How to get reliable readings from photoresistors

Robert Hongpu Ma (Team: SnoozeBuster)

Dec 6, 2015

CMPT 433 Embedded System, Fall 2015

Pavel Kozlovsky, Hongpu Ma, Joey Parker, JunXuan Guan

Basics

A photoresistor, a.k.a. Light-Dependent Resistor (LDR), are resistors that are sensitive to incident light intensity. Generally, the resistance increases along with the light intensity. A common photoresistor has a characteristic curve determined by its material and manufacturing. Most of the time the correspondence is not linear or logarithmic, and is also affected by the wavelength of the light.

There are two other common (and more expensive) components similar to a basic photoresistor: photodiodes and "advanced light sensors". Photodiodes are more sensitive and have polarity. The term "light sensors" is a bit confusing, which can be used to refer to any component that is sensitive to light. Here we are more interested in those somewhat "advanced" light sensors that have amplifiers or logic circuits built in.

Problems

In this guide, we are going to solve two problems:

1. Wiring - how to wire a photoresistor to the board via ADC, which is the easy part.
2. Debouncing - how to debounce the readings without losing responsiveness.

These two problems arise because of the difference between simple photoresistors and advanced light sensors. First, packaged sensors commonly have a pair of positive and negative pins, plus an (analog) reading pin. Hence, it is straightforward to wire. Second, cheap photoresistors are so simple that their light-resistance correspondence is not mathematically beautiful. Furthermore, their characteristics can differ even within the same batch, and the readings are not stable enough in two senses: when the light intensity suddenly changes, the readings will bounce around wildly shortly before it stabilizes; even in an almost fixed environment, the photoresistor sometimes reads out unreasonably high or low values and becomes normal again after a short while.

Solutions

Wiring

Wiring is simple once we realize these two analogies: photoresistors are still resistors; Analog-to-Digital Converters (ADC) are basically voltmeters (at least on a lot of development boards), with the AIN pin as

"+" and the (ADC) ground as "-". Recall your high school physics: inserting a voltmeter into a series circuit will not help you much; also, if there is only one variable resistor in a simple circuit with power, its voltage will barely change regardless of its resistance (in a reasonable range).

The answer should now be obvious: insert another resistor R_s in series, and measure the voltage on the photoresistor (measuring the voltage on R_s will give an inversed correspondence). Here is a demo wiring (a trimmer/potentiometer is used as the other resistor, which will be explained later):

