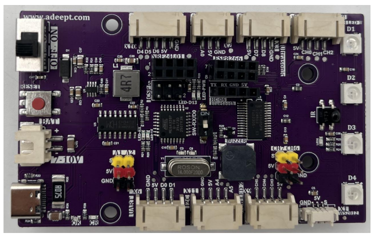
Lesson 7 Control the Servo to Rotate

7.1 Overview

In the realm of robot manufacturing and electronic projects, mastering servo control is of fundamental importance. This tutorial aims to provide in-depth knowledge on using Arduino for servo control, covering both direct control methods and extended control techniques via the PCA9685 chip.

7.2 Required Components

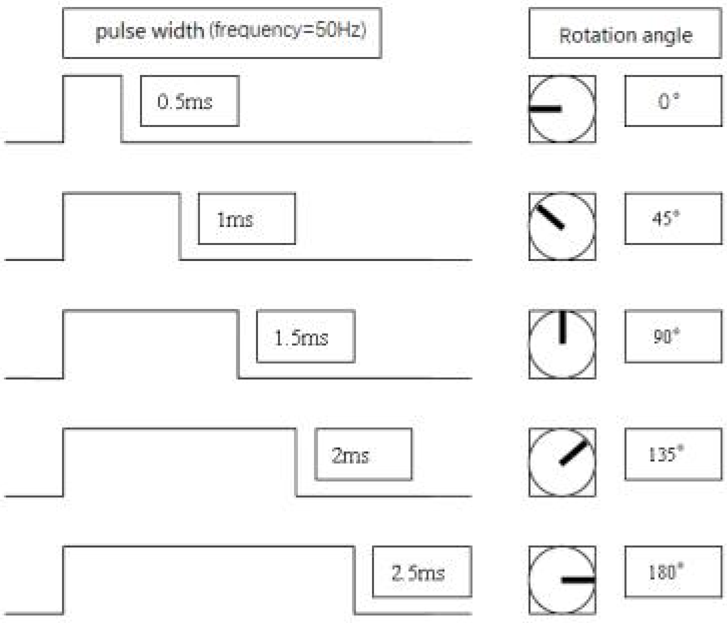
Components	Quantity	Picture
Adeept Robot Control Board	1	
Type-C USB Cable	1	
Servo	1	

7.3 Principle Introduction

7.3.1 Working principle

The servo operates based on a pulse positioning mechanism. Essentially, when the servo receives a pulse, it rotates by a corresponding angle to achieve displacement. The servo has the ability to send pulses, and for each rotation angle, a specific number of pulses are emitted. This forms a closed - loop system for pulse reception and transmission, enabling the system to precisely control the motor's rotation and achieve accurate positioning.

In practical scenarios, the Adeept robot control board sends PWM (Pulse - Width Modulation) signals to the servo. These signals are processed by the integrated circuit (IC) on the circuit board, which calculates the rotation direction of the driving motor. The rotation is then transmitted to the swing arm through a reduction gear. Simultaneously, the position detector returns a position signal to determine whether the set position has been reached. At a frequency of 50Hz, different pulse widths correspond to different rotation angles: a 0.5ms pulse corresponds to 0°, a 1ms pulse to 45°, a 1.5ms pulse to 90°, a 2ms pulse to 135°, and a 2.5ms pulse to 180°.



7.3.2 Principle of the Write() function

In programming, the write() function is utilized to control the rotation of the servo. For standard servos, this function rotates the servo shaft to the corresponding angular position. For continuous - rotation servos, the write() function can set their rotation speed. A value of 0 represents full - speed rotation in one direction, 180 represents full - speed rotation in the opposite direction, and 90 represents a stop. In this tutorial, we focus on standard servo motors.

