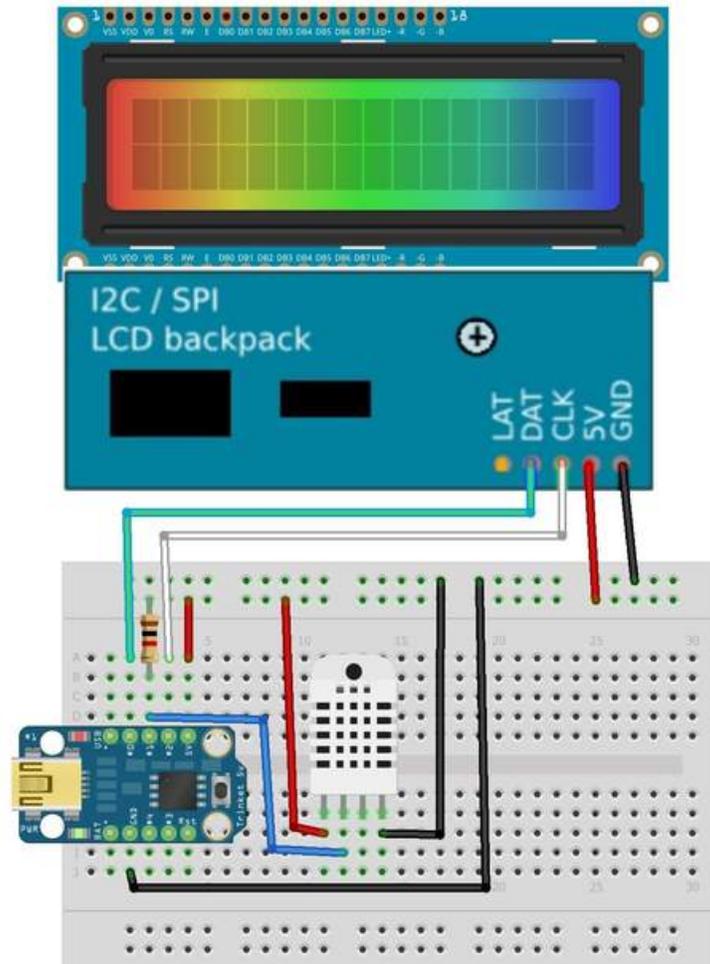


Trinket Temperature & Humidity LCD Display

Created by Mike Barela



Last updated on 2018-08-22 03:37:40 PM UTC

Guide Contents

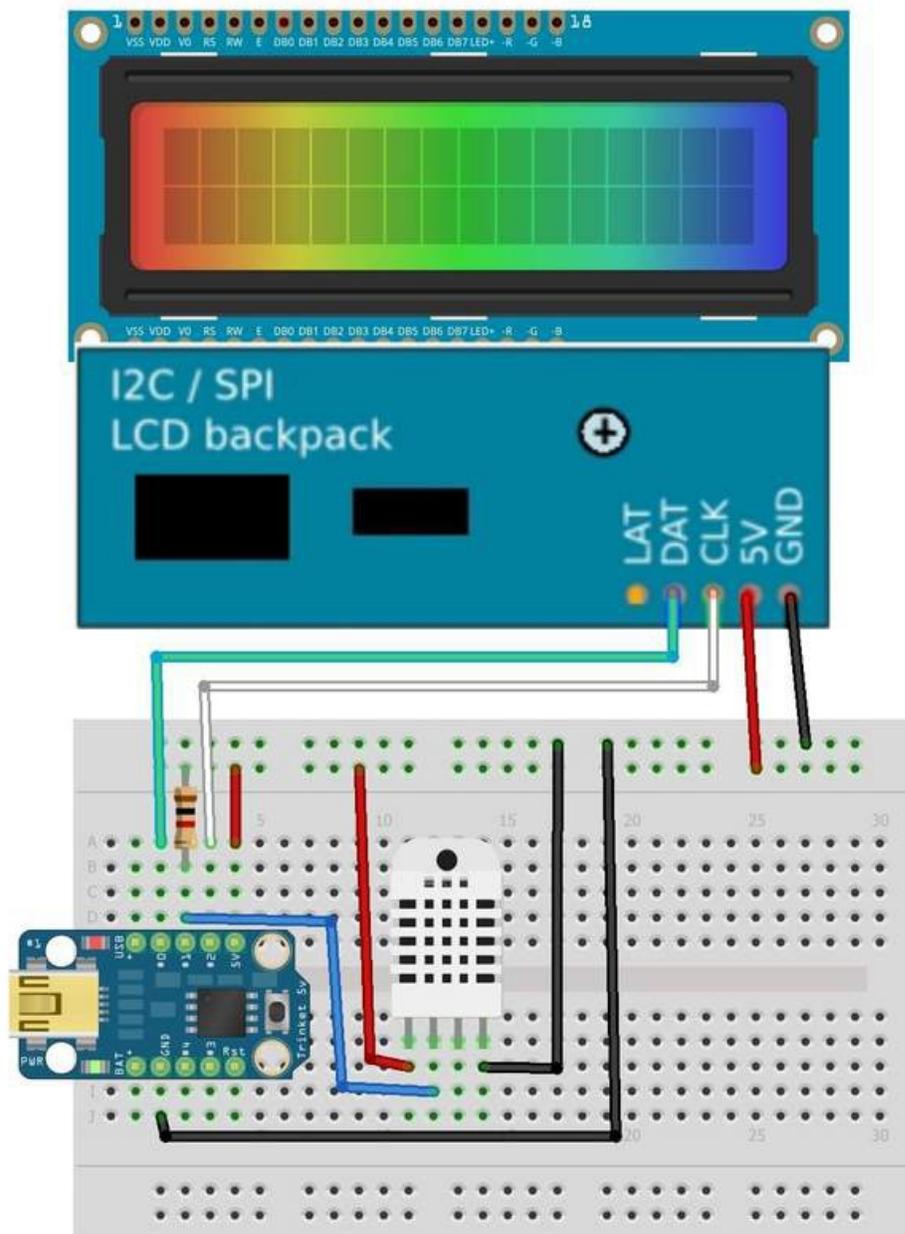
Guide Contents	2
Overview	3
Software Libraries Used	4
The LCD Display	5
What if I have no display?	7
Adding the Sensor	8
Code	10
Debugging and Going Further	12
Going Further	12

Overview

The first thing a microcontroller project must do is communicate, often with us humans. While the Trinket mini microcontroller does not have a serial monitor built in (like the Arduino Uno), it can talk over various protocols including software serial, I2C (two wire), and SPI. Adafruit sells a wide array of I2C devices including a backpack to interface with a number of nice liquid crystal (LCD) displays - perfect as it only requires two of the five Trinket pins.

Monitoring sensors is very common for Internet of Things (IoT) projects. Here we'll select the popular DHT series of temperature and humidity sensors.

This project can be placed in a very small enclosure and used anywhere environmental monitoring is needed. The code and concepts may be used in a number of your own projects.



Software Libraries Used

To maximize the functionality of software on Trinket, two Arduino software libraries need to be installed. See the [All About Arduino Libraries \(https://adafru.it/aYM\)](https://adafru.it/aYM) tutorial for details on how to download and install these libraries.

- [Adafruit_LiquidCrystal \(https://adafru.it/leO\)](https://adafru.it/leO) library - this provides support for I2C displays that Adafruit sells
- [TinyDHT \(https://adafru.it/cKe\)](https://adafru.it/cKe) library - this library is optimized for the Atmel ATtiny processors like Adafruit Trinket and Arduino Gemma. It uses integer math to avoid pulling in the code-heavy floating point support libraries.

The LCD Display

Adafruit carries many character LCD display varieties with multiple sizes and backlight colors.

