Demo 2

What is XCP?

→ XCP (Universal measurement and calibration protocol) is a standardized protocol used for measurement and calibration tasks in automotive and embedded systems.
while XCP is usually used over high speed communication protocols like CAN or ethernet, its possible to implement a simplified version of XCP over UART for measurement and calibration purposes, especially in scenarios where higher bandwidth or realtime performeme is not requiered.

A general outline of how you might implement XCP over UART:

1. define XCP frames: XCP communication is based on frames that include commands and data.
Define a subset of XCP frames that are relevant to your measurement and calibration tasks. These may include commends for reading and writing memory, setting measurement parameters and triggering events

2. Message format: define a message format for transmitting XCP frames over UART.
This format should include headers to identify the type of frame or any necessary metadata.

1. frame header:

include aheader at the begging of each message to identify the type of frame and any additional information needed for processing the frame.

This header should include fields such as:

→ frame type: identifies the type of XCP frame (eg.: command frame / response frame)

→ frame length: specifies the length of frame payload

→ checksum: optional field for error checking if needed.

2. frame payload:

The payload of the message contains the actual XCP frame data.

This includes the XCP command or response data.

In code → the frame header and payload should be represented as structs.

when transmitting the XCP frame over UART, you would construct the message by populating the frame header and payload fields, then serialize the message into bytes and send it over UART

On the receiving side, you would parse the received bytes, extract the frame header and payload, and process the XCP frame accordingly.

3. Implement XCP Commands: Implement logic to handle XCP commands on both the sender (calibration tool) and receiver (ECU) sides. this involves parsing incoming frames, executing the corresponding actions (eg.: reading or writing memory), and generating appropriate responses.