Week 1 (Day 1 ~ Day 7): 기본기 확보 & SW/미래 모빌리티 트렌드 이해

Day 1

1. **SW/컴퓨터공학 기초 오리엔테이션**
    - 소프트웨어 전반적인 개념(하드웨어 vs 소프트웨어, 운영체제 개념, 프로그래밍 언어의 역사 등)
    - 임베디드 프로그래밍이 일반 애플리케이션 프로그래밍과 다른 점
2. **미래 모빌리티 트렌드**
    - 자율주행 자동차, 커넥티드 카, 전기차, 모빌리티 서비스 등 개념적으로 파악
    - 각 트렌드별 핵심 기술(센서, 통신, 인공지능, 보안 등)

Day 2

1. **C 언어 기초 1**
    - 자료형, 변수, 제어문(if, switch, for, while)
    - 기본 입출력, 배열, 문자열
2. **실습**: 간단한 C 프로그램 작성 (입력받아 처리 후 출력)

Day 3

1. **C 언어 기초 2**
    - 함수, 포인터 기초, 구조체/공용체, 열거형
    - 메모리 구조(스택/힙), 동적 메모리 할당 malloc/free
2. **실습**: 포인터와 구조체를 활용한 간단 예제

Day 4

1. **C 언어 심화 & 모던 C**
    - 포인터 심화(함수 포인터, 이중포인터)
    - C99, C11 등 표준 문법 차이
    - inline, static, volatile 등 키워드 사용 시 주의사항(임베디드 관점)
2. **실습**: volatile 키워드를 활용한 간단 메모리 맵핑 예제

Day 5

1. **C++ 기본 문법**
    - 객체지향 개념(클래스, 객체, 캡슐화, 상속, 다형성)
    - C++에서 제공하는 표준 라이브러리(STL) 소개: vector, map 등
2. **실습**: C++로 간단한 클래스 설계 & STL 컨테이너 연습

Day 6

1. **모던 C++ (C++11~17 핵심)**
    - auto, lambda, range-based for, smart pointer(unique_ptr, shared_ptr)
    - move semantics, constexpr 등
2. **실습**: 스마트 포인터를 이용한 리소스 관리 예제, 람다 함수 연습

Day 7

1. **차량 및 SW Engineering 개요**
    - 자동차 ECU 개념, 전장 아키텍처(ECU, Sensor, Actuator), CAN/LIN/Ethernet 등 통신 소개
    - 차량용 SW 개발과 일반 SW 개발의 차이(실시간성, 안전성 표준)
2. **정리 & 주말 복습**:
    - 지금까지 배운 C/C++ 문법 요약, 짧은 퀴즈/실습으로 이해도 점검

Week 2 (Day 8 ~ Day 14): 임베디드 기본 + 차량 아키텍처 심화

Day 8

1. **자동차 구조 및 전장 시스템 이해**
    - 파워트레인, 바디, ADAS, IVI(In-Vehicle Infotainment)의 구성 요소
    - ECU 간의 통신과 전원/배선 구조(E/E Architecture)
2. **실습(개념 실습)**: CAN 통신 시뮬레이션 툴 소개(무료 툴 혹은 오픈소스 기반)

Day 9

1. **임베디드 SW 개발 기초**
    - 임베디드 시스템과 일반 PC 환경 차이(리소스 제약, 실시간성, 하드웨어 직접 제어)
    - 크로스 컴파일 환경, 빌드/링커 스크립트 기초
2. **실습**: 간단한 임베디드 예제(ARM Cortex-M 시뮬레이터 사용, LED on/off)

Day 10

1. **임베디드 C 프로그래밍 기초**
    - 레지스터 접근, 인터럽트 개념, 메모리 맵(Mapped I/O)
    - volatile/const 올바른 사용, bitwise 연산
2. **실습**: 가상 개발보드(시뮬레이터)에서 버튼 인터럽트 처리 예제

