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

Weather and Climate

Weather and Climate

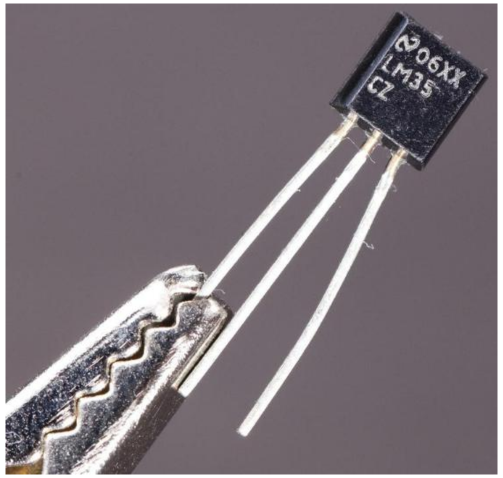
- Weather is one of the few things we humans can’t control. Yet.
- The immediately obvious components of weather include temperature and humidity.
- To forecast weather, you can measure the atmospheric pressure.

Experiment: Is It Hot in Here?

- An LM35 sensor reports temperature with varying voltage. Because it’s an analog resistance sensor, it’s easy to use with Arduino. Using it with Raspberry Pi requires an analog-to-digital converter chip.
- The LM35 has three leads (Figure 12-1), similar to a potentiometer. The output lead voltage U is converted to temperature T with a simple formula:

$T = 100*U$

温度/℃	电压/V	说明
2	0.02	最低测量温度
20	0.2	室温
100	1.0	沸水的温度
150	1.5	最高测量温度



如果要想测量零下摄氏度的温度，可以使用LM36或者查找LM35手册，用其他连接方法。

LM35 Code and Connection for Arduino

- Figure 12-2 shows the wiring diagram for Arduino. Hook it up as shown, and then run the code from Example 12-1.

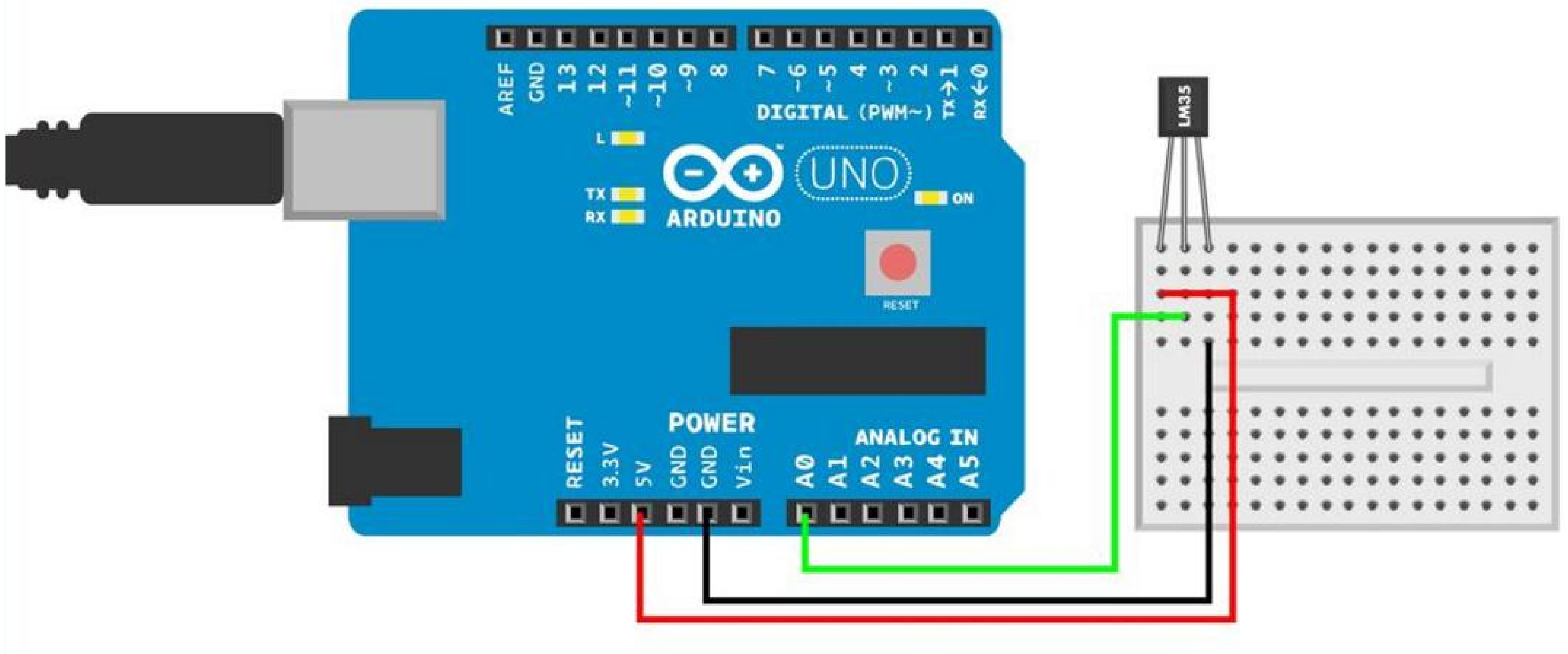


Figure 12-2. Arduino/LM35 connections

Example 12-1. temperature_lm35.ino

```
// temperature_lm35.ino - measure temperature (Celsius) with LM35 and print it
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int lmPin = A0;
void setup(){
		Serial.begin(115200);
		pinMode(lmPin, INPUT);
}

float tempC(){
		float raw = analogRead(lmPin);  //
		float percent = raw/1023.0; //
		float volts = percent*5.0; //
		return 100.0*volts;  //
}

void loop(){
		Serial.println(tempC());
		delay(200); // ms
}
```

LM35 Code and Connection for Raspberry Pi

- Figure 12-3 shows the connec diagram for the Raspberry Pi.
- Hook it up as shown, and then the code in Example 12-2.

