Using ARDUINO with the Atmel ATtiny85 Microcontroller

No doubt you have seen heaps of interesting applications for Arduino boards. But what if you want to use some of those ideas in a design of your own using the Atmel ATtiny85 microcontroller? It actually is quite easy and you can use Arduino software. Interested? Lawrence Billson takes up the story.

The ATtiny **microcontrollers from Atmel** are an ideal way to add simple programmable logic to your circuits. For example, the ATtiny85: it costs just a couple of dollars or so and with only eight pins it is an easy way to get started with adding a microcontroller to your own design.

And if you are not a software guru, the chip can be programmed using the free Arduino IDE (integrated development environment), making short work of simple electronics projects.

The ATtiny85 chip has five general purpose input-output (GPIO) pins. Three of them are capable of reading analog voltages while the other two are capable of "analog" output – more on that later.

Other than writing your program to the chip's built-in flash memory, all it really needs is a ground (0V) connection and a voltage of +2.7 to +5.5V on its V_{CC} pin (8).

With a few lines of code, the ATtiny85 can replace numerous analog or digital ICs and give your design the flexibility of being reprogrammable.

Although the Arduino IDE allows you to program in C (technically C++), knowing the language isn't critical. With the very large "community" built around the platform, many applications can be programmed using "cut and paste" methods.

Much of the Arduino code you find on the 'net will run on the ATtiny85 with little or no modification at all.

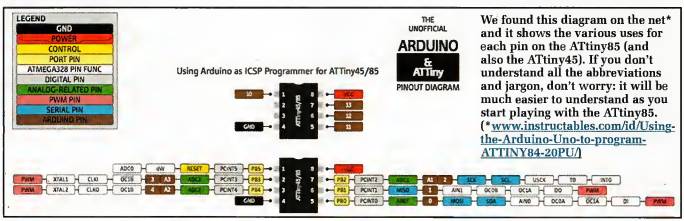
On paper, the ATtiny85 specs may seem underwhelming. It is an 8-bit micro with 8KB of rewritable flash memory for storing and executing your program, 512 bytes of EEPROM for storing things like configuration or calibration variables from your project and another whopping 512 bytes of RAM.

But don't let the meagre sounding specs fool you.

Using the freeware Arduino IDE, your code (or cut & paste effort) is transformed into tight, fast machine language using the built-in avr-gcc compiler.

In times gone past, a compiler for embedded processors was difficult to use and cost thousands of dollars – a huge barrier to entry. As well as being free, the Arduino software hides all of the 'engine room' parts like the compiler, chip 'fuses' and linker scripts.

Although the Arduino IDE is tailored for Arduino (or



We found this diagram on the net* and it shows the various uses for each pin on the ATtiny85 (and also the ATtiny45). If you don't understand all the abbreviations and jargon, don't worry: it will be much easier to understand as you start playing with the ATtiny85. (*www.instructables.com/id/Using-the-Arduino-Uno-to-program-ATTINY84-20PU/)

clone) boards, with only a few minor tweaks, it'll program your ATtiny chips nicely.

Development history

The ATtiny85 is based around Atmel's AVR architecture. This began life as a graduate project by two students from the University of Norway in 1996. They were looking to build a microcontroller that was based around flash memory.

Using flash memory allows a microcontroller's code to be changed without needing to expose chips to UV light or replace external ROMs.

Another advantage was that a product could be manufactured with a blank chip and programmed in the factory or field. If you pull apart many mass-produced products you may well find ICSP (In-Circuit Serial Programming) pads or pins on circuit boards for just this purpose.

Another problem the Norwegian students were attempting to solve was that of 'compiler bloat'. Chips like the Intel 8051, which was the dominant microcontroller at the time, use a complex instruction set (CISC) architecture.

While lending themselves to being programmed with assembly language, compiled languages would often become bloated as the compiler turned the program into machine language. This 'bloat' caused two problems: the code would become quite large and also quite slow to run.

As the AVR architecture took shape, the students worked closely with the authors of a professional compiler named "IAR". Being developed in parallel, the AVR evolved to be very good for running high level compiled languages.

