**Koneru Lakshmaiah Education Foundation**

**(Deemed to be University)**

**Course Code: 22SDEC06R**

**Course Name: Embedded Design**

**(Skill Development Project-3)**

**A Project Report**

**On**

**Free RTOS LED BLINKY USING STM32**

**SUBMITTED BY:**

2200049155 P GOVARDHAN

2200049167 D VENU

2200049164 N VINAY

2200049152 PAVAN

**UNDER THE GUIDANCE OF**

**P SIVANI MADAM**

**Associate Professor-ECE, HoD-EL&GE**



Green fields, Vaddeswaram – 522 502

Guntur Dist., AP, India.

**K L E F**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**(DST-FIST Sponsored Department)**



**DECLARATION**

The Project Report entitled **“Free RTOS LED BLINKY USING STM32”** is a record of bonafide work of **2200049155 P GOVARDHAN,2200049167 D VENU, 2200049164 N VINAY, 2200049152 PAVAN.** submitted in partial fulfillment for the award of B.Tech in Electronics and Communication Engineering to the K L University. The results embodied in this report have not been copied from any other departments/University/Institute.

**K L E F**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**(DST-FIST Sponsored Department)**



**CERTIFICATE**

This is to certify that the Project Report entitled “**Free RTOS LED blinky using stm32**” is being submitted **2200049155 P GOVARDHAN, 2200049167 D VENU, 2200049164 N VINAY,2200049152 PAVAN** submitted in partial fulfilment for the award of B.Tech in  Electronics and Communication Engineering to the K L University is a record of bonafide work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/ University/Institute.

Signature of Supervisor

Signature of the HOD Signature of the External Examiner

## ACKNOWLEDGMENT

It is great pleasure for me to express my gratitude to our honourable President **Sri. Koneru Satyanarayana**, for giving the opportunity and platform with facilities in accomplishing the project report.

I express the sincere gratitude to our Vice Chancellor, **Dr G P S Varma** for his administration towards our academic growth.

I express the sincere gratitude to our Pro VC, **Dr N Venkatram** for his administration towards our academic growth.

I record it as my privilege to deeply thank our principal, **Dr.T.K.Ramakrishna Rao** for providing us the efficient faculty and facilities to make our ideas into reality.

I express sincere gratitude to our pioneer **Dr. Suman Maloji,** Vice-Principal & HOD-ECE for his leadership and constant motivation provided in successful completion of our academic semester.

I express my sincere thanks to our project mentors **P SIVANI** , Associate Professor, ECE & HoD-EL&GE for their novel association of ideas, encouragement, appreciation, and intellectual zeal which motivated us to venture this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

**ABSTRACT**

This project focuses on designing and implementing a multitasking LED blinking system using Free RTOS and the STM32 microcontroller. Free RTOS is employed to manage multiple tasks, enabling each LED to blink at different intervals without blocking other system processes. The STM32 microcontroller, based on the ARM Cortex-M architecture, serves as the central controller, leveraging its efficient timers and GPIO capabilities for precise task scheduling.

The project demonstrates the key features of Free RTOS, such as task creation, priority-based scheduling, and delay handling, which are critical for real-time embedded applications. By assigning unique tasks to each LED, the system showcases the advantages of multitasking in maintaining responsiveness and modularity. Additionally, the system utilizes the FreeRTOS scheduler to ensure efficient CPU utilization and seamless execution of concurrent tasks.

The implementation provides a foundation for understanding FreeRTOS in embedded systems, emphasizing non-blocking operation and scalability. This project serves as a steppingstone for developing more complex real-time applications, such as IoT systems, robotics, or industrial automation, using STM32 and FreeRTOS.

**Literature Survey**

1.**Importance of FreeRTOS in Embedded Systems:** FreeRTOS is widely used in embedded systems for managing multitasking efficiently, enabling precise scheduling of multiple tasks without blocking delays.

