The Pequeno – An Arduino me-too-alike for PIC Microcontrollers

How to use the large variety of Arduino ’shields’ on the PIC Microcontroller.

pequeno(portuguese) adj. *small, little, minor, slender*. (amongst others)

Apologies in advance for the mixed bag this post has turned into, covering as it does the Pequeno board, a 28-pin SOIC header, Arduino prototype board and 2-wire LCD board, not to mention the source-code! As an aid to moving around the post I’ve included a small link menu to the various topics below.

We’ve all done it – looked sideways with not a little envy at a range of add-ons for a power tool, and wished that they were available for the one we bought.

There is no ’standard’ footprint for a PIC Micro-controller development/experimental board. Vendors have chosen to provide their own ‘take’ in each case, and who can blame them, selling ‘add-ons’ for a proprietary board increases revenue streams.

Given the maturity of the PIC Micro-controller, one would have thought an ‘Open Source’ standard board for experimenters might have been developed, but no, not yet. I’m as guilty as anyone else, tending to re-vamp and tweak my designs almost on a whim – but that’s another story.

Recently the Arduino project for the AVR Micro-controller was launched, and has become an immediate hit with experimenters and educators alike, with a large number of add-on/plug-in ’shields’ developed and available cheaply, not just by private vendors but also Open Source.
This has probably tempted some putative users of the PIC, to plump for the AVR offering instead, and who can blame them?

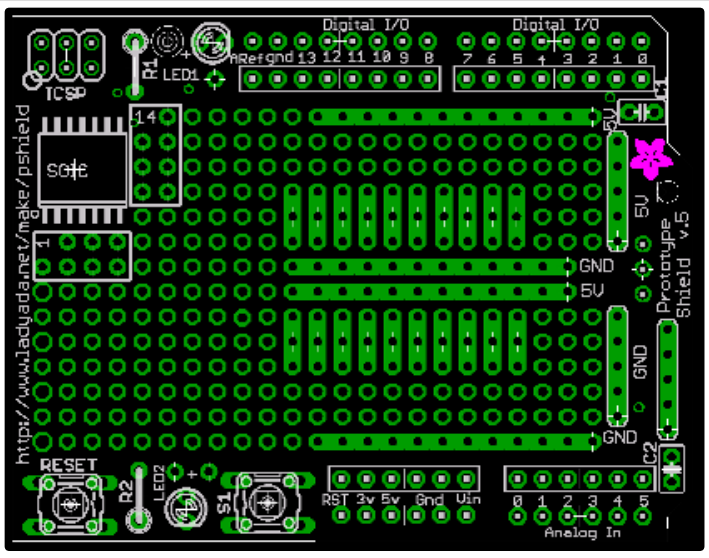
Back in 2004 when I again picked up the soldering iron after what seems a lifetime of programming the PC, I examined both the PIC and AVR controllers and decided to use the PIC – it was cheaper, there was more choice of devices, blah, blah. That’s not to say I have any negative criticism for the AVR controllers – I haven’t. But they offer me nothing I have already got in the PIC, and I now have 7 years of PIC development behind me.

This post briefly documents a small PIC Microcontroller development board that mimics the Arduino almost exactly – so that add-ons (’shields) developed for the Arduino can be used on a PIC Micro-controller. All printed circuit designs for the boards are available in the downloads section at the end of the post, together with software to test out the board.

As is my policy, I will try not to repeat stuff here that is already available elsewhere on-line, other than to give an overview, nor give long lists of links which have a habit of disappearing. If you are not already familiar with the Arduino, and the Arduino Shield designs, then swot up on them elsewhere.

The Arduino Shield.

Take a look at the layout of the following PCB which is freely available on-line:



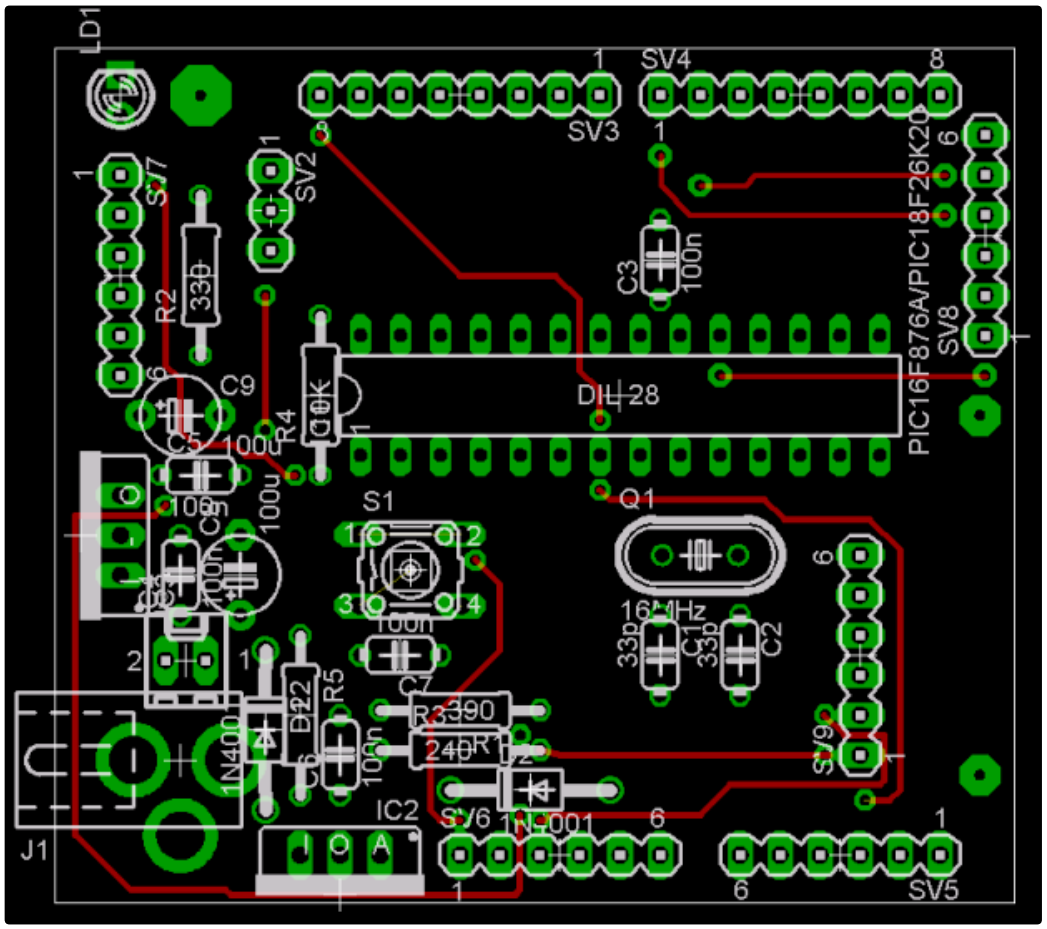
Arduino Prototype Shield PCB

Now concentrating mainly on the double-rows of pin-headers, there are 2 sets of 8 to the top of the picture, and 2 sets of 6 to the bottom. Moving clockwise from the 1st pin at the top, this is marked ‘AREF’, and the next ‘GND’. The rest of the top rows of pins are marked ‘Digital I/O’. Moving (clockwise) to the bottom row, the first rows of pins are marked ‘Analog In’, and numbered 5 thru 0. The next row of 6 are marked ‘VIn’, ‘GND’, ‘5V’, ‘3V’, and ‘RST’.

