Electricity and Magnetism

Electricity and Magnetism

- Can you feel the power radiate through your body? Of course not, even though electromagnetic radiation from power lines and cell phones is all around you.
- Electricity powers all your gadgets, especially the ones you build from this book.
- Hall sensors detect a magnetic field. They come in different varieties: some just detect the presence of a magnet, while others can tell you the strength of the magnetic field in teslas（单位特拉斯）.
- Voltage and current sensors work like a multimeter, measuring electricity passing through them. They can easily measure power that would otherwise break a microcontroller's analog input pin.

Electricity and Magnetism

- An electronic compass can tell you where magnetic north is. Better ones combine an accelerometer, so that they can point to north even when tilted.
- The sun's north and south poles change places about every 11 years. As of this writing, we're waiting for the flip to happen at any week now. However, you don't have to worry about the Earth's poles flipping on you. The last time that happened was 780,000 years ago.

Experiment: Voltage and Current

- In this experiment, you'll use the AttoPilot to measure voltage of a battery pack.
- Are the batteries running out, and how much power is your robot's motor draining? AttoPilot Compact DC Voltage and Current Sense measures voltage and current. It measures big voltage and current, and then reports it with analog output.
- AttoPilot can measure a lot of power. The most powerful model is rated for 50 V and 180 A. Power (P) is a function of voltage (U) and current (I):
  - P = UI = 50 V * 180 A = 9000 VA = 9000 W = 9 kW
- You are likely to also see this expressed with watts (W) as a function of voltage (V) and current (A):
  - W = VA = 50 V * 180 A = 9000 W = 9 kW
- The 50 V/ 180 A model of the AttoPilot sensor can handle 9 kilowatts. You can't. Stay safe and use only sensible voltages, like batteries and USB power.

Experiment: Voltage and Current

- AttoPilot voltage and current sense, conversion factors
- The AttoPilot has two analog outputs. One reports current; the other reports voltage. （两个模拟输出都以电压的形式报告测量的电压和电流）
- The maximum output is 3.3 V—considerably less than the maximum 50 V measured. The conversion factors for the 13.6 V / 45 A model used here are calculated from the component's maximum specs.
- Using 13.6 V / 45 A model as an example, here's the formula for current:
  - 3.3 V output / 45 A measured = 73.3 mV output / A measured
  - 3.3 V output / 13.6 V measured = 242.6 mV output / V measured

Experiment: Voltage and Current

- AttoPilot voltage and current sense, conversion factors
- But in your code, you'll switch things around:
  - 45 A measured/ 3.3 V output = 13.6363 A measured / V output
- So when you want to calculate the measured current, you can multiply the voltage you read from the current sense output by 13.6363.
- And here's the formula for voltage:
  - 13.6 V measured / 3.3 V output = 4.1212 V measured / V output
- When you want to calculate the measured voltage, you can multiply the voltage you read on the voltage sense output by 4.1212.

Experiment: Voltage and Current

型号	电压U，输出/测量	电流I，输出/测量	说明
13.6 V / 45A	242.3 mV / V	73.20 mV / V	本实验使用此型号
50V / 90 A	63.69 mV / V	36.60 mV / V	
50V / 180 A	63.69 mV / V	18.30 mV / V	9kW

ATTOPILOT CODE AND CONNECTION FOR ARDUINO

- Figure 10-1 shows the connections for Arduino. Wire it up as shown, and then run the sketch shown in Example 10-1.

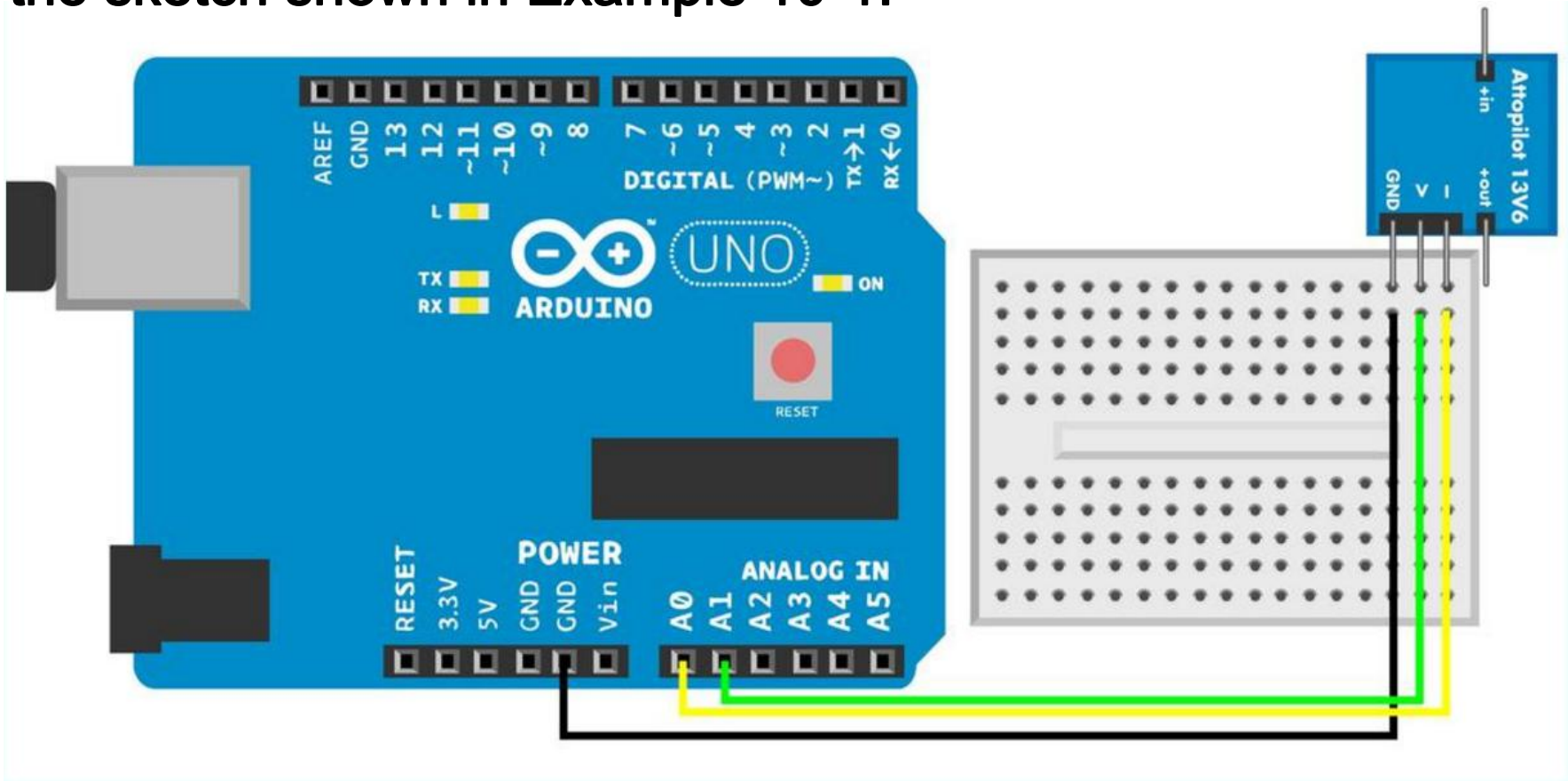


Figure 10-1. AttoPilot connections for Arduino

Example 10-1. attopilot_voltage.ino

```
// attopilot_voltage.ino - measure current and voltage with Attopilot 13.6V/45A
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int currentPin = A0;
int voltagePin = A1;

void setup(){
        Serial.begin(115200);
        pinMode(currentPin, INPUT);
        pinMode(voltagePin, INPUT);
}

float current(){
        float raw = analogRead(currentPin);
        Serial.println(raw);
        float percent = raw/1023.0;        //原始模拟数据为0~1023，计算占5V的比例
        float volts = percent*5.0;         //计算模拟输入的电压值
        loat sensedCurrent = volts * 45 / 3.3;      // 模拟电压值乘以电流的转换系数
        return sensedCurrent;              //返回电流值，单位安培 A
}
```

Example 10-1. attopilot_voltage.ino

```
// attopilot_voltage.ino - measure current and voltage with Attopilot 13.6V/45A
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
float voltage(){
        float raw = analogRead(voltagePin);
        float percent = raw/1023.0;
        float volts = percent*5.0;
        float sensedVolts = volts * 13.6 / 3.3;          // 模拟电压值乘以电压的转换系数
        return sensedVolts;                              // 返回电压值，单位伏特V
}

void loop(){
        Serial.print("Current: ");
        Serial.print(current(),4);
        Serial.println(" A");
        Serial.print("Voltage: ");
        Serial.print(voltage());
        Serial.println(" V");
        delay(200); // ms
}
```

AttoPilot Code and Connection for Raspberry Pi

- Figure 10-2 shows the connections
- Hook everything up as shown, and

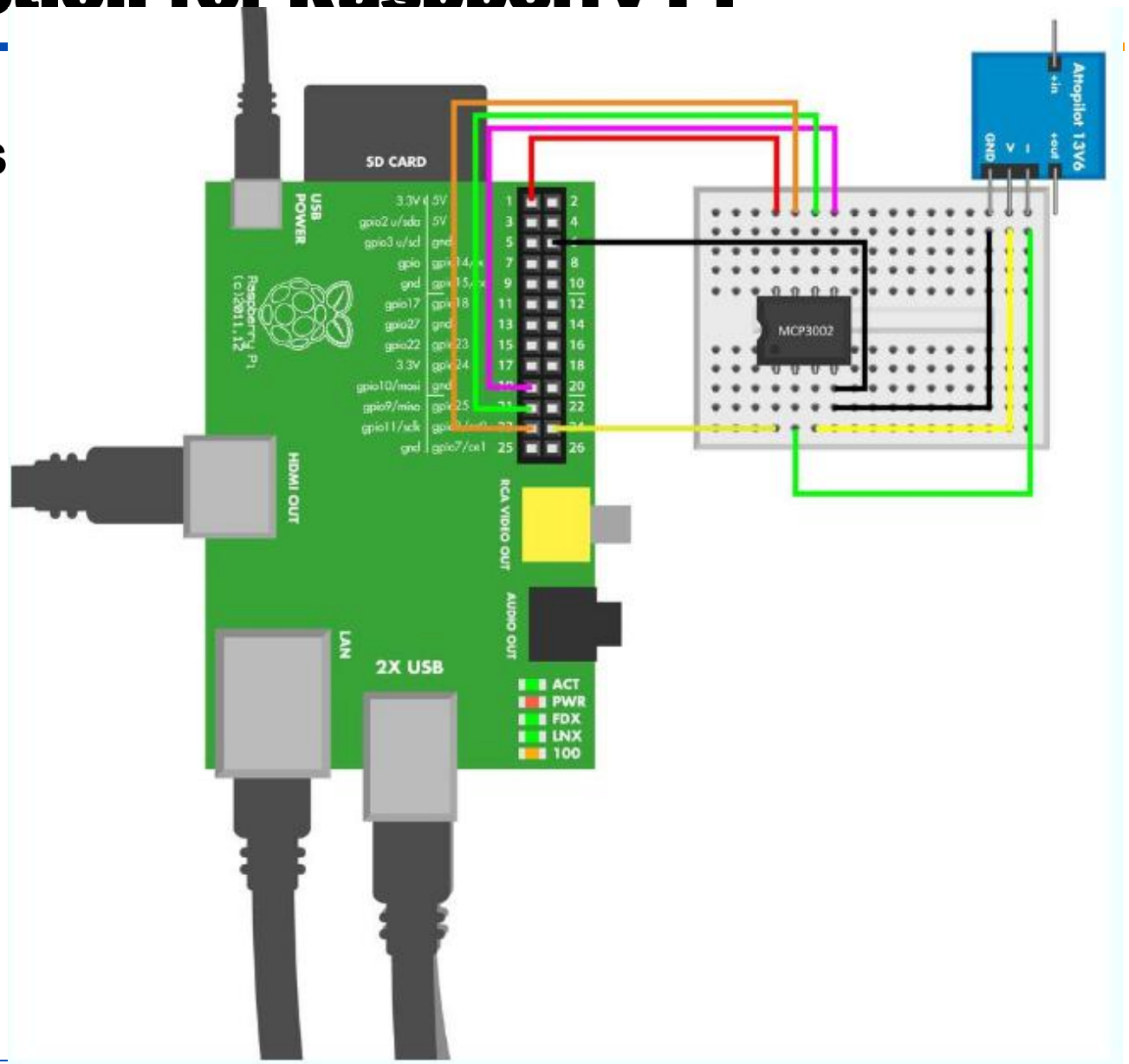


Figure 10-2. Raspberry Pi connections for the AttoPilot

Example 10-2. attopilot_voltage.py

```
# attopilot_voltage.py - measure current and voltage with Attopilot 13.6V/45A
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp   #
def readVoltage():
        raw = mcp.readAnalog(0,1)       #
        percent = raw / 1023.0  #
        volts = percent * 3.3   #
        sensedVolts = volts * 13.6 / 3.3        # V/V   #
        return sensedVolts      # V

def readCurrent():
        raw = mcp.readAnalog(0,0)
        percent = raw / 1023.0
        volts = percent * 3.3        s
        ensedCurrent = volts * 45.0 / 3.3      # A/V #
        return sensedCurrent    # A
```

