武汉工程大学

2018 届毕业（设计）论文

题　　目	基于嵌入式微控制器的魔方还原硬件系统设计
专业班级	电子信息工程 3 班
学　　号	1404200525
学生姓名	于靖一
第一指导教师	邹连英
指导教师职称	副教授
第二指导教师	
指导教师职称	
学院名称	电气信息学院

完成日期：　2018 年 6 月 12 日

基于嵌入式微控制器的魔方还原硬件系统设计

Design of Rubik's Cube Restoration Hardware System Based on Embedded Microcontroller

学生姓名 于靖一

第一指导教师 邹连英

第二指导教师

摘　要

本文通过比较常见的六中心旋转和单面旋转还原魔方机械结构，同时进行步进电机驱动控制选型和微控制器选型，设计出一种使用步进电机驱动的四爪夹持旋转魔方硬件结构。该系统使用 6 个步进电机，通过 2 个步进电机控制爪子的开合，4 个步进电机控制四面的旋转，使用爪子的开合旋转来使魔方整体旋转，从而达到旋转魔方每个面的目的。本系统将控制部分分为了运动控制器和解算控制器，两部分控制器通过串口发送和接收指令。运动控制器主要对应魔方的转动操作和命令的解析，而解算控制器则对应魔方的信息采集和解算还原步骤。其中魔方信息的采集采用摄像头进行颜色识别，按键和触摸屏可以进行修正；解算还原步骤采用降群法，通过串口发送解算完成的步骤。

本系统两部分可以相互独立，互不影响，同时采用了通用的通信协议，可以为以后的进一步升级提供方便。该系统克服了常见系统不便于拆装或旋转速度慢的缺点。并且具有模块化的特性，可维护性强，成本低，可广泛用于教学和展示环境。

关键词：魔方还原；步进电机；Arduino；STM32；降群法

Abstract

In this paper, by comparing the common six-center rotation and single-sided rotation to restore the Rubik's cube mechanical structure, the stepper motor drive control selection and the microcontroller selection, a four-jaw clamping rotary cube hardware driven by a stepper motor is designed. structure. The system uses six stepper motors, two stepper motors to control the opening and closing of the claws, four stepper motors to control four-sided rotation, using the pawl's opening and closing rotation to make the Rubik's cube rotate as a whole, so as to reach each side of the Rubik's Cube. the goal of. The system divides the control part into a motion controller and a solver controller. The two controllers send and receive instructions through the serial port. The motion controller mainly corresponds to the Rubik's rotation operation and the analysis of the command, while the solver control corresponds to the Rubik's cube information collection and solution restoration steps. The collection of Rubik's cube information uses the camera to identify the color, and the buttons and the touch screen can be modified; the solver restore step uses a Thistlethwaite method and sends the solved steps through the serial port.

The two parts of the system can be mutually independent and do not affect each other. At the same time, a common communication protocol is adopted, which can provide convenience for further upgrades. This system overcomes the disadvantages of common systems that are inconvenient to dismantle or rotate slowly. And it has modularity, high maintainability, low cost, and can be widely used in teaching and display environments.

Keywords: Rubik's Cube; stepper motor; Arduino; STM32; Thistlethwaite method

目　　录

第1章 绪论

1.1 引言

魔方，为3x3x3的立方体结构，每一面由1个中心块，4个棱块和4个角块组成[1]。魔方总共有6面，通过1个中心轴连接至各个中心块组合而成。当组合在一起的时候，每个零件会相互牵制而不会散开，并且每一面都可以水平转动而不影响到其他方块。本设计围绕着三阶魔方的还原进行，将讨论如何设计一种还原魔方的硬件系统，从而达到成本低、结构简单、易于维护和拓展性强的优点。

关于魔方的还原，德国数学家科先巴于 1992 年最先提出了一种用于寻找魔方还原方法的新思路，即在魔方的还原过程中，可以将还原过程划分为两个阶段，首先是将其转动为已知可以还原的颜色组合，然后再将已知可以还原的颜色组合还原[2]。限于当时的计算机存储空间，无法将已知的颜色组合全部统计，故到1995年，美国弗罗里达大学的里德通过计算发现，最多经过30次转动，就可以将魔方的任意一种颜色组合复原[3]。这个复原任意组合所需的最小转动次数即为“上帝之数”[4]。经过不断的改进和完善，“上帝之数”不断的被减小，最终2010年，美国加利福尼亚州科学家将之定为了20。

尽管魔方可以在很少的转动数目之内被还原，但所需存储公式的空间花费太大，并且需要的计算量太大，故本次设计旨在设计一种可以方便还原魔方的硬件系统，并不刻意追求速度。通过旋转魔方的各个面，对魔方进行还原。

