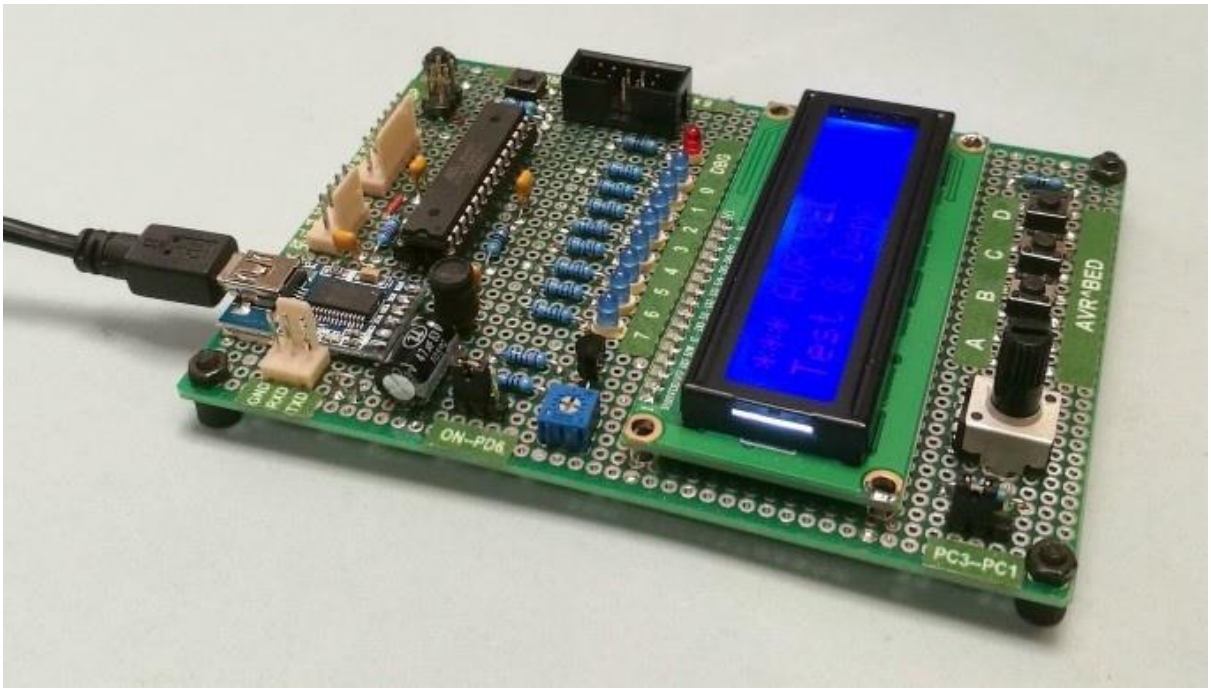
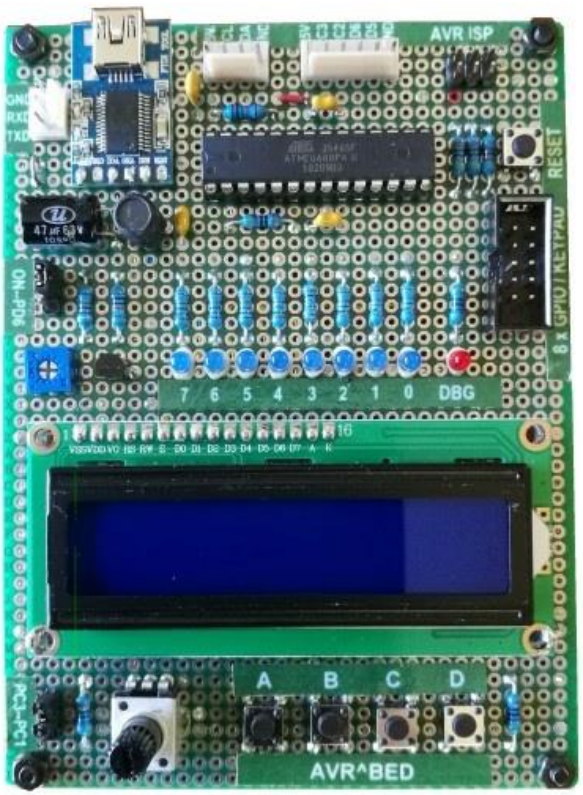
AVR-BED

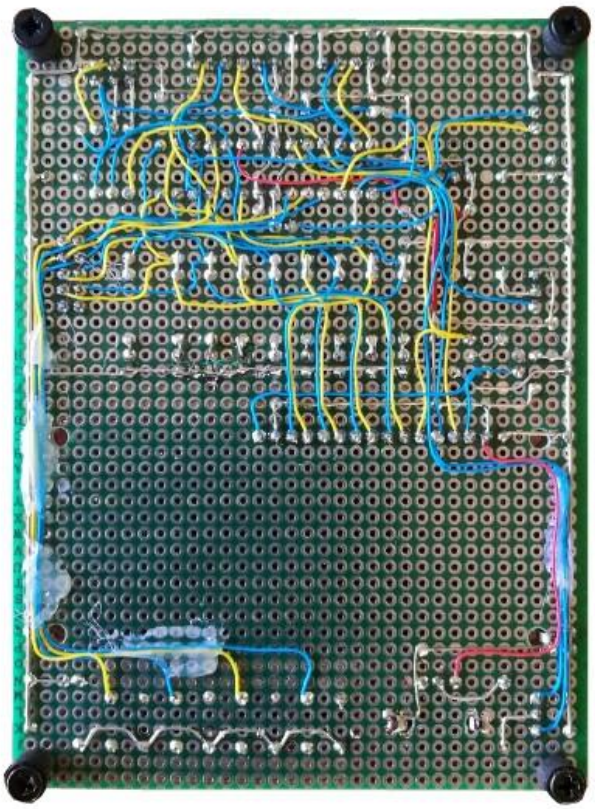
Embedded Development Board

Design, Construction and Operation

Michael J. Bauer
Swinburne University of Technology
(PAVE Division, TET Engineering Dept)
Melbourne, Australia







Foreword

Construction of prototype circuit boards for electronic apparatus is an essential skill for any engineer or technician planning a career in the electronics and communications industries.

