Potentiometer adjusting steering gear Experiment

Introduction of steering gear

In the robot electromechanical control system, the steering gear control effect is an important factor influencing the performance. The steering gear can be used as the basic output actuator in mems and model aircraft. Its simple control and output make the single-chip microcomputer system very easy to interface with it.

A steering gear is a position (Angle) servo driver for control systems that require constant change in Angle and can be maintained. Currently in high grade remote control toys, such as aircraft models, including aircraft models, submarine models; Remote-controlled robots are already widely used. Steering gear is a common name, is actually a servo motor. It can be rotated to any Angle between 0 and 180 degrees and then stopped exactly as you command, thus suitable for control systems that require Angle change and retention. Steering gear is an unprofessional name. It is actually a servomotor, a set of automatic controls, made up of dc motors, reduction gear sets, sensors and control circuits. What is automatic control? So-called automatic control - by using a closed-loop feedback control circuit to constantly adjust the output deviation - keeps the system output constant.。

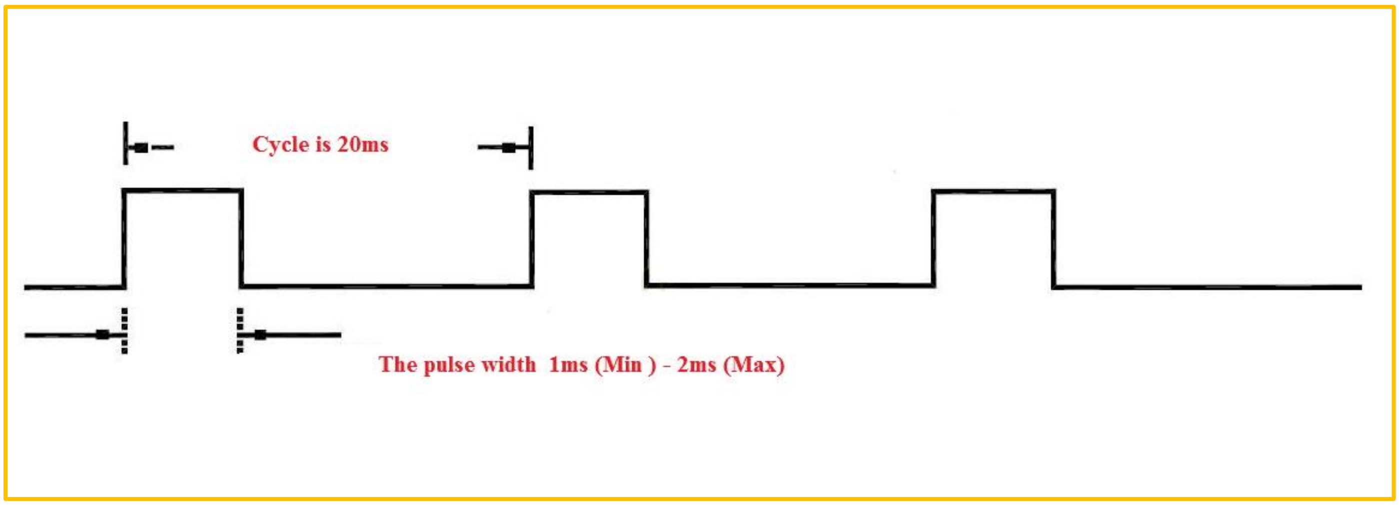
The working principle

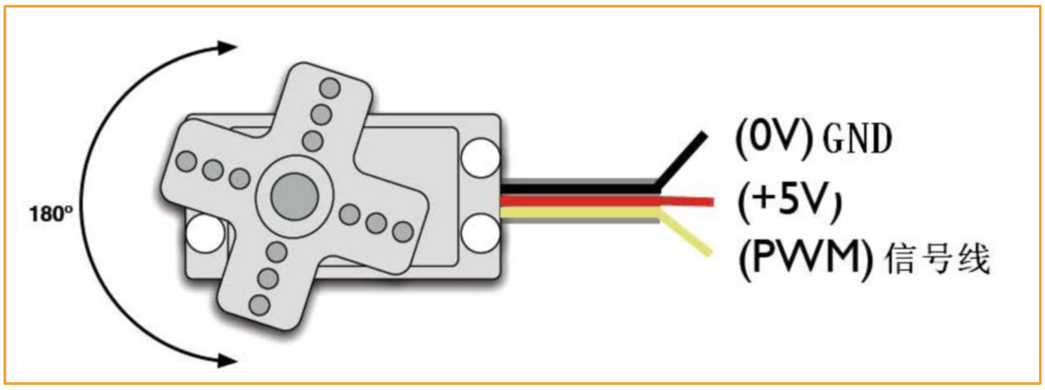
The steering gear control signal enters the signal modulation chip from the channel of the receiver to obtain the dc bias voltage. It has an internal reference circuit that generates a reference signal with a period of 20ms and a width of 1.5ms. The obtained dc offset voltage is compared with the voltage of the potentiometer to obtain the voltage difference output. Finally, the positive and negative output of the voltage difference to the motor drive chip determines the positive and negative rotation of the motor. When the motor speed is