Classified as a RISC (reduced instruction set computer), it allows for most instructions to be executed in a single clock cycle and it hasn't changed much in the last 20 years.

Knowing that flash memory was a key component in their design, the students from Norway knew they would need to take their chip design to a company that had experience making flash memory. At the time there were two – one based in Japan and Atmel in the United States. The Norwegians decided they spoke better English than Japanese and therefore approached Atmel.

Since their release in 1997, Atmel have sold hundreds of millions of AVRs. They are among the most popular microcontrollers being used by industry. Earlier this year, rival company Microchip (makers of the successful PIC microcontrollers) struck a deal to buy Atmel.

While the ink on the contracts isn't yet dry and speculation is rife, it's highly likely they'll keep the AVR line for years to come.

The ATtiny family is designed to be embedded into things. Tear apart a toaster or cordless drill and there's every chance you'll find one inside. They are available in DIP (through-hole) or a variety of surface-mount packages, and are equally at home on a breadboard or a mass-produced product.

In an interview on the excellent "embedded.fm" podcast, Atmel's Andreas Eieland talks about millions of their smaller chips finding their way into home pregnancy testers, of all things!

So what can you do with it? Controlling things like stepper motors and servos is easy, as is gathering data from temperature or humidity sensors. The ATtiny85 shines at smaller automation jobs. Instead of a 555 timer or some logic gates, I'll often grab an ATtiny85 for the same job. As a rule of thumb, if the application has only a couple of inputs and outputs, it might be a good choice.

If your application needs more pins or support for more complicated programs, the Micromite or larger AVR chips may be a better choice.

Getting started – what you'll need

You will need an AVR-specific ICSP programmer. Usually in the form of a USB attached gizmo, the ICSP allows the Arduino software on your computer to write its compiled program into the memory of your chip. The Freetronics unit will do the job well – see below.

As its name implies, the ICSP allows you to program your chip while it's in circuit.

But this is not really practical in the case of the ATtiny85 since most of the I/O pins are used by the ICSP and this will limit what you can connect to them. So it's best to program the chip on a breadboard before embedding it into your circuit.

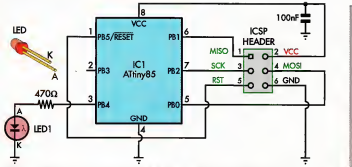
The 6-way connector that's standard on typical ICSPs isn't particularly breadboard-friendly either. So we will make up a simple 6-pin header as an adaptor to connect it to a breadboard.

You'll also need a computer (laptop or desktop) on which to write your programs – any PC that runs Windows, Linux of Mac OSX will be fine. The Arduino IDE can be freely downloaded from **arduino.cc**.

Other than that, you'll need some ATtiny85 chips and you're ready to get started.

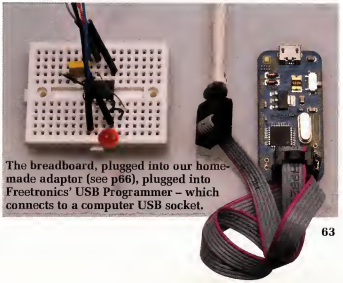
Your first ATtiny85 project

We start with the simple circuit shown in Fig.1. It uses four of the ATtiny85's I/O pins to connect to the ICSP header



SC 02017 **YOUR FIRST ATtiny85 PROJECT**

Fig.1: one chip, one LED and one resistor – you can hardly go wrong! At right is the layout on a mini breadboard.



The breadboard, plugged into our home-made adaptor (see p66), plugged into Freetronics' USB Programmer – which connects to a computer USB socket.

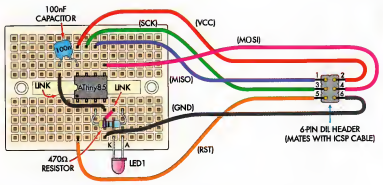
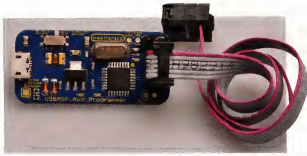


Fig.2: here's the breadboard layout for the Flashing LED project overleaf (Fig.1), along with the wiring for a 6-pin DIL header for programming.



