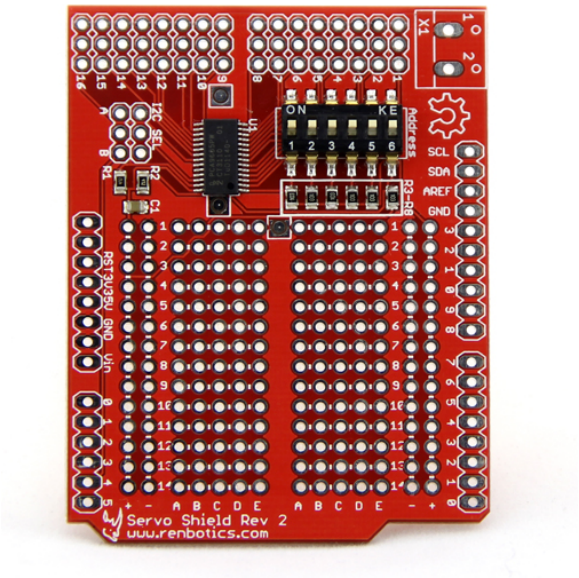
RENBOTICS
HOBBYIST ROBOTICS PARTS

Servo Shield 2



Applications

- Robotics
- Animatronics
- Mechatronic Art

Features

- 16 Servo Channels
- Convenient screw terminal for servo power supply
- 196 Point breadboard style prototyping area
- Compatible with Arduino Duemilanove, UNO, Leonardo and Arduino Mega 2560
- Dip switch to easily set and/or change address
- 50Hz or 60Hz mode
- Easy to use API

1. License

2. Disclaimer of Liability

Renbotics is not responsible for any special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, good-will, damage to or replacement of equipment or property, and any costs or recovering of any material or goods associated with the assembly or use of this product. Renbotics reserves the right to make substitutions and changes to this product without prior notice.

3. Description

The Renbotics Servo Shield Rev 2 is an Arduino-compatible shield that uses only two pins to drive 16 servos per shield. The Renbotics Servo Shield Rev 2 uses a PCA9685 over I2C to provide 16 free-running servo outputs. The address of the board is selectable via a DIP switch making it easy to stack up to 62 shields to control up to 992 servos. Two jumpers make it easy to select the Arduino host model used.

4. Features

- 16 Servo Channels
- Convenient screw terminal for servo power supply
- 196 Point breadboard style prototyping area
- Compatible with Arduino Duemilanove, UNO, Leonardo and Arduino Mega 2560
- Dip switch to easily set and/or change address
- 50Hz or 60Hz mode
- Easy to use API

5. Applications

- Robotics
- Animatronics
- Mechatronic Art

6. Parts List

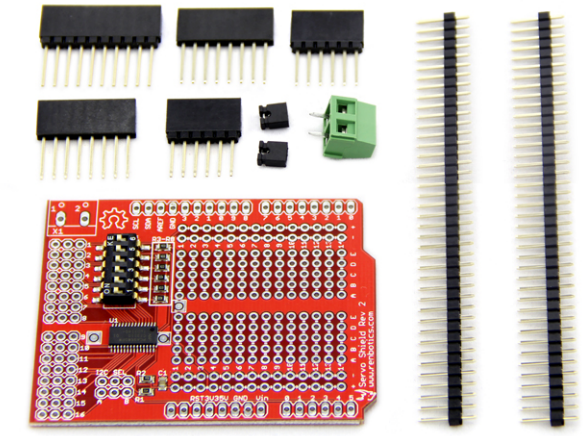
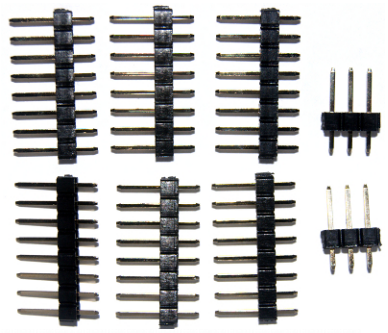


