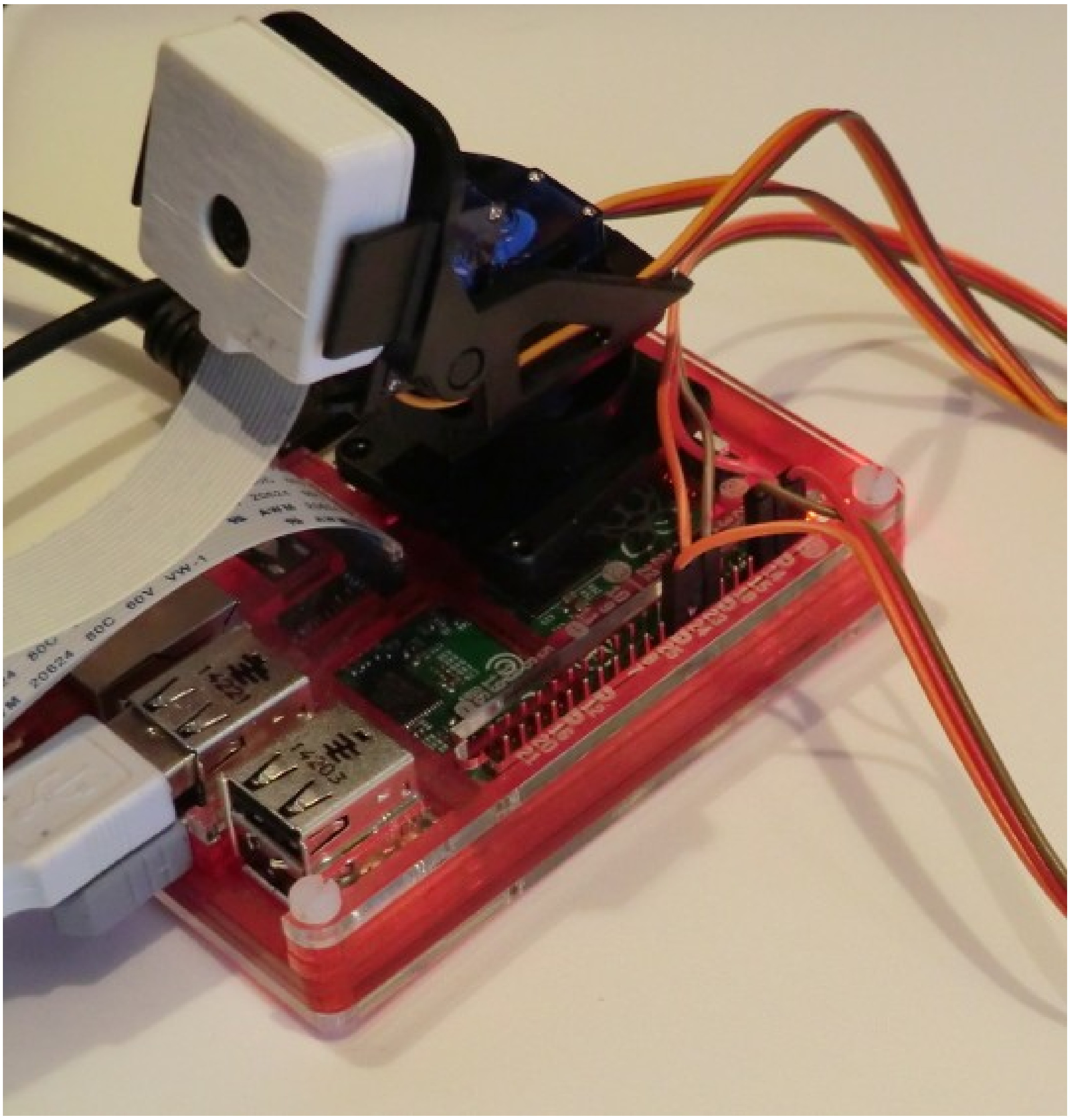
Servo How-to page.

A pan-tilt bracket controlled by two servo motors will allow you to move your Raspberry Pi camera module left/right and up/down, either locally or via the internet over SSH.

