College of Engineering
Department of Electrical and Computer Engineering

Final Report

Course No. & Title: CE-426 Real-Time Embedded Systems

Instructor's Name: Dr. Girma Tewolde

Due Date: 6/12/2024

Team Members:

(1) Dylan Lozon

(2) Chiny Miles

Abstract

This project is an implementation of an embedded chatbot application that runs on a microcontroller with a Real-Time Operating System (RTOS). The system accepts user input through a UART interface and responds with relevant information about Kettering University using a hashmap-based approach. The project demonstrates the effective utilization of RTOS services, including threads, message queues, semaphores, and interrupt handling. Key features include user input processing, response generation, keepalive messaging, and input echoing. The application showcases the integration of an RTOS in an embedded system, handling concurrent tasks, managing input/output operations, and implementing algorithms for chatbot functionality.

Table of Contents

Project Overview

This project is an embedded system chatbot application designed to run on a microcontroller with a Real-Time Operating System (RTOS). The main goals of the project are to enable user interaction through a UART interface, implement a simple chatbot functionality, and demonstrate the effective use of various RTOS services and objects.

Project Description

The chatbot application allows users to communicate with the system by sending messages over a UART terminal. The system echoes the user's input in real-time, allowing the user to intuitively interface with the system by seeing their input as they type it. When the user presses the enter key, the chatbot generates a response and sends it back to the user over the same UART terminal. The chatbot is designed to provide information specifically related to Kettering University, covering topics such as programs offered, admission requirements, tuition costs, and campus history.

System Architecture

The project is built upon the CMSIS-RTOS (Cortex Microcontroller Software Interface Standard - Real-Time Operating System) framework and consists of two main threads:

1. **keepaliveThread**: This thread sends a "still alive" message every 30 seconds to indicate that the system is running and responsive.
2. **handleInputThread**: This thread is responsible for processing user input characters received through the UART interface. It handles special characters like enter (to

terminate the input) and backspace (to remove characters), and stores the input in a buffer.

When the user completes their input, the `respond` function is called, which invokes the `generateResponse` function from `response_generator.c`. This function sanitizes the user input, searches for matching keys in a predefined hashmap, and returns the corresponding response value.

Key Features

1. **User Input and Output**: The system accepts user input via a UART interface and provides responses back to the user through the same channel.
2. **Chatbot Functionality**: The application implements a simple chatbot that can understand and respond to specific queries related to Kettering University using a hashmap-based approach.
3. **Multithreading and RTOS Integration**: The project utilizes two threads (`keepaliveThread` and `handleInputThread`) and demonstrates the use of RTOS services such as threads, message queues, and semaphores.
4. **Input Processing and Handling**: The `handleInputThread` processes user input characters, handles special characters like enter and backspace, and stores the input in a buffer.
5. **Response Generation**: The `generateResponse` function sanitizes the input, searches for matching keys in the hashmap, and returns the corresponding response value.

6. **Keepalive and Echo**: The `keepaliveThread` sends a "still alive" message every 30 seconds, and the `echoMessage` function echoes the received user input back to the UART for visual feedback.

RTOS Services and Objects Utilized

The project effectively demonstrates the use of various RTOS objects and services, including:

- Multithreading (`keepaliveThread` and `handleInputThread`)
- Message queues (`Q_UART`)
- Semaphores (`uart_semaphore`)
- UART interrupt handling (`USART1_IRQHandler`)
- Delay functions (`osDelay`)

Program Source Code

Please see attached FinalCode folder containing uVision project files.

Conclusion

This embedded chatbot project showcases the integration of an RTOS with an embedded system, enabling user interaction through a UART interface, implementing a simple chatbot application, and effectively utilizing various RTOS objects and services. The project demonstrates proficiency in multithreading, input/output handling, concurrency management, and algorithm implementation in an embedded environment.

Works Referenced

Rohde & Schwarz. (n.d.). Understanding UART. Retrieved from https://www.rohde-schwarz.com/us/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html

Parlez-vous Tech. (n.d.). 12C, UART and SPI. Retrieved from https://www.parlezvoustech.com/en/comparaison-protocoles-communication-i2c-spi-uart/

Konopka, Darek . “UART enabling” Accessed [6/9/2024] Konopka work

Previous students. “UART 2 enabling” Accessed [6/9/2024] https://sites.google.com/site/johnkneenmicrocontrollers/sci_rs232/fci_f107?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1