All of the pins are just simply what they say, and the ‘RST’, ‘Analog’, ‘AREF’, and ‘Digital’ can all be provided by a 28-pin PIC Micro-controller. On the controller board itself, you will probably want a means of programming the PIC without having to remove it from the board each time.

I’ve chosen to provide 2 distinct methods, Microchip’s tried and tested ICSP protocol, and also a boot-loader protocol based on the use of the TinyBld boot-loader and an FTDI RS232-TTL USB lead. My Pequeno board (Geddit?) layout is shown in the next picture, and is not to scale so that you can clearly see the various items as these are discussed.

Pequeno

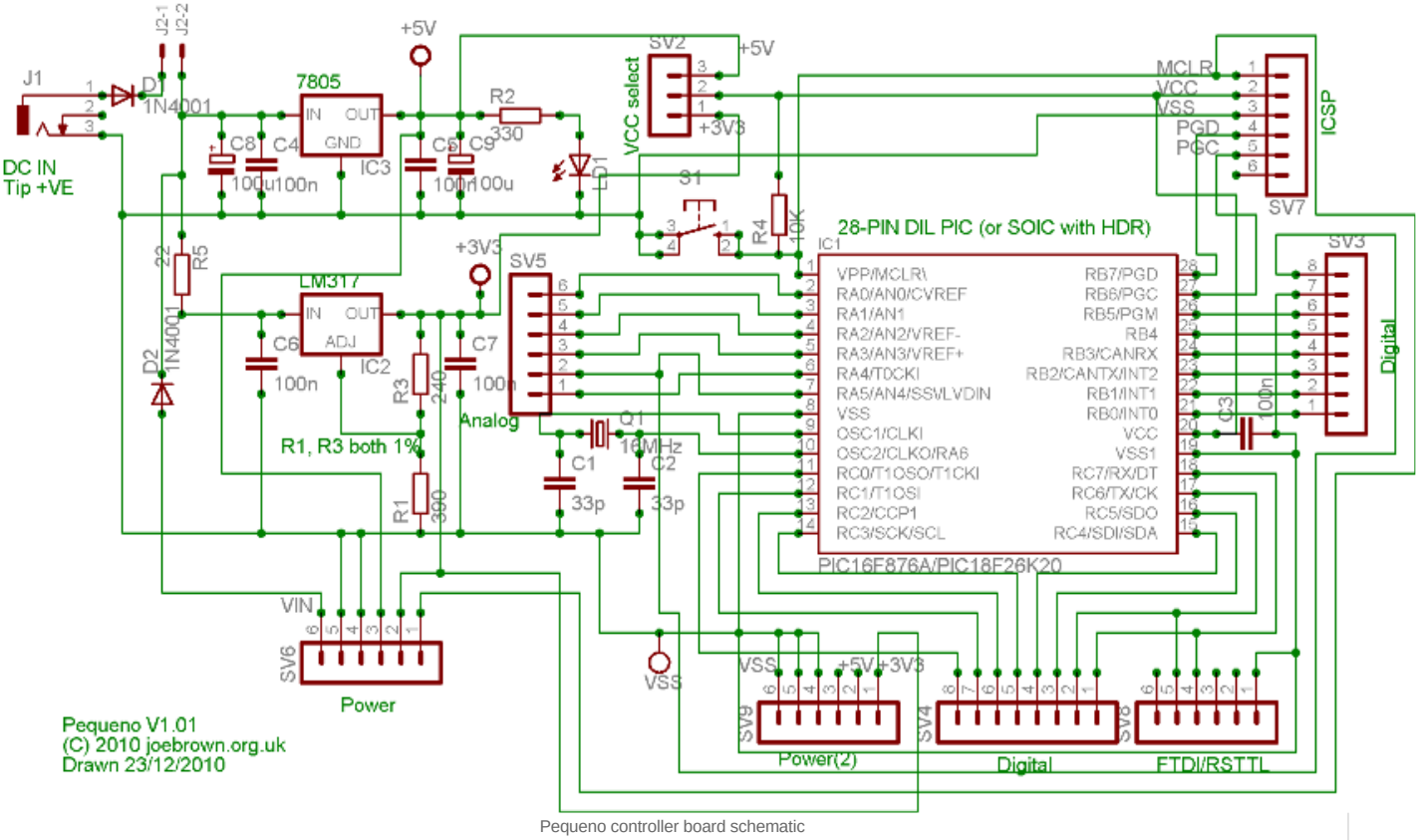


The Pequeno Controller Board

Pin Headers and jacks

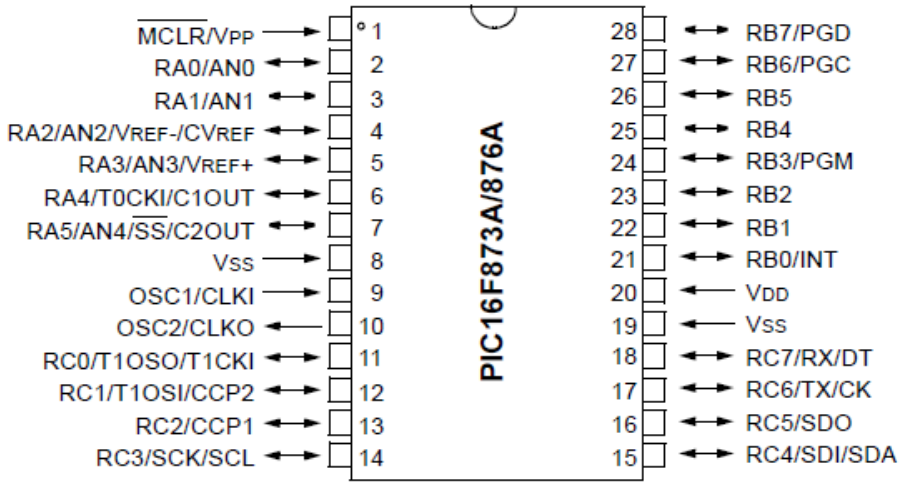
SV3, SV4, SV5, and SV6 provide connections as per those detailed above.
SV2 is the voltage selector for the controller – either 3v3 or 5v
SV7 is the ICSP connector.
SV8 is the FTDI RS232-TTL USB connector.
J1 is the external power connector.
J2 provides a link to break power from J1.
SV9 is a secondary power connector.