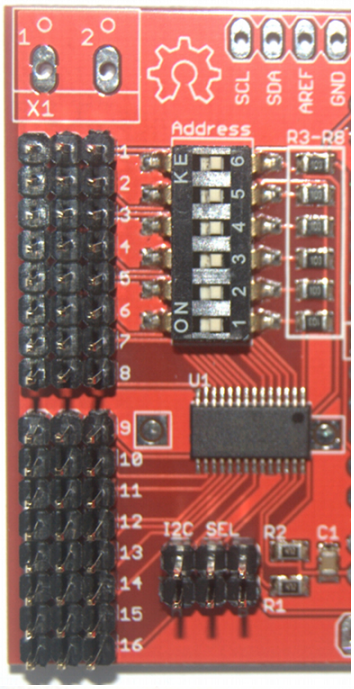
Image 2:Renbotics Servo Shield Rev 2 Parts

1 x Renbotics Servo Shield Rev 2 PCB with all surface mount components
1 x 10 pin Female Stacking Header
2 x 8 pin Female Stacking Headers
2 x 6 pin Female Stacking Headers
1 x 2 pin Screw Terminal
6 x 8 pin Male Breakaway Headers
2 x 3 pin Male Breakaway Headers
2 x Jumpers

7. Assembly

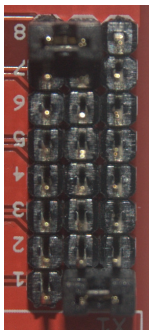


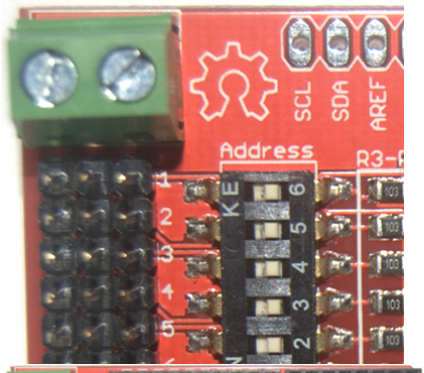
1. If you received two long Male Breakaway Headers cut them to 6 lengths of 8 pins and 2 lengths of 3 pins.



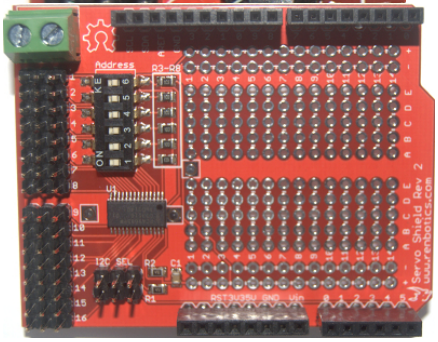
2. Solder the headers.

TIP: Use the jumpers to help hold the headers in place while soldering.





3. Solder the screw terminal.



4. Solder the Arduino Female Stacking Headers to match the configuration of your selected host board.

8. RC Servo Control Basics

A RC Servo is controlled by sending it a pulses ranging from 1ms to 2ms in duration, Pulse-width modulation (PWM), at 50Hz (50 pulses per second) or more. On a typical servo a 1.5ms pulse will center a servo at 90deg, a 1ms pulse will move the servo to 0deg and a 2ms pulse will move the servo to 180deg (See Image 8).

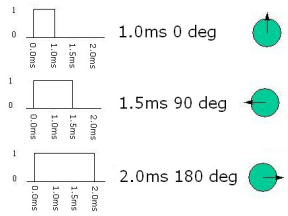


Image 8: Servo Control Overview

Image 9: Servo Cable

A typical RC Servo has three wires, one for the control signal and the other two for power (See Image 9).

The most common wire colors are:

Signal	White or Orange
Negative	Black
Positive	Red or Brown

On the Servo Shield the Negative (Black) wire always faces to the outside of the board.

