Arduino as a Gateway Drug

Eric Ayars
ayars@mailaps.org

July 25, 2012

1 Using Your Arduino as a Programmer

1.1 Put this software on the Arduino:

File/Examples/ArduinoISP (within Arduino IDE, upload it to the Arduino as you would any other Arduino program.)

1.2 Make a Serial Peripheral Interface (SPI) connection:

- MOSI — 11
- MISO — 12
- SCK — 13
- Reset — 10
- Vcc — Usually 5V on the Arduino, could be 3.3V also.
- GND — Any of the ground pins on the Arduino
- (optional) "Heartbeat" LED on pin 9
- (optional) "Error" LED on pin 8
- (optional) "Communication" LED on pin 7

There are two approaches one can take for re-programming these chips: in-system (ISP) and external. For in-system programming, make a 6-pin ISP header on your circuit. For external, put the chip in a socket and move it to a programmer and back any time changes are needed.

1.3 "Cores" added to the Arduino IDE

Download the cores from Google Code: `http://code.google.com/p/arduino-tiny/` Install them in a folder called 'hardware' in your Arduino sketchbook folder. Start (or restart) Arduino IDE and a bunch of new 'boards' will be available under the 'tools' menu.

Another source of very helpful information on this modified Arduino IDE is located at the MIT High-Low Tech website: `http://hlt.media.mit.edu/?p=1695`

2 Programming the chip

2.1 Setting the fuses

- Select "Arduino as ISP".
- Set the target device. (Don't get this wrong!)
- Select "Burn Bootloader" in Arduino IDE. (This must be done at least once for each chip.) Note: this does *not* burn a bootloader on the chip; the chip must still be processed via an SPI interface rather than via a serial connection. What it does is use the Arduino IDE's "burn bootloader" process to set the fuses on the chip to the desired clock source and speed.

2.2 Programming

Once the fuses have been set correctly, the chip can be programmed just like an Arduino — just use your ArduinoISP-programmed Arduino as an adaptor between the USB port on your computer and the SPI interface on the chip.

2.3 Troubleshooting

Double-check the connections: it's common to switch MOSI and MISO, which will break communications. Also check the rest of the wiring while you're at it!

If you get an error when you try programming the second (non-Arduino) chip about "not in sync", then put a 10μF capacitor between the Arduino Reset pin and ground. This will prevent automatic reset of the Arduino, which fixes a timeout problem that causes the error; but it will also make it more difficult to program the Arduino. Make the capacitor removable

unless you want the Arduino to be a programmer permanently. *This additional capacitor fixes the issue we were having in sessions 2–6 of the ALPhA workshop on 7/25/12.*

3 Some chip options

- ATtiny45/85: A cheap 8-pin chip with relatively large memory, this is usually the first chip I grab for simple applications.
  - 4/8 kB program memory
  - 256/512 B SRAM
  - 256/512 B EEPROM
  - 6 I/O pins, including 2 PWM outputs
  - 4 10-bit ADCs
  - Internal 1MHz, internal 8MHz, or external 20MHz clock (in addition to other options)
- ATtiny44/84: 14-pin chip, more inputs/outputs than the ATtiny85.
  - Same memory limits and clock speeds as ATtiny45/85
  - 12 I/O lines, including 4 pwm outputs
  - On-chip temperature sensor[1]
  - 8 10-bit ADCs
- ATtiny2313: Lots of I/O pins, not much memory.
  - 2 kB of program memory
  - 128 B SRAM
  - 128 B EEPROM
  - 18 programmable I/O lines, including 4 10-bit ADCs

4 Sample Circuits

4.1 Code for the below examples

- Morse code trainer: `http://physics.csuchico.edu/~eayars/code/morse.pde.html`

[1] I've not gotten this to work with this programming method yet, though!

- Waterlevel alarm: `http://physics.csuchico.edu/~eayars/code/treesaver.pde.html`
- Servo driver: `http://physics.csuchico.edu/~eayars/code/remote_key.pde.html`
- Fancart: `http://physics.csuchico.edu/~eayars/code/fancart.ino.html`

