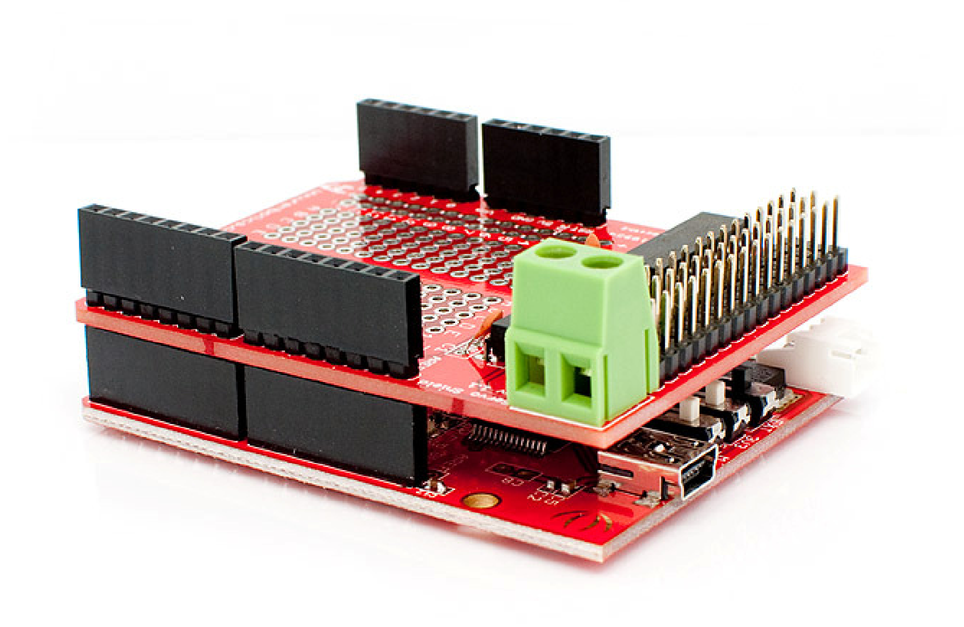
Renbotics Servo Shield



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 and Arduino Mega
- Easy to use API

Contents

Tables and Images

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 is an Arduino-compatible shield that uses two 4017 decade counters to drive up to 16 servos using only 4 pins (digital pins 6 to 9) and as little as one 8bit timer (Timer 2) in standard mode or two 16/8bit timers (Timer 1 and Timer 2 for Duemilanove or Timer 3 for Mega) in high accuracy mode. It also includes a 196 point breadboard style prototyping area.