We attached a Pan & Tilt bracket with two SG90 servos to GPIO pins 23 & 24 on a Raspberry Pi.



The black plastic bracket shown is available assembled & tested with two SG90 servo motors fitted for around £6 on eBay including UK delivery. http://www.ebay.co.uk/itm/291491236376

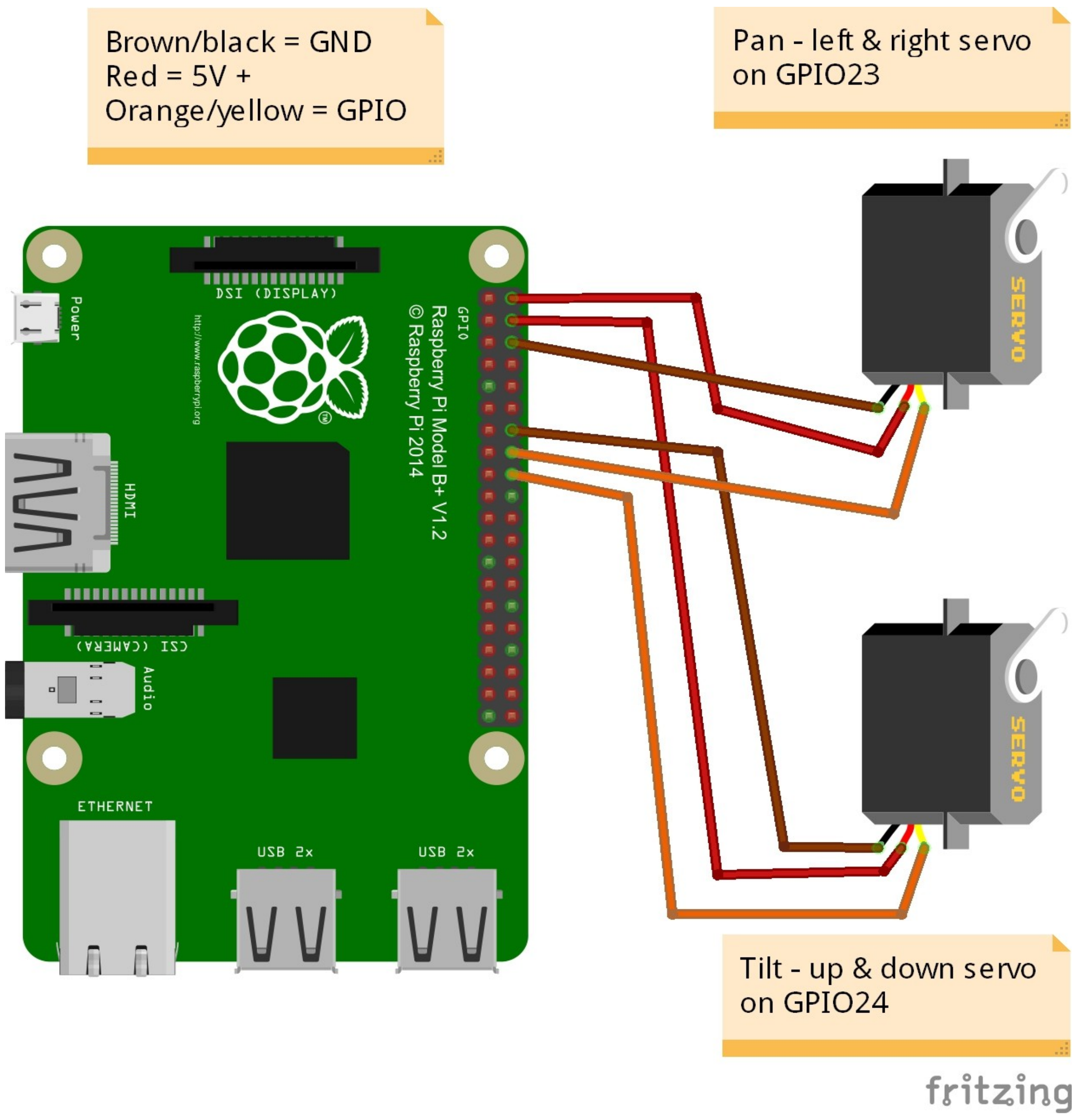
You can also buy the bracket and servos separately and assemble it yourself. Here's a good video guide: https://www.youtube.com/watch?v=kGZxuC0i6zQ

Wiring Diagram

Colour of wires on the servos connected to a B+ Pi.