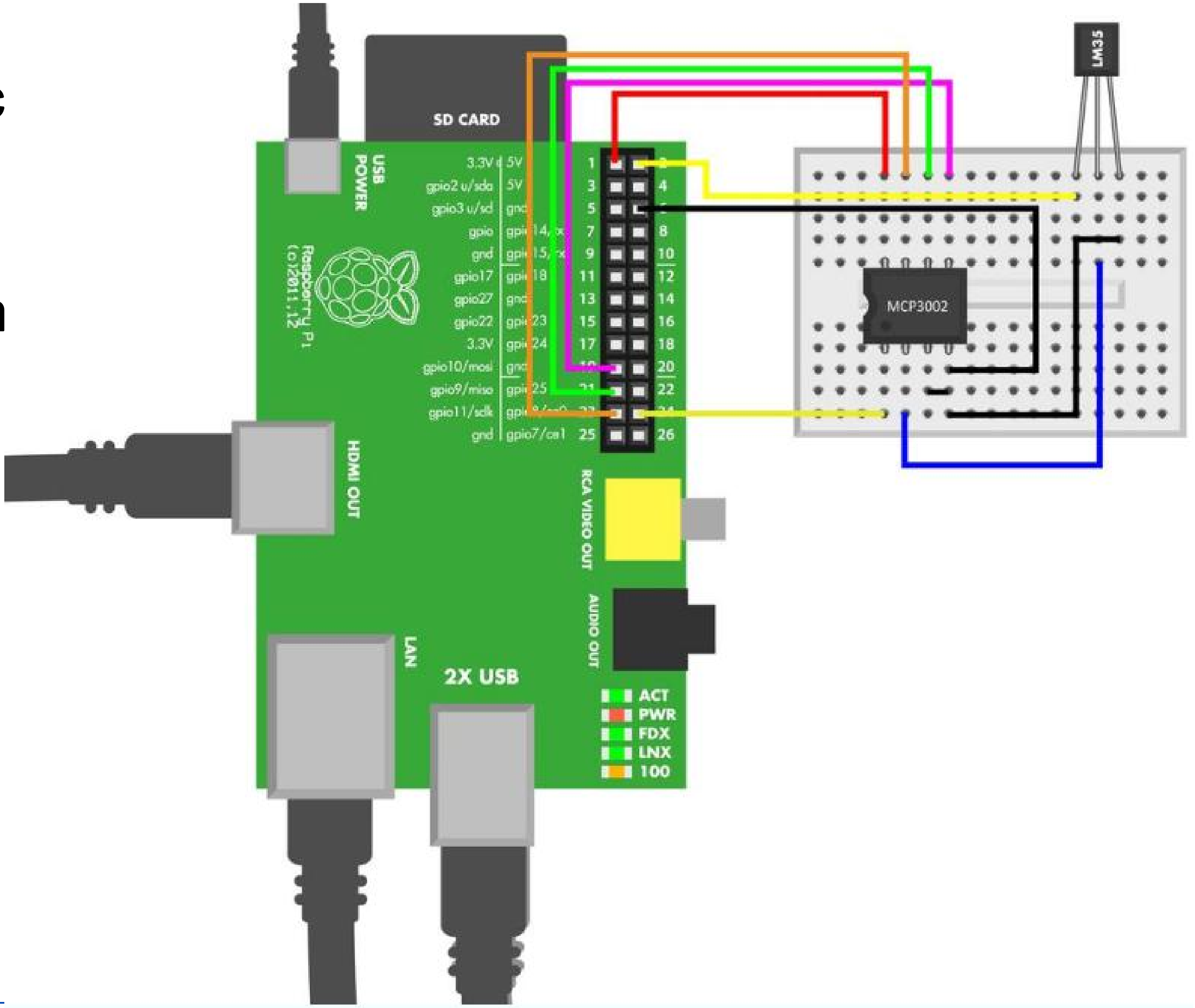


Figure 12-3. Raspberry Pi connections for the LM35

Example 12-2. temperature_lm35.py

```
# temperature_lm35.py - print temperature
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp   #
#LM35 data pin voltage keeps under 2V so no level conversion needed
def main():
	while True:
		rawData = mcp.readAnalog()	#
		percent = rawData / 1023.0	#
		volts = percent * 3.3	#
		temperature = 100.0 * volts	#
		print("Current temperature is %.1f C " % temperature)
		time.sleep(0.5)

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

Environment Experiment: Changing Temperature

- Air is a very good insulator. You’ll find that your patience will be tested if you take your LM35 temperature sensor (Figure 12-4) to the sauna, fridge, and balcony—it seems to change so slowly.
- To get quick changes, press an ice cube directly against the LM35. Avoid wetting your wires, microcontrollers, or breadboard. Now the heat is conducted away from LM35, and the change of temperature is rapid.

Environment Experiment: Changing Temperature

- Air is a very good insulator. You'll find that your patience will be tested if you take your LM35 temperature sensor (Figure 12-4) to the sauna, fridge, and balcony—it seems to change so slowly.
- To get quick changes, press an ice cube directly against the LM35. Avoid wetting your wires, microcontrollers, or breadboard. Now the heat is conducted away from LM35, and the change of temperature is rapid.

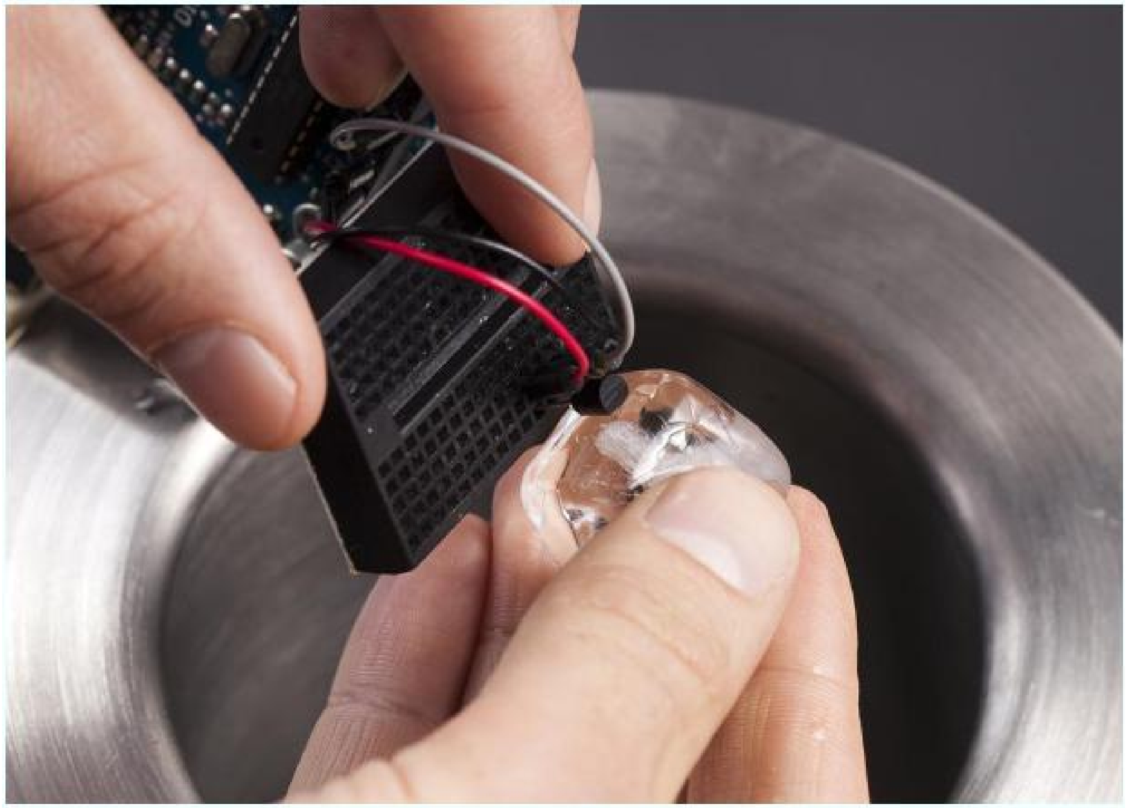


Figure 12-4. Measuring cold temperatures

Experiment: Is It Humid in Here?

- DHT11 measures humidity and temperature (Figure 12-5).

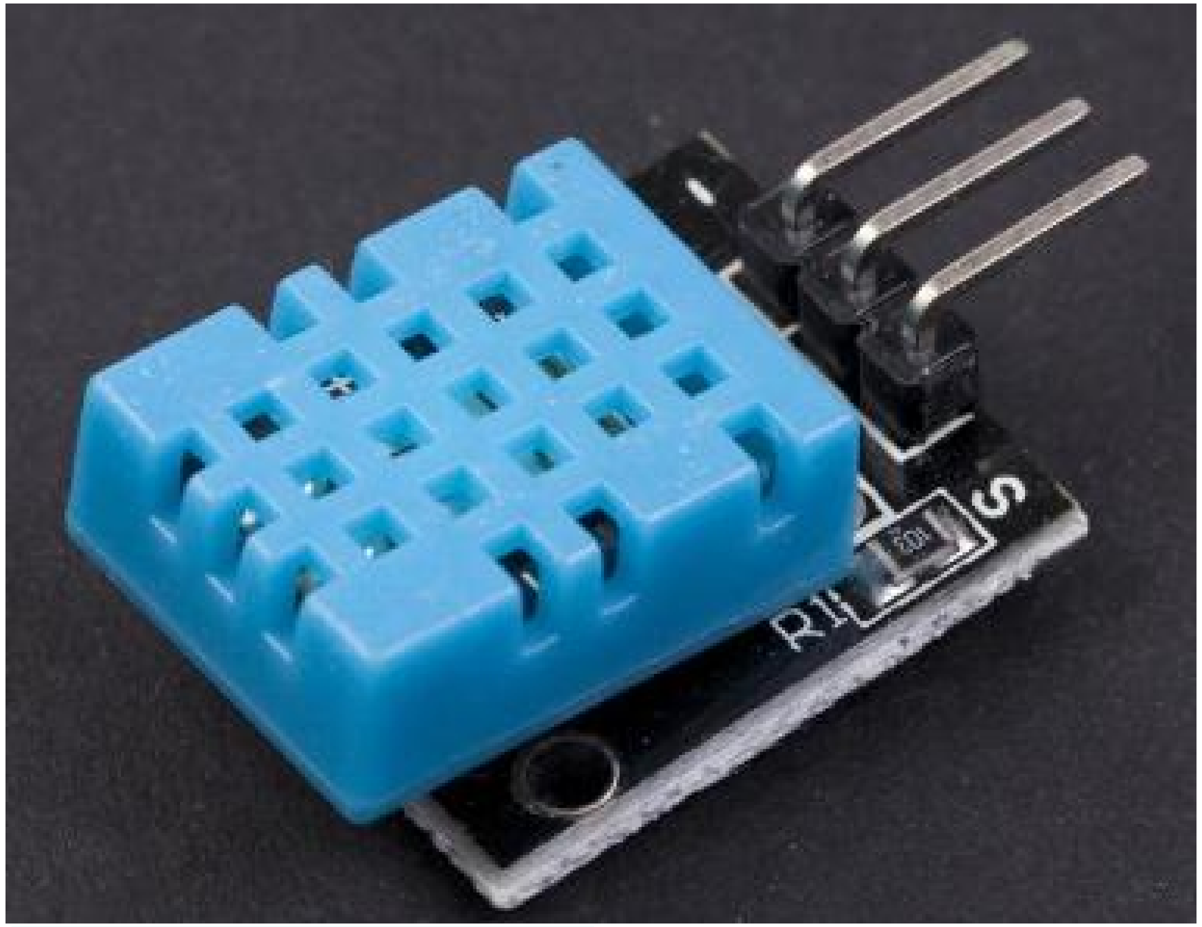


Figure 12-5. DHT11 humidity sensor

Experiment: Is It Humid in Here?

- The protocol that the DHT11 uses is weird. It sends bits as very short pulses very rapidly. Arduino doesn't have an operating system, so it's more real-time than Raspberry Pi and can easily read these pulses. But even Arduino needs to be coded in a special way, as even the built-in pulseIn() function is not fast enough.
- Raspberry Pi is less real-time and has a difficult time reading the pulses reliably. That's why you'll connect Arduino to Raspberry Pi and use Arduino to do the actual reading of the DHT11. The connection is made using serial over USB. If you want, you can easily apply this technique to any other sensor.
- The data pin is normally HIGH. To start reading, the microcontroller sends a LOW pulse of 18 milliseconds.
- The DHT11 sends 5-byte packets. As each byte is 8 bits, the packet is 5 B * 8 bit/B = 40 bits.

How Humid Is Your Breath?

- How can we change the measured humidity?
- Just buy an ultrasonic humidifier that uses a piezoelectric transducer to create mist from water, or book a flight to Thailand.
- No, wait, there is an easier way.
- You have a built-in humidifier in your lungs. Bring the sensor close to your mouth and breathe (Figure 12-6).
- Keep checking to see how values change by watching the Arduino Serial Monitor.

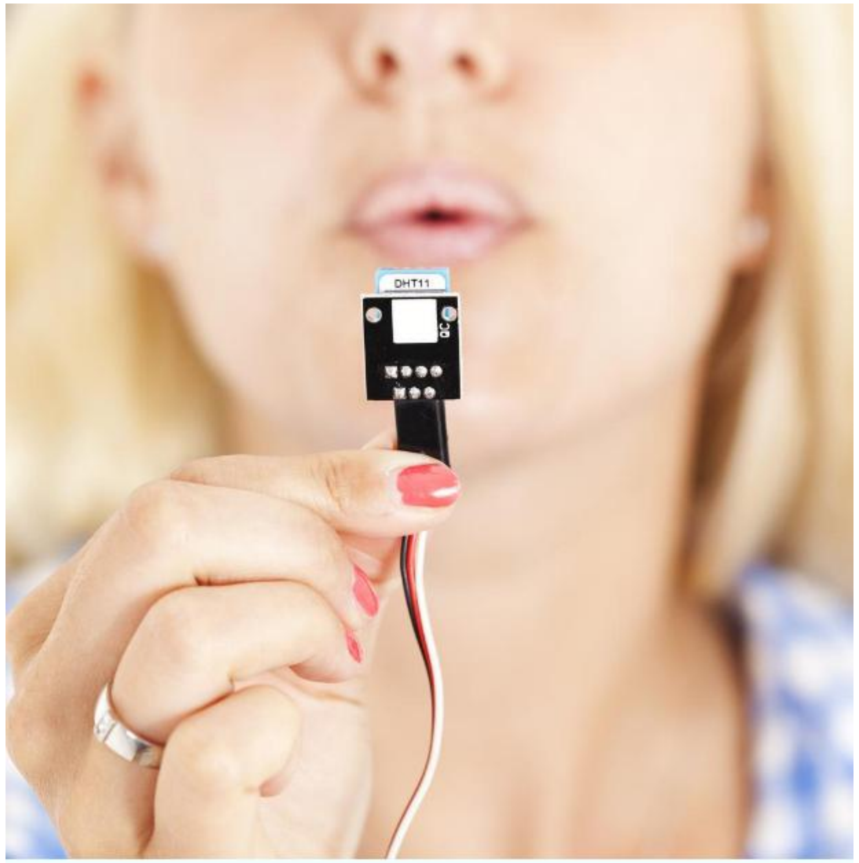


Figure 12-6. Breath to humidity sensor

DHT11 Code and Connection for Arduino

➢ Figure 12-7 shows the connections for Arduino. Wire them up as shown, and then run the code shown in Example 12-3.

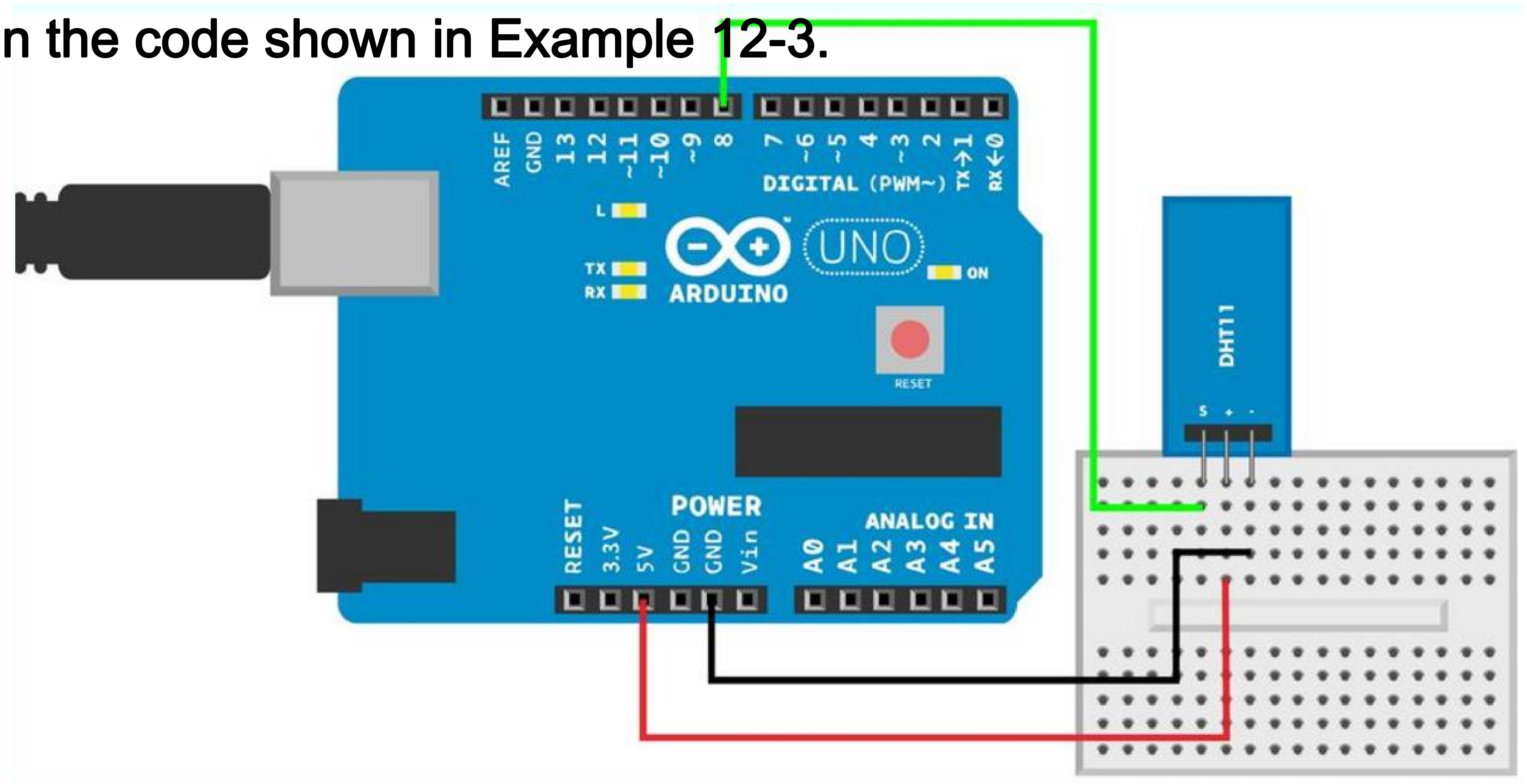


Figure 12-7. DHT11 humidity sensor connected to Arduino

Example 12-3. dht11.ino

```
// dht11.ino - print humidity and temperature using DHT11 sensor
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int dataPin = 8;
int temperature = -1;
int humidity = -1;
void setup() {
		Serial.begin(115200);
}

int readDHT11() {
		uint8_t recvBuffer[5];		//
		uint8_t cnt = 7;
		uint8_t idx = 0;
		for(int i = 0; i < 5; i++) recvBuffer[i] = 0; //
 // Start communications with LOW pulse
		pinMode(dataPin, OUTPUT);
		digitalWrite(dataPin, LOW);
		delay(18);
		digitalWrite(dataPin, HIGH);
```

Example 12-3. dht11.ino

// dht11.ino - print humidity and temperature using DHT11 sensor
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
        delayMicroseconds(40);      //
        pinMode(dataPin, INPUT);
        pulseIn(dataPin, HIGH);  //
// Read data packet
        unsigned int timeout = 10000; // loop iterations
        for (int i=0; i<40; i++)      //  {
                timeout = 10000;
                while(digitalRead(dataPin) == LOW) {
                        if (timeout == 0) return -1;
                         timeout--;
                }
                unsigned long t = micros();   //

                timeout = 10000;
                while(digitalRead(dataPin) == HIGH) { //
                        if (timeout == 0) return -1;
                        timeout--;
                 }
```

Example 12-3. dht11.ino

// dht11.ino - print humidity and temperature using DHT11 sensor
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
                if ((micros() - t) > 40) recvBuffer[idx] |= (1 << cnt);     //
                if (cnt == 0)   // next byte?
                {
                        cnt = 7;    // restart at MSB
                        idx++;      // next byte!
                 }
                else cnt--;
        }
        humidity = recvBuffer[0];     // %    //
        temperature = recvBuffer[2];  // C
        uint8_t sum = recvBuffer[0] + recvBuffer[2];
        if(recvBuffer[4] != sum) return -2;   //
        return 0;
}
```

Example 12-3. dht11.ino