7.3.3 PCA9685

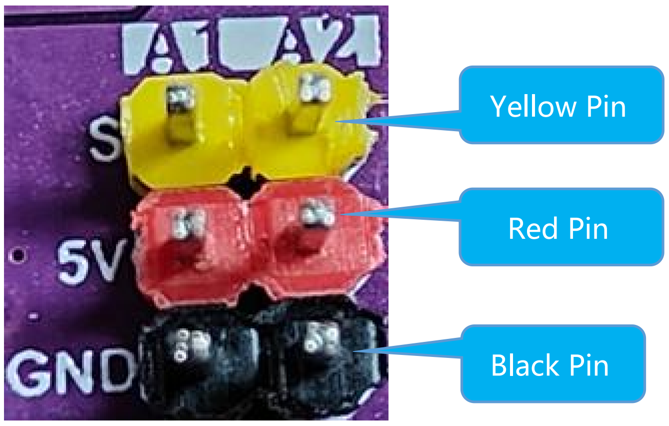
The PCA9685 is a 16 - channel, 12 - bit PWM controller. It is specifically designed to control devices such as servos and LEDs. The PCA9685 communicates with the main control device (like Arduino or Raspberry Pi) through the I2C bus, allowing the main control device to manage multiple servos, motors, or LEDs simultaneously.

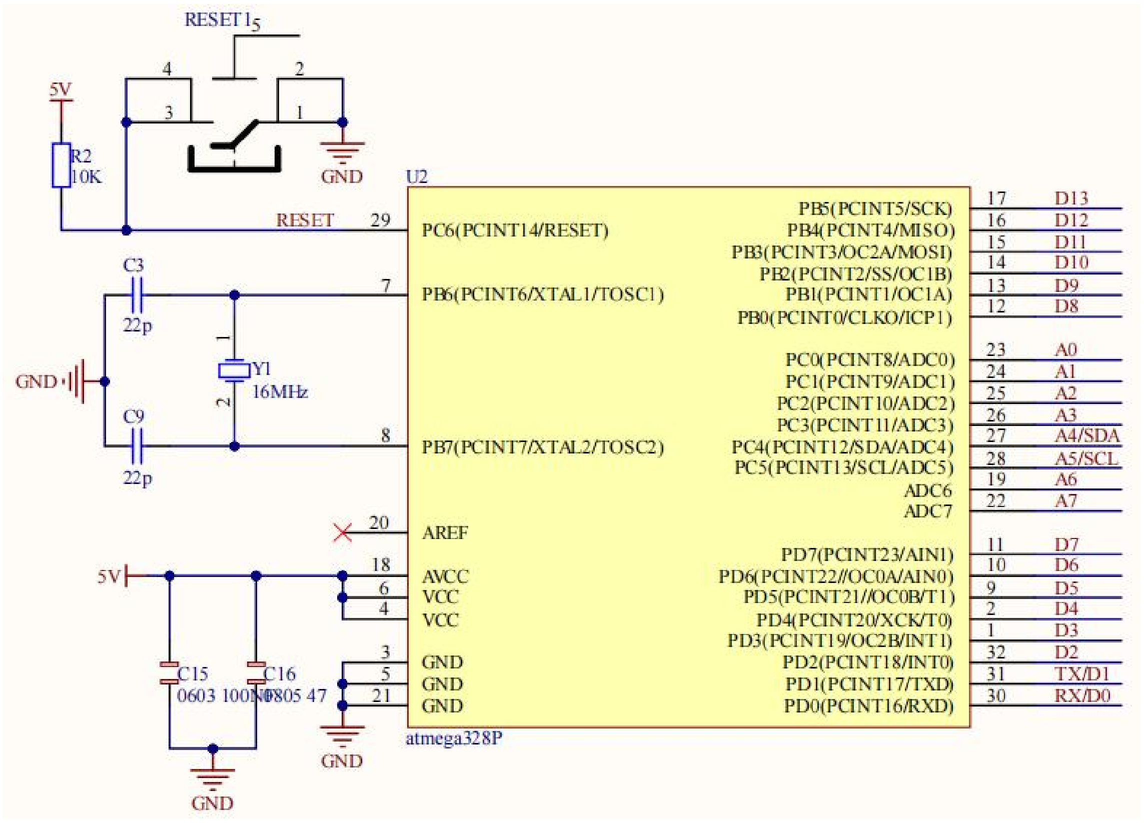
This chip integrates 16 independent PWM output channels. Each channel can adjust the pulse width to control the brightness of LEDs or the angle of servos. The PWM output frequency of the PCA9685 can be customized according to application requirements. By altering the pulse width, the brightness or position of external devices can be precisely adjusted.

