
Make:
Sensors
Projects and Experiments to Measure the World with Arduino and Raspberry Pi
Tero Karvinen, Kimmo Karvinen & Ville Valtokari

Light

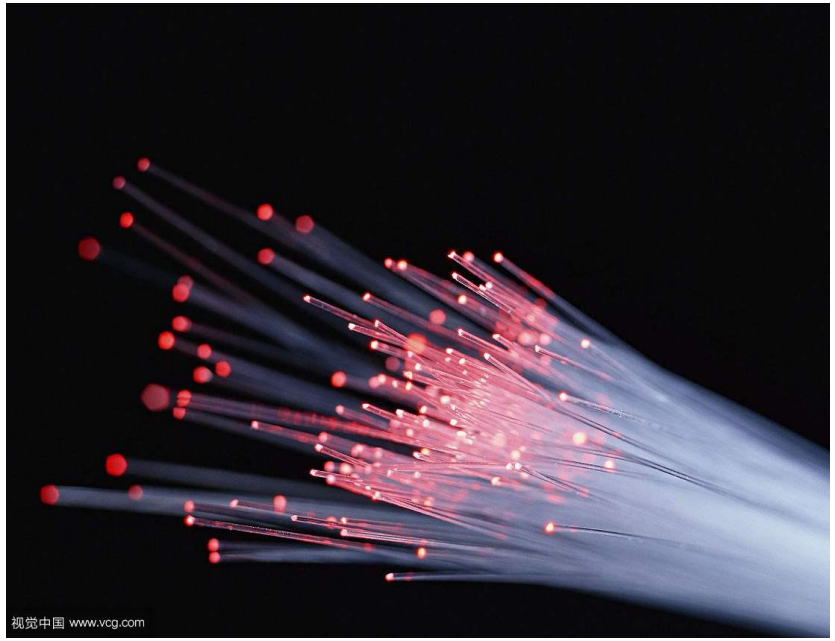
Light

- A robot follows a complicated path, seemingly without difficulty.
- A closer look shows a black line on the floor for the bot to follow.

- Later in the evening, backyard lights light up automatically when darkness falls.

Light

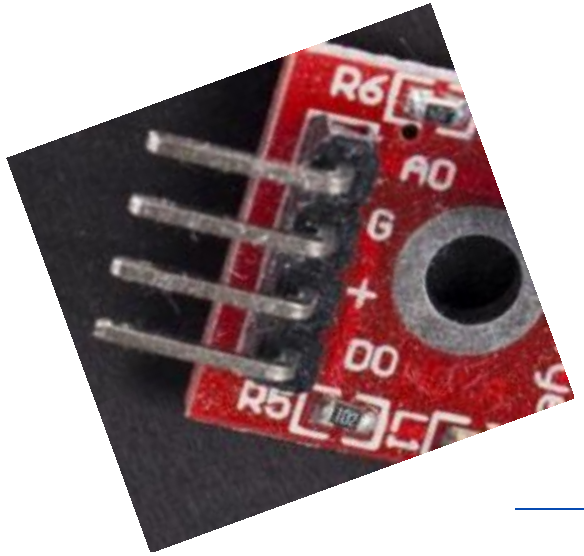
- Because color is reflected light, sensors can detect the color of a surface.
- With some creatively applied tubing, the direction of light can be detected, too.
- And if fire is the thing for your bot, there is a sensor for flame.

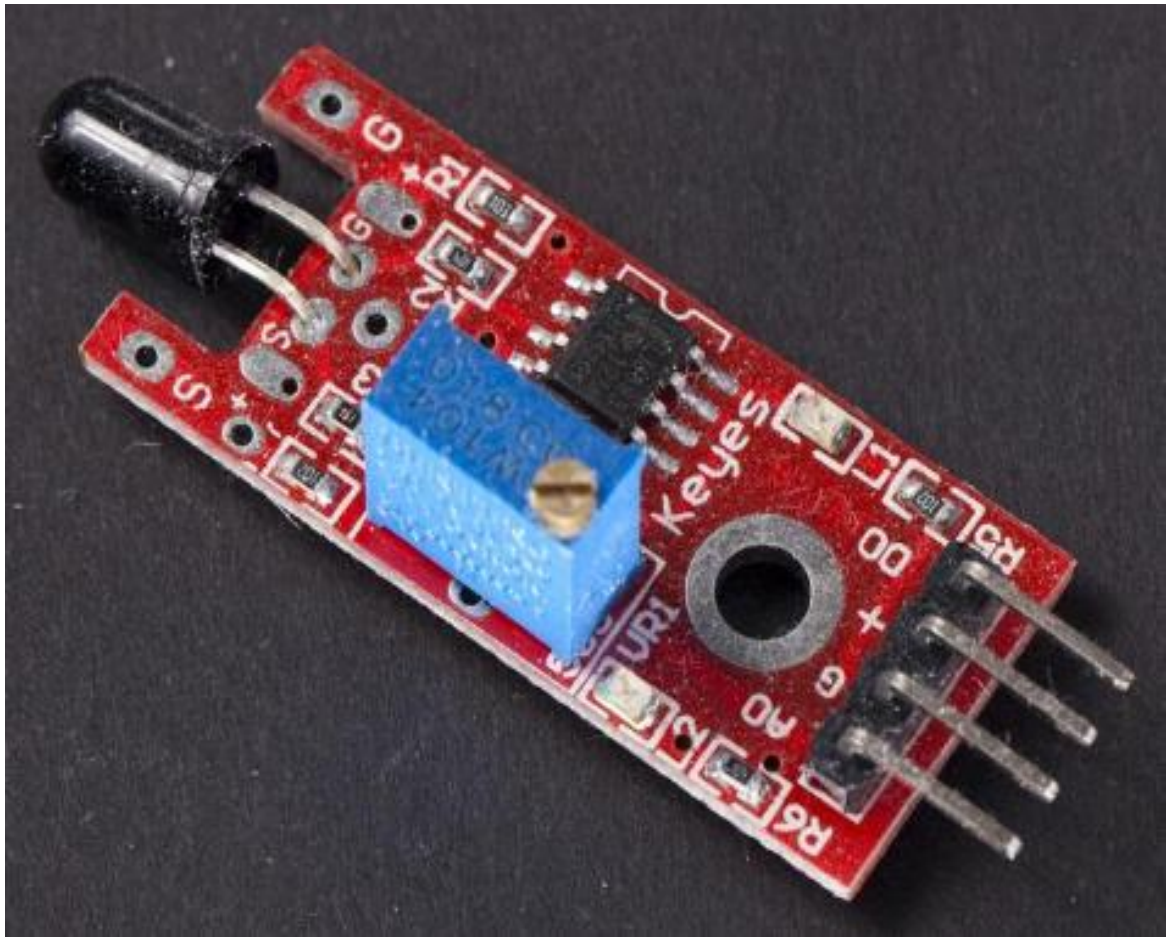


Contents

Experiment: Detecting Flame (Flame Sensor)

- Flames emit a range of infrared light not very common in ambient light.
- The KY-026 flame sensor reports the level of infrared light with a change of resistance (Figure 7-1).





Experiment: Detecting Flame (Flame Sensor)

- The KY-026 flame sensor provides two ways to measure flame:
  - digitalRead() analogRead()

- The code you'll write for flame detection is the same code you'd use for an analog resistance sensor: you use analogRead() to read the voltage of a pin.
- Using digital mode only is especially convenient with Raspberry Pi, because Raspberry Pi doesn't have an analog-to-digital converter.

Experiment: Detecting Flame (Flame Sensor)

Figure 7-2. Flame-following robot prototype (robot workshop in Austria)

Flame Sensor Code and Connection for Arduino

- Figure 7-3 shows the wiring diagram for the flame sensor with Arduino. Wire it up as shown, and then run the sketch shown in Example 7-1.

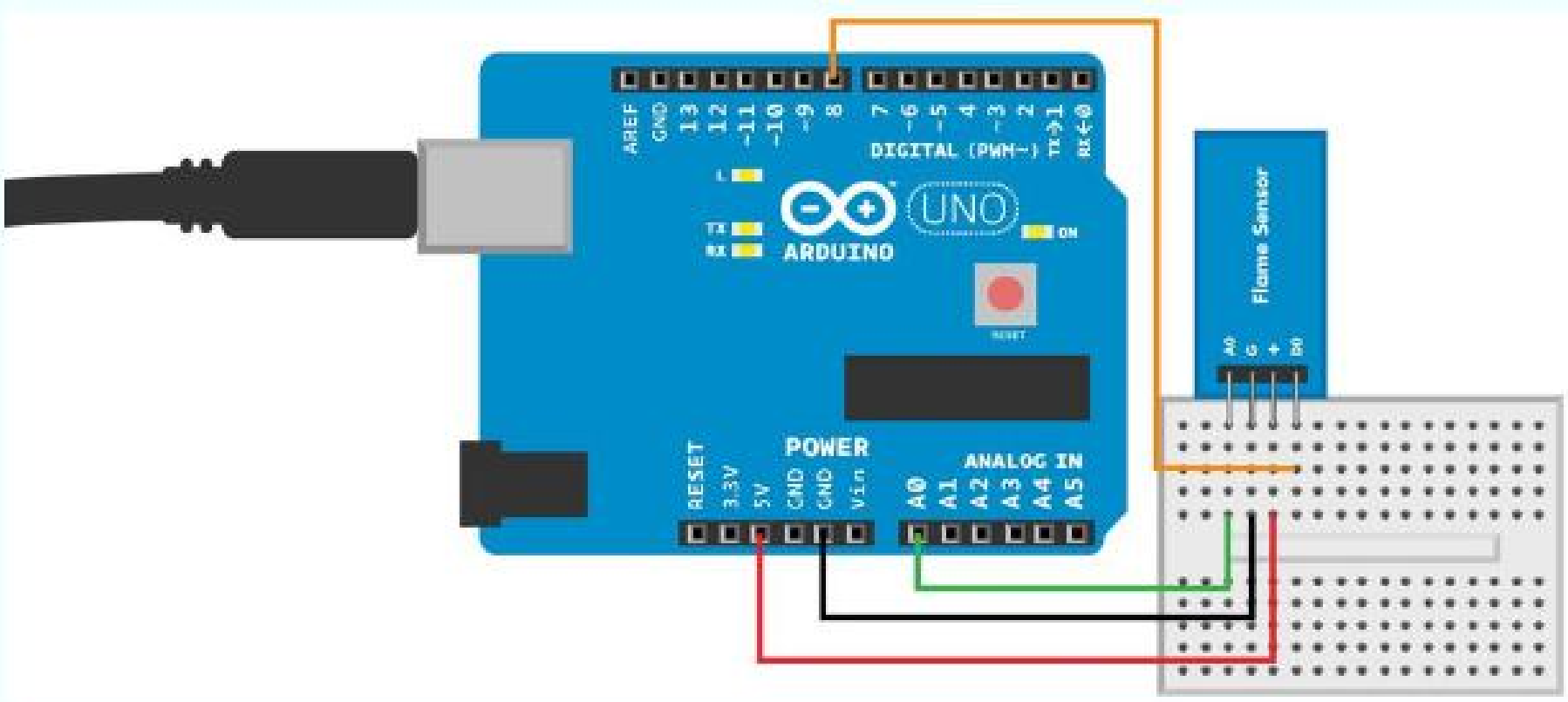


Figure 7-3. Flame sensor circuit for Arduino

Example 7-1. ky_026_flame.ino

// ky_026_flame.ino - report level IR light from flame to serial
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
const int analogPin = A0;
const int digitalPin = 8;
const int ledPin = 13;

void setup() {
    Serial.begin(115200);
    pinMode(digitalPin,INPUT);
    pinMode(ledPin,OUTPUT);
}
```

```
void loop(){
        int threshold = -1;          // 数字端口非0即1
        int value = -1;              // 模拟端口0..1023
        value = analogRead(analogPin);          //❶
        threshold = digitalRead(digitalPin);    //❷
        Serial.print("Raw: ");
        Serial.print(value);
        Serial.print(" Over threshold: ");
        Serial.println(threshold);
        delay(10);
        if(threshold==HIGH) {                   //❸
                digitalWrite(ledPin, HIGH);
        }else {
                digitalWrite(ledPin, LOW);
        }
}
```

Flame Sensor Code and Connection for Raspberry Pi

- Figure 7-4 shows the circuit for connecting the sensor to a Raspberry Pi.
- Wire it up and run the program shown in Example 7-2.

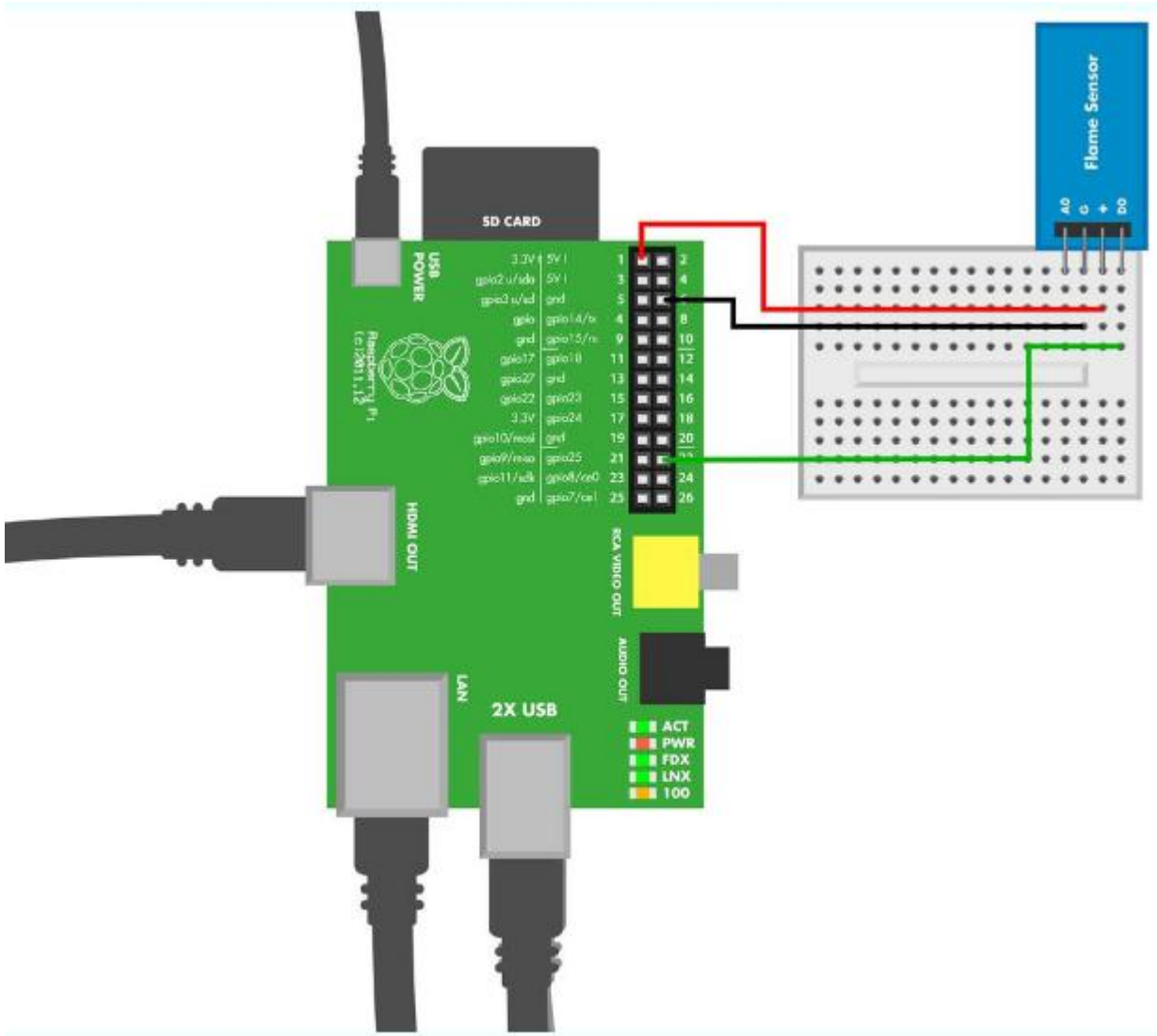


Figure 7-4. Flame circuit for Raspberry Pi

Example 7-2. ky_026_flame.py

