

# 嵌入式系统-第一次作业

## (1) Git 仓库的创建

1.1 在 Github 上建立新的 Repository,同时更改 Readme.md 文件的内容,加上 MIT 开源协议的内容与作业的需求。


### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

*Required fields are marked with an asterisk (\*).*


Owner \*

Repository name \*

 wWXTw

/

HW-Embedded System



Your new repository will be created as HW-Embedded-System.


The repository name can only contain ASCII letters, digits, and the characters `.`, `-`, and `_`.

Great repository names are short and memorable. Need inspiration? How about `glowing-octo-doodle` ?

Description (optional)


中南大学-嵌入式系统的相关作业提交仓库

☒

 Public

Anyone on the internet can see this repository. You choose who can commit.

☐

 Private

You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

HW-Embedded-System / README.md

in main

Cancel changes

Commit changes...

Edit

Preview

Code 55% faster with GitHub Copilot

Spaces 2 Soft wrap

1

2

3

4

5

6

7

8

9

10

11

12

13

# HW-Embedded-System

中南大学-嵌入式系统的相关作业提交仓库

**\*\*开源版权说明\*\***

本项目基于 **\*\*MIT License\*\*** 开源，任何人可以自由使用、修改和分发。

查看 `[LICENSE](LICENSE)` 文件获得具体信息。

**\*\*第一次作业任务需求\*\***

背景1: GitHub 是全球最大的开源代码托管平台之一，提供了强大的工具来支持开源项目的开发、协作和管理。GitHub 也是一个基于 Git 的代码托管平台，而Git工具主要用于软件开发和版本控制，是大型项目多人协作的重要工具。

作业1: 请在github或gitee上创建一个开源仓库，并利用Git工具进行不少于2次commit提交，在README.md文件中标注嵌入式系统作业需求、以及标注开源版权说明。

背景2: Deepseek 是一个专注于人工智能（AI）和自然语言处理（NLP）领域的模型或技术，主要应用于大语言模型（LLM）和 AI 生成任务。能够极大提升产品开发、文档撰写等工作。

作业2: 请利用大模型（如DeepSeek或Chatgpt）工具，协助自己完成一套嵌入式系统的软硬件设计说明书。

注意: 请将软硬件设计说明书写到word文档中，要包含详细的嵌入式芯片、传感器选型、实时操作系统、系统构架方案等，且最后加上自己对设计说明书的点评。

1.2 利用 Git 工具与创建仓库的 HTTP 链接在本地克隆一个 Git 仓库(利用 git clone 命令)

```
MINGW64:/d/Code/Embedded/HW






wwXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded
$ mkdir HW

wwXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded
$ cd HW

wwXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded/HW
$ git clone https://github.com/wwXTw/HW-Embedded-System.git
Cloning into 'HW-Embedded-System'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), 4.05 KiB | 2.03 MiB/s, done.
Resolving deltas: 100% (1/1), done.

wwXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded/HW
$ ls -alt
total 0
drwxr-xr-x 1 wwXTw 197121 0 Mar  9 17:19 HW-Embedded-System/
drwxr-xr-x 1 wwXTw 197121 0 Mar  9 17:19 ./
drwxr-xr-x 1 wwXTw 197121 0 Mar  9 17:18 ../
```

1.3 在本地仓库中添加两个 word 文档文件，依照 add-commit-push 的流程将这两个文档分两次提交到远程仓库。

名称	修改日期	类型	大小
 .git	2025/3/9 17:19	文件夹	
 HW 1-1.docx	2025/3/9 17:22	DOCX 文档	0 KB
 HW 1-2.docx	2025/3/9 17:23	DOCX 文档	0 KB
 LICENSE	2025/3/9 17:19	文件	2 KB
 README.md	2025/3/9 17:19	Markdown File	2 KB

```
wwXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded/HW/HW-Embedded-System (main)
$ git add 'HW 1-1.docx'

wwXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded/HW/HW-Embedded-System (main)
$ git commit -m "第一次commit"
[main ef3c762] 第一次commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 HW 1-1.docx

wwXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded/HW/HW-Embedded-System (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 324.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/wwXTw/HW-Embedded-System.git
e5339c5..ef3c762  main -> main
```


```
WWXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded/HW/HW-Embedded-System (main)
$ git add 'HW 1-2.docx'

WWXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded/HW/HW-Embedded-System (main)
$ git commit -m "第二次提交"
[main 438c691] 第二次提交
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 HW 1-2.docx

WWXTw@LAPTOP-NULS719B MINGW64 /d/Code/Embedded/HW/HW-Embedded-System (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 250 bytes | 250.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/WWXTw/HW-Embedded-System.git
ef3c762..438c691 main -> main
```





1.4 在 Github 上查看对应的提交记录,文档文件也顺利同步。

#### Commits

 main

All usersAll time

Commits on Mar 9, 2025

<b>第二次提交</b> WWXTw committed now	438c691  
<b>第一次commit</b> WWXTw committed 1 minute ago	ef3c762  

