Storage humidity and temperature monitor

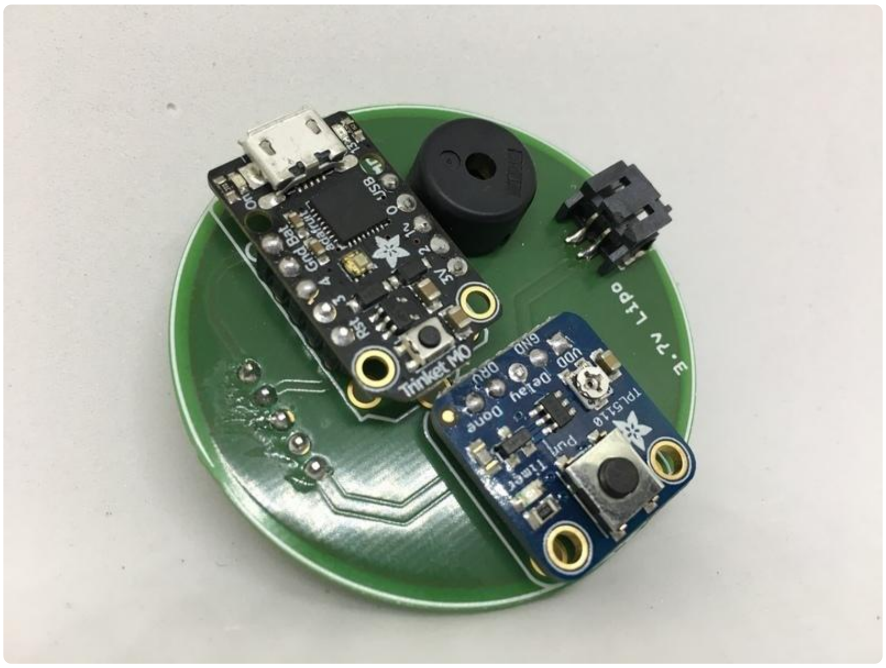
Created by Dave Astels

https://learn.adafruit.com/storage-humidity-and-temperature-monitor

Last updated on 2021-11-15 07:05:40 PM EST

Table of Contents

Overview



There are many things that should be stored at a more or less specific relative humidity. If not, they will dry out or get moldy. There are solutions for this, such as Bodeva (https://adafru.it/Bg6). These packets absorb or release moisture to maintain equilibrium at a specific relative humidity. The problem is that these things wear out over time and stop working. But how do you know? You can put indicator paper in the container, but unless it's clear you have to open it to check, and that messes up the equilibrium.

The answer? Technology, naturally. Goals for this project:

1. Audible alert if the relative humidity is too high or too low
2. Low power, last months on a single battery charge
3. No larger than a small jar

Parts List

This project uses the following parts:

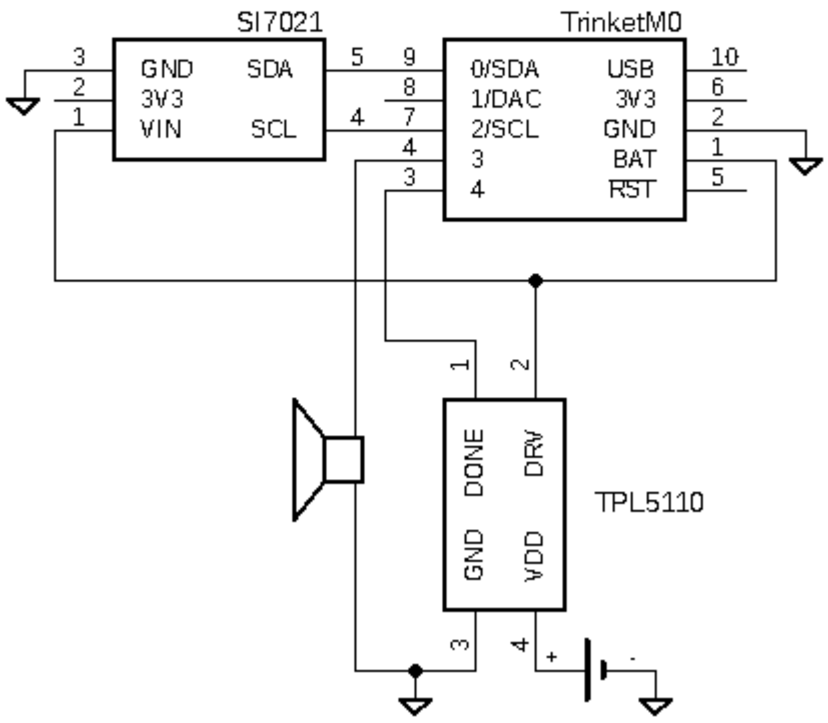
1 x Adafruit Trinket M0 Adafruit Trinket M0 - for use with CircuitPython and the Arduino IDE	https://www.adafruit.com/product/3500