1.2 国内外研究现状

1.2.1 魔方旋转标记方法

为了记录还原、打乱的过程或公式的步骤，一般使用“辛马斯特标记”（Singmaster notation）来书写（由大卫·辛马斯特发明）。该标记的规则为U(Up)、D(Down)、L(Left)、R(Right)、F(Front)和B(Back)分别代表魔方的上、下、左、右、前和后各面。如果将该面顺时针旋转，则直接写上符号；若是逆时针旋转，则在符号后加上“’”或是“i”；若是旋转180°，则在符号后加上“2”或是“²”。本文中对魔方还原的记录过程均采用辛马斯特标记方法，即“F2’”即为将魔方正面逆时针旋转180°。

1.2.2 还原魔方的机械结构设计

还原魔方的机械结构目前分为两大类，一类为六中心旋转结构，该结构是为了追求还原速度的最快，将原本的魔方结构进行改动，加入相关连接部件与机械结构结合；另一类为单面旋转结构，通过机械臂或者类机械臂装置模拟人类旋转魔方的操作，不用将魔方的结构进行修改[5]。

前者将魔方的各个面的中心块加以改造，添加相关的部件或者去掉部件与机械部分结合，通过机械部分的旋转带动魔方的旋转。一般对魔方的六个面均进行改装，方便进行六面的旋转。这种系统速度快，可将还原速度控制在几十秒左右，但破坏了魔方整体的结构，不方便取下进行打乱。

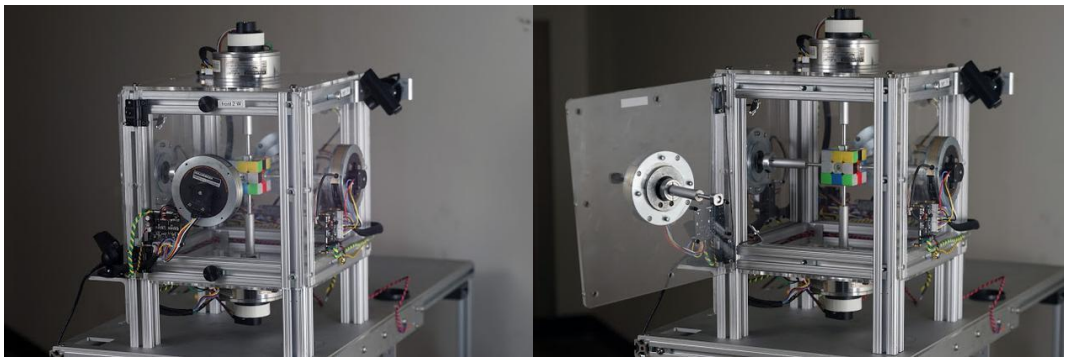


图 1.1 六中心旋转机械结构

后者一般模拟人类对魔方的旋转，每一次只针对魔方的单个面进行操作。比较典型的为使用 LEGO 公司的 MINDSTORMS 积木套件制作而成的，使用底座进行旋转魔方的单个面，有一个机械结构用来对魔方进行翻转，从而旋转不同面[6]。这种结构可以保持魔方机械结构的不变，但动作繁琐，多余步骤多，还原魔方的速度慢。

图 1.2 基于 LEGO 积木的单面旋转结构

1.3 本文工作及结构安排

本文的主要工作重点为设计一种还原魔方的硬件系统，通过嵌入式微控制器进行控制，将整个魔方的还原过程直观的展现。本设计有结构简单、易于维护和成本低的优点，并可用于教育和展览场景。该设计可拓展性强，通过简单的适配工作，可以将其他解魔方的算法移植到本硬件系统上，方便对各种算法进行测试和展示。

第 1 章：介绍了魔方的基本概念，转动的标记方法以及已有的魔方还原硬件结构。

第 2 章：从总体上对硬件系统进行阐述，对还原魔方的硬件系统进行了总体设计。

第 3 章：说明了系统的硬件设计，其中包括机械部分和电路设计部分。

第 4 章：从软件层面说明各项功能的实现方法和实现思路。

第2章 系统总体设计

2.1 魔方还原硬件系统需求与特点分析

基于嵌入式微控制器的魔方还原硬件系统是一个能将任意打乱的三阶魔方复原的机械系统。其中包括机械部分、驱动部分以及控制部分。其机械部分为 2 个步进电机皮带传动装置控制爪子开合，4 个步进电机操作魔方四个面进行旋转。驱动部分采用成熟的步进电机驱动芯片，可通过控制器对步进电机的转动角度进行精确控制。控制部分采用多嵌入式微控制器组合的方式，可以提高系统的可拓展性，同时对系统进行模块化设计，方便以后的维护及更新换代。