Freetronics' $22 USB ICSP Programmer for AVR & Arduino. The six-pin socket on the end of the IDE cable mates with 6-pin ICSP header pin "plug" we shown you how to make later. This board then plugs into your PC via the micro-USB socket (left edge) and enables you to program the ATtiny85. (www.freetronics.com/usbasp).

socket and one of the remaining I/O pins to drive a LED.

The first program you will use will simply flash that LED and that's all. But you have to start somewhere. The circuit of Fig.1 needs to be made using a small breadboard and we have shown the component layout in Fig.2. So get your parts and a breadboard together. (*See "Using Breadboards" immediately following this feature*).

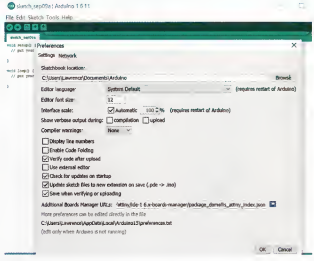
Note that you will need to solder six insulated wires to a 6-pin DIL header and that will provide the connection to the ICSP programmer. We also show a photo of the finished breadboard, ready to hook up to the ICSP and your PC.

Now you need to program the ATtiny85.

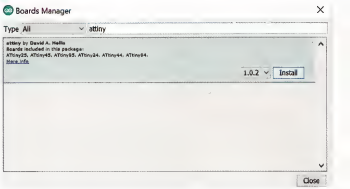
Begin by downloading and installing the latest release of the Arduino IDE. Be sure to say yes to installing all of the recommended drivers that are included with it.

The Arduino software comes ready to work with their officially branded boards. As we'll be using it to program ATtiny85 chip, we'll need to include support for it. You'll only need to do this once.

Once Arduino is installed, open the Preferences window and find the section for "Additional Boards Manager URLs" – paste in https://raw.githubusercontent.com/damellis/ATtiny/ide-1.6.x-boards-manager/package_damellis_ATtiny_index.json and click OK.



Under the "Tools" menu, select "Board:" , then click on "Boards Manager". Type ATtiny in the search box. Select the ATtiny library by David A. Mellis, and click 'Install'.

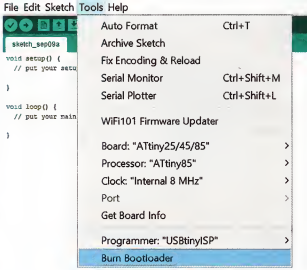


From now on, your Arduino IDE will know about the ATtiny85 chips and be ready to program them.

You'll need to tell Arduino about the chip we want to program. Under the "Tools" menu, select "Board <Name>" and you'll now see 'ATtiny' as an option. Select this. You must now go back in and give it some more details – in this example set:

Board - ATtiny
Processor - ATtiny85
Clock - 8MHz (internal)

Be sure to select the internal clock. If you accidentally



select an external clock your ATtiny85 can't be programmed unless you connect an external crystal.

Now we need to tell Arduino what type of ICSP we'll be using. For the Freetronics XC4237, select "USBasp".

Now you can go to "File", select "Examples", "Basics", and open "Blink".

The blink program normally tries to blink an LED connected to pin 13. But your ATtiny85 doesn't have quite that many! We have connected our LED to pin 4 (as in Fig.1), so you will need to change all of the references from "13" to "4".

MISO connects to MISO, MOSI connects to MOSI. Some programmers won't supply any power to the board so you may also need to connect up a power supply or batteries. Other programmers may have a jumper marked V_{OUT} which you can short, thus powering your board from the ICSP. Check with a multimeter to verify your V_{CC} line is between 2.5 and 5.5V.

For each new chip, you'll need to set its fuses. This tells the chip how to behave before it starts running any programs (eg, to use the 8MHz internal oscillator). Click on "Tools" then "Burn Bootloader". Keep an eye out for error messages.

If all has gone well so far, it's time to write your code to the chip Connect your ICSP programmer to the 6-pin header from the breadboard and connect the programmer to your PC.

