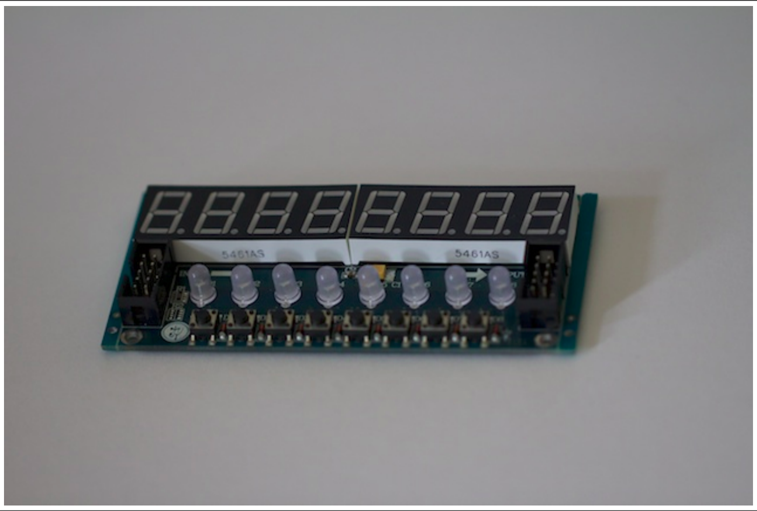
The TM1638 & the Raspberry Pi



One of the Raspberry Pi’s many features is that it’s easy to connect to displays: it has both a modern HDMI connector and a phono socket dispensing old-school composite video. However, there are times when one doesn’t want a large monitor around but would rather have a few seven-segment displays instead.

Whilst one could build that from scratch, one can also buy small modules with eight seven-segment displays, eight red-green LEDs and eight push buttons for about $7 (as of August 2012). I bought mine from dealextreme[1] but there might be other sources.

The boards are basically just the LEDs and switches, plus a TM1638 driver chip. The chip sits on a two-wire serial bus which makes it fairly easy to connect the boards to a computer/microcontroller of your choice. Of course, one needs a little bit of software, so I wrote some.

Arduino woz ’ere.

People have already worked out how to do all this on the Arduino:

- John Boxall wrote a blog about it[2] which describes the boards in great detail.
- Ricardo Batista wrote a library to handle the comms.[3]
- Marc[4] (via John above) found a datasheet.[5]

On the Raspberry Pi

Hardware issues

The only important difference between the Arduino and the Raspberry Pi in this case is that the Arduino’s a 5V beast but the Pi prefers 3.3V. Happily though the dealextreme board appears to cope perfectly well with the lower supply voltage.

Software

My code isn’t really a port of Ricardo’s Arduino library: I wanted a different API. However, I did copy his nice 7-segment font, and his code was very helpful when it came to understanding the data-sheet.

Installation

Before you start, you’ll need Mike McCauley’s nice bcm2835 library.[6]

You can then grab my TM1638 library from github[7] in a couple of ways.

If you’ve got autotools then you can just clone the repository:

```
$ git clone https://github.com/mjoldfield/pi-tm1638.git
$ cd pi-tm1638
$ autoreconf -vfi
$ ./configure
$ make
$ sudo make install
```

You might find this easier and faster though:

```
$ wget https://github.com/downloads/mjoldfield/pi-tm1638/pi-tm1638-1.0.tar.gz
$ tar xzvf pi-tm1638-1.0.tar.gz
$ cd pi-tm1638-1.0
$ ./configure
$ make
$ sudo make install
```

Generic AVR support

In a pleasingly symmetric way, Filipe Moraes has ported this back to generic AVR chips, and added a scroll feature. You can read about it in Brazilian Portuguese on his blog,[8] or grab the code from http://devpix.net/blog/wp-content/uploads/2013/05/tm1638pjt.zip.[9]

Documentation

If you’ve got doxygen installed the compilation process leaves HTML files in doc/html. Otherwise feel free to browse them on github.[10]

Examples

Three example programs are included with the software, and you’ll find them all in the examples directory:

- tm1638-hello: The canonical ‘Hello World’ program.
- tm1638-buttons: A simple demonstration which reads the buttons.
- tm1638-clock: A digital clock.