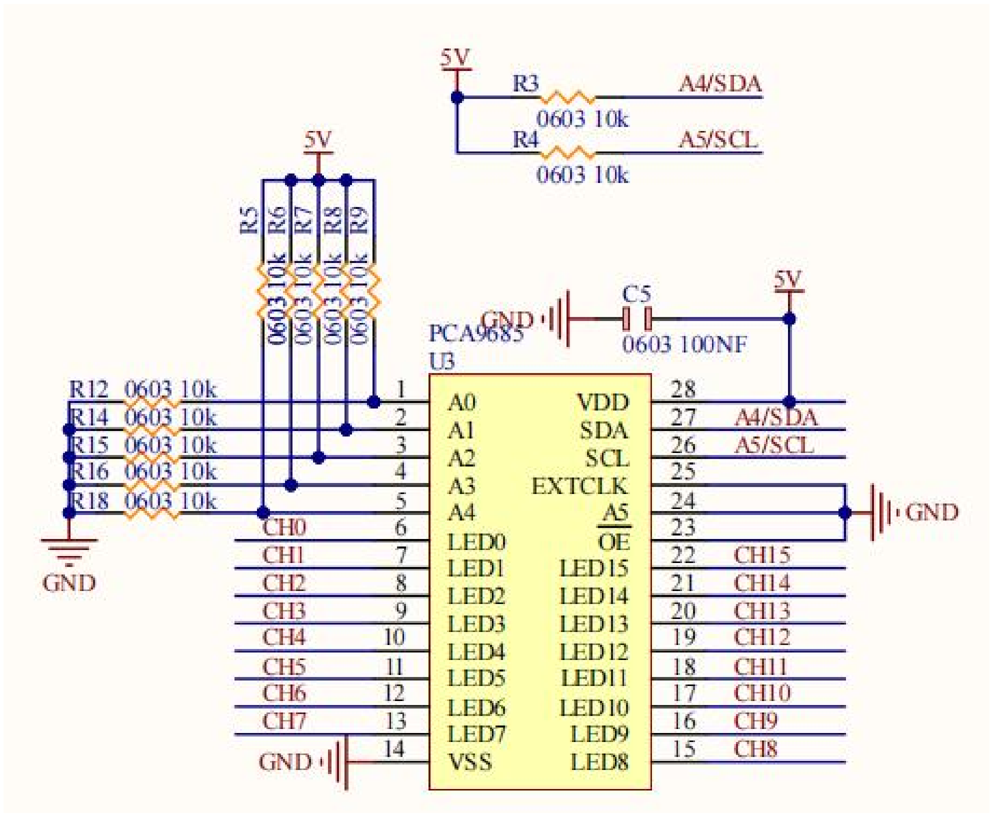
- 16-channel PWM output: PCA9685 can control up to 16 devices at the same time, such as servos or LEDs.
- 12-Bit Precision: Each channel's PWM output supports 12-bit resolution, meaning there are 4096 different brightness or angle levels to choose from.
- Multi-frequency selection: PWM output frequency can be set to meet the requirements of different equipment.

