**Arduino Code Development Notes**

**Dr. D. S. Stutts**

**2013-2-24**

Figured out how to install Arduino libraries following these instructions: <http://arduino.cc/en/Hacking/LibraryTutorial>

The search function seems to work, but keeps looping add-infinitum. The DS18B20 (which the library code correctly identified) with the Ni-Cr wire has address: 28 9A 96 C9 2 0 0 AE.

The example code provided by the developers follows:

#include <OneWire.h>

// OneWire DS18S20, DS18B20, DS1822 Temperature Example

//

// http://www.pjrc.com/teensy/td\_libs\_OneWire.html

//

// The DallasTemperature library can do all this work for you!

// http://milesburton.com/Dallas\_Temperature\_Control\_Library

OneWire ds(10); // on pin 10

void setup(void) {

Serial.begin(9600);

}

void loop(void) {

byte i;

byte present = 0;

byte type\_s;

byte data[12];

byte addr[8];

float celsius, fahrenheit;

if ( !ds.search(addr)) {

Serial.println("No more addresses.");

Serial.println();

ds.reset\_search();

delay(250);

return;

}

Serial.print("ROM =");

for( i = 0; i < 8; i++) {

Serial.write(' ');

Serial.print(addr[i], HEX);

}

if (OneWire::crc8(addr, 7) != addr[7]) {

Serial.println("CRC is not valid!");

return;

}

Serial.println();

// the first ROM byte indicates which chip

switch (addr[0]) {

case 0x10:

Serial.println(" Chip = DS18S20"); // or old DS1820

type\_s = 1;

break;

case 0x28:

Serial.println(" Chip = DS18B20");

type\_s = 0;

break;

case 0x22:

Serial.println(" Chip = DS1822");

type\_s = 0;

break;

default:

Serial.println("Device is not a DS18x20 family device.");

return;

}

ds.reset();

ds.select(addr);

ds.write(0x44,1); // start conversion, with parasite power on at the end

delay(1000); // maybe 750ms is enough, maybe not

// we might do a ds.depower() here, but the reset will take care of it.

present = ds.reset();

ds.select(addr);

ds.write(0xBE); // Read Scratchpad

Serial.print(" Data = ");

Serial.print(present,HEX);

Serial.print(" ");

for ( i = 0; i < 9; i++) { // we need 9 bytes

data[i] = ds.read();

Serial.print(data[i], HEX);

Serial.print(" ");

}

Serial.print(" CRC=");

Serial.print(OneWire::crc8(data, 8), HEX);

Serial.println();

// convert the data to actual temperature

unsigned int raw = (data[1] << 8) | data[0];

if (type\_s) {

raw = raw << 3; // 9 bit resolution default

if (data[7] == 0x10) {

// count remain gives full 12 bit resolution

raw = (raw & 0xFFF0) + 12 - data[6];

}

} else {

byte cfg = (data[4] & 0x60);

if (cfg == 0x00) raw = raw << 3; // 9 bit resolution, 93.75 ms

else if (cfg == 0x20) raw = raw << 2; // 10 bit res, 187.5 ms

else if (cfg == 0x40) raw = raw << 1; // 11 bit res, 375 ms

// default is 12 bit resolution, 750 ms conversion time

}

celsius = (float)raw / 16.0;

fahrenheit = celsius \* 1.8 + 32.0;

Serial.print(" Temperature = ");

Serial.print(celsius);

Serial.print(" Celsius, ");

Serial.print(fahrenheit);

Serial.println(" Fahrenheit");

}

I tested the search function successfully with five DS18B20s. The enumeration process goes like the following:

ROM = 28 70 5A 3A 3 0 0 19

Chip = DS18B20

Data = 1 6F 1 4B 46 7F FF 1 10 67 CRC=67

Temperature = 22.94 Celsius, 73.29 Fahrenheit

ROM = 28 52 CF 39 3 0 0 2C

Chip = DS18B20

Data = 1 6F 1 4B 46 7F FF 1 10 67 CRC=67

Temperature = 22.94 Celsius, 73.29 Fahrenheit

ROM = 28 9A 96 C9 2 0 0 AE

Chip = DS18B20

Data = 1 67 1 4B 46 7F FF 9 10 3B CRC=3B

Temperature = 22.44 Celsius, 72.39 Fahrenheit

ROM = 28 26 91 3A 3 0 0 D7

Chip = DS18B20

Data = 1 6B 1 4B 46 7F FF 5 10 49 CRC=49

Temperature = 22.69 Celsius, 72.84 Fahrenheit

ROM = 28 25 2F 3A 3 0 0 63

Chip = DS18B20

Data = 1 6D 1 4B 46 7F FF 3 10 70 CRC=70

Temperature = 22.81 Celsius, 73.06 Fahrenheit

No more addresses.

As can be seen, the search code also calculates the CRC value for each temperature data packet sent. Pretty sweet!

The second library I found, DallasTemperature, failed to find any DS18B20s when running the “Mulitple” example. It simply returned 0 degrees at a rapid rate – probably at 10 Hz.

2/26/13 11:19:20 AM

Met with the CS team consisting of Nathan, Katy, and Maddie this morning, and gave them a tour of our labs and an overview of our goals. We decided on the Processing/Arduino system as our development platform due to its ease of use, and OS-agnostic nature. The CS team will develop libraries and classes in the Processing language targeting both the MCU-based Arduino and the host-based Processing platforms to facilitate data acquisition and control applications for our mechanical engineering students. The first application will be a simple temperature control experiments consisting of a single DS18B20 temperature sensor (for which functional Arduino libraries already exist) and a low-power resistance heater. The heater consists of a short length of electrically insulated Ni-Cr wire, which will be powered through a low-side MOSFET switch controlled by a PWM output Arduino pin. My (DSS) next task is to finalize the heater drive and control hardware and firmware. This will consist, primarily, of porting existing C-code to the Processing-based Arduino, and also finalizing the power circuit design.