Report

Arduino-Based CO2 Monitoring and Ventilation Control System

Content

Introduction

Arduino-Based CO2 Monitoring and Ventilation Control System

In response to the growing concerns regarding indoor air quality and the impact of elevated carbon dioxide (CO2) levels on health and well-being, there is an increasing need for effective ventilation systems. In this context, the development of a reliable and intelligent system for monitoring CO2 concentrations and controlling ventilation becomes paramount. This report presents an Arduino-based solution that leverages the capabilities of the MQ135 sensor, a servo motor, and an electronic speed controller (ESC) to create a responsive and automated CO2 monitoring and ventilation control system.

Indoor spaces, especially those with inadequate ventilation, can experience elevated CO2 levels, which may lead to discomfort, fatigue, and potential health risks. The implemented system aims to address this issue by continuously measuring CO2 concentrations and adjusting ventilation accordingly. The core components of the system include an MQ135 sensor for accurate CO2 measurement, a servo motor for controlling ventilation, and an ESC to modulate the motor's speed based on the measured CO2 levels.

This report will delve into the details of the hardware components used, the functionality of the Arduino code, and the calibration processes involved. Additionally, insights into the potential applications and adaptability of the system will be discussed. The goal is to provide a comprehensive understanding of the design and implementation of the Arduino-based CO2 monitoring and ventilation control system, offering a versatile solution for enhancing indoor air quality in various settings.

Components

Here is a list of components used in the Arduino-based CO2 monitoring and ventilation control system described in the provided code:

1. Arduino Board:

- The brain of the system, responsible for running the code and controlling the connected components.

2. MQ135 Sensor:

- An analog gas sensor capable of detecting various gases, including carbon dioxide (CO2).

3. Bldc Motor:

- A motor that can be precisely controlled for rotational movement. In this system, it is used to control the Speed of the motor.

4. Electronic Speed Controller (ESC):

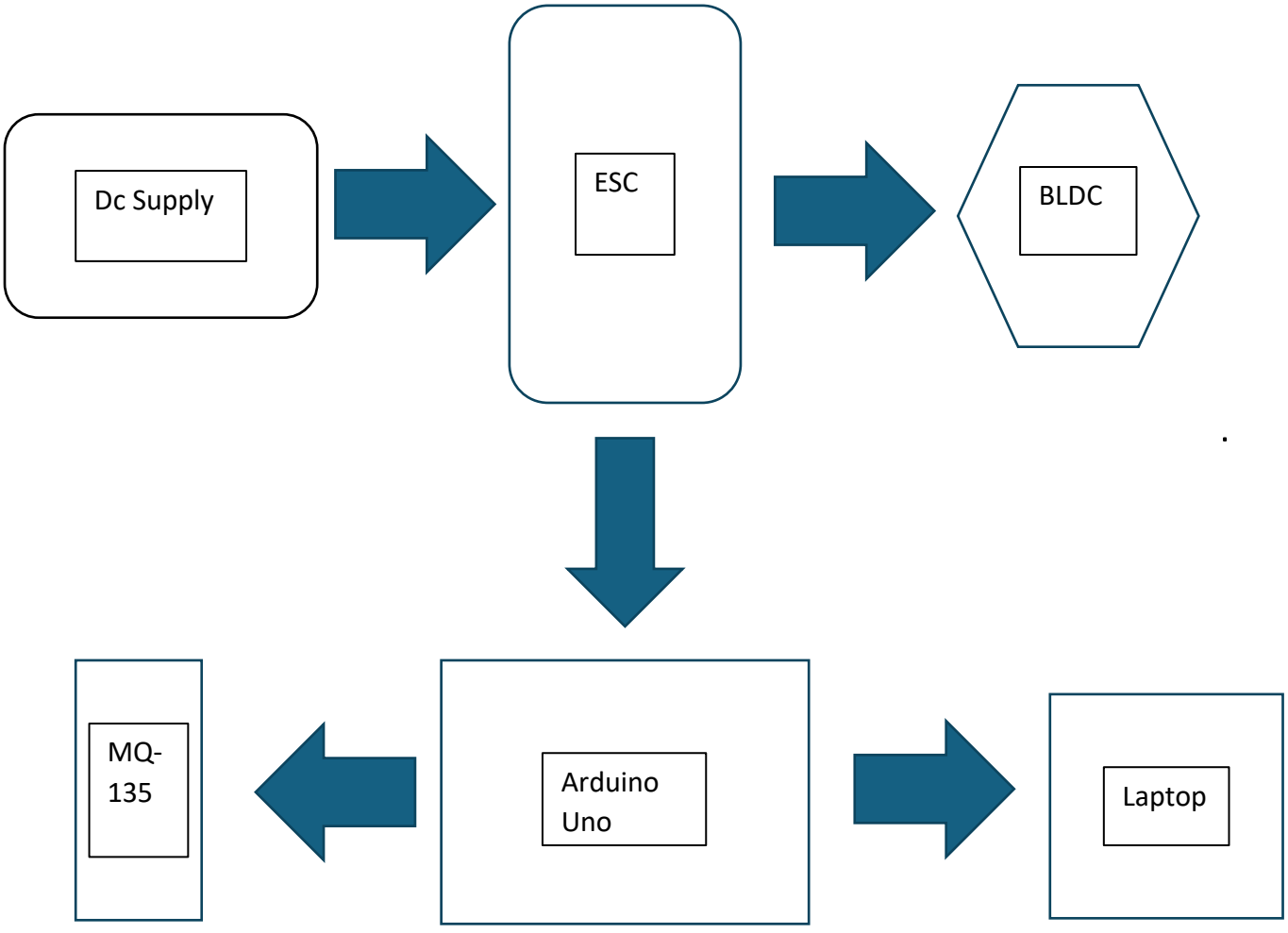
- A device that regulates the speed of an electric motor, typically used in conjunction with a bldc motor.