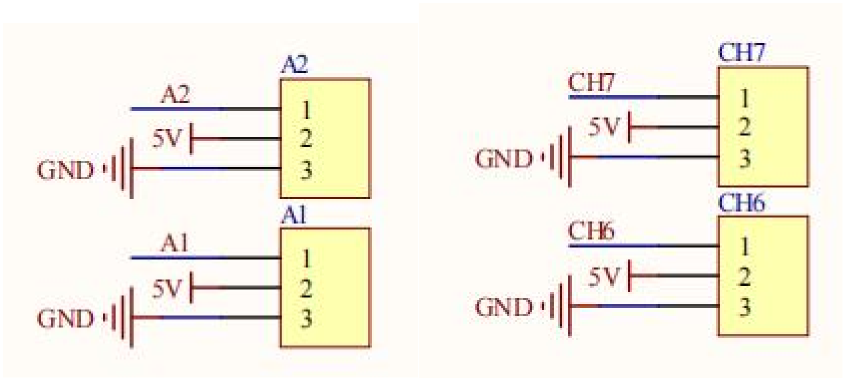
7.4 Wiring Diagram

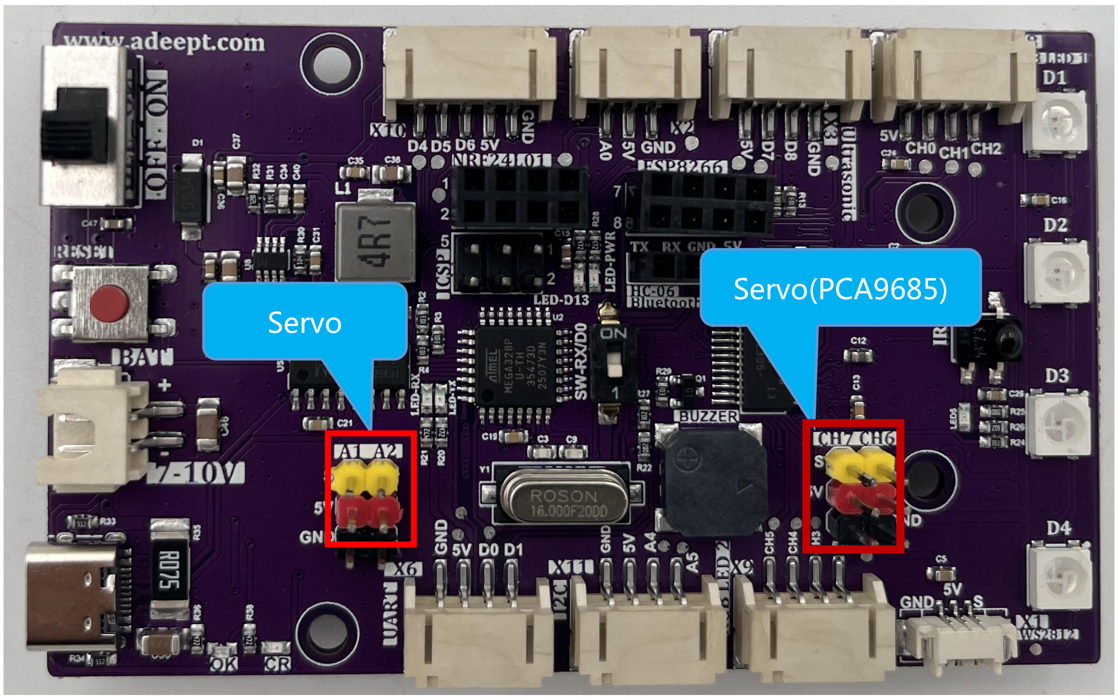
Connect the servo to ports A1/A2 and CH6/CH7, paying attention to the color correspondence: the yellow servo wire is connected to the yellow pin, the red servo wire is connected to the red pin, and the black servo wire is connected to the black pin.




RESET
5V
R2
10K
GND
U2
RESET
C3
22p
Y1
16MHz
C9
22p
PC6(PCINT14/RESET)
PB6(PCINT6/XTAL1/TOSC1)
PB7(PCINT7/XTAL2/TOSC2)
AREF
AVCC
VCC
VCC
GND
GND
GND
C15
0603 100NF
C16
0805 47
atmega328P
PB5(PCINT5/SCK) D13
PB4(PCINT4/MISO) D12
PB3(PCINT3/OC2A/MOSI) D11
PB2(PCINT2/SS/OC1B) D10
PB1(PCINT1/OC1A) D9
PB0(PCINT0/CLKO/ICP1) D8
PC0(PCINT8/ADC0) A0
PC1(PCINT9/ADC1) A1
PC2(PCINT10/ADC2) A2
PC3(PCINT11/ADC3) A3
PC4(PCINT12/SDA/ADC4) A4/SDA
PC5(PCINT13/SCL/ADC5) A5/SCL
ADC6 A6
ADC7 A7
PD7(PCINT23/AIN1) D7
PD6(PCINT22/OC0A/AIN0) D6
PD5(PCINT21/OC0B/T1) D5
PD4(PCINT20/XCK/T0) D4
PD3(PCINT19/OC2B/INT1) D3
PD2(PCINT18/INT0) D2
PD1(PCINT17/TXD) TX/D1
PD0(PCINT16/RXD) RX/D0