Building your own microcontroller development board will not only improve your workshop skills, but the completed product will serve well as a hardware platform for learning embedded programming skills. That is its primary intended purpose.

The AVR-BED is designed to be largely compatible with the "AVR-Pad" board used at Swinburne in microcontroller and embedded systems study units. Unlike the AVR-Pad, the AVR-BED's wiring is permanent. Once wired up, it never needs to be re-patched to suit different projects.

Having your own development platform avoids the need to re-wire a shared AVR-Pad, and/or to fix faults with it, at the start of each practical session. You can also work at home at your own convenience to improve your learning experience.

All of the parts needed to build an AVR-BED are available at ridiculously low prices from online suppliers via AliExpress, eBay, etc. Note that postage from Chinese suppliers, although very cheap (sometimes free), can take a while, typically 2 ~ 4 weeks. (See Parts List.)

Addendum (v2)

To make construction even simpler, an alternative design for the AVR-BED is based on the Arduino "Nano" board (rev 3). Compatible clones are available for less than $3.00 (AUD). The Nano board replaces many components on the AVR-BED, including the MCU chip (ATmega328P), USB-serial bridge, ISP programming port and Reset button.

Refer to the Addendum (starting on page 13) for more details.

References

[1] AVR-BED Library Reference Manual

[2] AVR-BED Test & Demo program (File: avrbed_test_demo_v#.#.hex)

[3] AVR-BED Code Library – Zip folder containing AVR-BED doc's and code samples … (download from www.mjbauer.biz/mjb_resources.htm)

[4] AVR-BED Nano / AVR X-mini Code Library (incl. program samples) … (download from www.mjbauer.biz/mjb_resources.htm)

[5] ATmega48/88/168 Data-sheet (download from Atmel's website)

Design

In common with Swinburne's "AVR-Pad", the AVR-BED is based on Atmel's ATmega88PA microcontroller. The (pin-compatible) ATmega328P may be substituted if more memory is needed. The board incorporates a 2-line x 16-character LCD module, a strip of 8 LEDs, four push-buttons (labelled A, B, C & D) and a rotary potentiometer providing a 0..5V analogue signal source. One extra LED is provided for use as a "debug" indicator or whatever.

In addition, the AVR-BED incorporates a USB-serial adapter (FTDI FT232RL breakout board) interfaced to the ATmega88 on-chip UART, allowing serial data communication with a host PC. The USB connector (mini-B) provides 5V DC power to the board. Applications for the USB-serial port, using a PC terminal emulator (e.g. PuTTY) include:-

- Program debugging aid ("trace" output),
- Data logging to the host PC,
- Command-line user-interface (CLI).

Contrary to the AVR-Pad, the AVR-BED does not have an on-board 4x4 keypad, LM335 temperature sensor, SPI DAC, low-pass RC filter, or piezo transducer (beeper). These and other devices can be connected externally, if required, via pin headers fitted on the board. A pin header also makes the IIC peripheral bus signals (SCL, SDA) available for I/O expansion.

The MCU pin allocations and peripheral interconnection scheme are designed so that all of the on-board resources (and external resources) are able to operate "concurrently", assuming appropriate driver software. A peripheral function library has been developed to support operation of on-board resources, plus an add-on 4x4 keypad. The function library helps to flatten the "learning curve" for beginners in embedded C programming.

The LED strip (8 LEDs connected to Port B) may be driven by writing directly to PORTB (output register). Other peripherals which share Port B (e.g. LCD, keypad) will not disrupt the LED display, provided that the other driver functions restore the bits in PORTB on exit and are not executed excessively frequently. The author's AVR-BED library behaves in this manner.

Pin PD7 is dedicated to the on-board "debug" LED, typically used for a "heart-beat" (1Hz pulse).

The potentiometer may be jumpered to either PC1/ADC1 or PC3/ADC3, or disconnected.

There is a jumper option to connect PD6/OC0A to the LCD backlight switch for control of brightness, or it may be jumpered to enable the LCD backlight always.

Six I/O pins are brought out to pin headers for external use: PC2, PC3, PC4, PC5, PD5 & PD6. These may be used variously for GPIO, PWM output, or ADC input. PC4/SDA and PC5/SCL may be configured to run an IIC peripheral bus, allowing a variety of external devices to be connected at the same time. The IIC bus SIL4 header also provides +5V and GND pins.

PC2/ADC2 and PC3/ADC3 are available for external analogue input or GPIO. Pins PC4 and PC5 may also be used as ADC inputs, or GPIO, provided the IIC peripheral bus is not needed.

Pin PD6/OC0A may be used for external GPIO or PWM output. Pin PD5/OC0B may also be used for PWM output, or GPIO, provided that neither button-D nor the add-on 4x4 keypad are needed.