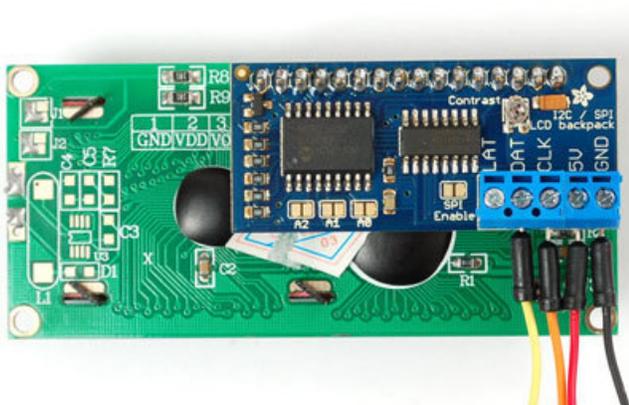
The [Adafruit I2C / SPI character LCD backpack \(http://adafru.it/292\)](http://adafru.it/292) allows you to control these displays by sending data over the two wire I2C interface. Standard LCDs require a large number of digital pins, straining the capability of even an Arduino Uno. Use of the I2C backpack reduces the pins needed considerably.

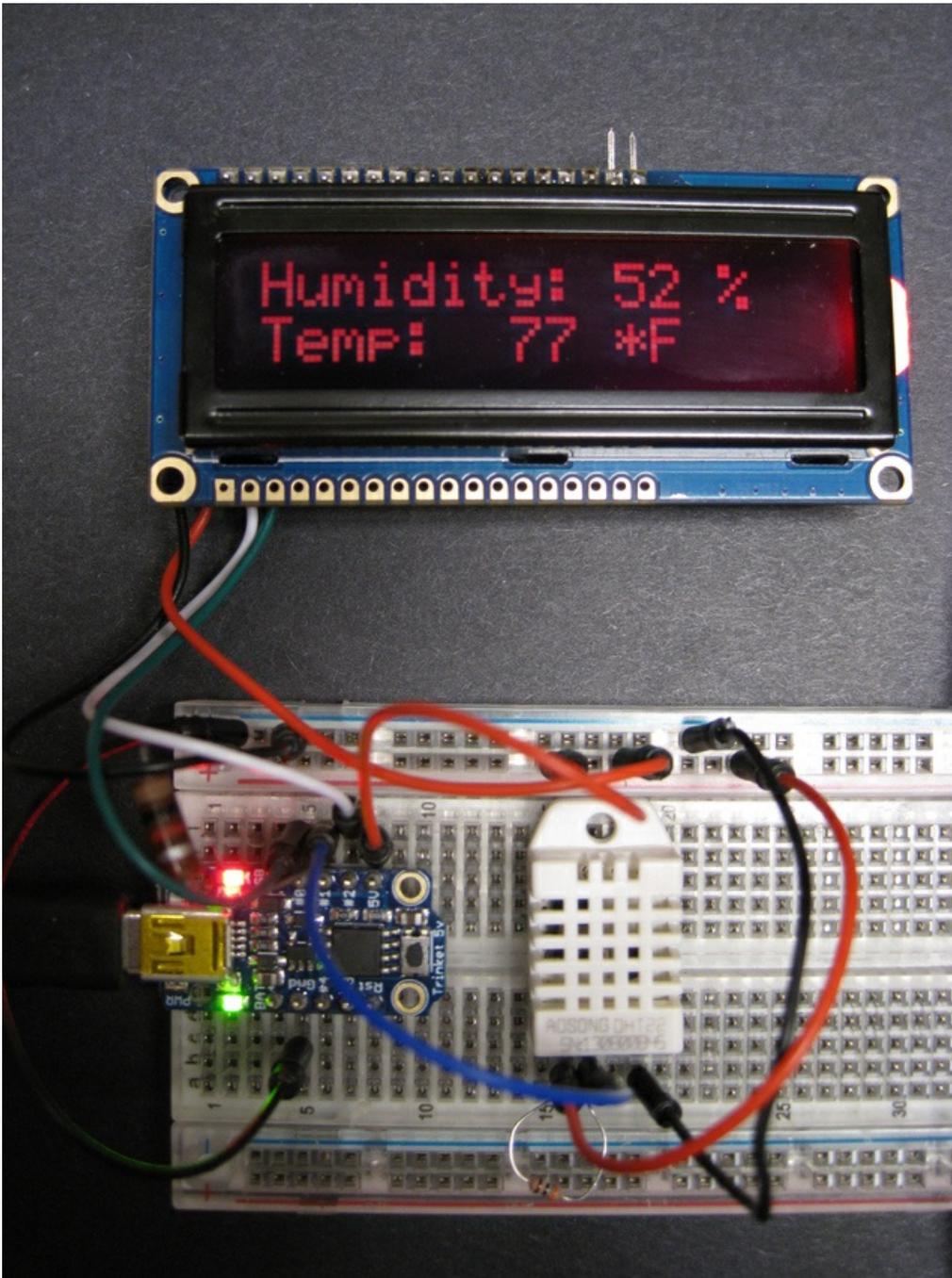
This project features a 16x2 display, displaying temperature and humidity without using a great deal of memory (important on a small microcontroller like Trinket).