```
// dht11.ino - print humidity and temperature using DHT11 sensor
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void loop() {
        int ret = readDHT11();
        if(ret != 0) Serial.println(ret);
        Serial.print("Humidity: ");
        Serial.print(humidity);
        Serial.println(" %");
        Serial.print("Temperature: ");
        Serial.print(temperature);
        Serial.println(" C");
        delay(2000); // ms
}
```

DHT11 Code and Connection for Raspberry Pi

- Would you like to combine Arduino and Raspberry Pi to get the best of both worlds? We hope so, because DHT11 doesn’t like the Raspberry Pi. This code shows you how you can read data from Arduino, in a way you can easily adapt to other projects.
- Even if it’s possible to use DHT11 from Raspberry Pi with complicated code, the result is not satisfactory. Luckily, Arduino can be used for fast low-level tasks, letting Raspberry Pi concentrate on the big picture.
- After you have tried the code, you should read Talking to Arduino from Raspberry Pi.

DHT11 Code and Connection for Raspberry Pi

- To use the serial port in Raspberry Pi, you must first release it from its default use as a login terminal. See Enabling the Serial Port in Raspberry Pi for instructions.
- Wire them up as shown in Figure 12-8, and then run the code shown in Example 12-4.

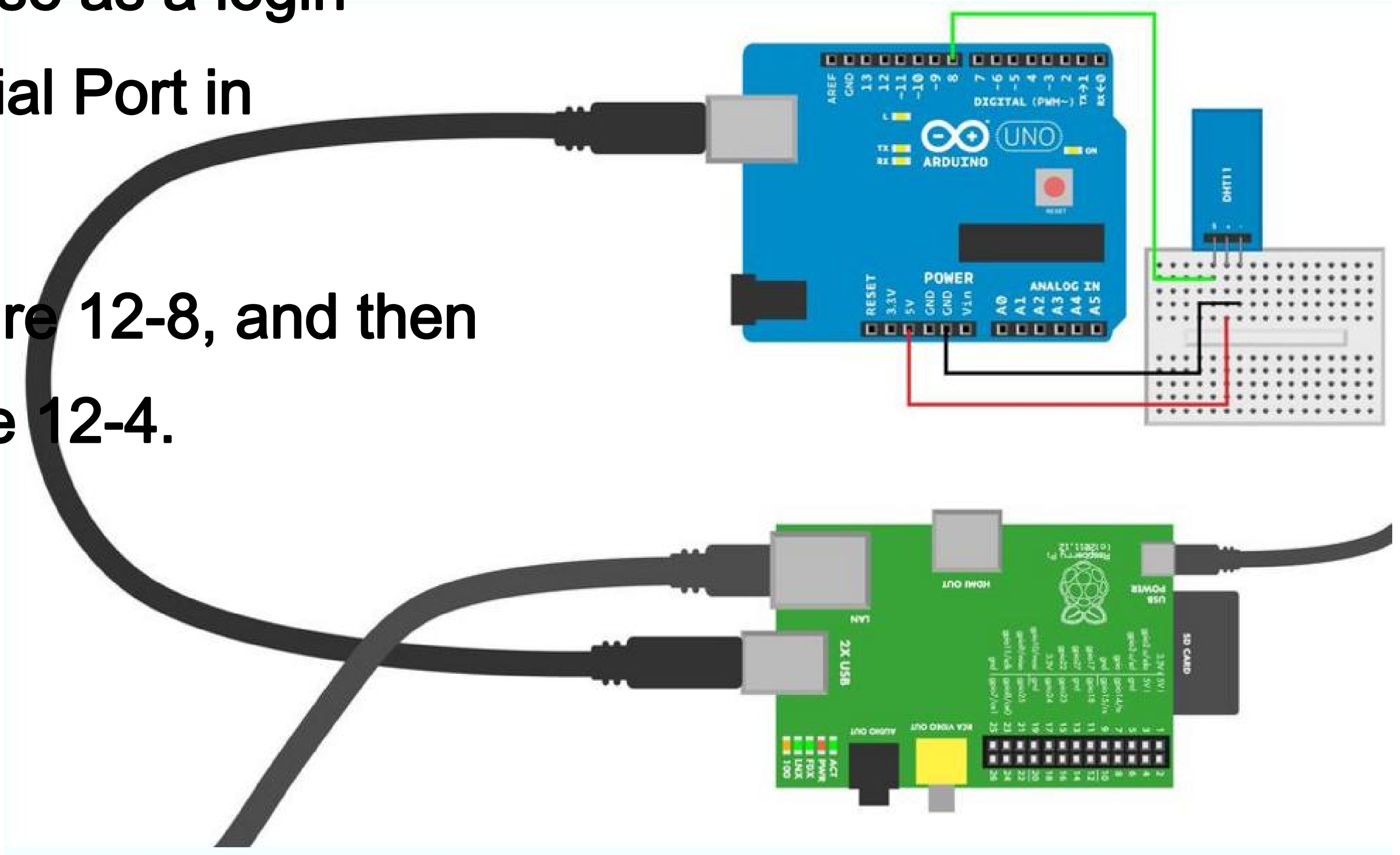


Figure 12-8. Arduino connected to Raspberry Pi

Example 12-4. dht11_serial.py

```
# dht11_serial.py - print humidity and temperature using DHT11 sensor
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import serial   #
def main():
        port = serial.Serial("/dev/ttyACM0", baudrate=115200, timeout=None)     #
        while True:
                line = port.readline()  #
                arr = line.split()      #
                if len(arr) < 3:        #
                        continue        #
                dataType = arr[2]
                data = float(arr[1])    #
                if dataType == '%':
                        print("Humidity: %.1f %%" % data)
                else:
                        print("Temperature: %.1f C" % data)
                time.sleep(0.01)
if __name__ == "__main__":
        main()
```

Talking to Arduino from Raspberry Pi

- Arduino is real-time, robust, and simple. Also, Arduino has a built-in ADC (analog-to-digital converter). Raspberry Pi is high level, and runs a whole Linux operating system. Why not combine the benefits?
- You can use Arduino to directly interact with sensors and outputs, and control Arduino from Raspberry Pi.
- First, make the sensors work with Arduino, and print the readings to serial. Use your normal desktop computer and Arduino IDE for this. You can write to serial with Serial.println(). Use the Serial Monitor (Tools→Serial Monitor) to see what your program prints. Only proceed to the Raspberry Pi part once you are happy with how the Arduino part of your project works.

Talking to Arduino from Raspberry Pi

- The connection between Arduino and Raspberry Pi is simple: just connect them with a USB cable.
- For a more intimate combination of the Arduino platform and Raspberry Pi, check out the AlaMode for Raspberry Pi, which incorporates an Arduino-compatible microcontroller board into a Raspberry Pi expansion that snaps right on top of your Pi. There's no need to connect them by USB, since they use the Raspberry Pi's expansion header.
- Arduino communicates with serial over USB. To read serial from Raspberry Pi, use Python and PySerial. For a complete example of using PySerial, see Figure 12-8 or Chapter 7 of Make: Arduino Bots and Gadgets.

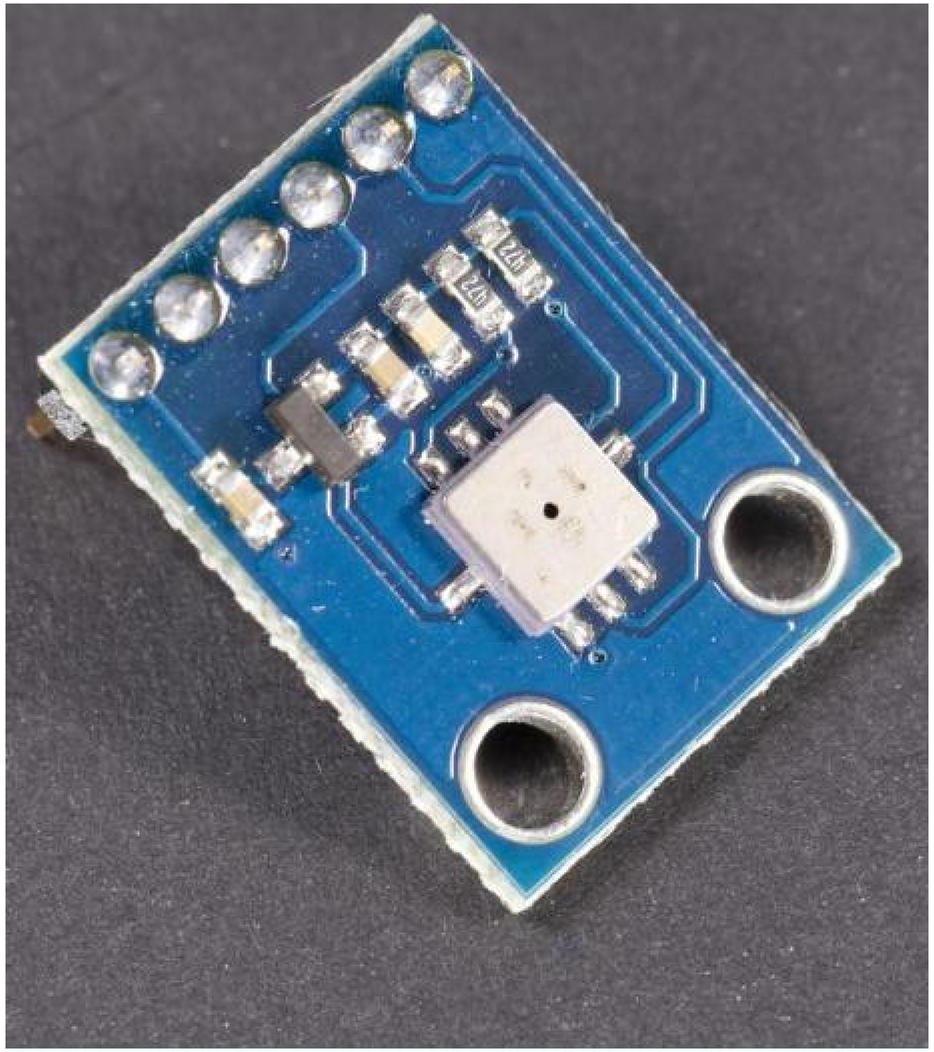
Talking to Arduino from Raspberry Pi

- You’ll have Arduino print normal text, then extract the numbers with Python’s string handling. In many prototyping applications, you don’t have to design a new low-level protocol.
- For example, consider an Arduino program printing the following:
  - Humidity: 83 %
- If you have read this into a Python string, use split() to separate the words into a list:

```
>>> s="Humidity: 83 %“
>>> s.split()
['Humidity:', '83', '%']
```

Atmospheric Pressure GY65

- Atmospheric pressure can help you forecast weather.
- High pressure means sunny, clear weather.
- Low pressure means rainy, snowy, or cloudy weather.



Y65 barometric pressure sensor

Atmospheric Pressure GY65

- Normal air pressure is about 1,000 hPa (hectopascals). The exact value depends on your altitude.

  1,000 hPa = 100,000 Pa = 1,000 mbar = 1 bar

- The pressure doesn’t vary much, even in extreme weather. The world records are a high of 1,084 hPa and a low of 870 hPa (inside a typhoon). Typical changes in normal weather are a few hectopascals.

- Pressure changes in the lower atmosphere are tiny compared with other everyday pressure changes. A hand-pumped bike tire has pressure of 4,000 hPa. Outside a passenger airplane over 30,000 feet, the pressure is less than 300 hPa.

Atmospheric Pressure GY65

- So what level of atmospheric pressure is considered high, then? Meteorologists say that high pressure is just relatively high, compared with the pressure of surrounding areas.
- You can still predict weather with pressure over time. If pressure goes up, better weather is coming. The faster it changes, the sunnier it will be.
- For a simpler forecast, you can compare current pressure to expected pressure for your altitude. You can find out your altitude with a GPS or Google Maps. According to SparkFun, the distributor of the GY65 module we used, a difference of +2.5 hPa means great, sunny weather. Conversely, low pressure of 2.5 hPa under the normal pressure means bad, rainy weather.
- GY65 is a breakout board for the BMP085 sensor. You can find the data sheet by searching for “BMP085 digital pressure sensor data sheet.”

GY65 Code and Connection for Arduino

- Figure 12-10 shows the connection diagram for Arduino. Hook it up as shown, and then run the code shown in Example 12-5.

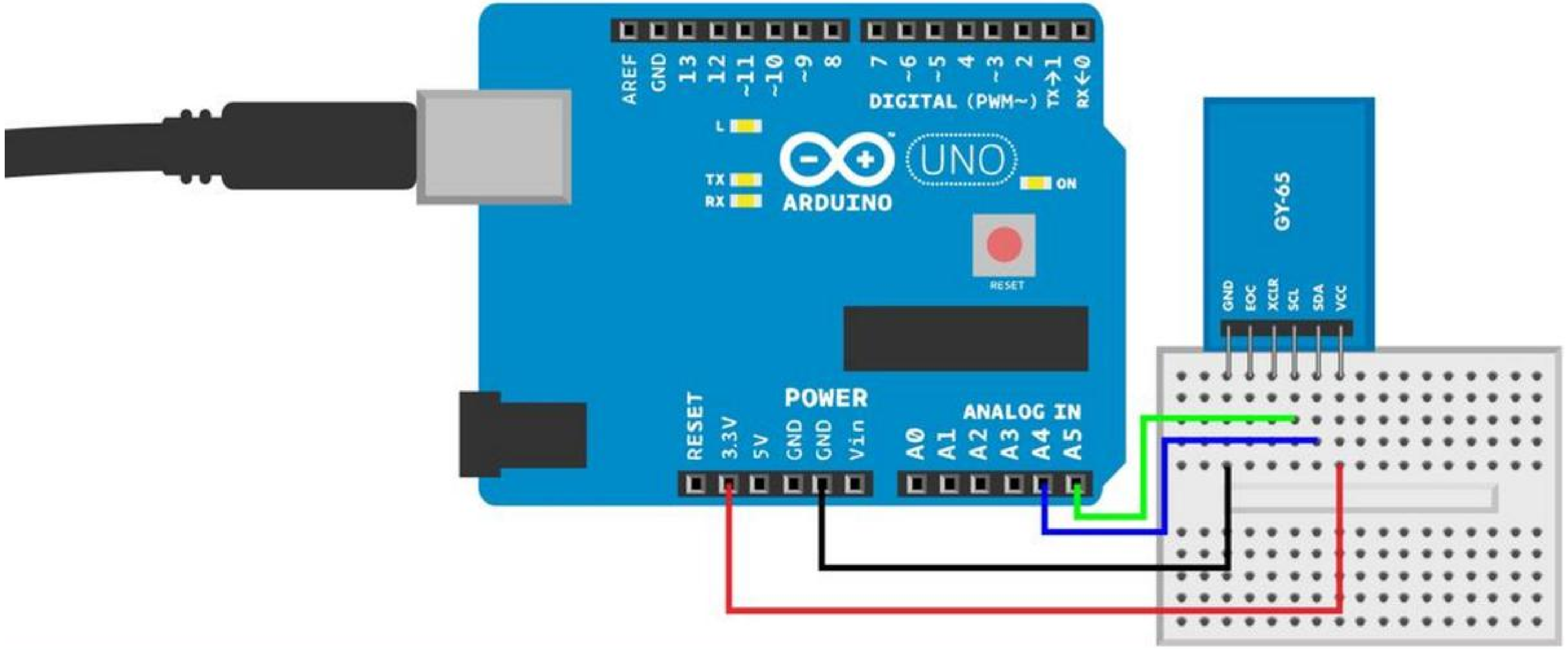


Figure 12-10. GY65 atmospheric pressure sensor connected to Arduino

Example 12-5. gy_65.ino