Example 10-2. attopilot_voltage.py

```
# attopilot_voltage.py - measure current and voltage with Attopilot 13.6V/45A
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
def main():
        while True:
                voltage = readVoltage()
                current = readCurrent()
                print("Current %.2f A" % current)
                print("Voltage %.2f V" % voltage)
                time.sleep(0.5) # s

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

Experiment: Is It Magnetic?

- A Hall effect sensor measures a magnetic field. Hall effect sensors are used in bike speedometers, where a magnet on the wheel helps the sensor count revolutions.
- A magnetic field causes electrons to divert from their straight path, causing a voltage change in a conductor. This is the Hall effect.
- The sensor reports a magnetic field as a voltage. This voltage can be read just like an analog resistance sensor, using analogRead() or botbook_mcp3002.readAnalog().

Experiment: Is It Magnetic?

- This effect is implemented in sensors from a variety of manufacturers.
- We used the KY-024 Magnetic Detecting Sensor Module (part number 232563) from http://dx.com, shown in Figure 10-3.

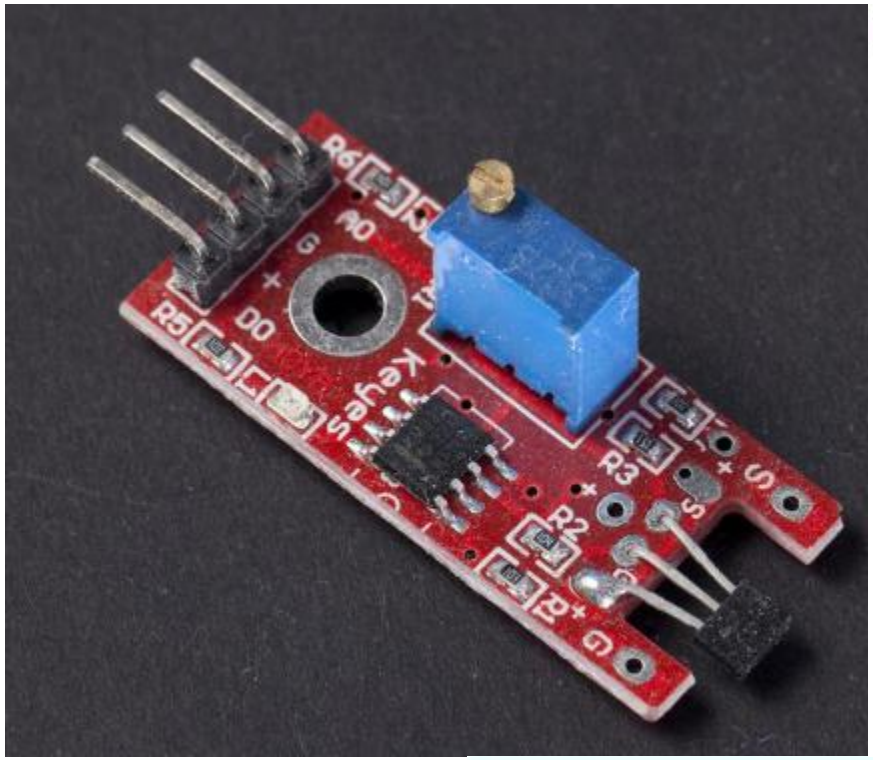


Figure 10-3. The KY-024 magnet detecting sensor

Hall Effect Sensor Code and Connection for Arduino

- Figure 10-4 shows the connections for Arduino. Wire it up as shown, and then run the sketch shown in Example 10-3.

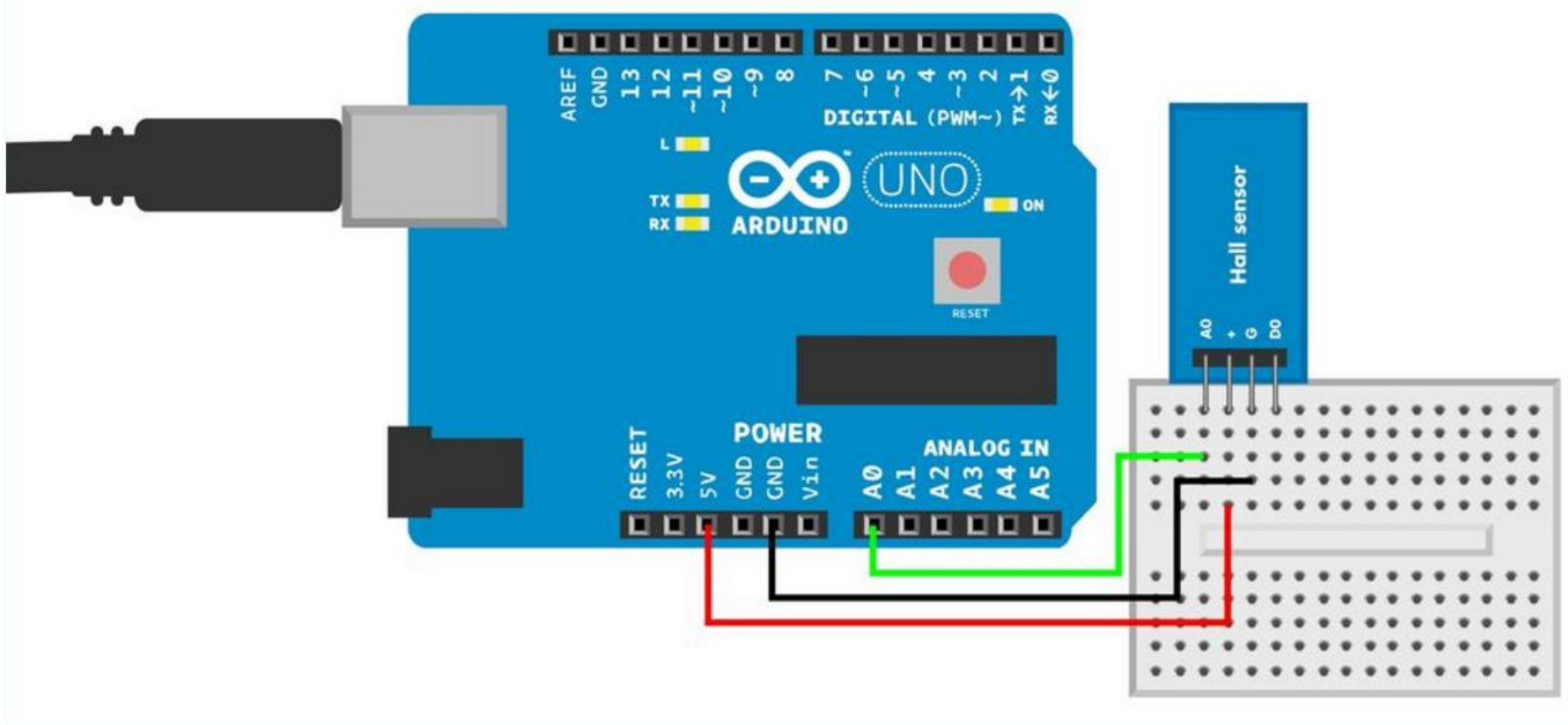


Figure 10-4. Arduino connections for the Hall effect sensor

Example 10-3. hall_sensor.ino

```
// hall_sensor.ino - print raw value and magnets pole
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
const int hallPin = A0;
int rawMagneticStrength = -1;   //初始化一个传感器无法获知的数值，如果出现此数值，说明程序有误需调试
int zeroLevel = 527;    //模拟端口在没有磁场时读取的原始值，此处可能需要根据实际情况调整。

void setup() {
        Serial.begin(115200);
        pinMode(hallPin, INPUT);
}
void loop() {
        rawMagneticStrength = analogRead(hallPin);
        Serial.print("Raw strength: ");
        Serial.println(rawMagneticStrength);
        int zeroedStrength = rawMagneticStrength - zeroLevel;
        Serial.print("Zeroed strength: ");
        Serial.println(zeroedStrength);
        if(zeroedStrength > 0) {
                Serial.println("South pole");
        } else if(zeroedStrength < 0) {
                Serial.println("North pole");
        }
        delay(600); // ms
}
```

```
/* 如果你知道当前传感器从电压转换到高斯的方法，
也就是知道转换系数，那么可以算出具体的磁场大小。
zeroedStrength * conversion
*/
```

Hall Effect Sensor Code and Connection for Raspberry Pi

- Figure 10-5 shows the connectio
- Hook them up as shown, and th

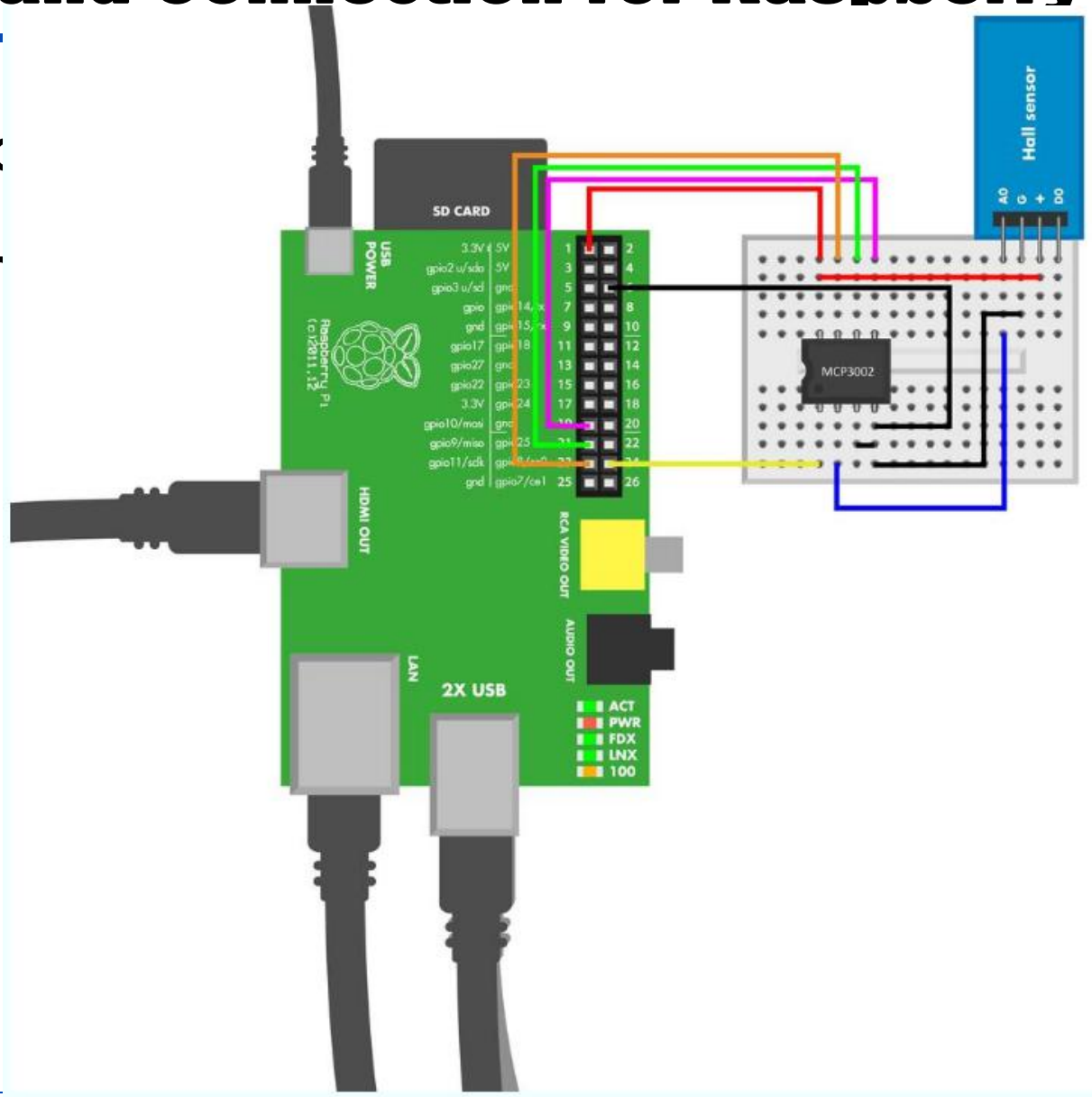


Figure 10-5. Raspberry Pi/Hall effect sensor connections

Example 10-4. hall_sensor.py

```
# hall_sensor.py - print raw value and magnets pole
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_mcp3002 as mcp   #
zeroLevel = 388 #

def main():
        while True:
                rawMagneticStrength = mcp.readAnalog()
                print("Raw strength:  %i " % rawMagneticStrength)
                zeroedStrength = rawMagneticStrength - zeroLevel
                print("Zeroed strength: %i " % zeroedStrength)
                if(zeroedStrength > 0):
                        print("South pole")
                elif(zeroedStrength < 0):
                        print("North pole")
                time.sleep(0.5)

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

Experiment: Magnetic North with LSM303 Compass-Accelerometer

- The LSM303 compass-accelerometer (Figure 10-6) gives you the direction of magnetic north. With the accelerometer, it can correct this reading for its orientation.