通过魔方还原硬件系统，可以使其对魔方各个面进行旋转，从而达到还原魔方的目的。本系统兼容性强，通过通用的接口输入即可实现对魔方各个面的控制。内置命令解析算法，可以将输入的新马斯特标记转换为具体的操作，对用户友好，方便后期二次开发。

控制部分的嵌入式微控制器采用 Atmel 8 位微控制器，该微控制器采用 Arduino IDE 开源开发平台进行编程，可拓展性强并且具有开发周期短的优点。此微控制器具有丰富的 GPIO（General Purpose Input Output）引脚，可对机械部分进行完整的控制。功能多，可适用于该系统的开发。

解算魔方的部分采用 ST 公司的 STM32F4 系列 32 位 MCU，可以实现摄像头输入魔方颜色信息，并将还原步骤显示在 LCD 显示屏上。显示直观，方便用户进行操作。

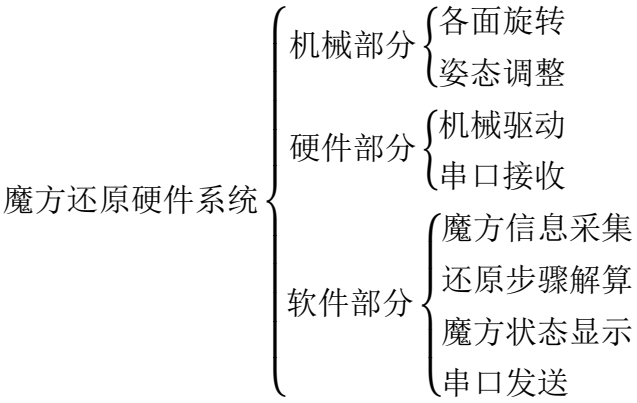


图 2.1 魔方还原硬件系统整体图

2.2 魔方还原硬件系统性能分析

本设计的核心问题是设计一种可以还原魔方的硬件系统，使任意打乱的魔方得以还原。该系统具有可移植性强，可拓展性强和易于维护的特点，所以本设计分几个大的部分，分别为机械部分、驱动部分和控制部分。

2.2.1 功能性需求分析

（1）机械部分：设计一种可以将魔方各个面进行旋转，并且不破坏魔方的整体结构的机械结构，同时要保持还原的高速和系统的可维护性，以及成本低。使用者可以将魔方手动打乱，然后将之放入本设计的机械结构中，通过旋转每一面将魔方逐步还原。

（2）驱动部分：该部分应选取一种合理的驱动方案，对机械部分采用的步进电机作为爪子的旋转动力，同时控制爪子的松紧进行驱动。由于本设计需要将还原魔方的时间尽可能缩短，所以需要在保证步进电机力度足够大的情况下，尽可能提高转动速度。这就要求设计一种持续可靠地以大电流驱动步进电机的驱动。

（3）控制部分：为了满足模块化的需求，本系统将控制部分分为两个部分：运动控制部分和解算控制部分。两部分之间进行通信，达到本设计模块化、接口化的设计需求。

其中运动控制部分为控制六个电机的旋转，通过编程实现对魔方各个面的旋转操作，并且接受上位机发出的控制信号，并对控制信号进行处理和解析。

解算控制部分通过摄像头或者按键综合输入魔方各面的颜色信息，并使用控制器内的魔方还原算法，产生魔方还原步骤并发送至下位机。显示屏上可以实时显示魔方还原进程以及各面颜色，方便进行对比与监视。

2.2.2 非功能性需求分析

（1）拓展性：魔方还原硬件系统作为魔方还原操作的实现，其还原操作的步骤多少和时间长短均与魔方还原的计算方法有关，因此设计一个拓展性强的系统，引出接口方便以后的性能提升是很重要的。本设计通过通用的接口与解算控制部分进行连接，方便以后对系统进行升级维护。

（2）稳定性：魔方的还原涉及很多步骤，并且前面的步骤会为后面的步骤打下基础，任何一个步骤的不正确操作必定会导致最终的还原失败。同时要保证系统在多次，大量的工作压力下保持良好的状态。本设计的过程通过理论的论证，并且通过在多次实验的过程中测定性能，以保证系统能长时间稳定运行。

（3）可操作性：由于摄像头采集图像的多变性，无法保证在所有时刻采集的魔方颜色信息均为可用，因此要加入相关的修正操作，必要时通过人为方式进行干预操作，保证各面颜色信息的录入成功。

第3章 系统硬件结构

3.1 机械结构设计

3.1.1 电机选型

还原魔方硬件系统的关键在于采取一种合理的方式，将魔方的各个面进行转动，同时每一面的转动角度应可控。在不破坏魔方整体结构的前提下，需要一种精确转动同时又有足够大的力矩的转动方式。