The PIC.
I've fitted a 28-pin 0.3inch wide DIL socket, which will accomodate virtually any 28-pin PIC Micro-controller. I had decided the board should be capable of running both mid-range and above, 8-bit PICs, and the board has been fitted with a 16Mhz crystal, and tested both with a 5 volt 16F876A (**4 MIPS**) and also a 3.3 volt 18F26K20. (**16 MIPS**) These are fit-alike apart from the actual package. I've provided a small header PCB to mount a SOIC version of the 18F26K20 on.
The schematic of the controller board is shown below.

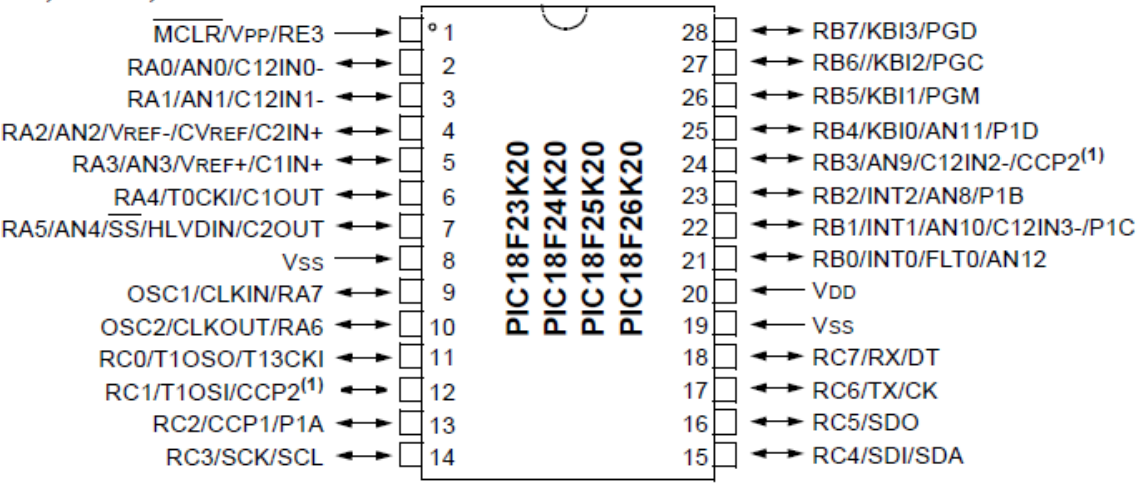


Pequeno controller board schematic

Differences between the 2 PICs I tested reflect the advancement in design, with the pin diagrams showing a plethora of facilities on the newer, PIC18F26K20.

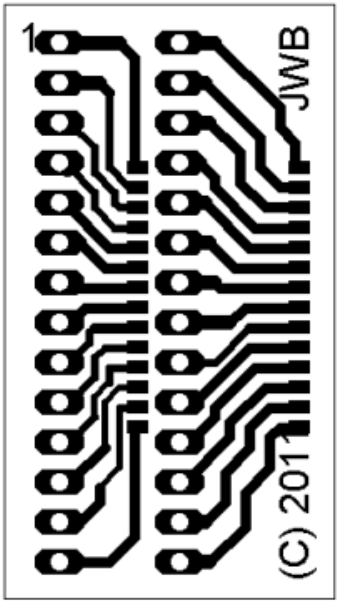


PIC16F876A Pin Diagram unashamedly stolen from Microchip datasheet



PIC18F26K20 pin diagram similarly stolen

28-pin SOIC Header.
If you have a favourite SOIC to DIL convertor header, then use that if you wish. Mine is crude, but effective and easy to manufacture, and most importantly, easy to solder.



28-pin SOIC to DIL Header PCB

My method of construction is as follows:

Place the drilled board, foil uppermost over a 0.1 inch matrix breadboard.
Push a single PCB pin (I used Cambion pins) through a hole until it bottoms in the breadboard.
Carefully snip off excess pin, leaving a (very) small amount of pin projecting above the surface of the board.
Solder, ensuring intimate contact between solder, pin and pad foil.
Repeat above for all 28 pins.
Take a pair of long-nosed pliers and carefully straighten the 'foot' of each pin on the SOIC PIC package, so that the package stands on it's toes. (like a ballerina)
Lightly tin opposite corner pads on the SOIC header, then solder the SOIC package in place, correctly aligned over the pads, and noting correct placement of pin 1.
If you find the package does not sit correctly, you've used too much solder on the DIL pins, or not cut them short enough.
Solder each pin in turn, then buzz out with a testmeter and resolve any short-circuits.

