Software Engineering and Applications 软件工程与应用, 2025, 14(2), 413-420
Published Online April 2025 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2025.142037

Hans汉斯

探讨不同主控芯片在自动行驶小车中的应用差异

梅红樱，刘　真，常世鑫，成　博，韩名扬，杜仕博，张腾飞，丁　翔，文　桦，温　华*

黄淮学院电子信息学院，河南　驻马店

收稿日期：2025年2月17日；录用日期：2025年4月17日；发布日期：2025年4月27日

摘　要

在当今智能化时代，嵌入式系统开发已广泛渗透日常生活的方方面面，是电子信息类学生的必修课。其中，51单片机和STM32单片机最具代表性，网络资源丰富，常被纳入学校的课程体系。然而，在参与电子设计大赛等学科竞赛时，学生常会遇到多种复杂题型，必有一道是以TI主控为芯片的硬件设计。基于此，为帮助大学生快速掌握TI系列开发板的应用，本文以自动行驶小车为例，通过理论分析、程序对比和实验验证三部分深入剖析TI芯片与常规芯片在实际应用中的差异。

关键词

TI MSPM0系列，STM32F1系列，自动行驶小车，声光提示

Discussion of the Discrepancies for the Application of Different Main Control Chips in Automatic Driving Cars

Hongying Mei, Zhen Liu, Shixin Chang, Bo Cheng, Mingyang Han, Shibo Du, Tengfei Zhang, Xiang Ding, Hua Wen, Hua Wen*

School of Electronic Information, Huanghuai University, Zhumadian Henan

Received: Feb. 17th, 2025; accepted: Apr. 17th, 2025; published: Apr. 27th, 2025

Abstract

In today's intelligent era, embedded system development has been widely penetrated into all

*通讯作者。

文章引用: 梅红樱, 刘真, 常世鑫, 成博, 韩名扬, 杜仕博, 张腾飞, 丁翔, 文桦, 温华. 探讨不同主控芯片在自动行驶小车中的应用差异[J]. 软件工程与应用, 2025, 14(2): 413-420. DOI: 10.12677/sea.2025.142037

aspects of daily life, and is a required course for electronic information students. Among them, 51 microcontroller and STM32 microcontroller are the most representative, rich in network resources, and are often included in the school curriculum system. However, when participating in discipline competitions such as electronic design competitions, students often encounter a variety of complex questions, one of which must be the hardware design of TI master chip. Based on this, in order to help college students quickly grasp the application of TI series development board, this paper takes the automatic driving car as an example and deeply analyzes the differences between TI chips and conventional chips in practical application through theoretical analysis, program comparison and experimental verification.

Keywords

TI MSPM0 Series, STM32F1 Series, Automatic Driving Car, Acousto-Optic Cue

1. 引言

嵌入式主控芯片，也称为微控制器(MCU)，在智能家居中可控制各种家电设备，如照明灯、门锁、风扇等，在安防设备中可控制摄像头的拍摄和传输等功能，在医疗领域可实现对心电图、手术室设备的精确控制和监测，在工业领域可实现对各种传感器的自动化监测，因此应用范围广泛而且多样化，在众多领域都发挥着至关重要的作用[1]。可以说从物联网到工业自动化，再到消费电子、医疗设备等，微控制器因其低成本、低功耗以及强大的集成能力而被广泛应用。51 单片机和 STM32 单片机最具代表性[2]，在大部分学校都会开设相关课程，而且网上资源也是应有尽有，学习起来也是很容易上手，相应的项目很容易完成。但是这两种主控芯片也有自己的缺点，比如 51 单片机在处理能力和外设集成度方面相对较弱，难以应对复杂的控制任务；STM32 虽然性能较强，但其开发环境配置复杂，库函数使用门槛较高，初学者往往需要花费大量时间熟悉其生态系统。此外，这两种芯片在低功耗设计和模拟信号处理精度方面也存在一定局限性，难以满足某些高端应用场景的需求。相比较，TI 的微控制器在性能方面，具有强大的处理能力和高精度，适合处理复杂的信号和执行控制任务[3]。其次在功能方面，集成多种外设，如模拟数字转换器、数字模拟转换器、定时器和通信外设，可简化设计，减少外部组件的需求，从而降低系统的复杂性和成本。但是，在普通二本院校电子信息类的人才培养方案中，不需要掌握该主控芯片的使用，因此大部分学生练习较少，对于初学者在学习使用这款单片机的时候总会感觉到无从下手，仅靠已有的 51 和 STM32 主控的知识，在参加电子设计大赛等学科竞赛时只能选择信号类、电源类、模拟类等，或者其他主控的控制类。因此，为了快速地让学生上手 TI 板主控，了解如何配置引脚、图形化配置引脚后如何调用这些配置好的引脚，官方库中的函数如何使用等等一系列问题，下面以自动驾驶小车为例，与 ST32F1 系列单片机进行对比分析，加深对 TI MSPM0 系列 MCU 的理解。

