Problem statement: Develop the code in Arduino to interface an LM35 temperature sensor (Assume the sensor is connected to Arduino Analog pin A0) with an Arduino Uno and control the onboard LED based on temperature readings:

1.Interface the LM35 temperature sensor with an Arduino Uno.
2.When the temperature falls below 30 degrees Celsius, make the onboard LED blink every 250 milliseconds.
3.If the temperature rises above 30 degrees Celsius, make the onboard LED blink every 500 milliseconds.

Note :- You are not supposed use Millis(), delay() and micros() function and you can use any library to perform this task.

CODE:

Here's the Arduino code to interface an LM35 temperature sensor with Arduino Uno and control the onboard LED based on temperature readings, without using `millis()`, `delay()`, or `micros()` functions:

```c++
#include <TimerOne.h> // Include TimerOne library

const int sensorPin = A0; // Analog pin connected to LM35
const int ledPin = 13;    // Onboard LED pin

double temperature;
int blinkInterval = 500; // Initial blink interval (ms)

void setup() {
  pinMode(ledPin, OUTPUT);
  Timer1.initialize(250); // Set timer for initial blink interval
  Timer1.attachInterrupt(blinkLED); // Attach interrupt for blinking
  Timer1.start();
}

void loop() {
  // Read voltage from LM35
  double voltage = analogRead(sensorPin) * 5.0 / 1023.0;

  // Convert voltage to temperature (Celsius)
  temperature = (voltage - 0.5) * 100;

  // Update blink interval based on temperature
  if (temperature < 30) {
    blinkInterval = 250;
  } else {
```

```
  } else {
    blinkInterval = 500;
  }

  // Timer will handle blinking at the set interval
}

void blinkLED() {
  digitalWrite(ledPin, !digitalRead(ledPin)); // Toggle LED state
  Timer1.write(blinkInterval); // Update timer for next blink
}
```

Explanation:

1. **Include Library:** We include the `TimerOne` library, which allows for generating interrupts at specific intervals without using `delay()`.
2. **Define Pins:** We define `sensorPin` for the LM35 and `ledPin` for the onboard LED.
3. **Global Variables:** We declare `temperature` to store the measured temperature and `blinkInterval` to hold the current blinking interval.
4. **Setup:**
   - `pinMode(ledPin, OUTPUT)`: Sets the LED pin as output.
   - `Timer1.initialize(250)`: Initializes Timer1 with an initial period of 250 microseconds.
   - `Timer1.attachInterrupt(blinkLED)`: Attaches an interrupt to Timer1, which will trigger the `blinkLED` function every time the timer overflows.
   - `Timer1.start()`: Starts Timer1.
5. **Loop:**
   - `voltage = analogRead(sensorPin) * 5.0 / 1023.0`: Reads the voltage from the sensor and converts it to a voltage value between 0 and 5 volts.

- `temperature = (voltage - 0.5) * 100`: Converts the voltage to temperature (°C) using the LM35's calibration factor (10mV/°C).
- **Conditional Statement:** Based on the temperature:
  - If temperature is below 30°C, set `blinkInterval` to 250 (faster blink).
  - Otherwise, set `blinkInterval` to 500 (slower blink).
6. **blinkLED function:**
- `digitalWrite(ledPin, !digitalRead(ledPin))`: Toggles the LED state.
- `Timer1.write(blinkInterval)`: Updates Timer1's period with the new `blinkInterval` value, defining the next blink timing.

Advantages:

- This code avoids using `delay()`, which can block the program execution, making it more efficient and responsive.
- It leverages the `TimerOne` library for accurate timing and allows the program to continue processing other tasks while blinking the LED.