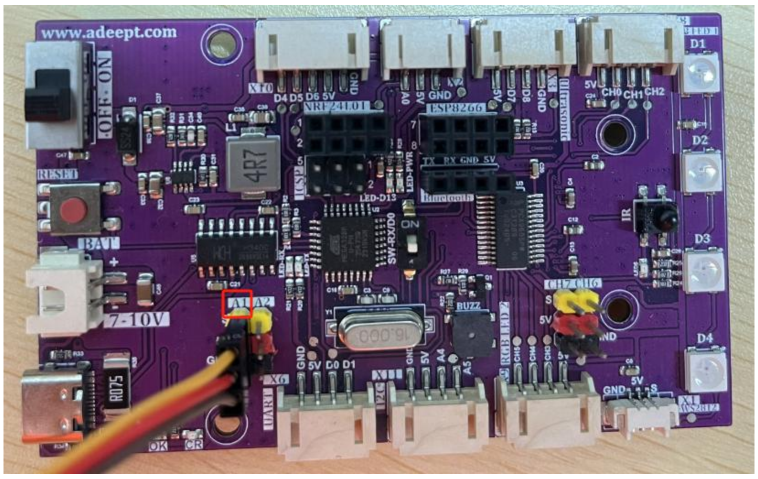

5V
R3 0603 10k A4/SDA
R4 0603 10k A5/SCL
R5 R6 R7 R8 R9
0603 10k
C5 0603 100NF
PCA9685
U3
R12 0603 10k
R14 0603 10k
R15 0603 10k
R16 0603 10k
R18 0603 10k
GND
A0 A1 A2 A3 A4
LED0 LED1 LED2 LED3 LED4 LED5 LED6 LED7
VSS
VDD SDA SCL EXTCLK A5 OE
LED15 LED14 LED13 LED12 LED11 LED10 LED9 LED8
CH0 CH1 CH2 CH3 CH4 CH5 CH6 CH7
CH15 CH14 CH13 CH12 CH11 CH10 CH9 CH8
A4/SDA
A5/SCL



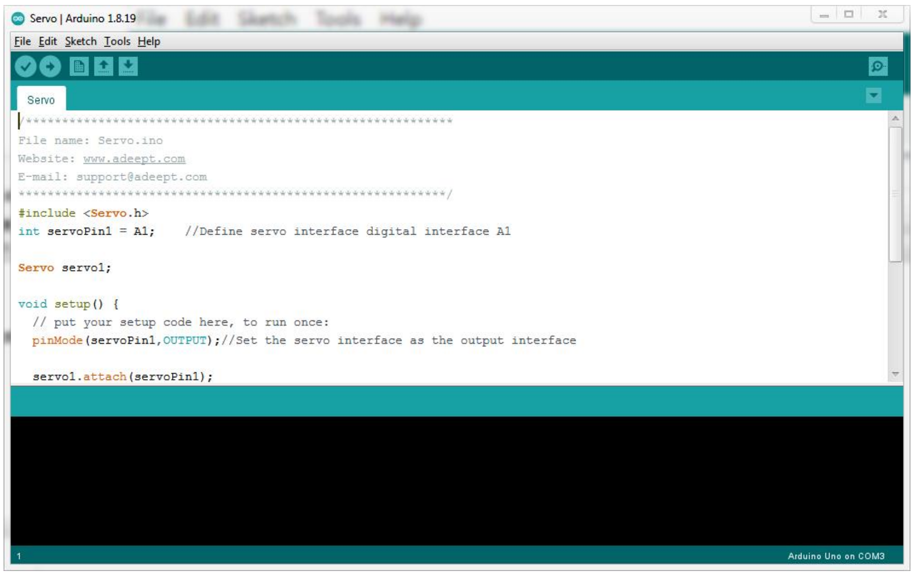


7.5 Demonstration

1. Connect your computer and Adeept Robot Control Board (Arduino Board) with a USB cable.
2. Please connect the servo wire to port **A1**.