constant, the potentiometer is rotated by the cascade reduction gear, so that the voltage difference is 0 and the motor stops rotating. The steering gear has the maximum rotation Angle, and the middle position refers to the volume from that position to the minimum Angle, and the maximum Angle is exactly the same. The most important part, the maximum rotation Angle varies with the steering gear, but the bandwidth in the middle position is fixed, i.e. 1.5 ms.

Steering gear control:

The control of the steering gear generally requires a time base pulse of about 20ms, and the high level part of the pulse is generally the Angle control pulse part within the range of 0.5ms~2.5ms. Take 180-degree servo as an example, then the corresponding control relationship is as follows:

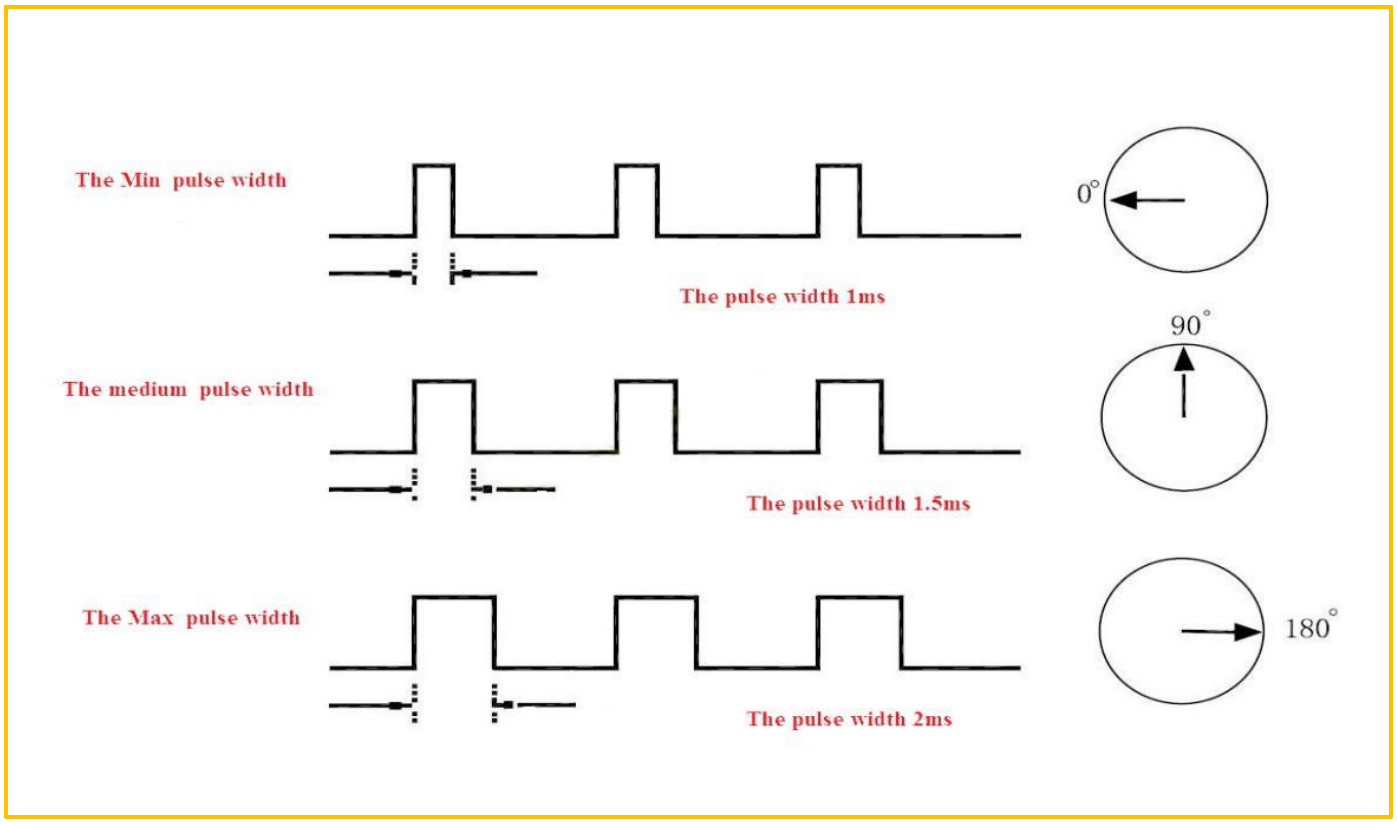
- 0.5ms--------------0degree；
- 1.0ms------------45degree；
- 1.5ms------------90degree；
- 2.0ms-----------135degree；
- 2.5ms-----------180degree；



