Enrique Olaizola (eao@sfu.ca)
Emanuel Pumnea (epumnea@sfu.ca)
Teyo Soler (tys2@sfu.ca)
Jesus Roberto Escamilla Yarahuan (jescamil@sfu.ca)

HOW TO: CONTROL DC MOTORS USING THE BEAGLEBONE BLACK'S GPIO's

If you were to try to find on-line guides to show you how to connect a motor driver and a DC motor to the BeagleBone Black, you will likely come up pretty frustrated. At the time of writing there many more guides geared towards less capable development boards such as Arduinos and RPis. As a result, we spent a great deal of time researching how to connect these components safely to the BeagleBone Black. One excellent source for general information on working with the BeagleBone Black is DerekMalloy.ie. However, even with his informative videos and sample code you will likely be left asking yourself how he figured out characterstic A or B about the how the BeagleBone works. We hope that this guide provide you, not only a set of steps to blindly follow to get your project working, but also answer how and where to find the necessary information; something that is often missing in many on-line tutorials.

Before we can start creating our circuit to control the motors from the BeagleBone Black (BBB), we first need to figure out whether any additional circuitry is required. For example, we may need to run our connections through resistors, diodes or transistors in order to make sure we don't supply excess current to the BBB's delicate GPIOs. Our first step then, is to research the specifications of both the BBB and the external components we will be connecting. Here is a list of the major components you will need to follow this guide:

1 – Polulo DC Motors
2 – Polulo Dual Motor Driver/Controller (Model#: TB6612FNG)
3 – 6V Battery pack
4 – 8 Resistors
5 – Two small bread boards (you can use one, just make sure it's large enough to contain the circuitry)
6 – Lots of jumper wires (or you can just purchase a 22-AWG wires, cut and strip as needed)

Finding the spec-sheet for your components

In our case, we will need to look up the specifications for both the motors and the motor driver. Since spec-sheets weren't provided with the hardware itself we will need to find the information on-line. Luckily, both the motors and motor drivers have a model number printed on stickers on either the component itself or the package they come in. The spec-sheet for the particular motor driver we are using in this guide can be found here:

http://www.pololu.com/file/0J86/TB6612FNG.pdf

We don't need everything listed on the spec-sheet. In particular, we are interested in how much the voltages the components are rated for as well as how much current they will sink (i.e. put out) and source (i.e. take in).Similarly, since we will ultimately be powering the motor controller from the BBB, we will need to ensure that the BBB is capable of providing the necessary voltage. Consequently, we will need to reference the following documentation for the BBB:

Enrique Olaizola (eao@sfu.ca)
Emanuel Pumnea (epumnea@sfu.ca)
Teyo Soler (tys2@sfu.ca)
Jesus Roberto Escamilla Yarahuan (jescamil@sfu.ca)

SRM: https://github.com/CircuitCo/BeagleBone-Black/blob/master/BBB_SRM.pdf?raw=true
TRM: http://www.ti.com/lit/er/sprz360f/sprz360f.pdf
Processor reference manual: http://www.openhacks.com/uploadsproductos/am3359.pdf

BeagleBone Black GPIOs:

We begin by determining what current we are able to source from the BBB's GPIOs first since this is where our motor controller will need to draw power for its logic circuitry. First we need to find a pin that can provide us with between 2.7 to 5.5V (I explain how I determine those values in the next section). Table 13 "Expansion Header P9 Pinout" on page 86 of the SRM lists 6 possible sources. Which one we pick depends on how we will be powering the BeagleBone when the circuitry is all connected. In our case, we will be powering the BBB from USB. Hence the pin number corresponding to SYS_5V will be appropriate. We determine this to be the correct header by reading section 6.1.3 "USB Power" of the SRM; in particular page 45.