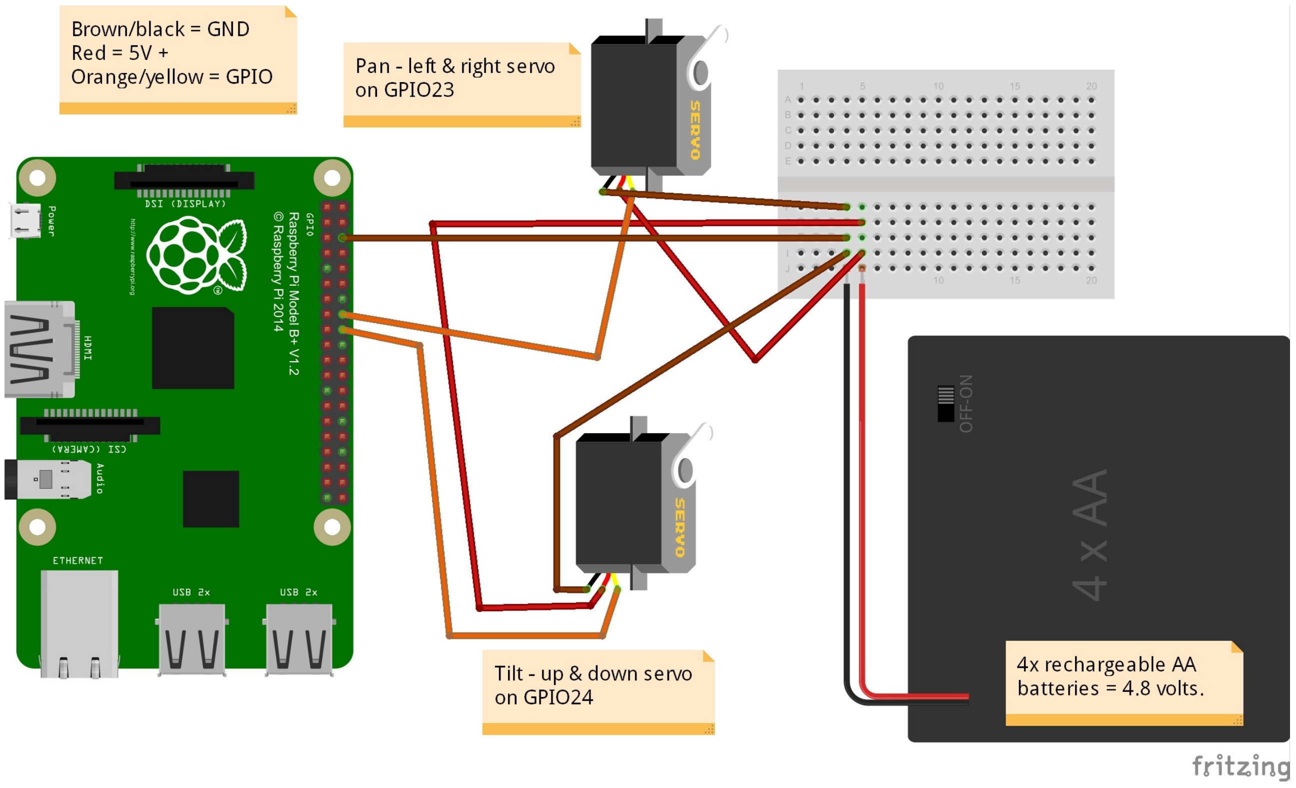
Red = 5V+
Brown = GND
Orange = GPIO23 (Pan - left & right) or GPIO24 (Tilt - up and down)

We attached the bottom Pan servo to GPIO23 & the top Tilt servo to GPIO24 to the Raspberry Pi using some orange, red & brown male-to-female DuPont breadboard cables from eBay.



