

16-channel Servo Controller

Technical Manual Rev 1r0

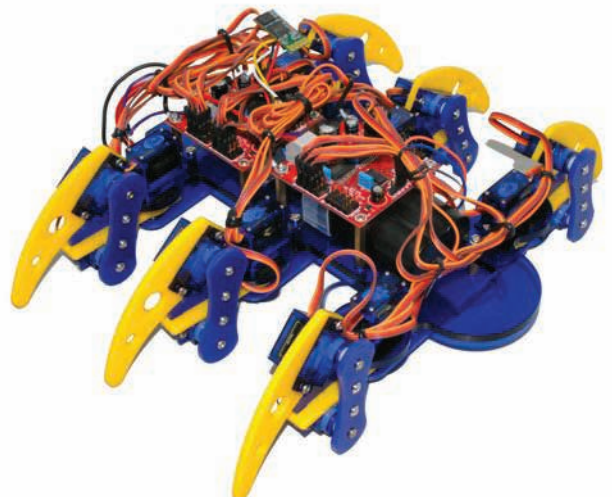
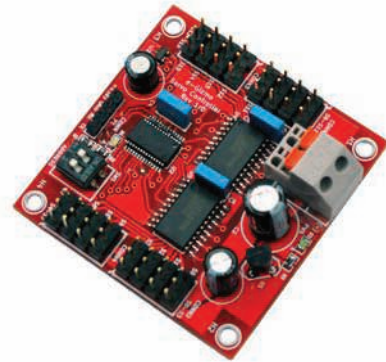
Hobby Servo motors has long gotten its own foothold in educational robotics applications. It is easy to see why. They offer an attractive alternative in place of stepping motors. They are cheaper, lighter, more efficient, and reasonably precise.

Controlling a hobby servo is not hard either. All the user code needs to do is to generate a PWM pulse corresponding to a desired servo shaft rotation and the servo will obligingly go. But there is this one caveat. Controlling a single or couple of servos on a microcontroller host busy with other task is possible and still is probably easy. But a herd of servo motor each fighting for host mcu attention may present a serious problem. They can quickly hog the host controller and leaving it little time to do other tasks. Overall system performance will suffer.

Using e-Gizmo 16 channel servo controller frees you host controller from this drudgery. E-Gizmo 16 channel servo controller is a dedicated controller capable of hosting up to 16 servos. Using simple positioning commands, users can precisely control the motion of up to 16 servos, independent from each other, all with relative ease.

Features

- 16 independent PWM servo control output
- 500-2500uS Pulse Width, 1uS timing resolution
- 48-50Hz refresh rate
- Serial (UART) communications port
- User configurable baud rate
 - 9600 bps (Default)
 - 19200 bps
 - 115200 bps
- Addressable, up to 4 boards (4x16 servos) can be controlled using a single UART port



PIN DESCRIPTION

Table 1: Power Input P2

| Pin | ID | Description |
|-----|-----|-----------------------------|
| 1 | (+) | + Power Input (6.0 -7.5VDC) |
| 2 | (-) | - Power Input (0V) |

Table 2: Interface Port P1

| Pin | ID | Description |
|-----|------|------------------------|
| 1 | +3v3 | +3v3 OUT, 50mA maximum |
| 2 | GND | Ground (0V) |
| 3 | RX | UART RX input |
| 4 | TX | UART TX output |

Table 3: Servo Motor Terminal CONN2 (Note 1)

| Pin | ID | Description |
|-----|----|--------------------------|
| 1 | S0 | Servo 0, OUT0 - V+ - GND |
| 2 | S1 | Servo 1, OUT1 - V+ - GND |
| 3 | S2 | Servo 2, OUT2 - V+ - GND |
| 4 | S3 | Servo 3, OUT3 - V+ - GND |

Table 4: Servo Motor Terminal CONN4 (Note 1)

| Pin | ID | Description |
|-----|----|--------------------------|
| 1 | S4 | Servo 4, OUT4 - V+ - GND |
| 2 | S5 | Servo 5, OUT5 - V+ - GND |
| 3 | S6 | Servo 6, OUT6 - V+ - GND |
| 4 | S7 | Servo 7, OUT7 - V+ - GND |

Table 5: Servo Motor Terminal CONN1 (Note 1)

| Pin | ID | Description |
|-----|-----|----------------------------|
| 1 | S8 | Servo 8, GND - V+ - OUT8 |
| 2 | S9 | Servo 9, GND - V+ - OUT9 |
| 3 | S10 | Servo 10, GND - V+ - OUT10 |
| 4 | S11 | Servo 11, GND - V+ - OUT11 |

Table 6: Servo Motor Terminal CONN3 (Note 1)

| Pin | ID | Description |
|-----|-----|----------------------------|
| 1 | S12 | Servo 12, GND - V+ - OUT12 |
| 2 | S13 | Servo 13, GND - V+ - OUT13 |
| 3 | S14 | Servo 14, GND - V+ - OUT14 |
| 4 | S15 | Servo 15, GND - V+ - OUT15 |

Table 7: LED Indicators

| ID | Description |
|----|---|
| D2 | Power Indicator. Illuminated when power is present |
| D1 | Comm indicator. Flashes when communication is in progress |

Note 1: Terminal arrangement as viewed from the PCB component side, power input connector at the top side.