The I2C backpack may be assembled and placed on the back of the display. See [the guide to backpack assembly \(https://adafru.it/ckf\)](https://adafru.it/ckf) to prepare your display and the backpack.

The color displays have a couple of extra connectors - pins 16, 17, and 18 control the three color backlights. If you connect pin 16, the I2C will control the red light. You can choose to put a jumper from one of the backlight pins to backpack pin 16 to choose a different color or connect the pins high to keep them on all the time. Making the pin choice before soldering on the backpack allows you the most flexibility in choosing your backlight color.

Or you can just go with a ['classic' blue & white 16x2 LCD \(https://adafru.it/cKw\)](https://adafru.it/cKw)





To test the display, wire the DAT pin to Trinket GPIO #0, the CLK pin to Trinket GPIO #2, 5V to the Trinket 5V line and GND to GND. Use the Fritzing wiring diagram on the previous page to check your wiring.

Ensure your Arduino IDE has been configured to program the Adafruit Trinket 5V 8 MHz for the Hello World test program (note for later: we will be changing this to Trinket 5V 16 MHz for the sensor program).

The display test program is a variation of the Hello World program. You need to install the Adafruit_LiquidCrystal library mentioned on the previous page.

Ensure you configure the Arduino IDE software for Trinket as noted in the tutorial [Introducing Trinket](#). Also select Trinket 5V 8 MHz in the Arduino Tools menu.

```

/*
Demonstration sketch for Adafruit i2c/SPI LCD backpack
using MCP23008 I2C expander and the Trinket mini microcontroller

This sketch prints "Hello World!" to the LCD
and shows a count in seconds. The circuit:

* 5V to Trinket 5V pin
* GND to Trinket GND pin
* CLK to Trinket pin GPIO #2
* DAT to Trinket pin GPIO #0
*/

// include the library code:
#include <Adafruit_LiquidCrystal.h>

// Connect via i2c, default address #0 (A0-A2 not jumpered)
Adafruit_LiquidCrystal lcd(0);

void setup() {
  // set up the LCD
  lcd.begin(16, 2);           // our display has 16 cols, 2 rows
  lcd.setBacklight(HIGH);    // Turn on the backlight
  lcd.print("hello, world!"); // Print a message to the LCD.
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis()/1000);
}

```

If you plan on having more than one LCD or I2C device, see [Changing the I2C Address \(https://adafru.it/cKg\)](https://adafru.it/cKg) for details.

What if I have no display?

Using the contrast potentiometer on the backpack (a small silver bump), turn the dial with a small screwdriver. Change the contrast until you can read the text.

