ABA AIOT PVT LTD
Pratik Rohane

Calibrating Servo Motor With PCA9685

OBJECTIVE

To determine the servo motor PWM pulse signal values for the corresponding angle values.

OVERVIEW

In this test, we are finding the maximum and minimum PWM signal value for the HiTech 755-MG servo motor to its maximum and minimum angle limit. So we can map the servo movement to all other angles in between the maximum and minimum angle limit with the help of maximum and minimum PWM signal value for this servo motor.

REQUIREMENTS

HARDWARE REQUIRED

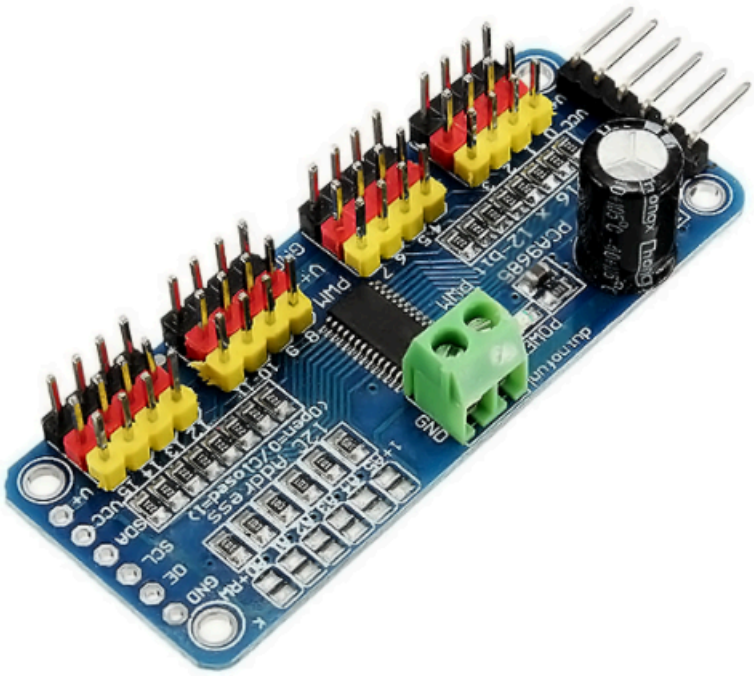
1. Arduino UNO
2. PCA9685 16-Channel 12-bit PWM Servo Driver
3. HiTech 755-MG Servo Motor
4. 6V Battery

SOFTWARE REQUIRED

1. Arduino IDE

COMPONENTS DESCRIPTION AND SPECIFICATION

PCA9685 16-Channel 12-bit PWM Servo Driver



The PCA9685 16-Channel 12-bit PWM Servo Driver will drive up to 16 servo over I2C serial communication protocol, so data is transferred bit by bit along SDA(Serial Data) line. And the clock is synchronized by the SCL(Serial Clock) line.

Pins

GND	Ground Pin
VCC	Power Pin (3-5V)
V+	Power Supply pin for Servos
SCL	Serial Clock
SDA	Serial Data
OE	Output Enable

Ports

16 Output ports each has 3 pins: V+, GND, PWM frequency Pin

HS-755MG Servo-Stock Rotation

Output Shaft Style	C24T Spline
Voltage Range	4.8V - 6.0V
No-Load Speed (4.8V)	0.28sec/60°
No-Load Speed (6.0V)	0.23sec/60°
Stall Torque (4.8V)	167 oz-in (12 kg.cm)
Stall Torque (6.0V)	200 oz-in (14 kg.cm)
Max PWM Signal Range	570-2400µsec
Travel per µs	.110°/µsec
Max Rotation	200°
Pulse Amplitude	3-5V
Operating Temperature	-20°C to +60°C
Current Drain - idle (4.8V)	8mA
Current Drain - idle (6.0V)	8.7mA
Current Drain - no-load (4.8V)	230mA

Current Drain - no-load (6V)	285mA
Continuous Rotation Modifiable	Yes
Direction w/ Increasing PWM Signal	Clockwise
Deadband Width	8µs
Motor Type	3 Pole Ferrite
Feedback Style	5KΩ Potentiometer
Output Shaft Support	1 Bearing and 1 Oilite Bushing
Gear Material	Metal
Wire Length	11.81" (300mm)
Weight	4.12 oz.. (117g)
Wire Gauge	22AWG
Servo Size	Large