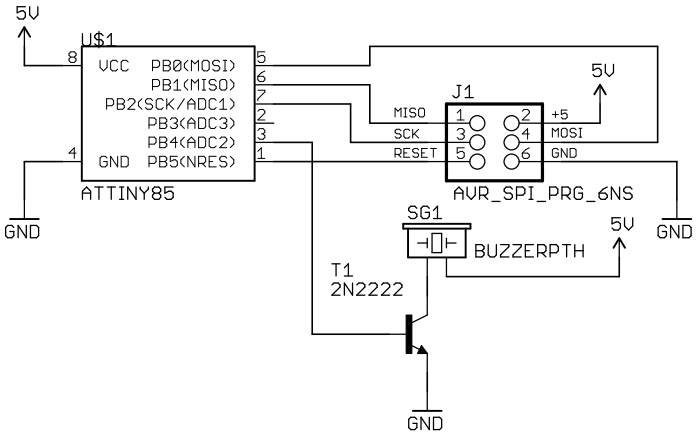

Figure 1: Simple Blinker / Pulse Generator / Morse Code Trainer using an ATtiny45/85

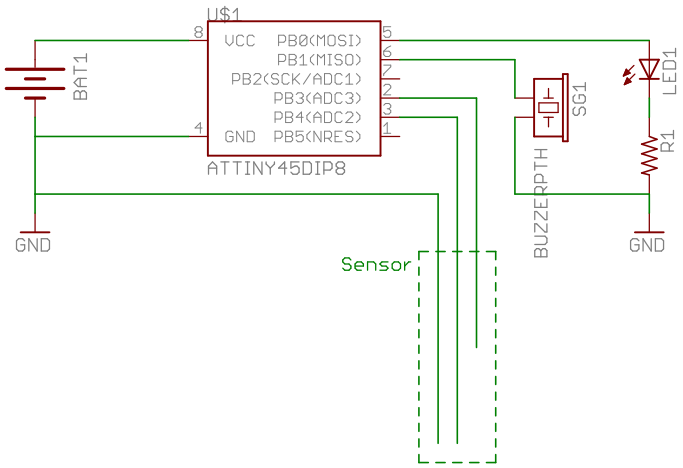

Figure 2: Waterlevel Alarm using an ATtiny45

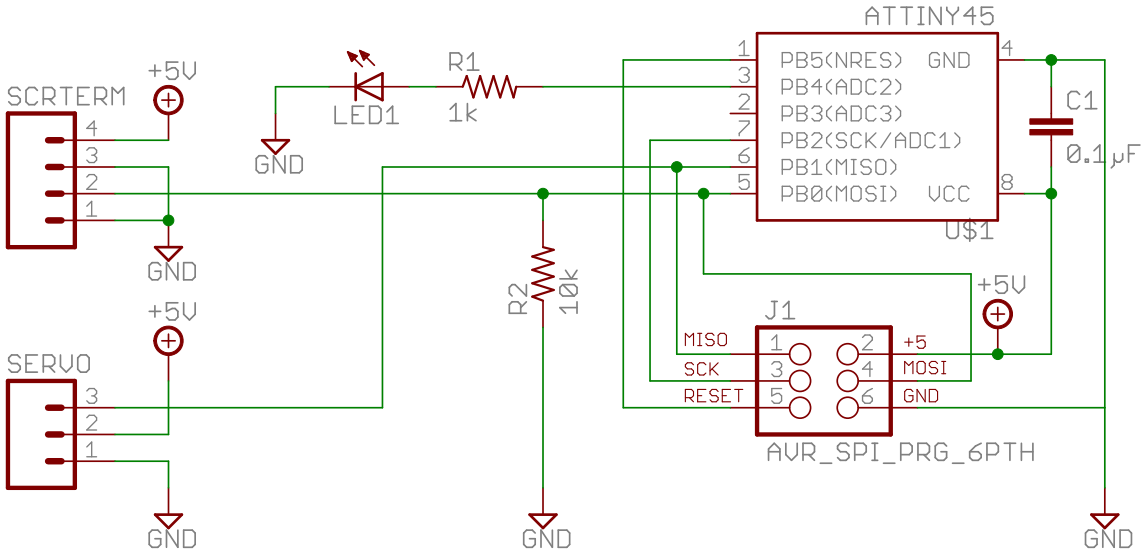

Figure 3: Two-position servo actuator schematic

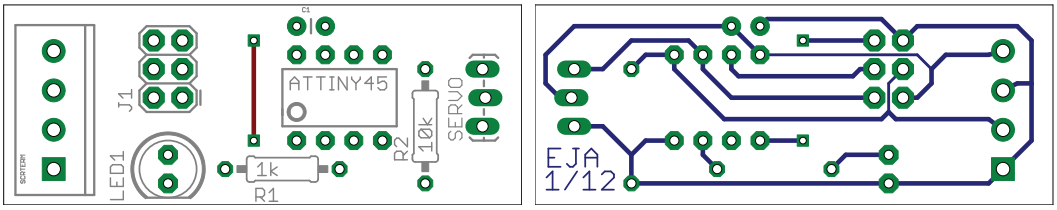

Figure 4: Two-position servo actuator board (Both sides shown)