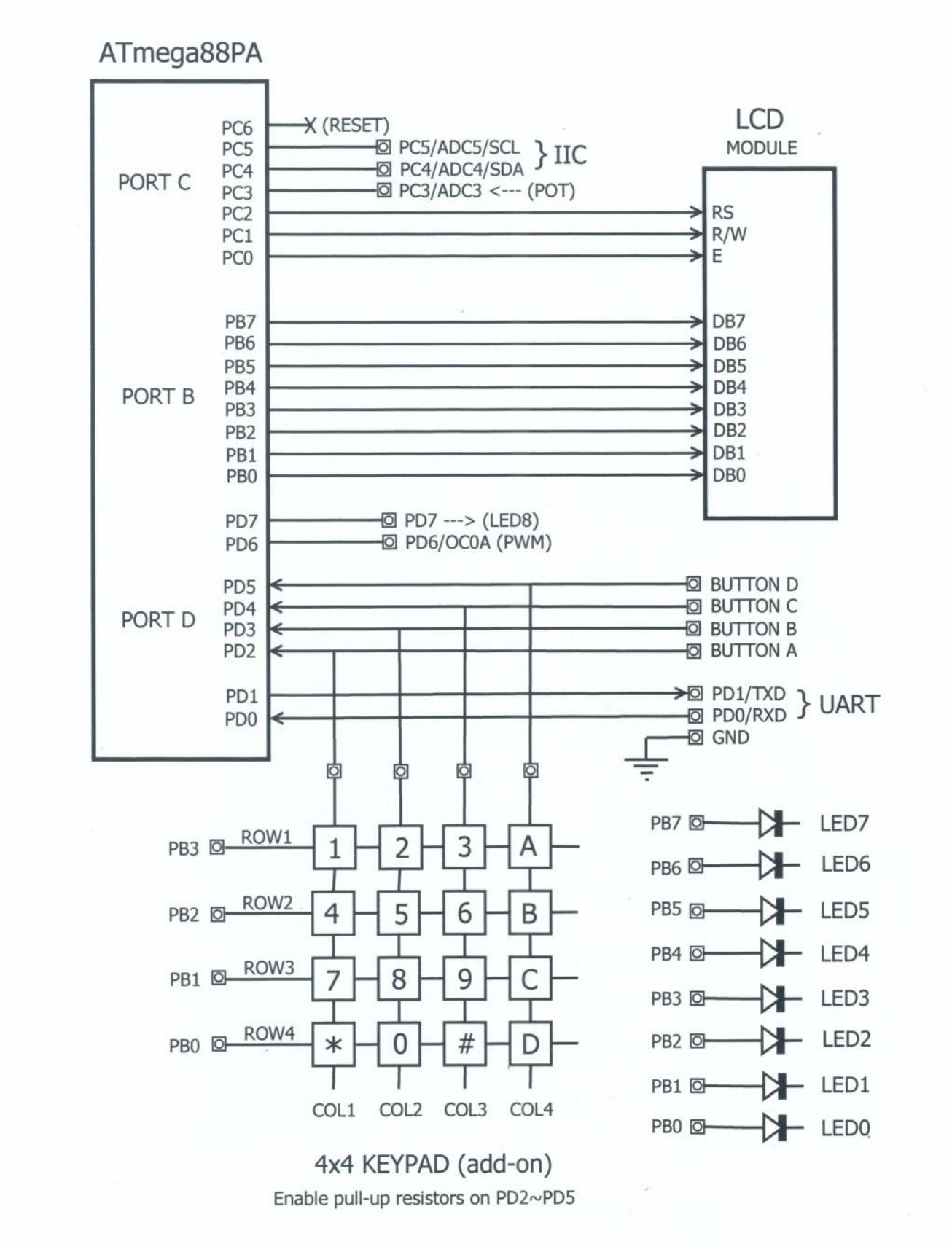


Fig. 1 – AVR-BED Schematic

Block diagram showing port pin assignments and peripheral connections.

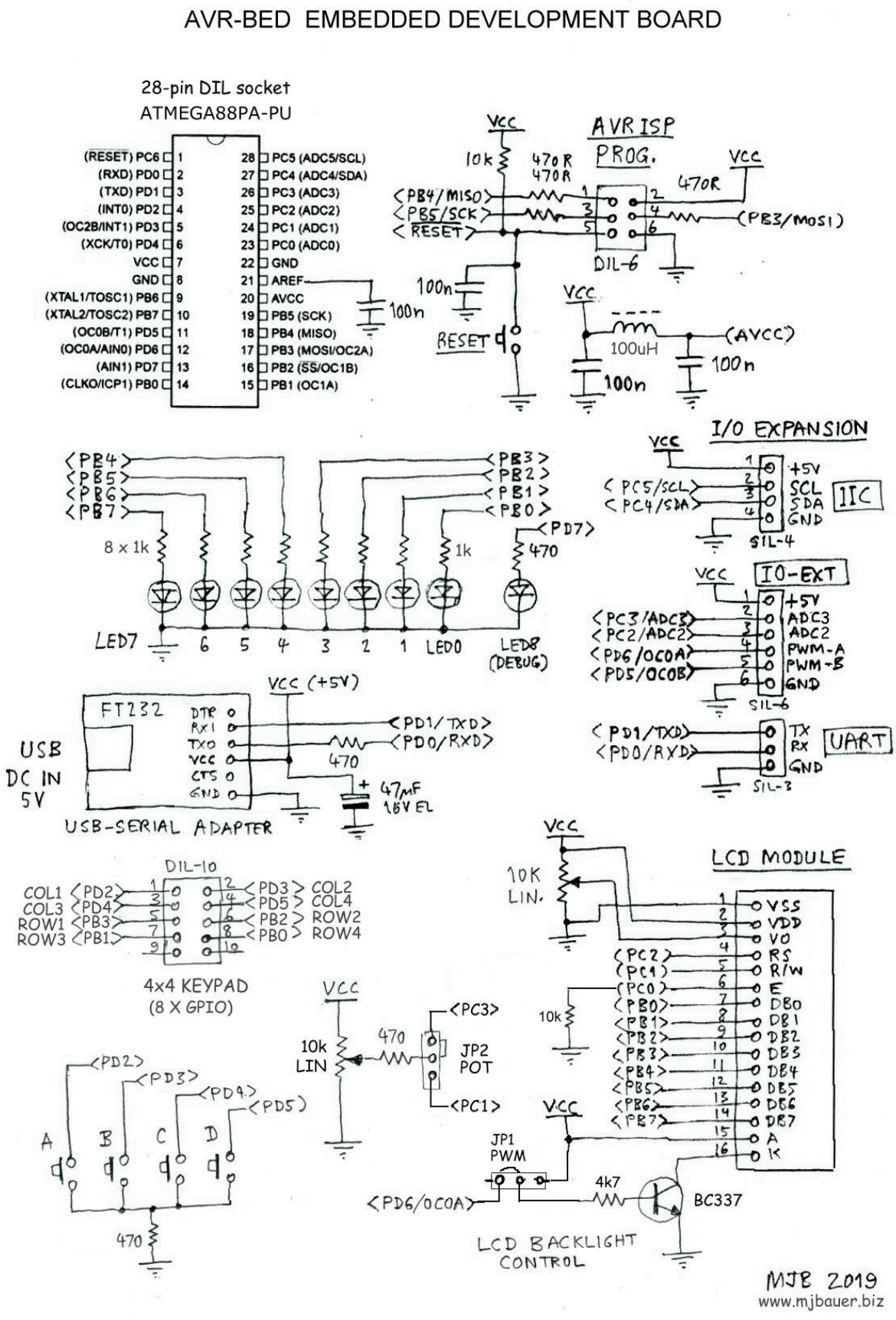
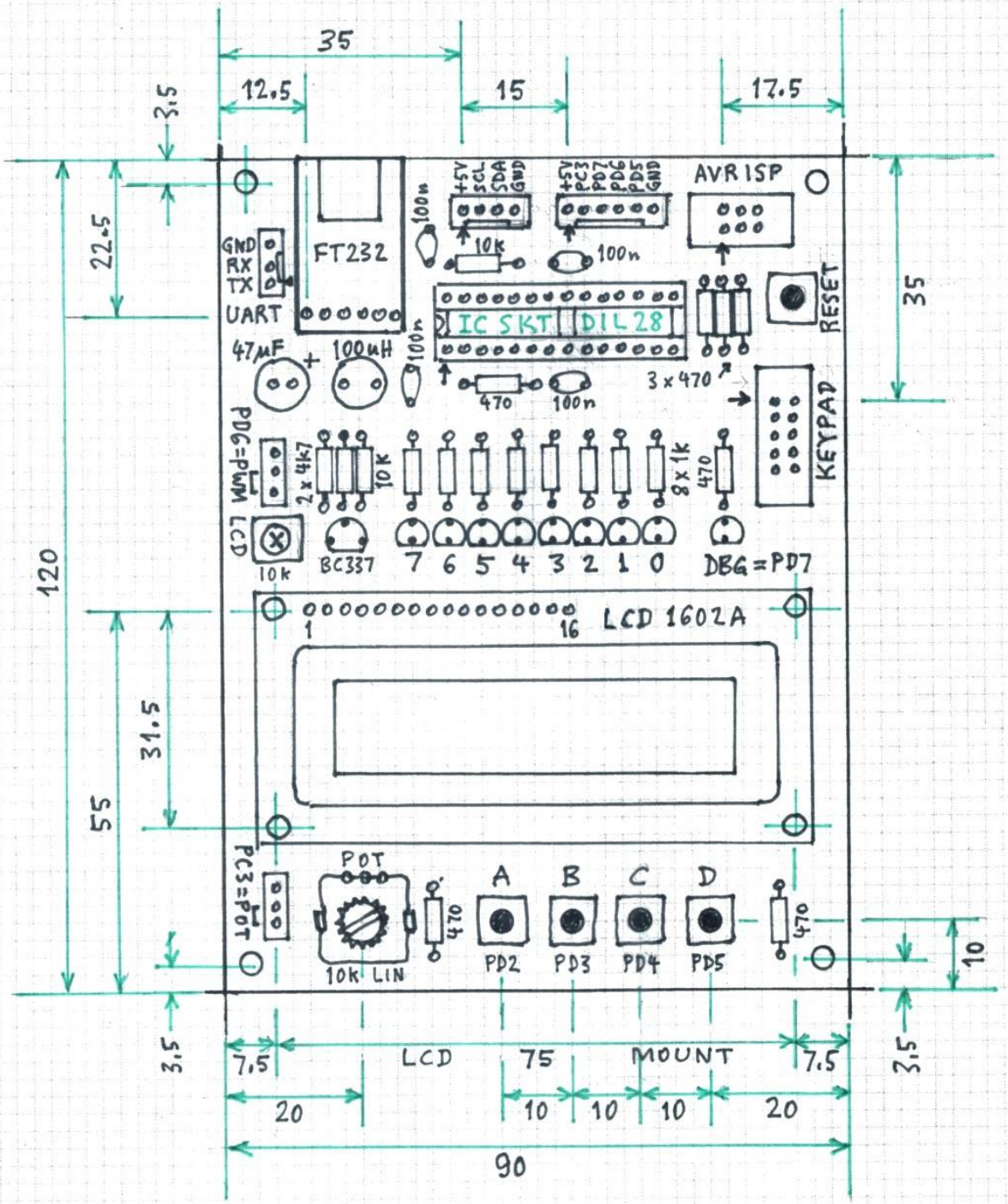