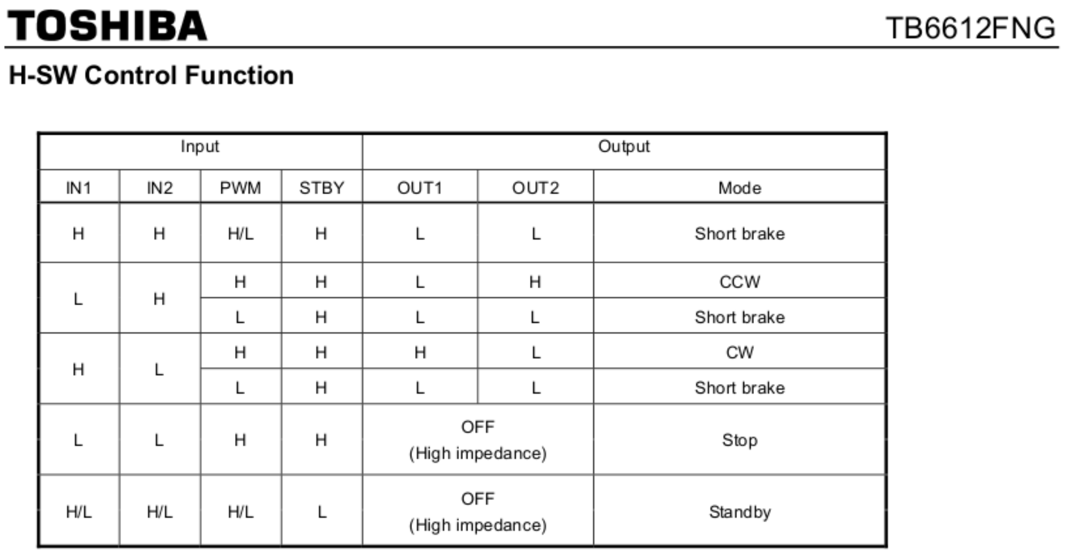
We now need to determine which GPIO pins will be suitable for driving the pins on the motor controller. The quick and dirty way of doing this is by again examining Tables 12 and 13 (pages 84 and 86 respectively) and looking for a pin with a name like GPIO<#>_<#> and mode0 (the next column over) set to gpmc_a* (you can look what this refers to in by referencing section 7.1 of the "AM335x Sitara Processors Technical Reference Manual" (TRM), but basically it refers to the memory bank the particular GPIO is controlled by).

OK, now we need to figure out how much current can be drawn from these GPIOs. For this we need to reference yet another document, the "Sitara AM335x ARM Cortex-A8 Microprocessors (MPUs)" Table 2-7 on page 20 (or there abouts, it's in that table trust me). In any case, we will see that we can source 6 mA and sink 8 mA on the GPIO pins. Now onto, looking at the spec-sheet for the motor driver/controller.

Motor Driver/Controller (A brief overview)

The spec-sheet for our motor controller provide us with all the information we need to figure out how we should interface it with the BBB. Nevertheless, the diagrams don't necessarily resemble the physical device exactly. That's OK, we only really need to look at the controller itself and the labels printed on the bottom. We can then just look up the name of each pin in the spec-sheet and see what voltage is required to power it and/or how much current it will draw or output. We also need to have a look at what configuration each pin must be in in order to make the motor spin.

Enrique Olaizola (eao@sfu.ca)
Emanuel Pumnea (epumnea@sfu.ca)
Teyo Soler (tys2@sfu.ca)
Jesus Roberto Escamilla Yarahuan (jescamil@sfu.ca)