This wiring arrangement worked fine for us on newer Raspberry Pi A+ and B+ models with a 2amp USB power adapter. If the Pi reboots/freezes, then you'll need to power the Servos from batteries.

Use this wiring arrangement if the previous wiring diagram didn't work reliably for you.



This is the more conventional way to wire to wire servos up to a Pi, but I only discovered it after I'd already connected the SG90 servo power directly to the Pi B+ & A+ models, and that had worked for me without problems.

If you want to completely remove the risk of damaging your Pi, then use the separate battery pack wiring diagram.

The Pi A+, B+ and Pi 2 have a more sophisticated power circuit than the previous models, so if you're running an older B board you may well find you need to wire the servos power to a separate battery pack as shown above.

In the diagram above, the orange GPIO wires from the servo connect to the Pi. The brown GND cables are connected to the battery & the Pi. The red 5v+ wires from the servo are only connected to the battery.

Software

We're using the open-source ServoBlaster code to control the servos from the BASH command line. We install it like this:

```
wget https://raw.githubusercontent.com/richardghirst/PiBits/master/ServoBlaster/user/servod.c
wget https://raw.githubusercontent.com/richardghirst/PiBits/master/ServoBlaster/user/Makefile
wget https://raw.githubusercontent.com/richardghirst/PiBits/master/ServoBlaster/user/init-script
```

```
make servod
sudo ./servod
```

Then test it's working with:

```
echo 5=50% > /dev/servoblaster
echo 5=10% > /dev/servoblaster
echo 5=90% > /dev/servoblaster
```

The commands above move servo number 5 (Pan left & right) on GPIO23 between centre (50%), left (10%) & right (90%) positions. The Tilt servo would be servo number 6 on GPIO24.

If you hear the servos juddering after they've stopped moving then you've moved them out of normal range. So for instance, moving servo 5 to 5% is too far for us, whereas 10% is still okay. This is why I've used 10% and 90% in my examples. You want to set the servos to 50% before assembling and fixing the pan & tilt bracket, if assembling it yourself.

You can check if servod is running with the command

```
ps ax
```

and close it with

```
sudo killall servod
```

Related Links

http://wiki.panotools.org/Panorama_scripting_in_a_nutshell
https://github.com/richardghirst/PiBits/tree/master/ServoBlaster
https://www.youtube.com/watch?v=kGZxuC0i6zQ

Shop links (lots of useful Raspberry Pi & Arduino stuff by us)

Getting more advanced.

While the examples shown on the previous pages work just fine, the default install of ServoBlaster does grab 8 of your GPIO pins, when we only needed two of them for our pan/tilt camera bracket. If you want to use other pins for regular non-servo work it's worth disabling those extra 6 pins. Here's how you do that:

```
nano servod.c
```

page down five times & change the line that says :

```
static char *default_p1_pins = "7,11,12,13,15,16,18,22";
```

