Chapter No.1
"Building Wireless XBee Motion Detectors"
In this package, you will find:
The author’s biography
A preview chapter from the book, Chapter no.1 "Building Wireless XBee Motion Detectors"
A synopsis of the book’s content
Information on where to buy this book

About the Author
Marco Schwartz is an electrical engineer, entrepreneur, and blogger. He has a Master's degree in Electrical Engineering and Computer Science from Supélec, France, and a Master's degree in Micro Engineering from EPFL, Switzerland.

He has more than 5 years of experience working in the domain of electrical engineering. His interests gravitate around electronics, home automation, the Arduino and Raspberry Pi platforms, open source hardware projects, and 3D printing.

He runs several websites based on Arduino, including the Open Home Automation website that is dedicated to building home automation systems using open source hardware.

He has written another book on home automation and Arduino called Home Automation with Arduino, CreateSpace. He has also published a book on how to build Internet of Things projects with Arduino called Internet of Things with the Arduino Yun, Packt Publishing.

For More Information:
Arduino Home Automation Projects

Home automation is a topic that has been around for many years. It includes everything that you can imagine to control and automate your home. The most widely spread example is the alarm system of your home. Motion sensors, contact sensors, and the central device that orchestrates your alarm system are generally the main components of any home automation system.

There are countless devices that are available for home automation. You can buy complete home automation devices from a lot of stores, and even get them installed in your home. However, many of these systems are very expensive, impossible to be customized for your own needs, and have outdated user interfaces.

On the other hand, we have the Arduino platform. Arduino is a platform that you can use to quickly prototype electronic systems. It is now used by millions of people around the world to build more complex systems. It is actually the perfect platform to build home automation systems. Because of the flexibility of the Arduino platform, we are going to see that it is easy to interface with various sensors and actuators that are usually found in many home automation systems. It can also be interfaced with many wireless systems, such as Wi-Fi, Bluetooth, or XBee.

In this book, we are going to see how to build home automation systems with Arduino. We will first build systems based on commercially available Arduino boards. For example, we are going to build a temperature sensor based on Arduino and Bluetooth. We are also going to integrate some of these systems in an Internet of Things perspective, by sending some data directly to a cloud service. Finally, we are also going to see how to hack commercially available devices and build your own home automation systems from scratch.

For More Information:
What This Book Covers

Chapter 1, Building Wireless XBee Motion Detectors, covers a very common topic in home automation: motion detectors. We are going to build a swarm of motion detectors based on the well-known XBee protocol and Arduino. We are also going to build a server-side interface to monitor the state of the XBee motion detectors.

Chapter 2, Control Lights from Your Phone or Tablet, covers another popular topic in home automation systems: controlling lights remotely. We are going to interface a relay with Arduino and a Wi-Fi chip so that you can control lights in your home not only from your computer, but also from your phone or tablet.

Chapter 3, Measuring the Temperature Using Bluetooth, focuses on measuring temperature and humidity with Arduino, and transmitting the result back to your computer using Bluetooth. We are also going to build a simple interface using Python so that you can access the measurements made by the Arduino and Bluetooth system.

Chapter 4, Weather Station in the Cloud with Xively, teaches you to tackle a very trendy topic: the Internet of Things. We are going to make basic weather measurements on our Arduino board, and then transmit this data via Wi-Fi to the cloud using a service called Xively. Using this service, we'll be able to monitor our data remotely from anywhere in the world.

Chapter 5, Monitor Your Energy Consumption in the Cloud, starts with the use of the same cloud service that was used in Chapter 4, Weather Station in the Cloud with Xively. However, in this case, we are going to send energy consumption data to the cloud. This way, you will be able to monitor data directly from the Xively interface.

Chapter 6, Hack a Commercial Home Automation Device, explores the idea of doing things differently. Instead of creating a new home automation system based on Arduino, we are going to hack an existing device so that you can control it from your computer. We are going to build a USB-controlled power switch, so you can control any device right from an interface running on your computer.

