Search

Type and hit enter

About Robotics
Australia How To
Linux Windows
Software Hardware
Tools WWW
Other RSS

Tag Cloud

how to problem domain sale website for sale pizza Linux pizza.com Windows solution investment fix plesk osCommerce pleo bluetooth Archos wireless installation error wrt54gl port rails Oracle Policy Automation OPA routerbot explorer robot fdisk wrt54g serial fake auction review certificate slow notes coupons SSL paypal configuration free store usps

Archives

February 2015
August 2014
July 2014
April 2014
March 2014
December 2013
November 2013
September 2013
August 2013
June 2013
May 2013
April 2013
March 2013
February 2013
January 2013
December 2012
June 2012
January 2012
December 2011
September 2011
August 2011
June 2011
February 2011
October 2010
July 2010
June 2010
May 2010
January 2010
September 2009
July 2009
June 2009
May 2009
February 2009
January 2009
November 2008
October 2008
September 2008
August 2008
June 2008
May 2008

Andrey Mikhalchuk’s Blog

Technoblog about life

Jun 20, 2011 Reading ATtiny85/45/25 Internal Temperature Sensor

← Juliana and Doyle Brothers Totally Rock! Controlling RGB LED using PICAXE-08: Basics of software PWM →

ATtiny85 has internal temperature sensor

I am working on one of my projects from RTFMs video blog (check out http://rtfms.com) that requires temperature sensing in a very small packaging. Naturally my choice is ATtiny85 – an awesome little chip from AVR that besides other goods (like 6ch PWM, serial interface etc) has internal temperature sensor. So I decided to use one. That wasn’t easy, but after a few hours of digging forums and datasheets I came up with a class that does the job with quite impressive reliability and precision.

First of all to run this code you need an Arduino-Tiny (http://code.google.com/p/arduino-tiny/) core and some kind of ISP. I’m using Arduino Duemilanove as ISP. See my RTFMs blog for more details about how to make it work this way, how to load correct core etc. AFAIK all popular cores will work fine with this code. I only recomment Arduino-Tiny because of its rich functionality that can be used to unleash more power of ATtiny85.
The solution consists of two files that you can copy into your project. Just copypaste them from this page:

InternalTemperatureSensor.h

```
#ifndef _INTERNAL_TEMPERATURE_SENSOR_H_
#define _INTERNAL_TEMPERATURE_SENSOR_H_

#define TEMPERATURE_SAMPLES 30
#define TEMPERATURE_ADJUSTMENT -13
#define EXTREMES_RATIO 5

#define MAXINT 32767
#define MININT -32767

/*
* Based on samples:
* – The internal temp sensor has horrible deviation. It’s output varies alot (+/-10 degrees/sec), so requires
* some “smoothing”. I’m removing extreme values and running rolling avg on the rest. This returns ok
* results on 20 samples and more. You can reduce number of samples to reduce response time, but this
* will diminish precision.
* – The linearity is ok and approximately 1.0 as the datasheet promised
* – The offset need to be calculated for each chip separately and provided in c’tor
*
* In order to calaculate the offset hook up terminal to pin 2 (Serial TX) 9600 8N1 and use
* temperature.print() method call. This will print out temperature readouts. Match it against your
* calibrated thermometer attached to the chip. Adjust Tos in c’tor if necessary.
* See chapter 17.12 of the ATtiny85 datasheet for a bit more details
*/

class InternalTemperatureSensor {
int offset;
float coefficient;
int readings[TEMPERATURE_SAMPLES];
int pos;

public:
InternalTemperatureSensor( float k, int o ) : offset( o ), coefficient( k ), pos( 0 ) {}
// Call it every time you need to prepare the chip to read sensor (i.e. in setup)
// If you’re using other ADCs besides temperature call it before each temperature reading
void init();
// Returns the current averaged temperature in LSB
```

April 2008
March 2008
February 2008
January 2008
September 2005

Links

CraftOut.com
Mikhalchuk's Site
MineStream
N2H Global Translator
Pizza.com
PublicQuotes.com

```
int in_lsb();
// Returns the current averaged temperature in degrees Celsius
int in_c();
// Returns the current averaged temperature in degrees Fahrenheit
int in_f();
// Returns the current averaged temperature in Kelvins
int in_k();
// Returns the current raw temperature reading from sensor
int raw();
// Prints the current temperature readings in various formats
void print();
};
#endif // _INTERNAL_TEMPERATURE_SENSOR_H_
```