Fig. 2 – AVR-BED Circuit Diagram

AVR-BED

PARTS LAYOUT + DIMENSIONS



MJB 2019 www.mjbauer.biz

Construction

The AVR-BED is built on a piece of prototyping board of the sort that has a matrix of isolated pads on a 2.54mm (0.1 inch) grid. Ideally, use a double-sided plated-through board made of fibre-glass (as opposed to phenolic/paper material which is inferior). These boards are readily available in size 90 x 150 mm. The board should be cut to measure 90 x 120 mm.

Drill four 3.5mm mounting holes in the corners, centred 3.5mm in from the edges of the board.

Referring to the layout drawing above, mark positions of mounting holes for the LCD module and drill 3mm holes there.

Mount the LCD module on the board using plastic spacers about 3 ~ 5mm high.

Feed bare tinned copper wires (24 ~ 26 AWG, length ~20mm) through the 16 connector pads, each passing through an aligned pad on the proto board. Bend the wires on top of the LCD module and under the proto board to keep them in place. Solder the wires on the LCD pads and on the proto board pads underneath. Trim excess wire from soldered pads.

Alternatively, a single-row 16-way pin header may be used to connect the LCD module to the board, but this would be more difficult to remove if the LCD module became faulty and needed to be replaced. On the other hand, there is no need for screws or spacers with this method.

Because the board will be upside-down often while soldering wires on the underside, it is recommended to attach temporary standoffs (6mm diameter, 20 ~ 30mm long) in each corner of the board, on the top-side, so that it rests stably upside-down on your workbench. Better still, if there is one available, use a circuit board vice-stand which holds the board in any desired position for soldering.

Next mount the USB-serial adapter module. Annoyingly, these usually come with a right-angle strip header already soldered in place. It has to be removed. The trick is to remove each pin individually. Apply heat with your soldering iron to a pin on the underside of the board. At the same time, grab the pin on the top-side with a small pair of pliers and pull it out.

Fix the USB adapter in place using a small piece of double-sided foam mounting tape of the sort used to mount mirrors, etc, on walls. Make sure the 6 connector pads are aligned with pads on the proto board. In similar manner to the LCD module, solder wires from the USB module to corresponding pads on the proto board. (Or use the pins that were removed.)

Proceed to solder remaining components in place, as shown in the layout drawing. Be sure to get the orientation correct for polarized components, including connectors. Pin 1 of each connector is marked with an arrow on the layout drawing. Use an IC socket (28-pin DIL narrow) for the ATmega88 MCU and don't fit the MCU in the socket until all wiring is completed and checked. Keep offcuts of wire trimmed from component leads. These come in handy for short wiring hops.