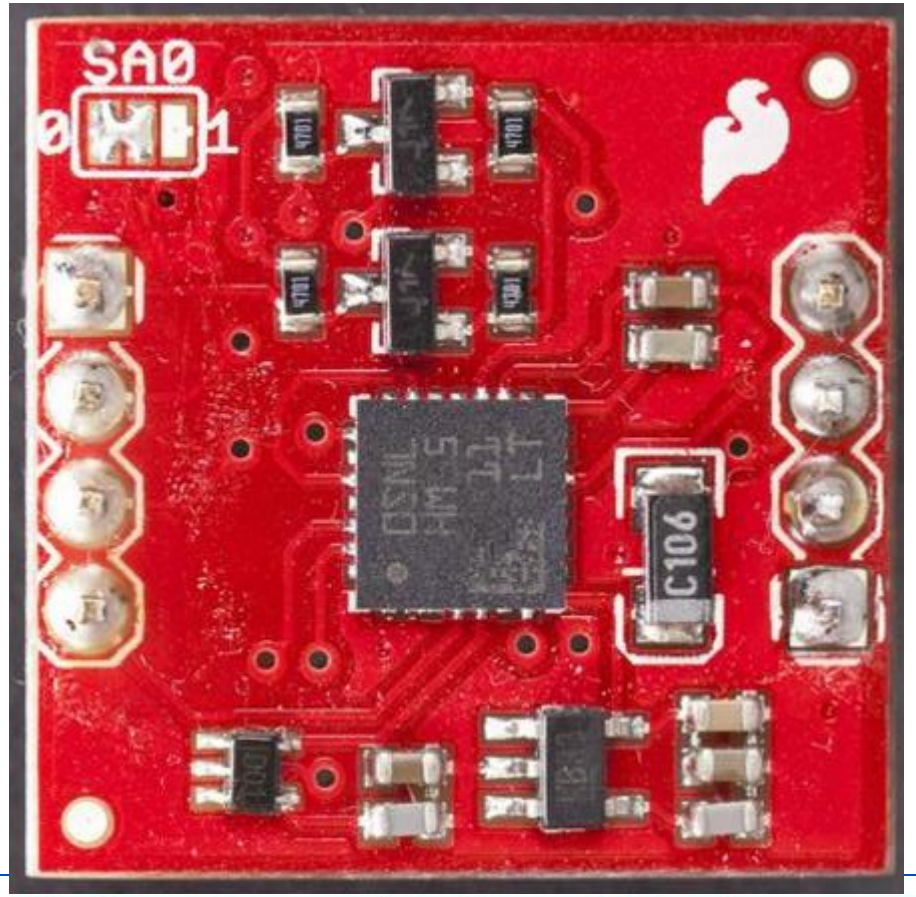


Figure 10-6. SparkFun's LSM303 compass-accelerometer

Experiment: Magnetic North with LSM303 Compass-Accelerometer

- If you have experience orienteering, you know to hold your compass horizontally when setting the map or taking a bearing. But with this sensor, your device can find north even sideways.
- Compass sensors, such as LSM303, are sensitive to external interference. Keep the compass away from power cables and big pieces of metal.
- The LSM303 board we used doesn't have any markings for north. To mark it yourself, turn the board so that the text “SA0” is oriented so you can read it left to right. North is to the right edge of the board when the text SA0 is properly oriented.
- In the output values, a heading of 0 is north.（heading=0，代表北极方向）

Calibrate Your Module

- The compass must be calibrated to get correct values. You can try it out without calibrating, and you should see values. However, you'll get correct values only after calibration.
- Here are the calibration steps:
  - Build the circuit for your chosen platform (Arduino or Raspberry Pi).
  - Run the program to see that you get some values.
  - Change runningMode = 0 in the code to put the device into calibration mode. In calibration mode, the program will show only minimum and maximum values for each axis.
  - Set maximum values initially to zero. For Raspberry Pi, change magMax[] and magMin[]. For Arduino, change each of the six magMax_x, magMax_y … magMin_z.
  - Wave and turn the device around, maybe drawing a figure-eight shape in the air. Continue until the values stop changing. If you can‘t get any higher or lower value, you have found min and max.
  - Hard-code the values into your code. These are the same maximum values you zeroed earlier, magMax[] and magMin[] or magMax_x…

Calibrate Your Module

- Once the calibration is done, change runningMode = 1 to use the sensor normally and see headings.
- Try running the code and calibrating the device. Once you have played with the compass, you can read about implementation details (if you want) in LSM303 Protocol.

LSM303 Code and Connection for Arduino

- Figure 10-7 shows the circuit for Arduino. Hook everything up as shown in the figure, and then run the sketch shown in Example 10-5.

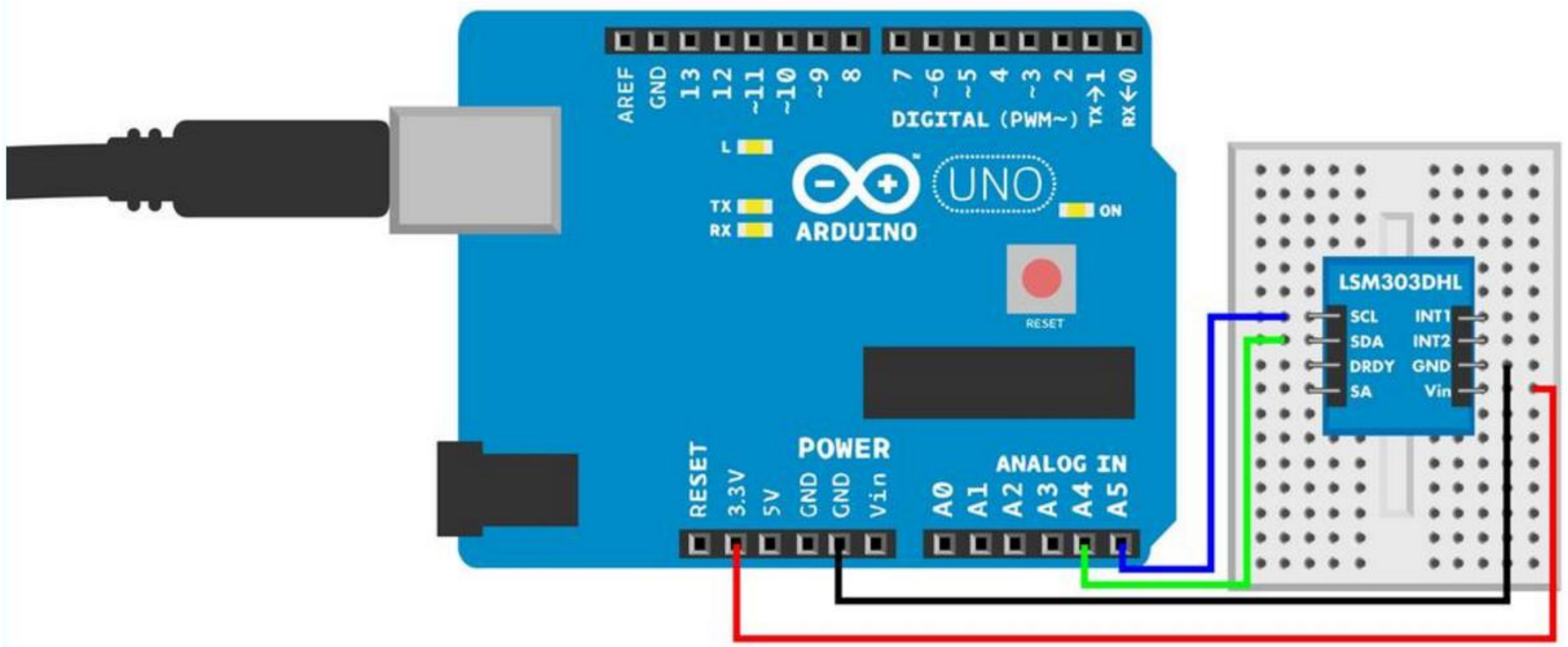


Figure 10-7. LSM303 compass-accelerometer circuit for Arduino

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
#include <Wire.h>
const char accelerometer_address = 0x30 >> 1;   // 地址最后一位是读写信息，向右移动得到真正地址
const char magnetometer_address = 0x3C >> 1;
```

Table 14. SAD+Read/Write patterns

Command	SAD[7:1]	R/W	SAD+R/W

```
const int runningMode = 0;      // 0：校准模式，1：正常模式。需要手工设置，重新编译再执行。
float magMax_x = 0.1; float magMax_y = 0.1; float magMax_z = 0.1;      //在校准之前，先初始化数据参数
float magMin_x = -0.1; float magMin_y = -0.1; float magMin_z = -0.1;
float acc_x = 0; float acc_y = 0; float acc_z = 0;                    //方向变量初始化为0
float mag_x = 0; float mag_y = 0; float mag_z = 0;
int heading = 0;
```

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari


void setup() {
        Serial.begin(115200);
        Wire.begin();      //Ardiuno的I2C库
        Serial.println("Initialize compass");
        initializelsm();      //初始化罗盘设备
        delay(100);
        Serial.println("Start reading heading");
}
```

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
void loop() {
        updateHeading();      //更新方位，更改全局变量，不用返回值
        if(runningMode == 0) {// 反复循环，直到最大值和最小值不变了，校准模式仅输出最大值和最小值
                magMax_x = max(magMax_x, mag_x);
                magMax_y = max(magMax_y, mag_y);
                magMax_z = max(magMax_z, mag_z);
                magMin_x = min(magMin_x, mag_x);
                magMin_y = min(magMin_y, mag_y);
                magMin_z = min(magMin_z, mag_z);      Serial.print("Max x y z: ");
                Serial.print(magMax_x); Serial.print(" ");
                Serial.print(magMax_y); Serial.print(" ");
                Serial.print(magMax_z); Serial.print(" "); Serial.print("Min x y z: ");
                Serial.print(magMin_x); Serial.print(" ");
                Serial.print(magMin_y); Serial.print(" ");
                Serial.print(magMin_z); Serial.println(" ");
        } else {
                calculateHeading();    Serial.println(heading);    //计算指向上方的单位向量
        }
        delay(100); // ms
}
```

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
void initializelsm() {
        write_i2c(accelerometer_address, 0x20, 0x27);       //
        write_i2c(magnetometer_address, 0x02, 0x00);        //
}
```

Table 17. Register address map

Name	Slave address	Type	Register address		Default
			Hex	Binary	
Reserved (do not modify)	*Table 14*		00 - 1F	--	--
CTRL_REG1_A	*Table 14*	rw	20	010 0000	00000111
MR_REG_M	*Table 16*	rw	02	00000010	00000011

Table 18. CTRL_REG1_A register

ODR3	ODR2	ODR1	ODR0	LPen	Zen	Yen	Xen

Table 76. MR_REG

0(1)	0(1)	0(1)	0(1)	0(1)	0(1)	MD1	MD0

1. This bit must be set to '0' for correct working of the device.

Table 19. CTRL_REG1_A description

ODR3-0	Data rate selection. Default value: 0 (0000: power-down, others: refer to *Table 20.*)
LPen	Low-power mode enable. Default value: 0 (0: normal mode, 1: low-power mode)
Zen	Z axis enable. Default value: 1 (0: Z axis disabled, 1: Z axis enabled)
Yen	Y axis enable. Default value: 1 (0: Y axis disabled, 1: Y axis enabled)
Xen	X axis enable. Default value: 1 (0: X axis disabled, 1: X axis enabled)

Table 78. Magnetic sensor operating mode

MD1	MD0	Mode
0	0	Continuous-conversion mode
0	1	Single-conversion mode
1	0	Sleep-mode. Device is placed in sleep-mode
1	1	Sleep-mode. Device is placed in sleep-mode

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
void updateHeading() {
        updateAccelerometer();
        updateMagnetometer();
}
```

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
void updateAccelerometer() {    //按照手册上的要求读取加速度传感器值，并更新全局变量
        Wire.beginTransmission(accelerometer_address);
        Wire.write(0x28 | 0x80);      // 准备读取加速度x，y，z三轴的值，每个值2个字节
        Wire.endTransmission(false);
        Wire.requestFrom(accelerometer_address, 6, true);     //
        int i = 0;
        while(Wire.available() < 6) {
                i++;
                if(i > 1000) {
                        Serial.println("Error reading from accelerometer i2c");
                        return;
                }
        }
```

OUT_X_L_A (28h), OUT_X_H_A (29h)

X-axis acceleration data. The value is expressed in 2's complement.

OUT_Y_L_A (2Ah), OUT_Y_H_A (2Bh)

Y-axis acceleration data. The value is expressed in 2's complement.

OUT_Z_L_A (2Ch), OUT_Z_H_A (2Dh)

Z-axis acceleration data. The value is expressed in 2's complement.

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

        uint8_t  axel_x_l = Wire.read();
        uint8_t  axel_x_h = Wire.read();
        uint8_t  axel_y_l = Wire.read();
        uint8_t  axel_y_h = Wire.read();
        uint8_t  axel_z_l = Wire.read();
        uint8_t  axel_z_h = Wire.read();
        acc_x = (axel_x_l | axel_x_h << 8) >> 4;
        acc_y = (axel_y_l | axel_y_h << 8) >> 4;
        acc_z = (axel_z_l | axel_z_h << 8) >> 4;
}
```

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
void updateMagnetometer() { //
        Wire.beginTransmission(magnetometer_address);
        Wire.write(0x03);
        Wire.endTransmission(false);
        Wire.requestFrom(magnetometer_address, 6, true);
        int i = 0;
        while(Wire.available() < 6) {
                i++;
                if(i > 1000) {
                        Serial.println("Error reading from magnetometer i2c");
                        return;
                }
        }
```

OUT_X_H_M (03), OUT_X_LH_M (04h)

X-axis magnetic field data. The value is expressed as 2's complement.

OUT_Z_H_M (05), OUT_Z_L_M (06h)

Z-axis magnetic field data. The value is expressed as 2's complement.

OUT_Y_H_M (07), OUT_Y_L_M (08h)

Y-axis magnetic field data. The value is expressed as 2's complement.