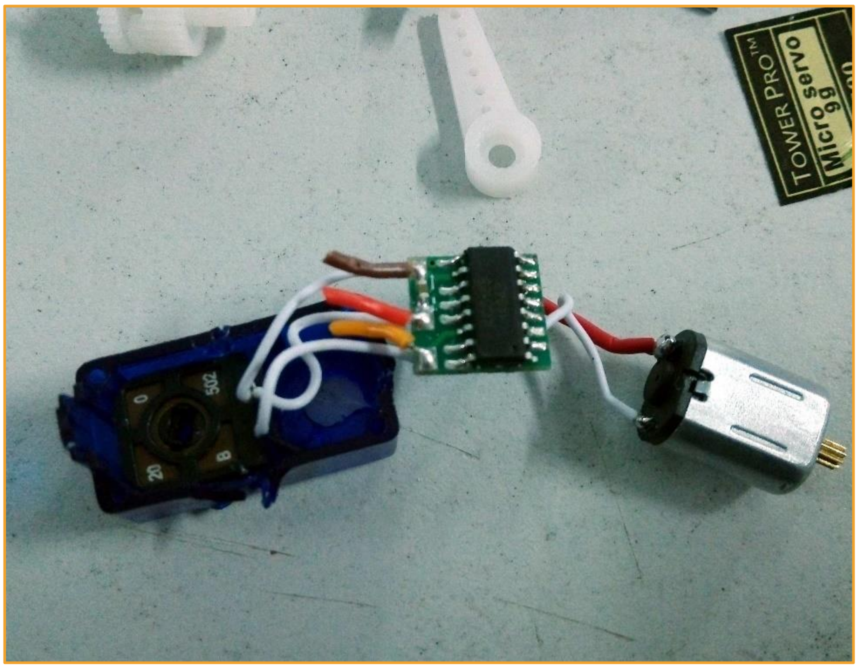


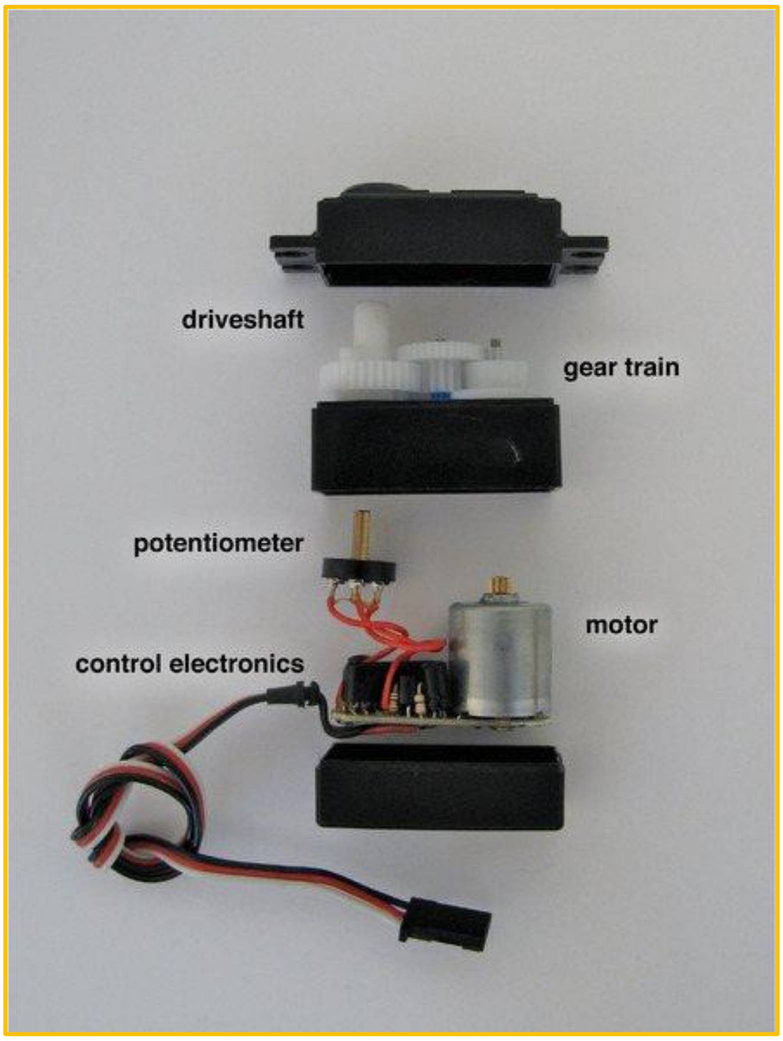
The rotation Angle is generated by a continuous pulse from the control line. This method is called pulse modulation. The length of the pulse determines the rotation Angle of the steering gear. For example, the steering gear rotates to a 1.5 millisecond pulse in the middle position (for a 180° steering gear, the middle position is 90°). When the control system issues a command to move the steering gear to a specific position and hold it at an Angle, the effect of external forces does not change the Angle. The Angle will not remain constant until the control system sends out pulses continuously to stabilize the steering Angle.

When the steering gear receives a pulse less than 1.5ms, the output shaft will be taken as the standard middle position and rotated anticlockwise at a certain Angle. When the received pulse is greater than 1.5ms, the output axis rotates clockwise. The maximum and minimum values may be different for different brands of steering gear, or even for different steering gear of the same brand.