2. 目标任务与方案

如图 1 所示，现有一场地面积为 220 cm × 120 cm，该场地上有两个对称半圆弧线，其半径为 40 cm，弧线为黑色，线宽 1.8 cm 左右，弧线的四个顶点分别定义为 A、B、C 和 D 点，其中 AB 之间的长度为 100 cm。场地上的其他部位没有任何的标记。请以 51 单片机和 TI MSPM0 系列 MCU 控制设计的自动行

驶小车，能在目标路径上行驶，完成相关的任务，比如：(1) 将小车放在位置 A 点，小车能自动行驶到 B 点停车，停车时有声光提示。用时不大于 15 秒。(2) 将小车放在位置 A 点，小车能自动行驶到 B 点后，沿半弧线行驶到 C 点，再由 C 点自动行驶到 D 点，最后沿半弧线行驶到 A 点停车，每经过一个点，声光提示一次。完成一圈用时不大于 30 秒。因此，通过这两个任务设定，可以很好地展示两个芯片的相同和不同之处。

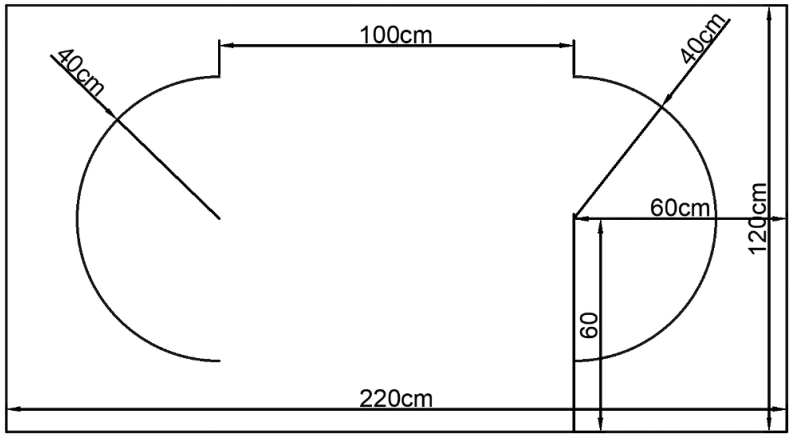

Figure 1. Mission map
图 1. 任务地图

因此，该款自动行驶小车主要有输入部分、主控和输出三大部分组成，为完成上述两个功能，主控系统主要以 TI MSPM0 G3507 和 STM32F103C8T6 两种单片机为主控芯片；输入部分有四路红外传感器[4]、MPU6050 姿态传感器[5]、按键模块，根据有没有感应到黑线等外界环境或者小车的航向状态控制小车的方向；输出部分有电机驱动模块、声光提示模块，当小车在红外和姿态传感器的作用下，小车会根据主控信息驱动电机的速度，当小车到达某位置时，能够进行声光提示。系统的整体设计框图如图 2 所示。

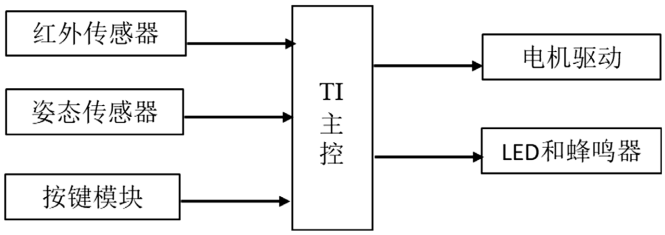

Figure 2. System design block diagram
图 2. 系统设计框图

3. 系统软件设计

该款自动行驶小车需要完成的两个功能互不干扰，因此，在这里采用两个按键选择需要完成的功能，主程序如图 3 所示，当程序复位后，按下不同按键，程序根据按键当前标志位的值执行当前任务。