PROCEDURE

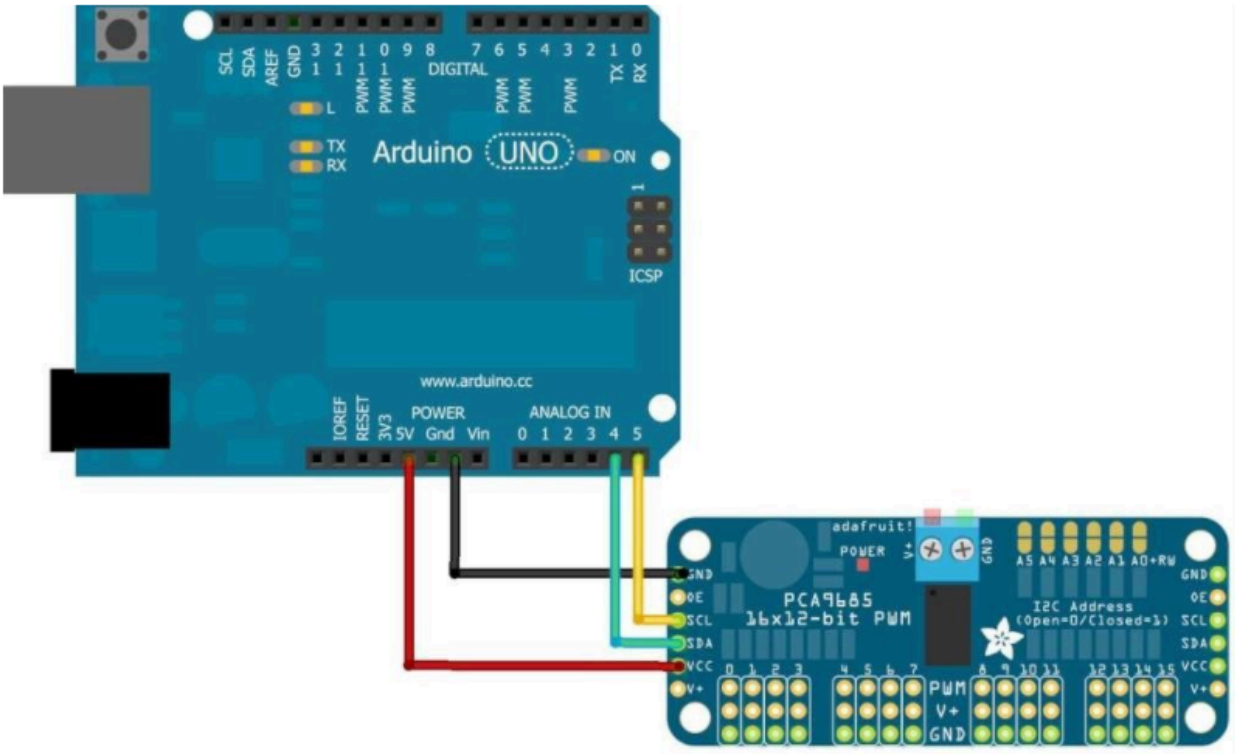
Installation

Install Arduino Library

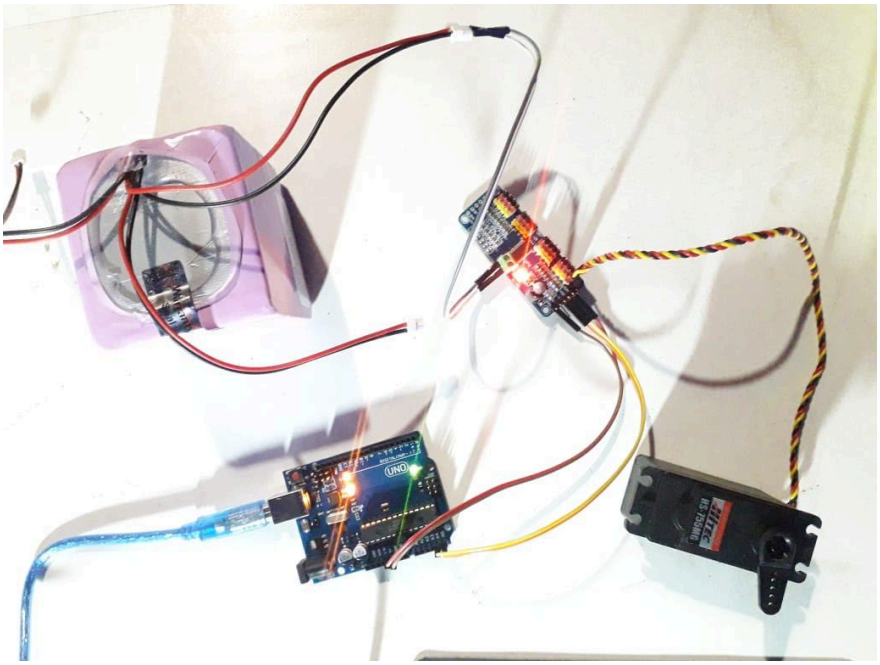
Step 1:
Download Zip File: **Adafruit-PWM-Servo-Driver-Library**
Step2:
Add into Arduino Libraries: ***Sketch >> Include Library >> Add .ZIP Library >> Select Library***