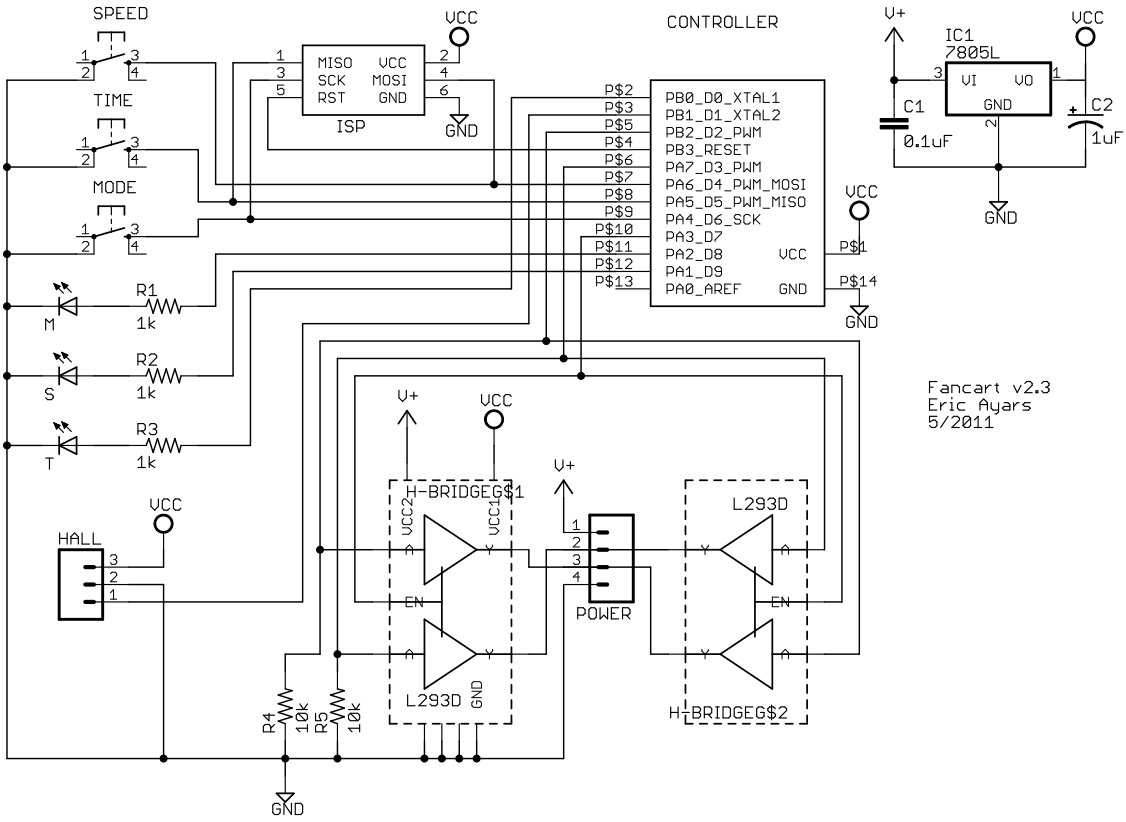

Figure 5: Programmable fan cart using ATtiny84

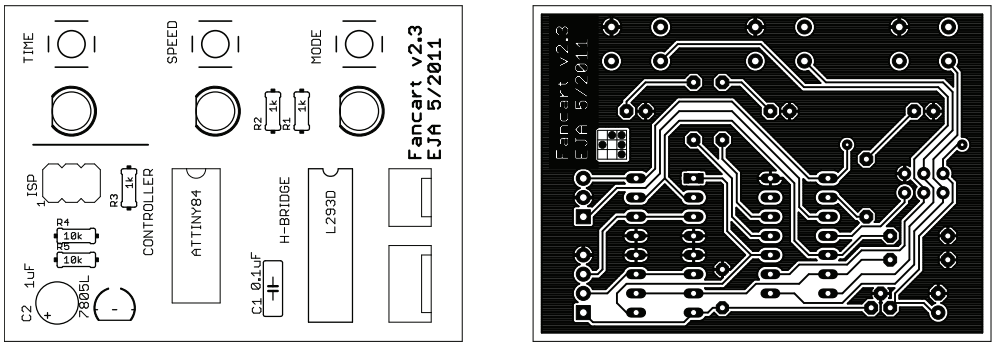

Figure 6: Programmable fan cart board (both sides shown)