若按下的是按键 1，进行任务选择后小车开始执行，此时小车从 A 点前进，小车将会根据前端的四路红外传感器信号来判别是否到达 B 点，当四路红外传感器至少有一路检测到黑线时，小车便到达 B 点，声光提示，任务结束；若任何一路都没检测黑线时，表示小车没有到达 B 点，继续执行任务一。

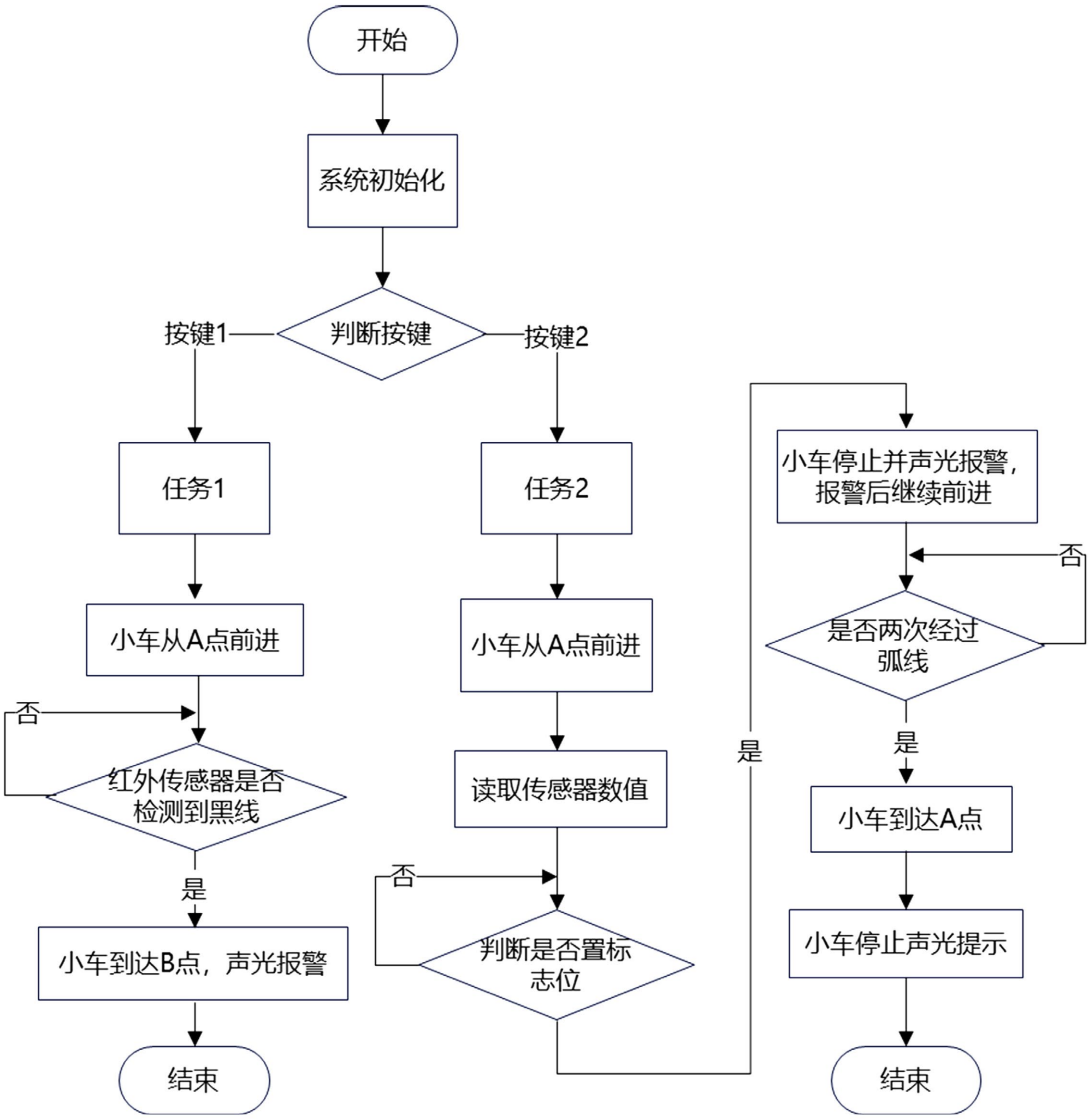

Figure 3. Program flow chart
图 3. 程序流程图

若按下的是按键 2，进行任务选择后小车开始执行，此时小车从 A 点前进，执行任务 2 时，MPU6050 姿态传感器起作用。在小车从 A 点到 B 点的过程中，四路红外传感器未检测到黑线，小车一直直行，此时 MPU6050 姿态传感器的航向角(yaw)值为 0；当小车的四路红外传感器的任何一路检测到黑线时，表明小车到达 B 点，主控芯片第一次进行声光提醒，随后便进入 BC 弧线段曲线行驶，在此段四路红外传感器都可以检测到黑线，主控芯片会根据算法让小车沿着黑线行驶，且姿态传感器的航向角先由 0 值变为 90 度，再由 90 度变为 0 度，表明小车第一次完整通过弧线，小车到达 C 点，主控芯片第二次进行声光提醒；随后小车再继续直行行驶，在这过程中，姿态传感器的航向角一直保持为 0，四路红外传感器也未检测到黑线，当四路传感器检测到黑线或者航向角由 0 开始增加时，表明小车到达 D 点，主控芯片第三次进行声光提醒，随后小车便在四路红外传感器和算法的作用下沿着 DA 弧线段进行曲线行驶，姿态传感器的航向角也是由 0 开始增加到 90 度，然后再减小到 0 度，直到四路红外传感器未检测到黑线、姿态传感器的航向角变为 0 时，表明小车回到 A 点，主控芯片第四次进行声光提醒，小车停止行驶。

4. 程序代码

4.1. 主程序

在功能实现层面，无论是基于 STM32 还是 TI MSPM0 G3507 的微控制器，程序都是采用 C 语言进行编写，如图 4 所示，且主程序的逻辑结构与编写方式两者几乎无异，两者都倡导模块化编程，通过调用开发者定义的函数即可完成主程序的构建。然而，它们在开发流程和工程文件管理上存在一些显著的差异。一方面，STM32 的开发环境通过开发者新建空白的工程导入库函数即可进行编程，不需要对硬件进行一些初始化，而 TI MSPM0 开发者需要使用官方提供的空白工程模板进行编写，在官方的空白模板中对硬件进行一些基本的配置即可。另一方面，在工程文件保存管理上，STM32 的开发环境中允许开发者可以直接保存在自定义的路径下，而在使用 Code Composer Studio Theia(CCS)进行 TI MSPM0 开发时，导入空白工程模板，软件会自动在电脑的 c 盘中开辟一个名为“ti”的文件夹进行存放开发者的工程，开发者需要在该路径下才能找到并管理自己的工程。