to

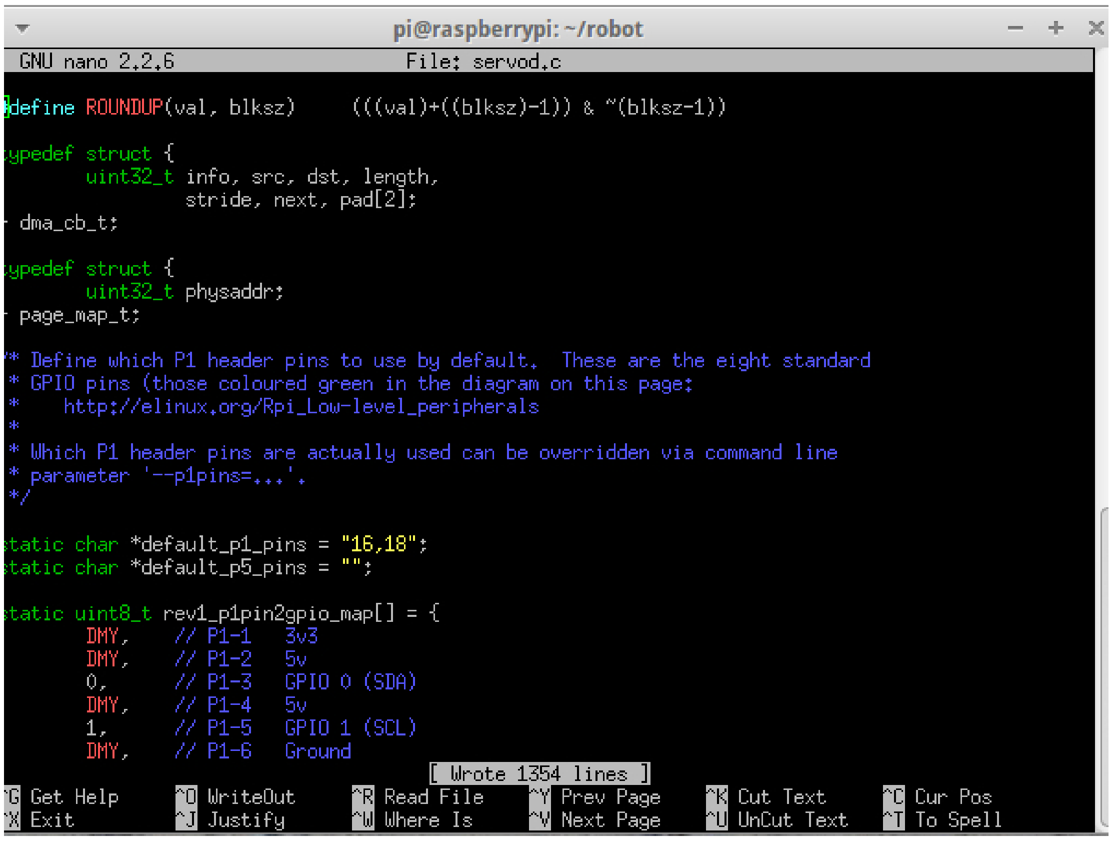
```
static char *default_p1_pins = "16,18";
```

And then recompile the servod command with

```
make servod
sudo ./servod
```

Now your servos are on number 0 & 1, because we're only using 2 GPIO pins for servos rather than 8. So to move the Pan servo (left & right), do this:

```
echo 0=50% > /dev/servoblaster
echo 0=10% > /dev/servoblaster
echo 0=90% > /dev/servoblaster
```