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

        uint8_t  axel_x_h = Wire.read();
        uint8_t  axel_x_l = Wire.read();
        uint8_t  axel_y_h = Wire.read();
        uint8_t  axel_y_l = Wire.read();
        uint8_t  axel_z_h = Wire.read();
        uint8_t  axel_z_l = Wire.read();
        mag_x = (int16_t)(axel_x_l | axel_x_h << 8);
        mag_y = (int16_t)(axel_y_l | axel_y_h << 8);
        mag_z = (int16_t)(axel_z_l | axel_z_h << 8);
}
```

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
//Heading to north
void calculateHeading() {
// Up unit vector
        float dot = acc_x*acc_x + acc_y*acc_y + acc_z*acc_z;  //计算向量的点积
        float magnitude = sqrt(dot);                          //计算大小（长度）
        float nacc_x = acc_x / magnitude;          // 用每个维度值除以长度得到指向上方的单位向量
        float nacc_y = acc_y / magnitude;
        float nacc_z = acc_z / magnitude;

// Apply calibration
        mag_x = (mag_x - magMin_x) / (magMax_x - magMin_x) * 2 - 1.0;          //应用校准数据
        mag_y = (mag_y - magMin_y) / (magMax_y - magMin_y) * 2 - 1.0;
        mag_z = (mag_z - magMin_z) / (magMax_z - magMin_z) * 2 - 1.0;
```

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
// East
        float ex = mag_y*nacc_z - mag_z*nacc_y;          //
        float ey = mag_z*nacc_x - mag_x*nacc_z;
        float ez = mag_x*nacc_y - mag_y*nacc_x;
        dot = ex*ex + ey*ey + ez*ez;                     // 将东方向归一化。
        magnitude = sqrt(dot);
        ex /= magnitude;  ey /= magnitude;  ez /= magnitude;      //

// Project
        float ny = nacc_z*ex - nacc_x*ez;     //
        float dotE = -1 * ey; //
        float dotN = -1 * ny;
// Angle
        heading = atan2(dotE, dotN) * 180 / M_PI;     //
        heading = round(heading);
        if (heading < 0) heading +=360; //
}
```

Example 10-5. lsm303.ino

```
// lsm303.ino - normal use and calibration of LSM303DLH compass-accelerometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari


void write_i2c(char address, unsigned char reg, const uint8_t data){
        Wire.beginTransmission(address);
        Wire.write(reg);
        Wire.write(data);
        Wire.endTransmission();
}
```

LSM303 Code and Connection for Raspberry Pi

- Figure 10-8 shows the circuit for Raspberry Pi.
- Hook everything up as shown, and then run the code listed in Example 10-6.

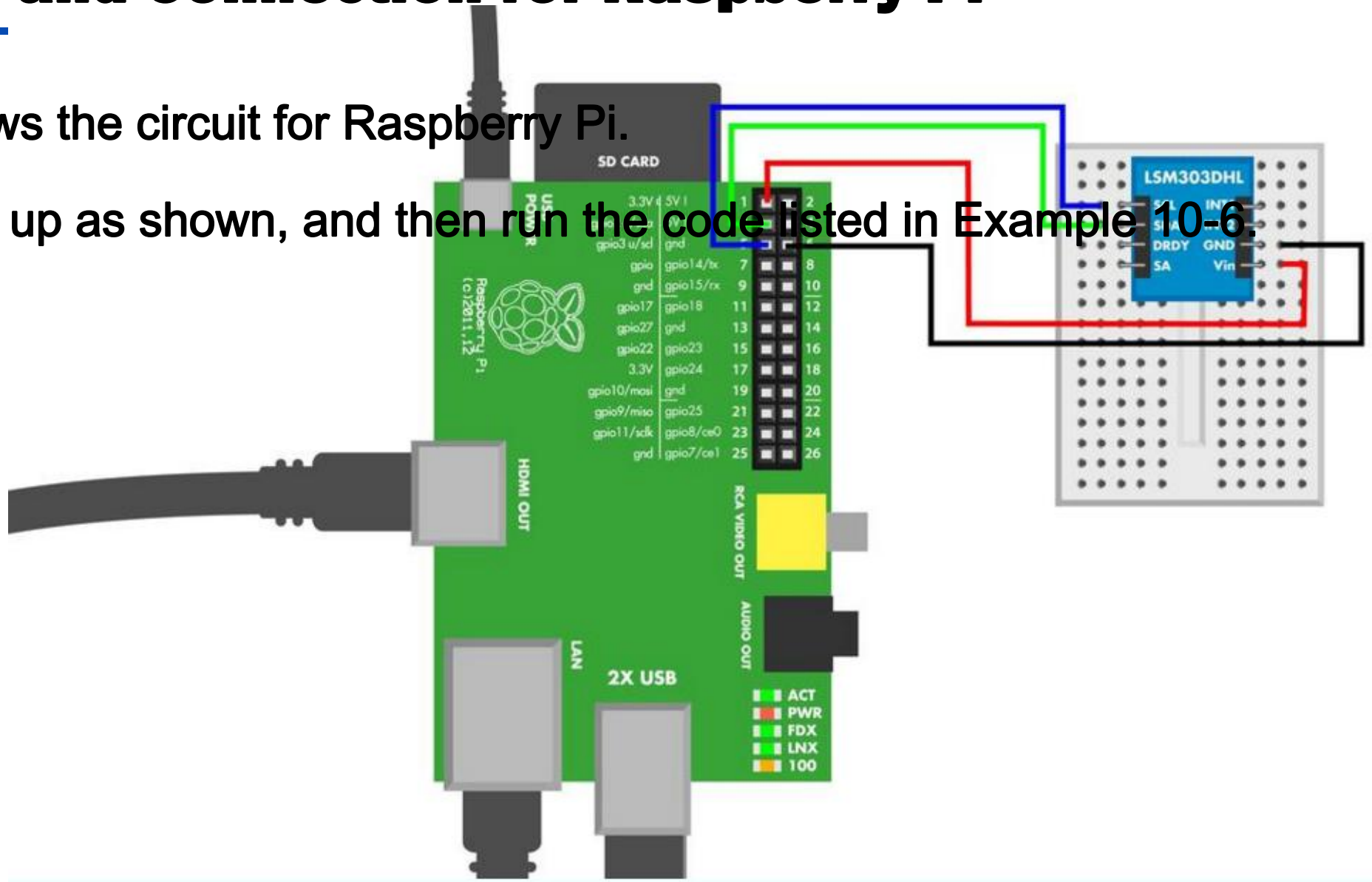


Figure 10-8. LSM303 compass-accelerometer circuit for Raspberry Pi

Example 10-6. lsm303.py

```
# lsm303.py - normal use and calibration of LSM303DLH compass-accelerometer
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import smbus # sudo apt-get -y install python-smbus
import struct
import math

accelerometer_address = 0x30 >> 1
magnetometer_address = 0x3C >> 1

calibrationMode = True  #

magMax = [ 0.1, 0.1, 0.1 ]      #
magMin = [ 0.1, 0.1, 0.1 ]

acc = [ 0.0, 0.0, 0.0 ] #
mag = [ 0.0, 0.0, 0.0 ]

heading = 0
```

Example 10-6. lsm303.py

```
# lsm303.py - normal use and calibration of LSM303DLH compass-accelerometer
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
def initlsm():
        global bus
        bus = smbus.SMBus(1)
        bus.write_byte_data(accelerometer_address, 0x20, 0x27)      #
        bus.write_byte_data(magnetometer_address, 0x02, 0x00)  #

def updateAccelerometer():
        global acc    #
        bus.write_byte(accelerometer_address, 0x28 | 0x80)    #
        rawData = ""
        for i in range(6):
                rawData += chr(bus.read_byte_data(accelerometer_address, 0x28+i)) #

        data = struct.unpack('<hhh',rawData) #
        acc[0] = data[0] >> 4 #
        acc[1] = data[1] >> 4
        acc[2] = data[2] >> 4
```

Example 10-6. lsm303.py

```
# lsm303.py - normal use and calibration of LSM303DLH compass-accelerometer
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
def updateMagnetometer():       #
        global mag
        bus.write_byte(magnetometer_address, 0x03)
        rawData = ""
        for i in range(6):
                rawData += chr(bus.read_byte_data(magnetometer_address, 0x03+i))

        data = struct.unpack('>hhh',rawData)
        mag[0] = data[0]
        mag[1] = data[1]
        mag[2] = data[2]
```

Example 10-6. lsm303.py

```
# lsm303.py - normal use and calibration of LSM303DLH compass-accelerometer
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
def calculateHeading():
        global heading, acc, mag
        normalize(acc)       ##normalize
        #use calibration data
        mag[0] = (mag[0] - magMin[0]) / (magMax[0] - magMin[0]) * 2.0 - 1.0   #
        mag[1] = (mag[1] - magMin[1]) / (magMax[1] - magMin[1]) * 2.0 - 1.0
        mag[2] = (mag[2] - magMin[2]) / (magMax[2] - magMin[2]) * 2.0 - 1.0

        e = cross(mag, acc)   #
        normalize(e)  #

        n = cross(acc,e)      #
        dotE = dot(e,[0.0, -1.0, 0.0])       #
        dotN = dot(n,[0.0, -1.0, 0.0])
        heading = round(math.atan2(dotE, dotN) * 180.0 / math.pi)     #
        if heading < 0:       #
                heading += 360      #
```

Example 10-6. lsm303.py

```
# lsm303.py - normal use and calibration of LSM303DLH compass-accelerometer
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
def normalize(v):       #
        magnitude = math.sqrt(dot(v,v))
        v[0] /= magnitude
        v[1] /= magnitude
        v[2] /= magnitude

def dot(v1, v2):        #
        return v1[0]*v2[0] + v1[1]*v2[1] + v1[2]*v2[2]

def cross(v1, v2):      #
        vr = [0.0, 0.0, 0.0]
        vr[0] = v1[1] * v2[2] - v1[2] * v2[1]
        vr[1] = v1[2] * v2[0] - v1[0] * v2[2]
        vr[2] = v1[0] * v2[1] - v1[1] * v2[0]
        return vr
```

Example 10-6. lsm303.py

```
# lsm303.py - normal use and calibration of LSM303DLH compass-accelerometer
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
def main():
        initlsm()
        while True:
            updateAccelerometer()       #
            updateMagnetometer()

            if calibrationMode: #
                magMax[0] = max(magMax[0], mag[0])
                magMax[1] = max(magMax[1], mag[1])
                magMax[2] = max(magMax[2], mag[2])
                magMin[0] = min(magMin[0], mag[0])
                magMin[1] = min(magMin[1], mag[1])
                magMin[2] = min(magMin[2], mag[2])
                print("magMax = [ %.1f, %.1f, %.1f ]" % (magMax[0], magMax[1], magMax[2]))
                print("magMin = [ %.1f, %.1f, %.1f ]" % (magMin[0], magMin[1], magMin[2]))
            else:
                calculateHeading()      #
                print(heading)
            time.sleep(0.5)

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

LSM303 Protocol

- To report measured values, LSM303 uses the industry standard I2C protocol. As I2C is quite strictly defined, it's usually easier than other similar protocols like SPI.
- LSM303 is an I2C slave, so the microcontroller (Arduino or Raspberry Pi) is the master. The master initiates communication by asking for values.
- I2C communication consists of simply sending and reading values that are specified in the component's data sheet.
- You can find the data sheet for this component by searching the Web for "LSM303DLH data sheet."

LSM303简介

- 电子罗盘是一种重要的导航工具，能实时提供移动物体的航向和姿态。随着半导体工艺的进步和手机操作系统的发展，集成了越来越多传感器的智能手机变得功能强大，很多手机上都实现了电子罗盘的功能。而基于电子罗盘的应用（如Android的Skymap）在各个软件平台上也流行起来。
- 要实现电子罗盘功能，需要一个检测磁场的三轴磁力传感器和一个三轴加速度传感器。随着微机械工艺的成熟，意法半导体推出将三轴磁力计和三轴加速计集成在一个封装里的二合一传感器模块LSM303DLH，方便用户在短时间内设计出成本低、性能高的电子罗盘。本文以LSM303DLH为例讨论该器件的工作原理、技术参数和电子罗盘的实现方法。

地磁场和航向角的背景知识

- 如图所示，地球的磁场象一个条形磁体一样由磁南极指向磁北极。在磁极点处磁场和当地的水平面垂直，在赤道磁场和当地的水平面平行，所以在北半球磁场方向倾斜指向地面。用来衡量磁感应强度大小的单位是Tesla或者Gauss（1Tesla=10000Gauss）。随着地理位置的不同，通常地磁场的强度是0.4-0.6 Gauss。需要注意的是，磁北极和地理上的北[illegible]1度左右的夹角。

地磁场和航向角的背景知识

- **地磁场是一个矢量，对于一个固定的地点来说，这个矢量可以被分解为两个与当地水平面平行的分量和一个与当地水平面垂直的分量。如果保持电子罗盘和当地的水平面平行，那么罗盘中磁力计的三个轴就和这三个分量对应起来，如图2所示。**