```c
#include "stm32f10x.h"
#include "Delay.h"
#include "Key.h"
#include "Motor.h"
#include "car.h"
#include "MPU6050.h"

uint8_t key=0;

int main(void)
{
  Motor_Init();
  Key_Init();
  MPU6050_Init();

  do{
    key=Key_GetNum ();
  }while(!key);

  Motor_SetSpeed_1(70);
  Motor_SetSpeed_2(70);

  while(1)
  {
   switch(key)
   {
     case 1:car_move_1();break;
     case 2:car_move_2();break;
   }
  }
}
```

```c
#include "KEY.h"
#include "Motor.h"
#include "Move.h"
#include "board.h"
#include "bsp_mpu6050.h"
#include "inv_mpu.h"
#include "stdio.h"
#include "ti/driverlib/dl_gpio.h"
#include "ti_msp_dl_config.h"


int main(void) {
  SYSCFG_DL_init();

  board_init();
  MPU6050_Init();
  DL_TimerG_startCounter(PWM_Motor_INST);
  int key = 0;
  Motor_Setspeed(0, 70);
  Motor_Setspeed(1, 70);
  Motor_ON();

  do {
    key = KEY_GetNUM();
  } while (!key);
  while (1) {

    switch (key) {
    case 1:
      Car_move_1();
      break;
    case 2:
      Car_move_2();
      break;
    }
  }
}
```

Figure 4. STM32 main program (left) and TI MSPM0 G3507 main program (right)
图 4. STM32 主程序(左)和 TI MSPM0 G3507 主程序(右)