Day 11

1. **소프트웨어 요구사항 분석 및 설계**
    - 차량 SW 개발 프로세스 상의 요구사항 정의 단계(ISO 26262, ASPICE 개략)
    - Use Case, 시퀀스 다이어그램, UML 개념
2. **실습**: 간단한 기능(예: "파워윈도우 제어")의 요구사항 분석 → UML로 다이어그램 작성

Day 12

1. **차량 인터넷 통신 이해**
    - 차량용 이더넷(Ethernet), TCP/UDP, SOME/IP 등 개념적 이해
    - 5G/커넥티드 카, OTA(Over-The-Air) 업데이트 개념
2. **실습**: PC에서 간단한 소켓 프로그래밍(UDP/TCP)으로 패킷 송수신 예제

Day 13

1. **자동차 SW 개발 프로세스**
    - V-Model, SPICE, ISO 26262(Functional Safety), Autosar 도입 과정
    - 개발 → 검증(Verification) → 검증 결과 피드백 구조
2. **실습**: 기존에 작성한 '파워윈도우 제어' 요구사항 기반, V-Model 상 설계/구현/테스트 개략 문서화

Day 14

1. **자동차 사이버보안 및 시큐어코딩 이해**
    - 공격 벡터(ECU 해킹, 통신 프로토콜 공격 등)
    - 시큐어 코딩 가이드라인(입력값 검증, 메모리 안전, 암호화 활용)
2. **주말 복습**:
    - 차량 구조, 임베디드 C, 요구사항분석/설계, 시큐어코딩 포인트 정리

Week 3 (Day 15 ~ Day 21): 임베디드 MCU & 실시간 운영체제, AUTOSAR 도입

Day 15

1. **임베디드 MCU 프로그래밍**
    - 8/16/32비트 MCU 차이점, 레지스터 맵, Peripheral(ADC, PWM, Timer 등)
    - 스케줄링, 타이머 인터럽트
2. **실습**: 가상 보드 또는 실제 보드에서 타이머 인터럽트를 이용한 LED 주기제어

Day 16

1. **임베디드 OS / RTOS 기초**
    - FreeRTOS 등 경량 RTOS 개념(테스크, 큐, 세마포어, 스케줄러)
    - 우선순위 기반 스케줄링, 동기화 기법
2. **실습**: FreeRTOS(또는 유사 RTOS) 환경에서 태스크 2~3개 구성하여 간단한 기능 구현

Day 17

1. **임베디드 리눅스 시스템 프로그래밍**
    - 임베디드 리눅스(yocto, buildroot 등) 개념
    - 유저 스페이스 vs 커널 스페이스, 디바이스 드라이버 개략
2. **실습**: 라즈베리파이/에뮬레이터에서 간단한 GPIO 제어(C 언어, sysfs 방식)

Day 18

1. **AUTOSAR 개요**
    - AUTOSAR Classic Platform vs Adaptive Platform
    - 레이어 구조(Complex Device Driver, ECU Abstraction, BSW, RTE, SWC)
2. **실습**(개념 위주): 간단한 AUTOSAR 구성 툴 시연영상(무료로 구할 수 있는 튜토리얼 자료 참조)

Day 19

1. **차량용 실시간 운영체제 기반 프로그래밍**
    - OSEK/VDX 표준, AUTOSAR OS 특징
    - 태스크, 자원(Resource), 카운터/알람
2. **실습**: 간단한 OSEK OS 시뮬레이터에서 태스크 생성, 우선순위 설정

Day 20

1. **소프트웨어 테스팅**
    - 단위테스트(구현 단위별), 통합테스트, 시스템테스트, HIL(Hardware-In-the-Loop) 개념
    - 코드 커버리지, 정적 분석, MISRA-C 규칙(자동차 분야)
2. **실습**: 단위테스트 툴(예: Google Test) 활용, 짧은 C 함수 단위테스트 작성