实际上对水平方向的两个分量来说，他们的矢量和总是指向磁北的。罗盘中的航向角（Azimuth）就是当前方向和磁北的夹角。由于罗盘保持水平，只需要用磁力计水平方向两轴（通常为X轴和Y轴）的检测数据就可以用式1

$$Azimuth = \arctan(\frac{Hy}{Hx})$$

计算出航向角。当罗盘水平旋转的时候，航向角在0- 360之间变化。

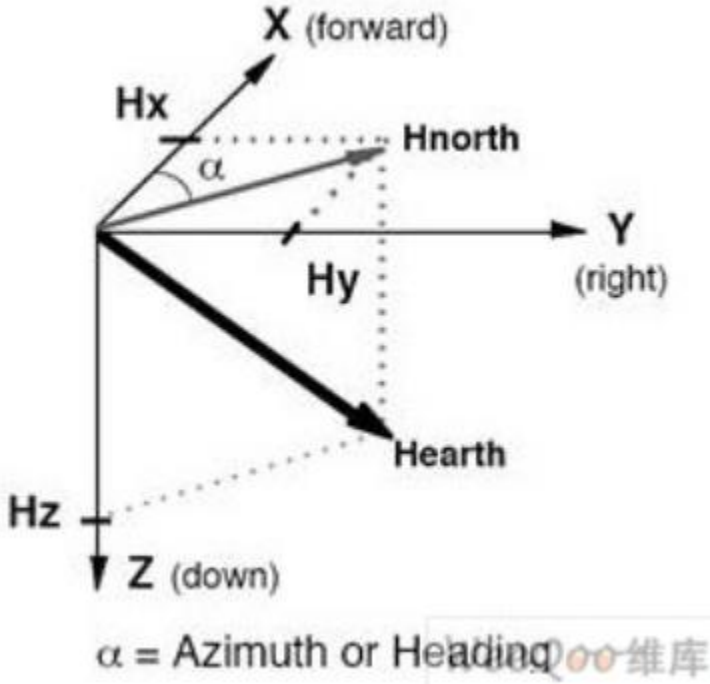
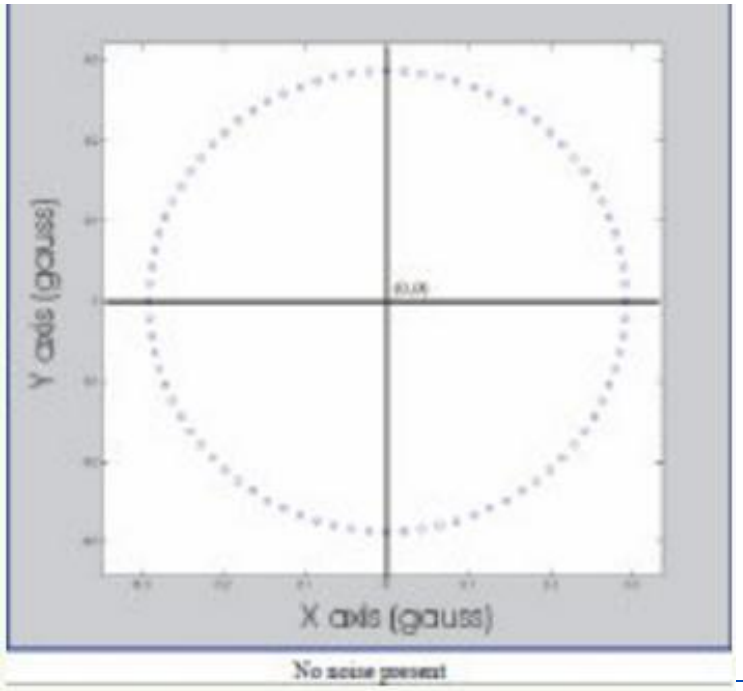


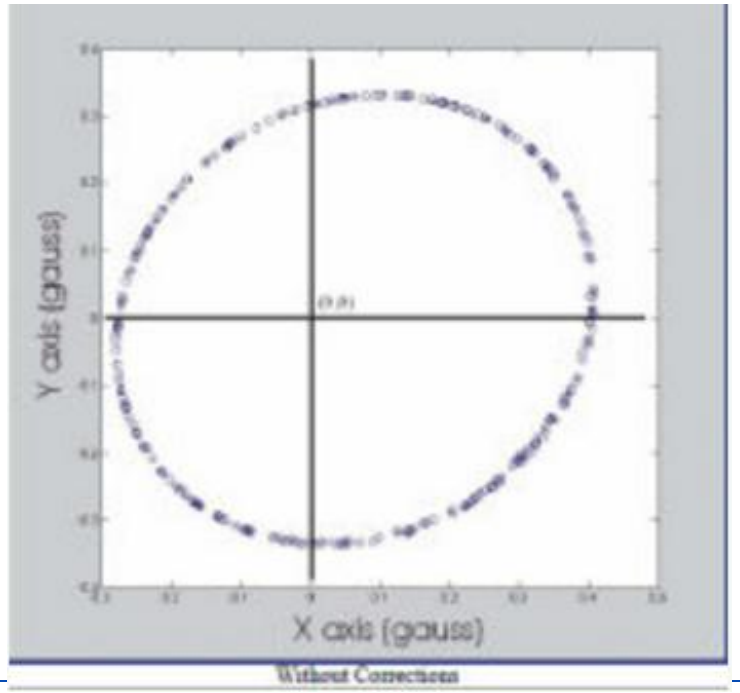
图2 地磁场矢量分解示意图

磁力计校准

- 如果磁力计在含有附加的局部磁场的环境中进行操作，磁力计的输出做附加的修正将是必要的。
- 在没有任何本地磁场的影响下，下图1可以通过旋转设备360°产生的平面，图2为引入本地磁场。



No noise present



Without Corrections

磁力计校准

- 修正的输出可以根据下面的方法来计算：

1）在磁场干扰的条件下进行，数据收集设备被旋转360°。

2）数据进行分析，以产生偏差的偏移和灵敏度的比例因子，以补偿所述干扰。

- 例子：从数据中发现的X和Y磁强计的最大和最小输出：

Xmin = -0.284gauss　　Xmax = +0.402gauss

Ymin = -0.322gauss　　Ymax = +0.246gauss

- 从中可以看出X轴的数据，X具有更大的反应，我们设置其比例系数为1，即Xs = 1
- 再计算其他比例系数：

$$Ys = \frac{(Xmax - Xmin)}{(Ymax - Ymin)}$$

磁力计校准

- **对于偏置补偿：**

$$Xb = Xs[1/2(Xmax - Xmin) - Xmax]$$

$$Yb = Ys[1/2(Ymax - Ymin) - Ymax]$$

正确的输出：

$$Xout = Xin*Xs + Xb$$

$$Yout = Yin*Ys + Yb$$

- **二.**

1）水平匀速旋转，收集XY轴数据

2）转动器材90度（Z轴）匀速转动以收集Z轴数据

$$Xoffset = （Xmax + Xmin）/2$$

$$Yoffset = （Ymax + Ymin）/2$$

$$Zoffset = （Zmax + Zmin）/2$$

磁力计校准

- **将磁力计读到的裸值减去offset，得到用做角度计算的Heading值**

  XH = X - Xoffset

  YH = Y - Yoffset

  ZH = Z - Zoffset

- **水平测试，得到的方位角** = arctanYH/XH
- **非水平测试，需要使用加速计进行倾角补偿，先计算出翻滚角Roll和俯仰角Pitch，然后计算Heading值：**

  XH = x*cos(P)+Y*sin(R)*sin(P)-Z*cos(R)*sin(p)

  YH = Y*cos(R)+Z*sin(R)

Compass Heading Calculation

- The sensor already knows where north is. It gives this value as a three-dimensional vector n = [nx, ny, nz].
- The compass sensor is really three-dimensional, and this north-pointing vector is already correct. So why do you need to do this math?
- Humans usually want a compass heading, a number between 0 deg and 360 deg. A compass heading is two dimensional. The compass heading tells how much we have turned right from north. The turn is measured by degrees, growing clockwise, as you turn right.
- To convert the three-dimensional north vector to degrees, you use the up vector from the accelerometer.

Compass Heading Calculation

- All vectors are relative to the device, so that Y points in the device-forward direction. Thus, the vector [0, 1, 0] would point directly device forward.

Arduino	Raspberry Pi	数值	说明
heading	heading	int, 0 deg .. 360 deg	北方和前方(y)之间的角度。我们要向右旋转多少度呢？
mag_x, mag_y…	mag[]	有符号整数，有符号整数的列表	指向北方的三维向量
acc_x, acc_y…	acc[]	有符号整数，有符号整数的列表	指向上方（重力的反方向）的三维向量
[0, 0, 1]			指向设备上方的向量（Z向上）
[0, 1, 0]			指向前方的向量（Y向前）
[1, 0, 0]			指向右方的向量（X向右）

Compass Heading Calculation

- Both Arduino and Raspberry Pi codes use the same kind of logic. For clarity, we'll use Python here, but the Arduino code works in the same way and uses similar variable names.
- First, read up and north from the sensor. Get these vectors:
  - acc (up gravitywise, 3d)
  - mag (north, 3d)
- The angle between these vectors can be anything.
- Calibration is then applied to mag, the north-pointing vector.

Compass Heading Calculation

- Next you need to calculate the east vector. This vector is perpendicular to both of the vectors mag and acc. So there is a 90-degree angle between acc (gravity up) and east, and there is a 90-degree angle between mag and east. Thus, you can calculate the east vector with a vector cross product.

  e = cross(mag, acc)

  e = normalize(e)

- After normalization, e is a unit vector (length 1) pointing east. This east is perpendicular to gravity up.Now the north (n) vector and east (e) vector form an NE plane. This NE plane is likely still in an angle to the device-horizontal XY plane.

Compass Heading Calculation

- To get the n and e planes aligned to gravity horizontal,n = cross(acc, e)
- Now you have north n and east e in the gravity-horizontal plane.
- The compass heading must be calculated for the user. The heading tells how many degrees to the right the device has turned from the north. From the device, a vector is projected to the NE plane. Then you can calculate the angle between the projected vector and north:
  - dotE = dot(e,[0.0, -1.0, 0.0])
  - dotN = dot(n,[0.0, -1.0, 0.0])
  - headingRad = math.atan2(dotE, dotN)
  - headingDeg = headingRad / (2*math.pi) * 360
- Finally, the user gets compass heading as degrees.

Experiment: Hall Switch

- A Hall switch (Figure 10-9) detects if a magnet is nearby. They are often used in measuring how fast a wheel spins. Before GPS, many bike speedometers used Hall switches.

Figure 10-9. Hall switch

Experiment: Hall Switch

- As you bring a magnet near the Hall switch, the code in this experiment prints "switch triggered."
- There are many Hall switches available from different manufactures. We used an affordable and robust Hall Magnetic Sensor (part number 141363) from http://dx.com. That sensor has a nice built-in LED that lights up when a magnet is near. It has weird wire colors: black is data (yes, black goes to D2 rather than the usual GND), blue is ground, and brown is +5 V.

Hall Switch Code and Connection for Arduino

- Figure 10-10 shows the wiring diagram for Arduino. Wire it up as shown, and then run the code shown in Example 10-7.

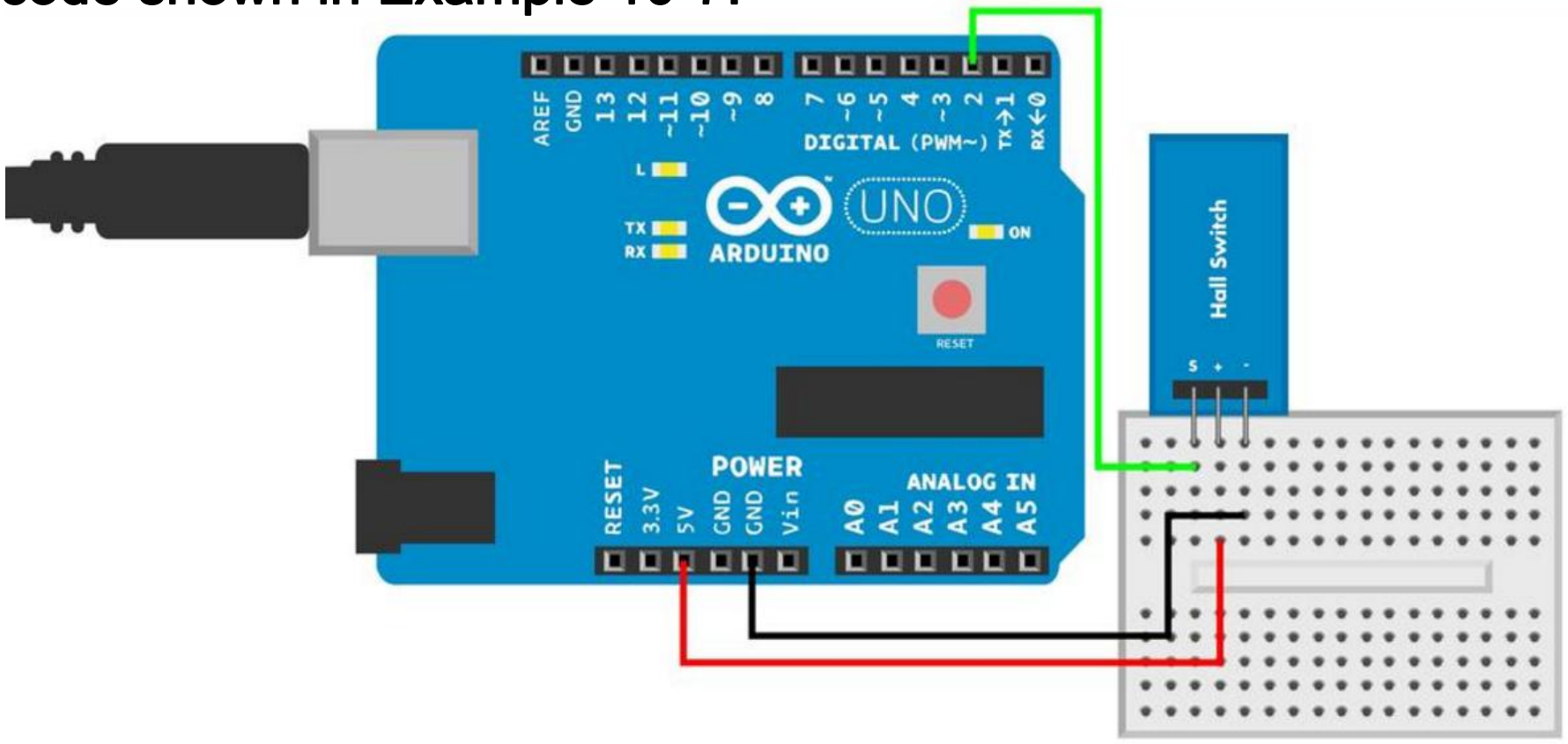


Figure 10-10. Arduino circuit for Hall switch

Example 10-7. hall_switch.ino

```
// hall_switch.ino - write to serial if magnet triggers the switch
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
int switchPin=2;

void setup() {
        Serial.begin(115200);
        pinMode(switchPin, INPUT);
        digitalWrite(switchPin, HIGH);
}

void loop() {
        int switchState=digitalRead(switchPin);       //
        if (switchState == LOW) {
                Serial.println("YES, magnet is near");
        } else {
                Serial.println("no");
        }
        delay(50);
}
```

Hall Switch Code and Connection for Raspberry Pi

- Figure 10-11 shows the ci
- Hook everything up as sh e 10-8.