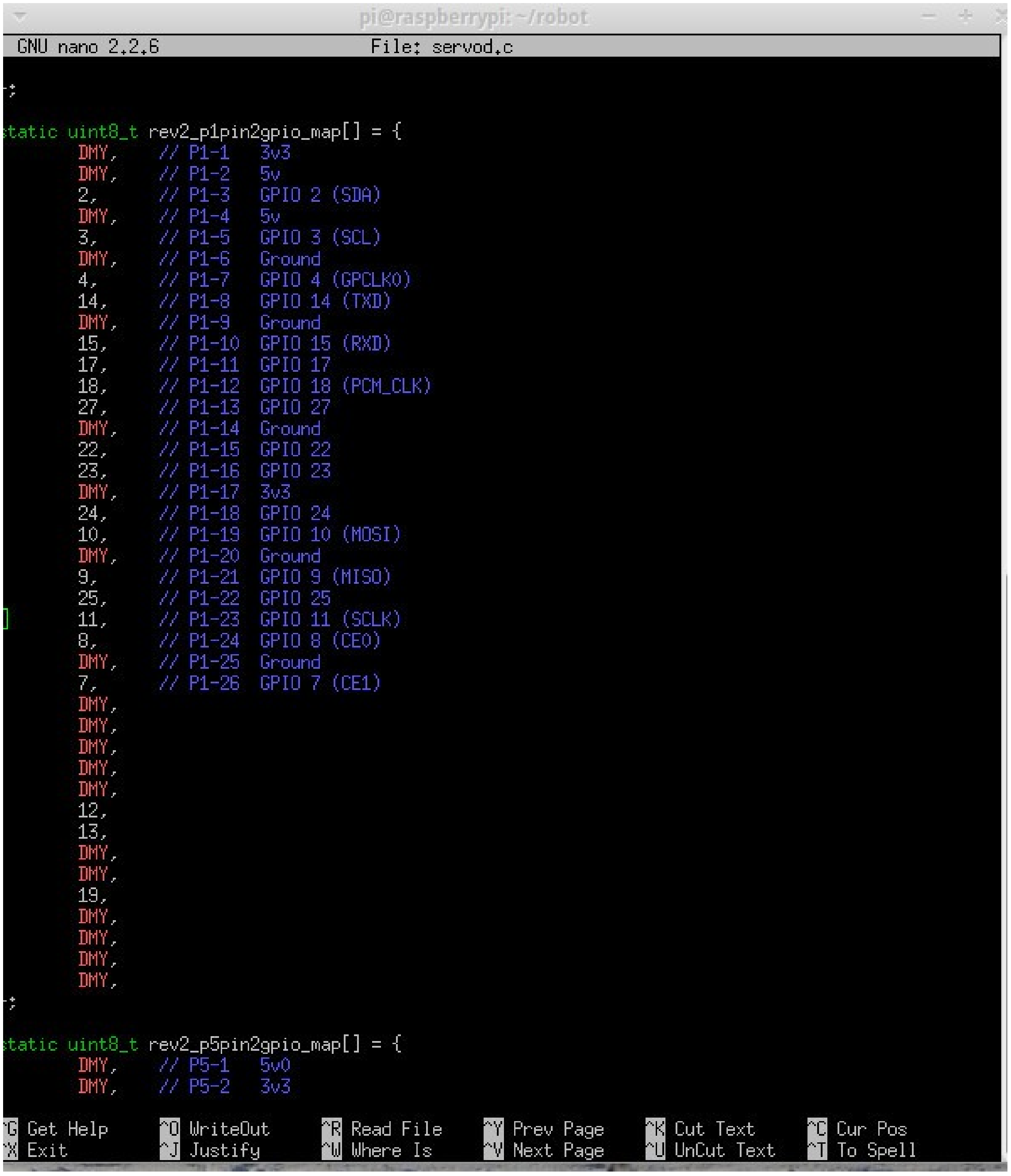


```
pi@raspberrypi: ~/robot
 GNU nano 2.2.6                    File: servod.c

#define ROUNDUP(val, blksz)     (((val)+((blksz)-1)) & ~(blksz-1))

typedef struct {
        uint32_t info, src, dst, length,
                 stride, next, pad[2];
} dma_cb_t;

typedef struct {
        uint32_t physaddr;
} page_map_t;

/* Define which P1 header pins to use by default.  These are the eight standard
 * GPIO pins (those coloured green in the diagram on this page:
 *    http://elinux.org/Rpi_Low-level_peripherals
 *
 * Which P1 header pins are actually used can be overridden via command line
 * parameter '--p1pins=...'.
 */

static char *default_p1_pins = "16,18";
static char *default_p5_pins = "";

static uint8_t rev1_p1pin2gpio_map[] = {
        DMY,    // P1-1   3v3
        DMY,    // P1-2   5v
        0,      // P1-3   GPIO 0 (SDA)