（1）直流电机

直流电机（DC Motor）是指依靠直流电驱动的电动机，其基本构造包括“电枢”、“场磁铁”、“集电环”和电刷。当线圈通电时，转子周围产生磁场，转子的左侧被推离左侧的磁铁，然后被吸引至右侧的磁铁，从而产生转动[7]。直流电机分为直流有刷电机、直流无刷电机和切换式磁阻电动机，其中使用最为广泛的是前两种步进电机。

直流电机控制电压及电流的大小就可以控制电机旋转的速度，但由于其控制方式通常采用 PWM 的控制方式，没有一个明确的量来反应旋转的角度，一般需要反馈控制系统才可以实现间接地控制电机位置

（2）步进电机

步进电机（Stepper Motor）是直流无刷电机的一种，其具有如齿轮状突起（小齿）相契合的定子和转子，可由切换流向定子线圈中的电流，以一定的角度逐步转动的电机[8]。

步进电机可以不需要转动量反馈检测或编码器装置，由于其收到一个脉冲就会旋转特定的角度，因此在不超过负载的情况下步进电机可以正确的将脉冲量转化为转动量，被广泛的用于各种精确控制系统中。

（3）伺服电机

伺服电机（Servo Motor）是对使用伺服机构的电机总称，即具有反馈控制的电机[9]。舵机为伺服电机的一种，被广泛采用于控制船舵、飞机舵等方向控制。伺服电机的控制为三环控制，即电流环、速度环和位置环，其分别能反应电机的角加速度、角速度和旋转位置。控制器通过三个反馈量控制电机的驱动电流，进而实现控制电机的速度和位置。而舵机则是对三环的伺服电机进行缩减，缩减为一环，即位置环。

伺服电机的特点在于精确度高，但由于其成本高，并且其精度受制于编码器的码盘，与控制算法也有很大关系，因此被用于机床、工业机械臂、机器人等需要高精度的环境。

结合项目的特点要求，需要精准的将魔方各面旋转 90° 或 180° ，由于直流电机的旋转角度无法做到精确控制，伺服电机虽然能精确控制角度，但其有力矩小和成本高的缺点，一般都将其与减速箱合用，结构复杂，所以我们选择步进电机作为系统的驱动电机。

步进电机只需要通过脉冲信号，即可实现高精度的定位，并可以在目标位置准确停止。本系统使用的是 4 相步进电机，其基本步级角为 1.8° ，所以可以将步进电机转一圈分为 200 等份（=360°/1.8°），可以以此方式来细分每次行进量作为定位基准。对于 4 个旋转用途的步进电机，使用的是 42*42*40mm 的外形尺寸、轴径 5mm、轴长 23mm 的具有 0.45N.m 的扭矩，额定电流为 1.7A 的 42 步进电机。对于 2 个可以操作爪子开合传动带驱动步进电机，使用的是 42*42*40mm 的外形尺寸、轴径 5mm、轴长 17mm 的具有 0.45N.m 的扭矩，额定电流为 1.5A 的 42 步进电机。经测试其在 0.6A 的电流下即可完成对魔方的旋转，在 1.0A 的电流下可以实现控制爪子的开合。

3.1.2 机械结构设计

为了实现一种可以在不破坏魔方的结构下对魔方进行还原的机械结构，本系统使用 4 个步进电机驱动爪子旋转，2 个步进电机驱动爪子开合的机械结构。大部分部件采用 3D 打印构件组成，4 个步进电机在 2 个带有同步带的步进电机的带动下完成开合的动作，同时 4 个步进电机还可以对魔方的 4 个面进行旋转。

由于步进电机的旋转步数精确，只需要给一定数量的脉冲，不需要反馈就可以将脉冲量转换为旋转的度数，为了精简系统，本硬件系统没有引入电机旋转量反馈。

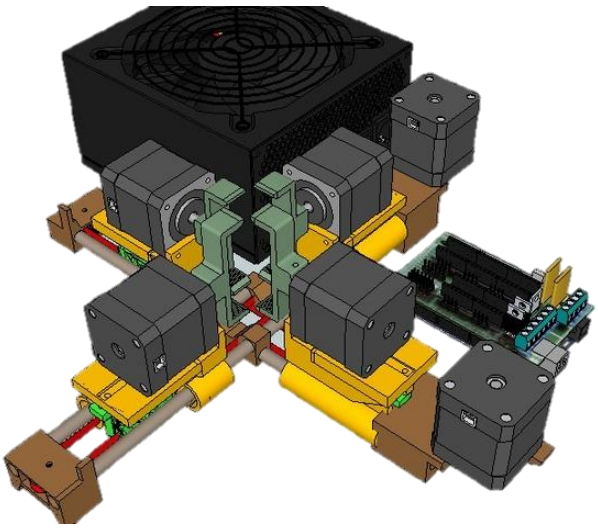
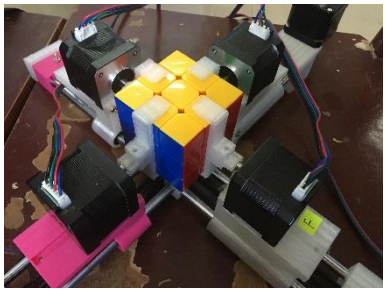
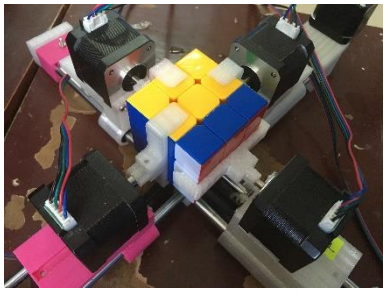


