Build a Simple Alarm System with Arduino

by Marco Schwartz of Open Home Automation
http://www.openhomeautomation.net/

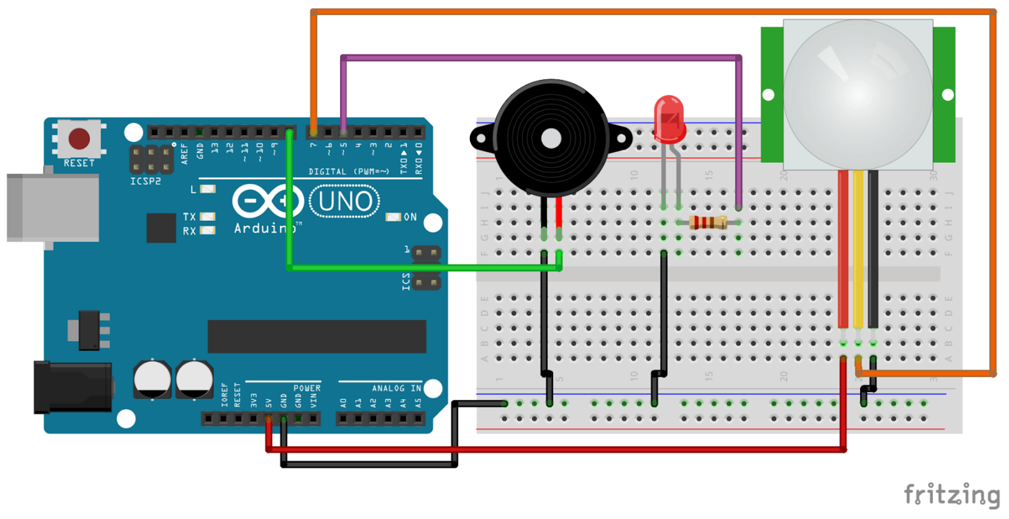
I love to build DIY home automation systems using open-source hardware, especially with platforms like Arduino. However, it can be a bit tricky to get started, and you might feel lost in front of all the options that are available to you.

This is why I wrote this guide to help you build your first home automation system using Arduino: a simple alarm system. We are going to interface a PIR motion sensor with Arduino. If motion is detected, we will flash an LED and make some sound with a small piezo buzzer. This simple project will give you the basics of home automation with Arduino. Let's start!

Here is the list of components you will need for this project:

- Arduino Uno (http://www.adafruit.com/product/50)
- PIR motion sensor (http://www.adafruit.com/product/189)
- LED (https://www.sparkfun.com/products/9590)
- 330 Ohm resistor (https://www.sparkfun.com/products/8377)
- Piezo buzzer (http://www.adafruit.com/product/160)
- Breadboard (http://www.adafruit.com/product/64)
- Jumper wires (http://www.adafruit.com/product/758)

We can now start assembling the project. To help you out, the schematic below summarizes the hardware connections:



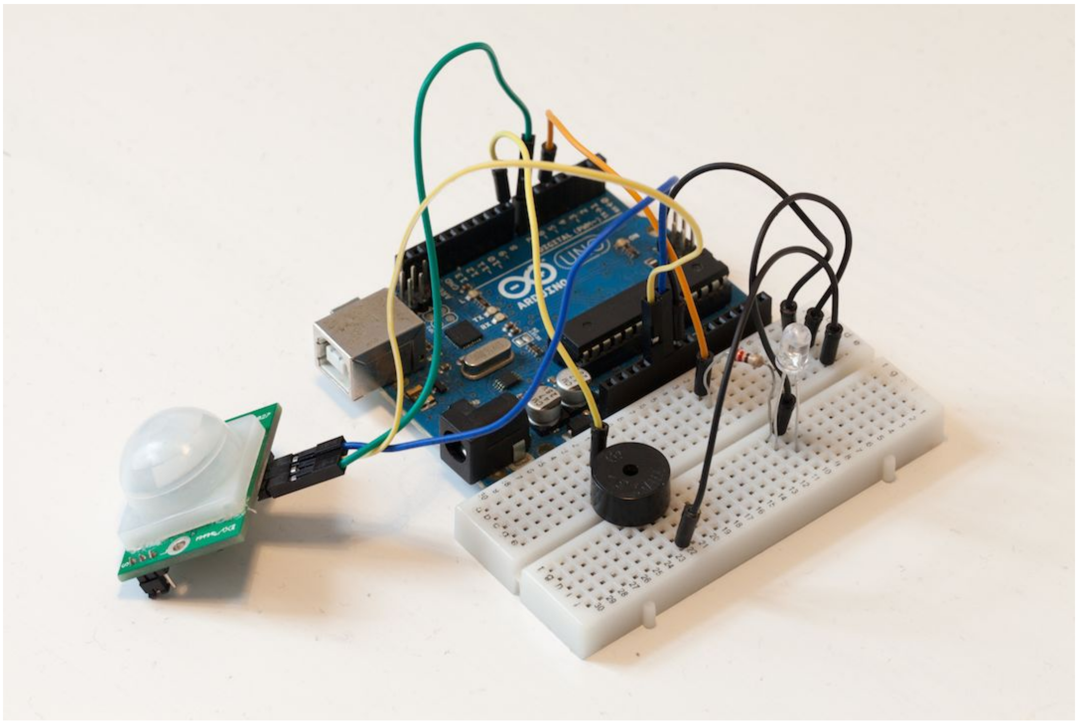
First, place all the components on the breadboard. After that, position the breadboard next to the Arduino board. Then, connect the PIR motion sensor to the breadboard. Connect the GND pin of the Arduino board to the blue rail of the breadboard as we will need to connect all devices to the same ground.