2. **STM32 Microcontrollers and FreeRTOS:** STM32 microcontrollers, with their ARM Cortex-M cores, provide the necessary hardware features like timers and GPIOs, making them ideal platforms for implementing FreeRTOS-based real-time applications.

**INDEX**

|  |  |
| --- | --- |
| **S.NO** | **TITLE** |
| 1 | Introduction |
| 2 | Aim of the Project |
| 3 | Proposed Methodology  (Include Block Diagram/Flow Chart) |
| 4 | Components Explanation |
| 5 | Implementation Methodology  (Circuit and Implementation) |
| 6 | Results and Discussions  (With Screenshots) |
| 7  8 | Source Code  Conclusion and Future Scope |

**1.Introduction**

FreeRTOS is a powerful real-time operating system that enables multitasking and efficient resource management in embedded systems. In this project, we demonstrate the use of FreeRTOS to control LED blinking on an STM32 microcontroller. The project showcases FreeRTOS's ability to handle concurrent tasks, manage real-time scheduling, and demonstrate non-blocking delays.

The STM32 microcontroller, equipped with an ARM Cortex-M core, is an ideal platform for developing embedded applications that require precise timing and multitasking. In this project, multiple tasks are created using FreeRTOS to independently control LEDs at different blink intervals. This highlights the multitasking capabilities of FreeRTOS, allowing for flexible and scalable system design.

Through this project, we aim to provide a practical understanding of FreeRTOS's core features, such as task creation, scheduling, and delay management, using STM32 as the hardware platform. The LED Blinky project serves as a foundational example for beginners, paving the way for more complex embedded system applications.

**2.** **Aim of the Project**

To design and implement a multitasking LED blinking system using the STM32 microcontroller and FreeRTOS. The project demonstrates the fundamental concepts of real-time operating systems, such as task creation, priority-based task scheduling, and non-blocking delays. The primary objectives of this project include:

1. **Multitasking Demonstration:** Create independent tasks to control multiple LEDs with different blinking patterns and frequencies.
2. **Real-Time Scheduling:** Utilize the FreeRTOS kernel to manage task execution in real time.
3. **Non-Blocking Delays:** Implement vTaskDelay to allow other tasks to execute during delays, showcasing FreeRTOS's multitasking capabilities.
4. **Scalability:** Develop a modular system that can be extended with additional tasks and features.
5. **User-Friendly Debugging:** Leverage STM32CubeIDE's RTOS-aware debugging tools to monitor task states and execution.
6. User-Friendly Output: Provide a clear and easily interpretable display of distance measurements on an LCD or a serial monitor.

**3.** **Proposed Methodology**

* **Task 1: Red LED Control**
  + Toggles the Red LED at an interval of 200ms.
  + Utilizes vTaskDelay for precise timing without blocking the CPU**.**
* **Task 2: Green LED Control**
  + Toggles the Green LED at an interval of 300ms.
  + Scheduled to execute concurrently with Task 1, demonstrating multitasking.
* **Task 3: Blue LED Control**
  + Toggles the Blue LED at an interval of 700ms.
  + Runs with the same scheduling mechanism, ensuring all tasks share CPU time as per their priorities.

**STM32 Microcontroller**

* The STM32 handles GPIO configurations for the LEDs and runs the FreeRTOS kernel to manage the tasks.

**Flow Chart**

1. **System Initialization:**
   * Configure the STM32 GPIO pins for Red, Green, and Blue LEDs using STM32CubeMX.
   * Enable FreeRTOS middleware and configure the system tick for the kernel**.**
2. **Task Setup:**
   * Create three tasks in the application code:
     + Task 1: Toggles the Red LED every 200ms.
     + Task 2: Toggles the Green LED every 300ms.
     + Task 3: Toggles the Blue LED every 700ms.
3. **Scheduler Start:**
   * Start the FreeRTOS scheduler, which begins executing the tasks based on their priorities and time slices.
4. **Task Execution:**
   * Task 1: Executes first, toggling the Red LED, then sleeps for 200ms using vTaskDelay.
   * Task 2: Executes next, toggling the Green LED, then sleeps for 300ms using vTaskDelay.
   * Task 3: Executes next, toggling the Blue LED, then sleeps for 700ms using vTaskDelay.
   * The FreeRTOS kernel manages task execution and ensures no task blocks other tasks.