N.B. All three examples hard code the pin numbers into the executable. So to run the examples you’ll need to make the following connections:

Raspberry Pi	TM1638 Board

Name	Pin	Name	Pin
3.3V	P1-01	Vcc	Pin 1
GROUND	P1-06	GND	Pin 2
GPIO 17	P1-11	DIO	Pin 4
GPIO 21	P1-13	CLK	Pin 3
GPIO 22	P1-15	STB1	Pin 5

Raspberry Pi Revision 2

Dominik Eschenmoser pointed out that if you’re using revision 2 of the Pi hardware, you have to chance the clock pin from GPIO21 to GPIO27.

Software

Once you’ve sorted out the hardware, doing simple things with the hardware is easy.

Let’s look at one of the example programs, which turns your Raspberry Pi into a digital clock. Stripping out the comments and error checks, here’s the code:

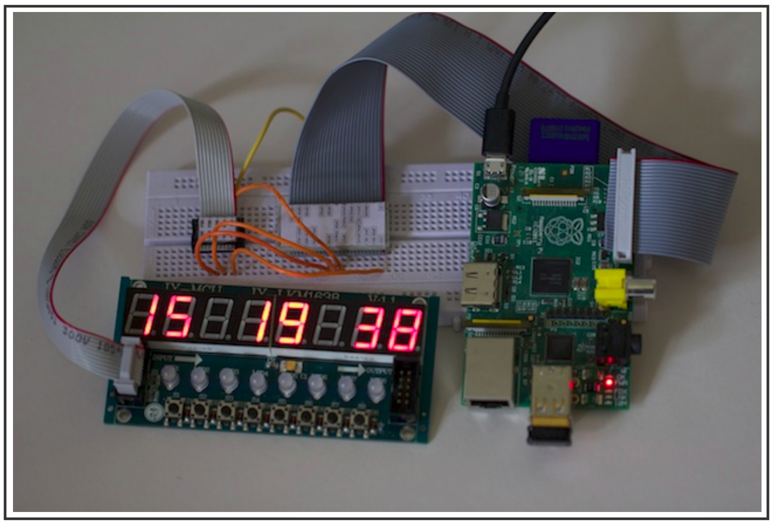
```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>

#include <bcm2835.h>
#include <tm1638.h>

int main(int argc, char *argv[])
{
  bcm2835_init();
  tm1638_p t = tm1638_alloc(17, 21, 22);

  while(t)
    {
      time_t now    = time(NULL);
      struct tm *tm = localtime(&now);

      char text[10];
      snprintf(text, 9, "%02d %02d %02d",
                  tm->tm_hour, tm->tm_min, tm->tm_sec);

      tm1638_set_7seg_text(t, text, 0x00);
      delay(100);
    }

  return 0;
}
```

If you’ve installed the tm1638 library, and saved the code above as clk.c, to compile and run it:

```
$ gcc -std=c99 clk.c -o clk -lbcm2835 -ltm1638
$ sudo ./clk
```

You need to run the program as root (which is what the sudo does above), so that the code can talk to the GPIO hardware.

Having done that little lot, you should get something like this: an automatic wireless clock:



References

1. http://www.dealextreme.com/p/8x-digital-tube-8x-key-8x-double-color-led-module-81873?item=8

2. http://tronixstuff.wordpress.com/2012/03/11/arduino-and-tm1638-led-display-modules/

3. http://code.google.com/p/tm1638-library/

4. http://www.freetronics.com

5. http://dl.dropbox.com/u/8663580/TM1638English%20version.pdf

6. http://www.open.com.au/mikem/bcm2835/

7. https://github.com/mjoldfield/pi-tm1638

8. http://devpix.net/blog/?p=323

9. http://devpix.net/blog/wp-content/uploads/2013/05/tm1638pjt.zip

10. http://mjoldfield.github.com/pi-tm1638/tm1638_8h.html

Revised by Martin Oldfield on 29 Dec 2013, original version 12 Aug 2012.
Contact: Martin Oldfield, ex-atelier@mjoldfield.com