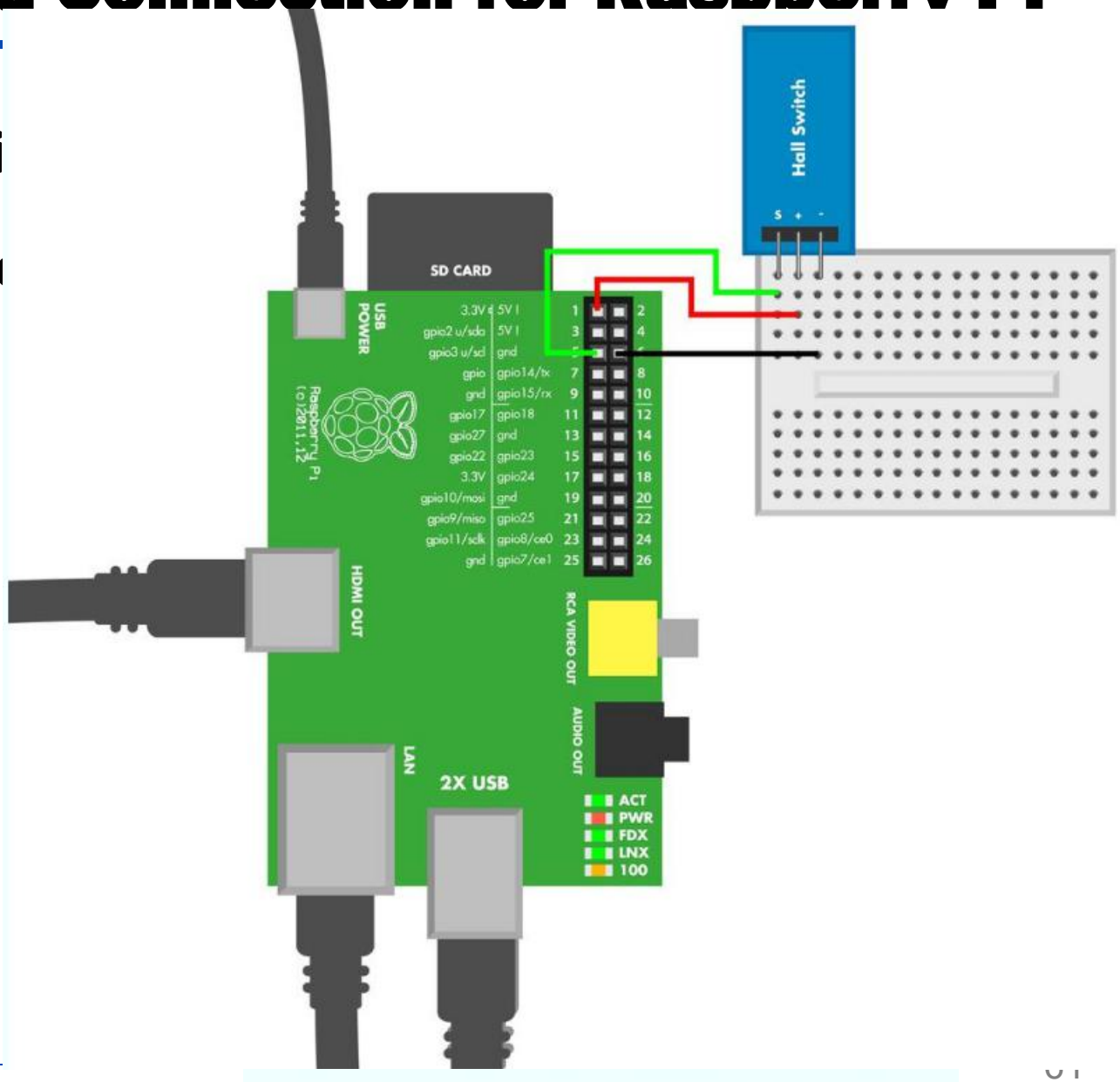


Figure 10-11. Raspberry Pi circuit for Hall switch

Example 10-8. hall_switch.py

```
# hall_switch.py - write to screen if magnet triggers the switch
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import botbook_gpio as gpio     #

def main():
        switchPin = 3   # has internal pull-up  #
        gpio.mode(switchPin, "in")
        while (True):
                switchState = gpio.read(switchPin)      #
                if(switchState == gpio.LOW):
                        print "switch triggered"
                time.sleep(0.3)

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

Test Project: Solar Cell Web Monitor

- Turn your Raspberry Pi into a web server and monitor the voltage of your solar cells remotely (Figure 10-12).

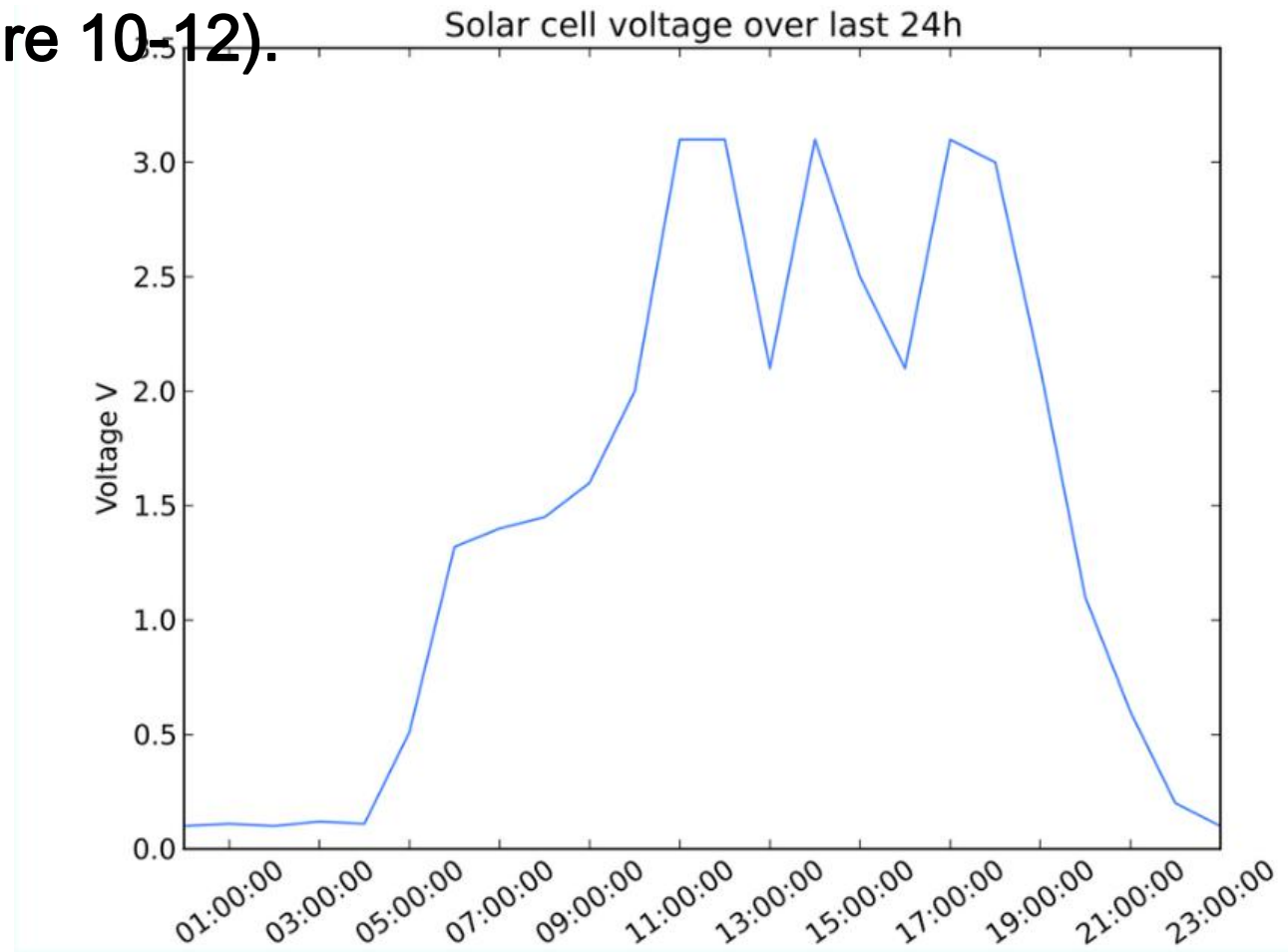


Figure 10-12. Solar cell power graph

Test Project: Solar Cell Web Monitor

- **What You'll Learn in the Solar Cell Web Monitor project:**
  - Measure voltage of your solar cells, and then report it on your own web server.
  - Turn the Raspberry Pi into a web server—using the most popular web server in the world!
  - Create timed tasks using the cron scheduler that keep running even if Raspberry is rebooted.
  - Draw graphs with Python matplotlib.
- **Big, public web servers use many of the same techniques you learn here. We teach Apache and cron to Linux students who work on servers, so these techniques aren't specific to embedded systems or robots.**

Connecting Solar Cells

- Do you still remember IKEA's Solvinden lamp, which we used to build Chameleon Dome? In this project we're going to use the leftover solar panels from it. If you didn't use Solvinden in the first place, just adjust these instructions to suit the solar cells you are using. First, desolder the red wire marked in Figure 10-13.

Connecting Solar Cells

- Do you still remember IKEA's Solvinden lamp, which we used to build Chameleon Dom tover solar panels from it. If you did ust these instructions to su der the red wire marked in Figure

Connecting Solar Cells

- Do you still remembe l to build Chameleon Dome? I ver solar panels from it. If you st adjust these instructions to suit th the red wire marked in Figure 10-

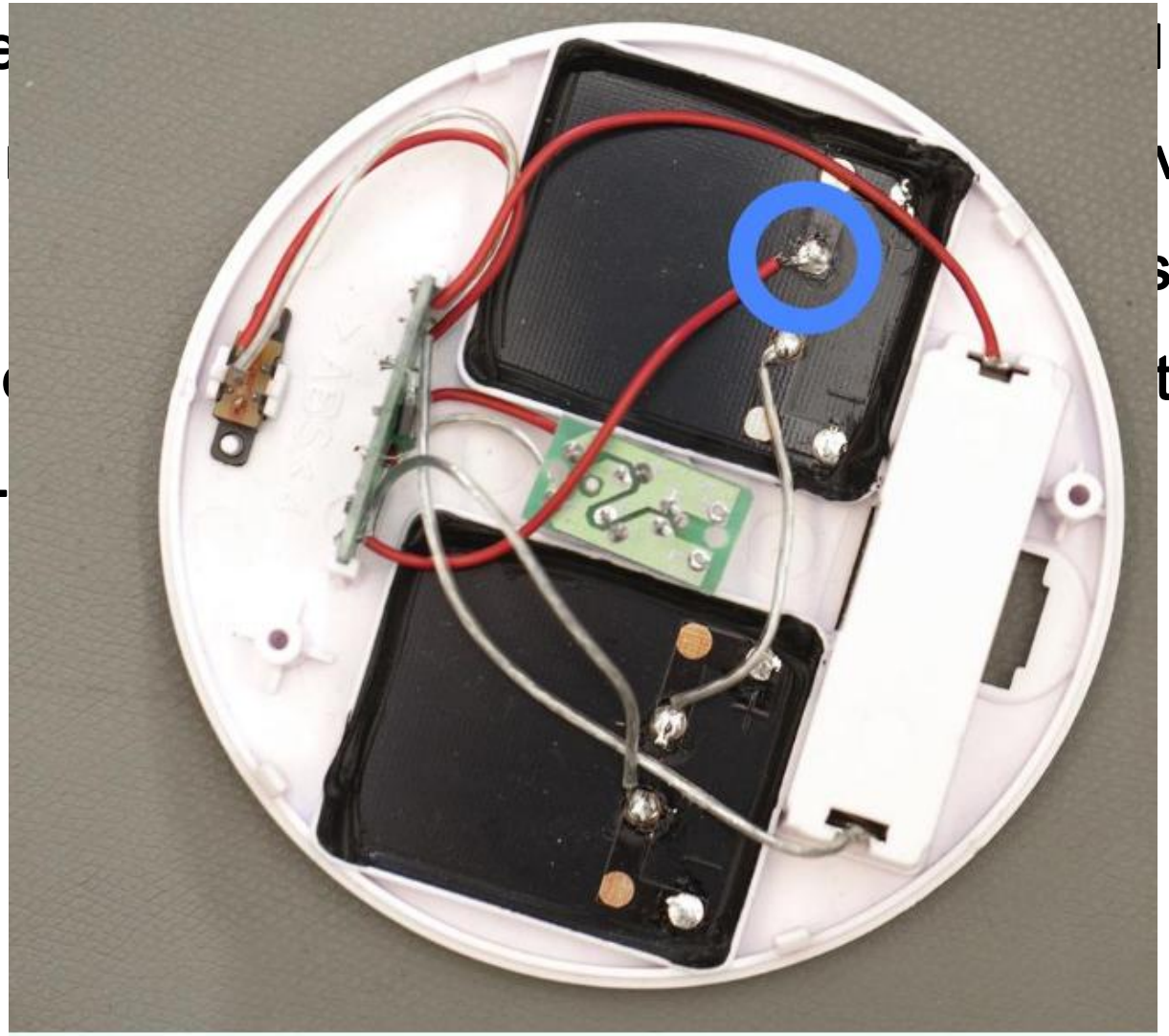


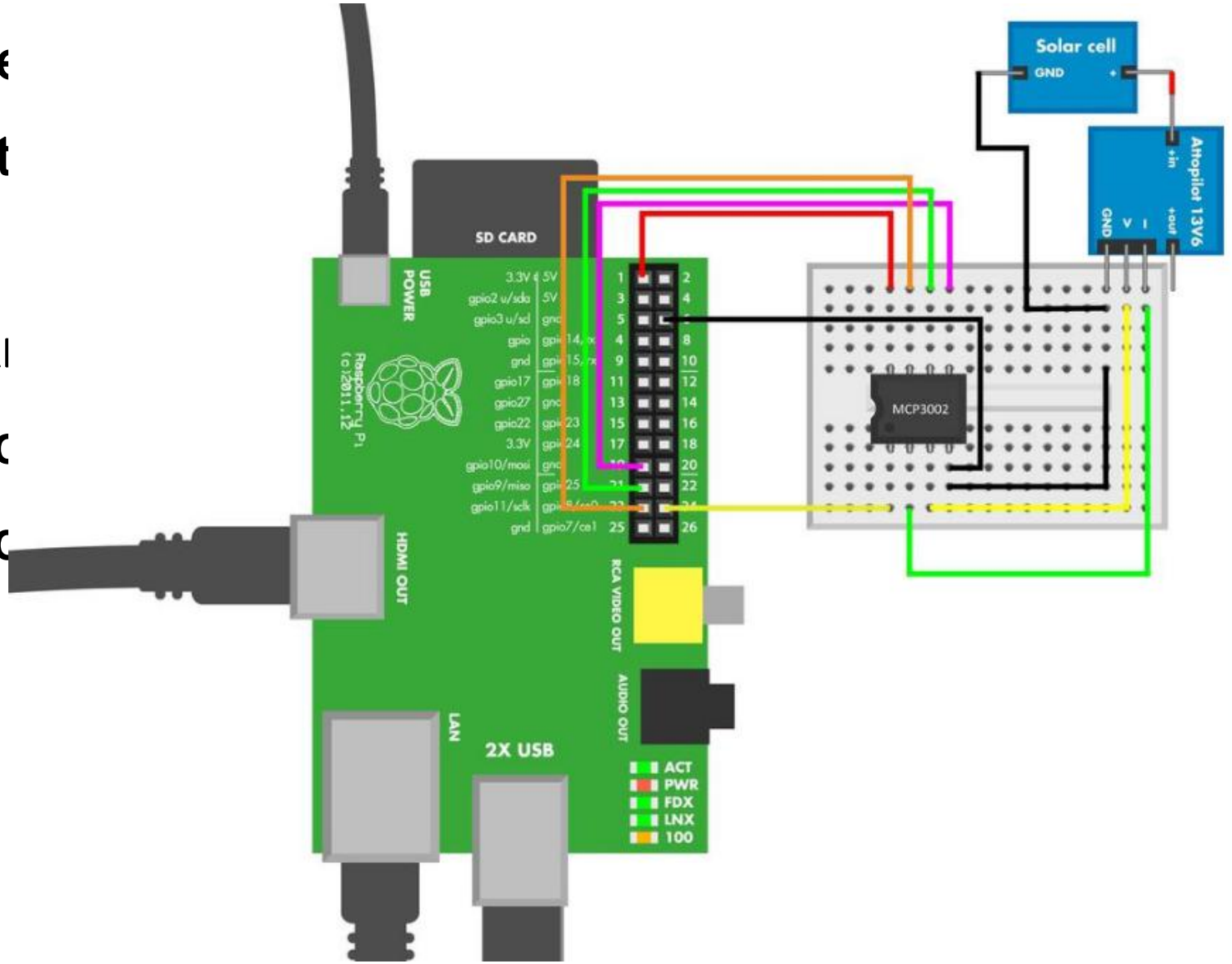
Figure 10-13. Desolder red wire

Connecting Solar Cells

- You don't need IKEA's Solvinden lamp to build this. Just connect a solar cell according to the circuit diagram Figure 10-17. Because you're using a current sensor with a maximum measured voltage of 13.6 V, make sure the solar cell or panel doesn't put out more than 13.6 V. If you're using a different solar cell, it probably has power leads, so you won't need to follow the Solvinden disassembly steps.

Connecting Solar Cells

- You don’ t nee ct a solar cell according t e using a current sensor sure the solar cell or pa ing a different solar c d to follow the Solvinden c



Connecting Solar Cells

- Cut the jumper wires as shown in Figure 10-14 and solder them to the current sensor and to the solar cells as shown in Figure 10-15. The final product is

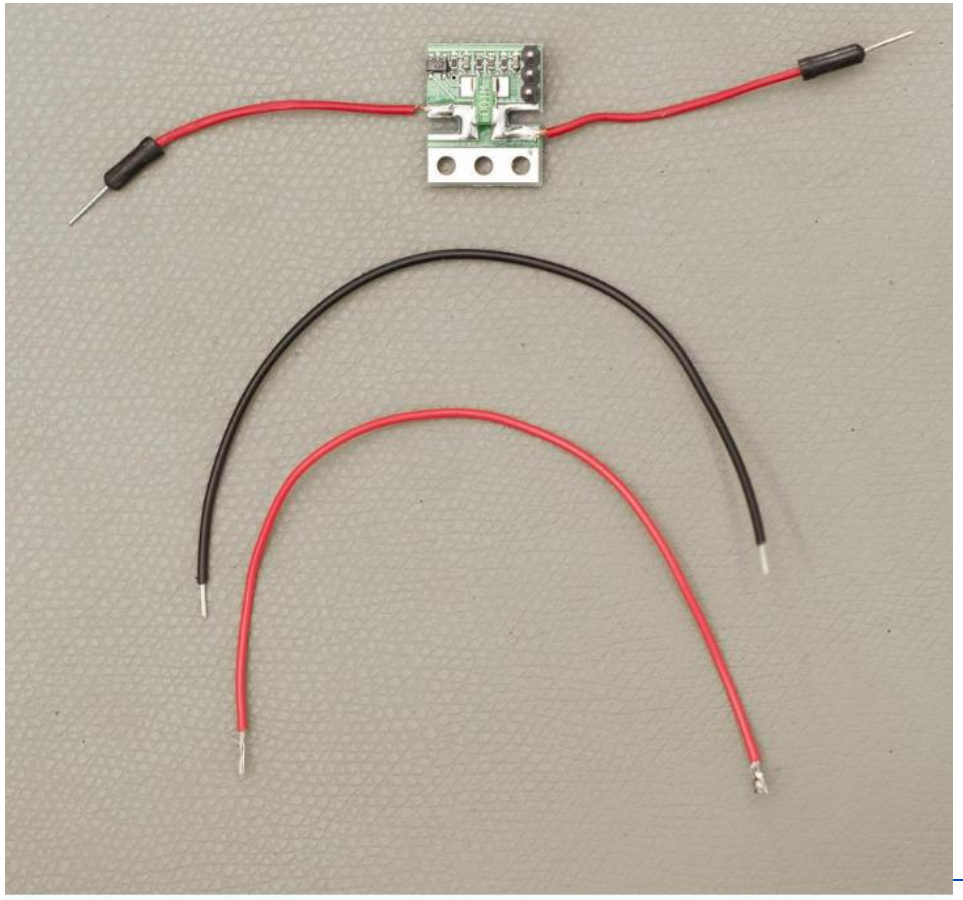


Figure 10-14. Jumper wires

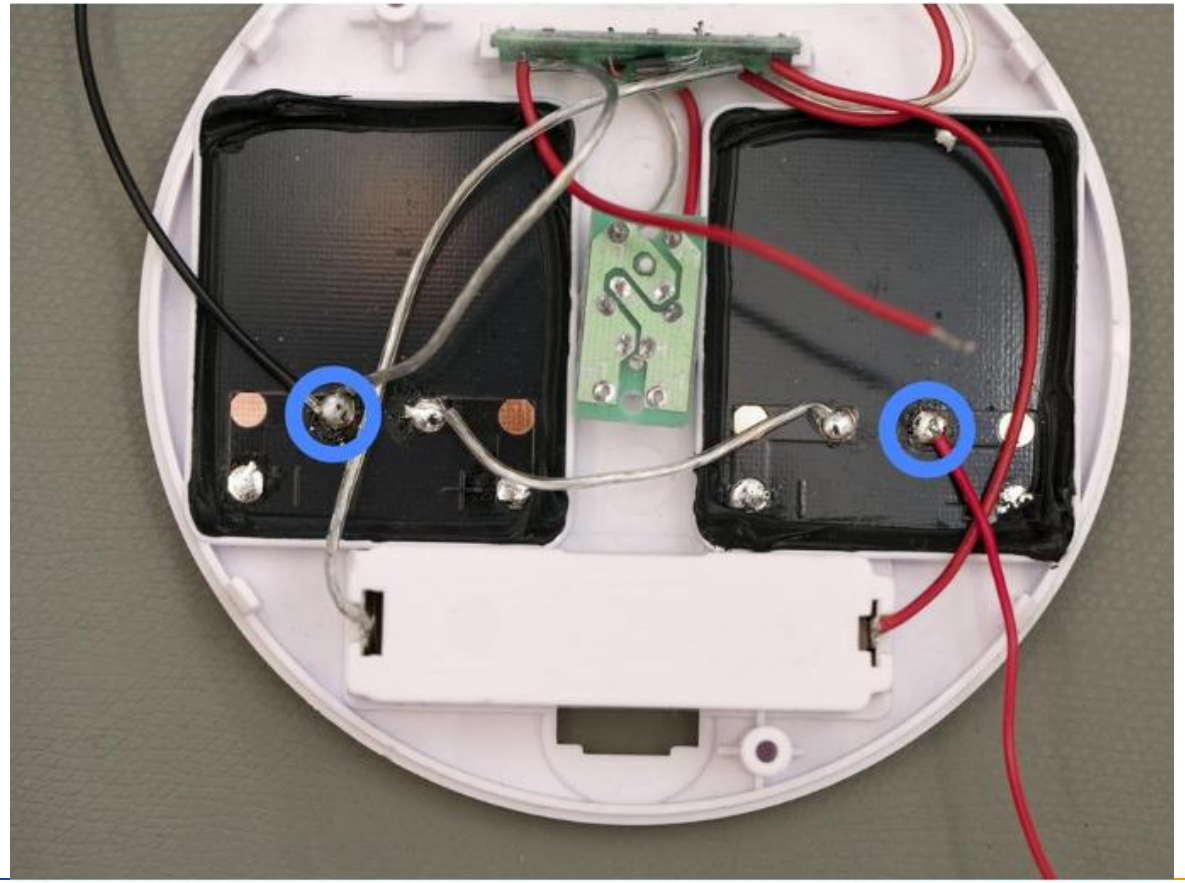


Figure 10-15. Jumper wires soldered to solar cells

Connecting Solar Cells

- Cut the jumper wires as shown in Figure 10-14 and solder them to the current sensor and to the solar cells as shown in Figure 10-15. The final product is show