3. Open "**03_Servo/Servo**" folder in "/Code", double-click "Servo.ino".



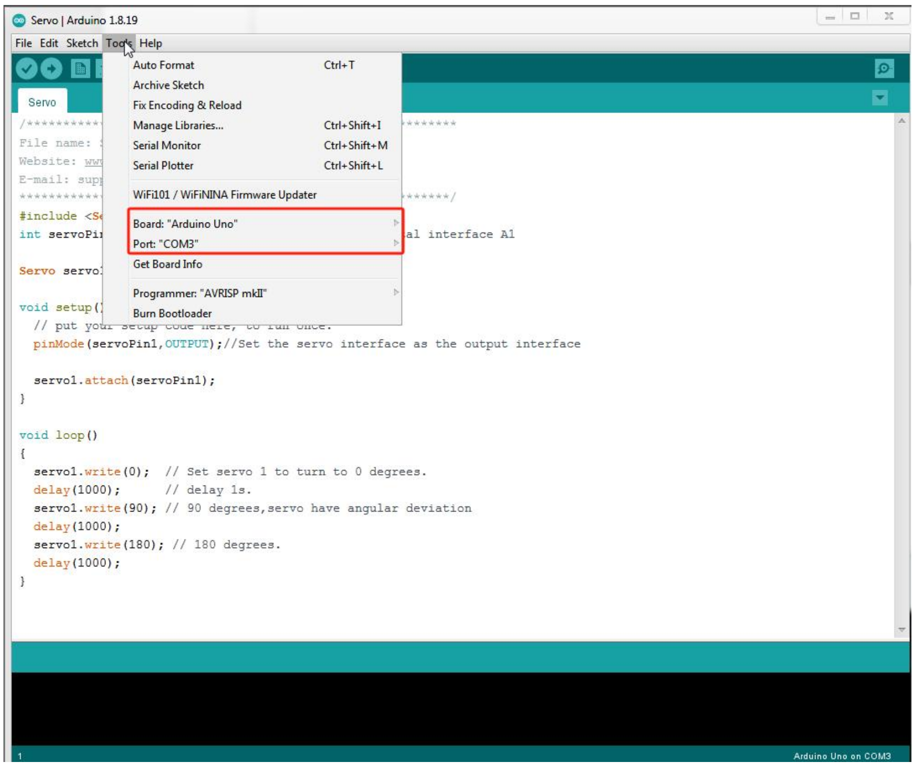
```
/*************************************************************
File name: Servo.ino
Website: www.adeept.com
E-mail: support@adeept.com
*************************************************************/
#include <Servo.h>
int servoPin1 = A1;    //Define servo interface digital interface A1

Servo servo1;

void setup() {
  // put your setup code here, to run once:
  pinMode(servoPin1,OUTPUT);//Set the servo interface as the output interface

  servo1.attach(servoPin1);
```

4. Select development board and serial port.

Board: Tools--->Board--->Arduino AVR Boards--->Arduino Uno

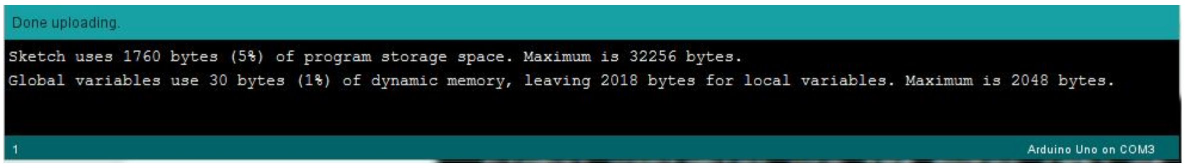
Port: Tools --->Port--->COMx

Note: The port number will be different in different computers.


Servo | Arduino 1.8.19
File Edit Sketch Tools Help
Servo
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
1
Arduino Uno on COM3

```
/***********                                 ********
File name: 
Website: ww
E-mail: sup
************                                 *******/
#include <Se
int servoPi                                  al interface A1

Servo servo1

void setup()
  // put your setup code here, to run once:
  pinMode(servoPin1,OUTPUT);//Set the servo interface as the output interface

  servo1.attach(servoPin1);
}

void loop()
{
  servo1.write(0);  // Set servo 1 to turn to 0 degrees.
  delay(1000);      // delay 1s.
  servo1.write(90); // 90 degrees,servo have angular deviation
  delay(1000);
  servo1.write(180); // 180 degrees.
  delay(1000);
}
```