Day 21

1. **Week 3 정리**
    - 임베디드 MCU, RTOS, AUTOSAR 핵심 개념 요약
    - 테스팅 + 시큐어 코딩 다시 점검
2. **테스트 & 퀴즈**:
    - "이런 면접 질문이 있다면 어떻게 답변?" 형태로 스스로 요약

Week 4 (Day 22 ~ Day 28): 통신 프로토콜 심화 & 응용 SW 개발, OTA

Day 22

1. **차량용 통신시스템 심화**
    - CAN 통신 상세(프레임 구조, Arbitration, 데이터 길이 등), LIN, FlexRay, MOST 소개
    - CAN FD(확장), UDS 진단 프로토콜
2. **실습**: CAN 시뮬레이터에서 프레임 송수신 & 필터링

Day 23

1. **임베디드 응용 SW 개발**
    - 계층적 구조로 모듈화, HAL(Hardware Abstraction Layer), Middleware 구성
    - 이벤트 기반 아키텍처 vs 폴링 기반 아키텍처
2. **실습**: 이전에 만든 RTOS 기반 예제 확장 → 센서 데이터 수집/송신 모듈 분리

Day 24

1. **OTA(Over-The-Air) 이해**
    - Bootloader, 펌웨어 업데이트 절차
    - 보안(암호화, 무결성 검증), 롤백 전략
2. **실습(개념)**: OTA 시퀀스 다이어그램 작성, 가상 환경에서 CRC 검증 예시

Day 25

1. **임베디드 리눅스 시스템 프로그래밍 심화**
    - 커널 드라이버 기초, Device Tree, udev 등
    - 멀티프로세스/멀티스레드, IPC(파이프, 소켓, 메시지큐)
2. **실습**: 라즈베리 파이 등에서 쓰레드 2개, IPC 메커니즘 연습(C 언어)

Day 26

1. **차량용 SW 개발 프로세스 심화(AUTOSAR, ISO 26262)**
    - Safety Concept, ASIL 등급, Autosar BSW 컴포넌트 이해
    - ECU Configuration, RTE 생성 과정 개괄
2. **실습**: 간단한 Autosar 툴(데모버전)로 SWC, RTE 매핑 시연

Day 27

1. **소프트웨어 요구사항 관리 & 형상관리**
    - 요구사항 추적성, 버전 관리(Git), CI/CD 개념(자동 빌드/테스트)
    - 실제 프로젝트에서의 이슈 트래킹(Jira, Redmine 등)
2. **실습**: Git flow로 간단한 팀프로젝트 시뮬레이션(Branch → Merge → Conflict 해결)

Day 28

1. **Week 4 정리**
    - 차량통신 심화(CAN/LIN/UDS), OTA, 임베디드 리눅스, Autosar, 형상관리
    - 인터뷰에서 자주 나오는 "CAN 통신에서 우선순위는 어떻게 정해지는가?" 등 Q&A
2. **주말 복습**:
    - 실습 코드 정리 및 간단 문서화

Week 5 (Day 29 ~ Day 35): 프로젝트 형식 종합 실습 & 사이버보안 실무

Day 29

1. **프로젝트 기획: 차량용 임베디드 SW 개발 프로젝트**
    - 예) "센서값 기반 경고등/알람 송출 + CAN 통신으로 중앙 ECU에 데이터 전송"
    - 요구사항 정의(안전성, 실시간성, 통신 규격)
2. **역할 분담(개인 학습 상황 가정)**
    - MCU 프로그래밍, RTOS 태스크, CAN 송수신, 시큐어 코딩 등

Day 30

1. **시스템 설계 & 요구사항 분석**
    - 하드웨어 블록다이어그램, SW 아키텍처(태스크, 큐, 이벤트 등)
    - 보안 요구사항(통신 암호화/인증 필요 여부 등)
2. **실습**: 설계 문서 초안 & UML 다이어그램 작성

