SIT 329 Advanced Embedded Systems

Task 5.1P

Ques 1. Short Introduction on what the embedded systems is supposed to do.

Ans 1: In this embedded system, the code is designed to create a temperature and humidity monitoring device with the following key features:

1. Sensor Reading: It uses a DHT22 sensor to measure temperature and humidity at regular intervals (every 5 seconds).

2. Data Logging: The system logs temperature and humidity readings along with timestamps using a Real-Time Clock (RTC).

3. Temperature-based LED indicator: An LED blinks at different frequencies based on the current temperature:

a. Fast blink (20Hz) when temperature is above 25 degrees.
b. Slow blink (0.5Hz) when temperature is below 10 degrees.
c. Medium blink (1Hz) for temperature between 10-25 degrees.

4. Serial Output: The system prints sensor readings, timestamps, and LED blink frequencies to the serial monitor for monitoring and debugging.

This setup allows for continuous environmental monitoring with a visual indicator (LED) of the current temperature range, making it useful for applications like basic weather stations, greenhouse monitoring, or home automation.

Ques 2: A table detailing the events expected and the frequency of these events if known. Indicate any timing risks in design of your interrupt service routines.

Event	Frequency	Timing Risk
Temperature/Humidity Reading	Every 5 seconds (0.2Hz)	Low Risk: The read interval is set to 5000ms, which should provide ample time for sensor reading and processing.
LED Blinking (High Temperature)	20Hz (when temp > 25)	Moderate Risk: Fast Blinking could interfere with other operations if not managed properly.
LED Blinking (Low Temperature)	0.5 Hz (when temp < 10)	Low risk: This slow blinking should not interfere with other operations.
RTC Timekeeping	Continuous (typically 1Hz)	Low Risk: RTC operations are usually handled by hardware and don't significantly impact CPU time.

Timing Risks in Interrupt Service Routine Design:

1. Sensor Reading Delay: The DHT22 sensor requires a specific timing protocol for communication. If the ’readSensor()’ function takes too long, it could potentially delay other critical operations.

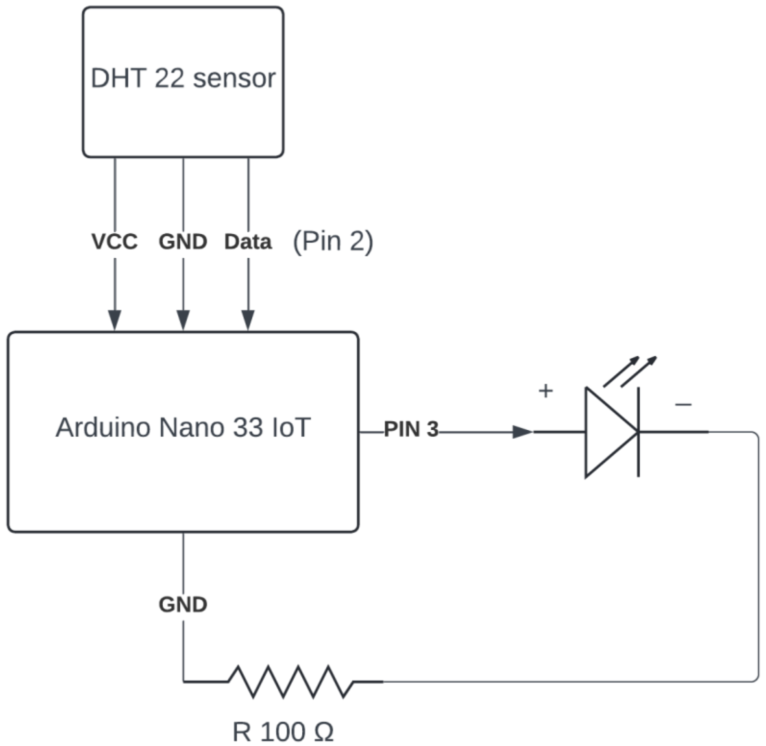
2. Serial Communication: Printing data to the serial monitor within the main loop could introduce delays, especially if large amounts of data are being transmitted frequently.

3. LED Blinking Implementation: The current implementation uses ‘millis()’ for non-blocking LED control, which is good practice. However, at high frequencies (20Hz), it may consume more CPU cycles checking and updating the LED state.