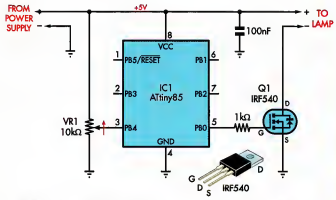
Holding down shift, click on the green arrow. This will compile your code and write it to the chip using the ICSP programmer.

If all has gone well, you'll have a blinking LED on your breadboard. Congratulations.

LED strobe

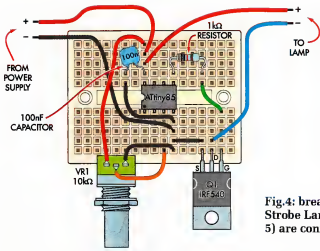
Our next circuit and program is for a simple LED strobe light. You have a wide choice of high-brightness LEDs of various colours for this job but I chose a Jansjo 2W LED lamp from Ikea. It comes with a handy plugpack power supply, to provide the LED with 4.5V DC.

Our ATtiny85 can modulate with an N-channel FET and the circuit is shown in Fig.3. Pin 4 of the ATtiny85 drives the gate of the Mosfet whereas in the previous circuit it just drove a LED via a 470Ω current-limiting resistor. The software is "Ikea_Strobe.ino".



ATTINY85 BASED STROBE LAMP

Fig.3: instead of flashing a LED directly, the strobe circuit drives a Mosfet which in turn drives a more powerful LED. VR1 varies the rate of the flashing LED.



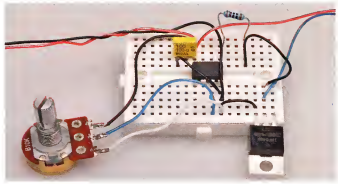


Fig.4: breadboard layout (along with a matching photo) for the ATtiny85 Strobe Lamp. Just remember that all of the north-south holes (in groups of 5) are connected inside the breadboard; all of the east-west holes are not.

But before you wire up the strobe circuit on a breadboard, as shown in Fig.4, you have to load the strobe software into the ATtiny85 using the breadboard layout of Fig.2.

In fact, we suggest you keep that Fig.2 breadboard as your dedicated ATtiny85 programmer.

Before uploading the strobe code, don't forget to "burn bootloader" to your new chip to set its fuses. Once the fuses are set, you can upload your code.

The strobe software task is divided into "start" and "loop" sections. When power is first applied to the micro, the start section is executed – this sets pin 0 as an output and pin 4 as an analog input.

The loop section is then executed. In this, the micro sets pin 0 high (switching on the Mosfet, allowing current to pass from the lamp to the power supply). The micro waits for 5ms and sets pin 0 low; turning off the lamp.

The micro then measures the voltage at the potentiometer wiper. Depending on the position of the potentiometer, the value measured will be between 0 and 1023. The micro then waits for that same number (ie, between 0 and 1023) of milliseconds, allowing the strobe to vary its 'off time'. As soon as this completes, the loop begins anew.

So having built the strobe breadboard of Fig.4, you can plug in your freshly programmed ATTtiny85 chip and you are ready to go.

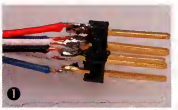
Audio Thermometer

This project makes use of the DS18B20 digital thermometer chip (or probe). Rather than displaying the temperature as a number, it plays a tone corresponding to the relative temperature it measures.

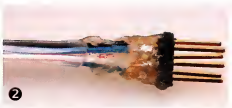
The DS18B20 is available in different package types – most commonly a TO-92 which looks just like a small transistor. It's also available in a waterproof probe suitable

How to make the 6-way ICSP connector

It's easy to make a connector for ICSP – all you need is a length of 2-way pin header (eg, Altronics P-5410) and carefully remove a 3-pin length. The wiring we used came from a length of 4-wire discarded telephone cable (yep, we never throw anything out!) It has colours of red & black (ideal for power) and blue & white (for everything else). You could also use female-male jumper leads and avoid some soldering.