```
// gy_65.ino - print altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h>#
include <gy_65.h>      //

 void setup() {
        Serial.begin(115200);
        readCalibrationData(); //
}

void loop() {
        float temp = readTemperature();
        float pressure = readPressure();      //
        float altitude = calculateAltitude(pressure); //
        Serial.print("Altitude: "); Serial.print(altitude,2);   Serial.println(" m");
        Serial.print("Pressure: ");          Serial.print(pressure,2);          Serial.println(" Pa");
        Serial.print("Temperature: "); Serial.print(temp,2);          Serial.println("C");
        delay(1000);
}
```

Using Arduino Libraries

- If you have a lot of code, it’s best to split it into multiple files. The Arduino code for GY65 is such code. Also, you’ll use the same code again in Test Project: E-paper Weather Forecast.
- The library is in a folder, within the same directory as the main program (you will probably also encounter libraries that need to be installed into your Arduino sketch folder’s libraries subdirectory). Here is the folder structure:

```
gy_65.ino           # the main program
gy_65/              # folder that contains the library
gy_65/gy_65.cpp     # code for the library
gy_65/gy_65.h       # prototypes of each library function
```

Using Arduino Libraries

- The location of the main program, gy_65.ino, should seem normal to you. Every Arduino project has one main program.
- The library is in its own folder. The code for the library is in gy_65.cpp. This code looks similar to other Arduino code you have seen.
- The header file, gy_65.h, contains the prototypes of the functions in the cpp file. Prototypes are just copies of the first line of each function in gy_65.cpp. For example, the header file has the line:

  ```
  float readTemperature();
  ```

GY65 Arduino Library Explained

- You can use the GY65 without going through the implementation details of the library. But if you want a detailed understanding of how communication with the GY65 works, read on.
- You can learn the communication protocol by studying the library itself, the Raspberry Pi program (Figure 12-11), or both.
- This section also introduces you to creating your own C++ libraries for Arduino. You should also see Arduino's documentation on Writing a Library for Arduino.
- The header file gy_65.h (see Example 12-6) has prototypes for each function. It's simply a list of functions in the cpp file, which is where you'll find the actual implementation of those functions.

GY65 Arduino Library Explained

Example 12-6. gy_65.h

```
// gy_65.h - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void readCalibrationData();
float readTemperature();
float readPressure();
float calculateAltitude(float pressure);
```

GY65 Arduino Library Explained

- The implementation of the definitions from the h file are in a cpp file, gy_65.cpp (Example 12-7). If some parts of this code seem demanding, you might want to review the I2C code explanation in Figure 8-5, as well as Hexadecimal, Binary, and Other Numbering Systems and Bitwise Operations.

Example 12-7. gy_65.py

```
// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Arduino.h>
#include <Wire.h>
#include "gy_65.h“
const char i2c_address = 0x77;
int OSS = 0; // Oversampling
const long atmosphereSeaLevel = 101325; // Pa
```

Example 12-7. gy_65.py

```
// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

struct calibration_data //
{
		int16_t ac1;
		int16_t ac2;
		int16_t ac3;
		int16_t ac4;
		int16_t ac5;
		int16_t ac6;
		int16_t b1;
		int16_t b2;
		int16_t mb;
		int16_t mc;
		int16_t md;
};

calibration_data caldata;
long b5;
```

Example 12-7. gy_65.py

```
// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int16_t swap_int16_t(int16_t value)     //
{
        int16_t left = value << 8;
        int16_t right = value >> 8;
        right = right & 0xFF;
        return left | right ;
}

unsigned char read_i2c_unsigned_char(unsigned char address)     //
{
        unsigned char data;
        Wire.beginTransmission(i2c_address);
        Wire.write(address);
        Wire.endTransmission();
        Wire.requestFrom(i2c_address,1);
        while(!Wire.available());
                return Wire.read();
}
```

Example 12-7. gy_65.py

```
// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void read_i2c(unsigned char point, uint8_t *buffer, int size)
{
	Wire.beginTransmission(i2c_address);
	Wire.write(point);
	Wire.endTransmission();

	Wire.requestFrom(i2c_address,size);

	int i = 0;
	while(Wire.available() && i < size) {
		buffer[i] = Wire.read();
		i++;
	}
	if(i != size) {
		Serial.println("Error reading from i2c");
	}
}
```

Example 12-7. gy_65.py

```
// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

int read_i2c_int(unsigned char address) {
	int16_t data;
	read_i2c(address,(uint8_t *)&data,sizeof(int16_t));
	data = swap_int16_t(data);
	return data;
}

void readCalibrationData()	// {
	Wire.begin();
	read_i2c(0xAA,(uint8_t *)&caldata,sizeof(calibration_data)); //

	uint16_t *p = (uint16_t*)&caldata;	//
	for(int i = 0; i < 11; i++) { //
		p[i] = swap_int16_t(p[i]);
	}
}
```

Example 12-7. gy_65.py

// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
float readTemperature() {       //
// Read raw temperature
            Wire.beginTransmission(i2c_address);
            Wire.write(0xF4); // Register
            Wire.write(0x2E); // Value
            Wire.endTransmission();
            delay(5); //
            unsigned int rawTemp = read_i2c_int(0xF6);
// Calculate true temperature
            long x1,x2;  float t;
            x1 = (((long)rawTemp - (long)caldata.ac6) * (long)caldata.ac5) / pow(2,15);
            long mc = caldata.mc;  int md = caldata.md;
            x2 = (mc * pow(2,11)) / (x1 + md);
            b5 = x1 + x2;
            t = (b5 + 8) / pow(2,4);
            t = t / 10;
            return t;     // Celsius
}
```

Example 12-7. gy_65.py

```
// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

long getRealPressure(unsigned long up){ //
	long x1, x2, x3, b3, b6, p;
	unsigned long b4, b7;
	int b1 = caldata.b1;
	int b2 = caldata.b2;
	long ac1 = caldata.ac1;
	int ac2 = caldata.ac2;
	int ac3 = caldata.ac3;
	unsigned int ac4 = caldata.ac4;

	b6 = b5 - 4000;
	x1 = (b2 * (b6 * b6) / pow(2,12)) / pow(2,11);
	x2 = (ac2 * b6) / pow(2,11);
	x3 = x1 + x2;
```

Example 12-7. gy_65.py

```
// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
        b3 = (((ac1*4 + x3) << OSS) + 2) / 4;
        x1 = (ac3 * b6) / pow(2,13);
        x2 = (b1 * ((b6 * b6) / pow(2,12)) ) / pow(2,16);
        x3 = ((x1 + x2) + 2) / 4;
        b4 = (ac4 * (unsigned long)(x3 + 32768) ) / pow(2,15);

        b7 = ((unsigned long)up - b3) * (50000 >> OSS);
        if (b7 < 0x80000000)  p = ( b7 * 2 ) / b4;
        else p = (b7 / b4) * 2;

        x1 = (p / pow(2,8)) * (p / pow(2,8));
        x1 = (x1 * 3038) / pow(2,16);
        x2 = (-7357 * p) / pow(2,16);
        p += (x1 + x2 + 3791) / pow(2,4);

        long temp = p;
        return temp;
}
```

Example 12-7. gy_65.py

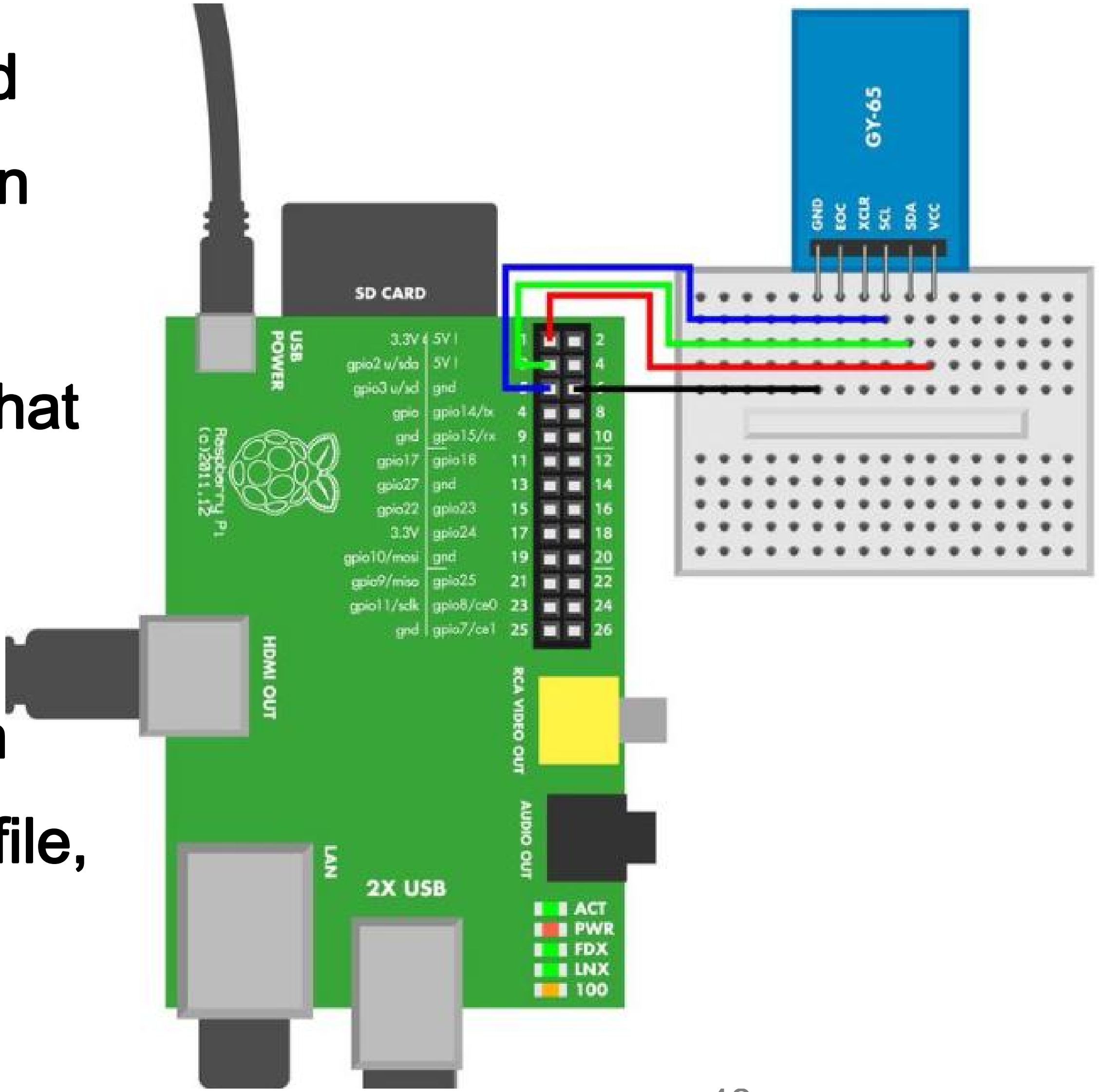
```
// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

float readPressure() {  //
// Read uncompensated pressure
        Wire.beginTransmission(i2c_address);
        Wire.write(0xF4); // Register
        Wire.write(0x34 + (OSS << 6)); // Value with oversampling setting.
        Wire.endTransmission();
        delay(2 + (3 << OSS));

        unsigned char msb,lsb,xlsb;
        unsigned long rawPressure = 0;
        msb = read_i2c_unsigned_char(0xF6);  lsb = read_i2c_unsigned_char(0xF7);
        xlsb = read_i2c_unsigned_char(0xF8);

        rawPressure = (((unsigned long) msb << 16) | ((unsigned long) lsb << 8) |
                        (unsigned long) xlsb) >> (8-OSS);
        return getRealPressure(rawPressure);
}
```

Example 12-7. gy_65.py

```
// gy_65.cpp - library for altitude, pressure and temperature with GY-65 BMP085
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

float calculateAltitude(float pressure) {       //
        float pressurePart = pressure / atmosphereSeaLevel;
        float power = 1 / 5.255;
        float result = 1 - pow(pressurePart, power);
        float altitude = 44330*result;
        return altitude; // m
}
```

GY65 Code and Connection for Raspberry Pi

- This Raspberry Pi code is easy to use, and you should try running it before reading it in detail.
- If you go through it line by line, you'll see that the code can be quite demanding to understand.
- With Raspberry Pi, the I2C communication with the GY65 sensor is not in a separate file, so the code is longer than the Arduino example.