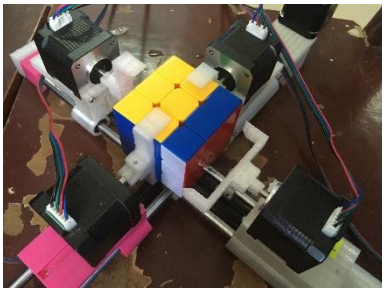
图 3.1 6 步进电机还原魔方机械结构

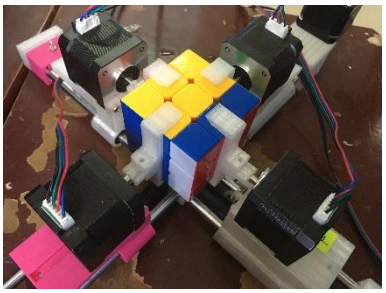
初始状态的魔方摆放为顶面向上，红色面向前，由于魔方的特性，无论怎样旋转，中心块都不会移动，确定了两个面的摆放位置，其他面的位置也就确定下来。本系统爪子闭合状态下可对魔方的前、后、左和右面进行旋转，当需要对魔方的面进行旋转 90° 操作时，

如图 3.2，爪子在旋转 90°后必须进行还原操作，才可以进行下一个旋转步骤的操作；当需要对魔方的面进行 180°旋转时，则无需进行还原操作即可进行下一个旋转步骤的操作，如图 3.3。


（1）原始状态


（2）F 爪子顺时针旋转 90°


（3）FB 爪子松开


（4）F 爪子恢复初始状态

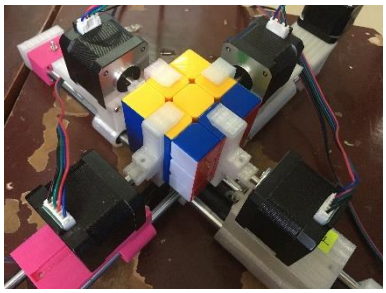
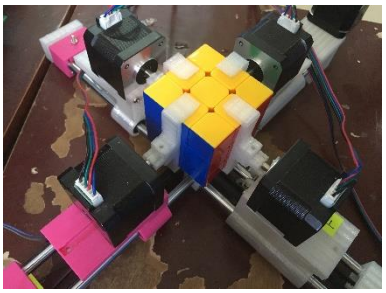

（5）FB 爪子夹紧

图 3.2 旋转 90°动作


（1）原始状态

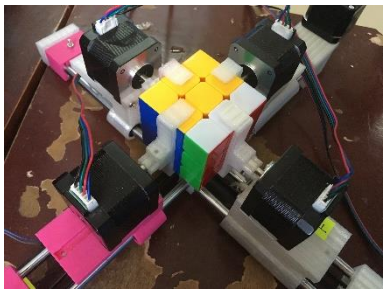
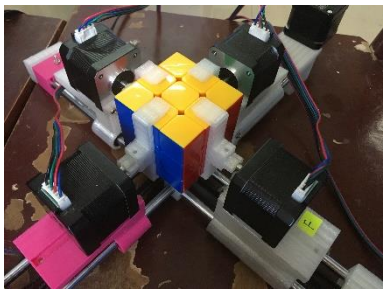
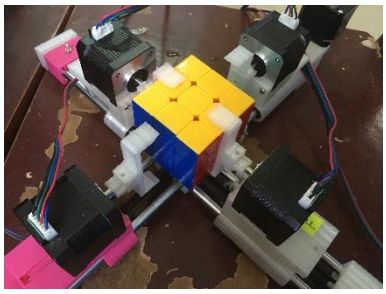