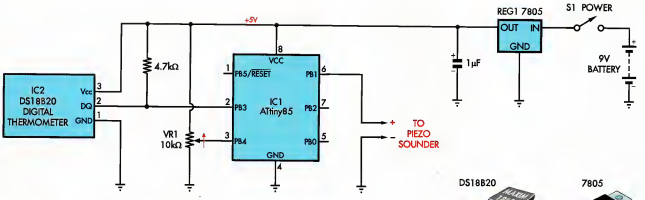
(1) Cut off a 3 x 2-way length of pin header and solder six wires to it. A red wire connects to the + terminal and a black to –; other colours can be what you have available.



(2) Apply a glob of hot melt glue (or silicone sealant if you don't have hot melt) over the soldered pins and back up the wires to keep the wires in position when it is being used. Allow to dry.

(3) Cover with a length of heatshrink tubing, right down onto the glue. This will stop it trying to pull apart as it is inserted and removed from the socket.

(4) Slide some short lengths of white heatshrink over each wire towards the plug, and some longer lengths of heatshrink over the opposite ends of each wire to make them stiffer. With a multimeter, identify which pin goes to which wire and write it on the white heatshrink. Shrink all heatshrink . . . and it's finished!



ATtiny85 BASED AUDIO THERMOMETER

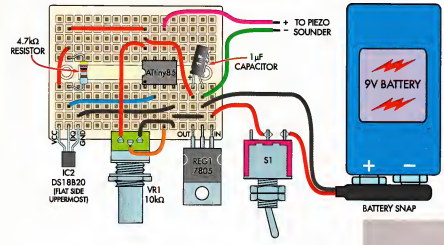


Fig.5 (above): the Thermometer uses a DS18B20, small solid-state digital thermometer chip, which will feed a number sequence to the ATtiny85 representing the temperature it is sensing. The ATtiny85 then generates a tone for the piezo sounder corresponding to the temperature.
Fig.6 (left): the breadboard layout for the audio thermometer. It's a little more complex so make sure the components and wire links, etc, are in the right place. You can also refer to the matching photograph (below).

for immersion into liquids up to about 120°C.

The circuit of the Audio Thermometer is shown in Fig.5 and the breadboard layout is Fig.6.

In this case we are using a 9V battery to power the circuit and this is reduced to 5V for the ATtiny85 and the DS18B20 thermometer.

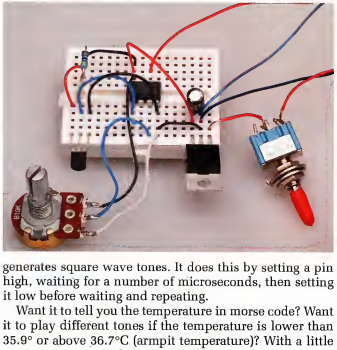
The data line from the DS18B20 is fed into the PB3 input, pin 3 and also pulled high with a 4.7kΩ resistor.

As with most Arduino programs, the Thermometer code is divided into the "Start" and "Loop" sections. An external library of functions is also loaded, to communicate with the DS18B20 thermometer. We simply tell the library which pin it's connected to, and request a temperature reading whenever we want.

The "Start" routine runs once as the chip is powered on. It initialises the DS18B20 and sets the PB1 pin (6) connected to the piezo to be an output. It also sets the pin connected to the potentiometer wiper as an analog input – this is used to vary the range of the tones.

The "Loop" function starts by requesting the temperature from the DS18B20. It then measures the analog value from the potentiometer wiper. The temperature value (reported in °C) can go as low as -55°C. As we'll be turning it into a frequency, we need to ensure it is a positive number. We do this by adding 60. We then multiply this number by the value of the pot to derive a frequency in Hertz.

The tinyTone function is then called to output this frequency to the piezo speaker for 600ms before the loop restarts. As its name implies, tinyTone is a function that



generates square wave tones. It does this by setting a pin high, waiting for a number of microseconds, then setting it low before waiting and repeating.

Want it to tell you the temperature in morse code? Want it to play different tones if the temperature is lower than 35.9° or above 36.7°C (armpit temperature)? With a little experimentation, either of these is quite simple.

As before, you will need to program the ATtiny85 with the breadboard of Fig.1 and then transfer it to the breadboard layout of Fig.6.

