RC SERVO EXAMPLE WITH BAE0910

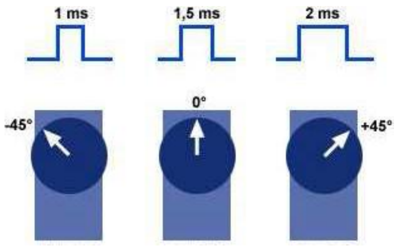
AN APPLICATION NOTE ON HOW TO SETUP AND CONTROL AN RC-SERVO FROM AN 1-WIRE NEWORK

INTRODUCTION

RC servos are hobbyist devices that are used to provide actuation.

Typically employed in radio-controlled models, their affordability and reliability has extended their use in small-scale robotics applications.

Most servos are operated at 4.8 V to 7.2 V DC.



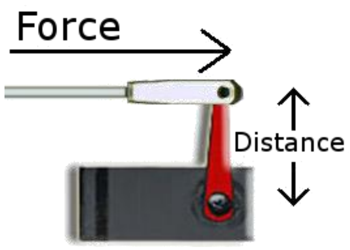
Servos are controlled by a pwm signal (pulse width modulation) at TTL level. It is a pulse repeated periodically. The width of the pulse defines the shaft position.

The pulse frequency is not critical, only the pulse width precision is important.
A pulse of 1.5 ms width will set the servo to its "neutral" position.
All servo's accept a pulse frequency of 50Hz but some servo's accepts frequency up to 400Hz to increase reactivity.

The TTL level needed to control the servo doesn't require specific interfacing components and allow direct connection to microcontrollers.

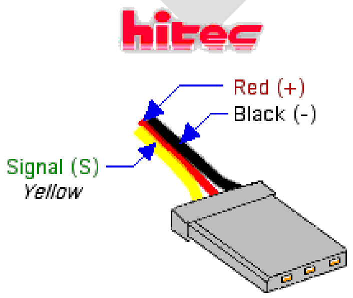
Physical characteristics vary from brands and models.
Most servos have a travel angle from 90° to 210°, some models allows multi-turns and other offer continuous rotation. Also there are several size and torque to fit any needs.
Examples:

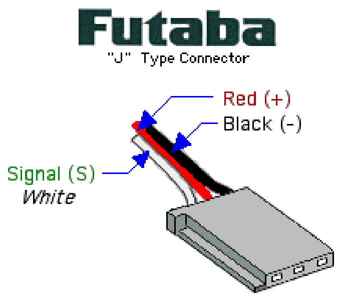
	Futaba S3003 standard	Hitec HSR-1425CR Continuous Rotation	Hitec HS-45HB micro	Futaba S3306 High-Torque
Torque	3,1Kg cm	3,3 Kg cm	1.1Kg cm	24Kg cm
Speed	0.22s / 60°	16 rpm	0.14s /60°	0.16s/60°
Weight	37 gr	45 gr	8 gr	125gr
Dimensions	40x20x36 mm	40x20x36 mm	23x10x22 mm	66x30x43 mm



A torque of 3kg cm means that the servo arm is capable of developing a 3Kg force at 1cm distance from the center, if the distance is doubled; the force is divided by two.

Historically, there were various servo connectors; however a universal connector is now becoming prevalent as shown below:





The center pin is the +5V supply and external pins are GND and signal . With this arrangement, plugging in a servo backwards won't hurt anything, it simply won't work.

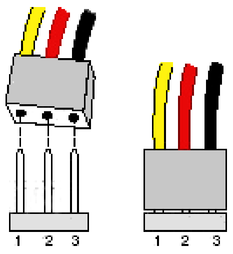
☞Always check the pin arrangement documented with your specific servo before plugging it.

RC SERVO CONNECTION ON BAE0910 BOARDS

The control signal needed by RC servo is easily produced by the PWM function of BAE0910 chip which is able to control up to four RC servo motors independently.