Internal Structure of Steering Gear





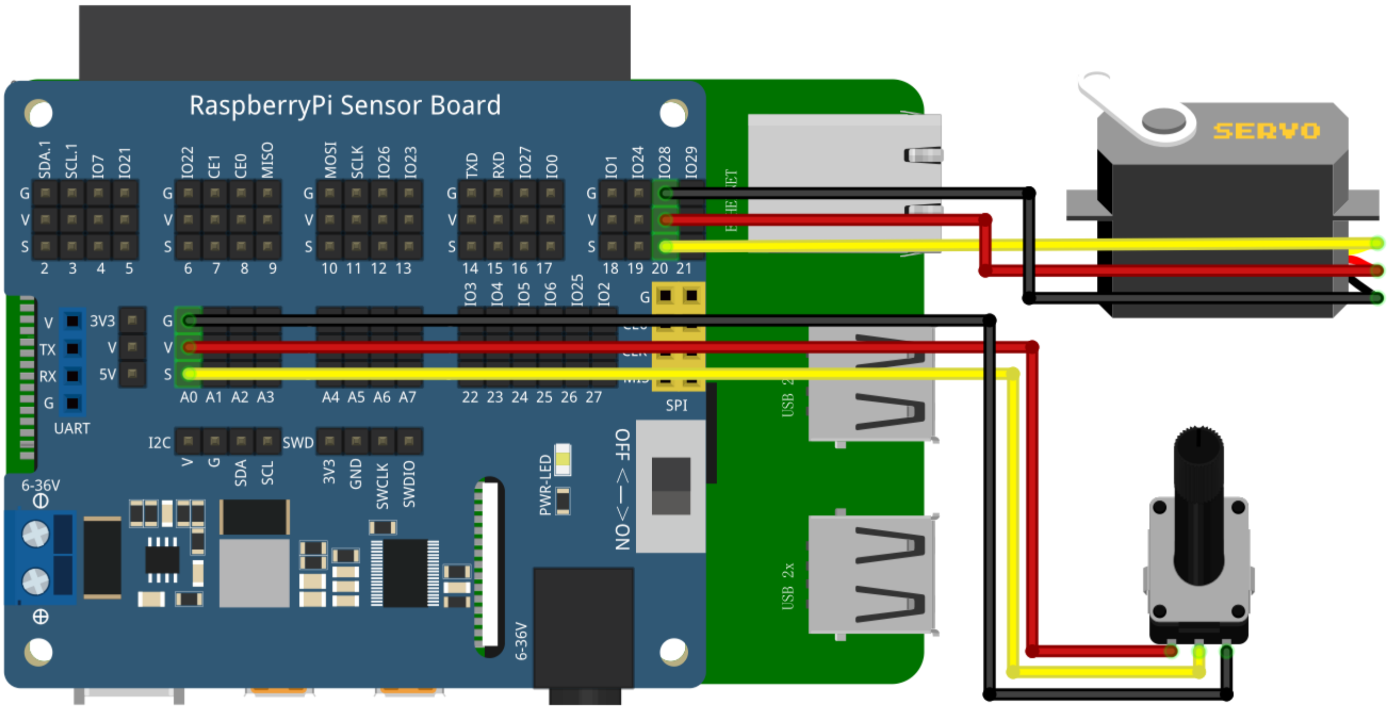
Experiment Purpose

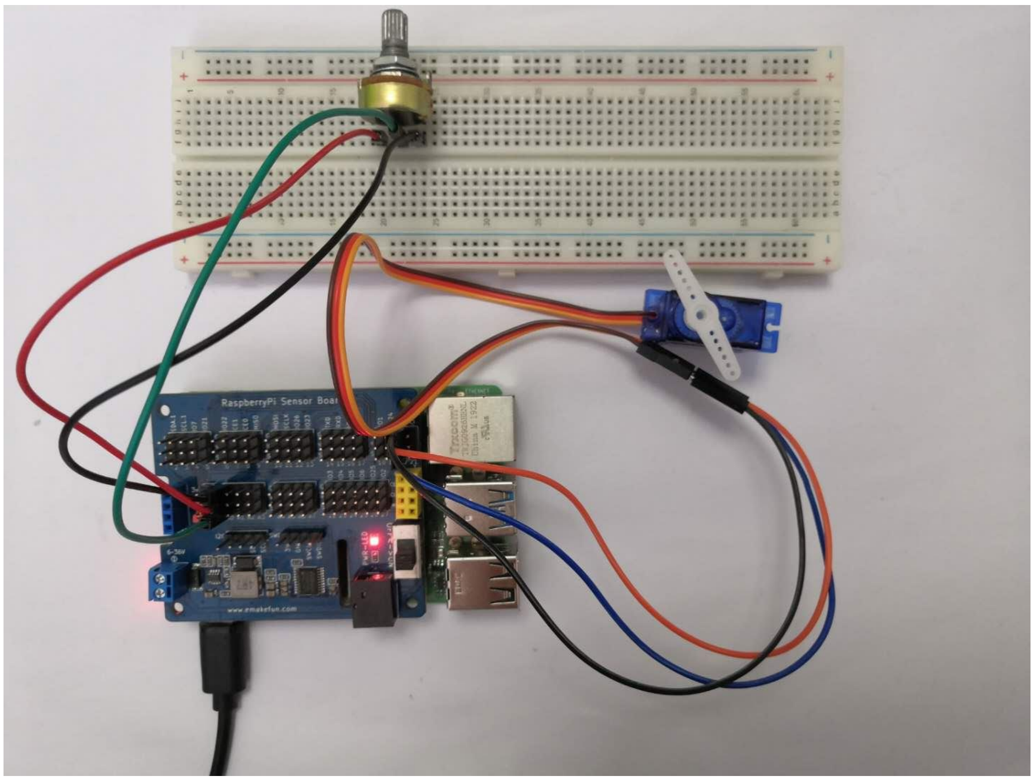
The aim is to controlling Steering Gear rotation through the potentiometer

Component List

- RaspberryPi motherboard
- Raspberry Pi expansion board
- Breadboard
- Power cord
- 10k potentiometer*1
- SG90 servo * 1
- Several jumpers

Wiring


RaspberryPi Sensor Board
SERVO
UART
I2C
SWD
SPI
PWR-LED
6-36V
USB 2x
ON <—> OFF



C++ program

```
#include <stdio.h>//Import the basic library
#include <wiringPi.h>//Import the Raspberry Pi WiringPi encoding IO control library
#include <wiringPiI2C.h>//Import the Raspberry Pi WiringPi coding I2C control library

int  value = 0 ;
float val= 0.0 ;
int cyc =20000;
int pwmPin = 28;

void pwm_fun(int temp)
{
    digitalWrite(pwmPin, HIGH);
    delayMicroseconds(500 + temp*500 / 45);
    digitalWrite(pwmPin,0);
    delayMicroseconds((cyc - (500 + temp*500 / 45)));
}

int main()
{
    wiringPiSetup();
    wiringPiI2CSetup(0x04);
