The **LM35** is a precision temperature sensor that provides an output voltage linearly proportional to the temperature in degrees Celsius. It is widely used due to its accuracy, ease of use, and ability to measure temperature without the need for calibration or any external components.

**Key Features of LM35:**

* **Linear output**: The output voltage of the LM35 increases by 10 mV per degree Celsius, making it simple to calculate the temperature.
* **Range**: The LM35 can measure temperatures in the range of −55°C to +150°C.
* **High accuracy**: ±0.5°C at room temperature and better accuracy at higher temperatures.
* **Low power consumption**: It operates at very low current, typically 60 µA.
* **Operates without external calibration**.
* **Low self-heating**: Ensures that the sensor’s own heat generation does not significantly affect the temperature measurement.

**How It Works:**

The LM35 outputs a voltage that is linearly proportional to the temperature. For example:

* At 0°C, the output voltage is 0V.
* At 25°C, the output voltage is 250 mV.
* At 100°C, the output voltage is 1V.

**Pin Configuration:**

The LM35 is typically available in a 3-pin TO-92 package (similar to a transistor) or in other packages like SOIC.

* **Pin 1 (Vcc)**: Power supply, typically 4-20V.
* **Pin 2 (Vout)**: The output voltage, which is proportional to the temperature.
* **Pin 3 (GND)**: Ground.

**LM35 in a Circuit:**

To use the LM35, you connect the Vcc pin to a 5V power supply, the GND pin to ground, and the Vout pin to an analog input on a microcontroller (e.g., Arduino, Raspberry Pi, etc.). The microcontroller can then read the voltage and convert it to temperature.

Temperature(°C)= Vout(inmV)/10

**MiCS-VZ-89TE CO2**

The **MiCS-VZ-89TE CO2 and VOC sensor** operates on the principle of metal-oxide gas sensing combined with advanced signal processing to measure the concentration of carbon dioxide (**CO2**) equivalents and volatile organic compounds (**VOCs**) in the air. Here’s a breakdown of the working principle:

**1. Metal-Oxide Sensing Element:**

At the core of the MiCS-VZ-89TE is a **metal-oxide semiconductor** sensing element, which is sensitive to changes in air quality due to gases like VOCs and CO2-equivalents.

* **Metal-oxide gas sensors** work by utilizing a **sensitive layer of metal-oxide materials**, typically tin dioxide (SnO₂), on the sensor’s surface. When VOCs or CO2-equivalents are present in the air, they react with the oxygen ions on the sensor's surface, causing a change in the sensor's electrical resistance.
* As more VOCs or CO2-equivalents are detected, the **sensor resistance changes** in proportion to the concentration of the gases.

**2. Sensing Process:**

* **VOCs** (volatile organic compounds) are emitted from materials like paints, cleaning products, and adhesives, and their levels in the air can fluctuate due to human activity, environmental conditions, and air circulation.
* **CO2-equivalents** are detected indirectly through the correlation between indoor CO2 levels and the accumulation of VOCs. The sensor provides an **estimated CO2 concentration** based on the VOCs detected because both CO2 and VOC levels increase due to human presence, cooking, smoking, or using cleaning products indoors.

When these gases interact with the metal-oxide sensing material, the sensor’s resistance changes. This change is proportional to the concentration of gases present in the environment.

**3. Signal Processing:**

The MiCS-VZ-89TE sensor includes an **internal microcontroller** that processes the resistance changes from the metal-oxide sensing element.

* The raw data from the sensor is processed through calibration and correction algorithms, ensuring that the sensor provides accurate and reliable readings.
* The sensor outputs two key values:
  + **CO2-equivalent concentration (ppm)**: This gives an estimate of the CO2 level in the air in **parts per million**.
  + **VOC index**: This index represents the air quality in terms of the detected VOC concentration. A higher index indicates poorer air quality.

**4. I²C Interface for Output:**

The MiCS-VZ-89TE sensor communicates with external devices (such as microcontrollers or computers) through an **I²C interface**. This digital interface simplifies the communication and integration of the sensor into various systems.

* The sensor sends the **CO2-equivalent** and **VOC index** as digital data via I²C. The microcontroller can then read this data and use it to display air quality information or take actions, such as adjusting ventilation systems or sending alerts when air quality drops below a threshold.