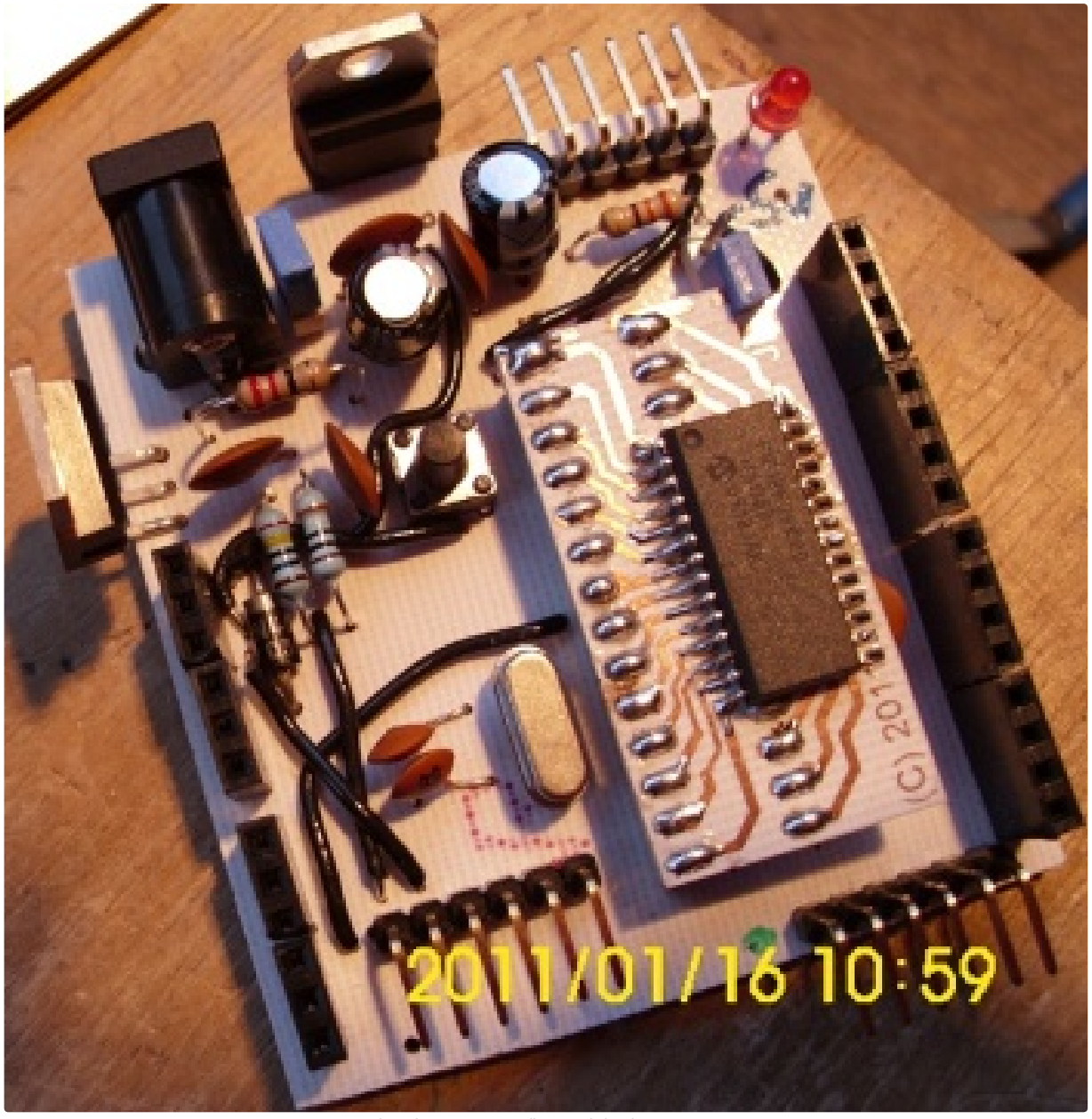
Programming a SOIC PIC.
Peace of mind is a wonderful thing, and if I had a mantra, achieving such would be part of it. Those familiar with my philosophy of leaving as little to chance as possible where development is involved will understand that for me, soldering a SOIC PIC to a header, and plugging it in to an untested controller board is total anathema. If the result won't program using the ICSP connector, whose fault is it?

My 'no crying' method is as follows:

- Build and*fully* debug the Pequeno board, *then*plug in a programmed 16F876A and fully test everything.
- Program the SOIC PIC*before*soldering it to the header the correct way around. How to do this is documented here:Versatile SOIC PIC Programming Header
- Buzz out every pin to it's neighbouring and opposite side pins, ensuring no shorts.
- Take a a deep breath, plug in the header the correct way around, and switch on.

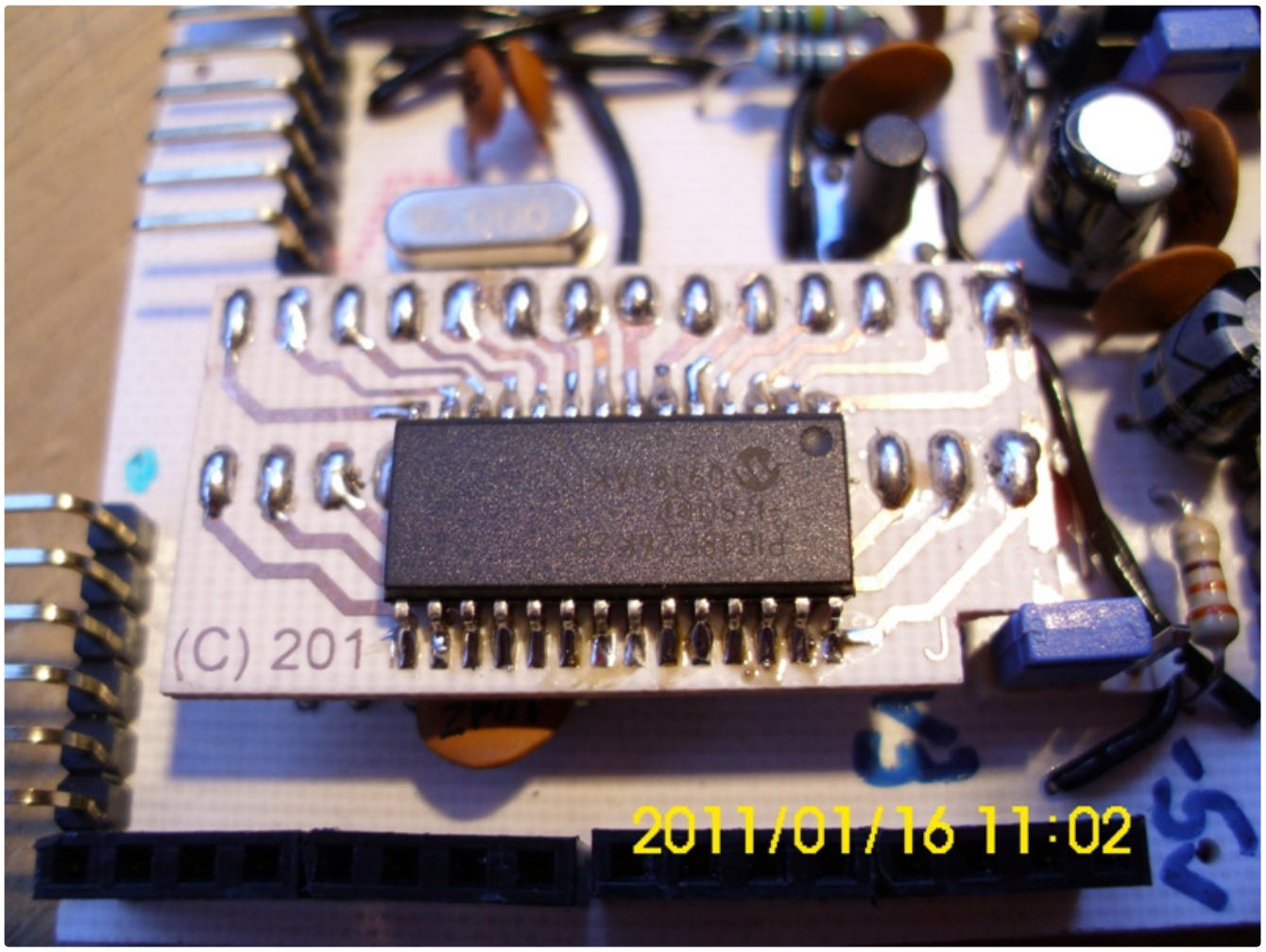
After initial programming with the TinyBld bootloader, I dispense with the ICSP dongle and use the FTDI RS232-TTL link cable to the PC. Ensure that you have a 3-volt version of this if you are using the 18F26K20 or similar low-voltage PIC.
A link to the relevant FTDI page is given after the Downloads section at the end of this post.