```
# ky_026_flame.py - report presence of IR light from flame to serial
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio      # ❶

def main():
    triggerPin = 222
    gpio.mode(triggerPin, "in")    # ❷
    flame = gpio.read(triggerPin) # ❸
    if(flame == gpio.HIGH):        # ❹
        print "Flame detected"
    else:
        print "No flame detected"
    time.sleep(0.5)

if __name__ == "__main__":
    main()
```

Environment Experiment: Flame Precision

- The flame sensor's built-in sensitivity resistor is very useful. It is especially important to adjust it so that ambient light won't trigger the sensor constantly. It can also be used to set the flame sensor to react to very specific level of flame.

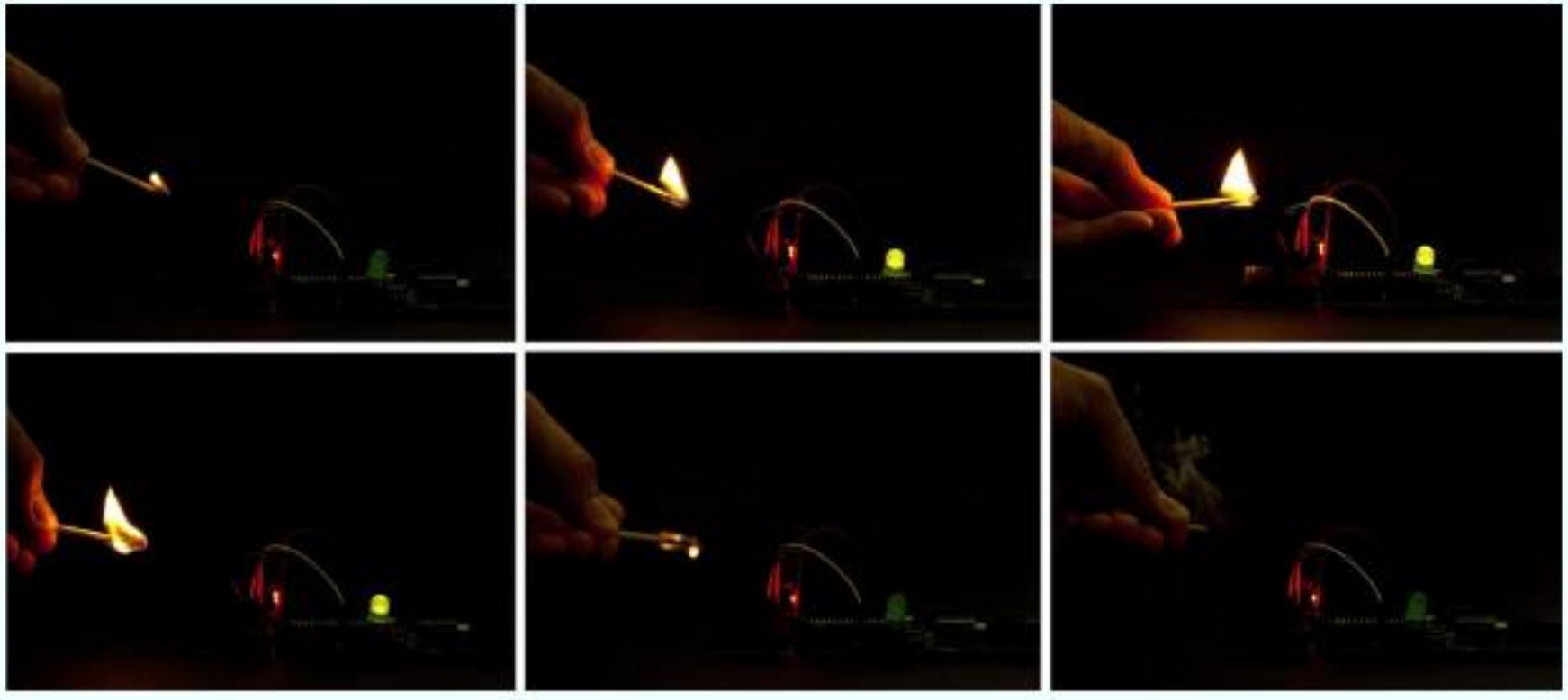
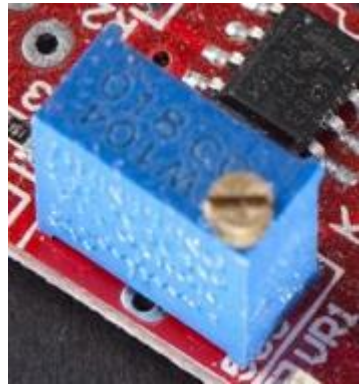


Figure 7-5. Adjusting and testing flame sensor sensitivity

Environment Experiment: Flame Precision

- First, put an extra LED between GND and pin 13 (see Figure 7-6). It's easier to see a full-sized LED than the onboard LED, so you'll be more likely to notice when the sensor is activated.
- Upload the flame sensor code (Example 7-1) to your Arduino.
- Turn the potentiometer all the way right, then turn it left until the LED goes out.
- It's probably a good idea to close the curtains,

  as strong sunlight can overpower other light sources.



Environment Experiment: Flame Precision

- Now, light a match. The LED should light again.
- Try adjusting the sensitivity potentiometer so that the sensor will react only to a full-size flame.
- Don’t get distracted watching the LED, or you might burn your fingers as the match burns down!

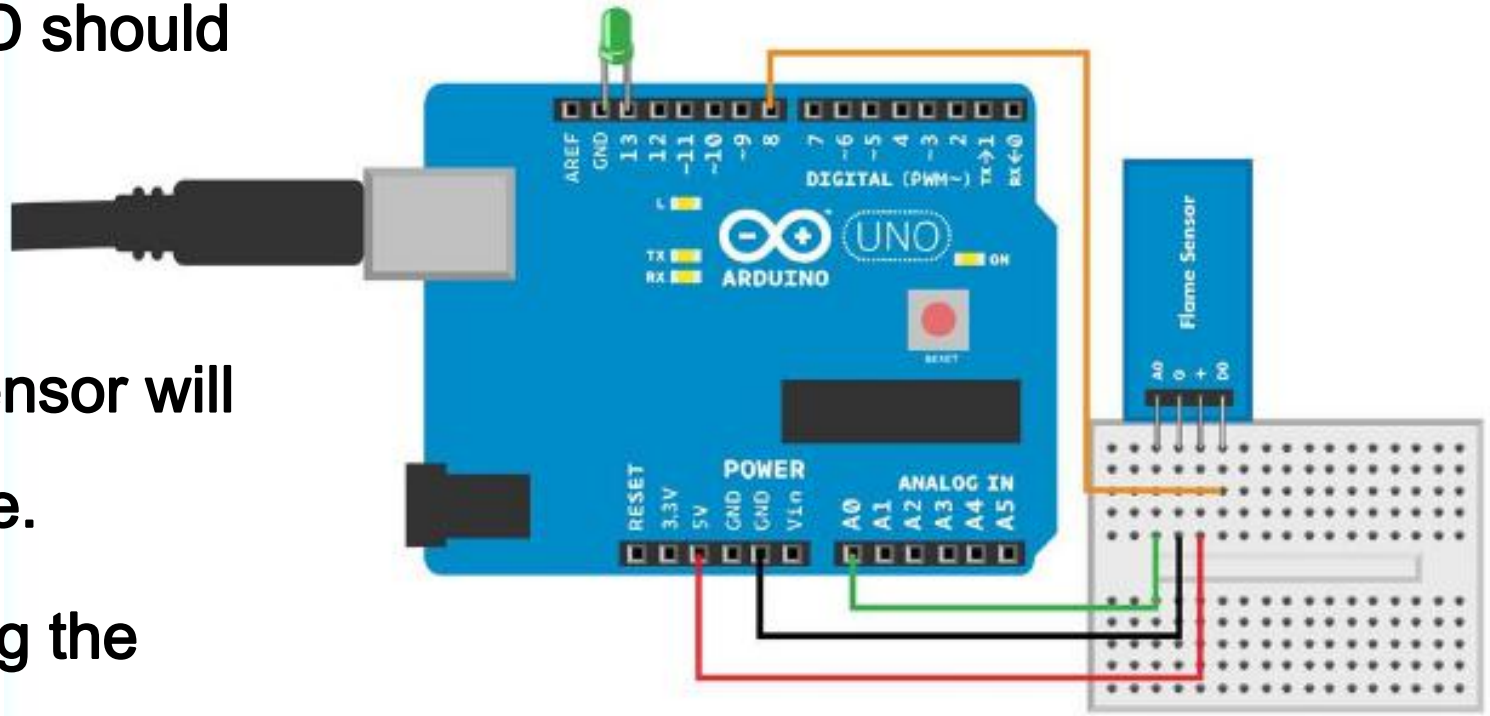


Figure 7-6. Flame sensor with LED

Contents

Experiment: See the Light (Photoresistor, LDR)

- A light-dependent resistor (LDR) changes its resistance according to the level of visible light. Its resistance is lower in bright light. LDR is also known as a photoresistor (see Figure 7-7).

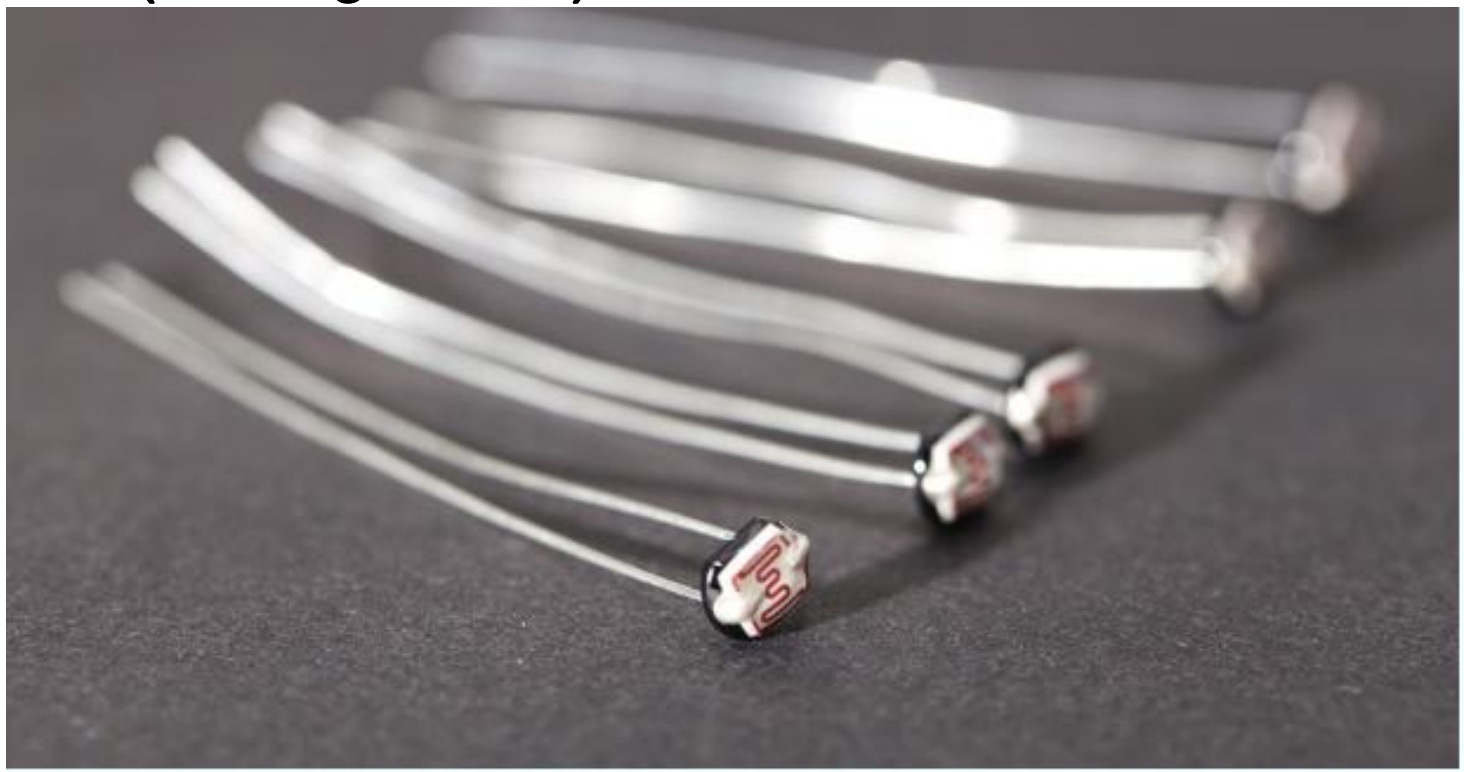


Figure 7-7. Photoresistor

Experiment: See the Light (Photoresistor, LDR)

- Photoresistors can turn on the lights when it's dark, detect if a dark box is opened in a lit room, and help create robots that love light.
- With some creative use of heat shrink tubing, an LDR can also detect the direction of light.

- When testing, you can simply put your finger on an LDR to make darkness fall. If you want bright light, you can point a flashlight at the LDR.

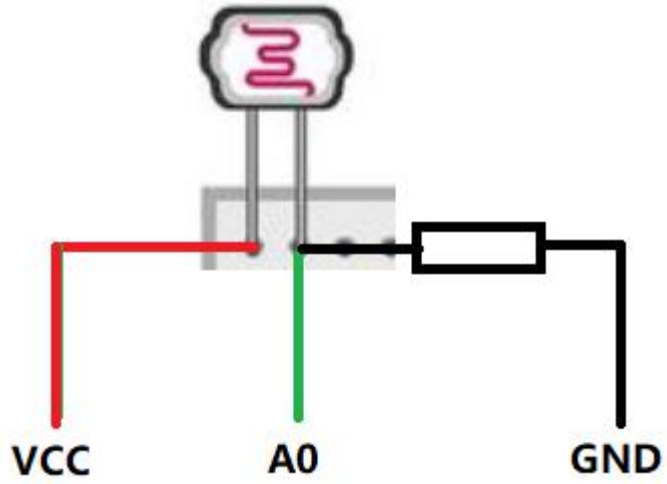
Experiment: See the Light (Photoresistor, LDR)

- To test a light-seeking robot in your lab, try covering it with a blanket (see Figure 7-8). This way, you don't have to shut down the lights in the lab or run between your lab and a dark room.

Figure 7-8. Ambient light hideout (robot workshop in Austria)

LDR Code and Connection for Arduino

- A photoresistor is just a two-legged variable resistor.
- With Arduino, it's just a matter of configuring the sensor with another resistor as a voltage divider, then using analogRead().
- In this way, a photoresistor is similar to many other analog resistance sensors.



LDR Code and Connection for Arduino

- Wire up the LDR as shown in Figure 7-9, and then run the sketch shown in Example 7-3.

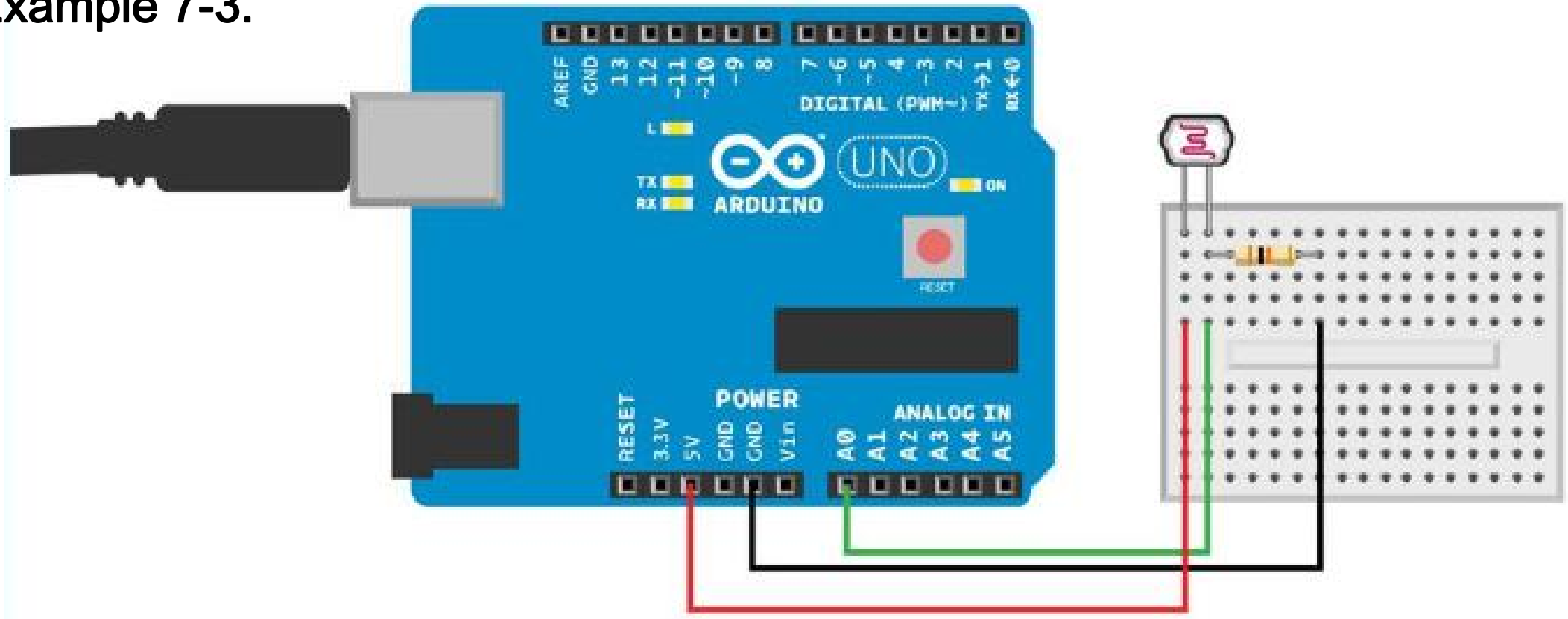


Figure 7-9. Photoresistor circuit for Arduino

Example 7-3. ldr_light_sensor.ino

