**Automatic Air Freshener with ATtiny85, PIR Motion Sensor, and Atomizer Humidifier**

**Table of Contents**

1. Introduction
2. Components Required
3. Circuit Diagram
4. Code Explanation
5. Assembly Instructions
6. System Requirements
7. Power and Current Calculations
8. Testing and Calibration
9. Troubleshooting

**1. Introduction**

This project involves creating an innovative automatic air freshener system using an ATtiny85 microcontroller, a PIR motion sensor, and an atomizer humidifier. The system is designed to enhance indoor air quality and user comfort by automatically dispensing a fragrant mist when motion is detected, providing an energy-efficient and low-maintenance solution suitable for diverse indoor settings.

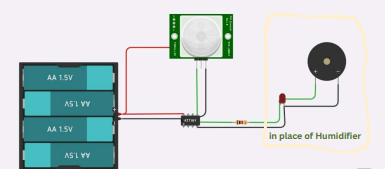
**Primary Objectives:**

* Enhance indoor air quality and user comfort.
* Provide an energy-efficient, low-maintenance solution.
* Enable deployment in diverse indoor settings.
* Automatically dispense fragrant mist only when the space is occupied.
* Implement timed release and cool down cycles.
* Use a low-power Tiny85 microcontroller.
* Ensure a compact, discreet design adaptable to room size and conditions.

**2. Components Required**

* ATtiny85 microcontroller
* PIR motion sensor
* Atomizer humidifier
* LED (for indication)
* Resistors (220Ω recommended for LED)
* Breadboard and jumper wires
* Power supply (3.7V Li-ion battery suggested)

**3. Circuit Diagram(!need update)**

****

**4. Code Explanation**

The code for this project detects motion using a PIR sensor and activates an LED (which can be replaced with the atomizer humidifier).

* **Constants**: motionPin, ledPin, activeTime, cooldownTime, and stabilityDelay are defined as constants for better code readability and maintenance.
* **setup():** Initializes the pins for the motion sensor and LED. A delay is added to allow the PIR sensor to stabilize after power-up.
* **loop():** Continuously checks the motion sensor. If motion is detected:
* The LED (or atomizer) is turned on for activeTime milliseconds.
* The LED (or atomizer) is then turned off, and the system waits for cooldownTime milliseconds before checking for motion again.
* A short stabilityDelay is added to ensure the system remains stable and avoids multiple triggers from a single motion event.

**5. Assembly Instructions**

1. **Connect the PIR Motion Sensor**:
   * VCC to 5V
   * GND to Ground
   * OUT to PB0 (Pin 5 on ATtiny85)
2. **Connect the LED (or Atomizer Control)**:
   * Anode (long leg) to PB1 (Pin 6 on ATtiny85) through a current-limiting resistor (220Ω recommended)
   * Cathode (short leg) to Ground
3. **Power the ATtiny85**:
   * Connect VCC to 5V
   * Connect GND to Ground
4. **Upload the Code**:
   * Use an Arduino as an ISP or another programmer to upload the code to the ATtiny85.

**6. System Requirements**

* **Detect presence of people using a motion sensor.**
* **Activate the atomizer/humidifier for 20 seconds when motion is detected.**
* **Automatically turn off the system for 15 minutes, even if motion is still detected.**
* **Operate using a low-power microcontroller (Tiny85).**

**7. Power and Current Calculations**

* **Power Consumption Calculation**:
  + Tiny85 MCU: 5 mA
  + PIR Motion Sensor: 100 μA (standby mode)
  + Atomizer/Humidifier: 500 mA (during activation)
  + Total current draw during atomization: 505 mA
  + Total current draw during standby: 5.1 mA

**8. Testing and Calibration**

* **Power On**: Once everything is connected, power on the system.
* **Motion Detection**: Move in front of the PIR sensor and observe the LED/atomizer.
* **Calibration**: Adjust the PIR sensor sensitivity if necessary.

**9. Troubleshooting**

* **No Response from Sensor**: Check connections, ensure the sensor is powered.
* **LED/Atomizer Doesn't Activate**: Verify the code is uploaded correctly, check connections.
* **Intermittent Activation**: Increase the stability delay or adjust PIR sensor sensitivity.
* **False Triggers:**

**If the device activates without detecting motion, there could be several causes:**

1. **Noise or Interference on the PIR Sensor**:
   * Ensure the PIR sensor is properly connected and not picking up electrical noise from nearby components.
   * Add a capacitor (e.g., 10µF) across the power supply pins of the PIR sensor to filter out noise.
2. **Incorrect Sensor Initialization**:
   * Add a delay in the setup() function to allow the PIR sensor to stabilize:

void setup() {

pinMode(motionPin, INPUT);

pinMode(ledPin, OUTPUT);

delay(30000); // Wait for 30 seconds to allow the PIR sensor to stabilize

}

1. **Default State of the Motion Pin**:
   * Use the internal pull-up resistor on the motion pin to ensure it has a default state:

void setup() {

pinMode(motionPin, INPUT\_PULLUP);

pinMode(ledPin, OUTPUT);

delay(30000); // Wait for 30 seconds to allow the PIR sensor to stabilize

}

1. **Sensitivity or Range Settings on the PIR Sensor**:
   * Adjust the sensitivity and range settings on the PIR sensor, usually done via potentiometers on the sensor module.