4. Overview

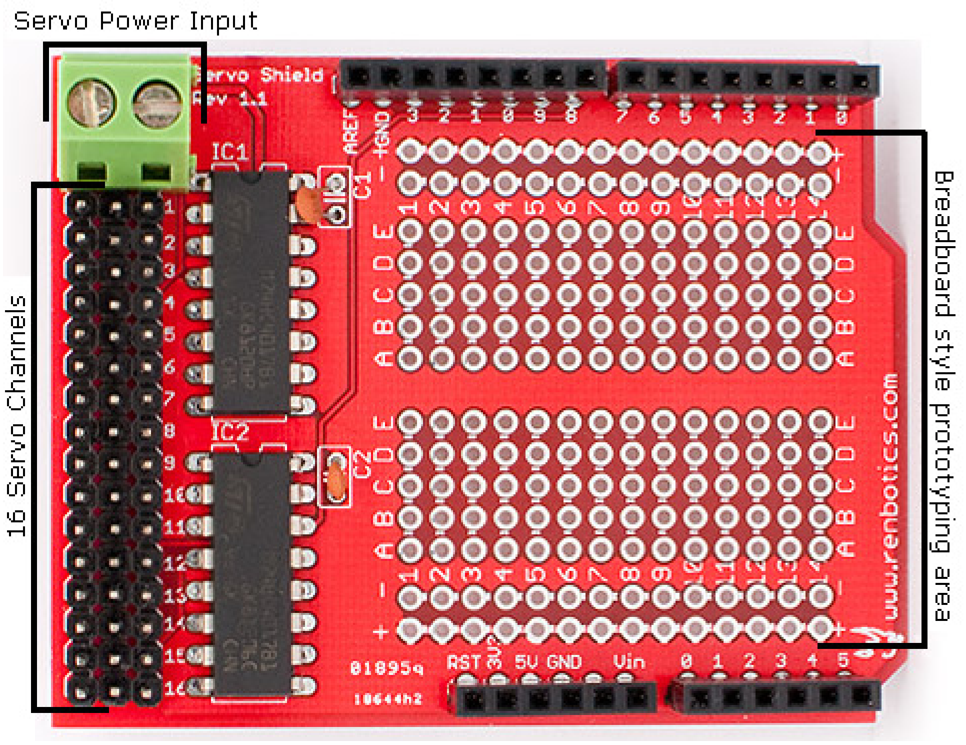


Image 1: Renbotics Servo Shield Overview

5. Features

- 16 Servo Channels
- Convenient screw terminal for servo power supply
- 196 Point breadboard style prototyping area
- Compatible with Arduino Duemilanove and Arduino Mega
- Easy to use API

6. Applications

- Robotics
- Animatronics
- Mechatronic Art

7. Parts List

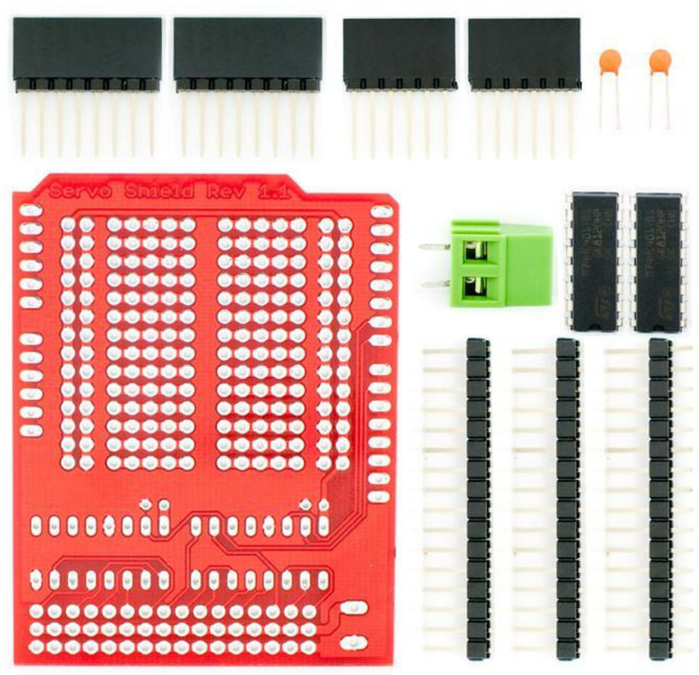


Image 2:Renbotics Servo Shield Parts

2 x 4017 Decade Counter DIP16
2 x 10nf Capacitors
2 x 6 pin Female Shield Stacking Headers
2 x 8 pin Female Shield Stacking Headers
1 x 2 pin Screw Terminal
3 x 16 pin Male Breakaway Headers

8. Assembly

Follow these 5 simple steps to assemble your Renbotics Servo Shield:

1. Solder the two supplied 10nF capacitors.

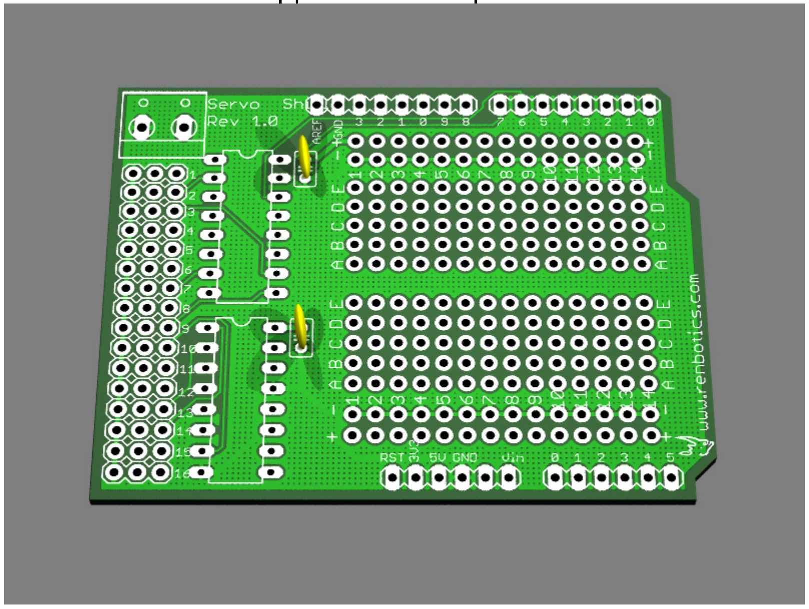


Image 3: Assembly Step 1

2. Solder the two supplied 4017 IC's.

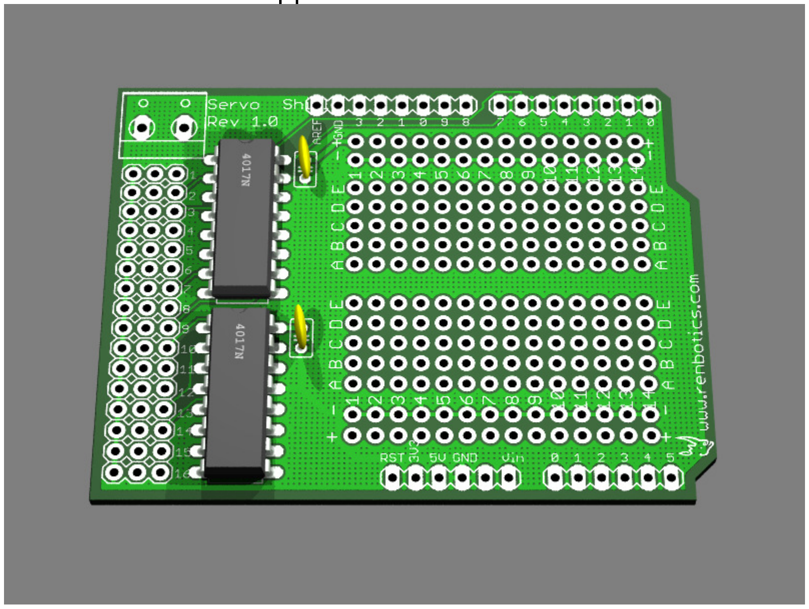


Image 4: Assembly Step 2

3. Solder the supplied servo headers.

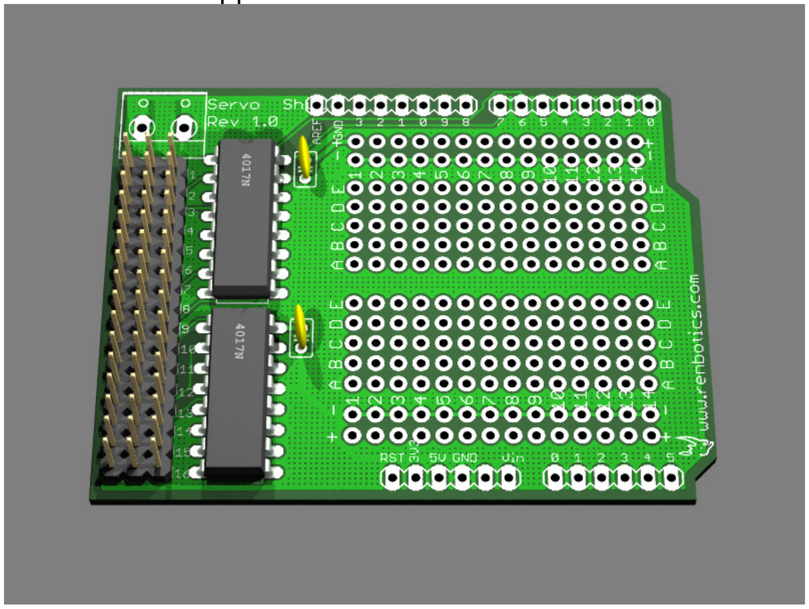


Image 5: Assembly Step 3

4. Solder the supplied stacking headers.

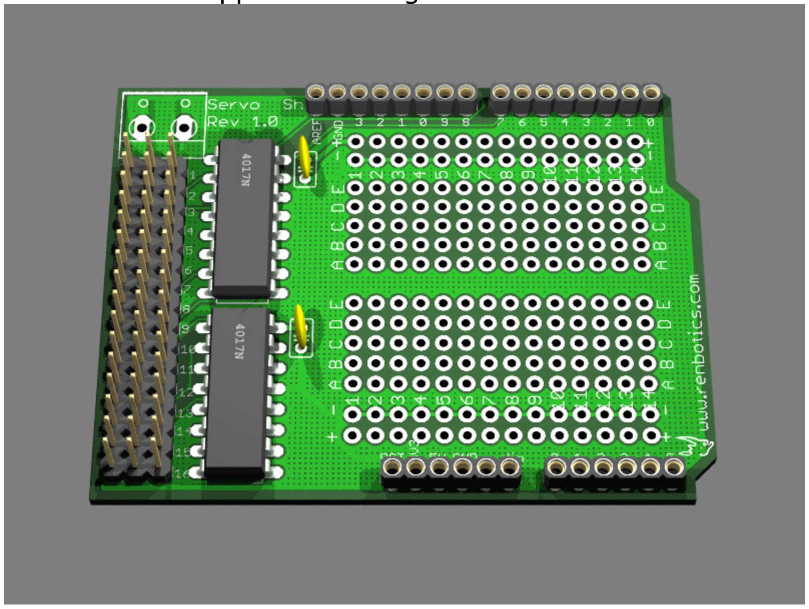


Image 6: Assembly Step 4

5. Solder the supplied screw terminal.

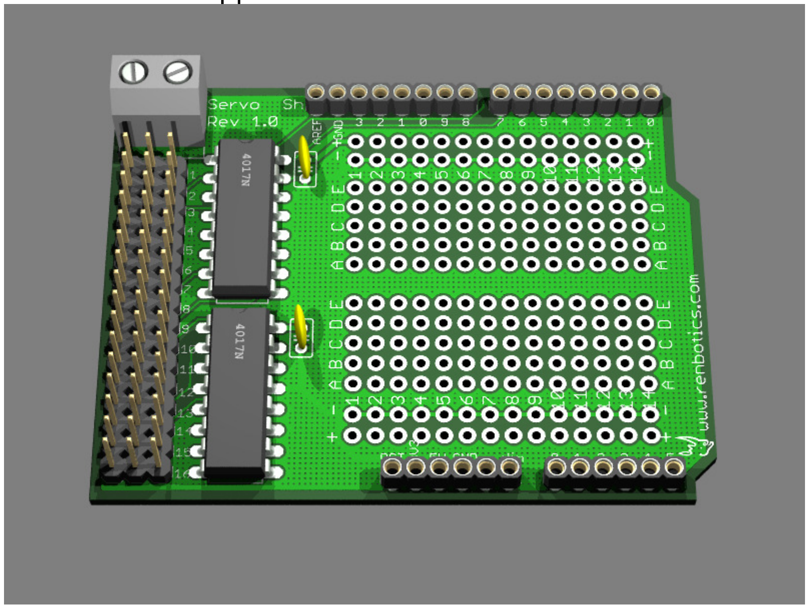


Image 7: Assembly Step 5

Your Renbotics Servo Shield is now ready to be used.

9. 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). 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).

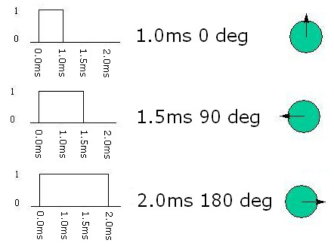


Image 8: Servo Control Overview

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.

The Servo Shield has two modes of operation; Standard and High Accuracy. In Standard mode the Servo Shield can move all 16 servos in 25us increments allowing for a resolution of 4.5deg per increment. Standard is supported on the Duemilanove (ATMega232 based) and Mega (ATMega128 based).

In High Accuracy mode the Servo Shield can move servos 1-9 on the Duemilanove and 1 – 16 on the Mega in 1us increments allowing for a resolution of 0.18deg per increment. On the Duemilanove servos 10 – 16 can only be operated in Standard mode.

10. Library

The Servo Shield Library is based on work done by Larry Barello in R/C pulse output unit based upon a 74HC4017 decade counter [1]. The latest libraries for the Renbotics Servo Shield are available at: http://renbotics.com/servoshield.zip

Functions

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

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

`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

`int getposition(int Servo);`

Returns the current position of the specified servo.

`int start();`

Starts the servo controller; returns 1 if instruction failed.

`int stop();`

Stops the servo controller; returns 1 if instruction failed.

Enabling High Accuracy Mode

To enable High Accuracy Mode simply edit the following file:

