// Collected data by ashabkarwar2003@gmail.com

PROJECT NAME : Data accession by using RTOS and CAN

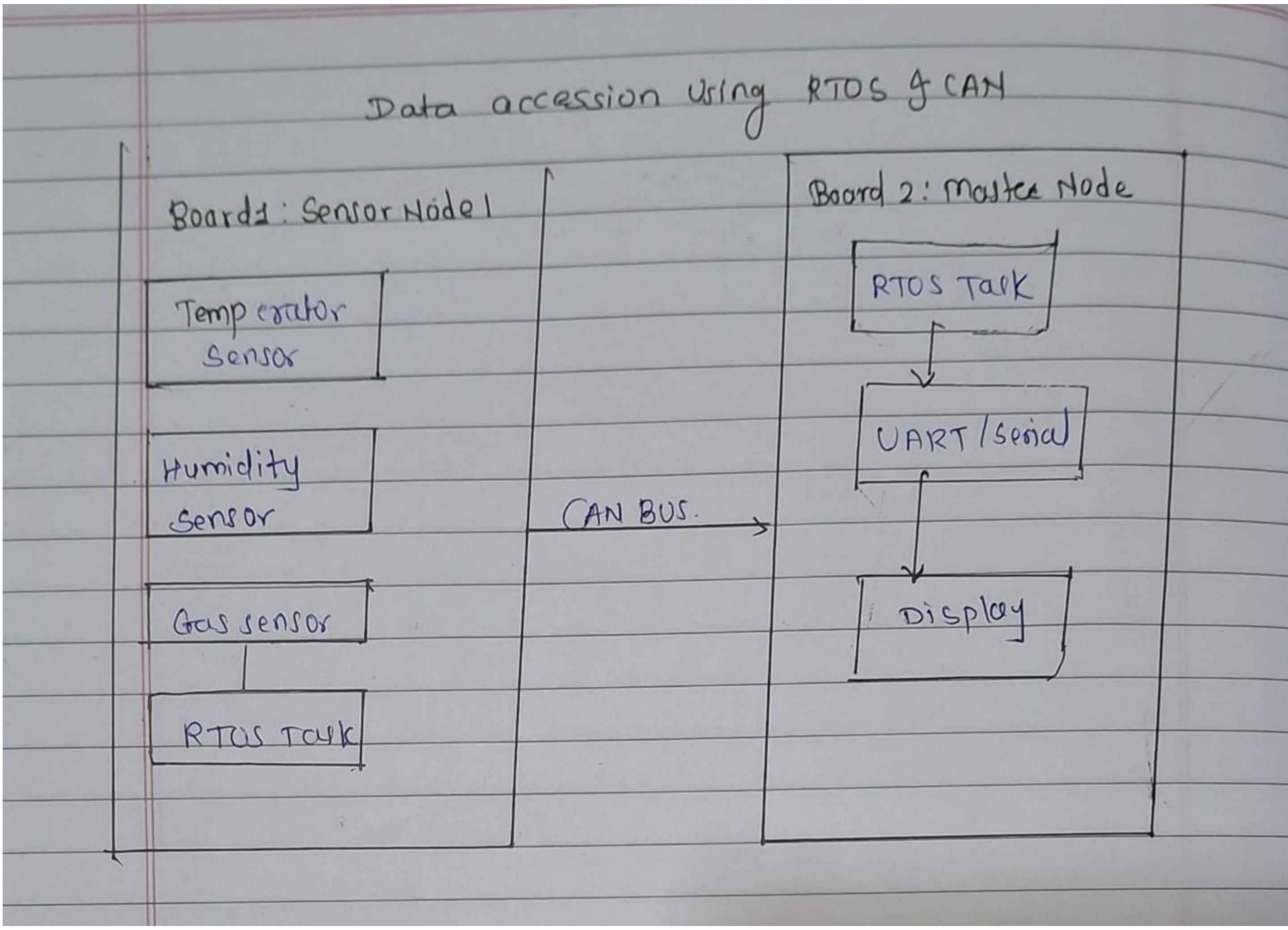
Project Overview:

To acquire data from sensors or other hardware devices and transmit it over CAN (Controller Area Network) using an RTOS (Real-Time Operating System). The system should handle real-time data processing, time-sensitive operations, and communication over the CAN bus.