**4**. **Components Explanation**

**STM32 Microcontroller:**

* **Description:**
  + The STM32 microcontroller serves as the central processing unit for the project.
  + It is based on the ARM Cortex-M core, offering powerful features like GPIOs, timers, and real-time performance.
* **Role in LED Blinky:**
  + Controls the LEDs by toggling GPIO pins.
  + Uses FreeRTOS tasks to manage concurrent blinking of multiple LEDs with different intervals.

**LEDs:**

* **Description:**
  + LEDs are used to provide visual output by toggling ON and OFF states.
* **Role in LED Blinky:**
  + The GPIO pins of the STM32 are connected to the LEDs, enabling them to blink at specific intervals defined by FreeRTOS tasks.
  + Each LED can be controlled by a separate task for multitasking demonstration.

**Power Supply:**

* **Description:**
  + A regulated power source is used to provide the required voltage levels for the STM32 and the LEDs.
* **Role in LED Blinky:**
  + Ensures stable operation of the STM32 microcontroller and LEDs.
  + Commonly, the STM32 board is powered via USB, and resistors are used to limit current to the LEDs.

**FreeRTOS:**

* **Description:**
  + FreeRTOS is an open-source real-time operating system integrated into the project to handle multitasking.
* **Role in LED Blinky:**
  + Manages the execution of multiple LED blinking tasks concurrently.
  + Schedules tasks based on priority and uses the system tick for timing.

5. **Implementation Methodology**

A circuit board with wires and wires

Description automatically generated

**Connections:**

1. **STM32 Pins:**
   * A2 (GPIO): Connected to the red LED with a current-limiting resistor (100–500 Ω).
   * A3 (GPIO): Connected to the green LED with a current-limiting resistor.
   * A4 (GPIO): Connected to the blue LED with a current-limiting resistor.
   * G (Ground): Common ground for the LEDs and the microcontroller.
2. **LEDs:**
   * Each LED (red, green, blue) is connected to a separate GPIO pin, allowing independent control.
   * Resistors are used to limit the current through the LEDs and protect them from damage.
3. **Power Supply:**
   * The STM32 board is powered via the micro-USB port, and the ground (G) pin ensures a common ground connection for the circuit.

**Implementation Steps for FreeRTOS LED Blinky Using STM32**

**Step 1: Hardware Setup**

1. Connect the A2, A3, and A4 GPIO pins on the STM32 board to the anodes of the red, green, and blue LEDs, respectively.
2. Place a 100–500 Ω resistor in series with each LED to limit the current.
3. Connect the cathodes of all LEDs to the ground pin (G) of the STM32 board.

**Step 2: STM32CubeMX Configuration**

1. Open STM32CubeMX within STM32CubeIDE.
2. Configure the GPIO pins (A2, A3, A4) as output pins for controlling the LEDs.
3. Enable FreeRTOS Middleware:
   * Go to the Middleware tab and enable FreeRTOS.

**6.RESULTS AND DISCUSSIONS**

A circuit board with wires and lights

Description automatically generated

**System Performance**

* **LED Blinking Accuracy:**
  + The FreeRTOS-based LED blinky project demonstrated accurate periodic LED blinking. The LED toggled every second (or at the specified interval), controlled by a task with a delay set via vTaskDelay.
  + The blinking rate remained consistent, and the LED accurately reflected the expected time intervals, demonstrating FreeRTOS's effective task scheduling.
* **Task Scheduling:**
  + Multiple tasks were executed concurrently, with the LED task having the highest priority. The scheduler successfully managed the task execution without any observable delays.
  + The use of FreeRTOS allowed the LED blinking task to be non-blocking. While the LED blinked, other tasks could run without interfering with the LED’s operation, showing how FreeRTOS supports multitasking efficiently.
* **System Tick Timer:**
  + The STM32's system tick timer, used by FreeRTOS to manage task switching and delays, was confirmed to be working as expected. Task switching occurred precisely, ensuring no jitter in the LED’s blinking cycle.