InternalTemperatureSensor.pde

```
#include "InternalTemperatureSensor.h"

void InternalTemperatureSensor::init() {
//analogReference( INTERNAL1V1 );
// ATTiny85 datasheet p140 (17.13.2), p137 (17.12)
// Configure ADMUX
ADMUX = B1111; // Select temperature sensor
ADMUX &= ~_BV( ADLAR ); // Right-adjust result
ADMUX |= _BV( REFS1 ); // Set Ref voltage
ADMUX &= ~( _BV( REFS0 ) | _BV( REFS2 ) ); // to 1.1V
// Configure ADCSRA
ADCSRA &= ~( _BV( ADATE ) |_BV( ADIE ) ); // Disable autotrigger, Disable Interrupt
ADCSRA |= _BV(ADEN); // Enable ADC
ADCSRA |= _BV(ADSC); // Start first conversion
// Seed samples
int raw_temp;
while( ( ( raw_temp = raw() ) < 0 ) );
for( int i = 0; i < TEMPERATURE_SAMPLES; i++ ) {
readings[i] = raw_temp;
}
}

int InternalTemperatureSensor::in_lsb() {
int readings_dup[TEMPERATURE_SAMPLES];
int raw_temp;
// remember the sample
if( ( raw_temp = raw() ) > 0 ) {
readings[pos] = raw_temp;
pos++;
pos %= TEMPERATURE_SAMPLES;
}
// copy the samples
for( int i = 0; i < TEMPERATURE_SAMPLES; i++ ) {
readings_dup[i] = readings[i];
}
// bubble extremes to the ends of the array
int extremes_count = TEMPERATURE_SAMPLES / EXTREMES_RATIO;
int swap;
for( int i = 0; i < extremes_count; ++i ) { // percent of iterations of bubble sort on small N works faster than Q-sort
for( int j = 0; j < TEMPERATURE_SAMPLES – 1; j++ ) {
if( readings_dup[i] > readings_dup[i+1] ) { // could be done with 3 XORs and no swap if you like fancy
swap = readings_dup[i];
readings_dup[i] = readings_dup[i+1];
readings_dup[i+1] = swap;
}
}
}
// average the middle of the array
int sum_temp = 0;
for( int i = extremes_count; i < TEMPERATURE_SAMPLES - extremes_count; i++ ) {
sum_temp += readings_dup[i];
}
return sum_temp / ( TEMPERATURE_SAMPLES - extremes_count * 2 );
}

int InternalTemperatureSensor::in_c() {
return in_k() - 273;
}

int InternalTemperatureSensor::in_f() {
return in_c() * 9 / 5 + 32;
}

int InternalTemperatureSensor::in_k() {
return in_lsb() + offset; // for simplicty I'm using k=1, use the next line if you want K!=1.0
//return (int)( in_lsb() * coefficient ) + offset;
}

int InternalTemperatureSensor::raw() {
if( ADCSRA & _BV( ADSC ) ) {
return -1;
} else {
int ret = ADCL | ( ADCH << 8 ); // Get the previous conversion result
ADCSRA |= _BV(ADSC); // Start new conversion
return ret;
}
}

void InternalTemperatureSensor::print() {
Serial.print( "> R:" );
Serial.print( raw(), DEC );
Serial.print( " L:" );
Serial.print( in_lsb(), DEC );
Serial.print( " K:" );
Serial.print( in_k(), DEC );
Serial.print( " C:" );
Serial.print( in_c(), DEC );
Serial.print( " F:" );
Serial.print( in_f(), DEC );
Serial.println( " # " );
}
```

Usage

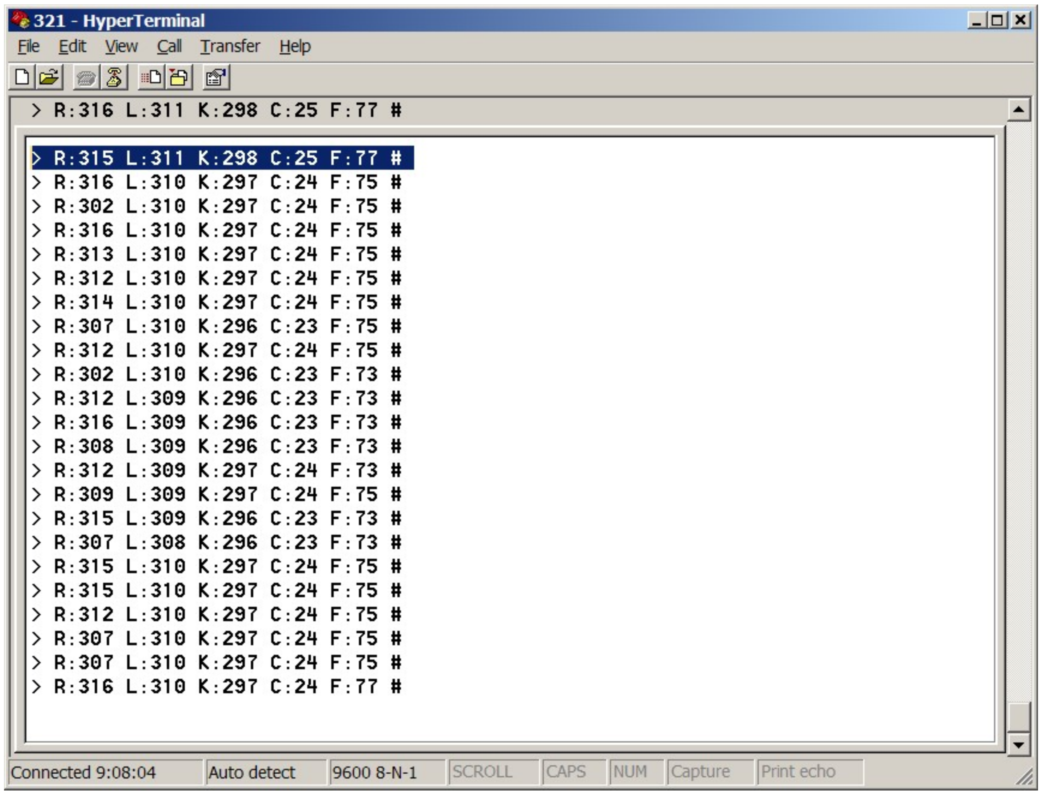
```
InternalTemperatureSensor temperature( 1.0, TEMPERATURE_ADJUSTMENT ); // The 1.0 argument is ignored by default, see the c’tor comments
void setup() {
temperature.init(); // Call init() in setup or every time after you modify ADCSRA or ADMUX
}

void loop() {
// temperature.print(); // uncomment this to debug sensor output via serial connection
int war_sensor_data = temperature.raw(); // This returns barely usable sensor output
int temperature_in_celsius = temperature.in_c(); // This returns more usable temperature in degrees Celsius
}
```