**5. Self-Calibration and Adaptation:**

* The MiCS-VZ-89TE features **internal self-calibration** to ensure long-term stability and accuracy in its measurements.
* The sensor can adjust to changes in its environment over time, ensuring that the outputs remain accurate even as it ages or as environmental conditions fluctuate.

**Summary of Working Principle:**

1. **Sensing Mechanism**: The metal-oxide semiconductor detects changes in resistance when exposed to VOCs and CO2-equivalents.
2. **Data Processing**: An internal microcontroller processes the raw data from the sensor and converts it into meaningful outputs (CO2-equivalent and VOC index).
3. **I²C Communication**: The processed data is transmitted over the I²C interface for easy integration into air quality monitoring systems.

**Pin Configuration:**

* **Pin 1 (VDD)**: Power supply (3.3V to 5V).
* **Pin 2 (GND)**: Ground.
* **Pin 3 (SDA)**: I²C data line.
* **Pin 4 (SCL)**: I²C clock line.

**Pin Connections:**

1. **VDD (Pin 1)** (MiCS-VZ-89TE) to **3.3V** (Tiva C).
2. **GND (Pin 2)** (MiCS-VZ-89TE) to **GND** (Tiva C).
3. **SDA (Pin 3)** (MiCS-VZ-89TE) to **PB3 (I2C0SDA)** (Tiva C).
4. **SCL (Pin 4)** (MiCS-VZ-89TE) to **PB2 (I2C0SCL)** (Tiva C).

**CODE – TEMPERATURE SENSOSR**

int sensorPin = A0; // Analog input pin that LM35 is connected to

int sensorValue = 0; // Variable to store the sensor value

float temperature = 0; // Variable to store calculated temperature

void setup() {

Serial.begin(9600); // Initialize serial communication

}

void loop() {

sensorValue = analogRead(sensorPin); // Read the analog input

// Convert the ADC value to voltage (assuming a 5V reference voltage)

float voltage = sensorValue \* (5.0 / 1024.0);

// Convert the voltage to temperature in Celsius

temperature = voltage \* 100.0; // 10mV per degree Celsius

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.println(" °C");

delay(1000); // Wait for a second before taking the next reading

}

**CODE CO2 SENSOR**

#include <Wire.h>

// MiCS-VZ-89TE I2C address

#define MICS\_VZ\_89TE\_ADDR 0x70 // Default address of the sensor

void setup() {

Serial.begin(115200); // Start serial communication for debugging

Wire.begin(); // Start I2C communication

delay(2000); // Delay for sensor warm-up

Serial.println("MiCS-VZ-89TE CO2 and VOC Sensor Initialized");

}

uint8\_t readSensor(uint8\_t reg) {

Wire.beginTransmission(MICS\_VZ\_89TE\_ADDR); // Start communication with sensor

Wire.write(reg); // Send register address to read

Wire.endTransmission(false); // Send stop condition but keep connection active

Wire.requestFrom(MICS\_VZ\_89TE\_ADDR, 1); // Request 1 byte from sensor

return Wire.read(); // Read and return the byte

}

void loop() {

// Read CO2 equivalent value

uint8\_t co2 = readSensor(0x08); // Placeholder register address for CO2 equivalent

// Read VOC index value

uint8\_t voc = readSensor(0x09); // Placeholder register address for VOC index

// Print sensor readings

Serial.print("CO2 Equivalent: ");

Serial.print(co2); // CO2 value in ppm

Serial.println(" ppm");

Serial.print("VOC Index: ");

Serial.println(voc); // VOC index value

// Wait before next reading

delay(1000); // 1 second delay between readings

}

Sound sensor

**How the Micro Electret Sensor Works**

The **Micro Electret Condenser Microphone** is a type of **capacitor microphone** that relies on changes in capacitance to detect sound waves. Here's a breakdown of its working principle:

**Working Principle:**

1. **Electret Material**: The microphone uses an **electret material**, which is a dielectric material with a permanent electrostatic charge. This material is placed between two plates (forming a capacitor). One plate is the diaphragm (a thin, flexible metal or plastic membrane), and the other is a backplate.
2. **Sound Waves**: When sound waves hit the diaphragm, it vibrates. These vibrations cause the distance between the diaphragm and the backplate to change, altering the capacitance of the capacitor formed by these two components.
3. **Capacitance Changes**: The changing capacitance creates small voltage variations due to the **electrostatic charge** on the electret material. These voltage variations are the microphone's output signal.
4. **FET Amplifier**: Because the generated signal is quite small, the electret microphone includes an internal **field-effect transistor (FET)** amplifier to boost the signal to a more usable level.
5. **DC Bias Voltage**: To operate the internal FET, the microphone needs a small **DC bias voltage** (typically between 1.5V to 5V, depending on the microphone). This bias voltage powers the FET, allowing the microphone to output a signal that can be read by an external circuit or a microcontroller.