Chapter 7, Build Your Own Home Automation System, goes one step further and shows you how to build your own home automation system based on Arduino. We'll cover how to build an Arduino system from scratch, how to design your own PCB, and finally how to design and 3D print a custom case for your project.

For More Information:
Building Wireless XBee Motion Detectors

In this chapter, we are going to build a project around a very common home automation sensor: a motion sensor. Ever noticed those small modules in white plastic that are in the upper corners in some rooms of people's houses, modules that turn red when you walk in front of them? That's exactly the same thing we are going to do in this project.

However, instead of using proprietary technologies, which is usually the case for these modules, we are going to base our system on Arduino. And for the communication part, we are going to use XBee modules, which are low-power radio modules that are widely used with the Arduino platform. These modules are based on the ZigBee protocol, which is also used in many commercial home automation systems.

Here are the major takeaways that we will see in this chapter:

- First of all, we will list all the hardware and software components that we need for this project. With these components, we will build one motion sensor module composed of an Arduino board, a motion sensor, and one XBee module.
- Then, we will test this first module; the motion sensor will be tested on its own, and we will also test the communication part by sending commands via the serial monitor of the Arduino software.
- Finally, we are going to build a web-based graphical interface that centralizes all the data of our XBee sensors. With a simple interface built on web technologies, you'll be able to instantly see if some motion is detected in your home.

For More Information:
Hardware and software requirements

For this first project, we will need Arduino boards, PIR motion sensors, and some XBee modules and XBee shields, depending on the number of sensors you want to have in your system. For just one sensor, you will need the following components:

- Arduino R3 board (https://www.sparkfun.com/products/11021)
- PIR sensor (http://www.adafruit.com/products/189)
- Series 1 XBee module (https://www.sparkfun.com/products/11215)
- Arduino XBee shield (https://www.sparkfun.com/products/10854)

Note that I added the link to the components I used for this project as a reference, but you can also choose to use other components.

The motion sensor needs to have three pins: two for the power supply and one signal pin. It should also use a 5V voltage level to be compatible with the Arduino Uno board that also operates at 5V.

For the XBee module, I used a Series 1 XBee module, 1mW, with a trace antenna (which means it doesn't require any external antenna). You could also use a module with an external antenna, but you would then have to connect the antenna to the module. I used Series 1 XBee modules for this project as they are easier to use than Series 2, which have functionalities we do not need for this simple project. This module has a range of about 100 meters without obstacles.

You will also need to connect your XBee module to your Arduino board. For that, each of my motion sensor modules will use an Arduino XBee shield from Sparkfun, but you can use other brands as well. It just needs to make the connections between the XBee module and the Arduino board.

Finally, you will need a way to communicate with these XBee modules from your computer. I used another XBee module (also Series 1, 1mW, with a trace antenna) connected to an XBee explorer board from Sparkfun, which is basically a USB interface board where you can plug any XBee module. I used the following components for the module connected to the computer:

- XBee explorer USB (https://www.sparkfun.com/products/8687)
- Series 1 XBee module (https://www.sparkfun.com/products/11215)

On the software side, you need to have the latest version of the Arduino IDE installed on your computer. For this project, the Arduino IDE 1.0.5 was used.
You will also need the aREST library for Arduino. You can find this library at the following link:

https://github.com/marcoschwartz/aREST

To install the library, extract all the files in a folder named aREST (or create this folder if it doesn't exist yet). Then, place this folder in your /libraries folder inside your main Arduino folder. You will also need to have a web server installed and running on your computer so that you can use the web interface that we are going to develop at the end of this chapter.