APPLICATION WIRING EXAMPLES

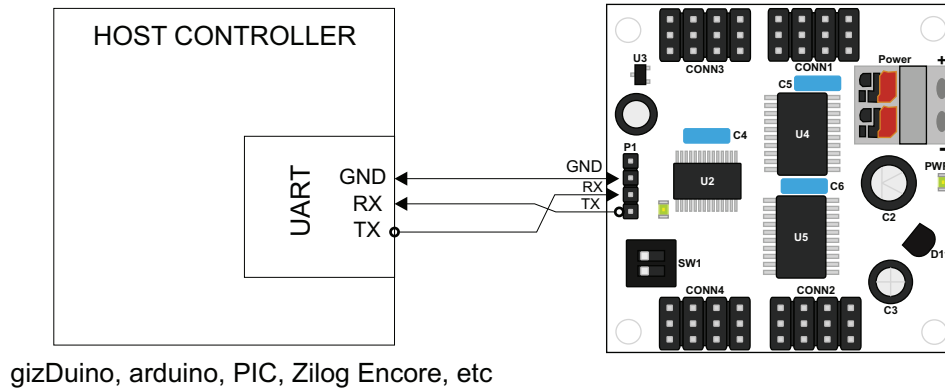


Figure 2. The Servo Controller interfaces with a host controller via a UART port. The host controller can be any microcontroller chip or board that has a UART port; hardware or simulated.

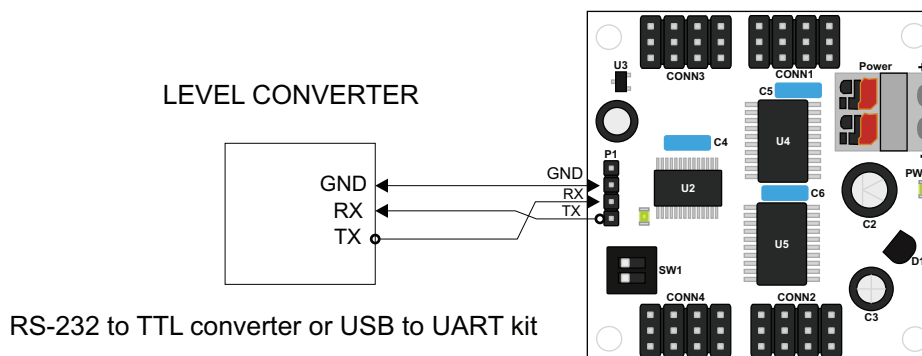


Figure 3. A PC can be used as a host controller, where you can program the motion sequence using your favorite software development kit SDK, such as the free Visual Basic Express running on Windows platform. A level converter kit (RS-232 to UART TTL) is required to connect it to a PC COM port. Alternatively, you can connect it using a PC USB port with the use of a USB to UART converter.

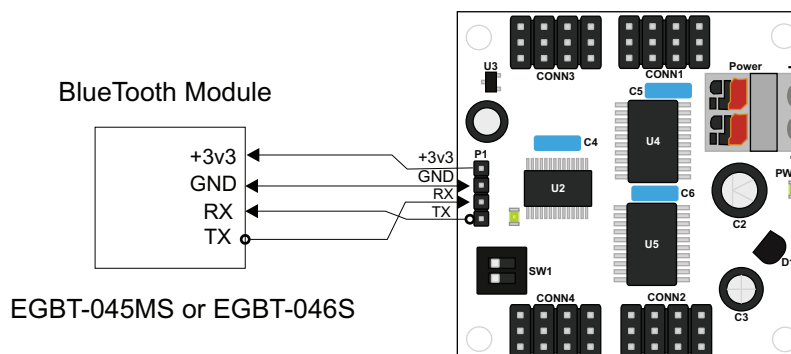


Figure 4. Go wireless! You can even connect it for wireless PC control using Bluetooth modules. Power to the Bluetooth module is provided by the on-board 3v3 supply of the servo controller.