Pin Connection

Driver Pin	Arduino UNO	Arduino Mega
VCC	VCC	VCC
GND	GND	GND
SDA	A4	20
SCL	A5	21



Hardware Setup



Test Code 1

Try out the sample example code given in the library.

Open Example Code: *File >> Examples >> Adafruit_PWMServoDriver >> Servo*

Sample Code

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

// called this way, it uses the default address 0x40
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();
// you can also call it with a different address you want
//Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x41);
// you can also call it with a different address and I2C interface
//Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x40, Wire);

// Depending on your servo make, the pulse width min and max may vary, you
// want these to be as small/large as possible without hitting the hard stop
// for max range. You'll have to tweak them as necessary to match the servos you
// have!
#define SERVOMIN  150 // This is the 'minimum' pulse length count (out of 4096)
#define SERVOMAX  600 // This is the 'maximum' pulse length count (out of 4096)
#define USMIN  600 // This is the rounded 'minimum' microsecond length based on the minimum pulse of 150
#define USMAX  2400 // This is the rounded 'maximum' microsecond length based on the maximum pulse of 600
#define SERVO_FREQ 50 // Analog servos run at ~50 Hz updates

// our servo # counter
uint8_t servonum = 0;

void setup() {
  Serial.begin(9600);
  Serial.println("8 channel Servo test!");

  pwm.begin();
  /*
   * In theory the internal oscillator (clock) is 25MHz but it really isn't
   * that precise. You can 'calibrate' this by tweaking this number until
   * you get the PWM update frequency you're expecting!
   * The int.osc. for the PCA9685 chip is a range between about 23-27MHz and
   * is used for calculating things like writeMicroseconds()
```

```
  * Analog servos run at ~50 Hz updates, It is important to use an
  * oscilloscope in setting the int.osc frequency for the I2C PCA9685 chip.
  * 1) Attach the oscilloscope to one of the PWM signal pins and ground on
  *    the I2C PCA9685 chip you are setting the value for.
  * 2) Adjust setOscillatorFrequency() until the PWM update frequency is the
  *    expected value (50Hz for most ESCs)
  * Setting the value here is specific to each individual I2C PCA9685 chip and
  * affects the calculations for the PWM update frequency.
  * Failure to correctly set the int.osc value will cause unexpected PWM results
  */
 pwm.setOscillatorFrequency(27000000);
 pwm.setPWMFreq(SERVO_FREQ);  // Analog servos run at ~50 Hz updates

 delay(10);
}

// You can use this function if you'd like to set the pulse length in seconds
// e.g. setServoPulse(0, 0.001) is a ~1 millisecond pulse width. It's not precise!
void setServoPulse(uint8_t n, double pulse) {
 double pulselength;

 pulselength = 1000000;   // 1,000,000 us per second
 pulselength /= SERVO_FREQ;   // Analog servos run at ~60 Hz updates
 Serial.print(pulselength); Serial.println(" us per period");
 pulselength /= 4096;  // 12 bits of resolution
 Serial.print(pulselength); Serial.println(" us per bit");
 pulse *= 1000000;  // convert input seconds to us
 pulse /= pulselength;
 Serial.println(pulse);
 pwm.setPWM(n, 0, pulse);
}

void loop() {
 // Drive each servo one at a time using setPWM()
 Serial.println(servonum);
 for (uint16_t pulselen = SERVOMIN; pulselen < SERVOMAX; pulselen++) {
  pwm.setPWM(servonum, 0, pulselen);
 }

 delay(500);
 for (uint16_t pulselen = SERVOMAX; pulselen > SERVOMIN; pulselen--) {
  pwm.setPWM(servonum, 0, pulselen);
```

```
  }

  delay(500);