（2）F 爪子顺时针旋转 180°

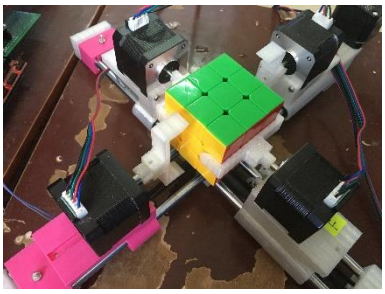
图 3.3 旋转 180°动作

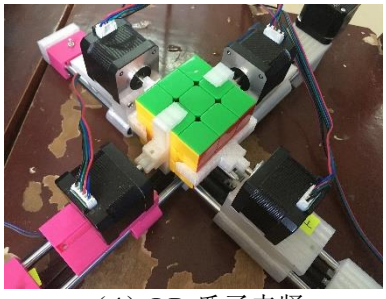
如果想对魔方的顶面或者底面进行旋转，则采取先将左右爪子分开，通过前后两个爪子的同时旋转，将魔方整体逆时针旋转 90°，然后左右爪子闭合，如图 3.4。现在的状态下旋转左右两个爪子就为旋转魔方的上下面。通过对魔方整体姿态的旋转，可以间接的实现对魔方六个面的旋转操作。

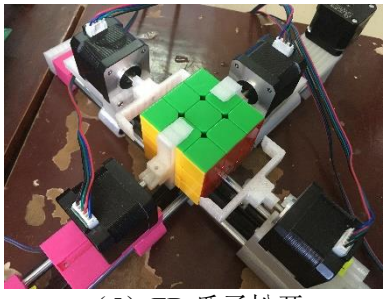
本系统经实际测试，性能良好，可以将魔方每一个面准确的进行旋转，步骤操作速度快。与 3D 打印部件的结合，可以大大提高系统的可定制性，针对不同的魔方可以制作不同的爪子进行夹持，通用性强，适用广泛。

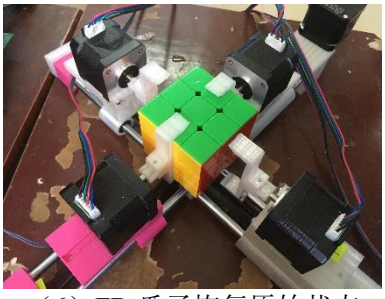

（1）原始状态


（2）LR 爪子松开


（3）FB 爪子逆时针旋转 90°


（4）LR 爪子夹紧


（5）FB 爪子松开


（6）FB 爪子恢复原始状态

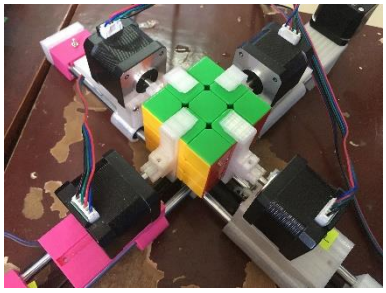

（7）FB 爪子夹紧

图 3.4 魔方姿态变换

3.2 驱动电路设计

3.2.1 电机驱动设计

魔方还原系统采用的是步进电机驱动的机械结构，对于步进电机的驱动电路，首先应满足可提供足够大电流的能力。嵌入式微控制器 ATmega2560 的 GPIO 最大的拉电流为 200mA，最大的灌电流为 200mA 对于单个引脚平均分配不到 50mA，这样的驱动能力是不能直接驱动步进电机的。因此要设计一种扩流的结构，使其能可控的对外部器件进行功率控制。

（1）ULN2003A

ULN2003A 驱动器具有一个高电压，高电流的达林顿三极管管阵列[10]。每个芯片中有 7 个 NPN 达林顿三极管对，单个三极管对可以用来驱动 500mA 的电流，最大输出电压可以到 50V。其典型应用为继电器驱动器，LED 驱动器，线路驱动器和逻辑缓冲器。ULN2003A 驱动器每个三极管对基极都串联了一个 2.7kΩ 的电阻，可以直接与 TTL 或 5V CMOS 器件一起工作。

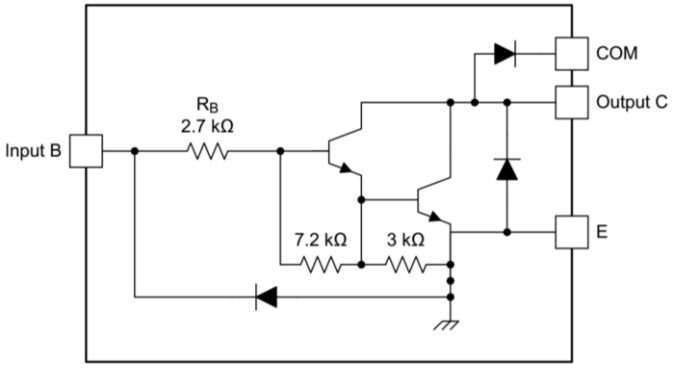


图 3.5 单个达林顿晶体管输出驱动结构框图

ULN2003A 有 16 个引脚，其中有 7 个输入引脚 IN，7 个输出引脚 OUT，1 个引脚 GND，还有一个 COM 引脚。其中的 COM 引脚可以悬空不接，其作用有两种，一是为了检测输出端是否为正常：如果将 COM 接地，负载一端接高电平，另外一端接芯片的输出，则电路就会导通，负载中就会有电流经过，可凭此判断负载是否正常；二是为了保护器件使用：如果外部器件接继电器或者电机，因为电感的作用，在关闭过程中产生低于电源电位的反电动势，容易击穿器件，可将 COM 接电源正，从而减少反电动势的影响。