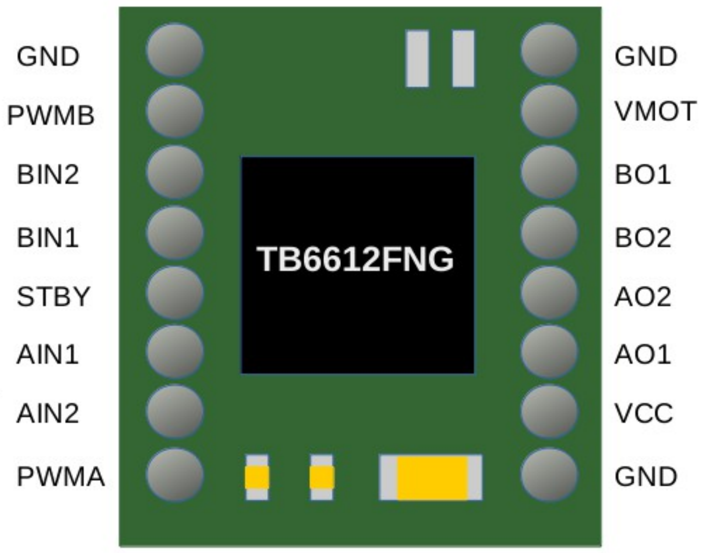


TOSHIBA TB6612FNG

H-SW Control Function

Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

The first table on Page 4 provides us with an overview of all possible pin configurations and what their effect is on the motor. Note that IN1 and IN2 listed in the table connect to two different GPIO pins on the BeagleBone, and will receive the data you write to those pins from your software. Correspondingly, Out1 and Out2 will be connected to the different connectors on the motors. By studying the table a bit more, you'll notice that the only way you will get the motors to do anything is if IN1 and IN2 are in a complimentary configuration, and both PWM and STBY are set to high. By the way PWM and STBY are also input pins that could be connected and controlled from your BBB's GPIO. However, they can also be connected to an external source of sufficient voltage, which would always keep them in a “High” state. Hence, in your software you will only need to write to GPIOs for IN1 and IN2 (or AIN0-1 and BIN0-1 as printed on the controller itself). You may wish to connect PWM to a GPIO in order to control how fast your motors spin. However, we won't cover that in this guide.



Enrique Olaizola (eao@sfu.ca)
Emanuel Pumnea (epumnea@sfu.ca)
Teyo Soler (tys2@sfu.ca)
Jesus Roberto Escamilla Yarahuan (jescamil@sfu.ca)

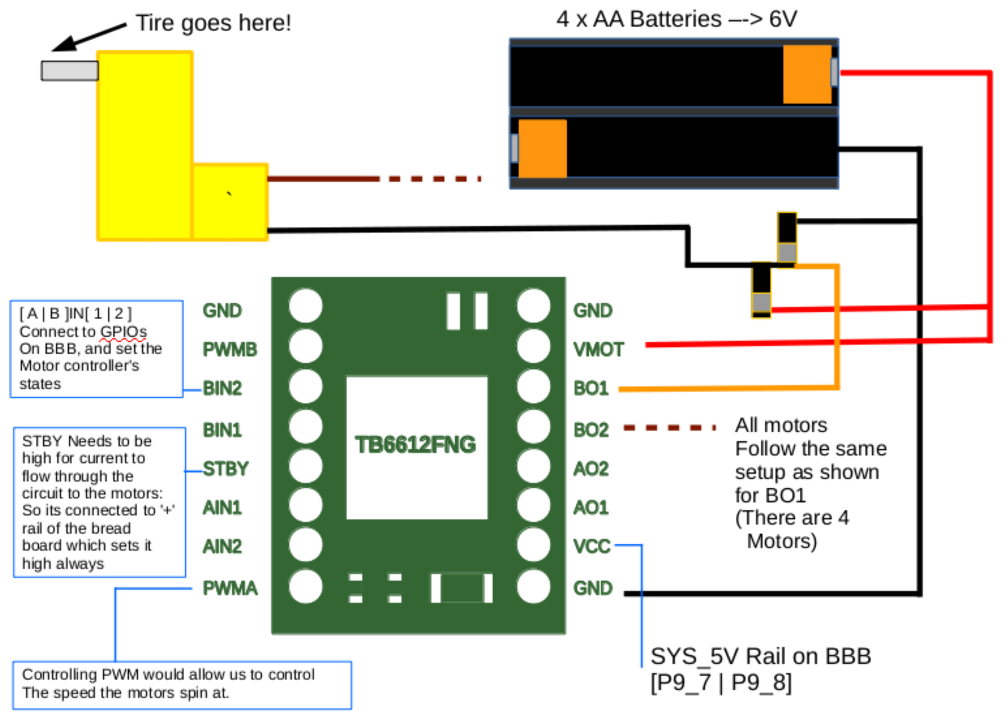
Now that we know how the motor controller connects to the BBB and to the motor, and what state each pin needs to be in for the motor to spin, we now need to worry about how we connect the motor driver to the BBB GPIOs (i.e. whether any additional circuitry is needed to protect the GPIOs or the controller itself). Remember that the BBB's GPIOs can only output a maximum of 6 mA. So if our driver requires more than that, no matter what we write to our connected GPIOs the motors will never spin. Consequently, we need to figure out how much current we need in order is to drive the logic circuitry in the motor driver.