Example 12-8. gy_65.py

```
# gy_65.py - print altitude,pressure and temperature to serial
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import smbus # sudo apt-get -y install python-smbus     #
import time
import struct

bus = None
address = 0x77
caldata = None

atmosphereSeaLevel = 101325.0
OSS = 0
b5 = 0
```

Example 12-8. gy_65.py

```
# gy_65.py - print altitude,pressure and temperature to serial
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def readCalibrationData():	#
	global bus, caldata
	bus = smbus.SMBus(1)
	rawData = ""

	for i in range(22):
		rawData += chr(bus.read_byte_data(address, 0xAA+i)) #
	caldata = struct.unpack('>hhhhhhhhhhh', rawData) #
```

Example 12-8. gy_65.py

```
# gy_65.py - print altitude,pressure and temperature to serial
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def readTemperature():
	global b5
	bus.write_byte_data(address, 0xF4, 0x2E)	#
	time.sleep(0.005)	#
	rawTemp = bus.read_byte_data(address, 0xF6) << 8	#
	rawTemp = rawTemp | bus.read_byte_data(address, 0xF7)
	x1 = ((rawTemp - caldata[5]) * caldata[4]) / 2**15
	x2 = (caldata[9] * 2**11) / (x1 + caldata[10])
	b5 = x1 + x2
	temp = (b5 + 8) / 2**4
	temp = temp / 10.0
	return temp
```

Example 12-8. gy_65.py

```
# gy_65.py - print altitude,pressure and temperature to serial
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def readPressure():     #
            bus.write_byte_data(address, 0xF4, 0x34 + (OSS << 6))
            time.sleep(0.005)
            rawPressure = bus.read_byte_data(address, 0xF6) << 16
            rawPressure = rawPressure | bus.read_byte_data(address, 0xF7) << 8
            rawPressure = rawPressure | bus.read_byte_data(address, 0xF8)
            rawPressure = rawPressure >> (8 - OSS)
#Calculate real pressure
            b6 = b5 - 4000
            x1 = (caldata[7] * ((b6 * b6) / 2**12 )) / 2**11
            x2 = caldata[1] * b6 / 2**11
            x3 = x1 + x2
            b3 = (((caldata[0] * 4 + x3) << OSS) + 2) / 4
            x1 = caldata[2] * b6 / 2**13
            x2 = (caldata[6] * ((b6 * b6) / 2**12 )) / 2**16
            x3 = ((x1 + x2) + 2) / 2*2
```

Example 12-8. gy_65.py

```
# gy_65.py - print altitude,pressure and temperature to serial
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
b4 = (caldata[3] * (x3 + 32768)) / 2**15
b4 = b4 + 2**16 # convert from signed to unsigned
b7 = (rawPressure - b3) * (50000 >> OSS)
if b7 < 0x80000000:
        p = (b7 * 2) / b4
else:
        p = (b7 / b4) * 2
x1 = (p / 2**8) * (p / 2**8)
x1 = (x1 * 3038) / 2**16
x2 = (-7357 * p) / 2**16
p = p + (x1 + x2 + 3791) / 2**4
return p
```

Example 12-8. gy_65.py

```
# gy_65.py - print altitude,pressure and temperature to serial
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def calculateAltitude(pressure):        #
        pressurePart = pressure / atmosphereSeaLevel;
        power = 1 / 5.255;
        result = 1 - pressurePart**power;
        altitude = 44330*result;
        return altitude
def main():
        readCalibrationData()
        while True:
                temperature = readTemperature()
                pressure = readPressure()
                altitude = calculateAltitude(pressure)
                print("Altitude %.2f m" % altitude)
                print("Pressure %.2f Pa" % pressure)        #
                print("Temperature %.2f C" % temperature)
                time.sleep(10)
if __name__ == "__main__":
        main()
```

Experiment: Does Your Plant Need Watering? (Build a Soil Humidity Sensor)

- A soil humidity sensor is a simple analog resistance sensor. Stick it in the soil to see if your plant needs watering.
- Normal tap water and groundwater contain diluted salts and other material. This makes water conductive. The soil humidity sensor (Figure 12-12) simply measures that conductivity.

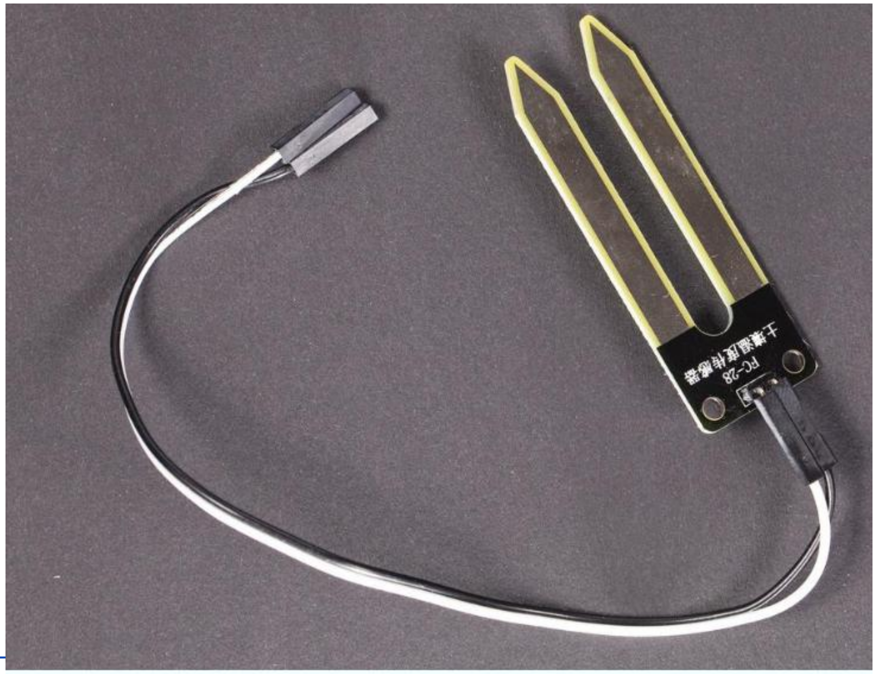


Figure 12-12. Soil humidity sensor

Experiment: Does Your Plant Need Watering? (Build a Soil Humidity Sensor)

- Some soil humidity sensors have a built-in circuit. The sensor used here doesn’t have built-in electronics, so you could make your own from two pieces of metal (for the sensor probes) and use Arduino or Raspberry Pi to measure the resistance. The circuit uses a 1 megohm resistor (brown-black-green).

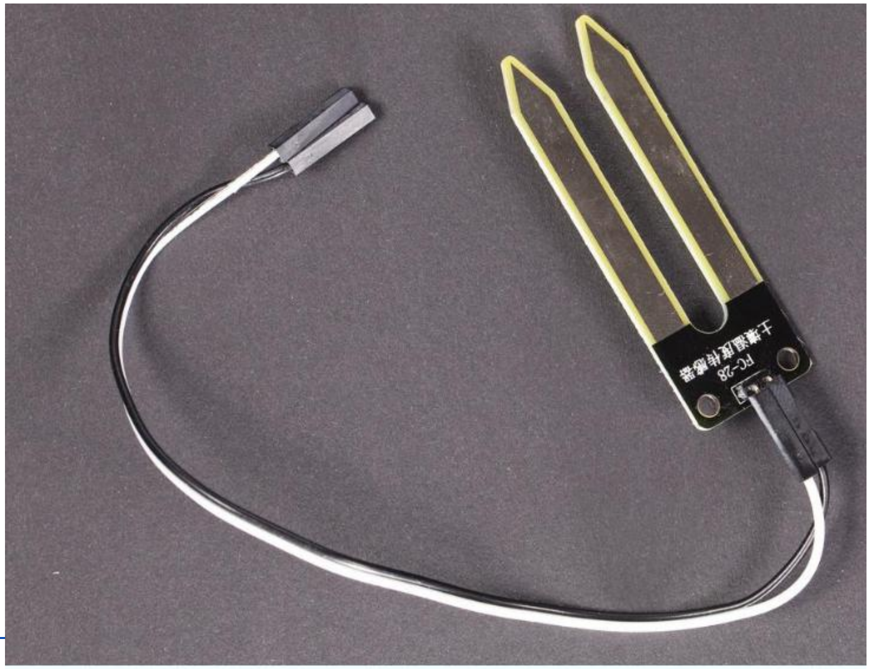


Figure 12-12. Soil humidity sensor

Soil Sensor Code and Connection for Arduino

- Figure 12-13 shows the wiring diagram for Arduino. Hook it up as shown, and then run the code from Example 12-9.

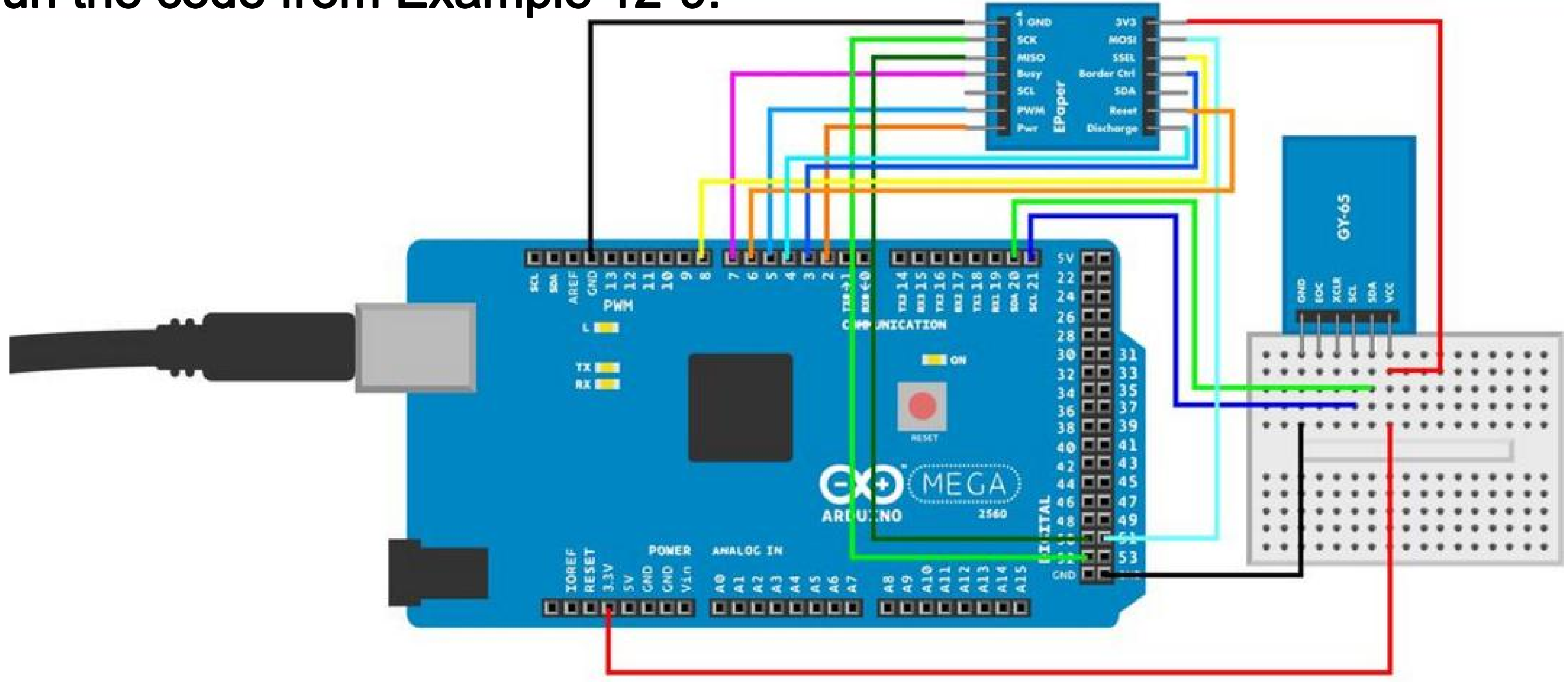


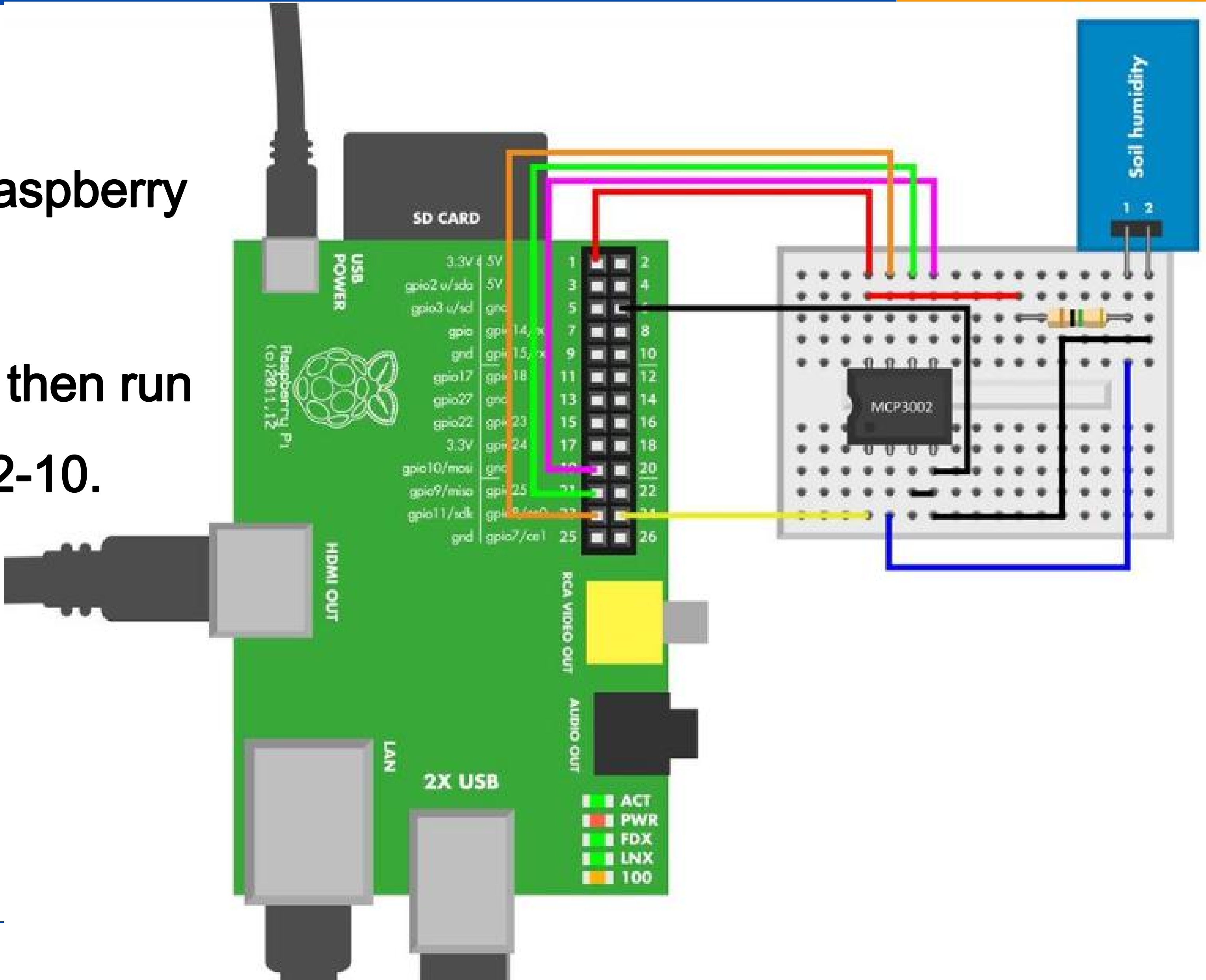
Figure 12-17. Connections on Arduino Mega

Example 12-9. soil_humidity_sensor.ino

```
// soil_humidity_sensor.ino - read soil humidity by measuring its resistance.
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int sensorPin = A0;
int soilHumidity = -1;

void setup() {
		Serial.begin(115200);
}

void loop() {
		soilHumidity = analogRead(sensorPin); //
		Serial.println(soilHumidity);
		delay(100);
}
```

Soil Sensor Code and Connection for Raspberry Pi

- Figure 12-14 shows the connection diagram for Raspberry Pi.
- Hook it up as shown, and then run the code from Example 12-10.