Day 31

1. **본격 구현(1): MCU 초기화 & RTOS 태스크 구성**
    - 클럭 설정, GPIO/ADC 설정, 인터럽트 구성
    - 태스크 생성 & 우선순위 결정
2. **실습**: 실제(또는 시뮬레이터) 프로젝트 작업 시작

Day 32

1. **본격 구현(2): CAN 통신 & 데이터 처리**
    - CAN 필터/마스킹 설정, 송신/수신 콜백
    - 수집 데이터 가공 로직(조건에 따라 경고등 ON/OFF)
2. **실습**: CAN 트래픽 모니터링 & 디버깅

Day 33

1. **본격 구현(3): 시큐어 코딩 적용 & 예외처리**
    - 입력값 검증, 임계영역 보호(세마포어 등), Watchdog Timer
    - 에러 처리 전략(에러코드, 시스템 리셋 등)
2. **실습**: Watchdog Timer 설정, 시큐어 코딩 가이드(예: MISRA) 준수 체크

Day 34

1. **테스트 & 디버깅**
    - 단위테스트, 통합테스트 진행
    - 디버거, 로거(UART/시리얼 출력) 통해 런타임 상태 확인
2. **실습**: FreeRTOS 통계 기능(task get stack high water mark 등)으로 자원 점검

Day 35

1. **HIL(Hardware In the Loop) or SIL(Software In the Loop) 개념 소개**
    - 실제 ECU 없이 시뮬레이터로 검증하기
    - 상위 수준의 시스템 테스트 시나리오 작성
2. **프로젝트 중간 점검**
    - 구현 완료도, 버그 목록, 수정 계획 정리

Week 6 (Day 36 ~ Day 40): 종합 마무리 & 면접 대비

Day 36

1. **프로젝트 마무리(1): 최종 기능 구현 & 문서화**
    - OTA나 추가 보안 기능(암호화) 간단 시뮬레이션 가능하면 적용
    - 최종 요구사항 대비 체크리스트 작성
2. **실습**: 코드 리팩토링, 주석 & 문서 보강

Day 37

1. **프로젝트 마무리(2): 발표자료 준비 & 시연**
    - 면접관에게 시연하듯 설계 의도, 문제 해결 과정, 테스트 결과 정리
    - 도출된 이슈(기술적 난관)와 해결책(또는 개선안)
2. **실습**: 프로젝트 시연 리허설(프레젠테이션 혹은 동영상 기록)

Day 38

1. **면접 대비: 기술 질문 정리**
    - C/C++ 문법(포인터, 메모리, 함수 포인터)
    - 임베디드 MCU/RTOS(인터럽트, 스케줄링, 동기화)
    - 차량 통신(CAN, LIN, Autosar) & 차량 SW 프로세스(ISO 26262, ASPICE)
    - 사이버보안, 시큐어코딩, OTA, ECU 구조
2. **Q&A 시뮬레이션**: 예상 질문 리스트 뽑고 직접 답안 작성

Day 39

1. **면접 대비: 문제풀이 & 실습 기반 질문 대응**
    - "포인터를 이용해 함수를 호출하려면?" 같은 코딩 테스트 질문
    - "CAN에서 Arbitration이란?", "Autosar 계층 구조?" 등 핵심 개념 구술
2. **테스트**: 구글링 없이 직접 화이트보드/에디터에 개념/코드 작성

Day 40

1. **최종 점검 & 피드백**
    - 전체 로드맵 복습, 중요 암기/개념 재정리(핵심 키워드, 면접 단골 질문)
    - 프로젝트 시연/발표 최종 리허설
2. **학습 이후 계획**
    - 부족한 파트나 더 깊이 파고들어야 할 파트(예: 커널 드라이버, AUTOSAR 툴)
    - 구체적으로 어떤 기업/직무 준비 시, 추가 자료/실습 방향