Next steps

Looking under the Examples in the file menu, you'll see some easy to follow examples. Because the ATtiny85

ATtiny85 pin functions

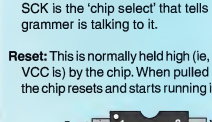
Digital: All of the I/O pins are capable of digital input and output. They can be set either high (VCC) or low (0V). They can also read a digital high or low as well.

Analog In: These pins are capable of reading a voltage of between 0 and your VCC voltage, providing a 10-bit number: 0V reads as "0" while VCC reads as "1023". If you need to measure higher voltages, you can use a voltage divider circuit to reduce the voltage going into this pin.

PWM: Pulse Width Modulation (PWM) output – these pins can simulate an analog voltage output by using PWM. Instead of adjusting the voltage, they can send shorter or longer pulses, thereby changing the average voltage. For applications like motors or lights this works well. You can set these pins to an 8-bit value (ie, 0 to 255). When set to a value of 0, the pin has a 0% duty cycle and is equivalent to 0V. At 255, it has 100% duty cycle and is equivalent to your VCC voltage.

ICSP Pins: Connect your ICSP to these pins to program your chip. MISO and MOSI stand for 'master in, slave out' and 'master out, slave in' respectively. SCK is the 'chip select' that tells the chip the programmer is talking to it.

Reset: This is normally held high (ie, at 5V or whatever VCC is) by the chip. When pulled briefly to ground, the chip resets and starts running its program again.



PINS ON THE ATTiny85

ICSP PINOUTS
(Top view, looking at a programmer)

You'll note the pin numbers in software don't correspond with the physical pin numbers of the chip. This diagram will help translate between the software world and the real world.

Parts you will need

First of all, you need the Freetronics ICSP Programmer for Arduino which you can buy on Freetronics' website (www.freetronics.com.au) for $22.00 plus shipping

See www.freetronics.com.au/blogs/news/8607215

It comes with a ribbon header cable (6-pin to 6-pin) and a short USB cable (type A to micro-B). And they'll throw in a mini protoboard for only $2.00 more – just what you need!

By the way, Freetronics also provide a PDF guide to using their Programmer, which readers may wish to use in conjunction with the description provided above.

Other main parts (Not a complete list... These components will allow you to build any one of the projects here but some components are common to all three).

1 Atmel ATtiny85 microcontroller (Altronics Z-5105)
1 DS18B20 digital thermometer chip (Altronics Z-7280)
1 IRF540N N-channel Mosfet (Altronics Z-1537; Jaycar ZT2466)
1 7805 5V regulator (Altronics Z-0505; Jaycar ZV1505)
1 red LED (Altronics Z-0700; Jaycar ZD0150)
1 Jansjo 2W LED lamp and 4.5V DC plugpack from Ikea
1 1μF 10V electrolytic capacitor
1 100nF polyester capacitor
1 470Ω resistor
1 1kΩ resistor
1 4.7kΩ resistor
1 10kΩ potentiometer
1 x 2 pin DIN plug (Jaycar PP0300)
1 x 2 pin DIN socket (Jaycar PS0340)
1 x 8 pin IC Socket (Jaycar PI6452)
6 300mm lengths single-core copper or tinned copper wire ("bell wire")
1 2x3-way DIL pin header (may to be cut down from larger – eg 2x10-way
(If not obtained above from Freetronics): 1 small breadboard (protoboard) (Altronics P-1020; Jaycar PB8817)

You can download the code (programs) required from www.siliconchip.com.au/

doesn't have many pins or built in peripherals (like SPI or I²C), some of those programs won't work but they can still give you many examples to copy to your code from.

Now is a good time to take a look at the Arduino community for other sources of inspiration and problem solving.

If you're having a problem with something, it's almost certain that you're not the first person to come across it and someone else will probably have solved it.

References:

www.atmel.com/images/doc0943.pdf
– shows how to use ICSP with other things connected to the pins. Embedded.fm episode 15
http://embedded.fm/www.instructables.com/id/Using-the-Arduino-Uno-to-program-ATTINY84-20PU/
– not the exact chip we're using here but gives a lot more information about programming the ATtiny series using Arduino. sc