Photos of prototype board



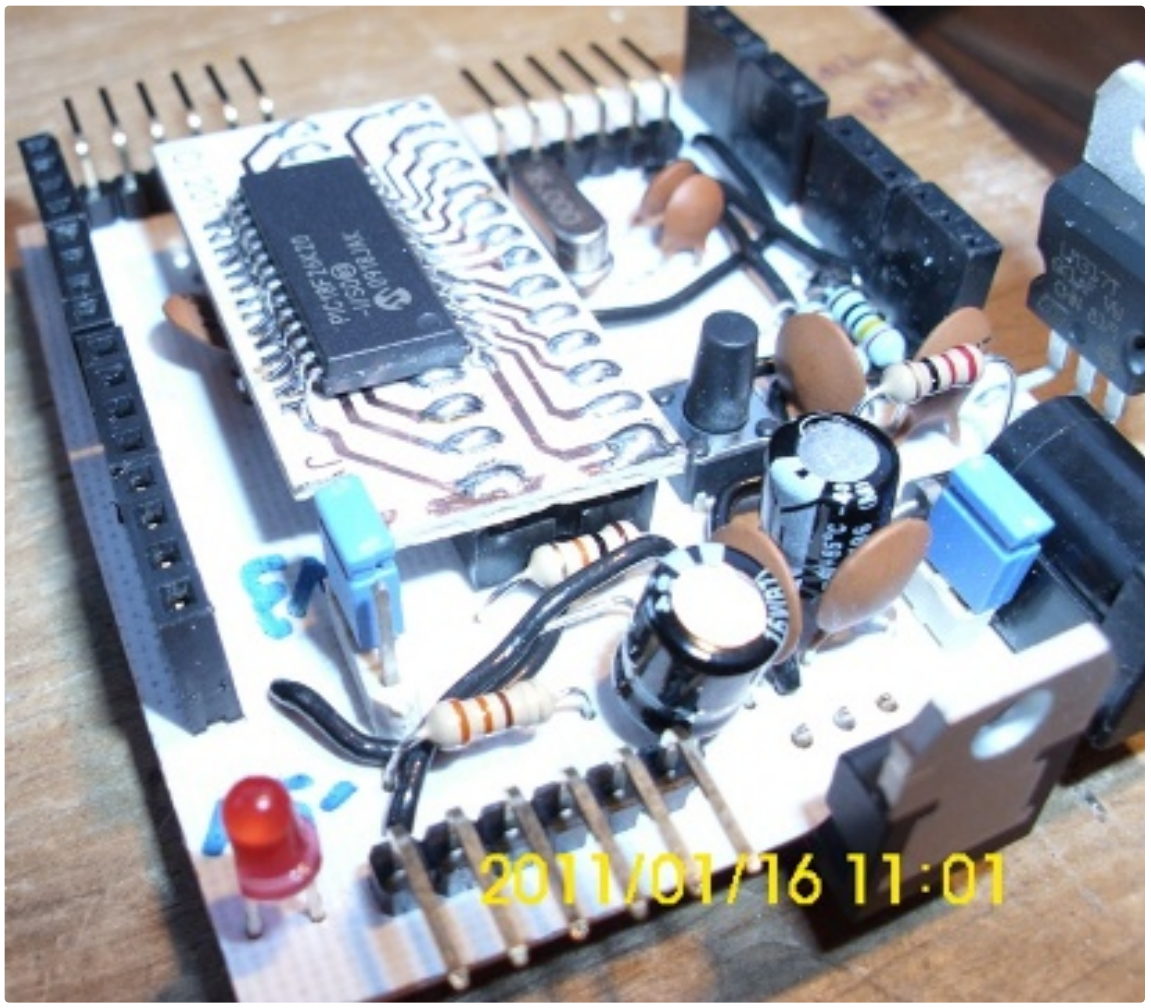
Top view of Pequeno Controller Board showing PIC18F26K20

I had to cut a small notch in the SOIC header board to clear the voltage selection link header. (top right) The header has been moved and re-oriented in version V1.01 of the PCB. This is shown more clearly in the close-up of the SOIC header below.



Pequeno - close-up of 18F26K20 SOIC PIC on header

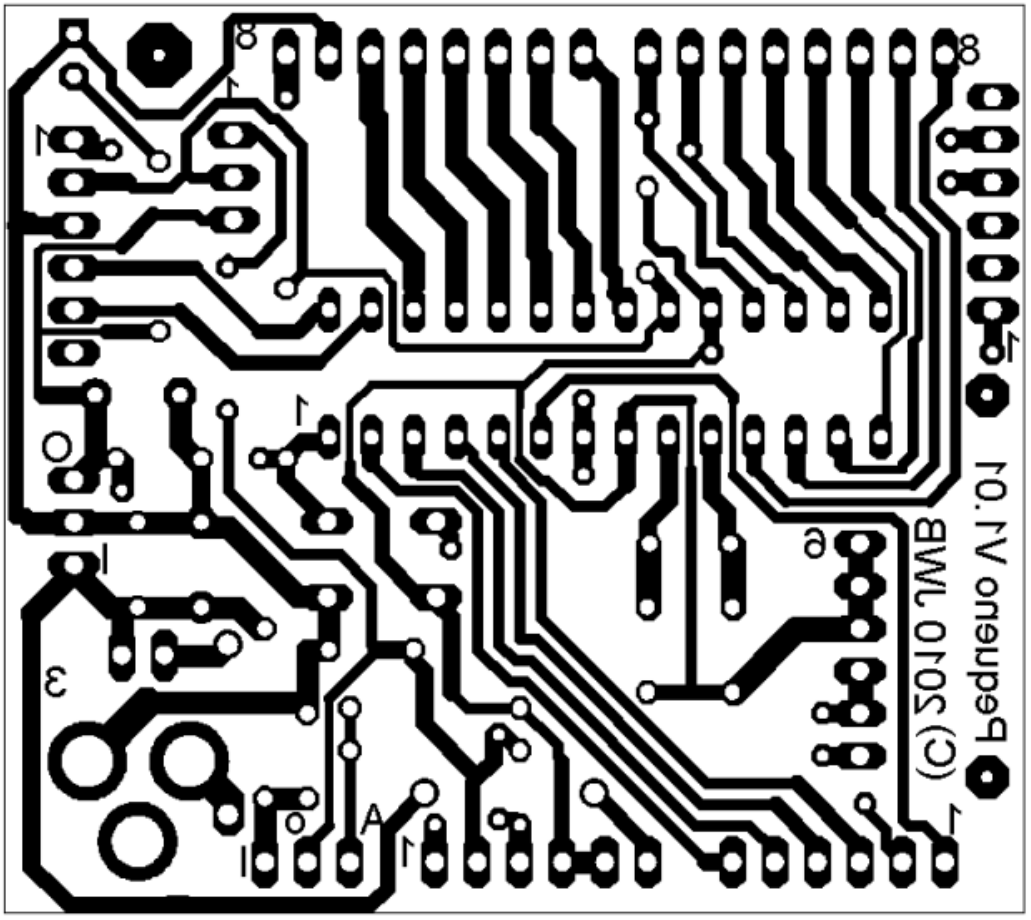
The following gives a view from top-right, clearly showing the clearance under the PIC18F26K20 SOIC package by standing it on it's 'toes'.