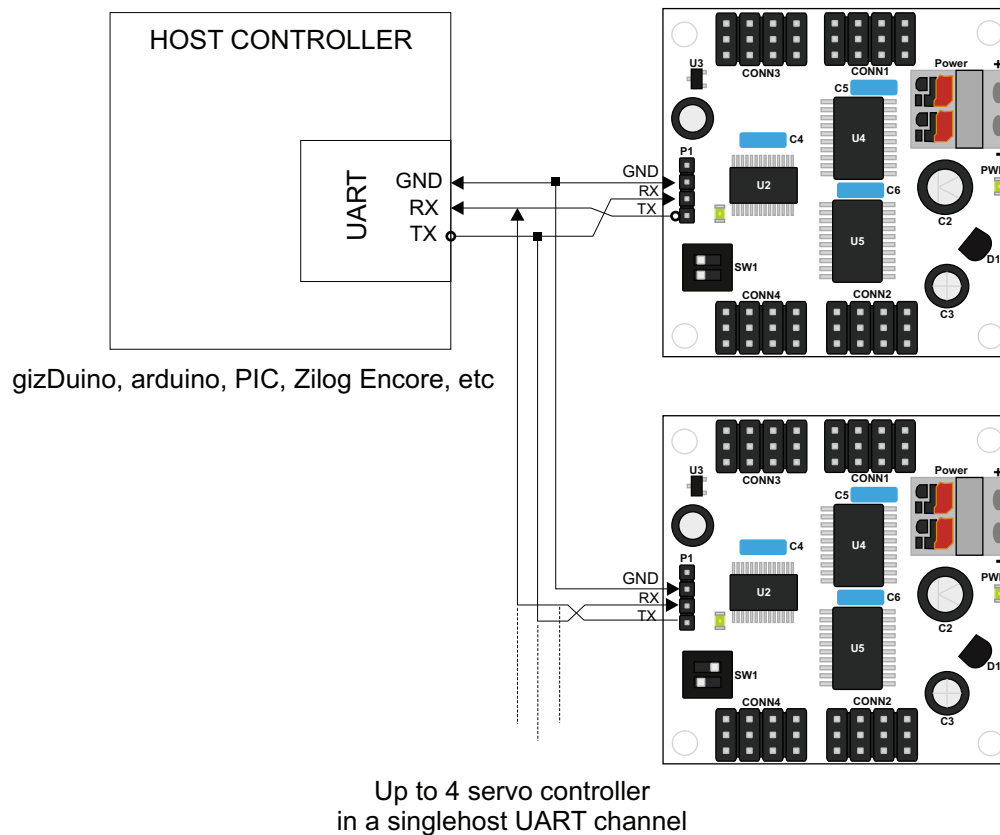


Figure 5. You can connect up to four 16-channel Servo Controller in a single UART host port, gaining effective control to up to 64 servo motors. The address switch must be correctly configured to properly control individual servo motors.

