# Servos Control

Control the servo (S1) on the Robduino expansion board to rotate at different angles.

## Device connection

### Hardware connection

Use a Type-B data cable to connect the Arduino Uno and the computer.

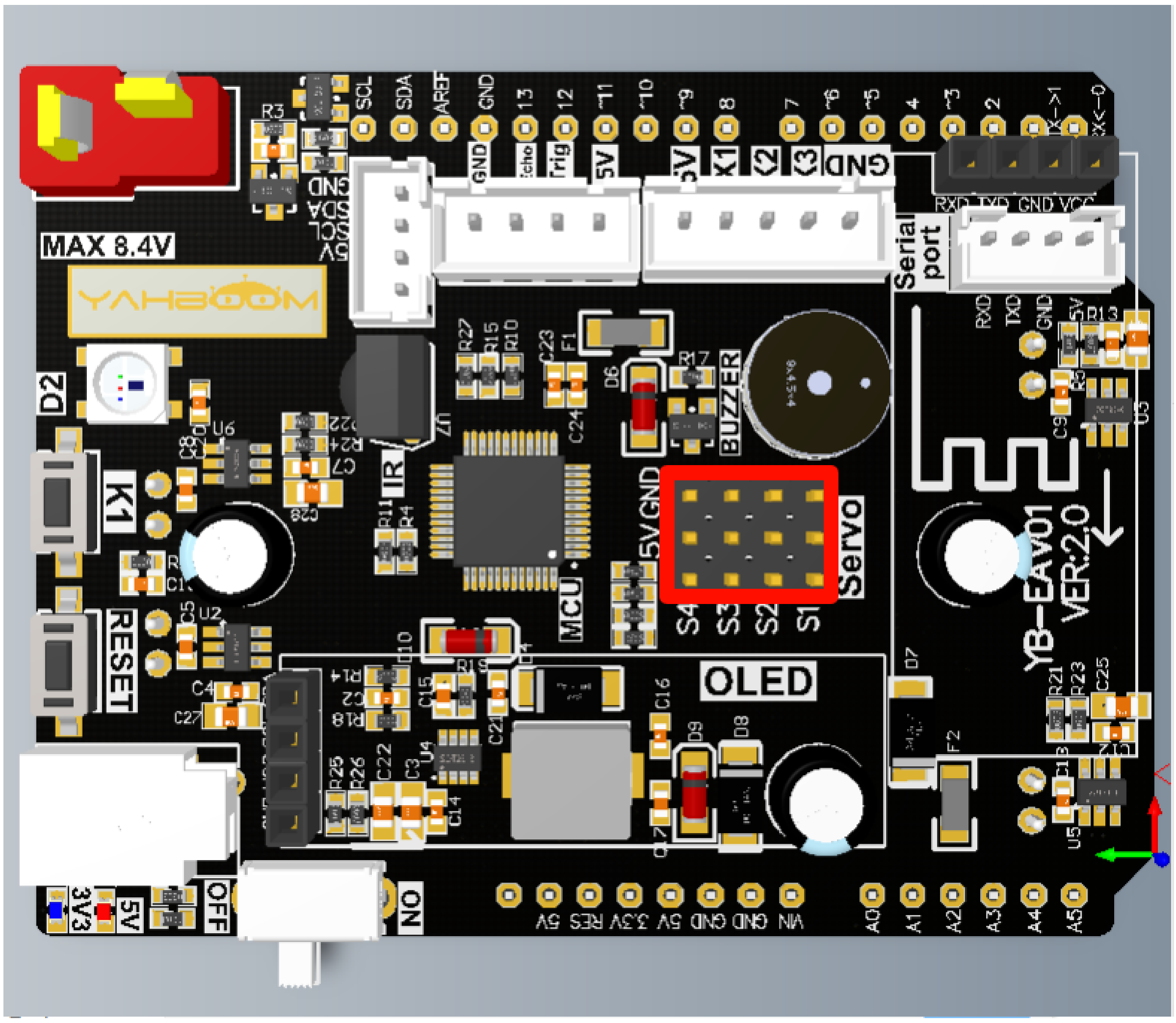
### Software connection

Open the "Arduino IDE" software and select the model and serial port number corresponding to the development board.

## Control the servo

The servo is directly driven by the underlying driver chip on the Robduino expansion board. The underlying driver chip and the Arduino Uno board use I2C communication, and the I2C address is 0x40.

The position selected by the red box is the location of the servo interface:



## Control principle

The angle of rotation of the servo is controlled by changing the duty cycle of the PWM signal that controls the servo signal pin.