```
// ldr_light_sensor.ino - report high level of light with built-in LED
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int lightPin = A0;
const int ledPin = 13;
int lightLevel = -1;

void setup() {
    Serial.begin(115200);
    pinMode(ledPin, OUTPUT);
}

void loop() {
        lightLevel = analogRead(lightPin);    // ❶
        Serial.println(lightLevel);
        if (lightLevel < 400) {                    // ❷
                digitalWrite(ledPin, HIGH);
        }
        else {
                digitalWrite(ledPin, LOW);
            }
        delay(10);
}
```

LDR Code and Connection for Raspberry Pi

- Connect the components as shown in Figure 7-10, and then run the code shown in Example 7-4.

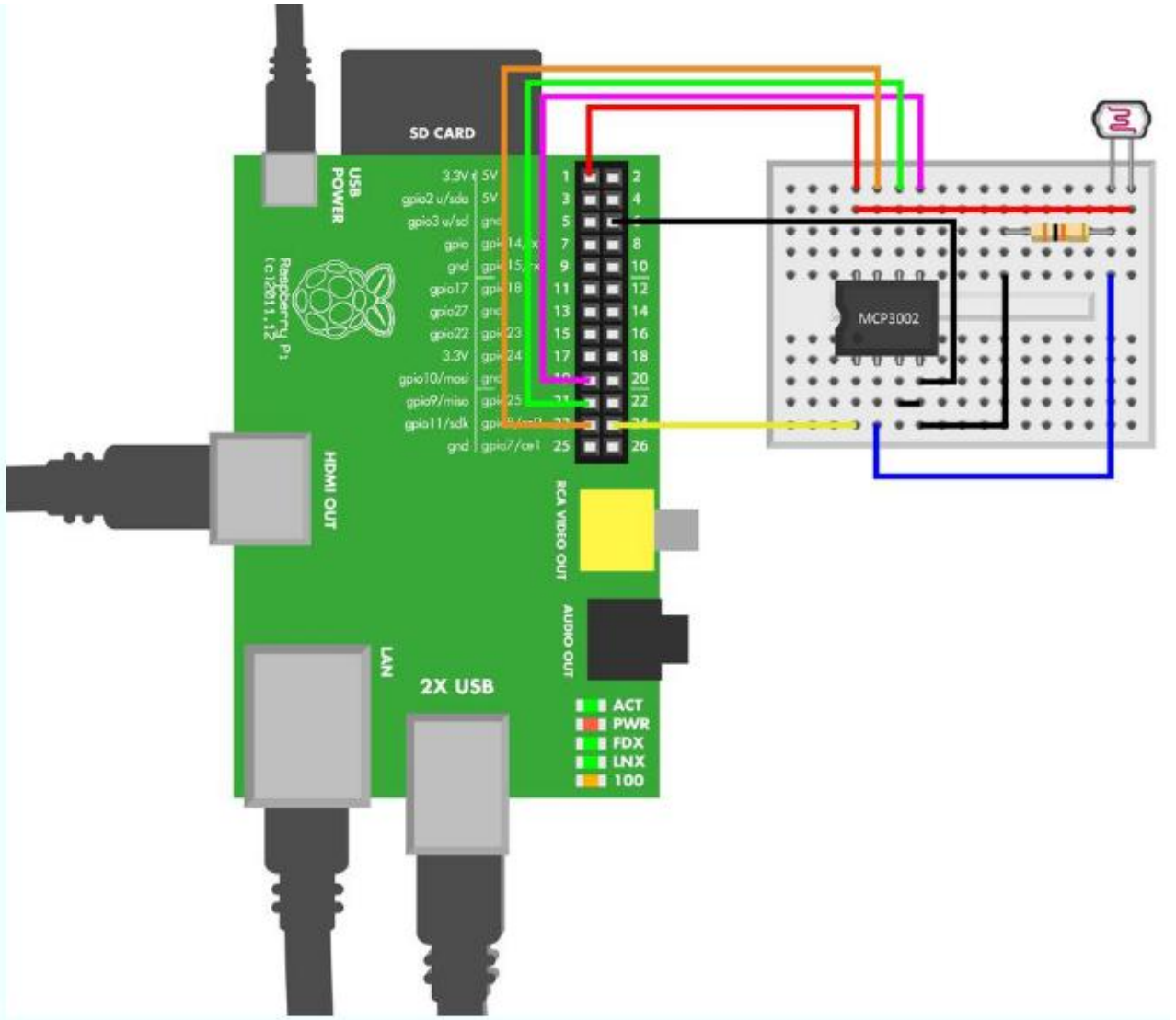


Figure 7-10. Photoresistor circuit for Raspberry Pi

Example 7-4. ldr.py

```
# ldr.py - sense light level and print to screen
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp

def main():
    while True:
        lightLevel = mcp.readAnalog()
        print("Current light level is %i " % lightLevel)
        time.sleep(0.5)

if __name__ == "__main__":
    main()
```

Environment Experiment: One Direction

- Would you like to know the direction that light is coming from, rather than just how bright it is?

- A naked photoresistor reacts to light coming from around it so you can't use it.

- for example, to turn a robot toward the light source.

Environment Experiment: One Direction

- There's a very easy solution for this. Take a piece of heat-shrink tubing and make a hood for the sensor, as shown in Figure 7-11.
- This prevents it from uncontrollably seeing light from every direction.

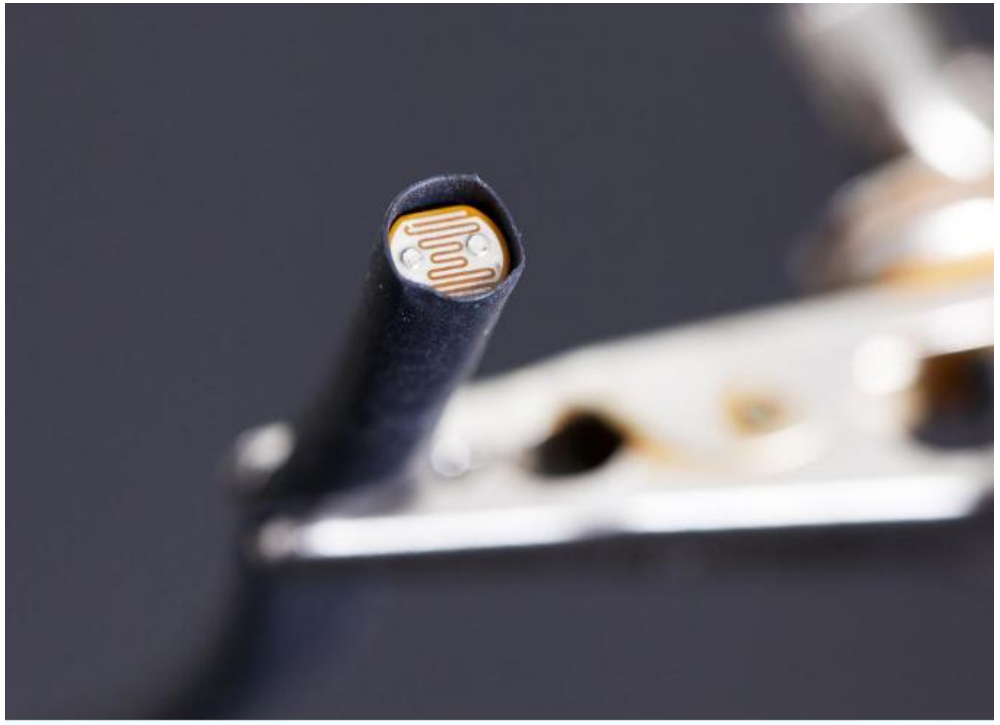


Figure 7-11. Preventing light from reaching the sensor

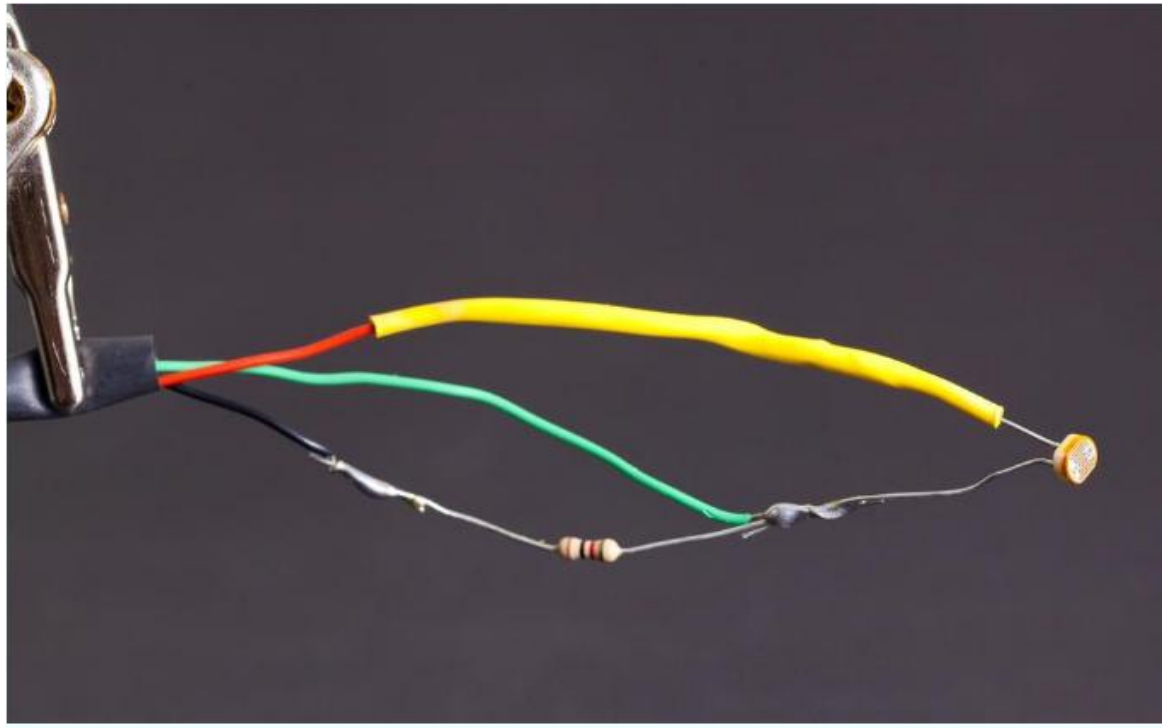


Figure 7-13. With heat shrink tubing, you can neatly package the wires

Environment Experiment: One Direction

- You can use a material other than heat-shrink tubing as long as it blocks light coming in from the side (see Figure 7-12).

Figure 7-12. Photoresistors inside plastic cup blinkers (robot workshop in Austria)

Contents

Experiment: Follow the Line

- Line following is an easy way to move a robot along a predefined path.
- The most common use is creating “rails” with black tape. We have used line avoidance to keep our mind (EEG) controlled robot inside its playground. Figure 7-14 shows a line detector.

Figure 7-14. Line tracking sensor

Experiment: Follow the Line

- Line detectors light the surface below with light, usually infrared.

- The surface is considered “white” if enough light is reflected back; anything else is considered a line.

Experiment: Follow the Line

- To know if your bot is going off the line from left or right, you can use two or three line detectors side by side .
- for example, if the center detector sees a line, but the other two see white, you know you’re following the line.

- There are ready-made line detection sensors available that combine multiple sensors into one.

Line Sensor Code and Connection for Arduino

- Because you are using a line detector with three leads here, no pull-up resistor is needed.
- One lead is for positive, another for signal, and the third for ground.
- The circuitry on the board includes any resistors or other components needed.

Line Sensor Code and Connection for Arduino

- Wire up the circuit as shown in Figure 7-15, and run the sketch shown in Example 7-5

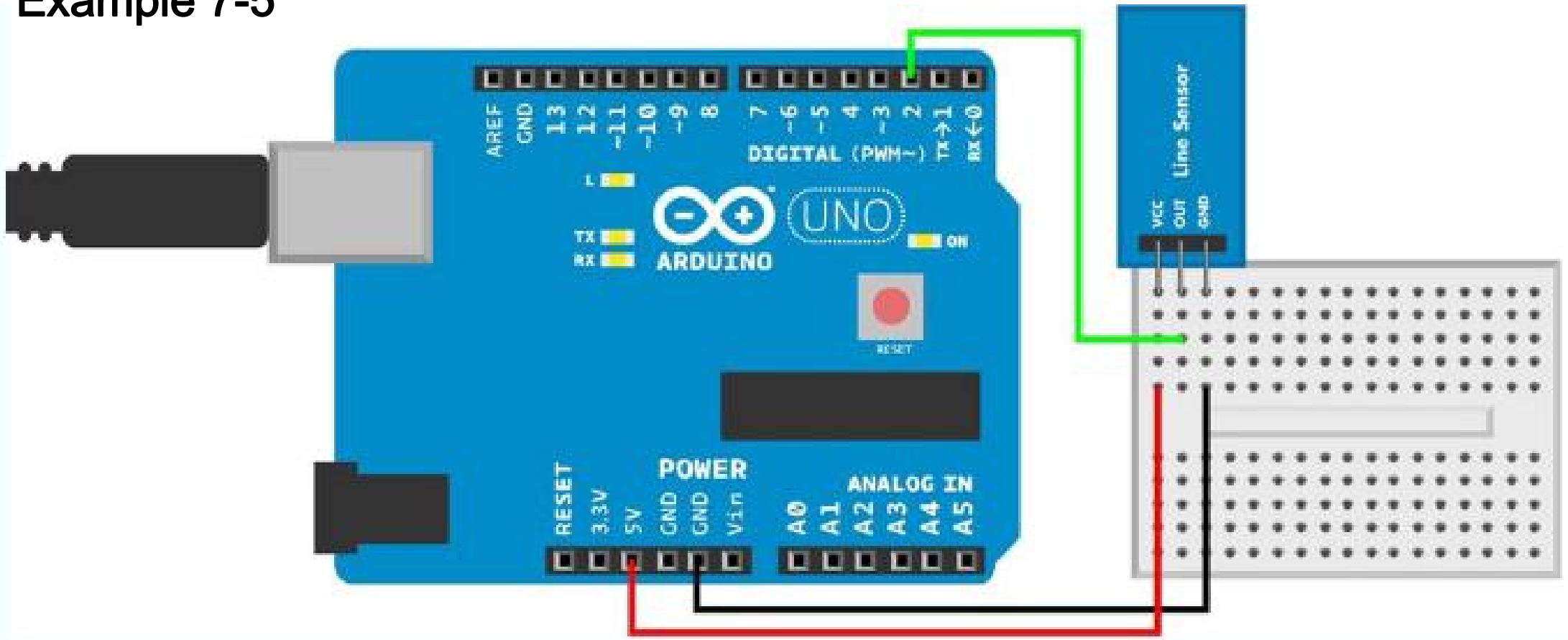


Figure 7-15. Line sensor circuit for Arduino

Example 7-5. line_sensor.ino