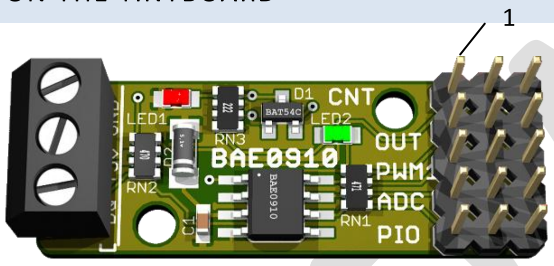
PWM1 and PWM2 (ADC) are the preferred pin to control the servos as they offer the best precision due to the hardware generated pwm signals.
If more servos have to be connected, OUT and PIO could also be used via the software PWM3 and PWM4 respectively.

In the following examples the servo connector are shown with the following convention:



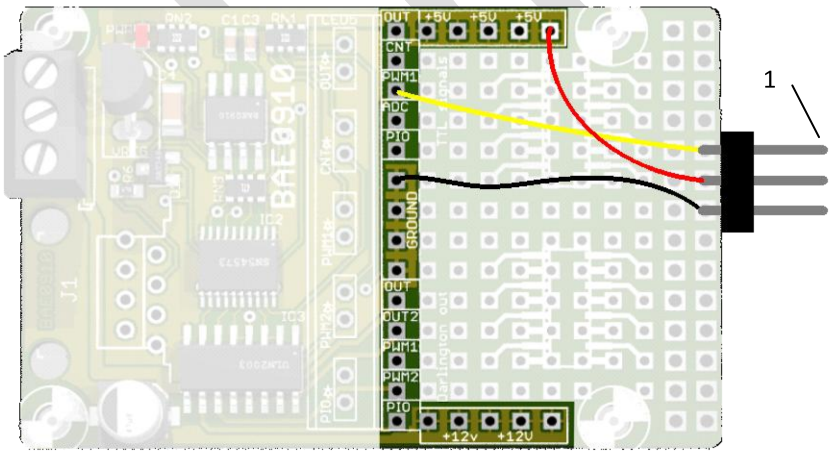
1	PWM output
2	5V
3	GND

ON THE TINYBOARD



On the BAE0910 tinyboard, the servos are directly connectible to the pin headers.
The OUT pin could also be used as software PWM3 but may require an external pull-up resistor with the tinyboard rev 1.01. (rev1.02 already contains the pull-up resistor)
Be sure to provide enough power on the +5V screw connector.

ON THE PROTOTYPING BOARD



On the BAE0910 prototyping board, a 3 pin 0.1" spaced male connector has to be soldered. No other components are needed. All four OUT,PWM1,ADC,PIO could be used from the TTL signals of the prototyping area to controls up to four servo's.
Be sure to provide enough 5V current to support demanding application with multiple servos. If the 5V provided by the onboard voltage regulator is not enough, you may need to install on the wiring area a dedicated 5Volts VREG powered from the 12V pad.

USING A SERVO WITH BAE0910

Controlling a servo from 1-Wire with BAE-0910 has never been easier. All pwm timing are internally generated by the chip. Configure the PWM mode then assign a setpoint in the DUTYn register and the 1-Wire slave chip will continuously produce the pwm signal. To change servo angle, simply vary the corresponding DUTY register value from 1000µs to 2000µs. Depending on servo model, the range for DUTY could be larger (500µs to 2500µs)

As long as the servo receive a pwm signal it does its best to maintain the assigned position; you will find a resistance if you manually try to rotate it in either direction.
To turn off the servo, set DUTY to zero. This will stop the pwm signal. In this mode servo position can be changed manually, but offer some resistance due to the high reduction ratio used internally.