Example 12-10. soil_humidity_sensor.py

```
# soil_humidity_sensor.py - read soil humidity by measuring its resistance.
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp

def main():
	while True:
		h = mcp.readAnalog()	#
		h = h / 1024 * 100	#
		print("Current humidity is %d %%" % h)
		time.sleep(5)

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

Test Project: E-paper Weather Forecast

- Create your own weather forecast on e-paper.
- The weather forecast is based on changes in atmospheric pressure.
- The e-paper display is quite special: you can see it well in bright light, it looks a bit like paper, and the picture stays on without consuming electricity.

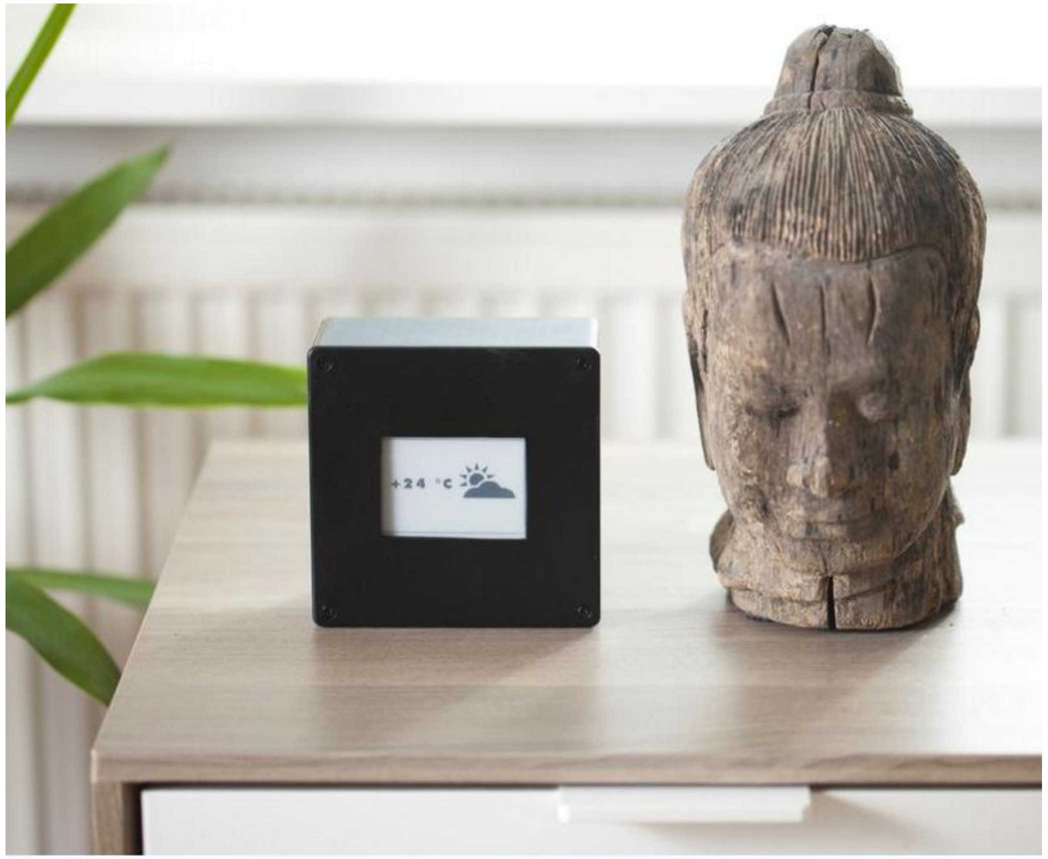


Figure 12-15. E-paper Weather Forecast

Test Project: E-paper Weather Forecast

- What You'll Learn
- In the E-paper Weather Forecast project, you'll learn how to:
  - Build a box that shows a graphical weather forecast.
  - Predict weather using atmospheric pressure.
  - Display graphics on e-paper with zero energy consumption.
  - Make Arduino sleep to conserve power.

Test Project: E-paper Weather Forecast

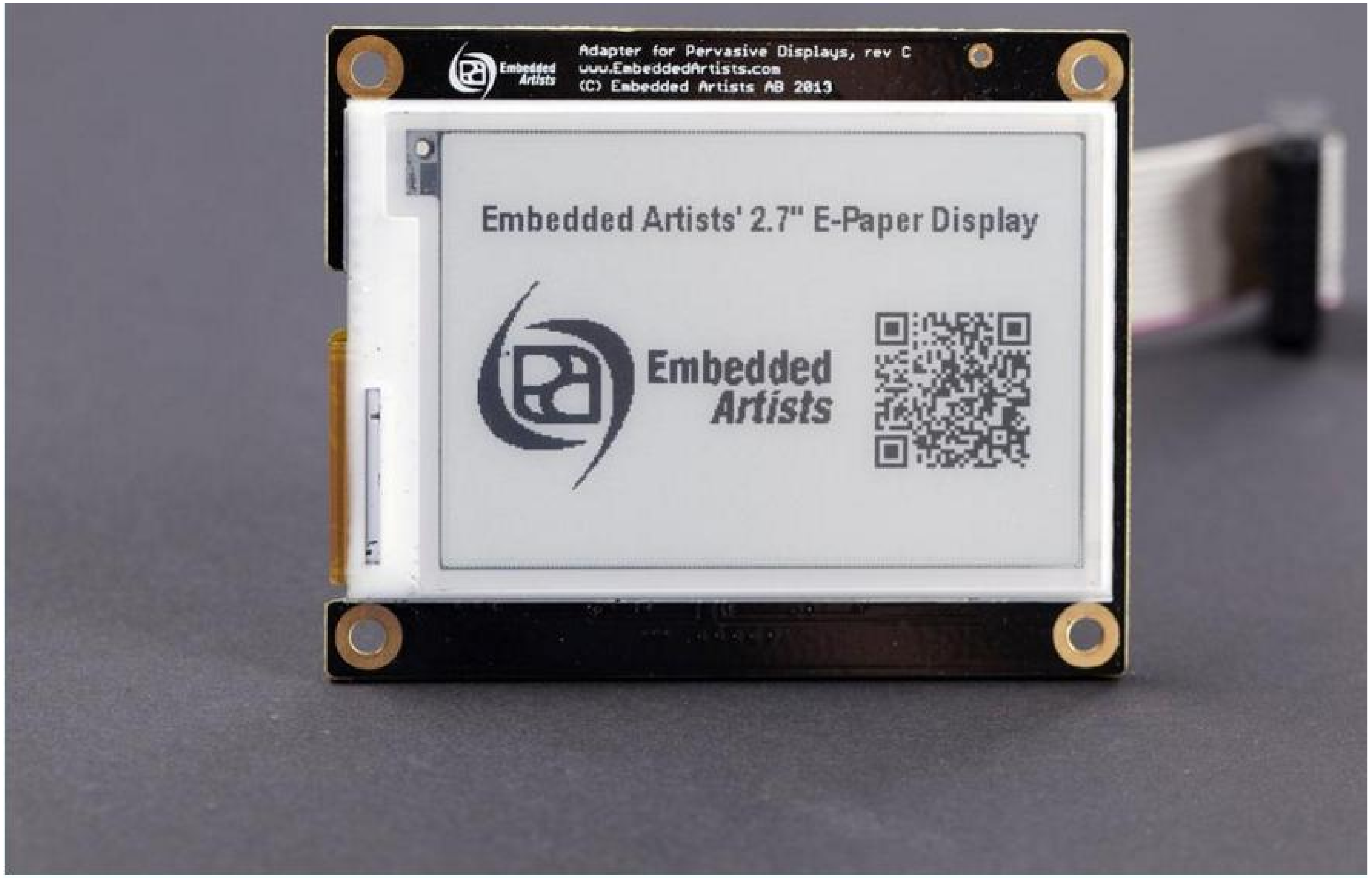

Adapter for Pervasive Displays, rev C
www.EmbeddedArtists.com
(C) Embedded Artists AB 2013
Embedded Artists' 2.7" E-Paper Display
Embedded Artists

Figure 12-16. E-paper display

Weather Forecast Code and Connection for Arduino

- The code combines many techniques. You can just build it first, and then learn about the implementation details once it works.
- To create your own version, it’s not required that you understand all the code. After you have your weather station running, have a look at drawScreen(). It’s the main function, and quite high level. For example, you could start by changing pos, the location where the plus sign is drawn:

```
int pos = 10;
drawCharacter(pos, 70, font,'+');
```

Weather Forecast Code and Connection for Arduino

Techniques used in this code:

- Reading data from the GY65 atmospheric pressure sensor (Atmospheric Pressure GY65).
- Working with hexadecimal numbers, binary numbers, and bitwise operations. See (Hexadecimal, Binary, and Other Numbering Systems and Bitwise Operations).
- Making Arduino sleep to save power. This uses low-level commands to write to ATmega (the chip that powers Arduino) registers.
- Drawing on e-paper display, using the EPD library.
- Storing images as header files (as in imagename.h). This is explained in detail in Storing Images in Header Files.

Weather Forecast Code and Connection for Arduino

- Figure 12-17 shows the connections for the Arduino Mega. Wire it up as shown, and then run the code from Example 12-11.

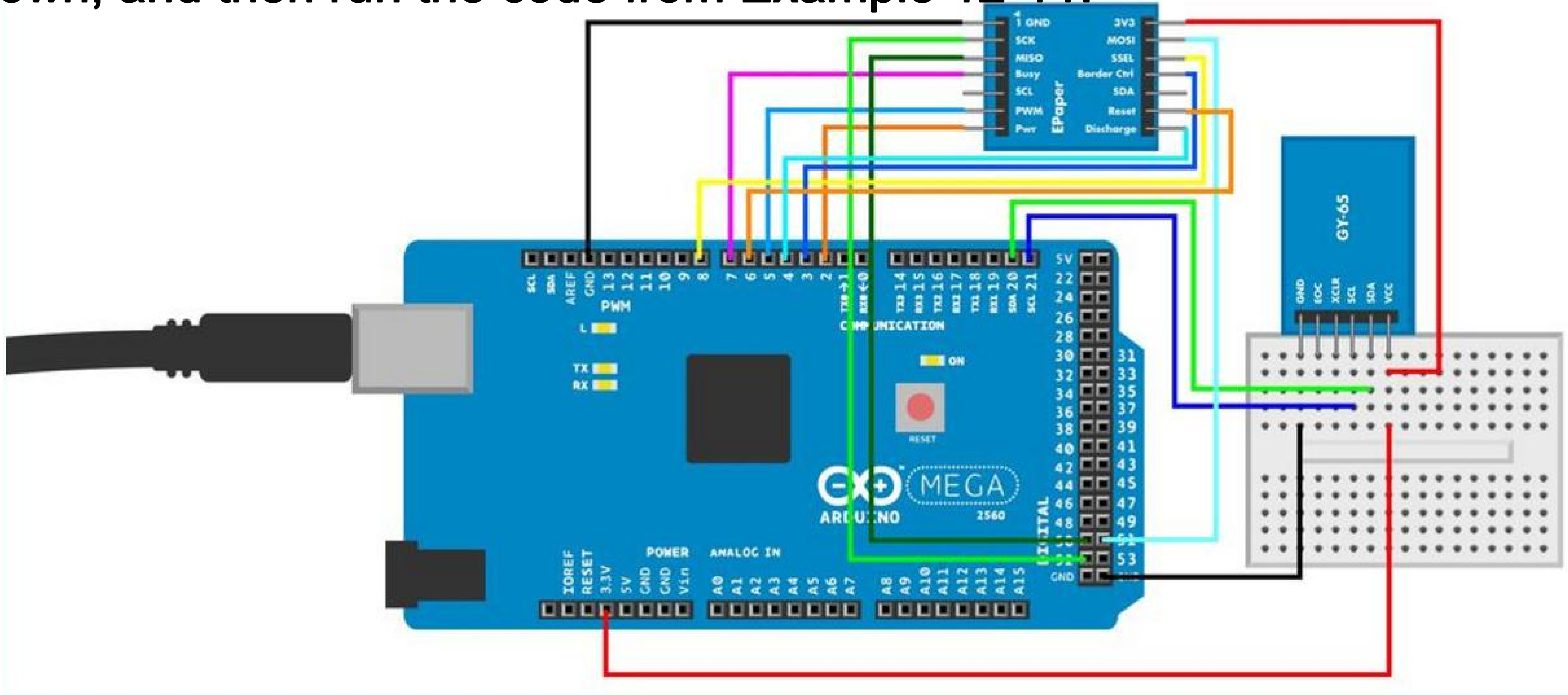


Figure 12-17. Connections on Arduino Mega

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <inttypes.h>
#include <ctype.h>

#include <SPI.h>
#include <Wire.h>#include <EPD.h>        //
#include <gy_65.h>      //
#include <avr/sleep.h>
#include <avr/power.h>

#include "rain.h"       //
#include "sun.h«
#include "suncloud.h«
#include "fonts.h«

uint8_t imageBuffer[5808]; // 264 * 176 / 8
```

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int pinPanelOn = 2;
const int pinBorder = 3;
const int pinDischarge = 4;
const int pinPWM = 5;
const int pinReset = 6;
const int pinBusy = 7;
const int pinEPDcs = 8;