**ii. How to Make the Micro Electret Microphone Work**

To make the **Micro Electret Microphone (AMB-707-RC)** work, you need to:

1. **Connect the Microphone Properly**: The microphone has three terminals:
   * **V+**: Power supply pin (typically 1.5V to 5V, depending on the microphone).
   * **GND**: Ground.
   * **Output**: The microphone's signal output pin, where the amplified audio signal is available.
2. **DC Biasing**: Since the microphone requires a DC bias to operate, you must supply power to it through a **biasing resistor**. A typical value is **2.2kΩ to 10kΩ** depending on the desired gain. This resistor limits the current to the internal FET, and the audio signal can be AC-coupled through a **capacitor** to remove the DC component.
3. **Amplification**: Even though the microphone has a built-in FET amplifier, the output signal is still relatively weak. To properly capture sound, you may need an external amplifier, such as an **operational amplifier** (op-amp) or an **audio amplifier** like the **LM386**.

**Circuit Example:**

Here’s how you can connect the **Micro Electret Microphone** to a microcontroller (like Arduino or Tiva C) for sound sensing:

**Circuit Connections:**

* **V+** (microphone power) to a 5V supply (through a biasing resistor).
* **GND** to ground.
* **Output** to an analog input pin of the microcontroller (through a capacitor).

**Basic Schematic:**

scss

Copy code

+5V

|

R (Bias resistor, 2.2kΩ to 10kΩ)

|

------- Microphone (AMB-707-RC)

| |

GND OUT ------------ [Capacitor (10µF)] --------- Analog Input (e.g., A0 on Arduino)

|

GND

* The **bias resistor** (R) provides the necessary DC voltage to power the microphone.
* The **capacitor** (typically 10µF) blocks the DC bias voltage and only allows the AC audio signal to pass through to the analog input.

**iii. Voltage Generated by the Sensor**

The **voltage output** of the microphone depends on the intensity of the sound and the biasing conditions. Here's an estimate of what you might observe:

* **Idle Condition**: When there is no sound, the microphone will output a small DC voltage (corresponding to the bias voltage divided by the bias resistor).
* **Sound Present**: When sound is present, the microphone will generate small AC voltage fluctuations on top of the DC bias voltage. The amplitude of these fluctuations depends on the sound's loudness:
  + For typical conversational speech, the microphone might output a signal in the range of **millivolts** (mV), such as **10 mV to 50 mV**.
  + For louder sounds (such as shouting), the output signal can reach up to **100 mV or more**.

**Example: Arduino Code to Measure Sound Level**

Here’s an example of code you can use to measure the microphone output with **Arduino** or Energia (for Tiva C).

cpp

Copy code

int micPin = A0; // Microphone connected to analog pin A0

int micValue = 0;

void setup() {

Serial.begin(115200); // Start serial communication for output

}

void loop() {

micValue = analogRead(micPin); // Read microphone value

Serial.println(micValue); // Print microphone value to serial monitor

delay(100); // Small delay for readability

}

**Steps to Measure Voltage:**

1. **Upload the code** to your Arduino or Tiva C using Energia.
2. **Open the Serial Monitor** (in Arduino IDE or PuTTY) to see the sound intensity as a raw analog value (0 to 1023, corresponding to 0V to 5V for Arduino or 0V to 3.3V for Tiva C).
3. **Calculate Voltage**: If you want to convert the raw analog value to a voltage reading, you can use the following formula:

cpp

Copy code

float voltage = (micValue \* 5.0) / 1023.0; // For Arduino (5V ADC range)

// For Tiva C (3.3V ADC range), use:

// float voltage = (micValue \* 3.3) / 4095.0; // Assuming a 12-bit ADC on Tiva

**Measured Voltage:**

* The voltage generated by the microphone will typically be in the **millivolt range** when detecting normal sounds.
* For example, a **loud clap** might generate a peak voltage of **50 mV to 100 mV**, while **background noise** will generate much smaller signals, around **10 mV** or less.