4. Lack of True Interrupts: The code does not use hardware interrupts for sensor reading or LED control. Instead, it relies on polling in the main loop, which could lead to missed events or delayed responses if the loop becomes blocked or delayed.

5. Potential for Long-Running Operations: if additional processing is added to the ‘readSensor()’ function in the future, it could lead to delays in the main loop execution, potentially affecting the LED blinking timing and sensor reading frequency.

Ques 3: Hardware system block diagram showing the connections and components. Indicate which pins are to be used, and any additional electronics required, e.g., resistors.



Ques 5: Code submission

Task5.1_Khushi | Arduino IDE 2.3.0

File Edit Sketch Tools Help

Arduino NANO 33 IoT

Task5.1_Khushi.ino Task5.1_khushi.cpp

```
#include <DHT.h>
#include <RTCZero.h>

// Pin Definitions
#define DHTPIN 2          // Pin where the DHT22 is connected
#define DHTTYPE DHT22     // DHT 22 (AM2302), AM2321
#define LEDPIN 3          // LED Pin (blinks based on temperature)

#define TEMP_HIGH 25      // Temperature threshold for high
#define TEMP_LOW 10       // Temperature threshold for low

DHT dht(DHTPIN, DHTTYPE);
RTCZero rtc;

// Timer settings
const unsigned long readInterval = 5000; // 5 seconds in milliseconds
unsigned long previousMillis = 0;

// LED blink settings
unsigned long ledBlinkInterval = 0; // Interval for LED
bool ledState = false;
unsigned long ledPreviousMillis = 0; // To keep track of LED blink timing

void setup() {
  Serial.begin(9600);
  Serial.println("Starting setup...");

  // Initialize LED pin
  pinMode(LEDPIN, OUTPUT);

  // Initialize DHT sensor
  dht.begin();
```

Task5.1_Khushi | Arduino IDE 2.3.0

File Edit Sketch Tools Help

Arduino NANO 33 IoT

Task5.1_Khushi.ino Task5.1_khushi.cpp

```
  Serial.println("DHT sensor initialized.");

  // Initialize RTC
  rtc.begin();
  rtc.setTime(0, 0, 0); // Set initial time: hours, minutes, seconds
  Serial.println("RTC initialized.");

  Serial.println("Setup complete. Waiting for sensor data...");
}

void loop() {
  unsigned long currentMillis = millis();

  // Check if it's time to read the sensor
  if (currentMillis - previousMillis >= readInterval) {
    previousMillis = currentMillis;
    readSensor();
  }

  // Blink LED based on the interval
  if (ledBlinkInterval > 0) {
    if (currentMillis - ledPreviousMillis >= ledBlinkInterval) {
      ledPreviousMillis = currentMillis;
      ledState = !ledState;
      digitalWrite(LEDPIN, ledState);
    }
  } else {
    digitalWrite(LEDPIN, LOW);
  }
}

void readSensor() {
```

Task5.1_Khushi | Arduino IDE 2.3.0

File Edit Sketch Tools Help

Arduino NANO 33 IoT

Task5.1_Khushi.ino Task5.1_khushi.cpp

```
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Check if readings are valid
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Print the values along with timestamp
  unsigned long currentTime = rtc.getEpoch();
  Serial.print("Timestamp: ");
  Serial.print(currentTime);
  Serial.print(" - Temperature: ");
  Serial.print(temperature);
  Serial.print(" C, Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  // Set the blink interval based on temperature and print the frequency
  if (temperature > TEMP_HIGH) {
    ledBlinkInterval = 50;   // 20Hz -> 50ms interval
    Serial.println("LED Blink Frequency: 20Hz");
  } else if (temperature < TEMP_LOW) {
    ledBlinkInterval = 2000; // 0.5Hz -> 2000ms interval
    Serial.println("LED Blink Frequency: 0.5Hz");
  } else {
    ledBlinkInterval = 1000; // 1Hz -> 1000ms interval
    Serial.println("LED Blink Frequency: 1Hz");
  }
}
```

Ques 6: Video Demonstration

https://youtu.be/2i_29w10H4Y