EPD_Class EPD(EPD_2_7, pinPanelOn, pinBorder, pinDischarge, pinPWM, pinReset,  pinBusy,
pinEPDcs, SPI);

float weatherDiff;
float temperature;

const int sleepMaxCount = 10;   // min
volatile int arduinoSleepingCount = sleepMaxCount;
```

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void setup() {
        Serial.begin(115200);
        pinMode(pinPanelOn, OUTPUT);
        pinMode(pinBorder, OUTPUT);
        pinMode(pinDischarge, INPUT);
        pinMode(pinPWM, OUTPUT);
        pinMode(pinReset, OUTPUT);
        pinMode(pinBusy, OUTPUT);
        pinMode(pinEPDcs, OUTPUT);

        digitalWrite(pinPWM, LOW);
        digitalWrite(pinReset, LOW);
        digitalWrite(pinPanelOn, LOW);
        digitalWrite(pinDischarge, LOW);
        digitalWrite(pinBorder, LOW);
        digitalWrite(pinEPDcs, LOW);
```

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
        SPI.begin();  //
        SPI.setBitOrder(MSBFIRST);    //
        SPI.setDataMode(SPI_MODE0);   //
        SPI.setClockDivider(SPI_CLOCK_DIV4);  //

        WDTCSR |= (1<<WDCE) | (1<<WDE);       //
        WDTCSR = 1<<WDP0 | 1<<WDP3;   //
        WDTCSR |= _BV(WDIE);  //
        MCUSR &= ~( 1 << WDRF);       //

        for(int i = 0; i < 5808; i++) //
                imageBuffer[i] = 0;

        readCalibrationData();        //
}
```

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

char characterMap[14] = {'+', '-', 'C', 'd',
                         '0', '1', '2', '3',
                         '4', '5', '6', '7',
                         '8', '9'};     //

void drawCharacter(int16_t x, int16_t y, const uint8_t *bitmap, char character) {
        int charIndex = -1;
        for(int i = 0; i < 14; i++) { //
                if(character == characterMap[i]) {
                        charIndex = i;
                        break;
                }
        }
        if(charIndex == -1)
                return;
        drawBitmap(x,y,bitmap,charIndex*25,0,25,27,350);     //
}
```

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void drawBitmap(int16_t x, int16_t y, const uint8_t *bitmap, int16_t x2,
                    int16_t y2, int16_t w, int16_t h, int16_t source_width) {   //

          int16_t i, j, byteWidth = source_width / 8;

          for(j=y2; j<y2+h; j++) {      //
                    for(i=x2; i<x2+w; i++ ) {
                              byte b= pgm_read_byte(bitmap+j * byteWidth + i / 8);
                              if(b & (128 >> (i & 7))) {
                                        drawPixel(x+i-x2, y+j-y2, true);
                              }
                    }
          }
}
```

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void drawPixel(int x, int y, bool black) { //
		int bit = x & 0x07;
		int byte = x / 8 + y * (264 / 8);
		int mask = 0x01 << bit;
		if(black == true) {
				imageBuffer[byte] |= mask;
		} else {
				imageBuffer[byte] &= ~mask;
		}
}
void drawBufferToScreen() {	//
		for (uint8_t line = 0; line < 176 ; ++line) {
				EPD.line(line, &imageBuffer[line * (264 / 8)], 0, false, EPD_inverse);
		}
		for (uint8_t line = 0; line < 176 ; ++line) {
				EPD.line(line, &imageBuffer[line * (264 / 8)], 0, false, EPD_normal);
		}
}
```

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void drawScreen() {     //
            EPD.begin();
            EPD.setFactor(temperature);
            EPD.clear();
            if(weatherDiff > 250) {       // Pa    // Sunny
                        drawBitmap(140,30,sun,0,0,117,106,117);     //
            } else if ((weatherDiff <= 250) || (weatherDiff >= -250)) {    // Partly cloudy
                        drawBitmap(140,30,suncloud,0,0,117,106,117);
            } else if (weatherDiff < -250) {    // Rain
                        drawBitmap(140,30,rain,0,0,117,106,117);  }
//Draw temperature
            String temp = String((int)temperature);
            int pos = 10;
            drawCharacter(pos,70,font,'+');       //
            pos += 25;
            drawCharacter(pos,70,font,temp.charAt(0));
            pos += 25;
```

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
        if(abs(temperature) >= 10) {
                drawCharacter(pos,70,font,temp.charAt(1));
                pos += 25;
        }
        drawCharacter(pos,70,font,'d');
        pos += 25;
        drawCharacter(pos,70,font,'C');

        drawBufferToScreen(); //

        EPD.end();

        for(int i = 0; i < 5808; i++) //
                imageBuffer[i] = 0;
}
```

Example 12-11. weather_station.ino

```
// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void loop() {
        Serial.println(temperature);  //
        if(arduinoSleepingCount >= sleepMaxCount) {   //
                readWeatherData();  //
                drawScreen();
                arduinoSleepingCount = 0; //
                arduinoSleep(); //
        } else {
                arduinoSleep();
        }
}

const float currentAltitude = 40.00; //Installation altitude in meters
const long atmosphereSeaLevel = 101325; // Pa
const float expectedPressure = atmosphereSeaLevel * pow((1-currentAltitude / 44330), 5.255);
```

Example 12-11. weather_station.ino

// weather_station.ino - print weather data to epaper
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
void readWeatherData(){
		temperature = readTemperature();
		float pressure = readPressure();
		weatherDiff = pressure - expectedPressure;
}

ISR(WDT_vect)   // {
		arduinoSleepingCount++;
}

void arduinoSleep() { //
		set_sleep_mode(SLEEP_MODE_PWR_DOWN);
		sleep_enable();
		sleep_mode(); //
		 sleep_disable();       //
		power_all_enable();
} //
```

Environment Experiment: Look Ma, No Power Supply

- E-paper displays use power only to change the picture on the display. They don't need any energy to leave a picture on the screen.
- You can try it yourself. Use Arduino to display something on your e-paper display. Then power down Arduino, and even disconnect the e-paper display. The picture stays unchanged. In fact, there is no visible difference at all between keeping e-paper connected or disconnecting it (Figure 12-18).

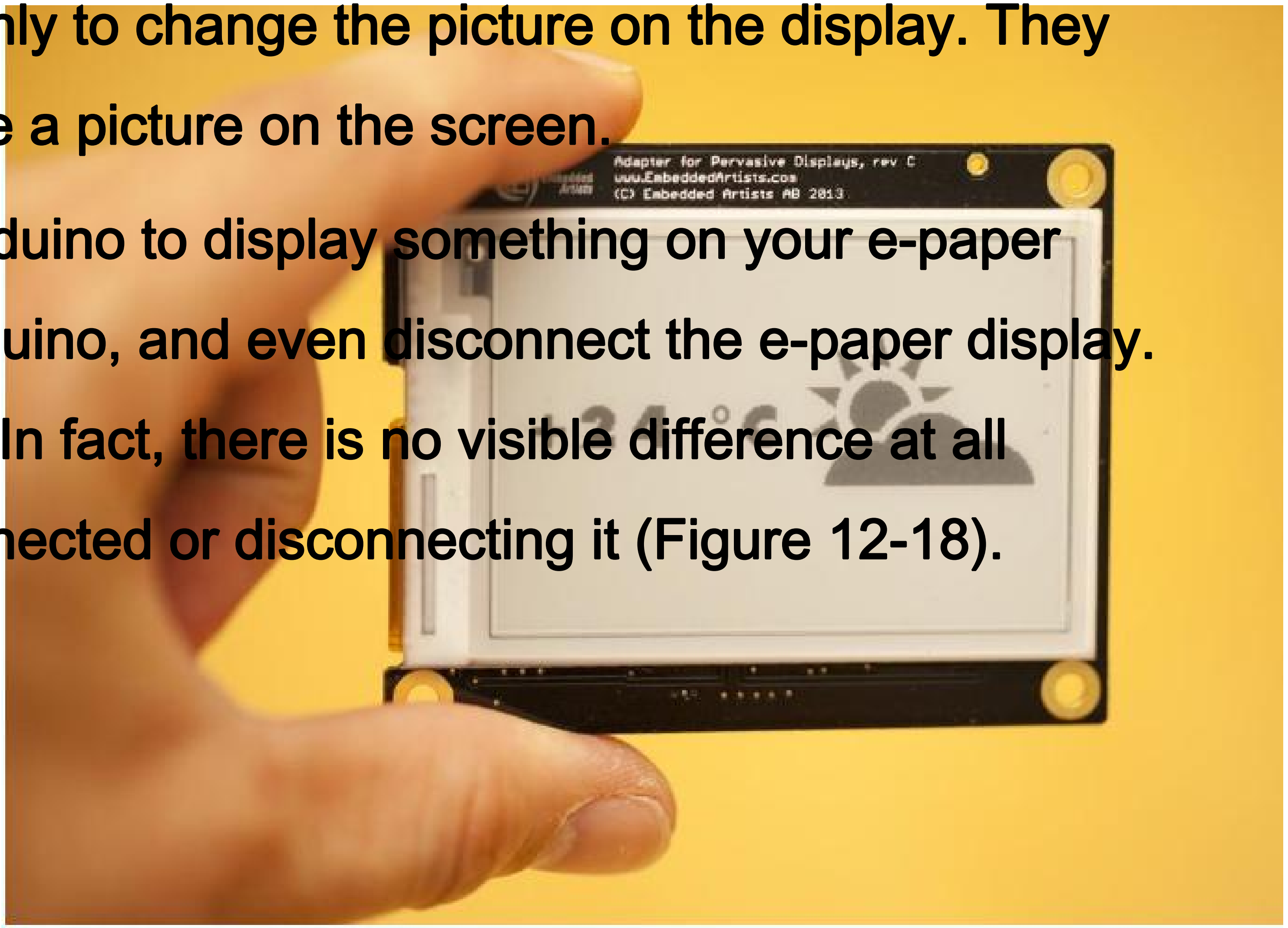

Adapter for Pervasive Displays, rev C
www.EmbeddedArtists.com
(C) Embedded Artists AB 2013
24°C

Figure 12-18. The same image stays on an e-paper display even without power

Storing Images in Header Files

- Your very own e-paper weather station can already show you the sun. What if you want to draw your own graphics?
- You can draw your images in an image editing program like Photoshop, and then convert the saved BMP images to the C code headers we use in the sketch.
- First, use your favorite drawing program to draw an image. Save your image in the BMP format. Put the image into the same folder with the other images in this project (images/).
- Next, you'll need to convert the BMP image (sun.bmp) to a C header file (sun.h). You can use the included image2c-botbook.py script for this.

Storing Images in Header Files

- First, install the requirements: Python and Python Imaging Library.
- Mac and Windows users will need to download the Python Imaging Library from http://www.pythonware.com/products/pil/. Linux users have it easier.
- For example, here's how to install the needed libraries:

```
$ sudo apt-get update
$ sudo apt-get -y install python python-imaging
```

Storing Images in Header Files

- If your source image is not called foo.bmp, change the filenames in the script. Then you’re ready to convert.
- These commands work in all of Linux, Windows, and Mac.
- The first command assumes you’re in the subdirectory containing the example code for this book. You can download the sample code from http://botbook.com.

```
$ cd arduino/weather_station/
$ python image2c-botbook.py
BMP (117, 106) 1
```

- The file is now converted. With the ls or dir command, you can see that foo.h was created in the current working directory.

Storing Images in Header Files

- You have now converted your BMP image to a C header file.
- In foo.h, your image is now C code:

```
// File generated with image2c-botbook.py
const unsigned char sun [] PROGMEM= {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ..
0x7F, 0xF0, 0x00, 0xFF, 0xC0, 0x00, 0x1F, 0xF8, 0x00, // ..
```

- This format is convenient for programming. Each byte represents eight pixels, one pixel per bit. For example, hex 0xFC is number 252. In bits, it’s 0b11111100. This means eight bits side by side: black black black black, black black white white.

BMP to C Conversion Program

- Image2c-botbook.py converts a BMP image to C header file.
- If you need some “code golf” (see http://codegolf.com), try making it work with PNG source files.
- This code uses hexadecimal numbers and bitwise operations. To review them, see Hexadecimal, Binary, and Other Numbering Systems and Bitwise Operations.

Example 12-12. Image to C header code

```
# image2c-botbook - convert a BMP image to C for use in eInk and LCD displays
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import Image    #
import math
imageName = "images/foo.bmp"    #
outputName = "foo.h"    #

im = Image.open(imageName)      #
print im.format, im.size, im.mode

width, height = im.size
pixels = list(im.getdata())     #
length = int(math.floor(width * height / 8.0))  #
carray = length * [0x00]        #
```

Example 12-12. Image to C header code

# image2c-botbook - convert a BMP image to C for use in eInk and LCD displays

# (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
for y in xrange(0, height):    #
        for x in xrange(0, width):     #
                pixel = pixels[y * width + x]   #

                bit = 7 - x & 0x07      #
                byte = x / 8 + y * (width / 8)  #

                mask = 0x01 << bit      #
                if pixel == 0:
                        carray[byte] |= mask    #
                else:
                        carray[byte] &= ~mask
```