1 x TPL5110 Breakout Adafruit TPL5110 Low Power Timer Breakout	https://www.adafruit.com/product/3435
1 x Si7021 Breakout Board Adafruit Si7021 Temperature and Humidity Sensor Breakout Board	https://www.adafruit.com/product/3251
1 x Piezo Buzzer Piezo Buzzer - PS1240	https://www.adafruit.com/product/160
1 x Battery Connector JST-PH 2-Pin SMT Right Angle Breakout Board	https://www.adafruit.com/product/1862
1 x LiPo Battery Lithium Ion Polymer Battery - 3.7v 500mAh	https://www.adafruit.com/product/1578
1 x LiPo Charger Adafruit Micro Lipo - USB LiIon/LiPoly charger - v1	https://www.adafruit.com/product/1304

Hardware

When this need arose, I was playing around with the new SAMD21 boards. The Trinket M0 was one such board. Tiny yet capable enough to do plenty of useful things. It exposes an I2C interface so connecting sensors is easy. And I just needed one: a relative humidity sensor and I knew just the sensor for the job.

For my smart home sensor nodes, I am using the Si7021 temperature and relative humidity sensor. Adafruit has this on a breakout board which makes it easy to use. The breakout also has the benefit of being breadboardable while you are prototyping. My need was for a relative humidity sensor, but having a temperature sensor bundled in could be useful for some applications.



Additionally I needed some way of alerting when humidity is out of range. For this I used a plain piezo buzzer that my code can control the pitch of.

Humidity change in a closed environment should be slow, and those humidity control packs last quite a while, so I felt no need to have it checked frequently. Taking a humidity reading every couple hours is plenty. This also means less power being used over a period of time, so battery power is easier to deal with. I decided to use a 500 mAh lipo battery as it was about the same size as the space I was building for and would tuck under the circuit board. My deployed units have been working for 6 months on the initial charge.

To make this work I used the TPL5110 breakout that will power up the Trinket every so often (up to about every 2 hours). When the job is done a single output tells the TPL5110 to power down the Trinket and start the timer again. This makes the code much simpler: no loop, no sleeping, just check the humidity and shut down if it's good. If not, alert until it is, then shut down.

Software

If you are unfamiliar with programming the Trinket M0 with the Arduino IDE, you can get familiar with this board with the Introduction to the Adafruit Trinket M0 tutorial (https://adafru.it/Agg).

The code is pretty much as simple as the hardware. The falling/rising tone arrays contain a sequence of frequencies that are played in sequence by the `chirp()` function. Which set of tones is played is decided based on the argument to `chirp()`.

The `check_rh()` function reads the sensor and compares the humidity with the target, allowing for +/- 5% variability. If the reading is low, -1 is returned, +1 if it's high, and 0 if it's in range.

The `warn_if_out_of_range()` is passed the initial reading comparison result, and immediately returns if the reading was in range. If it was out of range it sounds the appropriate alarm, waits briefly, and checks again. This repeats until the reading is back in range (or the battery goes empty).

Being an Arduino sketch, there are the usual `setup()` and `loop()` functions. `setup()` initializing things and calls `warn_if_out_of_range()`. `loop()` simply toggles the shutdown signal to the TPL5110 to put the system to sleep.

The result is that the system wakes up, and if the reading is in range goes immediately back to sleep. Otherwise it stays awake periodically sounding the alarm until it goes back in range.

The current code only looks at the relative humidity. The Si7021 can also provide temperature readings which could be useful if desired.