(2) 利用 AI 大模型生成嵌入式系统的软硬件说明书

## 发动机 ECU 嵌入式系统软硬件说明书

### 1. 概述

发动机电子控制单元（ECU，Engine Control Unit）是现代汽车发动机管理的核心嵌入式系统。其主要功能是实时监测发动机运行状态，计算最优控制参数，并执行燃油喷射、点火时机、排放控制等操作，以提高发动机性能、降低油耗和减少排放。本说明书详细介绍 ECU 的软硬件方案。

### 2. 硬件设计

#### 2.1 嵌入式处理器选型

ECU 需要具备强大的实时计算能力，通常采用以下类型的处理器：

处理器型号	制造商	架构	主频	主要特点
Infineon TC397	Infineon	TriCore	300 MHz	多核架构，高实时性，支持 ASIL-D
Renesas RH850/F1K	Renesas	RH850	240 MHz	低功耗，适用于车载环境
NXP MPC5777C	NXP	PowerPC	264 MHz	3 核架构，适用于发动机控制
ST SPC58	STMicroelectronics	PowerPC	200 MHz	车规级 MCU，支持 AUTOSAR

#### 2.2 传感器选型

ECU 依赖多个传感器采集数据，常用传感器如下：

传感器	型号	作用
氧传感器（O <sub>2</sub> Sensor）	Bosch LSU 4.9	监测废气氧含量，优化燃烧
进气压力传感器（MAP）	Bosch TMAP	计算发动机负载
曲轴位置传感器（CKP）	Honeywell 1GT101DC	确保喷油/点火同步
凸轮轴位置传感器（CMP）	Melexis MLX90215	监测气门相位
节气门位置传感器（TPS）	Bosch TPS 5V	计算油门开度

温度传感器（ECT）                      NTC 10K                      检测发动机冷却液温度

2.3 存储单元

ECU 需要存储数据用于计算和存储标定参数，常见存储方案如下：

**Flash ROM（固件存储）：** 2MB-4MB NOR Flash，用于存储程序代码（例如 Spansion S29GL256S）

**EEPROM（参数存储）：** 64KB-256KB EEPROM（例如 Atmel AT24C256）

**RAM（运行数据）：** 256KB-1MB SRAM（例如 Cypress CY7C1049G）

2.4 通信接口

ECU 需要与其他车载控制单元和传感器进行通信，主要采用以下接口：

通信协议	说明
CAN（Controller Area Network）	主要用于车载 ECU 间通信，符合 ISO 11898 标准
LIN（Local Interconnect Network）	用于低速传感器和执行器通信
SPI/I <sup>2</sup> C	连接 EEPROM、传感器等外设
UART	主要用于调试和诊断

3. 软件架构

3.1 实时操作系统（RTOS）

ECU 需要高实时性，常用 RTOS 如下：

RTOS	主要特点
AUTOSAR OS	车规标准，高度模块化，支持多任务管理
FreeRTOS	轻量级，可移植性强
QNX Neutrino	高可靠性，支持 POSIX
VxWorks	低延迟，广泛应用于汽车和工业控制

3.2 软件架构层次

软件采用分层架构，包括以下层次：

- 硬件抽象层（HAL）**：屏蔽不同硬件平台的差异，提供标准化接口。
- 设备驱动层（Drivers）**：实现对传感器、执行器等外设的驱动。
- 实时操作系统（RTOS Kernel）**：提供任务调度、中断管理等功能。
- 应用层（Application）**：包含发动机管理逻辑，如燃油喷射、点火控制、排放控制等。



## 4. 关键功能模块

### 4.1 燃油喷射控制

- 读取进气压力、发动机转速等数据，计算最佳喷油量
- 控制喷油器开启时长，提高燃油效率

### 4.2 点火控制

- 计算点火时机，确保最佳动力输出
- 根据发动机负载调整点火提前角

### 4.3 电子油门控制

- 读取节气门位置传感器数据，计算目标油门开度
- 通过 PWM 控制节气门电机，实现平稳加速

## 4.4 故障诊断 (OBD-II)

1. 监测发动机传感器数据，检测异常情况
2. 通过 CAN 总线向车辆诊断系统报告故障码 (DTC)

## 5. 功能安全 & 可靠性

ECU 需符合 ISO 26262(功能安全) 标准，确保系统安全可靠。

安全机制	说明
冗余传感器	关键传感器（如曲轴位置传感器）采用双冗余设计
看门狗定时器	防止程序死循环，提高系统稳定性
故障检测与恢复	发生故障时进入降级模式，确保发动机最低可用功能

## 6. 评价

本次使用 ChatGPT 生成的汽车发动机控制单元 ECU 的软硬件说明书，总体上来说效果良好，大语言模型生成的软硬件参数与架构设计专业度较高，同时也比较符合现代业界规范与前沿技术，对我学习并了解嵌入式系统与 ECU 相关知识具有很大的帮助。