    // Drive each servo one at a time using writeMicroseconds(), it's not precise due to calculation rounding!
    // The writeMicroseconds() function is used to mimic the Arduino Servo library writeMicroseconds() behavior.
  for (uint16_t microsec = USMIN; microsec < USMAX; microsec++) {
    pwm.writeMicroseconds(servonum, microsec);
  }

  delay(500);
  for (uint16_t microsec = USMAX; microsec > USMIN; microsec--) {
    pwm.writeMicroseconds(servonum, microsec);
  }

  delay(500);

  servonum++;
  if (servonum > 7) servonum = 0; // Testing the first 8 servo channels
}
```

This example code is used for 8 servo motor testing which sweeps one by one from SERVOMAX to SERVOMIN range defined in the program. We cannot use the code as it is for our servo motor, as these values from the pulse length vary from a different brand to brand. First, we need to find the SERVOMAX and SERVOMIN pulse values by calibration.

PROBLEMS

Using the Max PWM signal range given values, the servo motor is unable to move to the desired angle and makes vibration and noise at extreme positions.

Solution

The servo motor we are using is HS-755MG Servo-Stock Rotation 200 degree which has Max PWM signal range from 570 to 2400 usec which means that if we give 500 PWM value to the drive will position to 0 degrees and if we give 2500 PWM value drive will position motor to 200 degrees in this code.

When we start the motor and try to get at 0 or 200 degrees using the default PWM pulse range you will find that motor vibrating and making noise, which is unwanted for us. So, we need to do calibration and find the perfect values for the PWM pulse range. We can find this using ***trial and error methods to calibrate the motor.***

Calibration Process

By Calibrating the PWM pulse value we can solve the first problem. For that, we have to try the different values near to 500 which position to 0 degrees without making any vibration in the motor and that will be MIN_PULSE value. In the same way, we have to find the value near 2500 which will position motor 200, and that will MAX_PULSE value.

By trial and error, we have to try multiple values for MIN_PULSE and MAX_PULSE, Starting with MIN_PULSE, I check 500 values making vibration and noise, so I change values with another which is inside in the range of 500 to 2500. By continuously changing the value I got 600 is perfect, one on which doesn't make any vibration and noise.

Move on to the MAX_PULSE, when I check 2500 which also makes vibration and noise so I move this value inside in the PWM range and I find 2450 is the best option.
We got the MIN_PULSE and MAX_PULSE values, let's check for the intermediate values like 90 or 180 degrees. May if we are lucky, we can get that position of servo but in my case, I am not. So I need to change the MIN_PULSE and MAX_PULSE values, now choose the values from the range of 600 to 2450 and check for the intermediate angle if it is correct then our select values are perfect. Otherwise, we have to do this again, in my case I got MIN_PULSE as 600 and MAX_PULSE as 2450 which are working almost correctly.

Test Code 2

This code is a modification of the first one with calibrated minimum pulse and maximum pulse values.

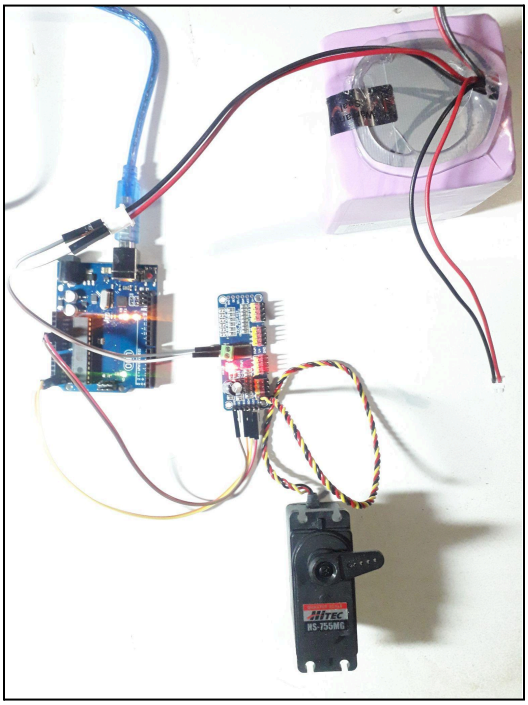
```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

#define MIN_PULSE 600
#define MAX_PULSE 2450
#define SERVO_FREQ 50

int angle = 100;

void setup(){
  Serial.begin(115200);
  Serial.println("Servo Testing!!!");
  pwm.begin();
  pwm.setPWMFreq(SERVO_FREQ);