CONFIGURATION FOR A SERVO CONNECTED ON PWM1

Configure the registers as follows:

TPM1C=4C	// 1MHz clock, (1 duty unit = 1µs)
PERIOD1=20000	// period of 20000µs = 50Hz
DUTY1=1500	// duration of the pulse = 1500µs (=neutral position for servo)

Controlling the servo:

Set DUTY1 to a value in the range from 1000 to 2000 to turn servo accordingly.
Depending on servo model, the range for DUTY could be larger (700 to 2400)

CONFIGURATION FOR A SERVO CONNECTED ON PWM2 (ADC)

Configure the registers as follows:

TPM2C=4	// 1MHz clock, (1 duty unit = 1µs)
PERIOD2=20000	// period of 20000µs = 50Hz
DUTY2=1500	// duration of the pulse = 1500µs (=neutral position for servo)
ADCC=0	// select PWM2 instead of ADC

Controlling the servo:

Set DUTY2 to a value in the range from 1000 to 2000 to turn servo accordingly.
Depending on servo model, the range for DUTY could be larger (700 to 2400)

CONFIGURATION FOR A SERVO CONNECTED ON SOFTWARE PWM3 (OUT)

Configure the registers as follows:

TPM1C=4	// 1MHz clock, (1 duty unit = 1µs)
PERIOD1=20000	// period of 20000µs = 50Hz
DUTY3=1500	// duration of the pulse = 1500µs (=neutral position for servo)
OUTC=0	// select PWM3 instead of OUT function

Controlling the servo:

Set DUTY3 to a value in the range from 1000 to 2000 to turn servo accordingly.
Depending on servo model, the range for DUTY could be larger (700 to 2400)

CONFIGURATION FOR A SERVO CONNECTED ON SOFTWARE PWM4 (PIO)

Configure the registers as follows:

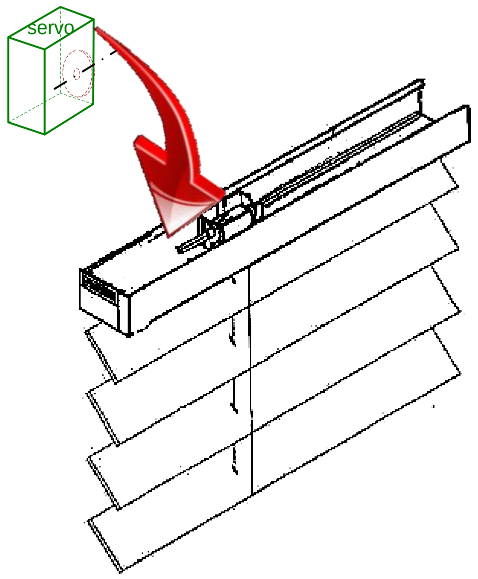
TPM2C=4	// 1MHz clock, (1 duty unit = 1µs)
PERIOD2=20000	// period of 20000µs = 50Hz
DUTY4=1500	// duration of the pulse = 1500µs (=neutral position for servo)
PIOC=0	// select PWM4 instead of PIO function

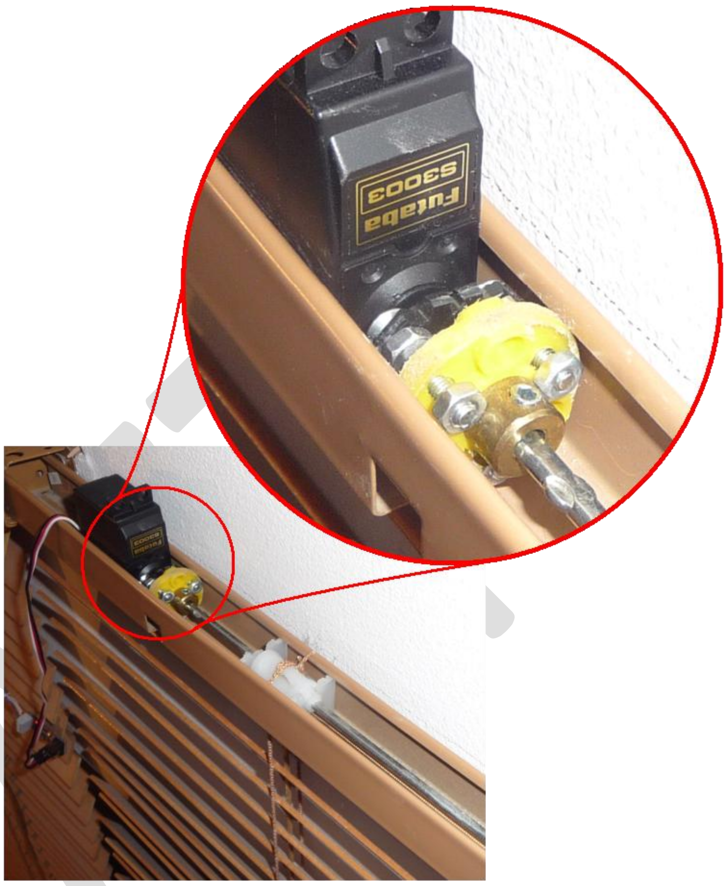
Controlling the servo:

Set DUTY4 to a value in the range from 1000 to 2000 to turn servo accordingly.
Depending on servo model, the range for DUTY could be larger (700 to 2400)

EXAMPLE: SERVO CONTROLLED BLINDS

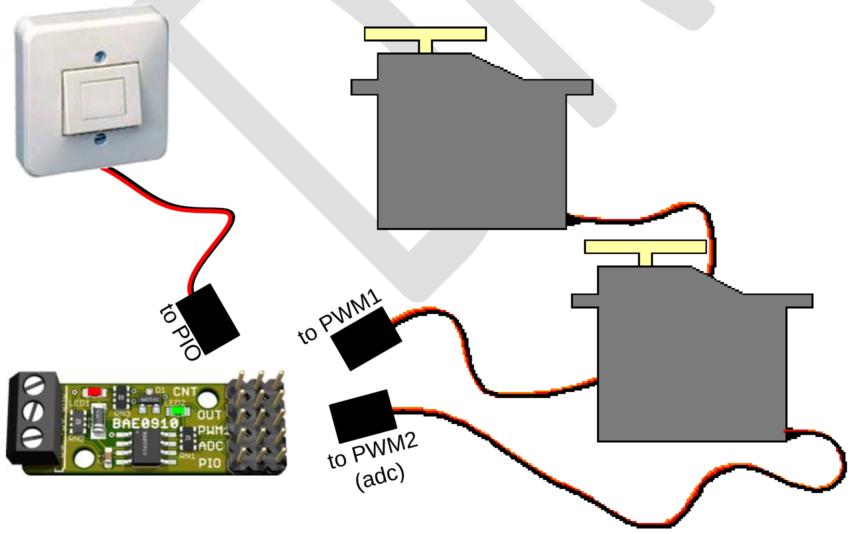
RC Servo motors are ideal actuators to transform a standard Venetian blind to a fully automated system.





Controlling the blind from your linux is cool but there is a major drawback:
As the manual control is removed, your wife will not appreciate to log on the system to turn the blades on desired position!

The solution is quite simple: add a push button connected to a one of the available inputs and store a small AE code in the non volatile eeprom memory of the bae board.