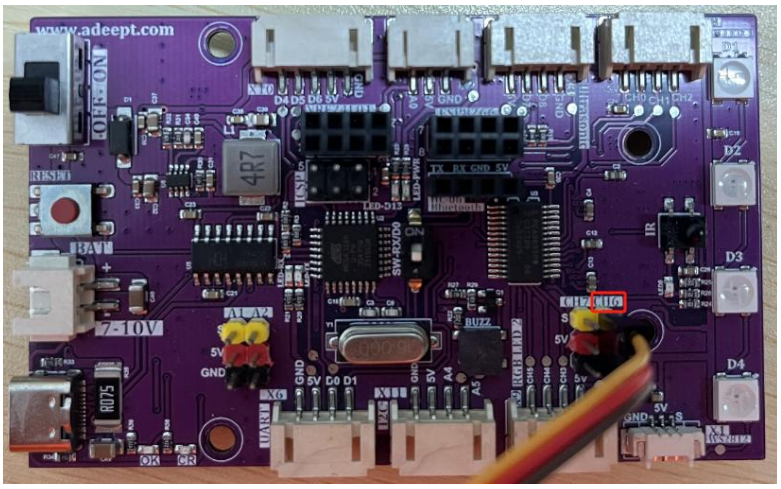
5. After opening, click to upload the code program to the Arduino. If there is no error warning in the console below, it means that the Upload is successful.



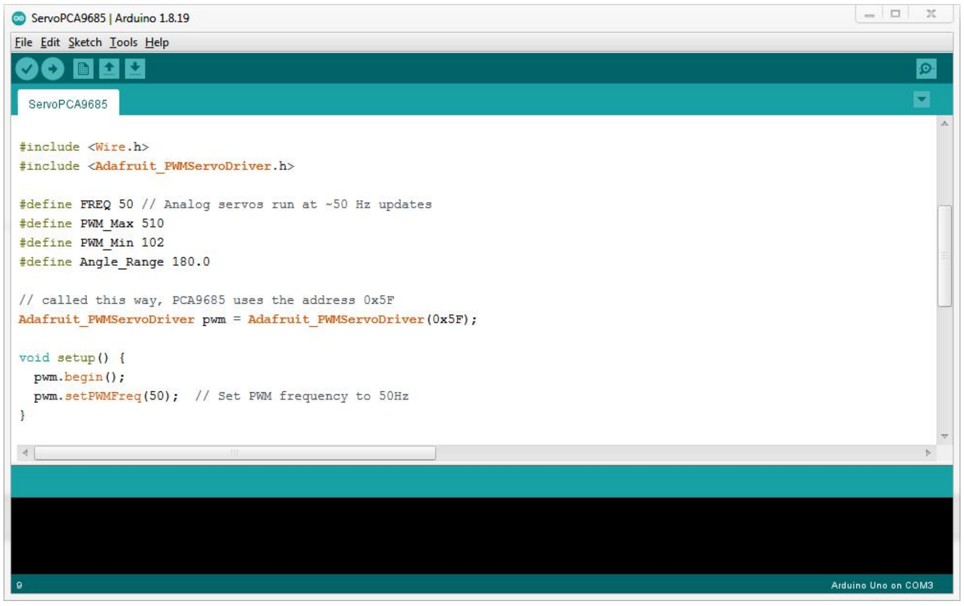
```
Done uploading.
Sketch uses 1760 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 30 bytes (1%) of dynamic memory, leaving 2018 bytes for local variables. Maximum is 2048 bytes.
```

6. After the program runs successfully, the onboard servo will rotate in sequence to 0 degrees, 90 degrees, and 180 degrees, maintaining each angle for 1 second, and so on.

7. Please connect the servo wire to port **CH6**.



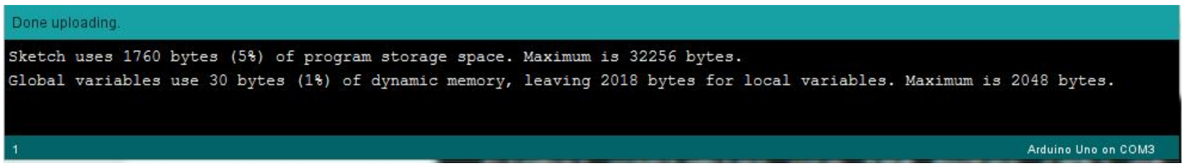
8. Open “**03_Servo/ServoPCA9685**” folder in “/Code”, double-click “ServoPCA9685.ino”.



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

#define FREQ 50 // Analog servos run at ~50 Hz updates
#define PWM_Max 510
#define PWM_Min 102
#define Angle_Range 180.0

// called this way, PCA9685 uses the address 0x5F
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x5F);

void setup() {
  pwm.begin();
  pwm.setPWMFreq(50);  // Set PWM frequency to 50Hz
}
```

```
9                                                    Arduino Uno on COM3
```

9. After opening, click to upload the code program to the Arduino. If there is no error warning in the console below, it means that the Upload is successful.



