (uint8_t pos = oldPos; pos >= newPos; pos -= 1) {
          delay(delayServo);
          pwm.setPWM(2, 0, angleToPulse(pos));
        }
        break;
      }
  }
}

void controlServo3(uint8_t oldPos, uint8_t newPos)
{
  if (oldPos <= newPos) switchServo = UP; else switchServo = DOWN;

  switch (switchServo) {
    case UP: {
        for (uint8_t pos = oldPos; pos <= newPos; pos += 1) {
          delay(delayServo);
          pwm.setPWM(3, 0, angleToPulse(pos));
        }
        break;
      }
    case DOWN: {
        for (uint8_t pos = oldPos; pos >= newPos; pos -= 1) {
          delay(delayServo);
          pwm.setPWM(3, 0, angleToPulse(pos));
        }
        break;
      }
  }
}

void controlServo4(uint8_t oldPos, uint8_t newPos)
{
  if (oldPos <= newPos) switchServo = UP; else switchServo = DOWN;

  switch (switchServo) {
    case UP: {
        for (uint8_t pos = oldPos; pos <= newPos; pos += 1) {
          delay(delayServo);
          pwm.setPWM(4, 0, angleToPulse(pos));
        }
        break;
      }
    case DOWN: {
        for (uint8_t pos = oldPos; pos >= newPos; pos -= 1) {
          delay(delayServo);
          pwm.setPWM(4, 0, angleToPulse(pos));
        }
        break;
      }
  }
}

void robotServoExample() {
  controlServo0(140, 140);
  controlServo1(145, 145);
  controlServo3(85, 85);
  controlServo4(155, 155);
}

void setup() {
  pwm.begin();
  pwm.setPWMFreq(FREQUENCY);
}
```

```
void loop() {
    robotServoExample();
}
```

The story with the MG996R servo

The first failed servomotor was the one who turned the robot arm. It has failed quite easily – I have noticed some smoke being emitted during operation. The second failed servo was at the base. This one no longer recognized the position. Regardless of the commands for a particular position, it randomly changes its position. The cause was not using the Bluetooth module. I tried to position the servo motor without attaching the communication module.

The first servo failed after about 30 minutes of operation. The second was more friendly and failed after about another 2 hours of operation. During this time I've used the robot arm to test different positions.

I'm in the situation of having two failed MG996R servo motors while I start working to solve two problems that have actually increased my blood pressure: random shakes(vibrations) and noise.

Initially, I thought I do something wrong and that's why the servo motors were broken so fast. After a search on Google, it looks like I'm not the only one who worked with the MG996R and have broken them in the testing phase. Finally, I ordered some new servos and I start working to see where I was wrong and how to fix this.

I ordered three MG996R servo motors. They came quickly and I started to test them. It's not the first time when I bought something very cheap (the price for an MG996R is around €6/$7) and did not work. This time was something very special for me. From three servo motors, all of them did not work.

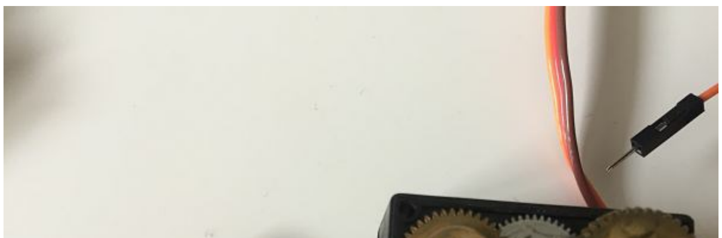
With three new servo motors that did not work, I thought it would be better to search for an alternative. I've looked for alternatives having the same size that could fit into the SainSmart robotic arm. I found Power HD Servo 1501MG. They are the same size as the MG996R plus higher torque.

Of the five MG996R servo motors used in the upgraded version of the robot arm (the original version of the arm uses 4 MG996R), there are three left. Two MG996R servos were replaced by 1501MG. One MG996R servo drives the shoulder yaw, another on the wrist yaw, and a third on the robot claw.

How I fix the MG996R

Any of the below methods require training in electronics and some experience with screws, gearboxes, and grease. Take great care when you work inside the servo, you will do it on your own responsibility.

1. Two of the servo motors I've fixed just by removing the screws. I control them to move back and forth from position 30 to 160 without putting the cover of the gearbox, then I fixed the cover of the gearbox… and it works. Sounds a little stupid, but this is what I did and works. I suspect the problem was in the position of the gears inside the gearbox. Most likely, the DC motor was rotating without engaging all of them. There may be assembly errors or transport shocks.
2. One of the new servos had some bigger problems. One gear doesn't have one or two teeths. Besides that, I found a piece of plastic inside the gearbox. Fortunately, I have a burned servo from where I took the spare pieces, I mounted back all the pieces and it worked.



MG996R missing theets

Conclusion on the hobby servos

- try as little as possible to manually change the position of your servo. This can quickly destroy the gearbox and the servo can give fails;
- the servo motors used in hobby projects works well between positions 10 and 170. If you place a position less than 10 or greater than 170 it makes a noise and gets into vibrations;
- the power supply of the servo must be as close as possible to the 6V value. A higher voltage can burn it, and a lower voltage makes it run randomly. Also, please do not try to feed servos directly from the Arduino board or any other controller used in your project. You will have many surprises;
- if you work with multiple servo motors at once, for example for a robot arm, you would want to program them one by one. If one of the servos has an incorrect command and moves to a less desired position, it has a big impact on the other servos. The robot arm can touch the work table or other objects around it and broke the gearbox of the servo;
- if you do not have a stable power supply source of 6V, it is necessary to use an electrolytic capacitor between the power supply wires of the servo motor;

Related Posts:

- Bluetooth Controlled Arduino Robot Arm
- Motor Drivers For 24V Brushed DC Motors
- I Set Up the Sabertooth 2 X 25A Motor Driver To Control 4 DC Motors With Arduino UNO. The Solution for Autonomous Robots.

Posted in: Robots
Tagged: Arduino

4 comments » Write a comment

har 06.04.19 @ 7:01 pm Log in to Reply

where is the tutorial part of this article? you privided ZERO instructions

prasanna 06.07.21 @ 3:58 pm Log in to Reply

no, he gives a lot of valuable pieces of information

Chris 02.07.19 @ 9:14 am Log in to Reply

Good base to elaborate from but I can't go passed the fact that code is unoptimized. It is 5 times too long for what it does…

Calin Dragos-George 02.07.19 @ 9:32 am

Log in to Reply

Hello Cris,

Thank you for your comment. The Arduino sketch is used for five servo motors and I think it's helpful than having the code for only one servo motor.

Leave a Reply

You must be logged in to post a comment.