        DMY,    // P1-4   5v
        1,      // P1-5   GPIO 1 (SCL)
        DMY,    // P1-6   Ground
                               [ Wrote 1354 lines ]
^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

Using the ServoBlaster library with Raspberry Pi A+, B+ & Pi2

When the ServoBlaster library was put together the default values allow for the 26 pin GPIO connector on the rev1 & rev2 version A and B Raspberry Pi boards. If you have the newer Raspberry Pi A+, B+ or Pi2 board with a 40 way GPIO connector, and you want to use those extra pins with ServoBlaster, then we need to edit the rev2 values in servod.c file before re-compiling.

This example allows you to use the GPIO pins 18, 12, 13 & 19 required for the TiddlyBot robot kit.



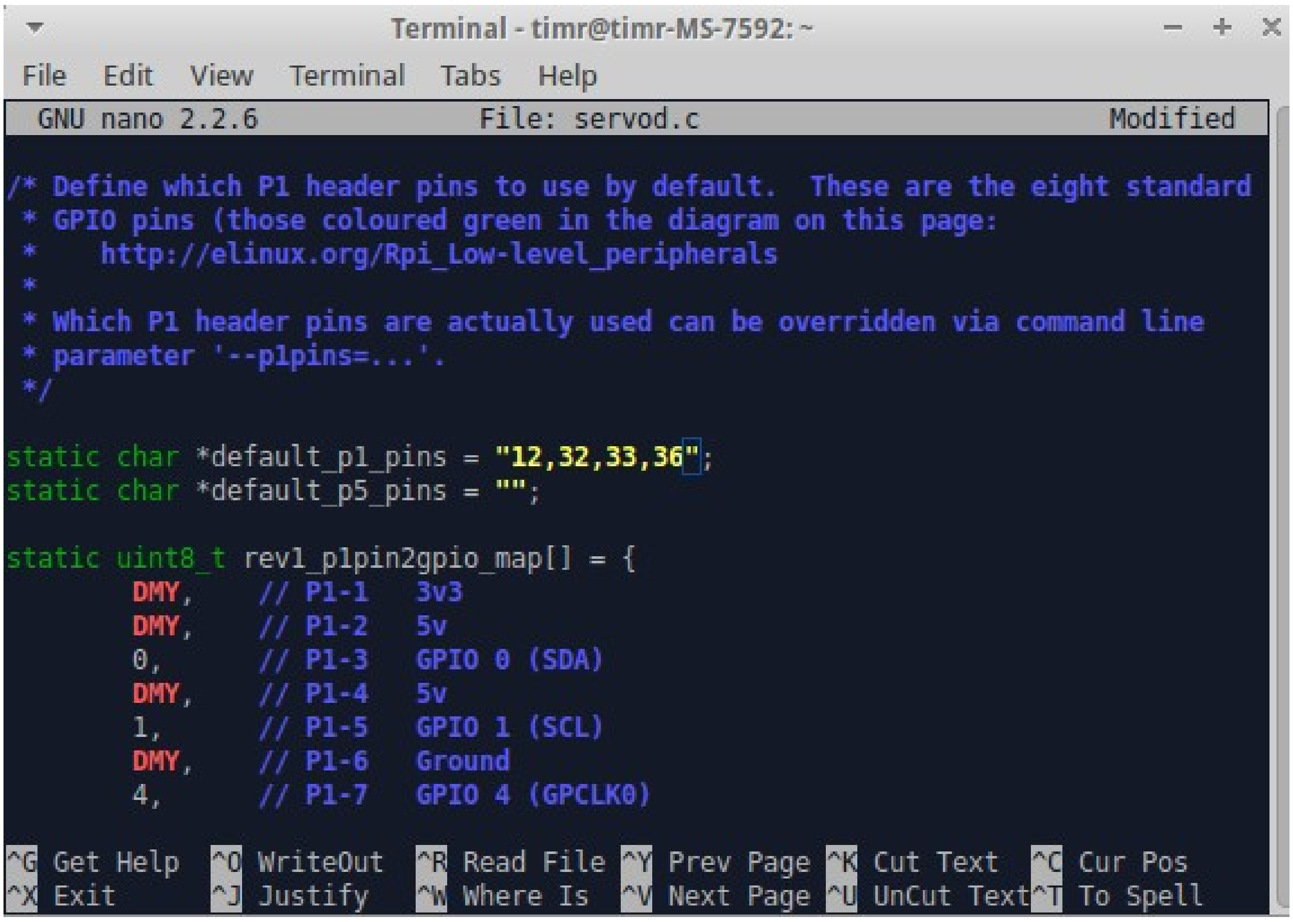
```
pi@raspberrypi: ~/robot
 GNU nano 2.2.6                    File: servod.c