Explanation

ATtinyX5 uses ADC4 for reading data from the internal sensor. In init() method all necessary magic with the registers is done to prepare the system for reading this data. You need to call this method every time you want to configure ADMUX and ADCSRA for retrieving the temperature sensor data. If the only ADC channel you use in your program is the temperature sensor’s ADC4 then you should only call it in setup(), i.e. once. This reduces the temperature reading procedure time at least by factor two.

Method InternalTemperatureSensor::raw() returns the raw data from the sensor. Unfortunately this data is barely usable. To illustrate the problem take a look at the output from InternalTemperatureSensor::print() debug method. The first column is what the sensor returns. The output rate is approximately 5 lines per second, so within 2 seconds the sensor returned values with 18 degrees difference.



```
> R:316 L:311 K:298 C:25 F:77 #

> R:315 L:311 K:298 C:25 F:77 #
> R:316 L:310 K:297 C:24 F:75 #
> R:302 L:310 K:297 C:24 F:75 #
> R:316 L:310 K:297 C:24 F:75 #
> R:313 L:310 K:297 C:24 F:75 #
> R:312 L:310 K:297 C:24 F:75 #
> R:314 L:310 K:297 C:24 F:75 #
> R:307 L:310 K:296 C:23 F:75 #
> R:312 L:310 K:297 C:24 F:75 #
> R:302 L:310 K:296 C:23 F:73 #
> R:312 L:309 K:296 C:23 F:73 #
> R:316 L:309 K:296 C:23 F:73 #
> R:308 L:309 K:296 C:23 F:73 #
> R:312 L:309 K:297 C:24 F:73 #
> R:309 L:309 K:297 C:24 F:75 #
> R:315 L:309 K:296 C:23 F:73 #
> R:307 L:308 K:296 C:23 F:73 #
> R:315 L:310 K:297 C:24 F:75 #
> R:315 L:310 K:297 C:24 F:75 #
> R:312 L:310 K:297 C:24 F:75 #
> R:307 L:310 K:297 C:24 F:75 #
> R:307 L:310 K:297 C:24 F:75 #
> R:316 L:310 K:297 C:24 F:77 #
```

ATtiny85 Internal Temperature Sensor Readouts

In order to make this data more usable method InternalTemperatureSensor::in_lsb does rolling average and removes the extremes from the input. The result you can see on the picture above: the in_lsb output marked as L: is way more usable. It's more accurate too. You can adjust the rolling average parameters, but overall I do not recommend making TEMPERATURE_SAMPLES less than 20 and greater than 50. If it's less than 20 then deviation grows significantly. Making TEMPERATURE_SAMPLES greater than 50 doesn't improve precision but wastes memory and slows down calculations. Also EXTREMES_RATIO defines how much of extreme values will be chopped, do not make it less than 2 (that will eliminate all values). Making it more than TEMPERATURE_SAMPLES will effective prevent removing extremes from the samples.
In addition to these 3 core methods a few utility methods in_c(), in_k(), in_f() are provided. They return temperature in degrees Celsius, Fahrenheit and Kelvins.

One more problem with the internal sensor is that it's not calibrated, so for each chip you need to use print() method to output the reading and adjust the second parameter of the constructor to match those values with your calibrated thermometer. The value -13 in the source code works only for one of my ATtinies, there is no guarantee it will work for others.
Overall the internal sensor is usable, but don't expect much from it. External sensors are still much better.

← Juliana and Doyle Brothers Totally Rock! Controlling RGB LED using PICAXE-08: Basics of software PWM →

This post is published in Hardware, Robotics.

No comments

No comments so far

Leave a Reply

You must be logged in to post a comment.

Site Map