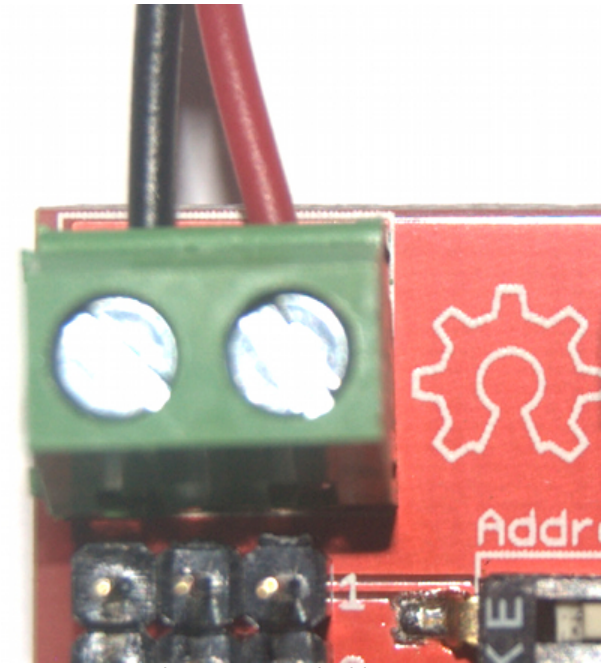


Image 10: Renbotics Servo Shield Rev 2 power wiring

9. Library

Installing the Library

Download servoshield.zip from http://www.renbotics.com/files/servoshield2.zip and extract it to your **arduino-[version]/hardware/libraries** folder.

Functions

`ServoShield2 ServoShield2(uint8_t addr, uint8_t mode)`

Initializes the Servo Shield 2 at the address specified using the mode specified. Two modes are supported 50Hz and 60Hz. The mode determines the rate at which the servo's positions are updated. The default mode is 50Hz.

e.g.
```
ServoShield2 servos = ServoShield2(127, 50);
```

`int setposition(int servo, int position);`

Sets the position of the specified servo. Returns 0 if successfully set; returns 1 if instruction failed.

e.g.
```
servos.setposition(1, 1500);
```

`int setbounds(int servo, int minposition, int maxposition);`

Sets the valid maximum and minimum bounds of the specified servo; returns 1 if instruction failed.

Defaults are 1000 and 2000.

e.g.
```
servos.setbounds(1, 1000, 2000);
```

`int getposition(int Servo);`

Returns the current position of the specified servo.

e.g.
```
int position = servos.getposition(1);
```

int start();

Starts the servo controller.

e.g.
```
servos.start();
```

int invertservo(int servo);

Inverts the signal sent to the servo, e.g. if invert is set and a position of 1000 is sent to the servo it will automatically be inverted to 2000.

e.g.
```
servos.invertservo(1);
```

10. Selecting Arduino Model

Different Arduino's and Arduino Compatible boards have their I2C pins, SCL and SDA, located in diffident places.

If you are using a new Arduino, e.g. the Leonardo, or an Arduino Mega the I2C pins are located to the left of digital pin 13. For these boards place the jumpers over position A.

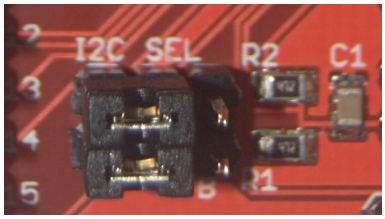


Image 11: Renbotics Servo Shield Rev 2 Jumpers in position A

For older Arduino's, e.g. the Duemilanove, the I2C pins are the same as analogue pins 4 and 5. For these boards place the jumpers over position B.