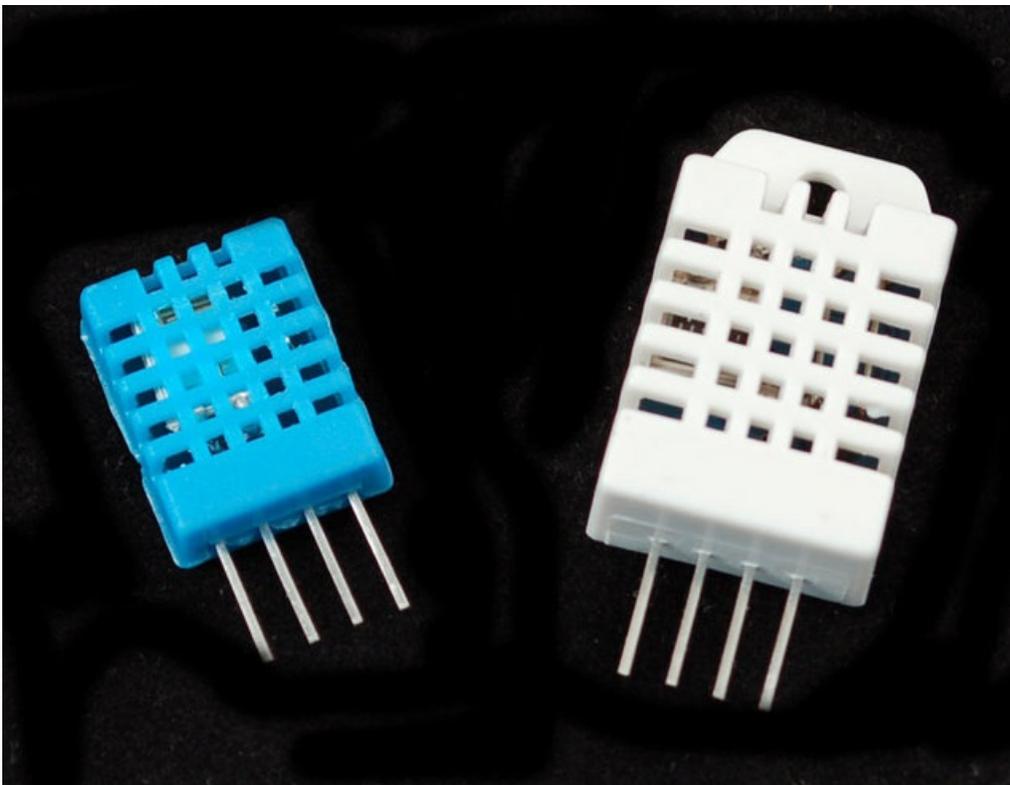
Adding the Sensor

Once you have the display connected to the Trinket, you can expand the project in many ways. One of the most popular is to add sensor(s) of some sort. [The Adafruit store has a wonderful selection to chose from \(https://adafru.it/cKh\)](https://adafru.it/cKh).

The easiest place to add a sensor is to Trinket GPIO #1. GPIO pins #0 and #2 are used for the display and #3 and #4 are shared with the USB port. Using #3 and #4 is perfectly fine, but you may have to disconnect the connections on those pins when uploading software. Also, limiting a project to pins #0 to #2 gives you a similar pinout to the Adafruit Gemma wearable controller.

This project demonstrates using the DHT22 temperature and humidity sensor. It provides a 0-100% humidity reading with 2-5% accuracy and -40 to 80°C temperature readings with $\pm 0.5^{\circ}\text{C}$ accuracy. The sensor only requires one digital pin for readings, perfect.

You may want to review the [Adafruit tutorial on DHT \(https://adafru.it/cKi\)](https://adafru.it/cKi) sensors for more background.



See the wiring on the Fritzing diagram (first page) and the picture (second page). It is fairly easy to connect to the DHT sensors. They have four pins - from left to right with the grilled opening facing you:

- VCC (3 to 5V power)
- Data out
- Not connected
- Ground

You may ignore pin 3, its not used. Place a 1K ohm resistor between VCC (5V) and the data pin to act as a pull up on the data line. Do not use the 10K resistor normally used from 5v to the data pin.