**Applications:**

* **Voice detection** or **sound level detection** for controlling devices like lights, alarms, or recorders.
* **Audio recording** in conjunction with an amplifier circuit.

**DC BIASING**

DC biasing is the process of applying a steady DC voltage to an electronic component, such as a transistor, diode, or microphone, to ensure it operates in the correct region of its functionality. In the case of sensors and especially electret microphones, DC biasing is crucial to power the internal amplifier and ensure proper signal processing.

In the Context of an Electret Microphone:

An electret microphone like the Micro Electret Condenser Microphone (AMB-707-RC) has an internal Field-Effect Transistor (FET) amplifier that boosts the weak signal generated by the microphone's diaphragm. This internal FET requires a small DC voltage (typically between 1.5V to 5V) to operate, and this is provided via DC biasing.

Without this DC bias, the microphone would not work correctly because the FET amplifier would not have the necessary power to amplify the sound signal.

How DC Biasing Works in an Electret Microphone:

DC Voltage Application: A DC voltage is applied to the microphone through a bias resistor. This resistor allows a small amount of current to flow through the microphone's internal FET amplifier.

Signal Extraction: The microphone generates an AC signal (the sound wave). Since the DC bias is applied, the output signal contains both the DC voltage and the AC audio signal. We only want the AC part (sound signal), so we use a capacitor to block the DC voltage and allow only the AC audio signal to pass.

Basic Circuit for DC Biasing (Electret Microphone Example)

Components Needed:

Micro Electret Microphone (AMB-707-RC or similar).

Resistor (2.2kΩ to 10kΩ, depending on the microphone).

Capacitor (typically 10µF).

Power supply (usually 3.3V to 5V).

Microcontroller or amplifier to read the output.

Circuit Diagram:

plaintext

Copy code

+5V

|

R (Bias resistor, 2.2kΩ to 10kΩ)

|

------- Microphone (AMB-707-RC)

| |

GND OUT ------------ [Capacitor (10µF)] --------- Analog Input (e.g., A0 on Arduino)

|

GND

Explanation:

Bias Resistor (R): This resistor provides the necessary current to power the internal FET amplifier of the microphone. Typical values range from 2.2kΩ to 10kΩ.

Capacitor: The capacitor is used to block the DC bias voltage so that only the AC audio signal passes to the next stage (which could be a microcontroller, an amplifier, or a speaker system).

Why is DC Biasing Important?

Ensures Operation of FET Amplifier: The FET inside the electret microphone needs a DC bias voltage to operate. Without this bias, the microphone would not produce a usable output.

Prevents Saturation: Without proper biasing, the output signal may get clipped or distorted, especially when dealing with higher sound pressure levels.

Signal Isolation: The capacitor ensures that the DC component is blocked, allowing only the relevant AC sound signal to pass through for further processing.

What Happens Without Proper DC Biasing?

If no DC bias is applied, or if the wrong value of bias resistor is used, the microphone may produce no output or a very weak signal.

The microphone may also become unstable, leading to distorted audio or nonlinear response to sound inputs.

Practical Example with Arduino (Energia for Tiva C)

If you’re using an electret microphone with a microcontroller like Arduino or Tiva C, you’ll connect the microphone’s output (after DC biasing) to one of the analog pins (e.g., A0), and the microcontroller can then read the audio signal and process it.

Here’s a basic example for reading the microphone’s AC signal (sound intensity):

cpp

Copy code

int micPin = A0; // Microphone output connected to analog pin A0

int micValue = 0; // Variable to store the analog reading

void setup() {

Serial.begin(115200); // Start serial communication for monitoring

}

void loop() {

micValue = analogRead(micPin); // Read microphone value (AC signal)

Serial.println(micValue); // Print the value (range: 0 to 1023 for 10-bit ADC)

delay(100); // Small delay for readability

}

In this example:

DC Biasing is done externally using the resistor and capacitor.

The analog input pin reads the microphone's AC signal, which is the sound data, and prints it to the serial monitor.