    pinMode(pwmPin, OUTPUT);
    while(1)
    {
        value =  wiringPiI2CReadReg16(0x04, 0x10);
        val = float(value * 180) / 4095;
        printf("%d\n", int(val));
        pwm_fun(int(val));
        delay(500); // 1 second delay
    }
}
```

Python program

```
import time
import smbus as smbus
import RPi.GPIO as GPIO
import signal
import atexit
```

```
atexit.register(GPIO.cleanup)
servopin = 20
cyc = 0.2
ADC=smbus.SMBus(1)#Declare to use I2C 1
GPIO.setmode(GPIO.BCM)
GPIO.setup(servopin, GPIO.OUT)

def pwm_change(temp):
    GPIO.output(servopin, True)
    time.sleep(0.0005 + float(temp)*0.0005 / 45)
    GPIO.output(servopin, False)
    time.sleep(0.02 - (0.0005 + float(temp)*0.0005 / 45))

while True:
    ADC.write_byte(0x04, 0x10)#Write a byte to the slave
    val = ADC.read_word_data(0x04, 0x10)
    val = float(val * 180) / 4095
    print(val)#Raspberry Pi reads the data returned by the expansion board and prints it
out
    pwm_change(int(val))
    time.sleep(0.05)#Delay 1 second
```