Pequeno - view from top right showing clearance under 18F26K20

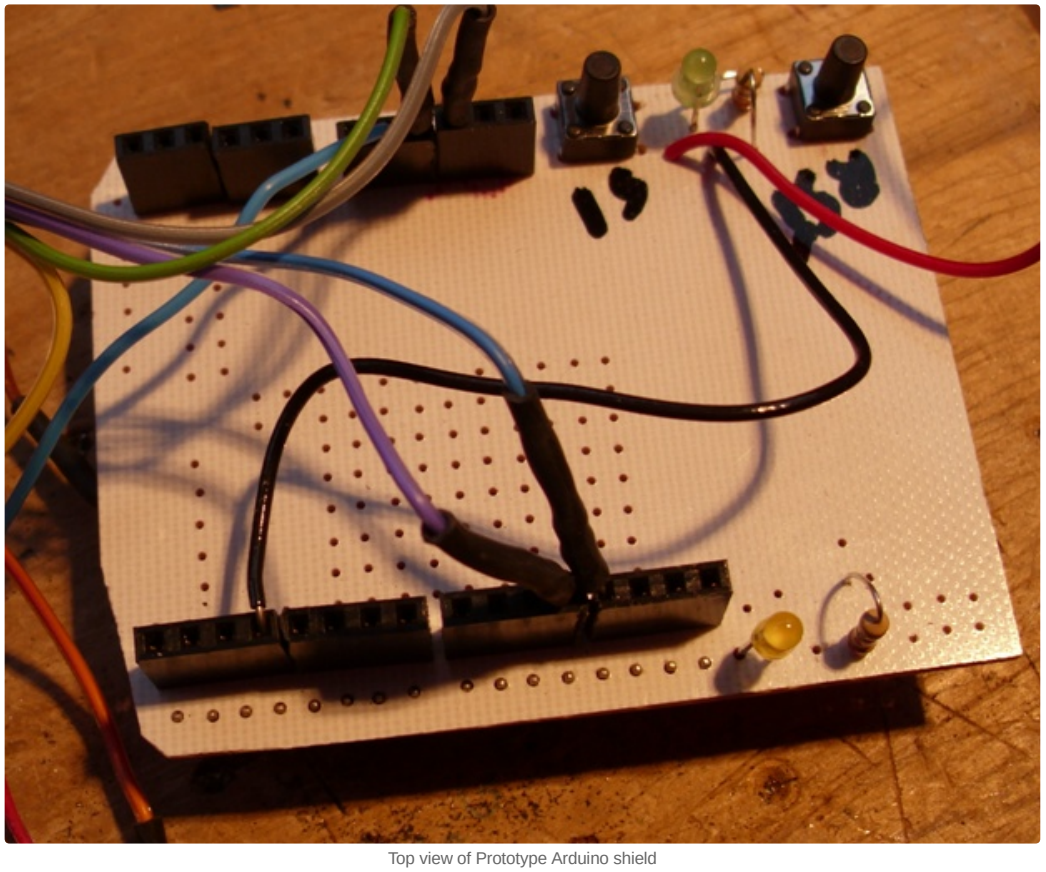
Note I aligned both voltage regulators so that the fins don't foul the 1st 'shield' when this is fitted. I anticipate fitting a small cooling fin to each regulator when extra power is needed for some of the add-on 'shields'. Note also, I've raised the 22 ohm resistor a little from the PCB top to aid cooling.

The following shows the bottom foil pattern for the Pequeno board. The full Eagle project is given in the download section.



Pequeno PCB bottom foil pattern.

Testing.
I made a version of the prototype Arduino 'shield' shown in the 1st photo of this post, leaving off the top foil and SMD components.