The LEDs should be mounted such that they are raised 2 ~ 3mm off the board. This prevents strain on the leads caused by thermal stress (after soldering) which can destroy a LED. Suggestion: To make standoffs for LEDs, cut 9 short lengths (~3mm) off some PVC insulation sleeving (2.5mm inside diameter) which may be stripped from suitable cable.

Referring now to the underside wiring diagram, run bare tinned copper wire around the perimeter of the board, following the outermost pads, steering clear of mounting holes. This is the GND rail. Complete all connections to GND with bare wire, as shown in the diagram.

Most of the connections to Vcc (+5V) can also be made using bare wire. Otherwise, use hook-up wire on the underside, or a wire link on the top side, in either case using insulated wire.

Signal wiring is best done with 30AWG insulated hook-up wire on the underside of the board. Kynar insulation is preferable, but expensive and hard to find these days. PVC insulated wire is acceptable and much cheaper. Get a few different colours. A few metres will suffice.

Carefully following the schematic diagram, complete the signal wiring. Cut hook-up wires a few mm longer than direct point-to-point distance, so they can be moved out of the way when adding further wires. Strip about 1.5 ~ 2mm of insulation from each end, taking care not to nick the wire as this will likely cause it to break off at the solder joint.

Where there is more than one wire to be soldered to the same pad, they need to be held in place and soldered together. There are many connections to port B pins. The LED series resistor leads can be trimmed so that there is about 2.5mm excess length. The wire stubs can be bent over to hold one or more wires in place on the pad while soldering. (See magnified view of LED resistor pads.) Avoid wiggling existing wires about too much, because they break off easily.

When done wiring, it is highly recommended to check it for point-to-point continuity and for short-circuits using a multi-meter on the low Ohms range (or beeper). In particular, check that Vcc is not shorted to GND! Then fit the ATmega88 MCU chip in its socket. Set the LCD contrast adjust trimmer to about half way. Place a jumper on the "LCD backlight" header so that the LCD backlight is always ON, i.e. not controlled by PD6.

Now it's time for the smoke test. Connect your AVR-BED to a 5V power source via the USB port. The LCD backlight should be on. If not, remove the power cable and trouble-shoot the problem. Otherwise, check with a DMM that +5V is present on all Vcc points, starting at the FT232 module. Double check voltages on Vcc and GND pins on the AVRISP pin header. If the DC voltages are correct and you don't see any smoke, your AVR-BED is ready for testing with software.

Operation

In the AVR-BED documentation pack, you should find a test program. This is a file named "avrbed_test_demo.hex". You'll need Atmel Studio 7 IDE software installed on your PC or Mac. You will also need a programming tool, i.e. Atmel AVR-ISP mkII, or compatible. Chinese clones are available from online suppliers for about $15~$20 plus postage.

NB: The AVR-ISP USB driver incorporated in the Atmel Studio install does not support the cheap Chinese clones, but there is a 3rd-party driver available which does. Details on where to find the driver software and installation instructions may be found in the AVR-BED doco pack.

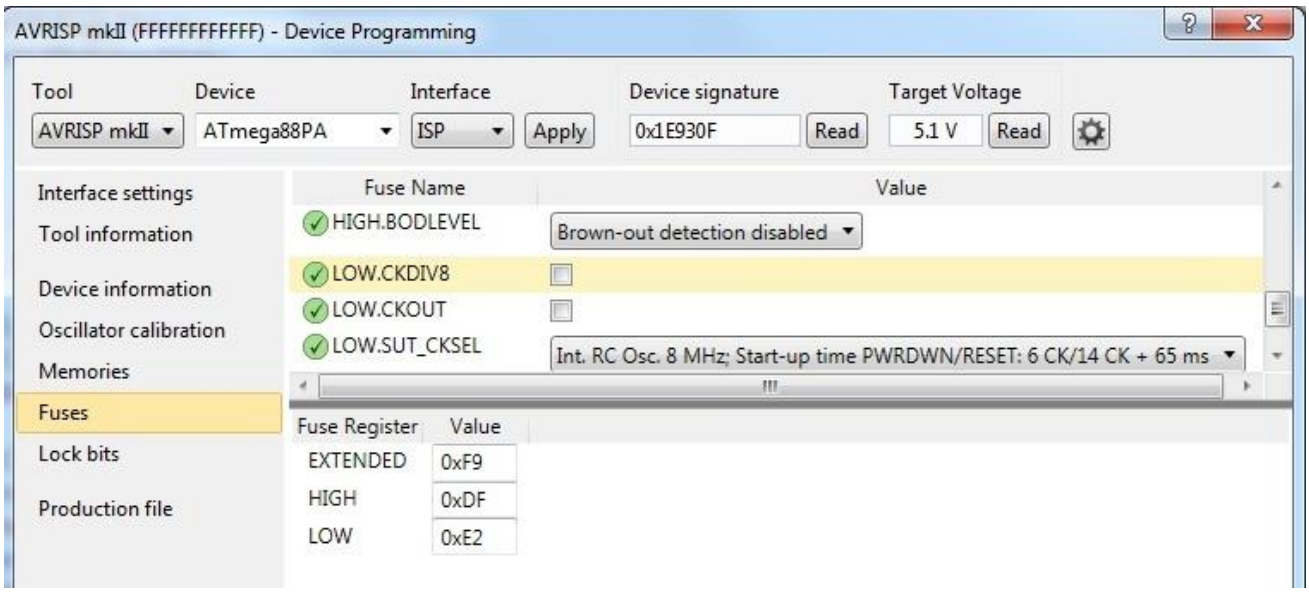
How to load the test program (hex object file) into Atmel Studio 7 and download it to the ATmega88 MCU flash program memory…

Plug your AVRISP mkII (clone) into a USB port on the host PC then start Atmel Studio. Connect the ISP ribbon-cable plug to the ISP header on your AVR-BED and power up the board.