For the LED, connect the resistor in series with the LED anode on the breadboard (the anode is the longest pin on the LED). Then, connect the other pin of the resistor to Arduino pin 5. The other side of the LED must be connected to the Arduino ground.

For the PIR motion sensor, connect the GND pin to the Arduino ground, VCC to the Arduino 5V pin, and SIG pin to Arduino pin 7.

For the Piezo buzzer, connect the positive pin (marked with a +) to Arduino pin 8, and the other pin to the Arduino ground.

This is a picture of the fully assembled project:



Now that the hardware is assembled, we can start writing the Arduino sketch for our simple alarm system. This is the complete code for this part:

```
// Pins
const int alarm_pin = 8;
const int led_pin = 5;
const int motion_pin = 7;

// Alarm
boolean alarm_mode = false;

// Variables for the flashing LDED
int ledState = LOW;
long previousMillis = 0;
long interval = 100;  // Interval at which to blink (milliseconds)

void setup()
{
  // Set pins to output
  pinMode(led_pin,OUTPUT);
  pinMode(alarm_pin,OUTPUT);

  // Wait before starting the alarm
  delay(5000);
}

void loop()
{
  // Motion detected ?
  if (digitalRead(motion_pin)) {
    alarm_mode = true;
  }

  // If alarm mode is on, flash the LED and make the alarm ring
  if (alarm_mode){
    unsigned long currentMillis = millis();
    if(currentMillis - previousMillis > interval) {
      previousMillis = currentMillis;
      if (ledState == LOW)
        ledState = HIGH;
      else
        ledState = LOW;
    // Switch the LED
    digitalWrite(led_pin, ledState);
    }
    tone(alarm_pin,1000);
  }
}
```

Let's now see the details of this code. It starts by declaring the pins to which the different components are connected to:

```
const int alarm_pin = 8;
const int led_pin = 5;
const int motion_pin = 7;
```

We will store the fact that the alarm is on or not inside a variable:

```
boolean alarm_mode = false;
```

We will also have a variable to make the LED flash when the alarm is on:

```
int ledState = LOW;
long previousMillis = 0;
long interval = 100;  // Interval at which to blink (milliseconds)
```

Now, inside the setup() function of the sketch, we need to set the pins for the LED and the Piezo as outputs:

```
pinMode(led_pin,OUTPUT);
pinMode(alarm_pin,OUTPUT);
```

We also wait for 5 seconds, so the alarm doesn't turn on right away:

```
delay(5000);
```

In the loop() function of the sketch, we continuously check the state of the PIR motion sensor. If some motion has been detected, we set the alarm variable to true:

```
if (digitalRead(motion_pin)) {
  alarm_mode = true;
}
```

Now, if the alarm is on, we do two things: continuously flash the LED and initiate the Piezo buzzer to make some noise. This is done by the following piece of code:

```
if (alarm_mode){
  unsigned long currentMillis = millis();
  if(currentMillis - previousMillis > interval) {
    previousMillis = currentMillis;
    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;

    // Switch the LED
    digitalWrite(led_pin, ledState);
    }

    tone(alarm_pin,1000);
}
```

You can now test the project. Upload the code to the Arduino board using the Arduino IDE. Try waving your hand in front of the sensor after the initial 5 seconds delay has passed. You should hear the alarm turn on and see the LED flashing continuously. To turn it off again, simply press the red reset button on the Arduino board.

This is already the end of this guide about open-source home automation! I hope this simple project gave you an idea of what you can do with Arduino for home automation applications.

If you now want to learn more about building home automation systems with Arduino, you can go further by reading the book that I wrote about building a wireless security camera with Arduino:

Building a Wireless Security Camera with Arduino

And if that’s not done yet, you can of course follow my website on Facebook & on Twitter.

Thanks again, and all the best for your home automation projects!

Marco Schwartz
contact@openhomeautomation.net