Example 12-12. Image to C header code

# image2c-botbook - convert a BMP image to C for use in eInk and LCD displays
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
fileHandle = open(outputName, "w")      #
index = 0fileHandle.write("//
File generated with image2c-botbook.py\n")
fileHandle.write("const unsigned char sun [] PROGMEM= {\n")
for b in carray:
        fileHandle.write("0x%02X, " % b)        #
        index += 1
        if index > 15:
                fileHandle.write("\n")
                index = 0
fileHandle.write("};\n")
```

Enclosure Tips

- We used a plastic box made by Hammond for our weather forecaster. But how do you make an odd-shaped hole like the one we need here?
- First draw the shape you want to cut out on the box lid. Drill holes in each corner of the shape with a large drill bit (10-20 mm).
- Then start going from corner to corner with a jigsaw blade.
- Finish the hole with file and sandpaper (see Figure 12-19).

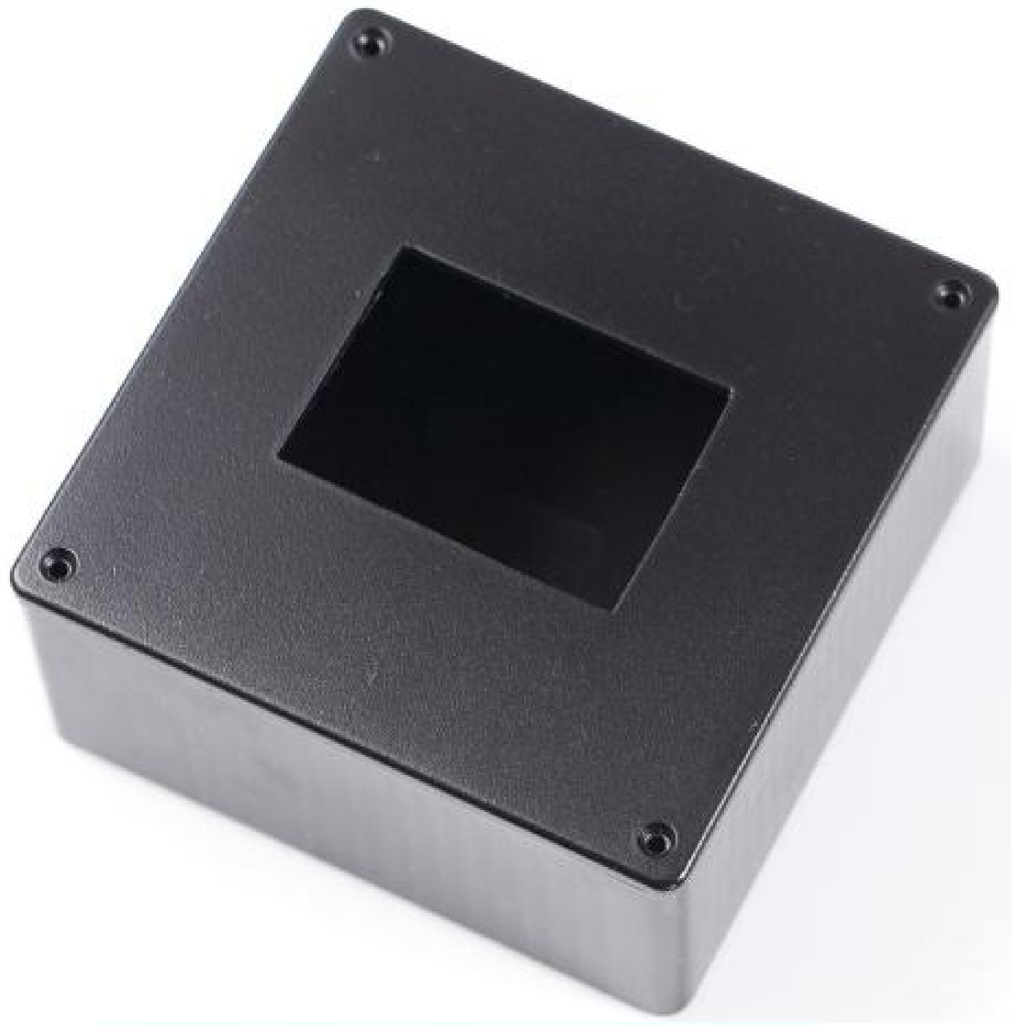


Figure 12-19. Hole for the screen

Enclosure Tips

- Use hot glue to attach the screen to the box as shown in Figure 12-20.

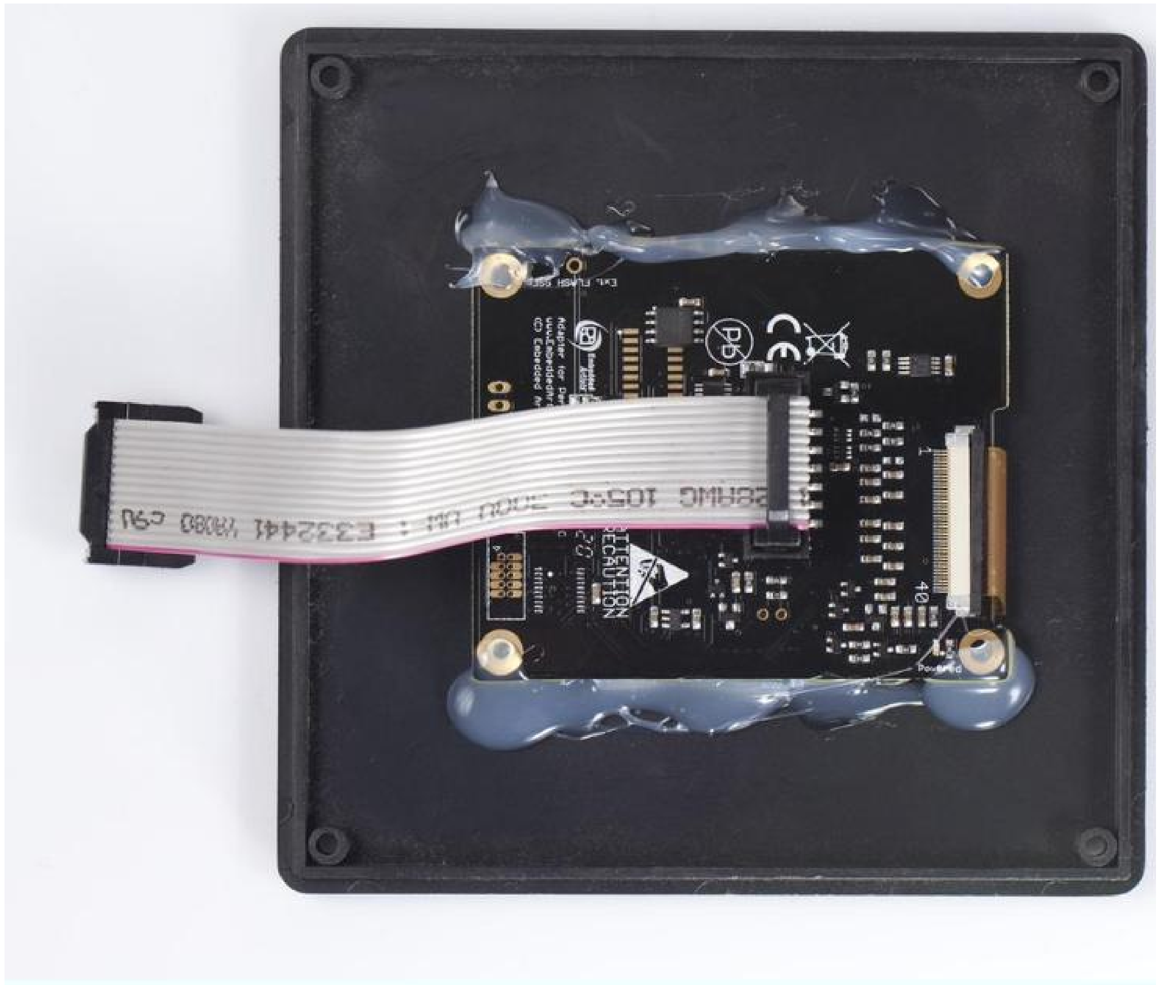


Figure 12-20. Screen hot glued

Enclosure Tips

- On the back of the box, we made a pattern of small holes (see Figure 12-21).
- Without those, air would not be able to get to the sensor.

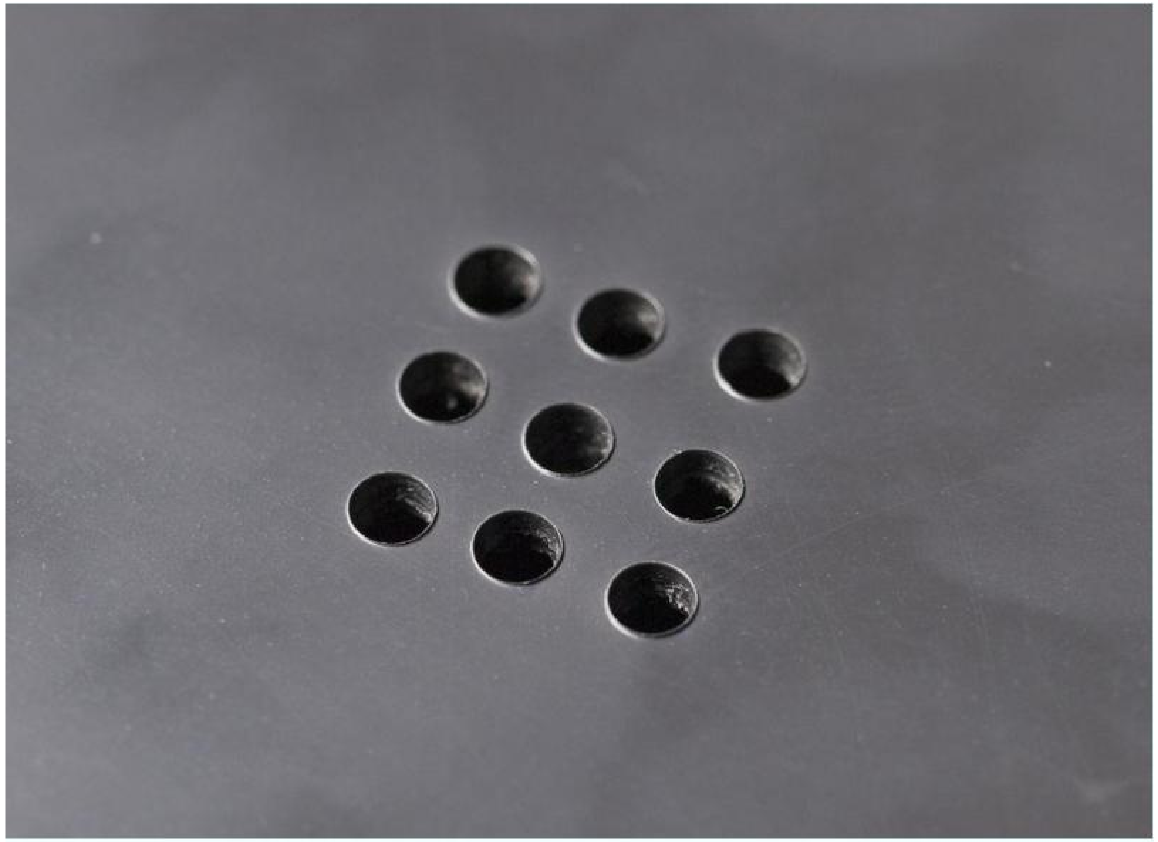


Figure 12-21. Holes to let the air in

Enclosure Tips

- The finished gadget is shown in Figure 12-22.

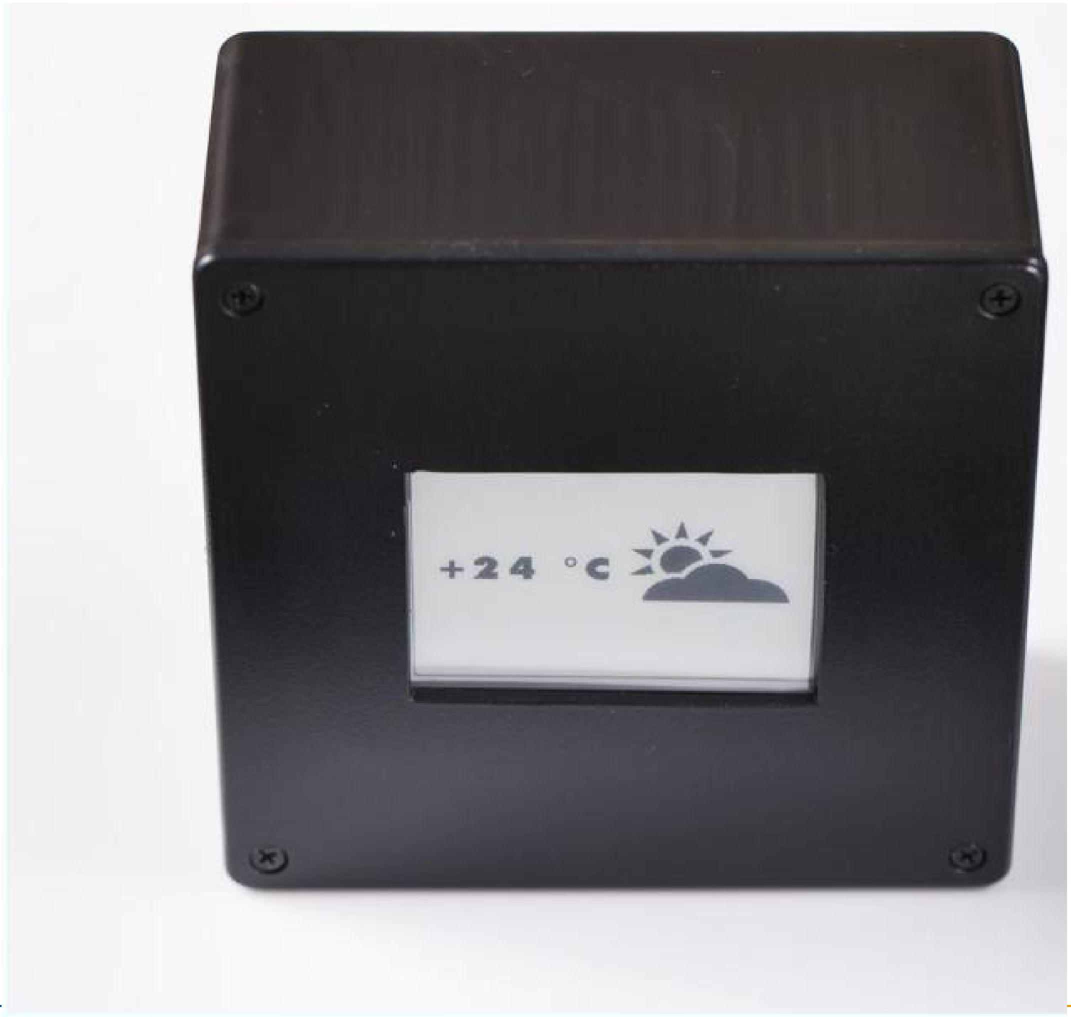


Figure 12-22. The finished gadget