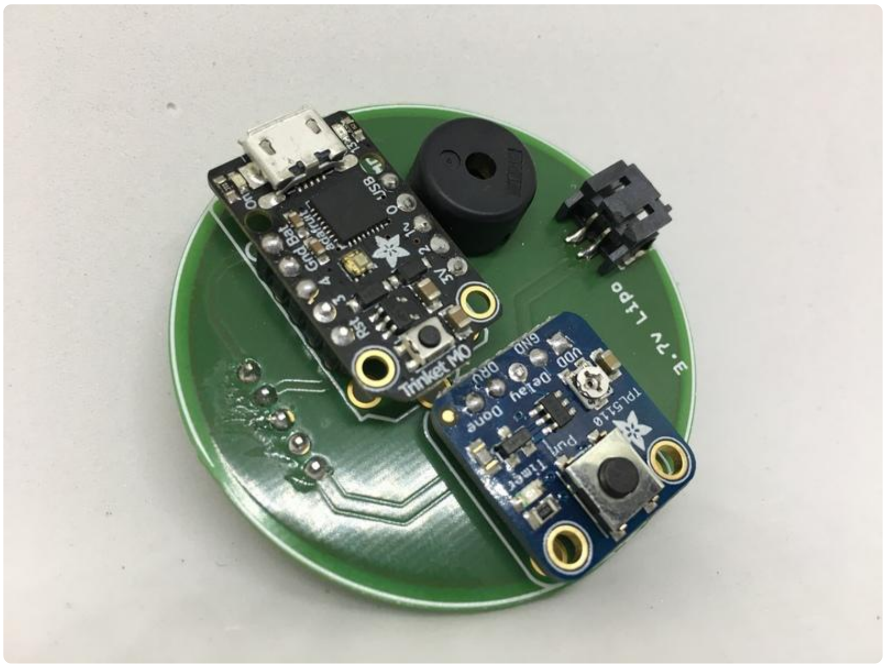
```
// Humidity Monitor
// Copyright (C) 2017 Dave Astels
// Released until the MIT license
//
// The trinket is controlled by an external Power Timer Breakout
//     (https://www.adafruit.com/product/3435)
// 1. Every however often, the timer will power up the trinket which will run
//    this file.
// 2. Relative humidity is checked
//    - if it's within range, the trinket tells the timer to shut it down and start
//      the timing cycle
//    - if it's out of range, it beeps & flashes the neopixel for 2 seconds, then
//      sleeps for 10 seconds, then goes to 2

#include "Adafruit_Si7021.h"
#include <Adafruit_DotStar.h>
#include <SPI.h>
```