如果使用 ULN2003A 驱动步进电机，应将步进电机每一相与芯片输出端 OUT 连接，同时 COM 接电源，微控制器接 IN，当控制器的引脚输出大于 3V 的电平时，三极管导通，从而使输出端与地相连，得以驱动外部步进电机。

（2）L298N

L298N 为 ST 公司的高电压、大电流的双 H 桥电机驱动芯片[11]。具有耐压值高，工作电流大的特点，其工作电压最大可达 46V，每一个 H 桥在直流驱动的情况下可以达到 2A 的工作电流。

H 桥（H-Bridge）是一种被广泛应用的直流电机驱动电路，由 4 个晶体管所构成。当 Q1、Q4 导通，Q2，Q3 闭合时，电流从左到右流过电机；当 Q2，Q3 导通，Q1，Q4 闭合时，电流从右向左流过电机。通过 4 个晶体管的配合工作，可以使电机正转或者反转。

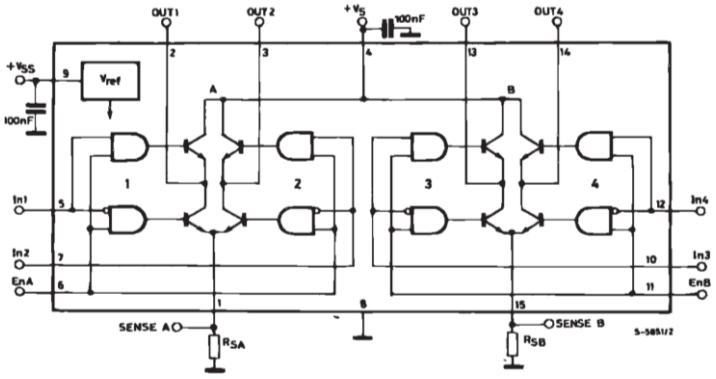


图 3.6 L298N 结构框图

对于 L298N，其中有多个逻辑器件，可以防止误操作导致的 Q1 和 Q2 或 Q3 和 Q4 同时导通，防止短路造成的电源烧毁被广泛采用于各种电机的驱动控制。

（3）A4988

A4988 是一个完整的微型步进电机驱动，其中有内置晶体管为了简化外围电路[12]。A4988 可以用来驱动两相四线步进电机，有 1 步，1/2 步、1/4 步、1/16 步细分模式，最大输出电压可达 35V，最大输出电流可达±2A。A4988 包括一个固定的关断电流调节器，可用于慢或混合衰减模式。

A4988 中内置两路 H 桥 D-MOS 管，并且还有外围的控制电路。微控制器仅需发送一个脉冲到 STEP 引脚上，步进电机就会旋转一步，这可以极大的方便微控制器的编程操作，简化程序。A4988 为 28 引脚的 QFN 封装，尺寸仅为 5*5*0.9mm，非常小巧，可用于各种多路步进电机的控制应用中。

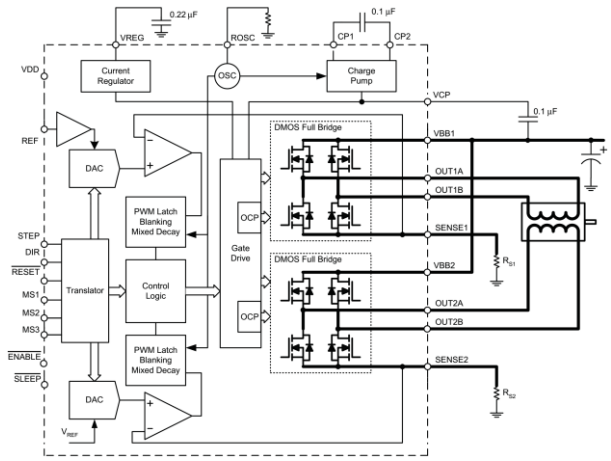


图 3.7 A4988 结构框图

（4）DRV8825

DRV8825 是一个完整的步进电机驱动，可以用于打印机、扫描仪和其他自动化装置[13]。其中包括了两个H桥驱动和一个微型步进电机控制器，可以用来驱动两相四线制步进电机，最大驱动电流为 2.5A 最大驱动电压为 45V。