```
// line_sensor.ino - print to serial if we are on a line
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = 2;
const int ledPin = 13;
int lineFound = -1;

void setup() {
    Serial.begin(115200);
    pinMode(sensorPin, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
        lineFound = digitalRead(sensorPin);      // ❶
        if(lineFound == HIGH) {
                Serial.println("Sensor is on the line");
                digitalWrite(ledPin, HIGH);
        } else {
                Serial.println("Sensor is off the line");
                digitalWrite(ledPin, LOW);
        }
        delay(10);
}
```

Line Sensor Code and Connection for Raspberry Pi

- As a line sensor is a digital sensor, its connection to Raspberry Pi is a simple as with Arduino.
- Figure 7-16 shows the circuit diagram.
- Wire it up as shown, and then run the code shown in Example 7-6.

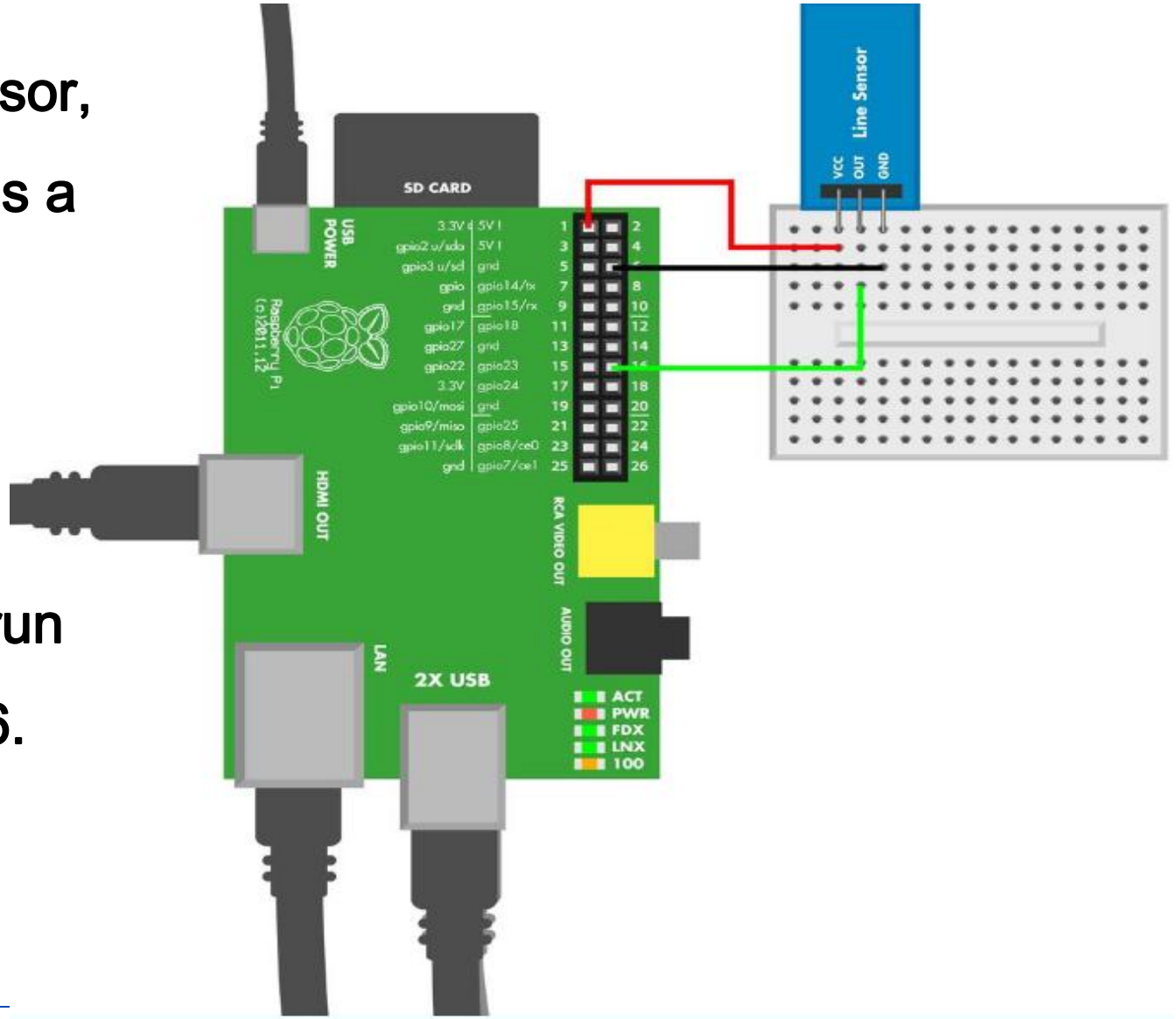


Figure 7-16. Line sensor circuit for Raspberry Pi

Example 7-6. line_sensor.py

```
# line_sensor.py - print to serial if we are on a line
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import os
import botbook_gpio as gpio                    #

def main():
    linePin = 23
    gpio.mode(linePin, "in")
    while True:
        lineState = gpio.read(linePin)         #
        if( lineState == gpio.HIGH ):
            print "Sensor is on the line"
        else:
            print "Sensor is off the line"
        time.sleep(0.5)

if __name__ == "__main__":
    main()
```

Environment Experiment: Black is White

- As you already know, an infrared sensor sees the world differently than we do.
- Different materials reflect light differently, and sometimes objects that appear dark can be so reflective that the sensor thinks they are white.
- If your line-following robot acts strange, it might be the surface texture—not the code—that's playing tricks on you.

Figure 7-17. Believe it or not, everything you see here is white

Environment Experiment: Black is White

- Usually black is black and white is white, but the total opposite has happened to us. When we presented our mind-controlled robot at Maker Faire, we brought tape and cardboard in order to make a platform with black borders and white center.
- The idea was that our robot would turn back when it saw black, keeping it on the platform.
- Surprisingly, the line detector saw tape and cardboard in inverted colors, as one was very reflective and the other very matte.

Environment Experiment: Black is White

- You can adjust the line follower sensitivity by adjusting its onboard potentiometer as shown in Figure 7-18.

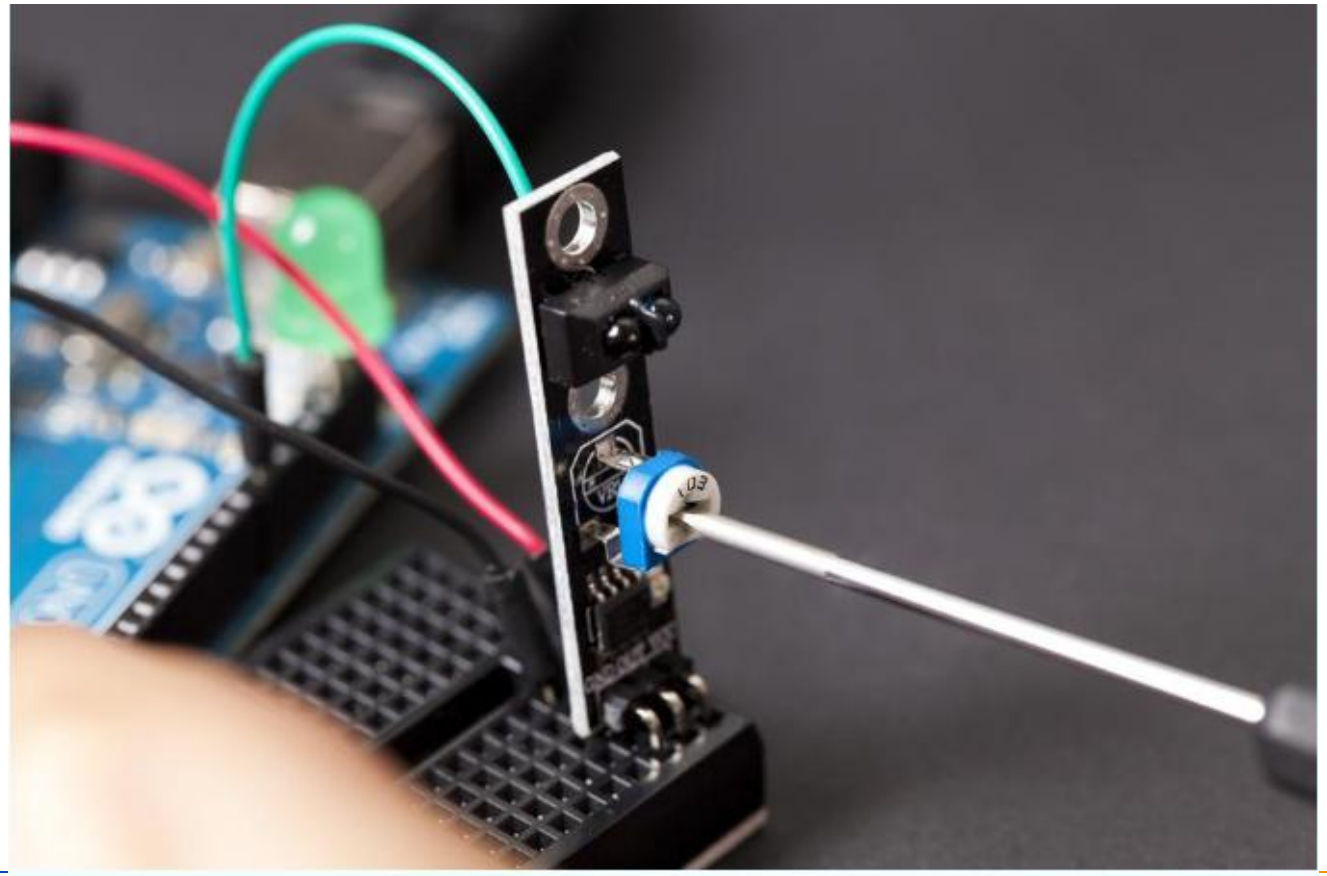


Figure 7-18. Adjusting line detector's sensitivity

Environment Experiment: Black is White

- Upload the code shown in Example 7-7 to Arduino.
- We have changed the previous example slightly so that now the serial monitor says BLACK or WHITE depending on what it sees.

Example 7-7. line_sensor_black_or_white.ino

```
// line_sensor_black_or_white.ino - line follow sensor. Signal low when over black line.
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = 2;
const int ledPin = 13;
int lineFound = -1;

void setup() {
    Serial.begin(115200);
    pinMode(sensorPin, INPUT);
    // 无需上拉电阻，因为传感器已经具备该功能.