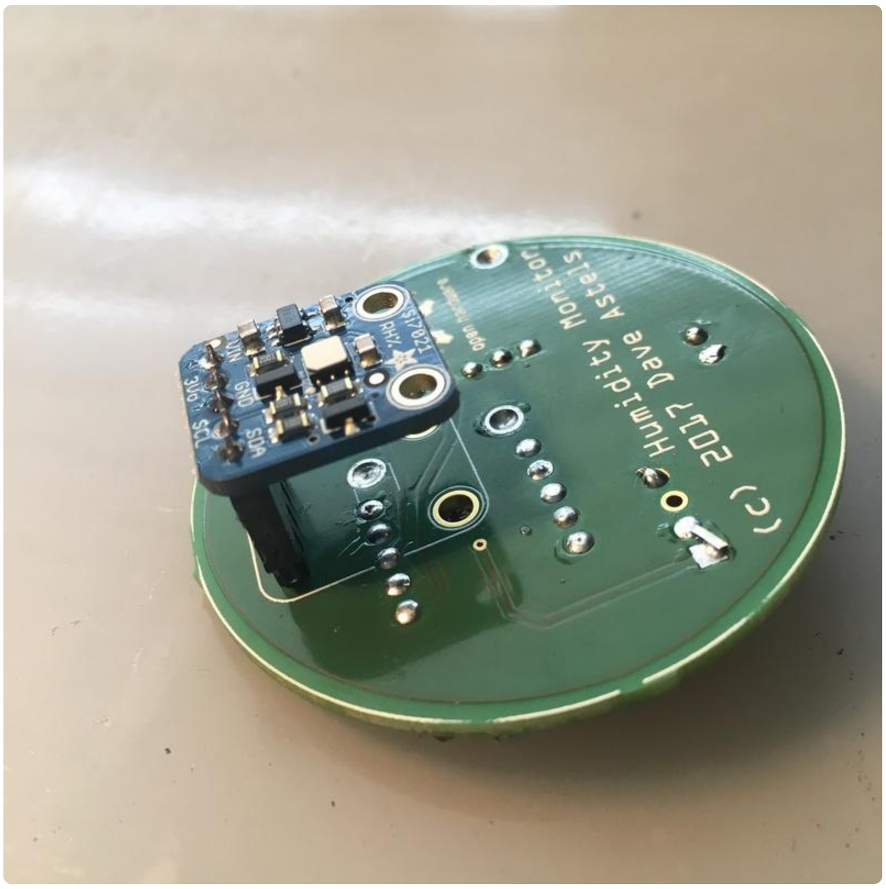
```
const unsigned long falling_tones[] = { 2000, 1980, 1960, 1940, 1920, 1900, 1880,
1860, 1840, 1820, 1800, 1780, 1760, 1740, 1720, 1700, 1680, 1660, 1640, 1620 };
const unsigned long rising_tones[] =  { 2000, 2020, 2040, 2060, 2080, 3000, 3020,
3040, 3060, 3080, 4000, 4020, 4040, 4060, 4080, 5000, 5020, 5040, 5060, 5080 };

const int dotstar_data_pin = 7;
const int dotstar_clock_pin = 8;
const int sound_pin = 3;
const int sleep_pin = 4;

const int target_humidity = 61.0;
const unsigned long alert_interval = 20000;
const unsigned long alert_duration = 500;
const unsigned long comparison_delay = alert_interval - alert_duration;
const unsigned long number_of_freq_steps = 20;
const unsigned long alert_freq_step_time = alert_duration / number_of_freq_steps;

Adafruit_Si7021 sensor = Adafruit_Si7021();
//Adafruit_DotStar strip = Adafruit_DotStar(1, DOTSTAR_BRG);

//------------------------------------------------------------------------------
// Sound the alert.

void chirp(boolean direction)
{
  const unsigned long *freqs = direction ? rising_tones : falling_tones;
  for (int i = 0; i < number_of_freq_steps; i++) {
    tone(sound_pin, freqs[i]);
    delay(alert_freq_step_time);
  }
  noTone(sound_pin);
}


//------------------------------------------------------------------------------
// Measure relative humidity and return whether it is under/in/over range.

int check_rh()
{
  float relative_humidity = sensor.readHumidity();
  if (relative_humidity < (target_humidity - 5)) {
    return -1;
  } else if (relative_humidity > (target_humidity + 5)) {
    return 1;
  } else {
    return 0;
  }
}

//------------------------------------------------------------------------------
// Repeatedly check the humidity and light the neopixel and sound the buzzer as
// appropriate, for as long as the reading is out of range.
// Initial comparison result is passed in.

void warn_if_out_of_range(int comparison)
{
  while (comparison != 0) {
    chirp(comparison > 0);
    delay(comparison_delay);
    comparison = check_rh();
  }
}



void setup()
{
  pinMode(sleep_pin, OUTPUT);
  digitalWrite(sleep_pin, LOW);
  //  strip.setPixelColor(0, 0);
  //  strip.show();

  sensor.begin();
  warn_if_out_of_range(check_rh());

}

void loop()
{
  digitalWrite(sleep_pin, HIGH);
  delay(50);
  digitalWrite(sleep_pin, LOW);
  delay(50);
}
```

Building

I made a prototype board to test out the design. Pictures below show the sequence of build steps. This can be assembled on a breadboard (https://adafru.it/Bgk) or protoboard (https://adafru.it/oIC) easily, and the circuit board shown is available from Aisler (https://adafru.it/BgN).

You will need to solder the supplied male header pieces on the sides of the breakouts. This may best be done with the header trimmed to the proper number of pins and pressed into a breadboard then placing the breakout board on top. You may wish to refer to the tutorial Collin's Lab: Soldering (https://adafru.it/Bgl).