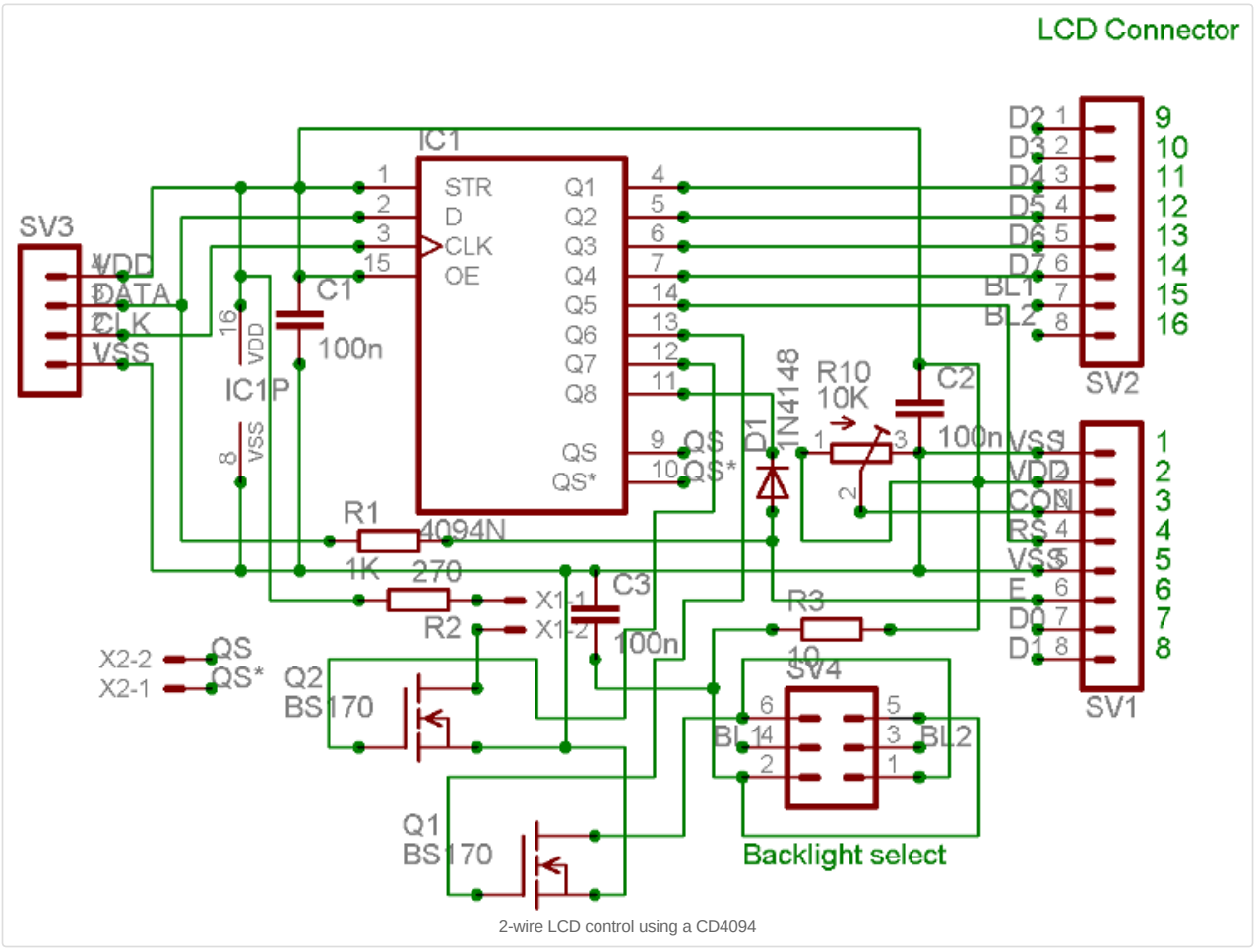


Top view of Prototype Arduino shield

The photo above clearly shows the two rows of header pins/sockets, enabling shields to be stacked one above the other on top of the Pequeno. Also, note the RST switch top right – necessary to manually reset the Pequeno board, when it's own reset switch is made inaccessible.

2-wire LCD Controller

In the course of doing this project, I wanted to add the facility to display to a 20X4 LCD screen. This would have perhaps merited a shield in it's own right – but I wanted to use as least port pins as possible, so plumped instead for a 2-wire LCD controller using a standard LCD, a shift register and a few other components. No originality is claimed for this idea – several circuits exist already online, but this is my version.
Since the LCD only uses 6 bits of the shift register, I have provided a facility to switch a backlight on or off, together with a 'spare' output. The schematic is given below:



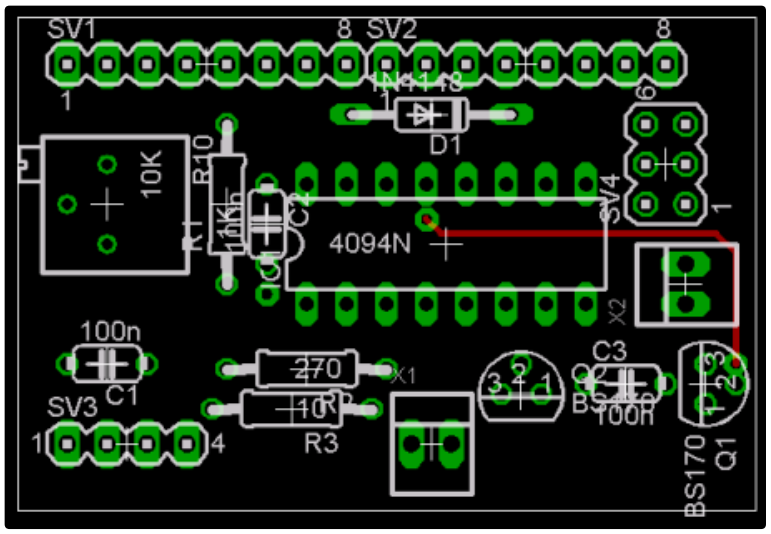
2-wire LCD control using a CD4094

A few words about the circuit:

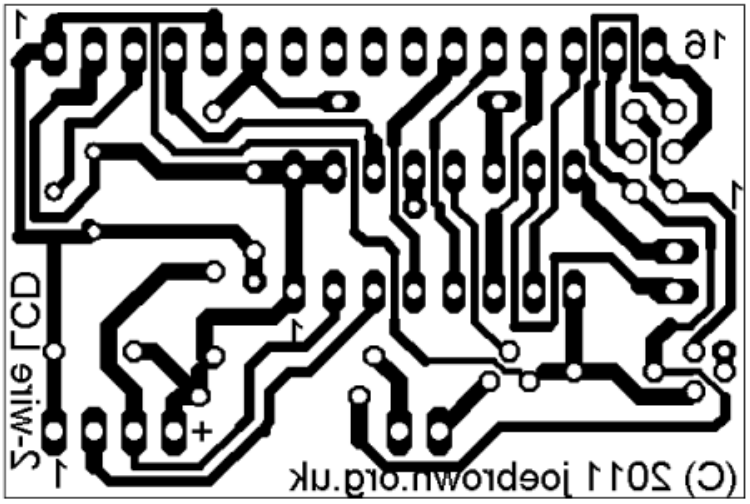
- X1 provides connection to the mosfet drain and a 270 ohm resistor to 5v. I anticipated use of an LED here, but changing the resistor value will faciliate use of a small relay etc.
- X2 provides output from the shift register for expansion – ie another shift register.
- SV3 are the connections from the PIC controller and PSU. Only 2 port pins are required.
- SV4 provides polarity selection for the backlight LED. All LCD's are not created equal – some with anode and kathode switched on the pin header.
- Finally, SV1 and SV2 provide connection to the LCD itself. I've numbered the LCD connections to the right of the pin header sockets. Note that a 4-bit interface to the LCD is used, necessitating two bytes per command through the 2-wire interface.

Note that although I used a 18F26K20 (3.3 volts) on the Pequeno board, the LCD and it's 2-wire controller CD4094 run on 5 volts. The PIC and CD4094 are quite happy with the logic levels available. (they are both*outputs* from the PIC)
Always ensure you have the correct VDD voltage selected for the inserted PIC Microcontroller before switching on!

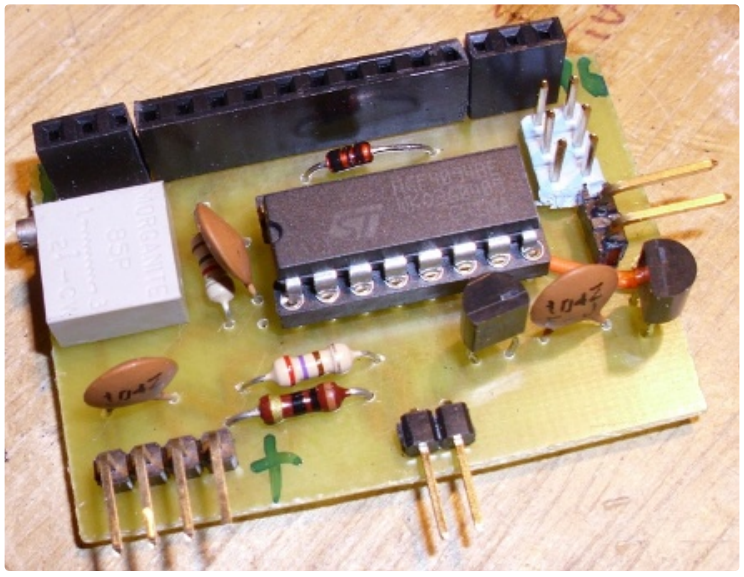
The top (component) view of the small PCB is shown below, followed by the foil pattern and a few photos of the completed prototype in use with the Pequeno board.