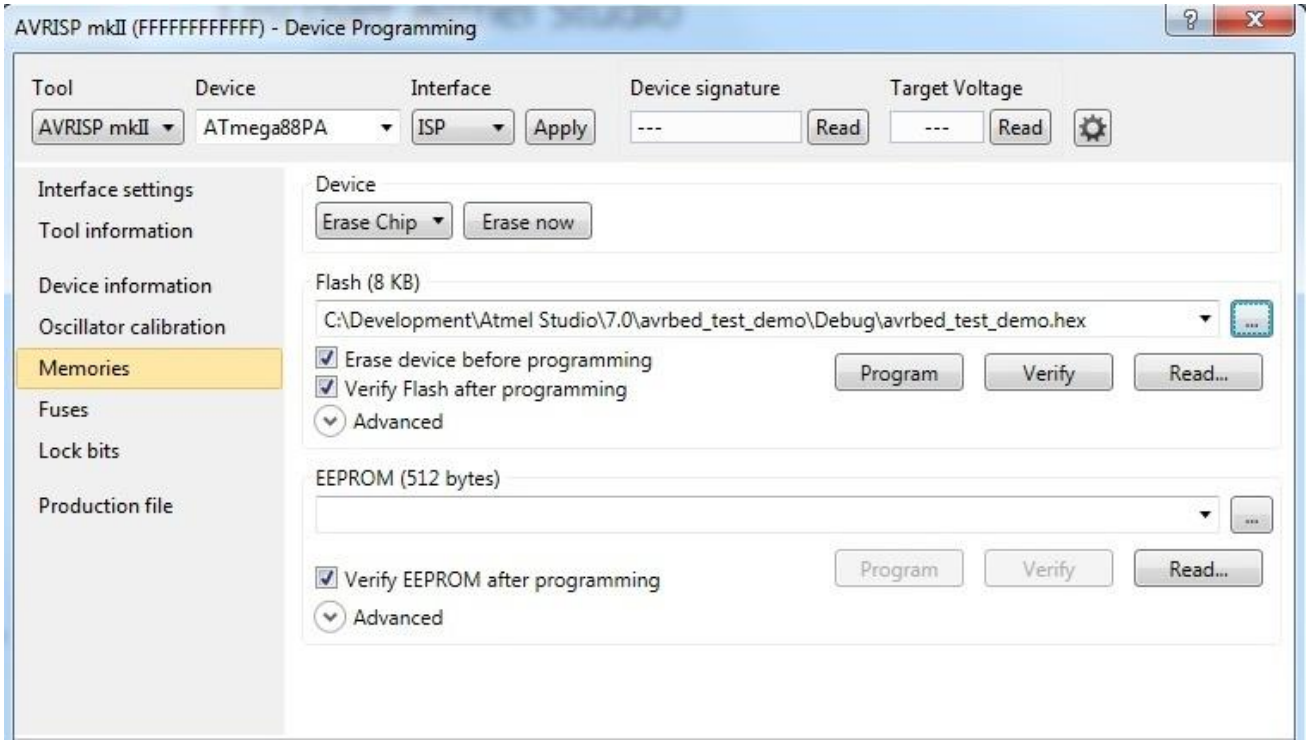
From the Tools menu, select Device Programming. A pop-up window entitled "Device Programming" should appear with "Interface Settings" selected from the left side menu. Enter the target device, ATMEGA88PA, from the drop-down list. Click the Apply button. You should see a slider to set the ISP clock rate. If it is shown, leave the ISP Clock set to 125kHz. If not, the problem is most likely the USB driver is not installed properly.

Now click the “Read” button under Device Signature. The AVRISP should fetch an ID number out of the ATmega88 MCU and display it (6 hex digits). If it fails, there could be a fault in the ISP header wiring, but first check that the ISP plug is in the right way around.

Select “Fuses” from the left side menu. Scroll down the list of Fuse Names until you see the name “LOW.CKDIV8”. Clear the tick-box if not already cleared. This makes the CPU internal RC oscillator run at 8MHz, whereas the default is 1MHz.



Next select “Memories” from the left side menu. Under the heading “Flash (8 KB)” there is a field showing the selected object file. Click the browse button (at right end of this field) and navigate to the folder where you copied the object file: avrbed_test_demo.hex. Select this file.



Click the “Program” button. The program should download to your AVR-BED. When the download is finished, the program should start running. It’s wise to disconnect the AVRISP tool and press the RESET button on the board to ensure the tool doesn’t affect program operation. (Note: The ISP interface shares a few I/O pins on Port B of the MCU.)

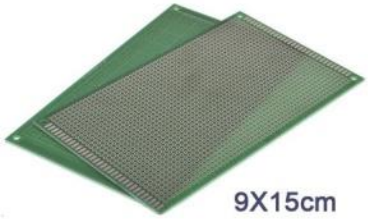
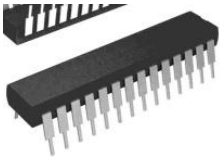
When you have the test program running, follow the prompts on the LCD screen, assuming the LCD module is functioning properly. If not, trouble-shoot the problem before proceeding. Refer to comments in the C source code to understand how the program operates.

The program works best with a terminal emulator app (e.g. “PuTTY”) running on the host PC. When you plug the AVR-BED USB cable into the host PC for the first time, it should install driver software for the USB-serial bridge (FTDI FT232RL) automatically. If this doesn’t happen, perhaps because you’re running an older version of Windows, you might need to download a driver from FTDI’s website and install it manually.

Assuming the USB driver installed OK, open up Windows “Device Manager” utility and click on “Ports (COM & LPT)”. You should see the FTDI USB-serial device listed. Note the number of the associated “COM” port and use this number when setting up PuTTY for serial comm’s. Set the Baud rate in PuTTY to 9600. When a PuTTY session is started, press the RESET button on your AVR-BED. A start-up message should appear in the terminal window.

Have fun!