    pinMode(ledPin, OUTPUT);
}
```

```
void loop() {
        lineFound = digitalRead(sensorPin);
        if(lineFound == 1) {
                Serial.println("BLACK");
                digitalWrite(ledPin, HIGH);
        } else {
                Serial.println("WHITE");
                digitalWrite(ledPin, LOW);
        }
         delay(50);
}
```

Contents

Experiment: All the Colors of the 'Bow

- A color sensor measures the color of a surface and returns values for red, green, and blue (Figure 7-19). For each basic color (red, green, blue), a color sensor has a color filter on top of a photodiode. The sensor for each color is read like any analog resistance sensor.

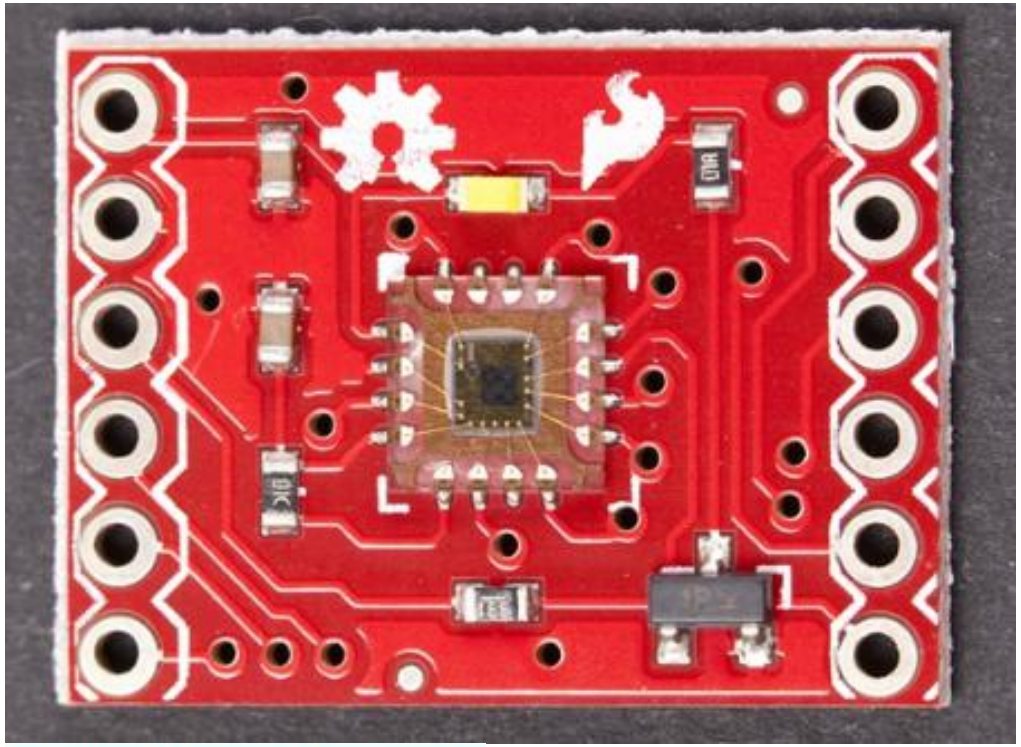


Figure 7-19. Color sensor

Experiment: All the Colors of the ’Bow

- The experiment prints RGB (red, green, blue) values of the light it sees, one value for each color.
- Colors are simply names for specific wavelengths of light. For example, 555 nanometers (nm, which describes the light’s wavelength) or 540 terahertz (THz, its frequency) is green.
- Some animals can see colors humans don’t see, like ones in the infrared or the ultraviolet range.
- The basic colors really are fundamental to humans. The only colors that the human eye can see are red, green, and blue. There are three types of cone cells in the retina, one type for each color.

Color Sensor Code and Connection for Arduino

- Figure 7-20 shows the circuit diagram for the color sensor and Arduino. Wire it up as shown, and then run the sketch shown in Example 7-8.

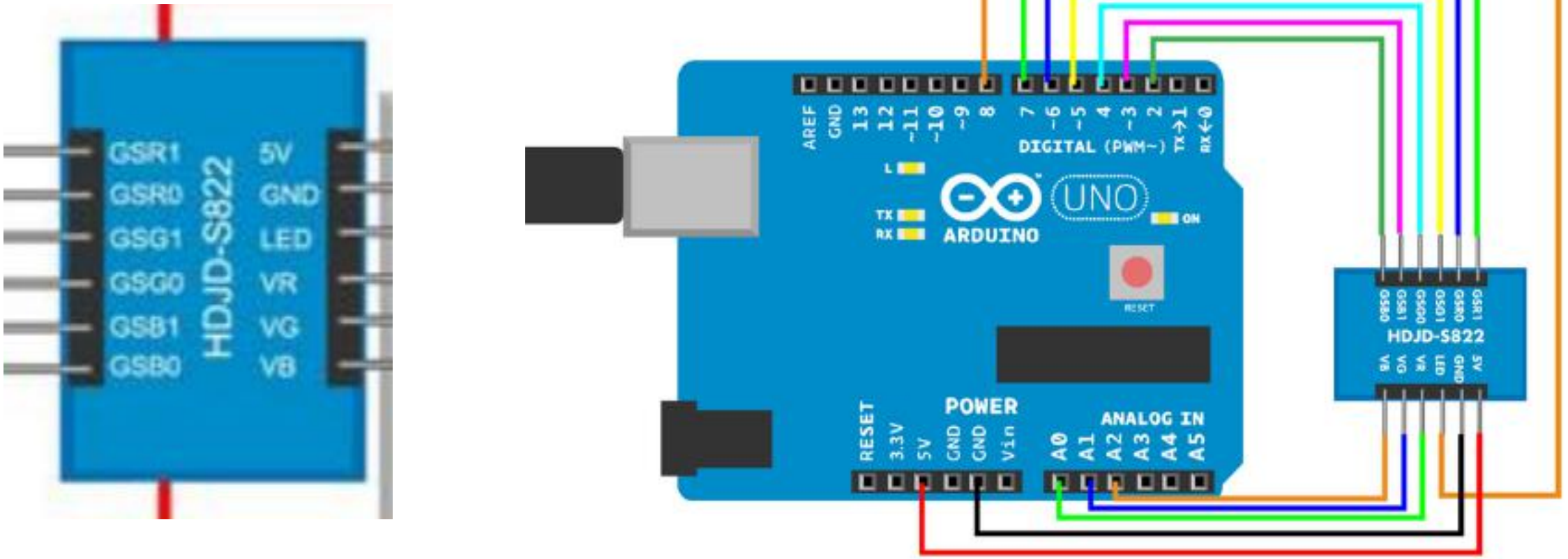
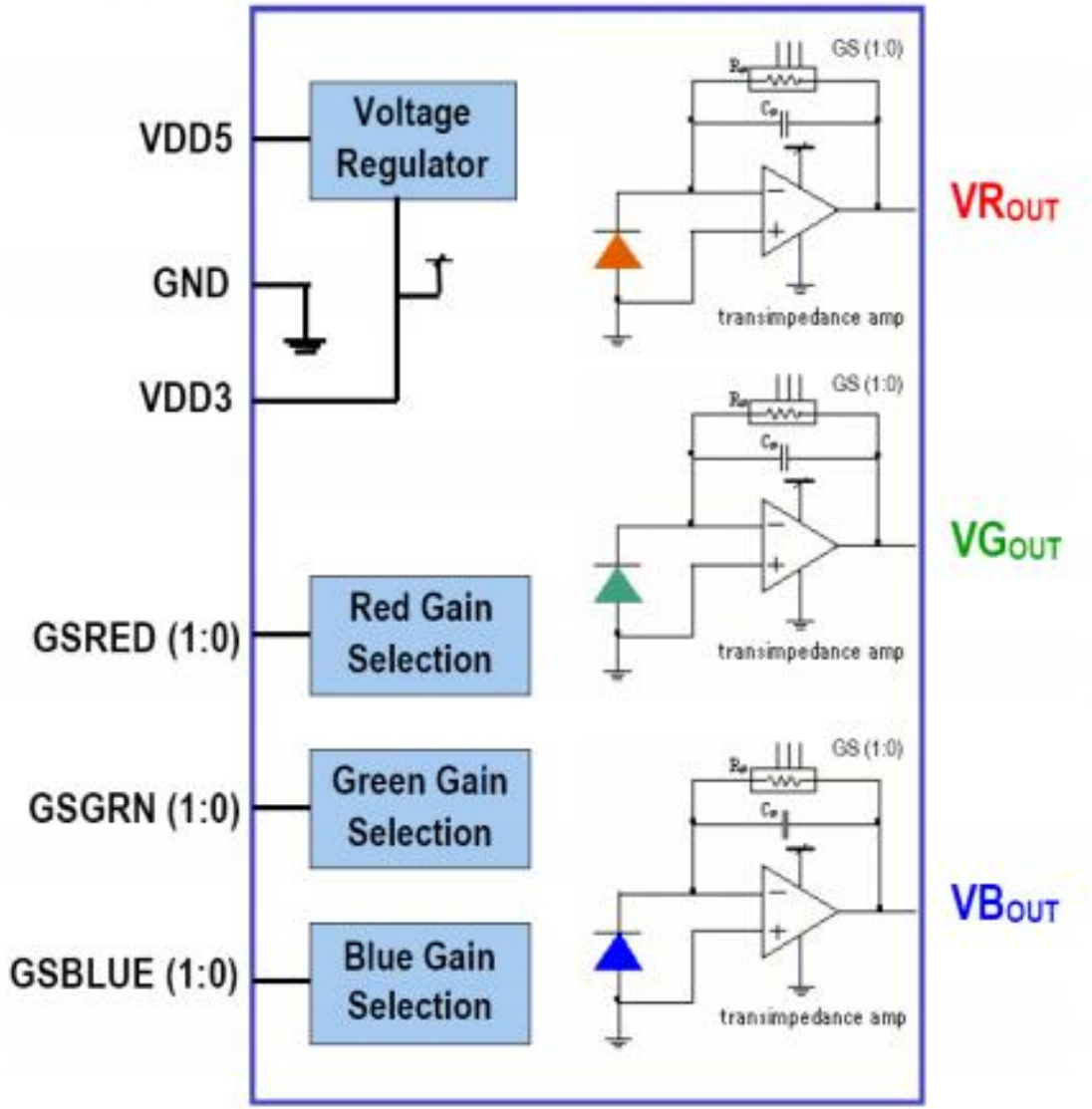
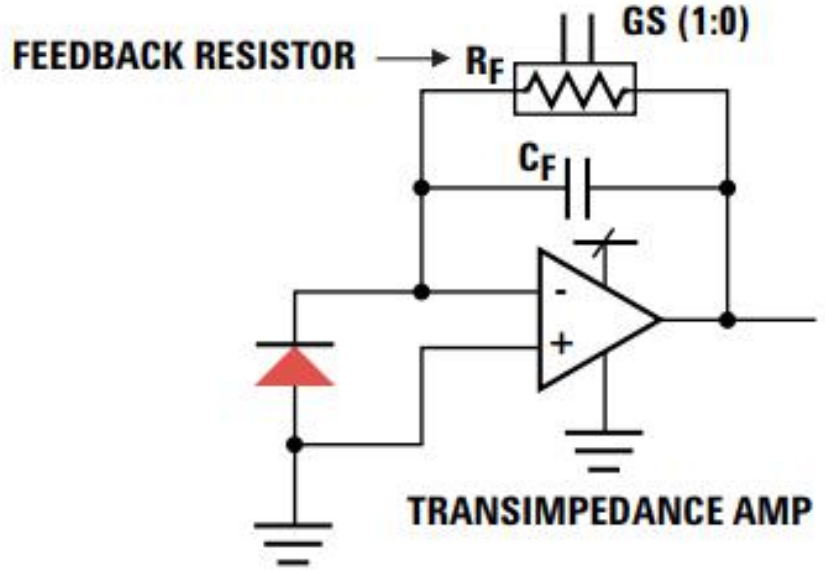


Figure 7-20. Color sensor circuit for Arduino

Experiment: All the Colors of the ’ Bow

Sensor IC Block Diagram





Gain Selection Feedback Resistor Table

GSRED1	GSRED0	GSGRN1	GSGRN0	GSBLUE1	GSBLUE0	Feedback Resistor
0	0	0	0	0	0	3.00 MΩ
0	1	0	1	0	1	0.75 MΩ
1	0	1	0	1	0	0.40 MΩ
1	1	1	1	1	1	1.50 MΩ

Example 7-8. color_sensor.ino

```
// color_sensor.ino - sense color with HDJD-S822-QR999 and print RGB value
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int gsr1Pin = 7;     // ❶ 指定增益模式引脚。如果需要校准颜色，可以添加增益量，以此增加检测颜色灵敏度
const int gsr0Pin = 6;
const int gsg1Pin = 5;
const int gsg0Pin = 4;
const int gsb1Pin = 3;
const int gsb0Pin = 2;

const int ledPin = 8;      // ❷ LED会照亮物体的表面，对于该传感器来说非常关键。照明使用的LED内置在传感器上。

const int redPin = A0;     // ❸ 读取红色、绿色和蓝色亮度的引脚
const int greenPin = A1;
const int bluePin = A2;

int red = -1;              // ❹ 这些变量用于存储从传感器读取到的数值。为了便于调试，初始化为不可能的数值
int green = -1;
int blue = -1;
```

Example 7-8. color_sensor.ino

```
// color_sensor.ino - sense color with HDJD-S822-QR999 and print RGB value
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int gsr1
const int gsr0
const int gsg
const int gsg0
const int gsb
const int gsb0

const int ledP

const int redP
const int gree
const int blue

int red = -1;
int green = -1
int blue = -1;

void setup() {
        Serial.begin(115200);
        pinMode(gsr1Pin, OUTPUT);
        pinMode(gsr0Pin, OUTPUT);
        pinMode(gsg1Pin, OUTPUT);
        pinMode(gsg0Pin, OUTPUT);
        pinMode(gsb1Pin, OUTPUT);
        pinMode(gsb0Pin, OUTPUT);
        pinMode(ledPin, OUTPUT);
        digitalWrite(ledPin, HIGH);    // ❺ 打开传感器内置LED照亮物体的表面
        digitalWrite(gsr1Pin, LOW);  // ❻ 关闭所有颜色增益 （设置为00)
        digitalWrite(gsr0Pin, LOW);
        digitalWrite(gsg1Pin, LOW);
        digitalWrite(gsg0Pin, LOW);
        digitalWrite(gsb1Pin, LOW);
        digitalWrite(gsb0Pin, LOW);
}
```

Example 7-8. color_sensor.ino

```
// color_sensor.ino - sense color with HDJD-S822-QR999 and print RGB value
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int gsr1
const int gsr0
const int gsg
const int gsg0
const int gsb
const int gsb0

const int ledP

const int redP
const int gree
const int blue

int red = -1;
int green = -1
int blue = -1;

void setup() {
    Serial.begin(
    pinMode(gsr
    pinMode(gsr
    pinMode(gsg
    pinMode(gsg
    pinMode(gsb
    pinMode(gsb
    pinMode(led
    digitalWrite(l
    digitalWrite(g
    digitalWrite(g
    digitalWrite(g
    digitalWrite(g
    digitalWrite(g
    digitalWrite(g
}

void loop() {
    int redValue = analogRead(redPin);          // ❼
    int greenValue = analogRead(greenPin);
    int blueValue = analogRead(bluePin);
    redValue = redValue * 10 / 1.0;             // ❽
    greenValue = greenValue * 10 / 0.75;
    blueValue = blueValue * 10 / 0.55;
    Serial.print(redValue);
    Serial.print(" ");                          // ❾
    Serial.print(greenValue);
    Serial.print(" ");
    Serial.print(blueValue);
    Serial.println(" ");
    delay(100);
}
```

Color Sensor Code and Connection for Raspberry Pi

- Figure 7-21 shows the circuit for Raspberry Pi. Hook it up as shown, and then run the code shown in Example 7-9.

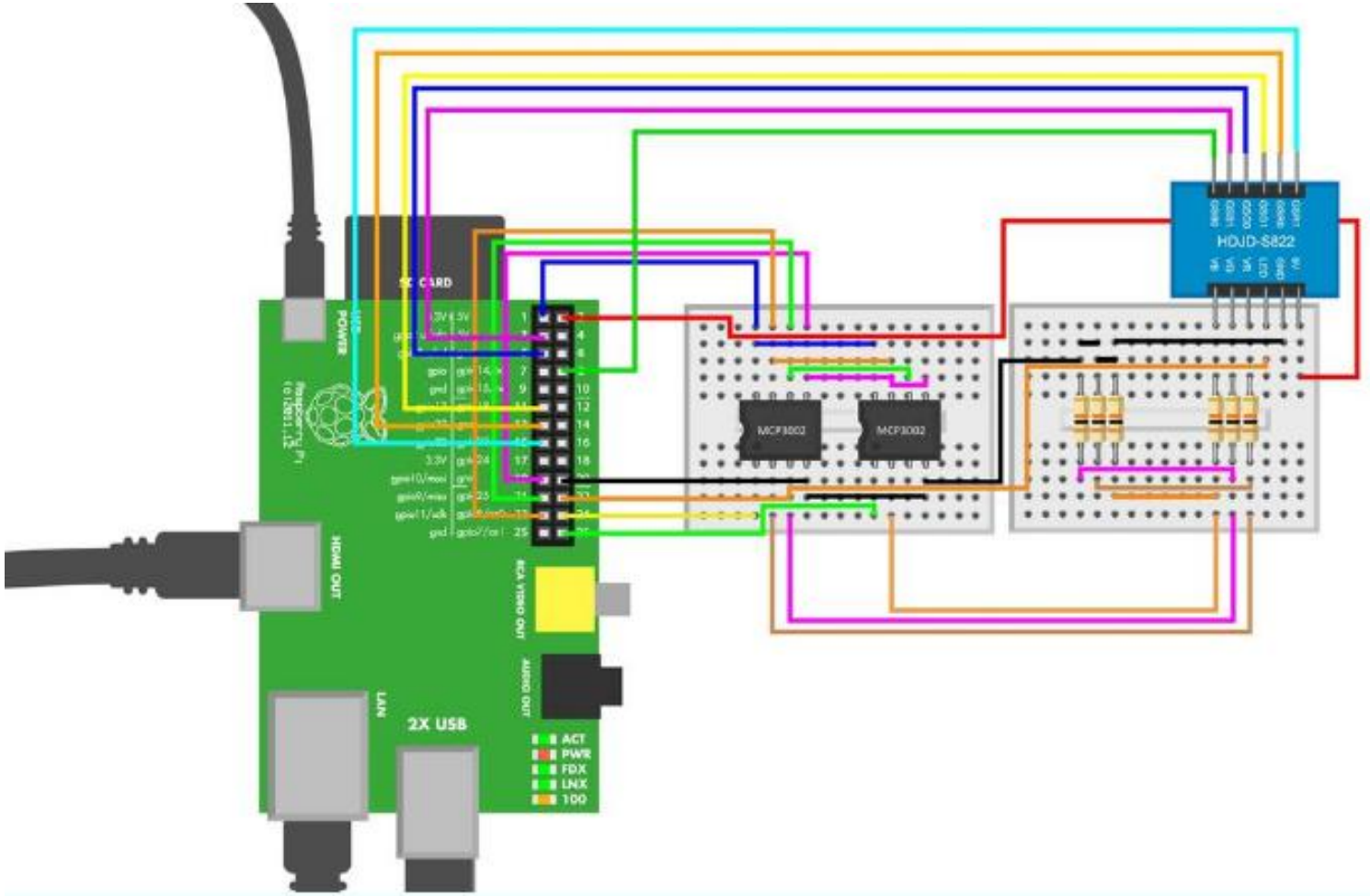


Figure 7-21. Color sensor circuit for Raspberry Pi makes your head spin

A Tangled Mess

- Does the Raspberry Pi circuit for color sensor make you want to scream?
- The number of wires makes off-by-one mistakes and loose wires more likely.
- There are two basic solutions to this problem:
  - simplify the circuit here and there
  - combine Arduino and Raspberry Pi.
- Our favorite solution is combining the two platforms.

A Tangled Mess

- You can read the sensor with Arduino, and then send the RGB values to Raspberry Pi using USB-Serial.
- With Arduino's built-in analogRead(), the circuit is simple (Figure 7-20).
- Then use USB to connect Arduino to Raspberry Pi.
- In Raspberry Pi, you can read serial-over-USB using pySerial.

A Tangled Mess

- A third method that could slightly reduce wiring is to use an ADC (analog-to-digital converter) with more channels, such as MCP3008.
- To use such a chip, you would have to modify the botbook_mcp3002 library.
- If you never plan to use gain pins, you can try connecting them to ground, thus keeping them LOW and disabling all gains. This could also simplify the connections.

Example 7-9. color_sensor.py