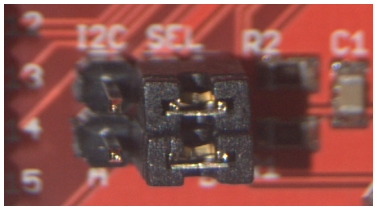


Image 12: Renbotics Servo Shield Rev 2 Jumpers in position B

12. Addressing

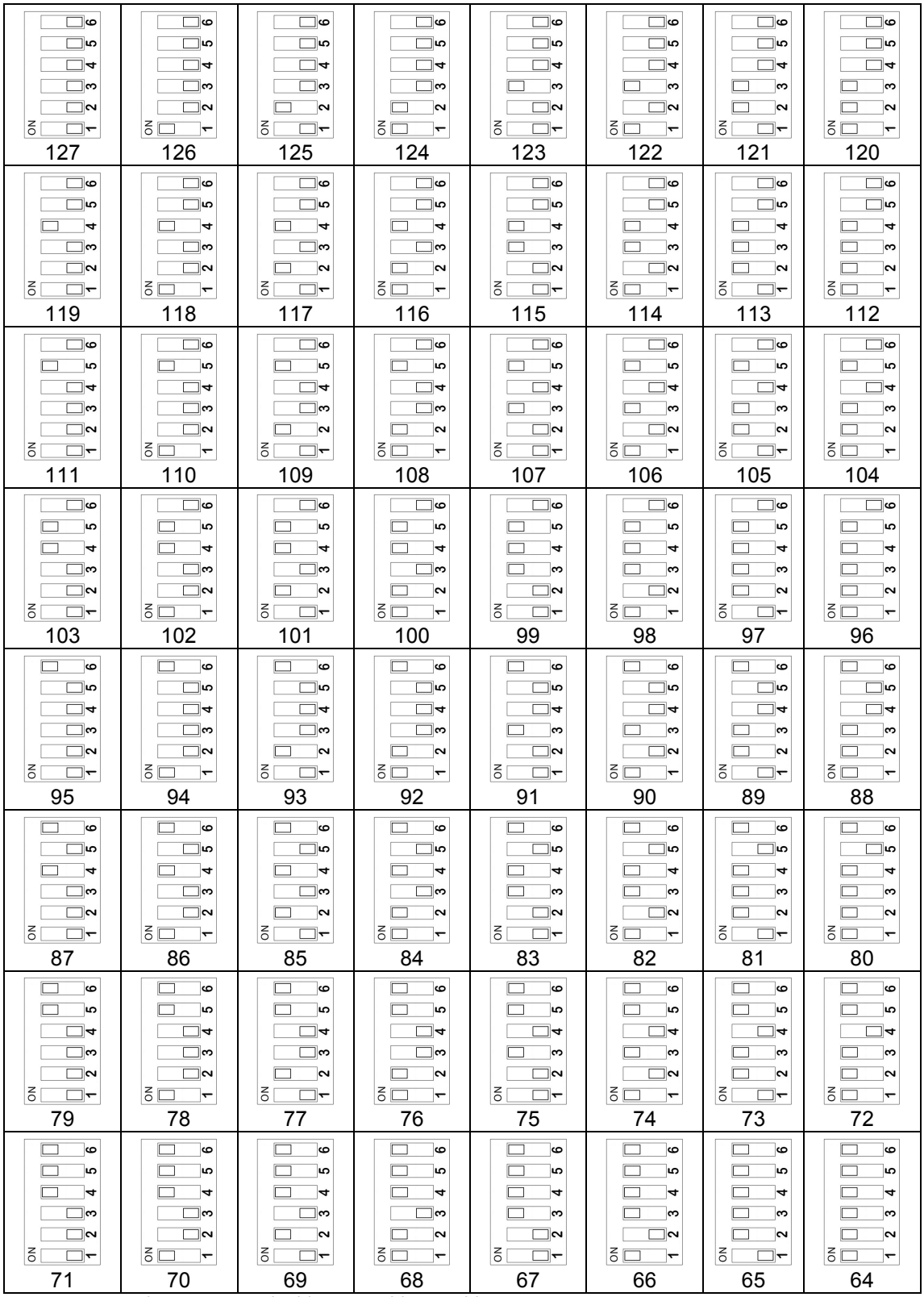


Image 13: Renbotics Servo Shield Rev 2 address table

Appendix A Sample Sketches

Sample 1: Simple servo sweeper

```
#include <Wire.h>
#include <ServoShield2.h>

ServoShield2 servos = ServoShield2(127);//Address of 127, using 50Hz mode

void setup() {
  Serial.begin(9600);
  Serial.println("Initializing...");
  servos.start();

  for (int servo = 0; servo < 16; servo++)//Initialize all 16 servos
  {
    servos.setbounds(servo, 1000, 2000);  //Set the minimum and maximum pulse duration
    servos.setposition(servo, 1500);      //Set the initial position of the servo
  }
  Serial.println("Init Done");
}

void sweep()
{
  Serial.println("Sweeping");

  for(int pos = 1000; pos < 2000; pos += 20)  //Move the servos from 0 degrees to 180
  {
    for (int i = 0; i < 16; i++)
      servos.setposition(i, pos);
     delay(1);
  }

  for(int pos = 2000; pos >= 1000; pos -= 20) //Move the servos from 180 degrees to 0
  {
    for (int i = 0; i < 16; i++)
      servos.setposition(i, pos);
    delay(1);
  }
}

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