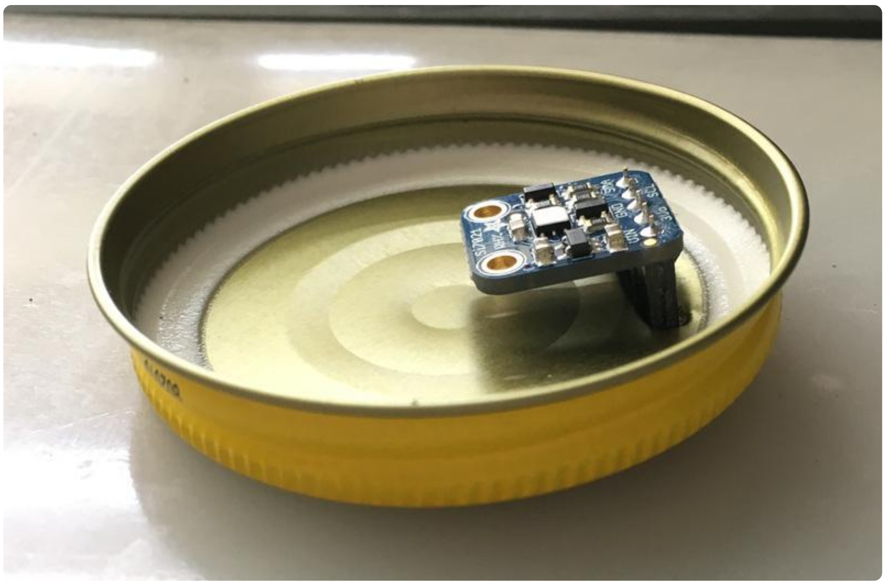
Si7021
RH/T
VIN
GND
SDA
SCL
3Vo

Humidity Monitor
2017 Dave Astels
(c)
Si7021
RH/T
VIN
GND
SDA
SCL
3Vo

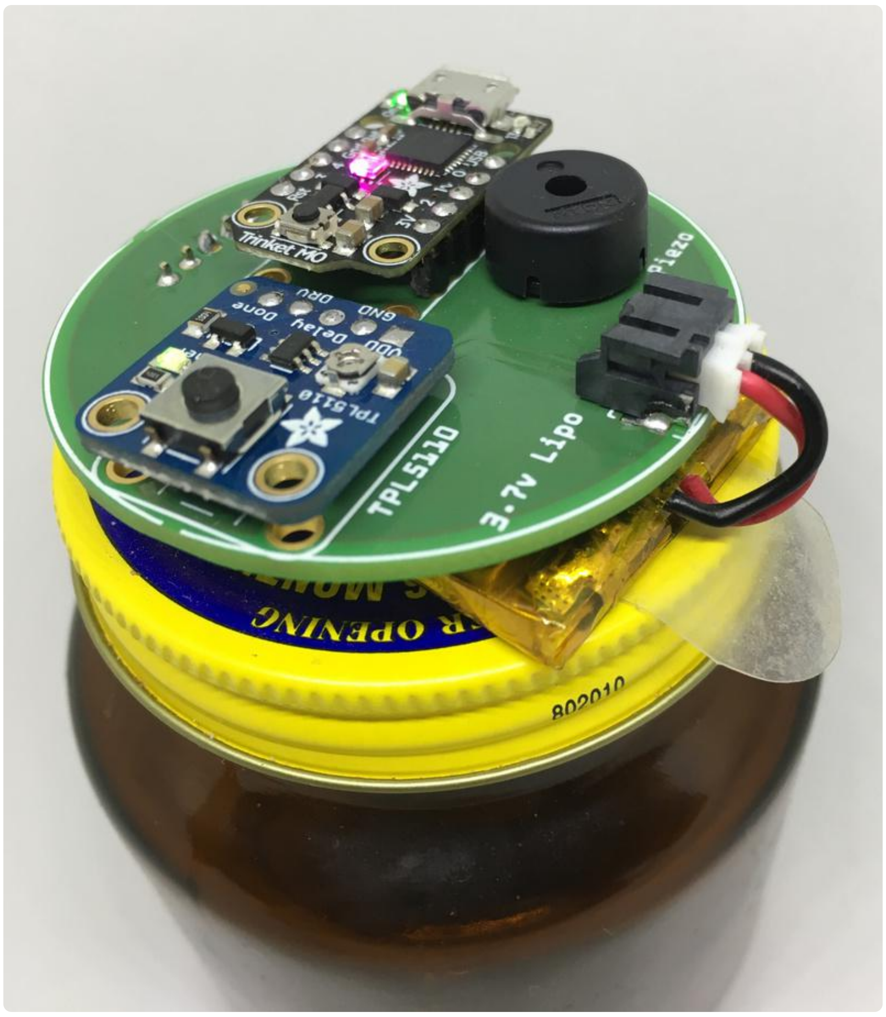
The sensor is placed inside the jar and, for my build, long headers are used to go through the lid of the jar to the electronics on top of the lid.

A four conductor wire could also be used between the board in the jar and the board outside.

Be sure to seal the lid connection so humidity cannot get inside. Also, do not let metal lid edges touch bare wires which could cause an electrical short circuit.

REFRIGERATE AFTER OPENING
THEN USE WITHIN 6 MONTHS
ENSUITE
M MAY 12 2019
D1
SUIVANTS
UTILISER DANS LES 6 MOIS
REFRIGERER APRES L'OUVERTURE





The design has proven itself and prototype units have been working reliably for 6 months on the initial battery charge. A more productized version 2 is currently being designed.