Code Documentation

I start by including the FreeRTOS library (line 12). In the setup (lines 14-25), I initialize Serial communication, set up tasks for FreeRTOS, and start the scheduler. Since FreeRTOS handles task scheduling, the loop function (line 27) remains empty.

The first task, TaskBlinkExternal (lines 33-55), is responsible for flashing an external LED for 100ms on and 200ms off, using `vTaskDelay`. The function includes a `void*` parameter (line 45) required by FreeRTOS but unused in the tasks, and casting it to `void` suppresses compiler warnings about unused variables. This line is used in all functions passed to `xTaskCreate`, so I don't explain it again for other tasks.

Next, I define an interrupt to drive the four-digit seven-segment display (lines 73-98), which prevents display flickering by updating the display without delay. The interrupt uses `num_to_display` (line 61) to display the temperature, replacing the last digit with an "F" for Fahrenheit. The `display_digit` function (lines 100-122) remains the same as in Lab 3. TaskDisplayNumber (lines 124-153) sets up pinModes and starts the interrupt timer. After setup, this task is no longer needed and calls `vTaskDelete` to remove itself from the task list.

The most complex task, TaskReadSensor (lines 155-213), communicates with the DHT11 sensor via a two-way serial interface. I use the datasheet as a reference to implement this function, starting with a 1-second delay, as recommended by the datasheet. The task reads sensor data by sending a request pulse, waiting for the response, and reading the sensor's data signal. Comments provide a high-level overview of the protocol. The code between `cli` and `sei` is a critical section due to the protocol's microsecond-level timing sensitivity. After reading, I verify the checksum, and if correct, I set `num_to_display` to the computed temperature, which the interrupt handler will display. Three helper functions—`read_sensor_byte` (lines 215-232), `read_sensor_bit` (lines 234-255), and `wait_for_sensor` (lines 257-269)—simplify the serial protocol implementation.

To conclude, I have two simple tasks. TaskTurnOnFan (lines 271-292) checks if the temperature exceeds `fan_temp` and turns the fan on if necessary. TaskSetFanTemp (lines 294-320) adjusts the `fan_temp` threshold by reading a value from a potentiometer (or any other analog signal) and setting `fan_temp` between `min_temp` and `max_temp`. This gives the user control over the temperature at which the fan activates.