Table 1. AVR-BED Parts List

Component	Description	Qty	Cost A$
	Prototyping board 90 x 150 mm Dot matrix, 2.54mm (0.1”) grid Fibreglass (1.5mm) Double-sided, plated through holes	1	2.60 (free postage)
	USB-serial adapter module FTDI FT232RL breakout board with USB mini-B connector	1	2.60 (+0.24)
	LCD module 1602A 2 line x 16 characters LED backlight (white on blue, or black on yellow/green) HD44780 (compatible) controller Dimensions: 75 x 36 mm Connector strip (16-way) upper LHS	1	1.20 (+1.30)
	IC socket, 28 pin, DIL, narrow (0.3”)	1	2.70 (qty 5)
	Microcontroller ATMEGA88PA-PU (or ATMEGA328P-PU) 28-pin plastic DIP	1	4.40 (+0 pp) (qty 2)
	LED, 3mm round, diffused lens 8 x blue + 1 x red, or… 8 x red + 1 x green, or… (your choice of colours)	9	1.00 (+0) (qty 100)
	Potentiometer, rotary, 10k ohm B-taper (linear) PCB mounting, vertical 12 x 12 mm (approx.)	1	1.50 (+0) (qty 10)
	Push-button switch, SPST, tactile PCB mount (straight) 6 x 6 mm Short actuator (2 ~ 4mm)	5	1.00 (qty 20)
	Prototyping (wire wrap) wire Insulated (Kynar or PVC) 30AWG Assorted colours (10 colours x 2m)	5m	2.00 (+2.00) (20m)

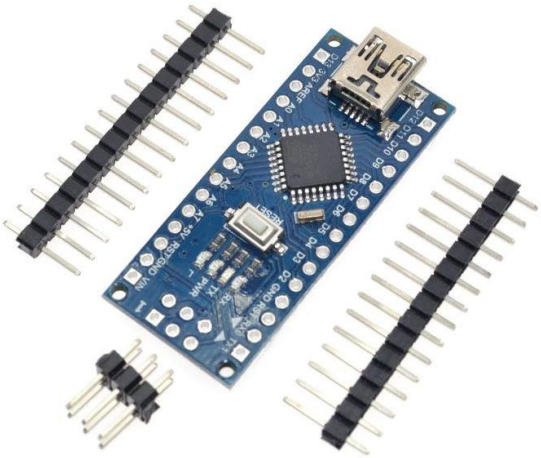
	USB cable Type-A (host) to mini-B (device) Length 1.0 ~ 1.5m	1	0.80 (+0)
	Keypad 16 keys wired in 4x4 matrix 8-way connector strip Telephone dialler layout	1	2.60 (+1.12)
	Programming Tool Atmel AVRISP mkII (compatible*) with USB cable and 10-to-6-pin ISP adapter (*Chinese clone, cost AU $15 ~ $20)	1	16.30 (+7.70)
	Wire jumpers, set of 20 Length 150 ~ 200 mm Female contacts both ends Assorted colours (for connecting external I/O devices)	1 set	1.17 (+1.88)
Sundry components available from Jaycar, etc.	PCB pin header, 6 way, double row (2x3)	1	9.00 (+ pp)
	PCB pin header, 10 way, double row (2x5)	1	
	PCB pin header, 3 way, single row, shroud	1	
	PCB pin header, 4 way, single row, shroud	1	
	PCB pin header, 6 way, single row, shroud	1	
	PCB pin header, 3 way, single row, plain	2	
	Transistor, BC337 (TO92 pkg)	1	
	Trim pot. 10k linear, single turn, PCB horiz.	1	
	Capacitor, mono ceramic, 100nF (50V)	4	
	Capacitor, electro. RB 47uF (16~63V)	1	
	Inductor / RFC, ferrite, 100uH (200mA)	1	
	Resistor, metal film, 470 ohm, ¼ W	8	
	Resistor, metal film, 1k ohm, ¼ W	8	
	Resistor, metal film, 4.7k ohm, ¼ W	1	
	Resistor, metal film, 10k ohm, ¼ W	2	
	Total cost (estimate):		**$60.00**

Addendum – “AVR-BED Nano” (alternative build)

To make construction even simpler, an alternative design for the AVR-BED is based on the Arduino “Nano” board (rev 3). Compatible clones are available for less than $3.00 (AUD). The Nano board replaces many components on the AVR-BED, including the MCU chip (ATmega328P), USB-serial bridge, ISP programming port and Reset button. This design greatly reduces the number of hook-up wires needed to be soldered on the underside of the board.

The “AVR-BED Nano” (aka “Nano-BED”) works in the Arduino environment, of course, because that is what it was designed to do. However, the Nano can be programmed in the Atmel Studio IDE as well, without requiring any additional hardware Programming Tool (i.e. an AVRISP mkII). See Appendix A for instructions on how to configure Atmel Studio 7 to program the Nano board using the on-chip Arduino bootloader.

The picture below shows the envisaged AVR-BED Nano prototype board layout. Note that this picture is not a photograph of a real Nano-BED, but rather a (badly) “Photoshopped” image. The construction method is similar to the original AVR-BED. The Arduino Nano usually comes with pin headers intended to be soldered onto the board. The base (prototyping) board should have soldered onto it a pair of SIL strip headers with female contacts (like an IC socket) into which the Nano board is plugged.



The suggested Nano-BED layout (pictured) leaves plenty of spare room for additional I/O devices, for example: LM335 temperature sensor, SPI DAC & low-pass filter, piezo beeper, etc.

Following is the wiring diagram for the AVR-BED Nano. The diagram doesn’t include a 10-way DIL pin header for connection of a numeric keypad as in the original AVR-BED, but this may be added if needed. The 10-way header may be used for other I/O expansion purposes as well.