Technical Stack:

- **Microcontroller:** used STM32, LPC1768, etc.
- **RTOS:** RTOS
- **Communication Protocol:** CAN (Controller Area Network)
- **Programming Language:** Embedded C
- **Development Tools:** STM32CubeIDE
- **Debugging Tools:** Oscilloscope, CAN Analyzers

Block Diagram:



Key Components of the Project

1. **RTOS (Real-Time Operating System)**:
    - **Role**: The RTOS ensures that all tasks in the system are executed within a specific time frame, which is critical for real-time applications like data acquisition.
    - **RTOS Tasks**: Tasks such as sensor data acquisition, CAN data transmission, and system monitoring will be managed by the RTOS.
    - **Common RTOS Options**: FreeRTOS, embOS, CMSIS RTOS are some of the popular RTOS options used for embedded systems.
2. **Microcontroller**:
    - **Role**: The microcontroller acts as the brain of the system. It runs the RTOS and performs the data acquisition and CAN communication tasks.
    - **Examples**: STM32, LPC1768, and other ARM Cortex-based microcontrollers are good options for such projects because they typically support both CAN and RTOS.
3. **CAN (Controller Area Network)**:
    - **Role**: CAN is used for communication between microcontrollers or devices in a network. In this project, it is used to transmit the acquired sensor data.
    - **Speed**: CAN supports high-speed data transmission (up to 1 Mbps), making it suitable for real-time applications.
    - **Message Format**: CAN messages contain an ID (identifier), data, and CRC (error-checking), which are used to ensure reliable communication.
4. **Data Acquisition**:
    - **Role**: This involves reading sensor data, such as temperature, humidity, or any other type of environmental or mechanical data.
    - **Devices**: Sensors such as temperature sensors (e.g., LM35), accelerometers (e.g., MPU6050), or pressure sensors can be used to acquire data.
5. **Task Management and Synchronization**:

- **Role**: The RTOS will handle multiple tasks (e.g., data acquisition, CAN communication, sensor reading, etc.) concurrently, ensuring that each task runs at the right time.
- **Inter-task Communication**: RTOS features like queues, semaphores, or mailboxes can be used to facilitate communication between different tasks.

Project Workflow

1. **Task Creation**:
    - You will define multiple tasks in your RTOS to handle different parts of the project. For example:
        - **Data Acquisition Task**: This task will periodically read data from sensors (e.g., temperature, pressure).
        - **CAN Communication Task**: This task will be responsible for sending the acquired data over the CAN bus.
        - **System Monitoring Task**: This could monitor the health of the system and handle any errors or exceptions.
2. **Data Acquisition**:
    - The **Data Acquisition Task** reads data from sensors. For example, if you are using a temperature sensor, this task would periodically read the temperature value from the sensor.
    - The data could be processed (e.g., converting raw sensor values to actual measurements) before sending it via CAN.
3. **CAN Communication**:
    - After data is acquired, the **CAN Communication Task** packages the data into CAN frames and sends them over the CAN bus.
    - The data packet will contain the sensor readings, along with the sensor ID, timestamp, or other necessary information.
    - CAN messages are typically transmitted using a predefined identifier (e.g., message ID), which helps in filtering and prioritizing messages on the bus.
4. **Inter-Task Communication**:

- If tasks need to share data, the RTOS provides inter-task communication mechanisms such as:
    - **Queues**: To pass data from the Data Acquisition Task to the CAN Communication Task.
    - **Semaphores**: To signal when data is ready to be sent or when the system is in a particular state.
    - **Mutexes**: To protect shared resources, ensuring that only one task has access to the resource at a time.

5. **Error Handling**:
    - CAN communication might experience errors such as bus contention, message loss, or transmission errors. The system will need to handle such errors and possibly retransmit the data or perform retries.
6. **Real-Time Performance**:
    - The RTOS ensures that each task runs at the right time, with real-time constraints, such as ensuring the data is acquired and transmitted on time.
    - For example, if the system needs to acquire data every 100ms, the RTOS will manage the timing of this task to ensure it happens at precise intervals.

Key Features:

- Implemented **RTOS** to manage **multiple real-time tasks** like data acquisition, CAN transmission, and error handling.
- Acquired data from **analog/digital sensors** and stored it temporarily in buffers.
- Transmitted sensor data over the **CAN bus** to a master controller system.
- **Task synchronization** was managed using **semaphores** and **queues** in RTOS.
- **CAN communication** was designed for **multi-node setup** to simulate real-world network traffic.
- Used **interrupt-driven** and **polling** methods for time-critical operations

System Architecture

- **Microcontroller**: The microcontroller runs the RTOS and manages the hardware peripherals (ADC for sensor input, CAN controller for CAN communication, etc.
- **RTOS Kernel**: The RTOS kernel schedules tasks based on priority and time constraints.
- **CAN Bus**: Multiple microcontrollers or devices can be connected to the CAN bus, and each can send/receive data. Each device can be identified by a unique CAN message ID.

- **Applications:** This type of project is often used in automotive systems, industrial automation, robotics, and more, where real-time data acquisition and communication are critical.
- **Industrial Automation**:
  - In industrial applications, sensor data (e.g., pressure, temperature, humidity) can be acquired from machines and transmitted to a central control system over CAN.
-
-
- **Robotics**:
  - Robots equipped with sensors for navigation or system health monitoring could acquire data and transmit it via CAN to a control system or central server for analysis.

Learning Outcomes:

- Hands-on experience in **multitasking and real-time scheduling** using an RTOS.
- In-depth understanding of **CAN protocol** architecture and implementation.
- Practical skills in **inter-task communication** .
- Developed expertise in **real-time embedded s**