To configure your XBee modules, you will also need to have the XCTU software installed. You can find it at the following URL:


**Hardware configuration**

The hardware configuration of this project is not really complex. For each motion sensor module you want to build, you'll need to do the following steps. The first one is to plug an XBee module on the XBee shield. Then, you need to plug the shield into your Arduino board, as shown in the following image:

![XBee Module on Arduino Shield](image)

For More Information:

Now, you can connect the motion sensor. It has three pins: VCC (for the positive power supply), GND (which corresponds to the reference voltage level), and SIG (which will turn to a digital HIGH state in case any motion is detected). Connect VCC to the Arduino 5V pin, GND to Arduino GND, and SIG to Arduino pin number 8 (the example code uses pin 8, but you could also use any digital pin). You should end up with something similar to this image:

You will also need to set a jumper correctly on the board so we can upload a sketch. On the XBee shield, you have a little switch close to the XBee module to choose between the XBee module being connected directly to the Arduino board serial interface (which means you can’t upload any sketches anymore) or leaving it disconnected. As we need to upload the Arduino sketch first, you need to put this switch to DLINE, as shown in this image:

For More Information:
You will also need to connect the XBee explorer board to your computer at this point. Simply insert one XBee module to the board as shown in the following image:

Now that this is done, you can power up everything by connecting the Arduino board and explorer module to your computer via USB cables.

If you want to use several XBee motion sensors, you will need to repeat the beginning of the procedure for each of them: assemble one Arduino board with an XBee shield, one XBee module, and one motion sensor. However, you only need one USB XBee module connected to your computer if you have many sensors.

**Interfacing the PIR sensor with Arduino**

First off, you are going to leave XBee aside and simply check if the motion sensor is working correctly. What you will do in the first sketch is print out the readings from the motion sensor on the serial port. This is the complete code for this part that you can just copy and paste in the Arduino IDE:

```cpp
// Simple motion sensor
int sensor_pin = 8;

void setup() {

```
Building Wireless XBee Motion Detectors

Serial.begin(9600);
}

void loop() {

  // Read sensor data
  int sensor_state = digitalRead(sensor_pin);

  // Print data
  Serial.print("Motion sensor state: ");
  Serial.println(sensor_state);
  delay(100);
}

Downloading the example code and colored images
You can download the example code files and colored images
for this Packt book that you have purchased from your account
at http://www.packtpub.com. If you purchased this book
elsewhere, you can visit http://www.packtpub.com/support
and register to have the files e-mailed directly to you.

Let's see what this code does. It starts by declaring the pin on which the sensor is
connected, in our case 8. In the setup() function of the sketch, we initialize the serial
connection with the computer, so we can print out the results on the serial monitor.

Then, in the loop() part of the sketch, we read out the state of the motion sensor
using a simple digitalWrite() command, and store that result into a variable. This
state is then simply printed out on the serial port every 100 ms.

You can now upload the sketch to your Arduino board and open the serial monitor.
This is what you should see:

Motion sensor state:0
Motion sensor state:1
Motion sensor state:1
Motion sensor state:1
Motion sensor state:0
Motion sensor state:0

If you can see the state of the sensor changing when you wave your hand in front of
it, it means that the sensor is working correctly and that you can proceed to the rest
of the project.
Programming an XBee motion detector

You are now going to modify the sketch slightly so that it can transmit the state of the sensor to a central interface running on your computer. However, you not only want to transmit the state of the motion sensor, but also an ID identifying the sensor that is detecting the motion. Programming the detector starts by importing the right libraries:

```c
// Libraries
#include <aREST.h>
#include <SPI.h>
```

The `aREST` library implements a REST API for Arduino. REST stands for REpresentational State Transfer, and is widely used in web applications such as Software as a Service (SaaS) applications. In our case, we will use this library to standardize the communication with the central interface that will run on the computer. In this project, the REST commands will be sent over the XBee connection that acts as a serial port from the Arduino point of view.