Trinket GPIO Pin #1 is shared with the onboard red LED. Connecting the DHT, the sensor requires a 1,000 ohm pullup resistor (not provided) between Trinket GPIO #1 and 5V to pull up the voltage to be read by the Trinket.

Code

Be sure to set your Board type as Adafruit Trinket 5V 16 MHz, and Programmer as USBtinyISP in the Arduino IDE under the Tools menu. This is a change from many Trinket programs which were run on 8 MHz, take note.

To get the timing correct on the DHT sensor, the Trinket is clocked to 16 MHz (same as the Arduino Uno) in the first line in the setup() routine. This requires the 5V Trinket as the 3 volt Trinket is not guaranteed to function at 16 MHz. This also limits use to Trinket and not Gemma.

The code:

```
/*
Demonstration sketch for Adafruit LCD backpack
using MCP23008 I2C expander and DHT Temperature/Humidity Sensor
Uses the 5 volt Trinket mini microcontroller with the
Trinket set at 16 MHz due to timing reading the DHT sensor

This sketch prints the temperature and humidity to the LCD

The circuit:
* 5V to Arduino 5V pin
* GND to Arduino GND pin
* Display i2c backpack CLK to Trinket GPIO #2
* Display i2c backpack DAT to Trinket GPIO #0
* DHT Temperature Sensor to Trinket GPIO #1 (with 1K ohm pullup to 5V)

// Connect DHT pin 1 (on the left) of the sensor to +5V
// Connect DHT pin 2 of the sensor to whatever your DHTPIN is
// Connect DHT pin 4 (on the right) of the sensor to GROUND
// Connect a 1K resistor from pin 2 (data) to pin 1 (power) of the sensor
*/

// include the library code
#include <Adafruit_LiquidCrystal.h> // LiquidCrystal using the Wire library
#include <TinyDHT.h> // lightweight DHT sensor library
#include <avr/power.h> // needed to up clock to 16 MHz on 5v Trinket

// Uncomment whatever type sensor you are using!
// #define DHTTYPE DHT11 // DHT 11
#define DHTTYPE DHT22 // DHT 22 (AM2302)
// #define DHTTYPE DHT21 // DHT 21 (AM2301)
#define TEMPTYPE 1 // Use Fahrenheit (0 for celsius)

// Trinket GPIO #1 would be better but LED prevents digital talk with DHT sensor
#define DHTPIN 1 // Sensor connected to GPIO #1 (use a
// 1K ohm pullup to 5V to make it work!)
DHT dht(DHTPIN, DHTTYPE); // Define Temp Sensor

// Connect display via i2c, default address #0 (A0-A2 not jumpered)
Adafruit_LiquidCrystal lcd(0);

void setup() { // first line sets a 5V Trinket to 16 MHz operation
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1); // 5V Trinket: run at 16 MHz
  dht.begin(); // Initialize DHT Temperature Sensor

  lcd.begin(16, 2); // set up the LCD's number of rows and columns:
```

```

    lcd.setBacklight(HIGH); // Set backlight (HIGH - on)
}

void loop() {
    int8_t h = dht.readHumidity();           // Read humidity
    int16_t t = dht.readTemperature(TEMPTYPE); // read temperature

    lcd.setCursor(0, 0);
    if ( t == BAD_TEMP || h == BAD_HUM ) { // if error conditions (see TinyDHT.h)
        lcd.print("Bad read on DHT");      // print error message
    } else {
        lcd.print("Humidity: ");           // write to LCD
        lcd.setCursor(10,0);
        lcd.print(h);
        lcd.setCursor(12, 0);
        lcd.print(" % ");
        lcd.setCursor(0, 1);
        lcd.print("Temp:");
        lcd.setCursor(7, 1);
        lcd.print(t);
        lcd.setCursor(10, 1);
        lcd.print("°F");
    }
    delay(2000); // Read temp every second (2000 ms) (DHT sensor max rate)
}

```

The above code compiles to 4,760 bytes of 5,310 available. This leaves 550 bytes of code if you wish to add additional functionality. Note that adding display text will use the available space quickly. Also, if you use decimal (floating point) numbers, you will most likely exceed the space available. The Arduino IDE built-in functions to do floating point math are somewhat large (this is why the DHT library was edited to create the TinyDHT library: to use integer math, which limits precision to one degree and one percent but saves space).