AE CODE FOR LOCAL CONTROL OF BLINDS

```
// AutomationEngine assembly source.
// Used to control two venetian blinds from both linux/crontab and local pushbutton
// right blind SERVO connected PWM1
// left blind SERVO connected PWM2
// push button is connected on PIO
// VARIABLES:
// B_USERA       to open the blinds, linux store 1 to usera and 2 to close them
//               once actuation is terminated, usera is cleared automatically
// B_USERB       retain current state, to check current state, read userb
#define BLIND_OPEN       2200    //servo position for open
#define BLIND_CLOSE      600             //servo position fo closed
#include "bae0910.inc"

#eeprom 0, 0     // start_page, end_page
#org  $00        // Code generated at address specified.

begin:
        NOP     // Allow auto start of process #0 on poweron, first byte of page0 should be a NOP
init:

        SET.B   B_TPM1C,4        // Pre Scaler (1us resolution)
        WAIT    1
        SET.W   W_PERIOD1,20000
        WAIT    1
        SET.B   B_TPM2C,4        // Pre Scaler (1us resolution)
        WAIT    1
        SET.W   W_PERIOD2,20000
        WAIT    1
        SET.B   B_PIOC,18 // enable pio as input with pullup
        START   1,check_button

main:  //the main loop wait a request by checking value on usera
        WAIT    1
        CMP.B   B_USERA,1
        BEQ             open
        CMP.B   B_USERA,2
        BEQ             close
        BRA             main
open:
        SET.W   W_DUTY1,BLIND_OPEN   //actuation code to open
        WAIT    1
        SET.W   W_DUTY2,BLIND_OPEN
        BRA     done
close:
        SET.W   W_DUTY1,BLIND_CLOSE      //actuation code to close
        WAIT    1
        SET.W   W_DUTY2,BLIND_CLOSE
        BRA     done
done:
        WAIT    32                       // allows two seconds actuation before de-energizing the servo's
        CLR.W   W_DUTY1
        WAIT    1
        CLR.W   W_DUTY2
        SET.B   B_USERB,B_USERA  //retain blind state
        CLR.B   B_USERA          // clear last request
        BRA     main             // and goto main loop to wait next request

check_button:  // process that check the local button, only one button to simply invert state on each click.
        WAIT    1
        CMP.B   B_PIO,1
        BEQ     check_button
invert_state:  //button pressed, then change blind orientation
        CMP.B   B_USERB,1
        BEQ     turn_close
turn_open:
        SET.B   B_USERA,1
        BRA     done1
turn_close:
        SET.B   B_USERA,2
        BRA     done1
done1:
        WAIT    64
        BRA     check_button
```

INSTALLATION STEPS ON YOUR OWFS BASED SYSTEM

Download the compiled file "blind-control.bin" from download area:

```
wget  http://www.brain4home.eu/attachments/blind-control.bin
```

erase the eeprom of your destination device:

```
echo 1 >/path/to/your/owfs/FC.000000000nnn/eeprom/erase.0
```

install AE program into the eeprom:

```
cp blind-control.bin /path/to/your/owfs/FC.000000000nnn/eeprom/page.0
```

power cycle the bae board or start manually the AE code with:

```
echo 1 >/path/to/your/owfs/FC.000000000nnn/910/pc0
```

At this point, your servo's will change position on every press on the pushbutton connected on PIO connector.

To control the blind position from your Linux:

Open blinds:

```
echo 1 >/ path/to/your/owfs/FC.000000000nnn/910/usera
```

Close blinds:

```
echo 2 > /path/to/your/owfs/FC.000000000nnn/910/usera
```

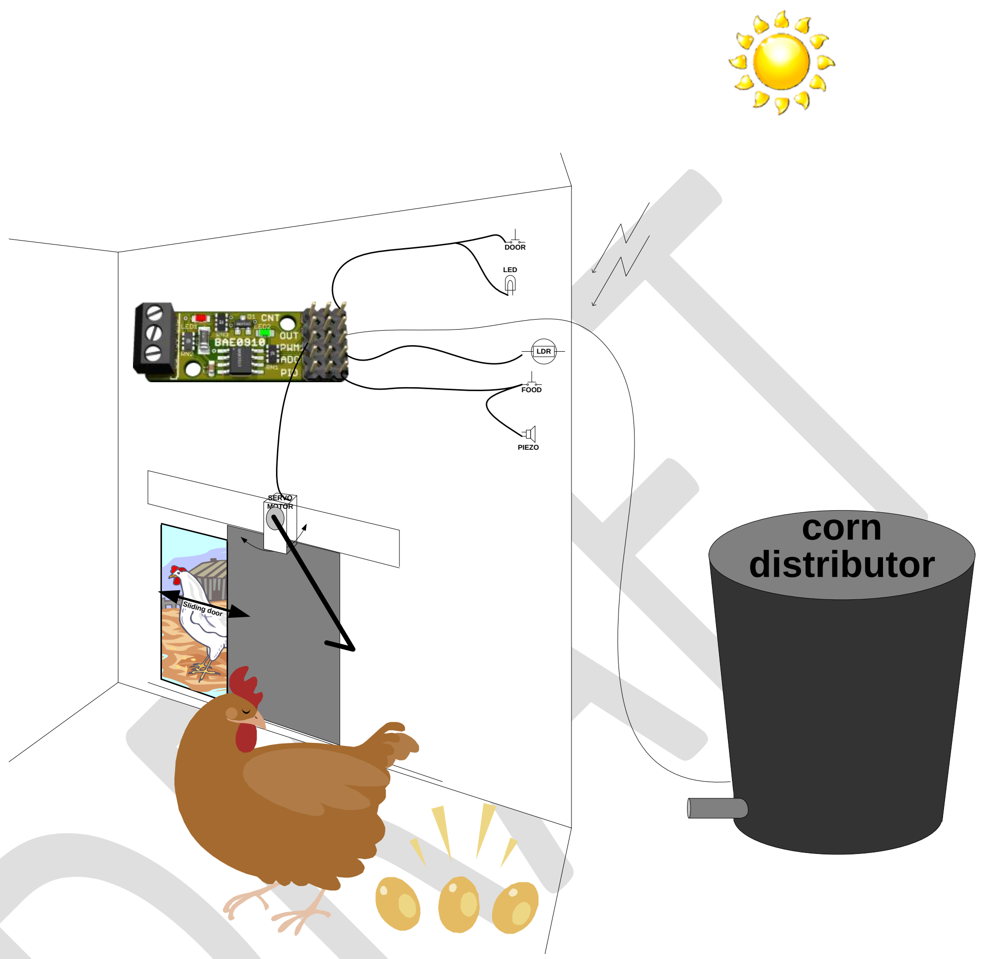
To know current blind state:

```
cat  /path/to/your/owfs/FC.000000000nnn/910/userb
```

(1 is when open, 2 is when closed)

You may also use crontab to schedule blind opening/closing.

ANOTHER EXAMPLE



See online for coming AN04 : light controlled door system for chicken house

SUPPORT

Online support is available via the forum on www.brain4home.eu and via the discussion list.
To subscribe, list-subscribe@brain4home.eu

AVAILABILITY

Chips and boards can be ordered online on www.brain4home.eu

CONDITION OF USE

The BAE chips are intended for hobbyist usage and are not approved for use where it constitute or may constitute a danger to human life or health.

TERMS OF LICENSE

The software embedded in the chips is protected by copyright laws. Customer is not allowed to reverse engineer, decompile, or disassemble the embedded software.

ABOUT THE AUTHOR

Pascal Baerten is primarily an IT consultant with technical background in automation. He followed A2 technical studies until 1985 where he played with CNC machines and pneumatic automates. Graduated in Computer Sciences from the Robert Shuman High school in Belgium in 1989, his thesis was titled "A terminal emulator" where he mastered serial communication and networking programming.

His first computer was a Sinclair ZX81, where he learned the basics of exploiting very constrained computing resources in assembler. Later, a Commodore 64 opened the way to interfacing computers with electronic toys.

Since 1990 he developed network based resource sharing solutions in assembler and C.: Telex server, Minitel server, mainframe front end, mail server, print server, text2speech telephone server, database gateway, IM server …

As skilled networking/server architect, he is working as IT consultant for large financial companies since 1997.

In parallel, developments in home automation have contributed to accumulate some experience with microcontrollers and embedded computing.

REVISION HISTORY

Revision #	Date	Description
0.1	May 13, 2010	Initial draft
0.11	May 15,2010	Typo corrections

BRAIN4HOME
HOME AUTOMATION