```
# color_sensor.py - sense color and print RGB value to serial
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp          # 库文件与本程序位于同一目录
import botbook_gpio as gpio

def initializeColorSensor():
    ledPin = 25
    gpio.mode(2,"out")
    gpio.mode(3,"out")
    gpio.mode(14,"out")
    gpio.mode(17,"out")
    gpio.mode(22,"out")
    gpio.mode(27,"out")
    gpio.write(2,gpio.LOW)             # 设置增益引脚（GS*）为低电平，关闭增益功能
    gpio.write(3,gpio.LOW)
    gpio.write(14,gpio.LOW)
    gpio.write(17,gpio.LOW)
    gpio.write(22,gpio.LOW)
    gpio.write(27,gpio.LOW)
    gpio.mode(ledPin,"out")
    gpio.write(ledPin, gpio.HIGH)      # 点亮传感器内置LED，便于传感器看清物体表面的颜色
```

Example 7-9. color_sensor.py

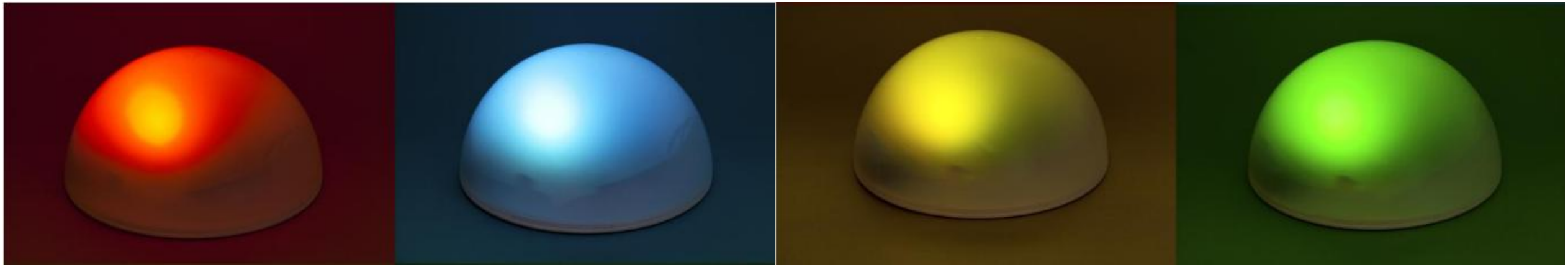
```
# color_sensor.py - sense color and print RGB value to serial
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import ti
import b
import b

def initia
    ledPi
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
    gpio.
```

```
def main():
    initializeColorSensor()
    while True:             #
        redValue = mcp.readAnalog(0, 0)
        greenValue = mcp.readAnalog(0, 1)
        blueValue = mcp.readAnalog(1, 0)    #  读取不同MCP3002芯片的不同通道
        redValue = redValue * 10 / 1.0;         #  根据手册，均衡颜色值，均衡之后三色度量相同
        greenValue = greenValue * 10 / 0.75;
        blueValue = blueValue * 10 / 0.55;
        print("R: %d, G: %d, B: %d" % (redValue,greenValue,blueValue))  # 输出颜色值
        time.sleep(0.1) # s


if __name__ == "__main__":
    main()
```

Contents

Test Project: Chameleon Dome

- Our final project for this chapter is a dome that changes color to match the surface it's sitting on.
- We'll use the color sensor code and display the color it senses with an RGB LED.
- When the whole thing is built in a solid package, the result is very impressive.



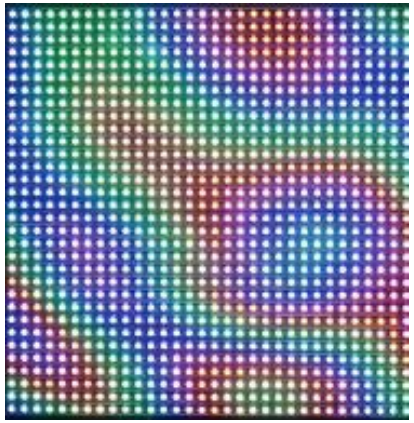
Test Project: Chameleon Dome

- **What You'll Learn，In the Chameleon Dome project, you'll learn how to:**
  - Build a device that changes color like a chameleon.
  - Show any color with an RGB LED.
  - Use a moving average to filter out random noise.
  - Use an easing function to map input values to output values.

RGB LED

- An RGB LED (Figure 7-22) packages three LEDs into one package.
- It looks just like a single LED.
- By mixing red, green, and blue, you can show any color.



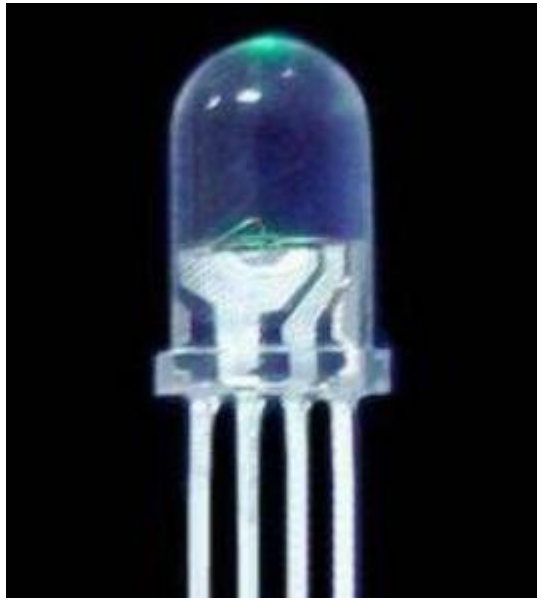


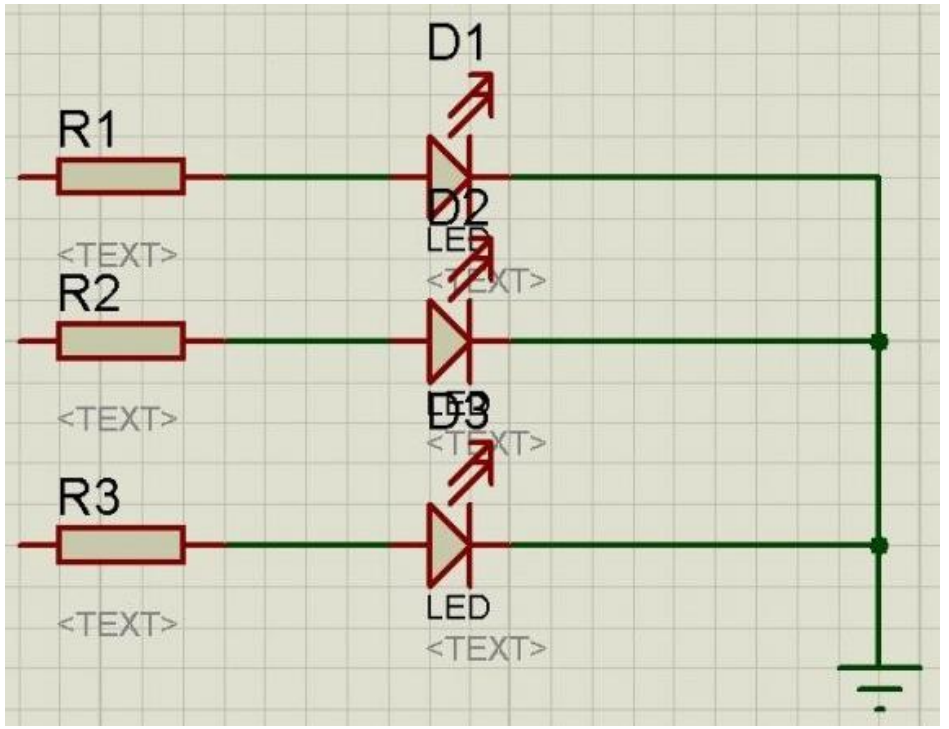

RGB LED

- The human eye has receptor cells for three colors: red, green, and blue. That's why those colors are used in televisions and other displays.
- Human perception has a strange feature (or a bug) that it sees combinations of frequencies as another frequency.
- This means that you can mix red light with green to get yellow.

RGB LED

- An RGB LED typically has four leads: one lead for each color (red, green, blue) and one common lead.





RGB LED

- Contrary to what you might expect, in many RGB LEDs, the common lead is often positive.
- A common positive lead is also called common anode.
- Because the common lead is positive, you'll have to take each color lead (red, green, blue) LOW (0 V or GND) to make each color shine.

RGB LED

- If your RGB LED is common cathode (common lead negative), you'll need to change the circuit and your code.
- For the circuit, you'll need to connect the common lead to ground instead of +5 V.
- For the code, instead of reversing the light intensity value with 255-color (for example 255-red) with analogWrite(), you must remove the 255- part (for example analogWrite(redPin, red)).

Finding the Leads of an RGB LED

- You can find the leads experimentally.
- Turn on Arduino by connecting it to USB. As you'll just use +5 V and GND, it doesn't matter which code is running on the Arduino.
- On Arduino, connect a red wire to +5 V and a black wire to GND.

Finding the Leads of an RGB LED

- Connect the two wires to any adjacent leads on the RGB LED.
- Keep trying adjacent leads around the LED until it lights up.
- Try a couple of other leads until you have lit the red, green, and blue elements of the RGB LED separately. Note that in order to do this, one lead (for a common anode LED, this would be the positive lead) had to stay connected to the red +5 V wire, while you had to connect each of the other three leads in turn to the black GND wire. Mark the common anode (positive) lead.

Finding the Leads of an RGB LED

- If you notice that the common lead is longer than the others, remember this fact so you can find it later. Otherwise, mark the common lead with a small piece of tape and write something on it (A for anode, + for positive, whatever helps you remember).
- If instead, you kept a common lead connected to the black GND wire and had to connect each of the other three leads in turn to the red +5 V wire, then you have a common cathode LED. If that's the case, see What About Common Cathode?
- Mark each of the remaining leads with their corresponding color (R, G, or B)

Example 7-10. hellorgb.ino

```
// hellorgb.ino - mix colors with RGB LED
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int redPin=11;    // ❶
const int greenPin=10;
const int bluePin=9;

void setup(){
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
}

void setColor(int red, int green, int blue){
        analogWrite(redPin, 255-red);          // ❷
        analogWrite(greenPin, 255-green);
        analogWrite(bluePin, 255-blue);
}