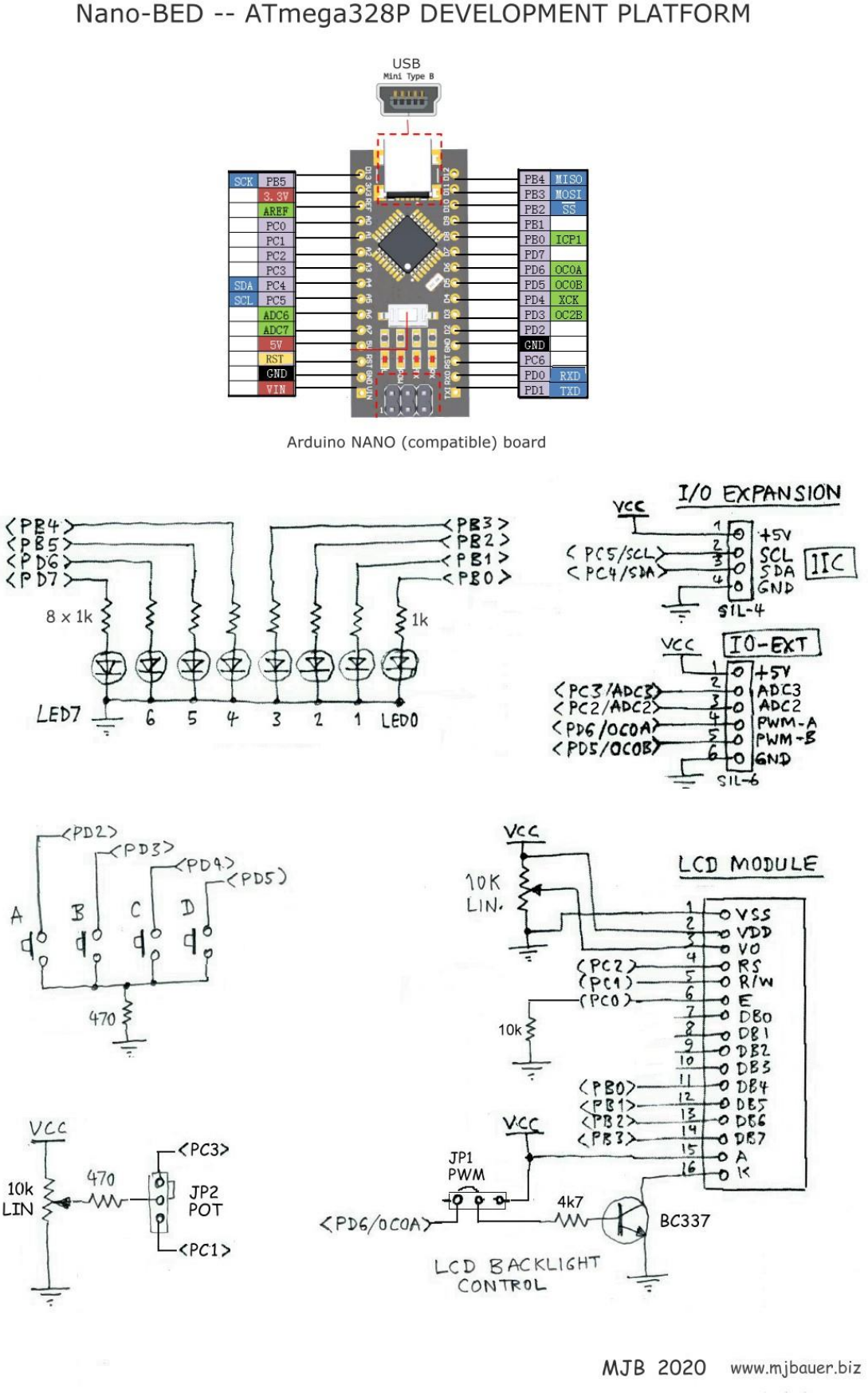


Fig. A1 – "AVR-BED Nano" Circuit Diagram

The only significant differences between the Nano-BED and the original AVR-BED are summarized in this table:

Platform	CPU clock freq.	PB7 pin alloc.	PB6 pin alloc.
AVR-BED	8 MHz (internal osc.)	GPIO (LED, LCD)	GPIO (LED, LCD)
Nano-BED	16 MHz (crystal)	N/A (XTAL2)	N/A (XTAL1)

Note in particular that the Nano board uses 2 pins of port B for the system clock (crystal). This precludes the use of port B to interface the LCD module in 8-bit bus mode. Fortunately, the same LCD module (1602A) can be interfaced to the ATmega328P in 4-bit bus mode, as shown in the circuit diagram (Fig. A1). With the exception of port B pins PB6 and PB7, all I/O pin allocations are compatible with the AVR-BED.

The change in LCD interface mode requires a modified peripheral function library to suit the Nano-BED. A compatible library is available on the author's website. (See References on page 1.) The same library can also be used with the Atmel ATmega328P "Xplained Mini" board.

Please be aware that the AVR-BED, Arduino Uno, Nano and Atmel "X-mini" boards all use different methods for programming the target AVR micro-controller. Refer to manufacturer's instructions (user guides, etc) for programming and (where supported) debugging facilities.

Arduino boards, although designed for programming in the Arduino IDE software environment, can also be programmed directly from within the Atmel Studio 7 IDE, without requiring any additional hardware device (AVRISP mkII). However, it is necessary to create an "external programming tool" (software configuration) in the IDE to drive the Arduino bootloader. Instructions to implement this facility can be found in Appendix A; also in the AVR-BED download package.

The Nano's on-board USB-serial bridge can be used for serial data communications in a user application program, in much the same way as the USB-serial adapter on the original AVR-BED. The library test program shows how to implement serial comm's using the UART functions to transfer data to and from a PC terminal application (PuTTY).

Appendix A – How to Program a Nano board under Atmel Studio

Programming the target device (Atmega328P) via the Nano on-board USB-serial bridge using the Arduino bootloader, i.e. without using an AVRISP mkII or any other additional hardware programming tool.

First you need to install Arduino IDE software on your PC, or at least the "AVRdude" components included in the Arduino install. (It's probably easier to install the complete Arduino package.)

Open Atmel Studio, click in the menu "Tools/External tools".

You will see a dialog box asking for some parameters, as follows...

In Title, enter: "`Program Nano board`" or any name you prefer.

In Command: `C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude.exe`

This may vary according to the installation directory of the Arduino IDE. Check the **avrdude.exe** directory in your installation.

In Arguments (all on one line):

```
-C "C:\Program Files (x86)\Arduino\hardware\tools\avr\etc\avrdude.conf" -p atmega328p
-c arduino -P COM4 -b 57600 -U flash:w:"$(ProjectDir)Debug\$(TargetName).hex":i
```

Again this can vary according to the installation directory of the Arduino IDE.

Replace COM4 (in the Arguments field) with the actual COM port your Nano board is connected to, as can be found in Windows **Device Manager**.

Check "Use output window" box. Click OK.

Done... You should see the new option "Program Nano board" in the tools menu.

How to Program an Arduino UNO board under Atmel Studio

Arduino Uno boards can be programmed under Atmel Studio using the same technique as the Nano board, as described above, except for some unfathomable reason, the serial bootloader Baud rate is different. In the Arguments field, replace "`-b 57600`" with "`-b 115200`".