5. Power Supply:

- A suitable power supply for the Arduino board, MQ135 sensor, servo motor, and ESC. Ensure that the voltage and current requirements of each component are met.

6. Wiring and Connectors: - Various wires and connectors to establish electrical connections between the components.

Block Diagram



Circuit

MQ135 Sensor to Arduino:

- Connect the analog output pin of the MQ135 sensor to Arduino's analog input A0.

BLDC Motor and ESC to Arduino:

- Connect the output wires of the Electronic Speed Controller (ESC) to the BLDC motor. The number of wires will depend on your specific BLDC motor and ESC. It's common to have three wires for power (Vcc, GND) and signal.
- Connect the signal wire of the ESC to a PWM-capable pin on the Arduino. Choose a suitable pin and update the code accordingly.
- Connect the power (Vcc) and ground (GND) wires of the ESC to the appropriate power and ground pins on the Arduino.

Power Supply:

- Ensure that the power supply can provide the required voltage and current for both the Arduino and the BLDC motor through the ESC.

Computer with Arduino IDE:

- Connect the Arduino board to a computer using a USB cable for programming and power.

Code

MQ135BLDC_Final.ino

Arduino Uno (arduino:avr:uno)
COM4

```
#define MQ_PIN A0 // Define the analog pin for MQ135 sensor
#define RL 10.0    // Load resistance in ohms
#define R0 76.63   // Sensor resistance in clean air

#define ESC_PIN 9   // Connect the signal wire of your ESC to pin 9 on Arduino

void WarmingSensor();
void RunMotorForInitialization();

void setup() {
  pinMode(ESC_PIN, OUTPUT);
  Serial.begin(9600);

  // Display information on Serial Monitor
  Serial.println("BLDC Motor Control Test");

  // Run the motor for initialization
  RunMotorForInitialization();

  // Warm up the sensor
  WarmingSensor();
}

void loop() {
  int sensorValue = analogRead(MQ_PIN);   // Read analog sensor value
  float Vout = sensorValue * (5.0 / 1023.0); // Convert to voltage (assuming 5V Arduino)

  // Calculate Rs using the provided formula
  float Rs = ((5.0 * RL) - (RL * Vout)) / Vout;

  // Calculate ratio
  float ratio = Rs / R0;

```

MQ135BLDC_Final.ino

```
30
31      // Calculate ratio
32      float ratio = Rs / R0;
33
34      // Apply the specific CO2 calibration formula
35      ratio = ratio * 0.3611;
36      float CO2_PPM = 146.15 * (2.868 - ratio) + 10;
37
38      // Print the sensor values and CO2 concentration to the Serial Monitor
39      Serial.print("CO2 PPM: ");
40      Serial.println(CO2_PPM, 2); // Print CO2 PPM with 2 decimal places
41
42      // Motor control based on CO2 concentration after initialization
43      if (millis() > 5000) {
44        if (CO2_PPM >= 100 && CO2_PPM <= 300) {
45          // Map CO2 range to PWM range (155 to 195)
46          int pwmValue = map(CO2_PPM, 100, 300, 171, 195);
47          analogWrite(ESC_PIN, pwmValue);
48          Serial.print("PWM Value: ");
49          Serial.println(pwmValue);
50        } else {
51          analogWrite(ESC_PIN, 0); // Stop the motor if CO2 is outside the desired range
52          Serial.println("Motor stopped.");
53        }
54      }
55
56      delay(1000); // Adjust the delay based on your sampling requirements
57    }
```

MQ135BLDC_Final.ino

```
52          Serial.println("Motor stopped.");
53        }
54      }
55
56      delay(1000); // Adjust the delay based on your sampling requirements
57    }
58
59    void WarmingSensor() {
60      Serial.println("Warming up sensor...");
61      delay(3000);
62      for (int i = 0; i <= 100; i++) {
63        Serial.print("Warming Sensor: ");
64        if (i < 100) Serial.print(" ");
65        if (i < 10) Serial.print(" ");
66        Serial.print(i);
67        Serial.println("%");
68        delay(100);
69      }
70    }
71
72    void RunMotorForInitialization() {
73      Serial.println("Running motor for initialization...");
74
75      // Gradually increase PWM from 173 to 180
76      for (int pwmValue = 160; pwmValue <= 180; pwmValue++) {
77        analogWrite(ESC_PIN, pwmValue);
78        delay(50);  // Adjust the delay based on your motor response time
79      }
80
81      delay(5000); // Wait for 5 seconds after reaching PWM 180
82
83      analogWrite(ESC_PIN, 0); // Stop the motor after initialization
84      Serial.println("Initialization complete.");
85    }
86
87
```

Conclusion

The Arduino-based CO2 monitoring and ventilation control system, integrating the MQ135 sensor, Brushless DC (BLDC) motor, and Electronic Speed Controller (ESC), stands out as a promising solution to elevate indoor air quality. The MQ135 sensor ensures precise CO2 monitoring, crucial for assessing indoor air conditions. Noteworthy is the substitution of the conventional servo motor with a more versatile BLDC motor, controlled by the ESC, providing dynamic adaptability to varying ventilation needs. System performance hinges on accurate sensor calibration and optimal motor and ESC configuration. Its broad applications, spanning residential, commercial, and industrial sectors, underscore its potential to enhance occupant well-being, energy efficiency, and contribute to healthier indoor environments. In conclusion, the system's intelligent design positions it as an accessible solution for sustainable and healthier indoor spaces, with future refinements poised to amplify its capabilities.