void loop(){
        setColor(255, 0, 0);                   // ❸
        delay(1000);
        setColor(255, 255, 255);
        delay(1000);
}
```

Finding the Leads of an RGB LED

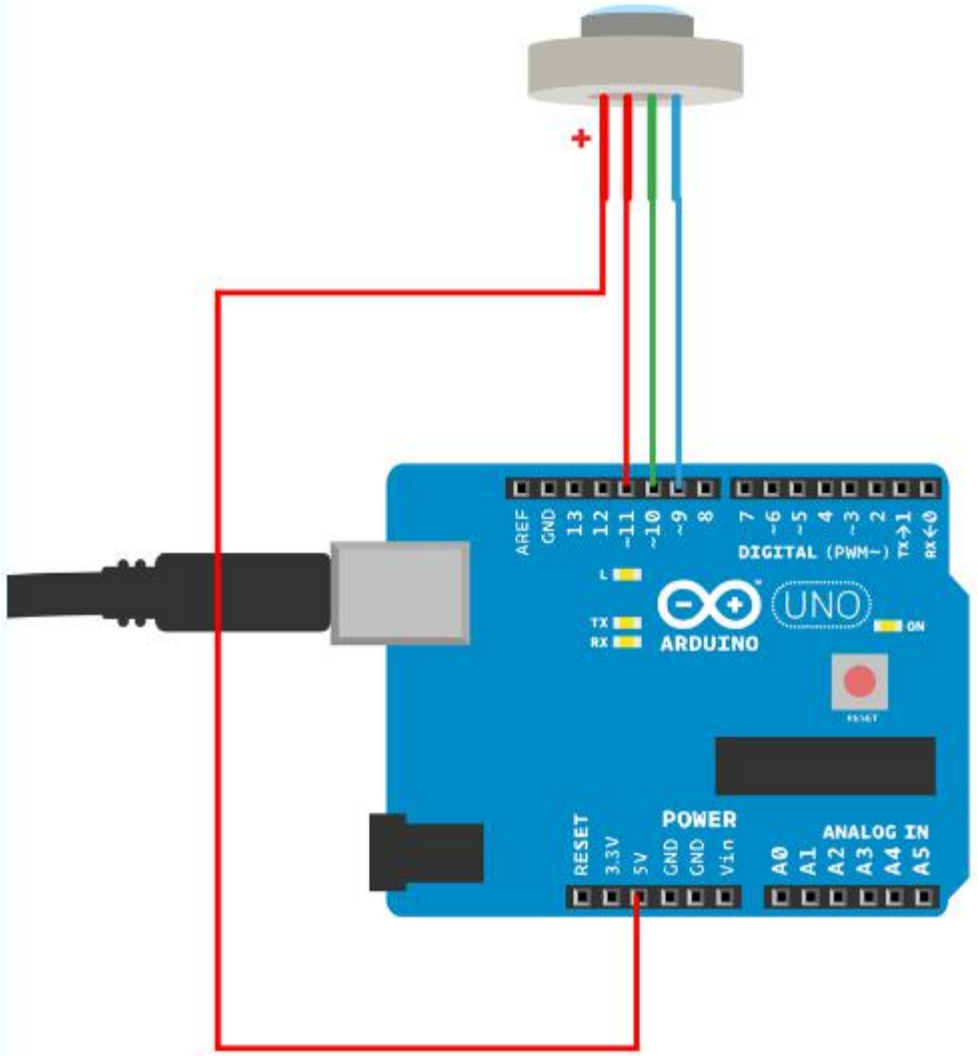
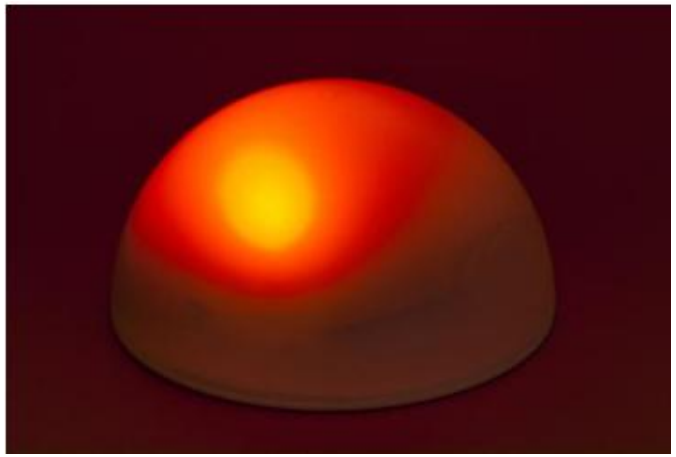


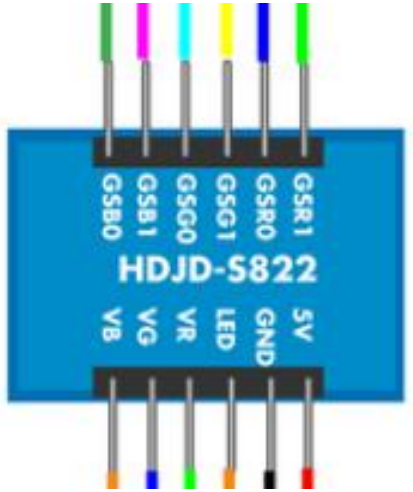
Figure 7-23. RGB LED connected to Arduino

Figure 7-24. Color-changing robot prototype made by students during two-day robot workshop

Combining Codes

- The Chameleon Dome combines an RGB LED with the color sensor you saw earlier in Color Sensor Code and Connection for Arduino.





Combining Codes

- Connect the color sensor and RGB LED to Arduino as shown in Figure 7-27.

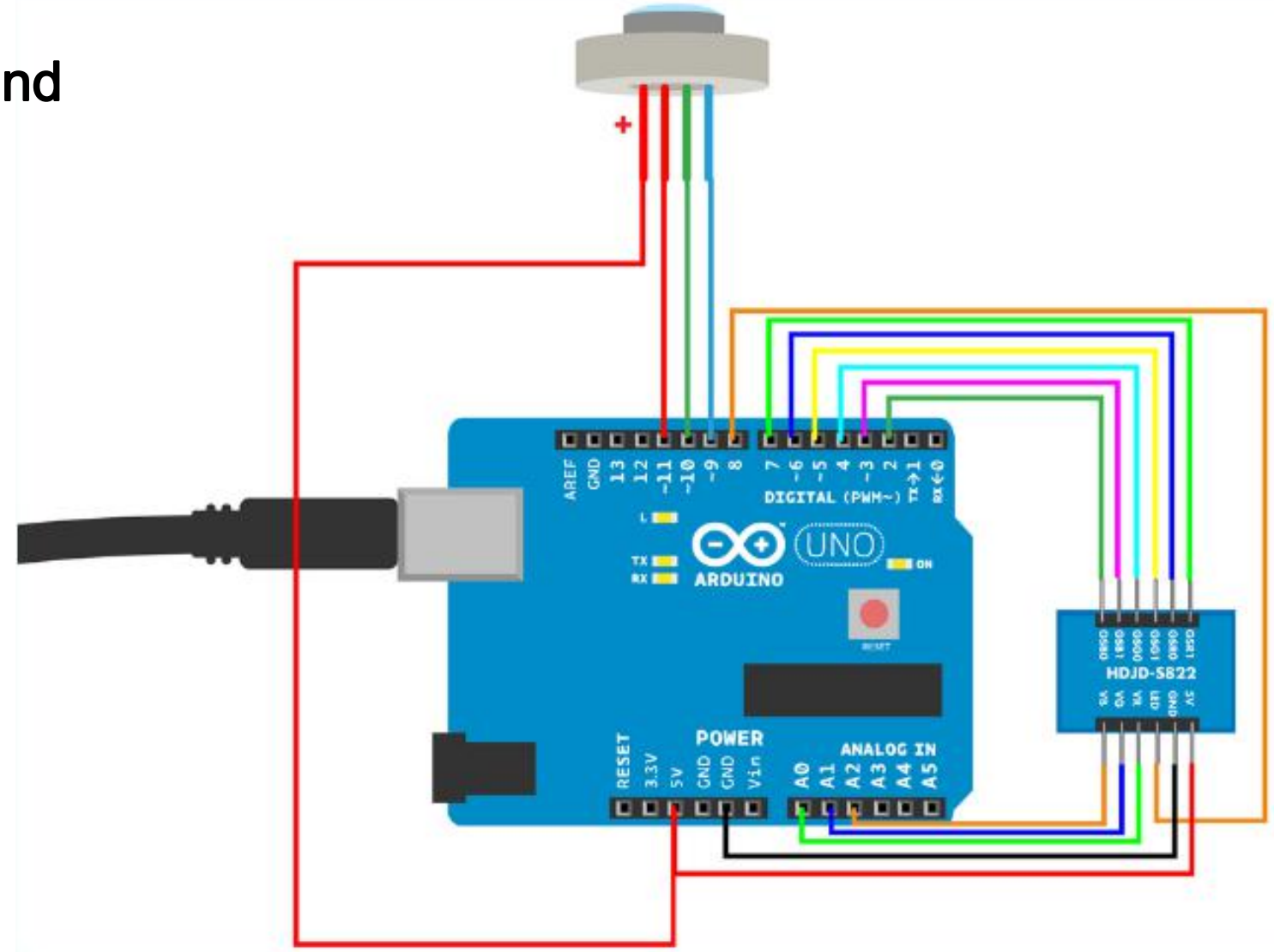
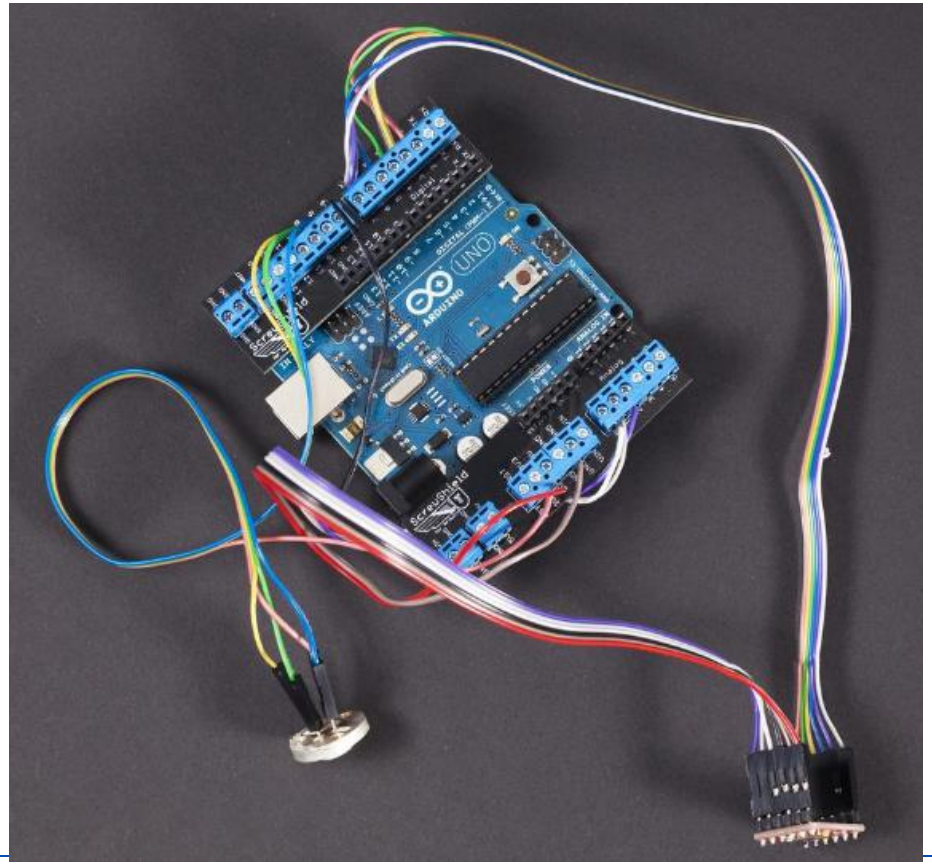


Figure 7-27. Chameleon Dome connections

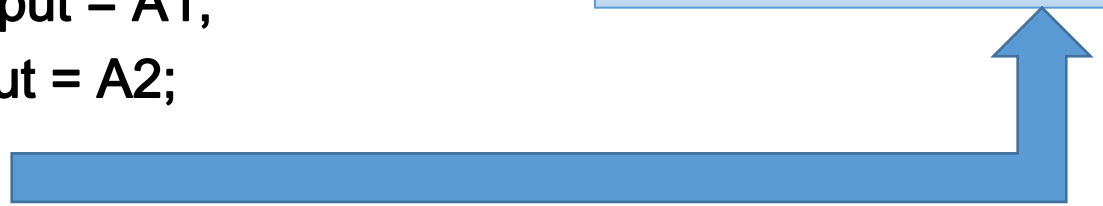
Combining Codes

- DuPont connector cables (see Figure 7-25) combined with a ScrewShield are a life-saver when you need to use a lot of pins (see Figure 7-26).



Example 7-11. chameleon_cube.ino

```
// chameleon_dome.ino - cube changes color to match the surface
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int gsr1Pin = 7;     // ❶
const int gsr0Pin = 6;
const int gsg1Pin = 5;
const int gsg0Pin = 4;
const int gsb1Pin = 3;
const int gsb0Pin = 2;

const int ledPin = 8;      // ❷

const int redInput = A0;  // ❸
const int greenInput = A1;
const int blueInput = A2;

const int redOutput = 11;          // ❹
const int greenOutput = 10;
const int blueOutput = 9;

int red = -1;                      // ❺
int green = -1;
int blue = -1;

const float newWeight = 0.7;       // ❻
```



Moving Average

- The Chameleon Dome looks nice when it smoothly changes values.
- It would look ugly if it erratically jumped from one color to another.

- But that's one of the perils of working with sensors: sometimes, for a blink of an eye, a red will look green for the sensor.
- Random noise is a common problem in any measurement, and sensors are no exception.

Moving Average

- Consider how I might measure the height of a tiny tree. I made multiple measurements to reduce the chance of error.
- I got these values:
  - 102 cm, 100 cm, 180 cm, 103 cm, 105 cm
- Most of us would likely drop the 180 cm measurement as a typo.
- Luckily, a computer can do that for you.

Moving Average

- Initially, you might think of storing the values in an array, and then calculating the average for the array on every round. Even though this would work, it would require a lot of code for such a small thing: an array, a pointer, an average, and some calculation on each iteration.
- Moving average to the rescue!
- You can calculate the average of current and previous value to get some smoothing. To get more data points without using a table, you can use a weighted moving average.

Moving Average

- You could give 70% weight to new input, leaving 30% (100% minus 70%) for old values.
  - input = 0.7*input + 0.3*old
  - old = input
- Because the old value is affected by the older data points, you don't need to use an array at all.

Example 7-11. chameleon_cube.ino

```
// chameleon_dome.ino - cube changes color to match the surface
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int gsr1Pin = 7;
const int gsr0Pin = 6;
const int gsg1Pin = 5;
const int gsg0Pin = 4;
const int gsb1Pin = 3;
const int gsb0Pin = 2;
const int ledPin = 8;
const int redInput = A0;
const int greenInput = A1;
const int blueInput = A2;
const int redOutput = 11;
const int greenOutput = 10;
const int blueOutput = 9;
int red = -1;
int green = -1;
int blue = -1;
const float newWeight = 0.7;

void setup() {
        Serial.begin(115200);
        pinMode(gsr1Pin, OUTPUT);
        pinMode(gsr0Pin, OUTPUT);
        pinMode(gsg1Pin, OUTPUT);
        pinMode(gsg0Pin, OUTPUT);
        pinMode(gsb1Pin, OUTPUT);
        pinMode(gsb0Pin, OUTPUT);
        pinMode(ledPin, OUTPUT);        //颜色传感器板载照明LED
```

Example 7-11. chameleon_cube.ino

```
// chameleon_dome.ino - cube changes color to match the surface
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int gsr1Pin = 7;
const int gsr0Pin = 6;
const int gsg1Pin = 5;
const int gsg0Pin = 4;
const int gsb1Pin = 3;
const int gsb0Pin = 2;
const int ledPin = 8;
const int redInput = A0;
const int greenInput = A1;
const int blueInput = A2;
const int redOutput = 11;
const int greenOutput = 10;
const int blueOutput = 9;
int red = -1;
int green = -1;
int blue = -1;
const float newWeight = 0.7;
```

```
    pinMode(redOutput, OUTPUT);
    pinMode(greenOutput, OUTPUT);
    pinMode(blueOutput, OUTPUT);
    digitalWrite(ledPin, HIGH);   //❼ 在测量之前照亮物体的表面
    digitalWrite(gsr1Pin, LOW);
    digitalWrite(gsr0Pin, LOW);
    digitalWrite(gsg1Pin, LOW);
    digitalWrite(gsg0Pin, LOW);
    digitalWrite(gsb1Pin, LOW);
    digitalWrite(gsb0Pin, LOW);
}
```

Example 7-11. chameleon_cube.ino