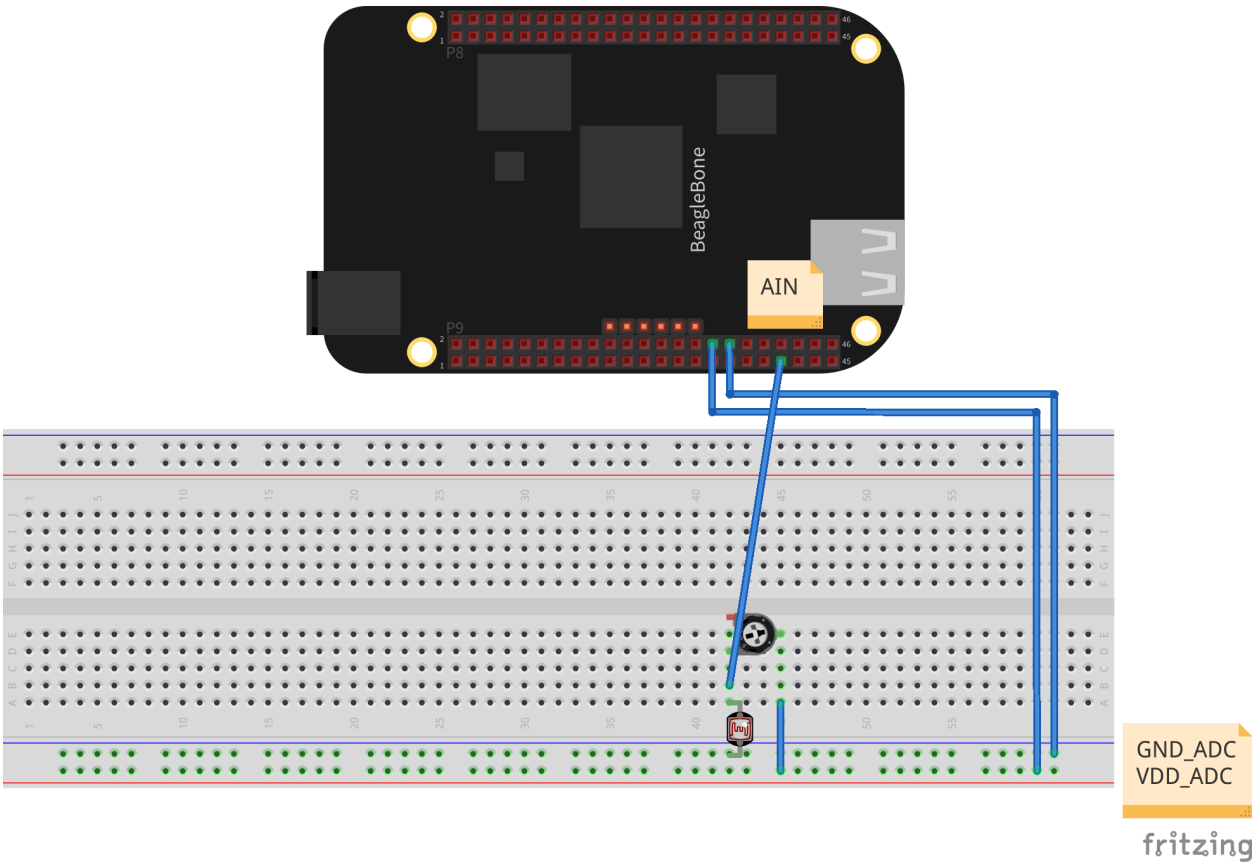

Figure 1: LDR Wiring

Assume the current resistance of the photoresistor is R_p, the resistance of the trimmer is R_s (fixed), and the voltage between VDD_ADC and GND_ADC is V. The ADC will read out the following raw voltage:

$$V_{out} = \frac{R_p}{R_p + R_s} V$$

The stronger the light is, the higher R_p, and thus the higher ADC reading.

R_s should be chosen to be close (at the same magnitude) to the resistance range of the photoresistor. Furthermore, if a trimmmer is used here, it can be tweaked to compensate the difference among different photoresistors so that we can get a somewhat identical range.

Debouncing

The major problem we want to overcome is the wild bounces and random fluctuations even, either caused by the ADC or the photoresistor itself. Anyway, a classic algorithm called "moving average" can be used

in this case. We build a ring buffer with the size of $N_{samples}$ and use a background thread to periodically poll from the photoresistor every *interval* period of time, pushing into the ring queue (overwriting existing values, if any). When we want a reading from the photoresistor, we calculate and return the average of the whole ring buffer instead of directly fetching a new reading from ADC. The query most likely happens on another thread, so pay attention to synchronization!

Here is a C-like pseudo code:

```
struct ring_queue_t {
    int values[N_SAMPLES];
    int position = 0;
    mutex_t mutex;
} ring_buffer;

void background_poll() {
    while (1) {
        int reading = read_from_ADC();
        lock(ring_buffer.mutex);
        values[position] = reading;
        if (++position >= N_SAMPLES)
            position = 0;
        unlock(ring_buffer.mutex);
        sleep(interval);
    }
}

float query() {
    int sum = 0;
    lock(ring_buffer.mutex);
    for (int i = 0; i < N_SAMPLES; i++)
        sum += ring_buffer.values[i];
    unlock(ring_buffer.mutex);
    return (float)sum / N_SAMPLES;
}
```

Several notes on the code:

1. Based on my experiment, $N_{samples} = 100$ and $interval = 1ms$ gives a very reliable and stable reading. And the responsive time is 0.1 seconds, still faster than human reaction time. However, it will lead to a CPU load of $30\% \sim 50\%$. Consider your specifications and make the right trade-off.
2. The code is not optimized. Something can be done to remove a lot of unnecessary computation or even the need for a lock in the query. Try it yourself.
3. The code does not take into account the initial edge case, where the ring buffer is not completely filled.

Troubleshooting and Notes

1. Please refer to Brian's ADC guide about how to read from ADC.
2. Keep in mind the sysfs path for the raw_voltage in ADC guide can change across reboots. The number postfix depends on the order kernel initializes all the hardware, which is not certain. Some heuristics are needed to locate the right path.
3. Get a voltmeter or multimeter if you want to play with all these stuff.
4. Check and double check all the wiring. Inserting a pin is not as firm as assignments.

5. Consider buying a packaged light sensor if all this above bothers you rather than interests you.
6. If you like the diagram in this guide, try Fritzing out!