4.2. 引脚配置

在 STM32 中需要使用库函数对引脚进行配置，配置的过程较为繁琐并且固定，最重要的是需要首先打开引脚所在的总线的时钟。以自动行驶小车中产生 PWM 波控制电机的转动为例，基本流程为打开定时器时钟、配置时基单元、配置输出比较单元、配置 GPIO 为复用推挽输出、最后运行控制，配置程序如图 5 所示。

```c
#include "stm32f10x.h"            // Device header

void PWM_Init(void)
{
  RCC_APB1PeriphClockCmd (RCC_APB1Periph_TIM2 ,ENABLE);
  RCC_APB2PeriphClockCmd (RCC_APB2Periph_GPIOA ,ENABLE);

  TIM_InternalClockConfig(TIM2);

  GPIO_InitTypeDef GPIO_InitStructure;
  GPIO_InitStructure .GPIO_Mode =GPIO_Mode_AF_PP;
  GPIO_InitStructure.GPIO_Pin =GPIO_Pin_0| GPIO_Pin_1;
  GPIO_InitStructure.GPIO_Speed =GPIO_Speed_50MHz ;
  GPIO_Init (GPIOA ,&GPIO_InitStructure);

  TIM_TimeBaseInitTypeDef Tim_InitStructure;
  Tim_InitStructure.TIM_ClockDivision = TIM_CKD_DIV1;
  Tim_InitStructure.TIM_CounterMode=TIM_CounterMode_Up;
  Tim_InitStructure.TIM_Period =100-1;   //ARR
  Tim_InitStructure.TIM_Prescaler =36-1;  //PSC
  Tim_InitStructure.TIM_RepetitionCounter =0;
  TIM_TimeBaseInit(TIM2 ,&Tim_InitStructure );

  TIM_OCInitTypeDef TIM_OCInitStructure;
  TIM_OCStructInit(&TIM_OCInitStructure);
  TIM_OCInitStructure.TIM_OCMode =TIM_OCMode_PWM1;
  TIM_OCInitStructure.TIM_OCPolarity =TIM_OCPolarity_High;
  TIM_OCInitStructure.TIM_OutputState =TIM_OutputState_Enable;
  TIM_OCInitStructure.TIM_Pulse =0;   //CCR
  TIM_OC1Init (TIM2,&TIM_OCInitStructure );
  TIM_OC2Init (TIM2,&TIM_OCInitStructure );

  TIM_Cmd (TIM2,ENABLE);
}

void PWM_SetCompar1(uint16_t compare)
{
  TIM_SetCompare1(TIM2,compare);
}
void PWM_SetCompar2(uint16_t compare)
{
  TIM_SetCompare2(TIM2,compare);
}
```