If the display is not showing values you expect, ensure you compiled it with the board being the Adafruit Trinket 5V 16 MHz (not 8 MHz). Also check the sensor wiring.

Debugging and Going Further

If you are having software problems:

- Ensure you have installed Trinket support in your Arduino IDE as listed in the [Introducing Trinket \(https://adafru.it/cEu\)](https://adafru.it/cEu) tutorial.
- Ensure you have installed the [TinyDHT \(https://adafru.it/cKe\)](https://adafru.it/cKe), and [Adafruit_LiquidCrystal \(https://adafru.it/leO\)](https://adafru.it/leO) libraries listed on the first page of this tutorial. The Adafruit standard DHT and the Arduino LiquidCrystal libraries will not work.
- Ensure you push the Trinket on board reset button before uploading your sketch, the red LED will blink when ready for upload, there is a 10 second window to do this.
- If you place a large amount of code or other libraries in a sketch, it is very easy to exceed the available code space on the Trinket. If your program absolutely will not fit, consider switching to an Arduino Uno, [Adafruit Boarduino \(https://adafru.it/cKj\)](https://adafru.it/cKj), or [Adafruit Flora \(https://adafru.it/aSZ\)](https://adafru.it/aSZ) with standard libraries.
- If you get errors similar to the one below, you may have included decimal numbers and the floating point library was added by the Arduino IDE, exceeding the amount of program space available.

```
arduino-1.0.1/hardware/tools/avr/bin/./lib/gcc/avr/4.3.2/./../avr/lib/avr25/crttn85.o:(.init9+0x2): relocation truncated to fit: R_AVR_13_PCREL against symbol `exit' defined in .fini9 section in /arduino-1.0.1/hardware/tools/avr/bin/./lib/gcc/avr/4.3.2/avr25/libgcc.a(_exit.o)
```

For project issues:

- If you get no display, go to the Hello World I2C sketch and ensure that works.
- If you have no display on Hello World, on the I2C backpack, use the contrast knob to change the LCD display contrast to a readable level. If you decided on an external potentiometer to change contrast and not pin 16 on the backpack, use that.
- If you get "Bad Read on DHT", the sensor is not talking to the Trinket correctly. Check your wiring and ensure you have a 1,000 ohm resistor from Trinket GPIO #1 to 5V. Resistors are commonly available at electronic stores on line and at retail locations like RadioShack.
- If you get 0% Humidity and 32 degrees F/ zero C, ensure Trinket 5V 16 MHz is selected as the Board type in the Arduino IDE Tools menu. The sensor code will not give correct readings on a Trinket 3V or at 8 MHz.

Going Further

You can change the temperature sensor to a [TMP36 \(http://adafru.it/165\)](http://adafru.it/165) or other sensor with some changes. GPIO #3 is analog pin A3 and GPIO #4 is analog pin A2 for reading analog sensors. But be sure you do the conversion from the analog level to temperature using integer math (best to use an int16_t integer size) to avoid linking in the big floating point library for only a few calculations.

A light sensor such as a [cadmium sulfide photocell \(http://adafru.it/161\)](http://adafru.it/161) could be read, as in the [Trinket / Gemma Mini Theramin \(https://adafru.it/cKk\)](https://adafru.it/cKk), on GPIO #3 or #4, for a simple light meter displayed on the LCD display.

Once you have the display basics, you can use these technique as the human interface to a wide array of projects.