VCC is the voltage required to power the logic circuitry in the motor controller. By referencing the “Operating Range” table on page 3, we find the minimum voltage we can use is 2.7 and the max is 5.5V. We can find the maximum and minimum voltage and current for all other pins on page 5 of the spec-sheet. In the case of IN1 (remember that this pin is labeled AIN0 on the controller itself), we would reference the “Control Input Voltage” and “Control Input Current” rows. You'll notice that under the “Symbol” column, Vih and Vil, each with it's corresponding tolerances. This refers to the amount of current needed to drive the input High or Low respectively. Also note that the cell where these voltages are specified relative to VCC. Finally, we note that the current required to drive our input pins ranges from 25 uA down to 1 uA. This means that the BBB would provide far more current that our controller's inputs need. This means we can safely, run a wire directly from our GPIOs on the BBB to the inputs on our motor controller.

<u>SOME ASSEMBLY REQUIRED.</u>

Now that we have all the knowledge we need in order to be sure we won't fry either our BBB's GPIO pins or our Motor controller we can now go ahead and assemble the circuit to connect them and the motors together. I provide a very simple schematic that shows how each of the pins from the motor controller can be connected to both the battery and GPIOs on the BBB. Recall from our earlier discussion how you select which pin on the BBB's breakout headers are appropriate for interfacing with the motor controller (the we I provided is a short cut, if you need more headers, consult the SRM for which mode will allow you to enable the desired functionality).

Enrique Olaizola (eao@sfu.ca)
Emanuel Pumnea (epumnea@sfu.ca)
Teyo Soler (tys2@sfu.ca)
Jesus Roberto Escamilla Yarahuan (jescamil@sfu.ca)


Tire goes here!
4 x AA Batteries --> 6V
[A | B]IN[1 | 2] Connect to GPIOs On BBB, and set the Motor controller's states
STBY Needs to be high for current to flow through the circuit to the motors: So its connected to '+' rail of the bread board which sets it high always
Controlling PWM would allow us to control The speed the motors spin at.
GND
PWMB
BIN2
BIN1
STBY
AIN1
AIN2
PWMA
TB6612FNG
GND
VMOT
BO1
BO2
AO2
AO1
VCC
GND
All motors Follow the same setup as shown for BO1 (There are 4 Motors)
SYS_5V Rail on BBB [P9_7 | P9_8]

Controlling the motors from software:

All that remains now is to drive the pins on the motor controller from within Debian. This is fairly trivial. Once you have connected the pins on the controller to the appropriate GPIOs (recall the earlier discussion on how to choose the appropriate pins on the BBB's breakout headers) all we have to do is calculate that pins number and echo it to the export file under /sys/class/gpio/. For example, if you had connected AIN0 to pin 16 on P8 (listed as GPIO1_14) and AIN1 to pin 17 (listed as GPIO0_27). Then we would do the following to export the pins into user-space:

- echo 46 > export   # 46 = 32x1 + 14
- echo 27 > export   # 27 = 32x0 + 27

If all went well you would have a couple of new files under /sys/class/gpio/. One named gpio46, the other gpio27. We now need to set the direction so that when we write to the 'value' file the system will know to push the value out to the GPIOs and consequently to the connected motor driver.

- echo out > gpio46/direction
- echo out > gpio27/direction

Enrique Olaizola (eao@sfu.ca)
Emanuel Pumnea (epumnea@sfu.ca)
Teyo Soler (tys2@sfu.ca)
Jesus Roberto Escamilla Yarahuan (jescamil@sfu.ca)

Finally, we can write to the 'value' file. Remember that for the motor to spin the values have to be complimentary, if one is set to 1 (High) the other must be set to 0 (Low) otherwise the motor won't spin.

- echo 1 > gpio46/value
- echo 0 > gpio27/value

Naturally, reversing the order will cause the motor to spin in the opposite direction. Writing code to do what was just demonstrated here is simply a matter of using the appropriate system calls for opening, writing, reading, and closing regular files. One thing to note is that you will need to run your code as root.