:

tatic uint8_t rev2_p1pin2gpio_map[] = {
        DMY,    // P1-1   3v3
        DMY,    // P1-2   5v
        2,      // P1-3   GPIO 2 (SDA)
        DMY,    // P1-4   5v
        3,      // P1-5   GPIO 3 (SCL)
        DMY,    // P1-6   Ground
        4,      // P1-7   GPIO 4 (GPCLK0)
        14,     // P1-8   GPIO 14 (TXD)
        DMY,    // P1-9   Ground
        15,     // P1-10  GPIO 15 (RXD)
        17,     // P1-11  GPIO 17
        18,     // P1-12  GPIO 18 (PCM_CLK)
        27,     // P1-13  GPIO 27
        DMY,    // P1-14  Ground
        22,     // P1-15  GPIO 22
        23,     // P1-16  GPIO 23
        DMY,    // P1-17  3v3
        24,     // P1-18  GPIO 24
        10,     // P1-19  GPIO 10 (MOSI)
        DMY,    // P1-20  Ground
        9,      // P1-21  GPIO 9 (MISO)
        25,     // P1-22  GPIO 25
        11,     // P1-23  GPIO 11 (SCLK)
        8,      // P1-24  GPIO 8 (CE0)
        DMY,    // P1-25  Ground
        7,      // P1-26  GPIO 7 (CE1)
        DMY,
        DMY,
        DMY,
        DMY,
        DMY,
        12,
        13,
        DMY,
        DMY,
        19,
        DMY,
        DMY,
        DMY,
        DMY,
:

tatic uint8_t rev2_p5pin2gpio_map[] = {
        DMY,    // P5-1   5v0
        DMY,    // P5-2   3v3

G Get Help     ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
X Exit         ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

You also need to edit the line at the bottom of the first page that says:

```
#define NUM_P1PINS        26
```
to
```
#define NUM_P1PINS        40
```

Changes needed to work with TiddlyBot

Change the values to 12,32,33,36 which are GPIO pins 18,12,13,19 on the A+, B+ & Pi 2.


Terminal - timr@timr-MS-7592: ~
File Edit View Terminal Tabs Help

```
  GNU nano 2.2.6              File: servod.c                          Modified

/* Define which P1 header pins to use by default.  These are the eight standard
 * GPIO pins (those coloured green in the diagram on this page:
 *    http://elinux.org/Rpi_Low-level_peripherals
 *
 * Which P1 header pins are actually used can be overridden via command line
 * parameter '--p1pins=...'.
 */

static char *default_p1_pins = "12,32,33,36";
static char *default_p5_pins = "";

static uint8_t rev1_p1pin2gpio_map[] = {
        DMY,    // P1-1   3v3
        DMY,    // P1-2   5v
        0,      // P1-3   GPIO 0 (SDA)
        DMY,    // P1-4   5v
        1,      // P1-5   GPIO 1 (SCL)
        DMY,    // P1-6   Ground
        4,      // P1-7   GPIO 4 (GPCLK0)

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Driving Continuous Motor Servos

On the last page we mentioned the TiddlyBoy robot, which uses two Continuous Servo motors to directly drive the two wheels. While the previous examples showed you how to move the SG90 servos through their maximum 180 degree arc, the continuous servos move through the full 360 degree arc, like regular wheels.

If you send a 0% value to a Continuous Servo it goes backwards until you give it a different command, if you send 50% value it stays still and 100% value makes it turn forwards. So in the same way we could address the SG90 servos from the command line, we can make both the left & right wheels on a TiddlyBot, go back, forward or stop with nothing more complicated than:

```
echo 0=50% > /dev/servoblaster
echo 0=100% > /dev/servoblaster
echo 0=0% > /dev/servoblaster
```

and

```
echo 3=50% > /dev/servoblaster
echo 3=100% > /dev/servoblaster
echo 3=0% > /dev/servoblaster
```

Channels 1 & 2 control the other two servo channels on the TiddlyBot board. The camera tilt servo is normally on channel 1.