### Common parameters

**Period**: The duration of a complete PWM waveform;

**Duty cycle**: The ratio of the high level duration to the cycle time;

**Frequency**: The reciprocal of the period is called the frequency, that is, the number of PWM periods generated per second;

```
Set the period of the PWM control signal to 20ms, that is, the frequency of 50Hz;
the high level time of the pulse determines the angle of rotation of the servo
```

## Control pins

| Peripheral module | Bottom-layer driver chip (using I2C communication with Arduino Uno board) |
|---|---|
| Servo 1 (S1) | LED0 (0) |
| Servo 2 (S2) | LED1 (1) |

| Peripheral module | Bottom-layer driver chip (using I2C communication with Arduino Uno board) |
|---|---|
| Servo 3 (S3) | LED2 (2) |
| Servo 4 (S4) | LED3 (3) |

```
Considering that the WiFi camera is installed on the servo, some angles of the servo rotation may cause the WiFi camera module and the expansion board or the module connection line to be squeezed, so the program limits the servo rotation angle range to [35°, 145°]
```

## Code Analysis

Here we only briefly introduce the code content. For detailed code, please refer to the corresponding code file, which is provided in the download area!

- Contains `Wire` and `Adafruit_PWMServoDriver` libraries

```
#include <Wire.h>                    // 包含Wire(I2C)通讯库 Include Wire library
#include <Adafruit_PWMServoDriver.h>  // 包含Adafruit PWMServoDriver库 Include Adafruit PWMServoDriver library
```

- Define the servo control pins and I2C addresses

```
// 定义舵机控制引脚 Define the servos control pin
#define Servo1_PIN 0
#define Servo2_PIN 1
#define Servo3_PIN 2
#define Servo4_PIN 3

// 定义底层芯片I2C设备地址 Define the IIC address of the Bottom-layer driver chip
#define Bottom_Layer_Driver_ADDR 0x40
```

- Define servo related parameters

```
// 定义舵机相关参数 Define servo related parameters
#define SERVO_MIN_PULSE_WIDTH 500
#define SERVO_MAX_PULSE_WIDTH 2500
#define SERVO_PULSE_WIDTH_CYCLE 20000
#define SERVO_MIN_ANGLE 0
#define SERVO_MAX_ANGLE 180
#define PWM_FREQUENCY 50
```

- Create an instance of the Adafruit_PWMServoDriver class

```
// Create an instance of the Adafruit_PWMServoDriver class
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(Bottom_Layer_Driver_ADDR);
```

- Set the servo rotation angle

```
/**
 * @brief 设置舵机旋转角度 Set the servo rotation angle
 * @param servoPin: 舵机控制引脚 Servo control pin
 * @param angle: 舵机旋转角度 Servo rotation angle
 * @retval 无 None
 */
void setServoAngle(uint8_t servoPin, int angle) {
  long pulseWidth = map(angle, SERVO_MIN_ANGLE, SERVO_MAX_ANGLE,
SERVO_MIN_PULSE_WIDTH, SERVO_MAX_PULSE_WIDTH);
  long PWM_Value = pulseWidth * 4096 / SERVO_PULSE_WIDTH_CYCLE;
  pwm.setPWM(servoPin, 0, PWM_Value);
}
```

- Initialization Code

```
void setup() {
  Wire.begin();                  // 初始化I2C通讯 Initialize I2C communication
  delay(1000);                   // 如果控制异常，可以适当增加延时 If the function is
abnormal, you can increase the delay
  pwm.begin();                   // PWM初始化 Initialize the Pulse Width
Modulation (PWM) library
  pwm.setPWMFreq(PWM_FREQUENCY);  // 设置PWM频率 Set the PWM frequency
}
```

- Looping code

```
void loop() {
  setServoAngle(Servo1_PIN, 90);
  delay(2000);
  for (int angle = 35; angle <= 145; angle += 15) {
    setServoAngle(Servo1_PIN, angle);
    delay(200);
  }
  for (int angle = 145; angle >= 35; angle -= 15) {
    setServoAngle(Servo1_PIN, angle);
    delay(200);
  }
}
```

## Experimental results

After compiling the program successfully, upload the code to the Arduino Uno development board.

After the program is started, servo 1 (servo of S1 interface) will rotate periodically: 90°→35°→145°→35° (90° stays for a longer time).

```
When burning a program, you cannot use other programs to occupy the serial port
or an external serial communication module (for example: WiFi camera module),
otherwise the program cannot be burned or an error message will be prompted!
```