**OLED Display for Real-Time Monitoring**

* In addition to the LED task, a simple task was added to monitor the system's task status and display it on an OLED screen (if applicable to your hardware). The OLED displayed information such as the current task execution and blinking status, ensuring that the system's real-time performance could be observed.

**Power Consumption**

* The STM32 microcontroller and the LED blinking task consumed minimal power, suitable for battery-operated setups. FreeRTOS’s task scheduling ensured that the processor wasn't kept unnecessarily busy, allowing for efficient power usage. FreeRTOS features like low-power sleep modes (using vTaskDelay) helped in reducing power consumption during idle periods.

**Discussions:**

* **Real-Time Performance:**
  + Using FreeRTOS provided an efficient and non-blocking solution for managing the LED blinking task. The LED's periodic blinking remained accurate even as other tasks were added to the system. This highlights the real-time capability of FreeRTOS, ensuring precise control over time-dependent operations.
* **Multitasking Benefits:**
  + While the basic LED blinking system can be achieved with traditional code, using FreeRTOS provided better scalability. Tasks such as monitoring sensors, handling communication, or controlling other peripherals can be added without disrupting the LED blinking functionality. FreeRTOS enables handling multiple tasks simultaneously, a key advantage in embedded system development.
* **Limitations and Improvements:**
  + Although FreeRTOS handled the LED blinking task effectively, there could be slight overhead in task switching, especially as more tasks are added to the system. In scenarios with strict timing requirements, it might be necessary to optimize the task priorities and scheduling to minimize context switch latency.
  + Future improvements could involve using FreeRTOS’s advanced features such as mutexes and semaphores for task synchronization or introducing more complex real-time operations.

A screenshot of a computer program

Description automatically generated**7.Source Code**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**8.** **Conclusion and Future Scope**

8.1**Conclusion**

The project utilizing the **STM32 microcontroller** and the **HC-SR04 ultrasonic sensor** for distance measurement successfully demonstrates how embedded systems can be effectively applied for real-time applications. The integration of hardware components, such as the ultrasonic sensor and the OLED display, with the STM32's powerful processing capabilities, enables accurate distance measurement and real-time data display. The project is both simple and cost-effective, providing precise results suitable for a variety of applications like obstacle detection, robotics, and automation systems. The system operates efficiently and reliably, offering an excellent example of the potential of embedded systems for distance-based applications.

**Future Scope**

1. **Integration with IoT:**
   * The system can be expanded to connect with IoT platforms, such as **AWS** or **Firebase**, to monitor and log distance data remotely. This can be particularly useful in **smart parking systems**, **industrial automation**, and **warehouse management**.
2. **Advanced Displays:**
   * To enhance user interaction, the LCD could be upgraded to an **OLED** or **touchscreen** for a more advanced and interactive graphical display of distance data.
3. **Multi-Sensor Setup:**
   * Using multiple ultrasonic sensors can provide **360-degree obstacle detection**, making the system more suitable for robotics, **drones**, or **automated guided vehicles** (AGVs).
4. **Machine Learning Integration:**
   * Machine learning algorithms could be introduced to analyze distance patterns, which would be beneficial for **predictive maintenance** or **anomaly detection** in industrial or autonomous systems.
5. **Improved Power Efficiency:**
   * The system can be optimized to use **lower power**, making it ideal for **battery-operated devices** or **solar-powered applications**.