Figure 5. STM32 Timer configuration
图 5. STM32 定时器配置

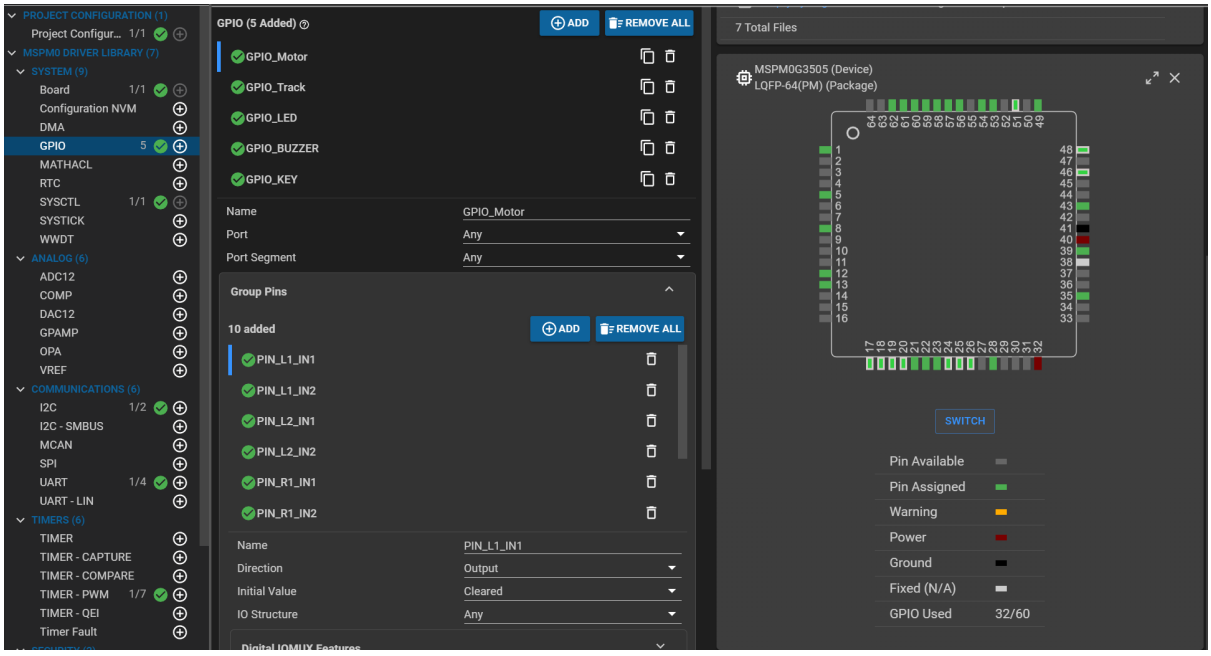

Figure 6. Sysconfig graphical user interface
图 6. Sysconfig 图形用户界面

TI MSPM0 G3507 支持使用 TI 官方提供的 Code Composer Studio Theia (CCS)进行程序调试，其内置的 SysConfig 工具通过图形用户界面(GUI)实现了外设、时钟、模拟模块、引脚多路复用器及安全性等关键程序组件的可视化配置[6]。该工具可根据用户输入的 GUI 配置自动生成带有完整注释的驱动程序和配置文件，并支持实时同步更新，即开发人员在修改 GUI 配置时，相关驱动程序和配置文件会自动同步调整，这不仅提高了代码的可读性和可维护性，还显著降低了开发难度。此外，SysConfig 提供两种视图模式，直观展示引脚与外设的配置关系，帮助开发者更好地理解硬件与软件的交互机制，其图形用户界面如图 6 所示。

因此，在 STM32 开发中，通常采用标准库进行配置，但这要求开发者必须深入理解标准库的使用方法。特别是在配置引脚时，开发者需要首先启用相应引脚的时钟总线，这就要求对单片机内部架构有较深入的理解。相比之下，TI MSPM0 系列 MCU 采用图形化的 Sysconfig 配置工具，其引脚配置过程更加直观便捷，大大降低了初学者的学习曲线，显著提升了开发效率。

4.3. 库函数调用

```
/* Defines for PWM_Motor */
#define PWM_Motor_INST                                                  TIMA0
#define PWM_Motor_INST_IRQHandler                                TIMA0_IRQHandler
#define PWM_Motor_INST_INT_IRQN                                  (TIMA0_INT_IRQn)
#define PWM_Motor_INST_CLK_FREQ                                         32000000
/* GPIO defines for channel 0 */
#define GPIO_PWM_Motor_C0_PORT                                             GPIOA
#define GPIO_PWM_Motor_C0_PIN                                       DL_GPIO_PIN_8
#define GPIO_PWM_Motor_C0_IOMUX                                    (IOMUX_PINCM19)
#define GPIO_PWM_Motor_C0_IOMUX_FUNC                 IOMUX_PINCM19_PF_TIMA0_CCP0
#define GPIO_PWM_Motor_C0_IDX                               DL_TIMER_CC_0_INDEX
/* GPIO defines for channel 1 */
#define GPIO_PWM_Motor_C1_PORT                                             GPIOA
#define GPIO_PWM_Motor_C1_PIN                                       DL_GPIO_PIN_3
#define GPIO_PWM_Motor_C1_IOMUX                                     (IOMUX_PINCM8)
#define GPIO_PWM_Motor_C1_IOMUX_FUNC                  IOMUX_PINCM8_PF_TIMA0_CCP1
#define GPIO_PWM_Motor_C1_IDX                               DL_TIMER_CC_1_INDEX
/* GPIO defines for channel 2 */
#define GPIO_PWM_Motor_C2_PORT                                             GPIOA
#define GPIO_PWM_Motor_C2_PIN                                      DL_GPIO_PIN_15
#define GPIO_PWM_Motor_C2_IOMUX                                    (IOMUX_PINCM37)
#define GPIO_PWM_Motor_C2_IOMUX_FUNC                 IOMUX_PINCM37_PF_TIMA0_CCP2
#define GPIO_PWM_Motor_C2_IDX                               DL_TIMER_CC_2_INDEX
/* GPIO defines for channel 3 */
#define GPIO_PWM_Motor_C3_PORT                                             GPIOA
#define GPIO_PWM_Motor_C3_PIN                                      DL_GPIO_PIN_12
#define GPIO_PWM_Motor_C3_IOMUX                                    (IOMUX_PINCM34)
#define GPIO_PWM_Motor_C3_IOMUX_FUNC                 IOMUX_PINCM34_PF_TIMA0_CCP3
#define GPIO_PWM_Motor_C3_IDX                               DL_TIMER_CC_3_INDEX




/* Defines for I2C_MPU6050 */
#define I2C_MPU6050_INST                                                  I2C0
#define I2C_MPU6050_INST_IRQHandler                              I2C0_IRQHandler
#define I2C_MPU6050_INST_INT_IRQN                                  I2C0_INT_IRQn
#define I2C_MPU6050_BUS_SPEED_HZ                                        100000
#define GPIO_I2C_MPU6050_SDA_PORT                                          GPIOA
#define GPIO_I2C_MPU6050_SDA_PIN                                    DL_GPIO_PIN_28
#define GPIO_I2C_MPU6050_IOMUX_SDA                                 (IOMUX_PINCM3)
#define GPIO_I2C_MPU6050_IOMUX_SDA_FUNC               IOMUX_PINCM3_PF_I2C0_SDA
#define GPIO_I2C_MPU6050_SCL_PORT                                          GPIOA
```