arduino-00XX\hardware\libraries\ServoShield\ServoShield.h

and change

```
//#define HIGHACCURACY
```

to

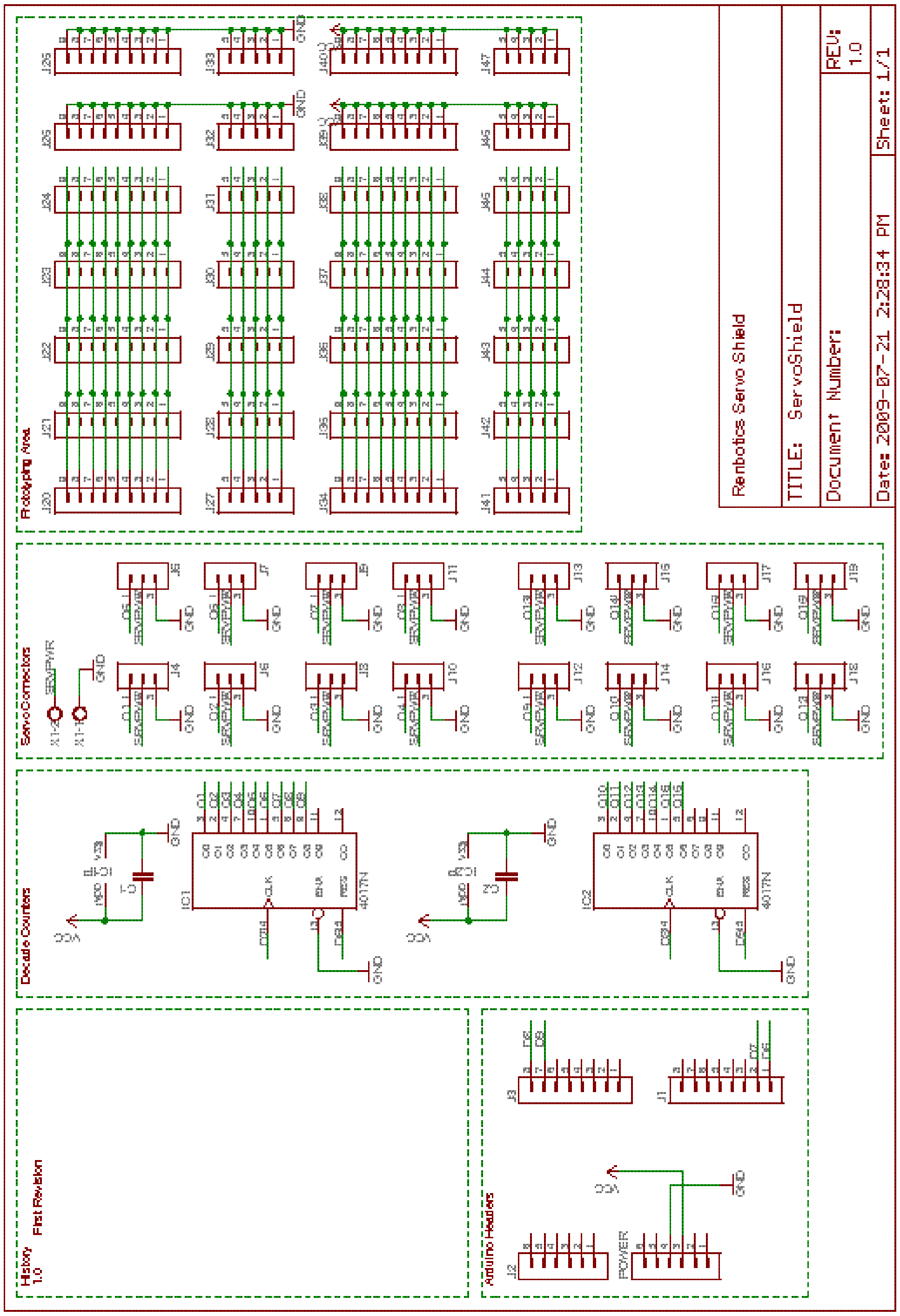
```
#define HIGHACCURACY
```

Appendix A Sample Sketches

Sample 1: Simple servo sweeper

```
#include <ServoShield.h>

ServoShield servos;                         //Create a ServoShield object

void setup()
{
  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
  }

  servos.start();                         //Start the servo shield
}

void loop()
{
  for(int pos = 1000; pos < 2000; pos++) //Move the servos from 0 degrees to 180 degrees
  {                                      //in steps of 1 degree
    for (int i = 0; i < 16; i++)         //for all 16 servos
      servos.setposition(i, pos);        //Tell servo to go to position in variable 'pos'

    delay(1);
  }

  for(int pos = 2000; pos >= 1000; pos--)//Move the servos from 180 degrees to 0 degrees
  {
    for (int i = 0; i < 16; i++)         //all 16 servos
      servos.setposition(i, pos);        //Tell servo to go to position in variable 'pos'

    delay(1);
  }
}
```

Appendix B Schematic



Appendix C References

[1] Larry Barello, *R/C pulse output unit based upon a 74HC4017 decade counter*, http://www.barello.net/Papers/AVR%20RC%20output.pdf