}
```

```
void setServo(uint8_t pwm_pin, int angle) {
  int pulse_wide;
  int analog_value;

  pulse_wide = map(angle, 0, 200, MIN_PULSE, MAX_PULSE);
  analog_value = int(float(pulse_wide) / 1000000 * SERVO_FREQ * 4096);
  pwm.setPWM(pwm_pin, 0, analog_value);
}

void loop() {
  setServo(0, angle);
}
```

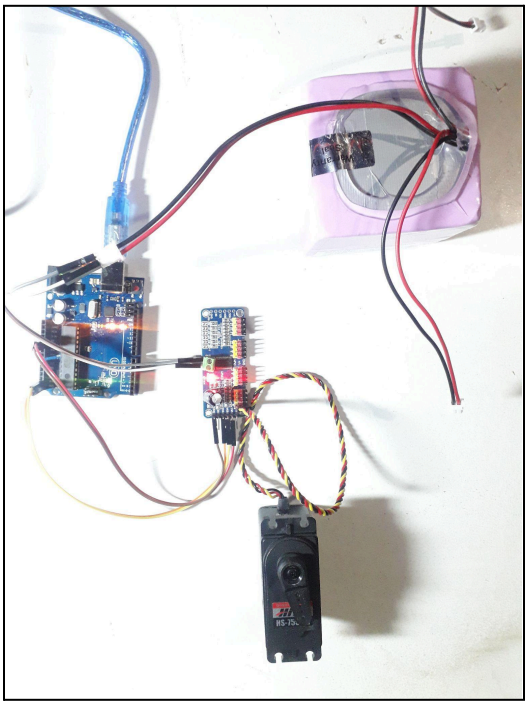
In this code we have to simply assign the value of the desired angle to the angle variable and upland the code to Arduino, Motor will move to that angle.

Testing

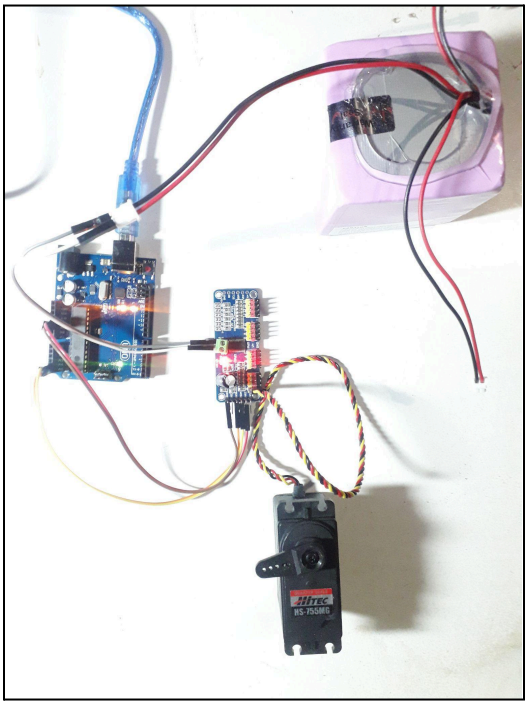
Servo Position at 0°



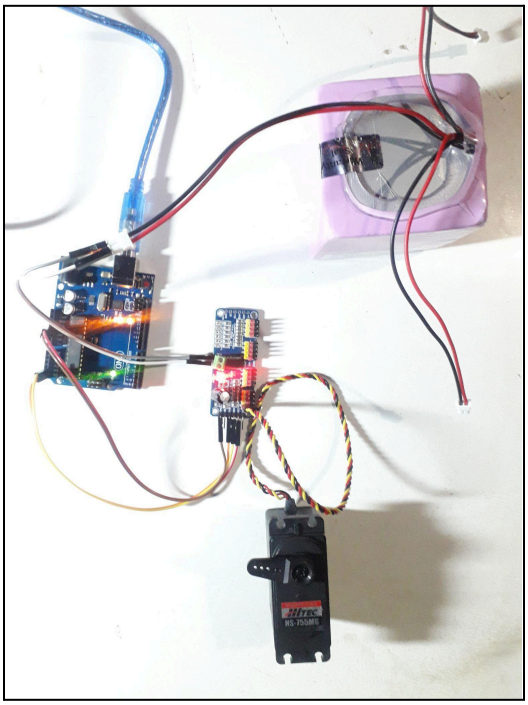
Servo Position at 90°



Servo Position at 180°



Servo Position at 200°



CONCLUSION

Given Max PWM Signal Range will not give the expected output, determining the MIN_PULSE and MAX_PULSE value for the PWM signal trial and error is the only method. We have to manually calibrate the minimum and maximum PWM pulse values to get the perfect angle position of the servo.

RESULT

Minimum PWM Pulse value = 600 usec
Maximum PWM Pulse value = 2450 usec
PWM Frequency = 50 Hz
Pulse Length = 1000000 usec per sec