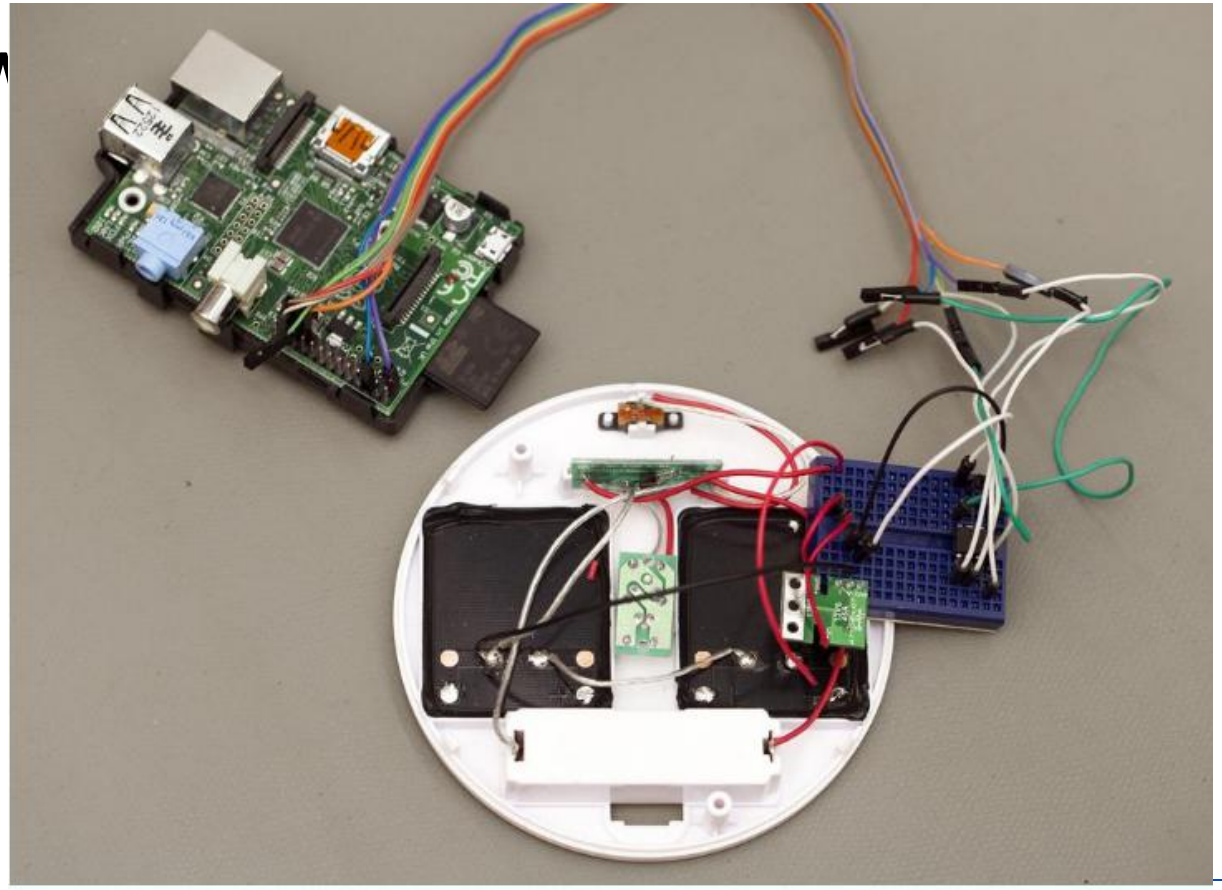
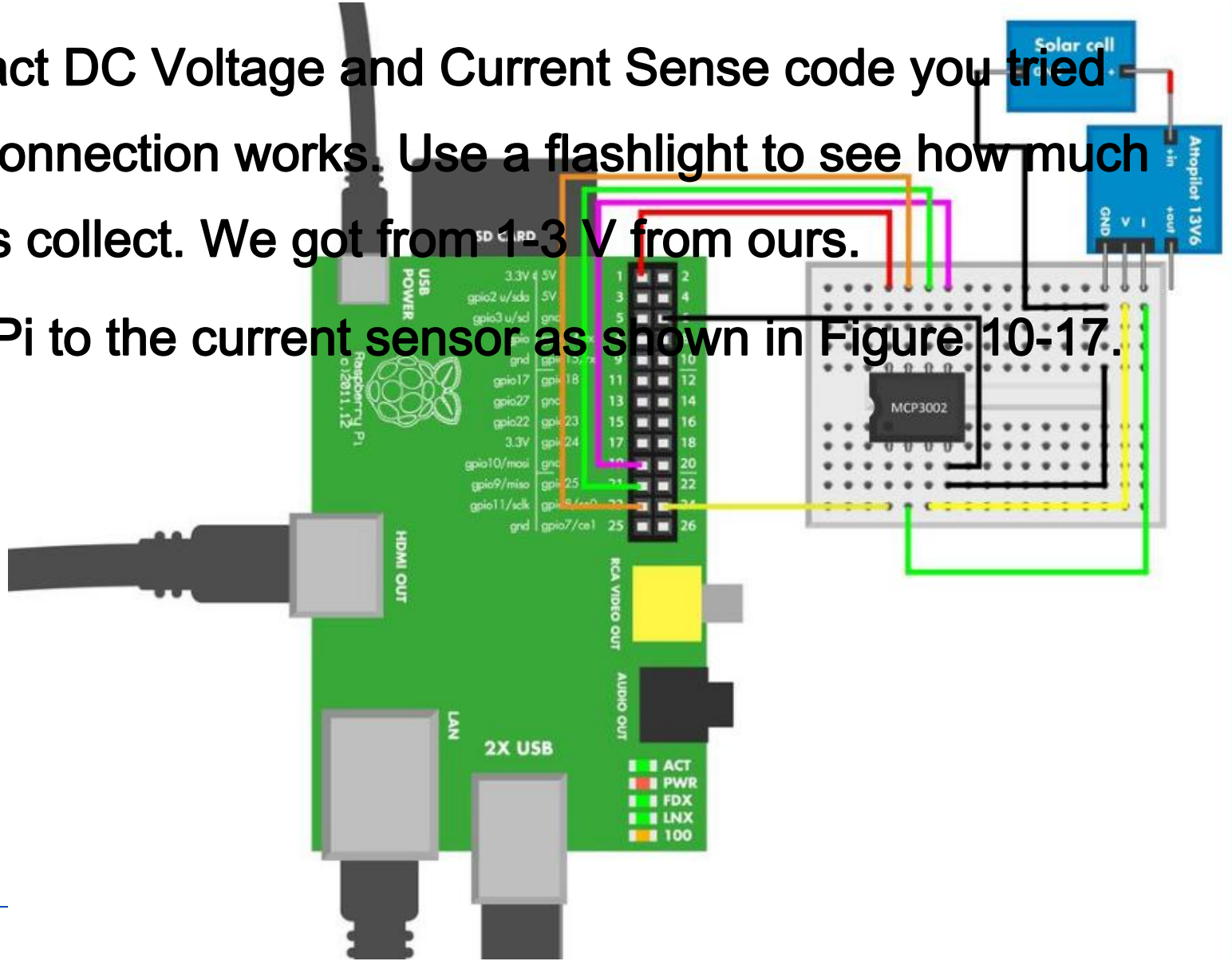


Figure 10-16. Everything connected

Connecting Solar Cells

- Use the AttoPilot Compact DC Voltage and Current Sense code you tried earlier to test that your connection works. Use a flashlight to see how much current your solar panels collect. We got from 1-3 V from ours.
- Hook up the Raspberry Pi to the current sensor as shown in Figure 10-17.



Turn Raspberry Pi into Web Server

- Apache is one of the most popular web servers on the planet.
- To turn Raspberry Pi into a web server, you must install Apache. The steps to install Apache on Raspberry Pi are exactly the same as you would use when installing Apache on a physical server, virtual server, or a development laptop.
- On your Raspbian desktop, open the command-line interface (LXTerminal; you'll find the icon on the left side of the desktop).
- Update the list of available packages, and then install Apache:
  - $ sudo apt-get update
  - $ sudo apt-get -y install apache2

Finding Your IP Address

- To access your web server from other computers, check your IP address:
  - $ ifconfig
- You can try browsing to this address, too. Just open Midori, and type “http://” and the IP address as the URL. For example, http://10.0.0.1. Obviously, you must use your own address.

Making Your Home Page on Raspberry Pi

- It's easy to set up and maintain a web page if you create it as a user home page. There are two steps: enabling users to make home pages and making the home pages.
- Allow users to make home pages. As this changes system-wide settings, you need to use sudo. The following commands enable the Apache user directory module, and then restart Apache:
  - $ sudo a2enmod userdir
  - $ sudo service apache2 restart

Making Your Home Page on Raspberry Pi

- Now it's time to create your (pi) home page directory. This doesn't require sudo privileges. The name of your home page directory, public_html, consists of the word "public", underscore ("_"), and the word "html". It must be written correctly.
  - $ cd /home/pi/
  - $ mkdir public_html
- If you want, you can make a test page:
  - $ echo "botbook">/home/pi/public_html/hello.txt
- Try it with the web browser: http://localhost/~pi/hello.txt

Solar Panel Monitor Code and Connection for Raspberry Pi

- This code requires matplotlib, a great free Python library for mathematical graphing. Install it with these commands:
  - $ sudo apt-get update
  - $ sudo apt-get -y install python-matplotlib
- Another useful tool for data visualization is Plotly.
- Running voltage_record.py once records a new data point and creates one graph, and then exits. No loop is needed, because you'll see how to use cron to run the program every five minutes.
  - cron：计划任务，是任务在约定的时间执行已经计划好的工作，这是表面的意思。在Linux中，我们经常用到cron 服务器来完成这项工作。cron服务器可以根据配置文件约定的时间来执行特定的作务。比如我们可以在配置文件中约定每天早上4点，对httpd 服务器重新启动，这就是一个计划任务;

Solar Panel Monitor Code and Connection for Raspberry Pi

- The measurement history is kept in /home/pi/record.csv.
- The generated plot file is put into /home/pi/public_html/history.png.
- Because you've already installed Apache, this file is published at the URL http://localhost/~pi/history.png.
- To visit that page from another device, you need to replace localhost with your Raspberry Pi's IP address.

Example 10-9. voltage_record.py