After importing the libraries, you need to declare the sensor pin and the ID of the module as follows:

```c
// Motion sensor pin and ID
int sensor_pin = 8;
String xbee_id = "2";
```

After this, you can create the instance of the `aREST` library that will handle the requests coming from the graphical interface:

```c
// Create aREST instance
aREST rest = aREST();
```

In the `setup()` function of the sketch, the first step is to start the serial communication. Be careful here, as the speed of the serial object has to be the same as the speed of your XBee modules, which is 9600 bauds by default:

```c
// Start Serial
Serial.begin(9600);
```

You can also set the ID of the module:

```c
// Give name and ID to device
rest.set_id(xbee_id);
```
Building Wireless XBee Motion Detectors

Note that if you are configuring more than one sensor, you need to change the ID of each sensor you are configuring. Now, thanks to the aREST library, the loop() part of the sketch is pretty simple. We simply have to handle the incoming requests from the computer that will come via the XBee serial interface:

```cpp
void loop() {

    // Handle REST calls
    rest.handle(Serial);

}
```

Now, the sketch is ready to be used.

All the code is available on the GitHub repository of the project:

https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter1

You can upload the sketch to your Arduino board by making sure that the switch is still set on DLINE. Once this is done, you can test the code locally via the serial monitor of the Arduino IDE. Open the serial monitor, make sure that the serial speed is set to 9600, and type the following:

```
/digital/8/r
```

This is the REST command to read a digital value from pin number 8 and return the value, which is exactly what we want to achieve. You should see the following data being returned, depending on the current state of the sensor:

```
{"return_value": 0, "id": "2", "name": "", "connected": true}
```

What we are interested in is the return_value field, which contains the result of the digitalRead() function. Try to wave your hand in front of the sensor to see if the return value changes accordingly. Note that the returned data is in the JSON format, which will be really important later when we are going to process this information and display it.

Now that you are sure that the code is working, you can switch over to XBee communication. For that, simply put the switch next to the XBee module to UART. Now, the serial port of the Arduino board is directly wired to the XBee module.

For More Information:

By default, all the XBee modules sold are configured on the same **Personal Area Network (PAN)** ID, which is 3332. This means that all the modules will receive data from other modules on the same PAN ID. For experimentation, you can leave it at its default value. However, you might want to change this later, in case your neighbor is, for example, using XBee devices as well.

To continue further, insert the XBee module you want to modify in the USB XBee explorer and open the XCTU XBee tool. Click on the top-left button to add a new device and select the USB explorer serial port. You should get the following screen:

You will then be able to change the PAN ID of your device. To configure all the modules in your network, just repeat the procedure for each XBee module.

### Building a graphical interface for your XBee motion detectors

Now that the hardware is completely configured, you can build the server-side code to read data from all your motion sensor modules. For this project, I used two modules only, but you can use more modules for the project.

As for many web applications, our control center for the XBee motion detectors will be built using four different languages: HTML, CSS, JavaScript, and PHP. We are going to use HTML and CSS to design the interface itself, JavaScript to handle the different elements of the interface and make the link with PHP, and finally use PHP to communicate directly with the XBee modules.
The first step is to design the interface. We'll basically have several blocks on our page, one block corresponding to one XBee module. For example, this is the HTML code for the first block:

```html
<div class="sensorBlock"><span class="sensorTitle">Sensor 1</span><span class="display" id="display_1"></span></div>
```

Now, you can have a look at the JavaScript code that will handle the different elements of the interface. Note that all this code is included inside a dedicated JavaScript file. What we are going to do here is check the value of the sensor at regular intervals via a `digitalRead()` command. To do this, we have to call a PHP file with the correct command as an argument:

```javascript
$.get( "xbee.php", {command: "/digital/8/r"}, function( data ) {

The result from the PHP file is some string data formatted in the JSON format. To actually create a usable object from this string, we can use the `parseJSON()` function of jQuery:

```javascript
json_data = jQuery.parseJSON(data);
```

Now, we have a JavaScript object that has the same properties as the data fields inside the JSON container. For example, to get the ID of the sensor module, you can just type:

```javascript
var sensorID = json_data.id;
```

Now that we know which sensor returned some data, we can read the `return_value` field inside the JSON object to know the status of the motion sensor on this module. If this value is equal to 0, we set the background color of the display block of this sensor to gray. If it is equal to 1, it means that some motion was detected. If this is the case, we change the background color to orange as follows:

```javascript
if (json_data.return_value == 0){
    $('#display_' + sensorID).css("background-color","gray");
} else {
    $('#display_' + sensorID).css("background-color","orange");
}
```

Let's now see the details of this PHP file that is called every time. We saw in the JavaScript code that the PHP file also receives an argument called `command`. This variable is retrieved inside the PHP file by:

```php
$command = $_GET['command'];
```
Inside this file, we also need to include the `php_serial` class:

```php
include "php_serial.class.php";
```

We also need to declare the right serial port for your XBee explorer device:

```php
$serial_port = "/dev/cu.usbserial-A702LF8B";
```

You will have to change this according to your own settings. To know which serial port the XBee explorer board is using, simply search for devices starting with `cu` inside your `/dev` folder if you are using Linux or OS X. If you are using Windows, you will see the list of the serial ports in your Configuration Panel.

You now need to configure the different settings of the serial connection we are about to open with the board:

```php
$serial = new phpSerial;
$serial->deviceSet($serial_port);
$serial->confBaudRate(9600);
$serial->confParity("none");
$serial->confCharacterLength(8);
$serial->confStopBits(1);
```

Now, you can actually open the connection:

```php
$serial->deviceOpen();
```

We can now send the command and read the answer:

```php
$serial->sendMessage($command . "\r");
$answer = $serial->readPort();
```

Now, we close the serial connection:

```php
$serial->deviceClose();
```

Finally, we send back the data that was read:

```php
echo $answer;
```

We are now ready to test our project. You can of course find all the code for this part on the GitHub repository of the project at the following website:

https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter1

---

For More Information:

Make sure that your web server is running, and that all the files of the project are placed inside a folder in your web server's main folder. You can now head over to this folder, usually accessible by typing localhost in your favorite web browser. Open the HTML file, and you should see something similar to the following screenshot:

![XBeemoction sensors]

Note that on this example screenshot, the second sensor I had in my system already detected some motion. Now try with your sensors by waving your hand in front of them; you should instantly see that the state of the sensor changes on the interface.

Of course, by modifying the HTML and JavaScript code, you can easily modify this interface to adapt it for more sensors.

Also, be aware that these modules don't need to be connected to your computer directly once the code is loaded onto them. You can just disconnect them from your computer and power them using a simple battery pack (the Arduino Uno board, for example, accepts up to 12 V of power on its jack input).

Note that you can also access this interface from a mobile phone or tablet, just by accessing the localhost folder on your computer. Of course, your computer has to be powered (and the web server running) to access the interface.

For More Information:
Summary

Let's now summarize what we did in this first project of the book. We built XBee motion sensors that are based on Arduino, and learned that you can put them wherever you want in your home. They communicate with a central interface where you can see the status of every sensor live.

Of course, we could have used other wireless communication devices for this project, such as Wi-Fi. However, XBee modules are very energy efficient compared to Wi-Fi modules, so it completely makes sense to use XBee for this project, as you might want to power these motion sensors from batteries.

Let's see what the major takeaways of this chapter were. We first chose the components for our project, and built our first XBee-based motion sensor using Arduino and a PIR motion sensor. We also connected an XBee module to our computer via a USB so it can communicate with the other modules.

Then, we tested our motion sensor module by first testing the PIR sensor connected to the Arduino Uno board. We also tested the sketch at this point that we used to access the board via XBee.

Finally, we built a web-based interface to visualize the information coming from the different XBee motion sensors live. This interface was tested with two modules, but it is made to easily extend to many more XBee modules.

In the next chapter, we are going to tackle another huge field of home automation: controlling lights. And for this project, we won't be using XBee, but we are going to interface Arduino with a Wi-Fi chip to control lights from any mobile device.
Where to buy this book

You can buy Arduino Home Automation Projects from the Packt Publishing website:

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.