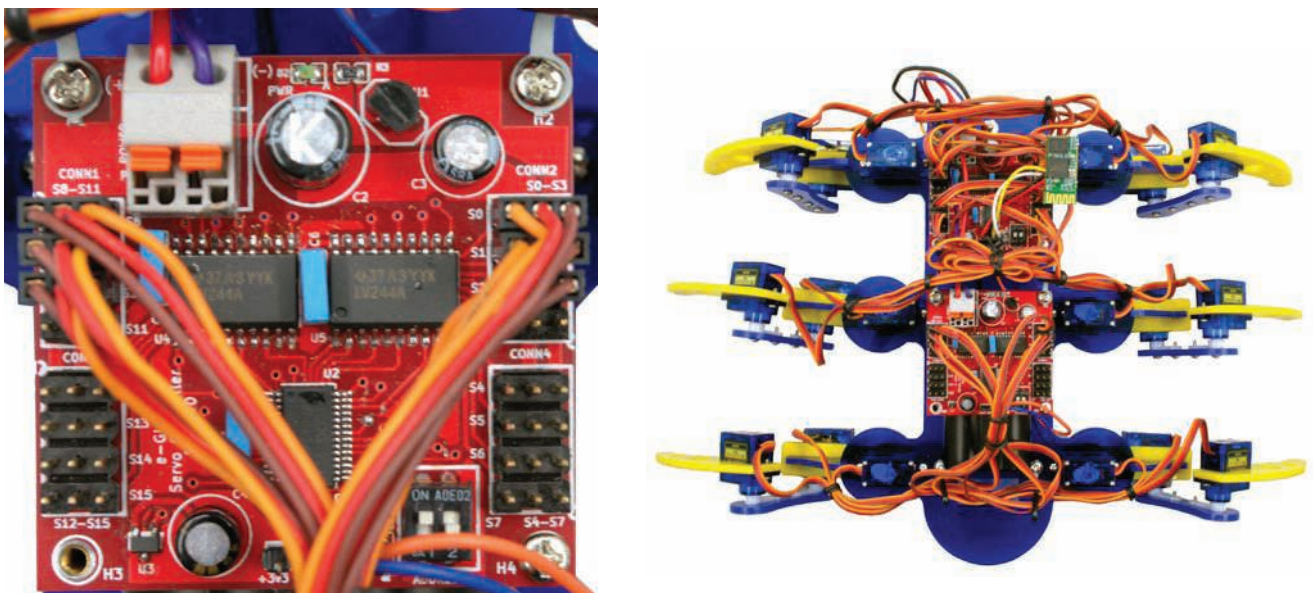


Figure 6. A six legged robot using a couple of Servo Controller. This robot has 18 servo motors, requiring the use of two Servo Controllers. This demonstration robot is remotely controlled by a PC via a Bluetooth link.

COMMUNICATIONS

Communications settings:

Baud Rate: 9600 (Default)
Start Bit : 1
Stop Bit : 1
Parity: None
Data: 8 bits
H/S : None

Note: UART port is 3.3V logic, 5V tolerant TTL port.

Where:

STX : Start of packet marker ASCII code = 0x02
MOTNUM : Motor ID Number in ASCII format
QUERY : Single character query request
ETX : End of packet marker ASCII code = 0x03

DATA PACKET FORMAT:

General Positioning

STX + MOTNUM , PULSEWIDTH, RATE + ETX

Where:

STX - Start of packet marker ASCII code = 0x02
MOTNUM - Motor ID Number in ASCII format
PULSEWIDTH - Pulse Width uS in ASCII format
RATE - Rate of displacement in ASCII format
ETX - End of packet marker ASCII code = 0x03

Query

STX + MOTNUM,QUERY + ETX

GENERAL POSITIONING

STX + MOTNUM , PULSEWIDTH, RATE + ETX

RATE: PWM Value will be incremented by this value every 1/50 sec until the current PWM value reaches the target PWM value. Use RATE to control the slewing speed of the servo motor.

PULSEWIDTH: is the target PWM pulse width (position) in uS. The controller will accept and generate any values between 500uS to 2500uS with 1uS resolution. Specifying PWM target values less than 500uS will cause the corresponding PWM output to disable. You can purposely disable a selected PWM output this way.

Positioning Example:

To send a 1500uS pulse to motor 13, the controller must send to the servo driver the following packet:

[STX]13,1500,10[ETX]

Represented in binary transmission format as:

| | | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|------|
| 0x02 | 0x31 | 0x33 | 0x2C | 0x31 | 0x35 | 0x30 | 0x30 | 0x2C | 0x31 | 0x30 | 0x03 |
| STX | 1 | 3 | , | 1 | 5 | 0 | 0 | , | 1 | 0 | ETX |

You can send more than one set of Motor positioning sequence in a packet.

Example:

[STX]5,750,10,6,1000,10,11,2423,5[ETX]

This packet will instruct motor number 5 to go to position 750uS, motor 6 to position 1000, and motor 11 to position 2423 at a rate of 10,10,5, respectively. Note that motor positioning command must always be sent in complete set of 3 numbers each, otherwise, you may get unexpected results. A packet must not exceed a total of 250 characters, inclusive of STX and ETX.

QUERY COMMANDS

STX + MOTNUM, QUERY + ETX

Controller Response
STX + RESPONSE + ETX

Only one query request per packet is allowed.

Motor ID numbering depends on addressing dip switch SW1 settings as described earlier in this documentation. The "S" query command treats all 16 servo motors in one board as a single motor group. The group is automatically resolved by selecting any one of the motor within the board/group.

S – Check if any motor in one group/board is still inching towards its target position. MOTNUM can be any motor ID number within a group.

Response :

- 0 - if one or more motor is/are still inching towards their respective target position.
- 1 - if all motors in the addressed board are all in position.

s – If the specified motor MOTNUM still inching towards its target position

Response:

- 0 – MOTNUM is still moving
- 1 – MOTNUM in position

V – Controller firmware version

Response: Firmware Version

R – Turn OFF all PWM Outputs

H – Set baud rate to high speed (115200bps)

M – Set baud rate to medium speed (19200bps)

L – Set baud rate to default low speed (9600bps)

Example:

Sending
[STX]2,S[ETX]

Controller response:

[STX]0[ETX] if any motor within the group (motor 0-15) is still inching

[STX]1[ETX] if all motors within the group are all in position.

Important note:

The controller reports motor position based on the current PWM value generated for each corresponding motor. There is no way for the controller to know if the servo motor is physically in position. Hence, S or s query is valid only for PWM rate of change that does not exceed the motor slewing speed.