```
// chameleon_dome.ino - cube changes color to match the surface
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void loop() {
    int redValue = analogRead(redInput);              // ❽
    int greenValue = analogRead(greenInput);
    int blueValue = analogRead(blueInput);
    redValue = redValue * 10 / 1.0;
    greenValue = greenValue * 10 / 0.75;
    blueValue = blueValue * 10 / 0.55;
    redValue = map(redValue, 0, 1023, 0, 255);      // ❾ 将输入非线性映射到输出
    greenValue = map(greenValue, 0, 1023, 0, 255);
    blueValue = map(blueValue, 0, 1023, 0, 255);
```

EASING INPUT TO OUTPUT

- Inputs and outputs can have different kinds of values. Your code must convert between these values.
- In the simplest case, output is just the input multiplied by a number. In that case, conversion is just a matter of multiplication.
- For example, Arduino's analogRead() has range of 0 to 1023, but analogWrite() has a range of 0 to 255. To convert between the ranges, You can use the linear mapping method:

in : out = 1023 : 255 ----> $\mathbf{out} = \frac{\mathbf{in}}{\mathbf{1023}} \times \mathbf{255}$

EASING INPUT TO OUTPUT

- Inputs and outputs can have different kinds of values. Your code must convert between these values.
- In the simplest case, output is just the input multiplied by a number. In that case, conversion is just a matter of multiplication.
- For example, Arduino's analogRead() has range of 0 to 1023, but analogWrite() has a range of 0 to 255. To convert between the ranges, You can use the linear mapping method:

in : out = 1023 : 255 ----> $\mathbf{out} = \boxed{\frac{\mathbf{in}}{\mathbf{1023}}} \times \mathbf{255}$

P

EASING INPUT TO OUTPUT

- Arduino’ s library even has a convenience function for this:
  - out = map(in, 0, 1023, 0, 255)
- You can see some examples of linear conversion in Table 7-2.

Table 7-2. Linearly mapping input to output, using map()

analogRead( )的输入值 范围 [0-1023]	输入值占最大输入值的 百分比 % -- P	analogWrite( )的输出值 范围 [0-255]
0	0	0.0
234	23	58
511	50	127
1023	100	255

EASING INPUT TO OUTPUT

- However, some outputs don‘t work well with an output that increases linearly, so you need to use easing.
- The RGB LED in this project is a good example of an output that works better with easing.

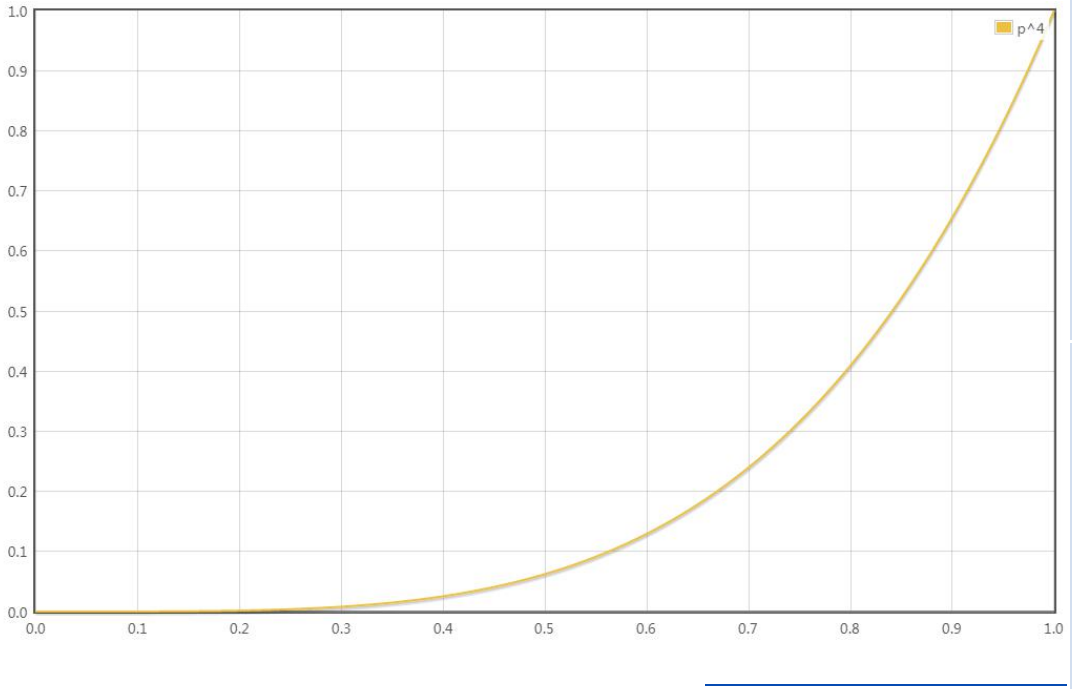
- Most RGB LED color mixing needs to happen with low values. Near the upper range of output values, everything becomes white, and individual colors are hard to discern.

EASING INPUT TO OUTPUT

- An easing function maps inputs to outputs in non-linear fashion. For an RGB LED, an exponential function is good. Otherwise, most values would just result in bright white light instead of colors like orange or violet.
- First, calculate the percentage for the input:
  - $p = in/1023$
- Then create the output non-linearly.
  - $out = 255 * p^4$

EASING INPUT TO OUTPUT

- Because percentage p has a maximum of 1.0 (100%), the exponent p4 values are always between 0.0 (0%) and 1.0 (100%). The exponent function creates the classic hockey-stick figure. You can see sample values mapped in Table 7-3.



	p	p^4	analogWrite( )
输入值前一半	0.0%	0	最小
	20%	0.2%	0
	40%	2.6%	6
	50%	6.2%	15
输入值后一半	60%	13.0%	33
	80%	41.0%	104
	90%	65.6%	167
	100%	100%	255

Example 7-11. chameleon_cube.ino

```
// chameleon_dome.ino - cube changes color to match the surface
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void loop() {
//继续继续前一页
      if(redValue > 255)        redValue = 255;          // ❿
      if(greenValue > 255)      greenValue = 255;
      if(blueValue > 255)       blueValue = 255;
      red = runningAverage(redValue, red);               // ⓫
      green = runningAverage(greenValue, green);
      blue = runningAverage(blueValue, blue);
```

Example 7-11. chameleon_cube.ino

```
// chameleon_dome.ino - cube changes color to match the surface
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void loop() {
//继续继续前一页
        Serial.print(red);     Serial.print(" ");
        Serial.print(green); Serial.print(" ");
        Serial.print(blue);    Serial.println(" ");
        if(red < 200 || green < 180 || blue < 180) {
                green = green - red * 0.3;      // ⑫ 通过实验得到的计算公式，可以让颜色更美一些
                blue = blue - red * 0.3;
        }
```

Example 7-11. chameleon_cube.ino

```
// chameleon_dome.ino - cube changes color to match the surface
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void loop() {
//继续继续前一页

      red = easing(red);          // ⑬让颜色的变化速度慢一些，而不是直接从一个颜色跳到另一个颜色
      green = easing(green);
      blue = easing(blue);
      setColor(red,green,blue);     // ⑭ 将RGB LED设置为刚计算得出颜色值
      delay(100);
}
```

Example 7-11. chameleon_cube.ino

```
// chameleon_dome.ino - cube changes color to match the surface
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int runningAverage(int input, int old) {                    //使用移动平均法平滑处理输入的随机噪声
    return newWeight*input + (1-newWeight)*old;
}

int easing(int input) {                                     //缓动技术，让输出（RGB LED）变得平缓和平滑
    float percent = input / 255.0f;
    return 255.0f * percent * percent * percent * percent;
}

int setColor(int r, int g, int b) {                         // ⑰
    analogWrite(redOutput, 255-r);                          // ⑱
    analogWrite(greenOutput, 255-g);
    analogWrite(blueOutput, 255-b);
}
```

RGB LED Shows Any Color

- **To get a variety of colors, you′ ll need to mix red, green, and blue at varying levels.**
  - If you set all three of them to full brightness, you′ ll get something pretty close to white light.
  - If you set all three of them to minimum brightness, you′ ll get nothing.

RGB LED Shows Any Color

- Arduino isn’ t capable of truly dimming an LED, because an LED can’ t be dimmed effectively.
- If you lower the voltage, the brightness will go down, but lower the voltage enough, and it simply turns off (usually well before you get all the way down to 0 V).

RGB LED Shows Any Color

- To work around this, Arduino uses *pulse width modulation* (PWM).
- To make an LED look like it's at 10% brightness, PWM will use a *duty cycle* of 10%. This means that Arduino will keep the LED on for 10% of the time, and off 90% of the time.
- But Arduino switches the LED on and off so fast (each cycle takes 2 milliseconds or 2000 microseconds) that you don't notice the flicker.
- At a 10% duty cycle, Arduino will turn the LED on for 200 microseconds, leave it off for 1800 microseconds, then start the on/off cycle again.

RGB LED Shows Any Color

- But in your code, you don't have to worry about these details.
- You can pretend that you're just sending a range of voltage (from 0 to 255) that corresponds to different brightness levels.

- Because the common lead is HIGH and each color lead is brightest when you take the lead LOW, you'll get the brightest red color with analogWrite(redPin, 0).

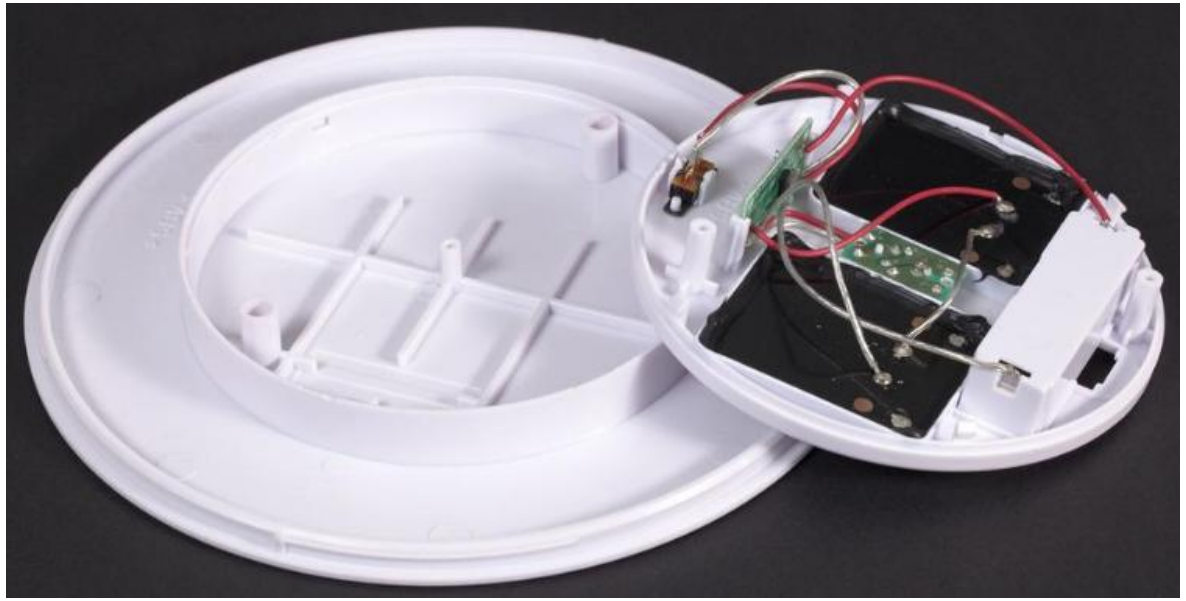
RGB LED Shows Any Color

- For any values between the minimum and the maximum, we use the value of the color (such as red, r) subtracted from 255:
  - analogWrite(redPin, 255-r)
- To get a feel for the basic colors, see Table 7-1.

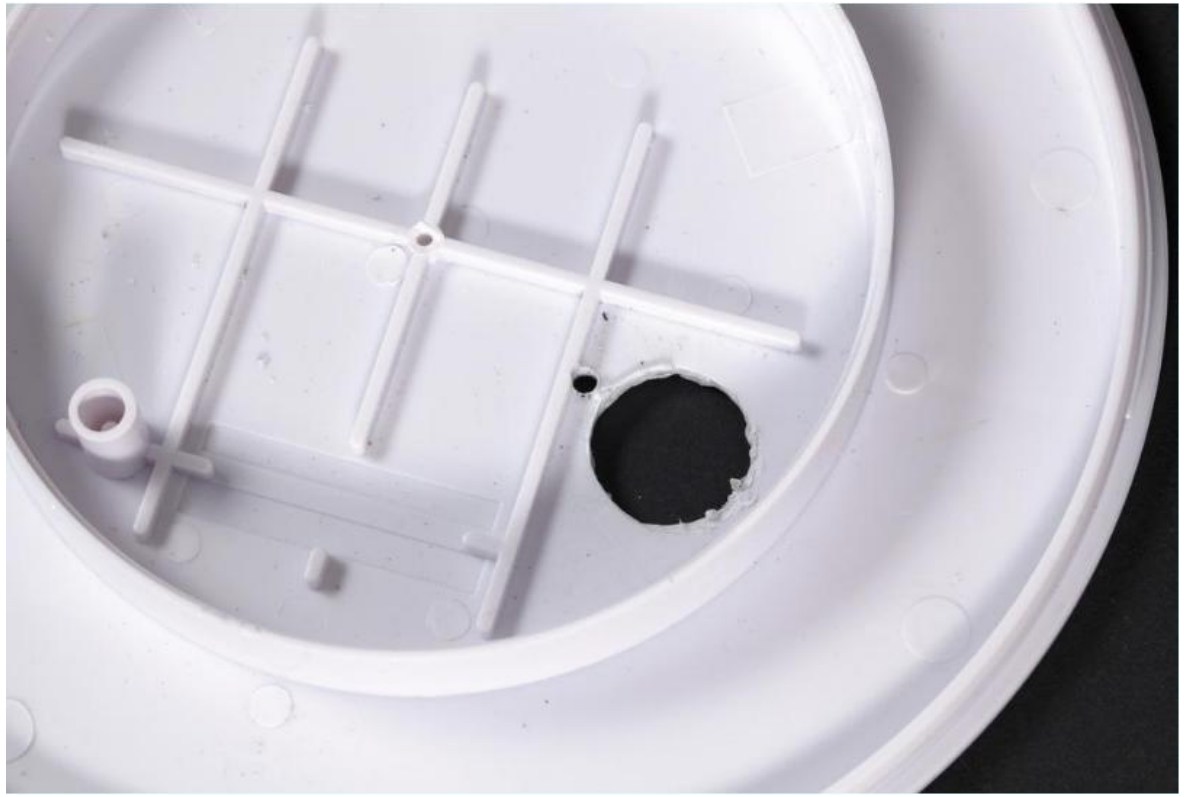
颜色	颜色的RGB值	设置颜色针脚的值	说明
黑色	（0, 0, 0）	（255, 255, 255）	关闭所有颜色
红色	（255, 0, 0）	（0, 255, 255）	
绿色	（0, 255, 0）	（255, 0, 255）	
蓝色	（0, 0, 255）	（255, 255, 0）	
白色	（255, 255, 255）	（0, 0, 0）	所有LED的亮度最大
通用公式	（r, g, b）	（255-r, 255-g, 255-b）	

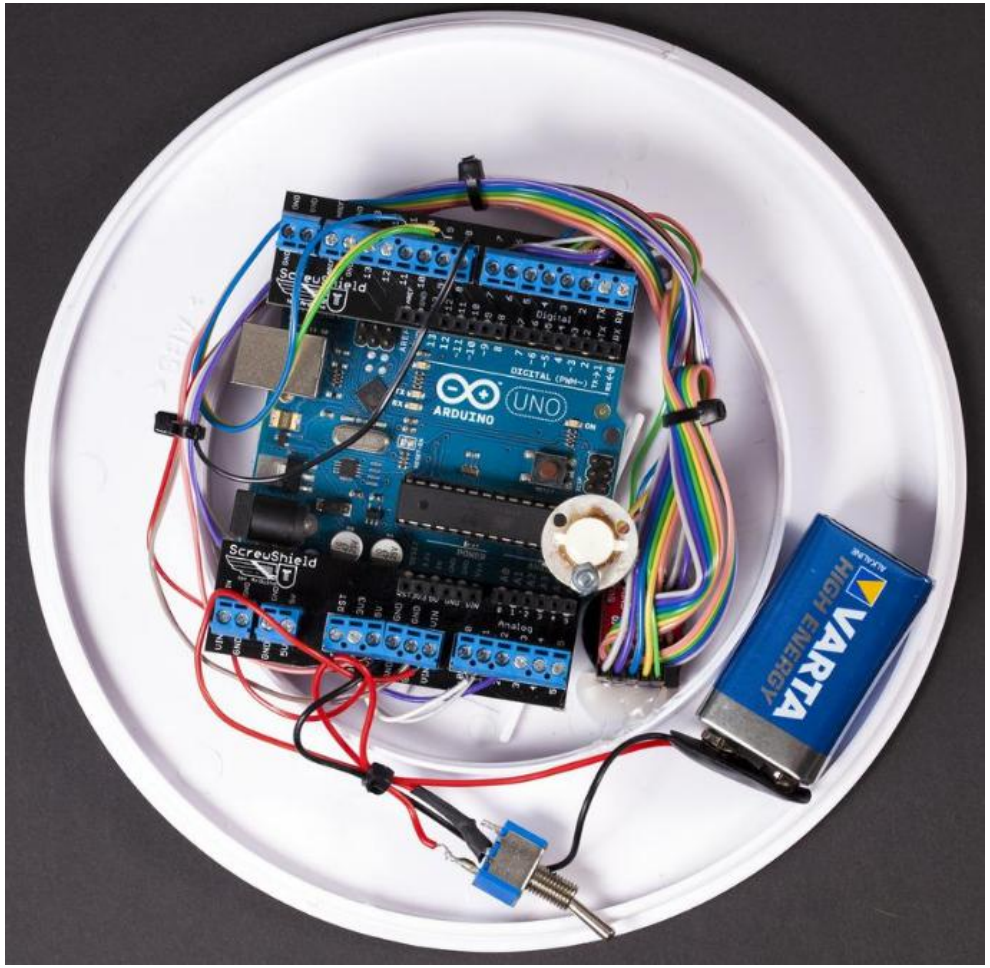
Dome Building Tips

Dome Building Tips



Dome Building Tips





Dome Building Tips

- Now just put the battery in the clip, turn the power switch on, close the dome, and enjoy your Chameleon Dome (see Figure 7-34).