Figure 7. Sysconfig configuration file
图 7. Sysconfig 配置文件

如图 7 所示，在 CCS 开发环境中，通过 SysConfig 工具完成引脚的图形化配置后，编译时 SysConfig 会自动在 ti_msp_dl_config.h 头文件中生成相应的引脚配置宏定义。这些宏定义的命名规则通常由引脚配置名称与后缀通过下划线连接组成，且引脚参数的修改一般不会影响宏定义名称，这种设计显著提升了引脚配置的灵活性和易用性，降低了开发门槛。完成引脚配置后，开发者仅需调用少量官方库函数(如 GPIO 读写函数)即可实现功能开发。

相比之下，在使用 STM32 进行开发时，往往需要频繁调用大量库函数，这种开发方式不仅要求开发者深入掌握库函数的使用方法，还伴随着大量重复性操作，随着代码规模的增长，开发周期会显著延长，整体开发效率较低。

5. 结论

本文通过理论分析、程序对比和实验验证，系统研究了 TI 系列芯片在嵌入式系统开发中的应用特点。研究表明，相较于传统的 STM32 单片机，TI 系列开发板在智能化应用场景中展现出独特优势。以自动行驶小车为例，TI 芯片的灵活性和高效性为复杂硬件设计提供了更优解决方案。本研究不仅为大学生快速掌握 TI 系列开发板的应用提供了实践指导，也为嵌入式系统教学改革和学科竞赛准备提供了新的思路。未来，随着智能化技术的不断发展，TI 系列芯片在嵌入式系统开发中的应用前景将更加广阔。

基金项目

河南省高等教育教学改革研究与实践项目“应用型本科高校电子信息类专业三育融合课程建设研究与实践”(项目编号：2021SJGLX533)；黄淮学院教育教学改革研究项目“课堂教学改革研究与实践-以《半导体光电子学》(项目编号：2019XJGLX0173)；黄淮学院高等教育教学改革研究项目“工程教育认证背景下智能照明技术课程教学策略研究与实践”(项目编号：2024XJGZLX34)；黄淮学院专创融合课程项目“智能照明技术”(项目编号：34132603)；黄淮学院青年骨干教师(项目编号：501618401002)；黄淮学院教育教学改革研究项目(项目编号：2024XJGLX08，项目名称：新工科背景下电子信息类专业课程思政建设研究与实践)。

参考文献

[1] 梁晶, 吴银琴. 嵌入式系统原理与应用[M]. 北京: 人民邮电出版社, 2021: 337.

[2] 苏李果, 宋丽. STM32 嵌入式技术应用开发全案例实践[M]. 北京: 人民邮电出版社: 2020: 321.

[3] 吴杰, 周涛. 基于 TI MSPM0 的嵌入式系统开发实践[J]. 单片机与嵌入式系统应用, 2022, 22(7): 12-16.

[4] 胡江瑜, 李光琴, 王莹. 基于 STM32 的循迹机器车设计[J].物联网技术, 2025, 15(2): 138-140+143.

[5] 龙江腾. 基于运动姿态传感器 MPU6050 的人体手势识别研究[D]: [硕士学位论文]. 抚州: 东华理工大学, 2021.

[6] 刘洋, 王磊. 嵌入式系统开发中的图形化配置工具研究[J]. 计算机应用研究, 2021, 38(10): 89-93.