```
# voltage_record.py - record voltage from solar cell and print history to png
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
import time
import os
import matplotlib       #要安装必要的库文件
matplotlib.use("AGG")   # 用于存储像素图形PNG的matplotlib常用格式
import matplotlib.pyplot as plt
import numpy
from datetime import datetime
from datetime import date
from datetime import timedelta
import attopilot_voltage  #前面章节制作的案例程序（测量电压电流）存储于目录attopilot_voltage/中
import shelve

historyFile = "/home/pi/record"    #用于存储数据的文件路径，shelve函数让数据存储更简单。
plotFile = "/home/pi/public_html/history.png"   # 创建图形PNG图像的路径
```

Example 10-9. voltage_record.py

```
# voltage_record.py - record voltage from solar cell and print history to png
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
def measureNewDatapoint():
        return attopilot_voltage.readVoltage()  #实际的测量命令，如测其他物理量，修改此命令即可

def main():
        history = shelve.open(historyFile)          #打开历史文件，如果没有则创建文件。
        if not history.has_key("x"):                #如果变量还没被shelve保存过.....
                history["x"] = []           #.....那么创建它们，shelve的历史记录属于字典类型，
                history["y"] = []           #  x键和y键分别存储一个列表
        history["x"] += [datetime.now()]  #将时间戳加到x键的值后
        history["y"] += [measureNewDatapoint()]#将测量数据追加到y键的值后
        now = datetime.now()                #剔除超过24小时的数据点
        sampleCount = 24 * 60 / 5
        history['x'] = history["x"][-sampleCount:]  //取列表最后24小时的数据,如 [-x:]就是后x个数据
        history['y'] = history["y"][-sampleCount:]
```

Example 10-9. voltage_record.py

```
# voltage_record.py - record voltage from solar cell and print history to png
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

        plot = plt.figure()     #
        plt.title('Solar cell voltage over last 24h')
        plt.ylabel('Voltage V')
        plt.xlabel('Time')
        plt.setp(plt.gca().get_xticklabels(), rotation=35)
        plt.plot(history["x"], history["y"], color='#4884ff')   #
        plt.savefig(plotFile)   #
        history.close() #

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

Timed Tasks with Cron

- Cron is our favorite way of performing timed tasks. Even if your Raspberry Pi shuts down, cron tasks are resumed after reboot.
- First make sure that you can run the Python program on the command line, with full path. Use the path to wherever you have downloaded or saved voltage_record.py .
  - $ python /home/pi/voltage_record/voltage_record.py
- If it executes normally, it's time to add it as a timed task to cron. Edit your user cron file with this:
  - $ crontab -e
- Add the following as the last lines of the file，Save the file:
  - */5 * * * * /home/pi/voltage_record/voltage_record.py
  - */1 * * * * touch /tmp/botbook-cron

Timed Tasks with Cron

- The first line tells cron to run the program whenever minutes are divisible by 5 (:00, :05…), ignoring hour, day of month, month, and day of week.
- The second line just creates an empty file /tmp/botbook-cron every minute. It‘s for checking up on cron, and you can delete this line later if you want. Wait for a minute, and then check if the file is there:

```
$ ls /tmp/botbook*
/tmp/botbook-cron
```

- Did ls show your file? Well done! You successfully added a timed task to cron.
- Whenever you want Raspberry Pi to automatically do something, use cron. Even if the power is cut, the timed tasks are run again when you boot up the Raspberry Pi.

crond简介

- at 命令是针对仅运行一次的任务，循环运行的例行性计划任务，linux系统则是由 cron (crond) 这个系统服务来控制的。Linux 系统上面原本就有非常多的计划性工作，因此这个系统服务是默认启动的。另外, 由于使用者自己也可以设置计划任务，所以， Linux 系统也提供了使用者控制计划任务的命令 :crontab 命令。
- crond是linux下用来周期性的执行某种任务或等待处理某些事件的一个守护进程，与windows下的计划任务类似，当安装完成操作系统后，默认会安装此服务工具，并且会自动启动crond进程，crond进程每分钟会定期检查是否有要执行的任务，如果有要执行的任务，则自动执行该任务。
- Linux下的任务调度分为两类，系统任务调度和用户任务调度。
  - 系统任务调度：系统周期性所要执行的工作，比如写缓存数据到硬盘、日志清理等。在/etc目录下有一个crontab文件，这个就是系统任务调度的配置文件。

crond简介

- **/etc/crontab文件包括下面几行：**

```
[root@localhost ~]# cat /etc/crontab
SHELL=/bin/bash                          第一行SHELL变量指定了系统要使用哪个shell，这里是bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin       第二行PATH变量指定了系统执行命令的路径
MAILTO=""                                第三行MAILTO变量指定了crond的任务执行信息将通过电子邮件发送给root用户，
                                         如果MAILTO变量的值为空，则表示不发送任务执行信息给用户
HOME=/                                   第四行的HOME变量指定了在执行命令或者脚本时使用的主目录
# run-parts
51 * * * * root run-parts /etc/cron.hourly
24 7 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

crond简介

- **用户任务调度：用户定期要执行的工作，比如用户数据备份、定时邮件提醒等。用户可以使用 crontab 工具来定制自己的计划任务。所有用户定义的crontab 文件都被保存在 /var/spool/cron目录中。其文件名与用户名一致。**
- **用户所建立的crontab文件中，每一行都代表一项任务，每行的每个字段代表一项设置，它的格式共分为六个字段，前五段是时间设定段，第六段是要执行的命令段，格式如下：**
- minute hour day month week command **其中：**
  - minute： 表示分钟，可以是从0到59之间的任何整数。
  - hour：表示小时，可以是从0到23之间的任何整数。
  - day：表示日期，可以是从1到31之间的任何整数。
  - month：表示月份，可以是从1到12之间的任何整数。
  - week：表示星期几，可以是从0到7之间的任何整数，这里的0或7代表星期日。
  - command：要执行的命令，可以是系统命令，也可以是自己编写的脚本文件。

crond简介

- 在以上各个字段中，还可以使用以下特殊字符
- 星号（*）：代表所有可能的值，例如month字段如果是
  件后每月都执行该命令操作。
- 逗号（,）：可以用逗号隔开的值指定一个列表范围，例
- 中杠（-）：可以用整数之间的中杠表示一个整数范围，
- 正斜线（/）：可以用正斜线指定时间的间隔频率，例如
  时正斜线可以和星号一起使用，例如*/10，如果用在mi