```
Done uploading.
Sketch uses 1760 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 30 bytes (1%) of dynamic memory, leaving 2018 bytes for local variables. Maximum is 2048 bytes.
1                                                    Arduino Uno on COM3
```

10. After the program runs successfully, the PCA9685 chip controls the servo of the CH6 channel to rotate sequentially to 90 °, 0 °, and 180 °, each for 1 second, and repeat in a loop. During this period, the characteristic of using a 12 bit register on the chip to control PWM is utilized to

calculate PWM values, accurately control the servo angle, and achieve periodic angle change control.

7.6 Code

Complete code refer to Servo.ino

```
/***********************************************************
File name: Servo.ino
Website: www.adeept.com
E-mail: support@adeept.com
***********************************************************/
#include <Servo.h>
int servoPin1 = A1;    //Define servo interface digital interface A1

Servo servo1;

void setup() {
  // put your setup code here, to run once:
  pinMode(servoPin1,OUTPUT);//Set the servo interface as the output interface

  servo1.attach(servoPin1);
}

void loop()
{
  servo1.write(0);  // Set servo 1 to turn to 0 degrees.
  delay(1000);      // delay 1s.
  servo1.write(90); // 90 degrees,servo have angular deviation
  delay(1000);
  servo1.write(180); // 180 degrees.
  delay(1000);
}
```

Complete code refer to ServoPCA9685.ino

```
/***********************************************************
File name: ServoPCA9685.ino
Description: Control the servo through PCA9685.
***********************************************************/
/*
PCA9685 can set the update frequency, and the time base pulse period of 20ms is equivalent to 50HZ
update frequency. PCA9685 uses a 12-bit register to control the PWM ratio. For 0.5ms, it is equivalent
to a register value of 0.5/20*4096=102. And so on as follows:
0.5ms————0 degrees: 0.5/20*4096 = 102
1.0ms———45 degrees: 1/20*4096 = 204
1.5ms——90 degrees: 1.5/20*4096 = 306
2.0ms——-135 degrees: 2/20*4096 = 408
2.5ms——-180 degrees: 2.5/20*4096 =510
*/

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

```

```
#define FREQ 50 // Analog servos run at ~50 Hz updates
#define PWM_Max 510
#define PWM_Min 102
#define Angle_Range 180.0

// called this way, PCA9685 uses the address 0x5F
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x5F);

void setup() {
  pwm.begin();
  pwm.setPWMFreq(50);  // Set PWM frequency to 50Hz
}

void loop() {
  setAngle(6, 0, 90);   // CH6 servo turn to 90°, servo have angular deviation
  delay(1000);          // delay 1s.

  setAngle(6, 0, 0);    // turn to 0°
  delay(1000);

  setAngle(6, 0, 180);  // rurn to 180°
  delay(1000);
}

// Convert PWM value to angle value.
int setAngle(int num, int start, int angle){
  // num: servo channel.
  // start: starting point of the pulse signal.
  // angle: servo angle.
  int pwm_value = int(PWM_Min +((PWM_Max - PWM_Min)/Angle_Range)*angle);
  Serial.println(pwm_value);
  pwm.setPWM(num, start, pwm_value);
  }
```

Code explanation

Servo.ino

Initialization phase:

The setup() function is called once at the start of the program. It configures the specified pin (A1 in this case) as an output pin for the servo control signal and attaches the servo object to this pin.

Loop control process:

Stage 1: The servo system first rotates to 0 degrees → Delay for 1 second

Stage 2: Then the servo rotates to 90 degrees→ Delay for 1 second

Stage 3: Finally, the servo rotates to 180 degrees→ Delay for 1 second

Forming a cyclic effect, the servo system cycles at these angles.

ServoPCA9685.ino

Initialization phase:

In the setup() function, the PCA9685 module is initialized using the begin() function. The PWM frequency is set to 50Hz, which is a common frequency for servo control. This frequency determines how often the servo receives new control signals.

Loop control process:

Stage 1: The servo system first rotates to 0 degrees → Delay for 1 second

Stage 2: Then the servo rotates to 90 degrees→ Delay for 1 second

Stage 3: Finally, the servo rotates to 180 degrees→ Delay for 1 second

Forming a cyclic effect, the servo system cycles at these angles.