Java program

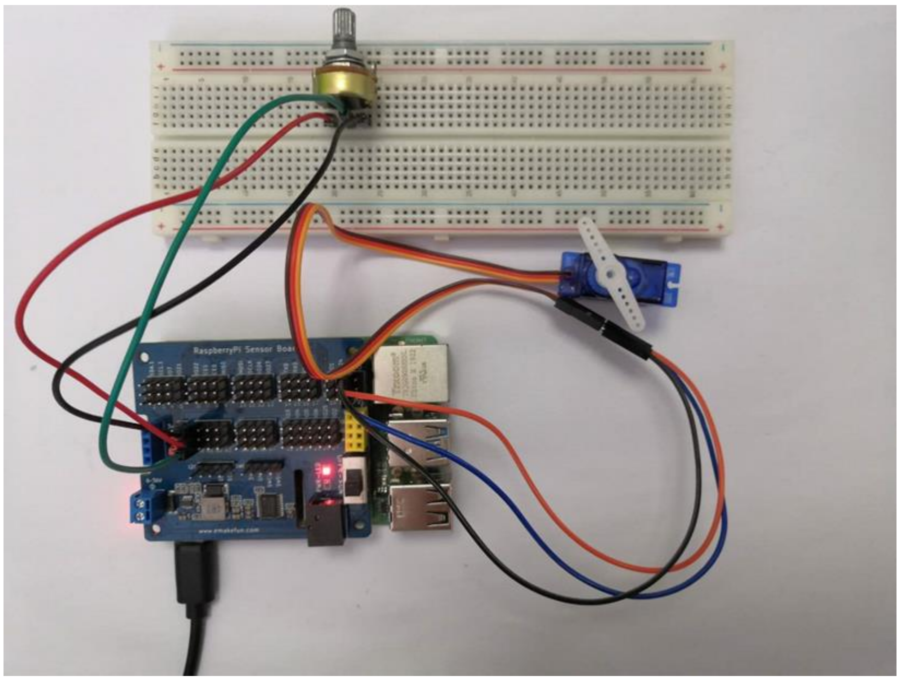
```
import com.pi4j.wiringpi.Gpio;
import com.pi4j.wiringpi.I2C;
import com.pi4j.wiringpi.GpioInterrupt;
import com.pi4j.wiringpi.GpioInterruptListener;
import com.pi4j.wiringpi.GpioInterruptEvent;

public class Potentiometer_adjustment {
    static int LEDPIN  = 21;
    static int value;
    static {
        // setup wiring pi
        if (Gpio.wiringPiSetup() == -1) {
            System.out.println(" ==>> GPIO SETUP FAILED");
        }

        Gpio.pinMode(LEDPIN, Gpio.OUTPUT);
    }
    static void led_pwm(int val){
        Gpio.digitalWrite(LEDPIN, Gpio.HIGH);
        Gpio.delayMicroseconds(500 + val*500 / 45);
        Gpio.digitalWrite(LEDPIN,Gpio.LOW);
        Gpio.delayMicroseconds((20000 - (500 + val*500 / 45)));
    }

    public static void main(String args[]) throws InterruptedException{
        int fd = I2C.wiringPiI2CSetup(0x04);
        for ( ; ;) {
            int Potentiometer_val =  I2C.wiringPiI2CReadReg16(fd, 0x10);
            Potentiometer_val = Potentiometer_val * 180 / 4095;
            System.out.println(Potentiometer_val);
            for(int i = 0; i < 10; i++) {
                Potentiometer_adjustment.led_pwm(Potentiometer_val);
            }
        }
    }
}
```

Experimental results



After we upload the program, we can change the angle of the steering gear by rotating the positioner. The angle of the steering gear is 0-180 degrees, which lays the foundation for the subsequent robotic arm.