2-wire LCD Component and links



2-wire LCD controller foil pattern



2-wire LCD controller completed

2-wire LCD being driven from Pequeno controller

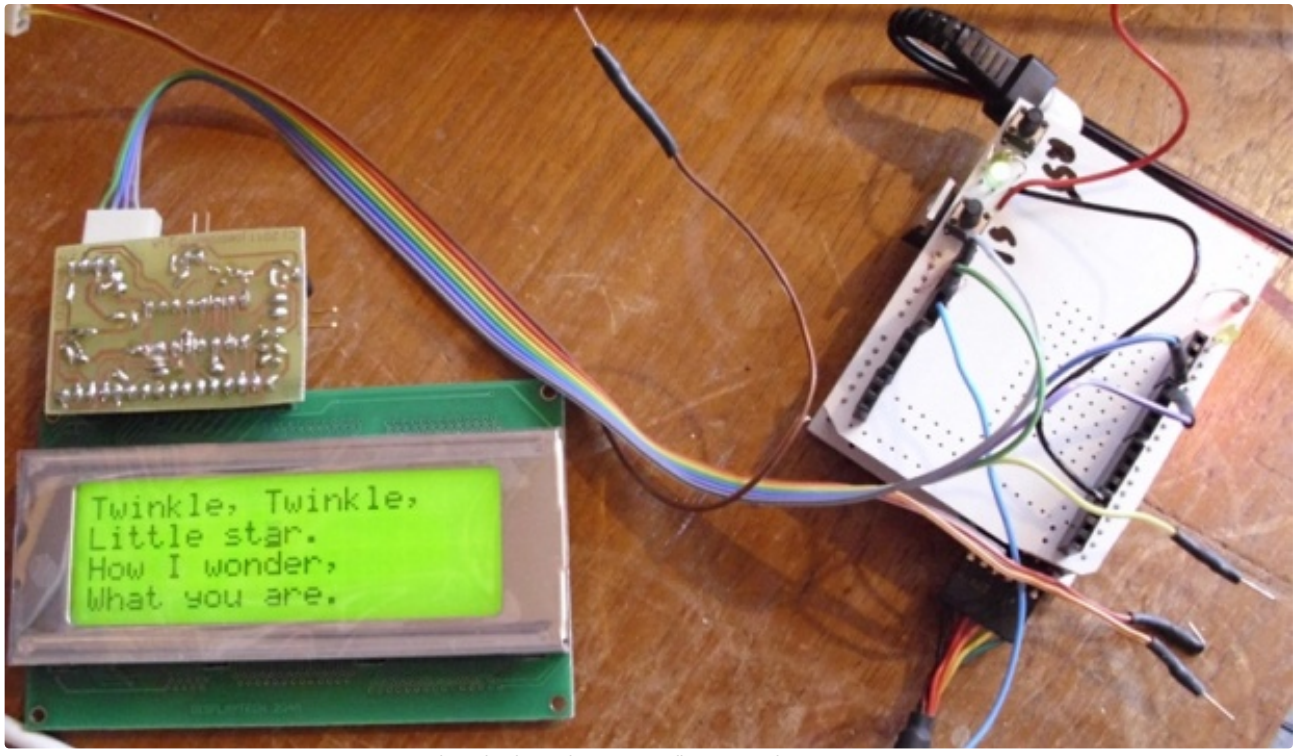


Photo showing 2-wire LCD controller connected to Pequeno

The Code
Below is an extract from the test code I used. The full source is given in the downloads. I've added the LCD routines to a sample supplied with the free version of SourceBoost compiler I used to build the code. (using both 16F876A and 18F26K20) Although I've embedded the LCD routines into the main code, (music.c) You could tidy these up and put them into a separate file + header, or even a library.

```
////////////
//
// 2-wire L
//
// JWB 4th
//
// Many ver
//
// Assume a
// sizes.
//
////////////

// write gi
void lcd_st
{
        whi
        {


        }
}

// Move cur
void lcd_li
{
        lcd
}

void lcd_li
{
        lcd
}

void lcd_li
{
        lcd
}

void lcd_li
{
        lcd
}

// clear a
void lcd_cl
{
        lcd
        lcd
}

void lcd_cl
{
        lcd
        lcd
}

void lcd_cl
{
        lcd
        lcd
}

void lcd_cl
{
        lcd
        lcd
}

// move cur
void lcd_go
{
        lcd
        del
}

// clear th
```

```
void lcd_cl
{
        lcd
        del
}

// home the
void lcd_ho
{
        lcd
}

void lcd_Cu
{
        lcd
}

void lcd_Cu
{
        lcd
}

void lcd_Cu
{
        lcd
}

// issue co
void lcd_cm
{
        lcd
        lcd
}

// draw a c
void lcd_ch
{
        lcd
        lcd
}

// write a
void lcd_ny
{
// I used a
 #define re
        int
        lcd
        //
        for
        {

        }
        //
        //
        lcd
        tog

        //
        lcd
        tog

        //
        lcd
        tog

        //
        lcd
        tog
        for
        {









        }
        tog
}
```

```
// initiali
void lcd_in
{
        uns
        lcd
        del
        tog
        del
        tog
        del
        lcd
        del
        lcd
        del
        lcd
        del
        lcd
        del
        lcd
        del
        lcd
        del
        lcd
        del
}

// toggle t
void toggle
{
        lcd
        lcd
}

// toggle t
void toggle
{
        lcd
        lcd
}
```

Downloads
The 3 Eagle schematics and PCB layouts discussed in this post are all in one Eagle project here:**http://joebrown.org.uk/images/Pequeno/Pequeno_PCBs.zip**
An annotated view of the top of the Pequeno board is here:**http://joebrown.org.uk/images/Pequeno/Pequeno_Top_V1.01.PNG**
The Eagle project file for the Arduino prototype shield board layout is here:**http://joebrown.org.uk/images/Pequeno/protoshield.zip**
The 'C' source-code, (a SourceBoost project) including conditional compilation for 16F876A and 18F26K20 is here:**http://joebrown.org.uk/images/Pequeno/2-wire-LCD_and_music_C_source-code.zip**
The TinyBld bootloader code for the 18F26K20 is here:**http://joebrown.org.uk/images/Pequeno/TinyBld18F26K20.zip**
The TinyBld bootloader code for the 16F876A is here:**http://joebrown.org.uk/images/Pequeno/tbuild_16F.zip**