其 MODE0、MODE1 和 MODE2 引脚可以提供设置为最大 1/32 细分，DECAY 引脚可以设置三种不同的衰减模式：快速、慢速和混合衰减模式。SLEEP 引脚可以控制驱动的开启和关闭，以进入低功耗模式。同时还有 FAULT 引脚，可以将驱动的状态输出，微控制器可以以此来判断驱动的状态来执行相应的动作。

DRV8825 与 A4988 相似，只需要 STEP 引脚输入脉冲，脉冲的个数为步进电机转动的步数，同时还可以通过 DIR 来控制步进电机旋转的方向。

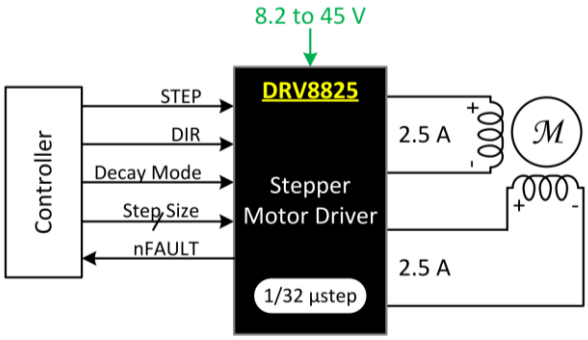


图 3.8 DRV8825 驱动电机最小系统

综上所述，ULN2003A 与 L298N 均为传统的电机驱动器，如果想将之用于步进电机的驱动，微控制器必须输出特定的时序，才可以让步进电机旋转，同时每个驱动需要占用 4 个单片机的 GPIO，6 个步进电机需要占用 24 个 GPIO，消耗控制器的资源量大。

A4988 和 DRV8825 为步进电机驱动器，微控制器只需要输出脉冲，步进电机中的控制电路就会产生相关的信号，使步进电机旋转一步，例如想让步进电机旋转 180°，则需要发送 100（180°/1.8°）个脉冲信号，发送信号的频率反映在步进电机上就为步进电机转动的速度，通过控制发出的脉冲个数和脉冲频率，就可以使步进电机以可控的速度旋转可控的角度。同时微控制器可以改变 DIR 输入的高低来控制步进电机正转与反转，具体的实现均通过芯片内部逻辑控制电路实现，微控制器仅需要控制输出引脚的高低电平，就可以达到目标的要求。使用集成的步进电机驱动器，只需要占用 12 个 GPIO，还可以大大的节省微控制器的资源，方便进行其他处理操作。

对于 A4988 和 DRV8825，前者最大工作电流为 2.0A，推荐工作电流为 1.5A；后者最大工作电流为 2.5A。考虑到单个步进电机的额定电流为 1.7A，步进电机驱动接近满载时发热量很大。为了提高系统稳定性，降低发热，采用后者作为步进电机的驱动器。

3.2.2 动作控制器设计

作为系统的中间层，动作控制器需要实现的是读取并处理上位机发送的步骤信息，并将步骤信息转换为具体的机械动作，控制步进电机驱动器执行相关的动作。

Arduino 为一种软件和硬件均具备的平台，使用了 Atmel AVR 微控制器，软件主要采用 Arduino IDE 进行编写程序，可以使用 C 或 C++进行编程，内置开源的丰富的库函数，可大大加快程序的编写速度，减少项目的开发周期。硬件平台具备采用 Atmel ATmega328P 微控制器的 Arduino Uno、Arduino Nano、Arduino Mini；采用 ATmega32U4 微控制器的 Arduino Leonardo、Arduino Micro；采用 Atmega2560 微控制器的 Arduino Mega2560 等。

对于 ATmega328P 和 ATmega2560，两者都为 8 位 MCU，时钟频率 16MHz，处理速度 16MIPS。前者有 23 个 GPIO（最大），2 个 SPI，1 个 I2C，1 个 UART，其 RAM 为 2KB，

EEPROM 为 1KB，Flash 为 32KB；后者有 86 个 GPIO（最大），5 个 SPI，1 个 I2C，4 个 UART，其 RAM 为 8KB，EEPROM 为 4KB，FLASH 为 256KB。考虑到系统采用大量的 GPIO 作为控制电机的输出，同时还有 2 个作为串口通信，1 个为系统状态指示灯，再加上串口程序下载、ISP、复位和晶振输入所占用的 IO 口，考虑到以后的系统拓展，最后采用 ATmega2560 作为系统动作微控制器，选用 Arduino MEGA2560 作为系统硬件控制板。

ATmega2560 微控制器外围电路简单，仅需要晶振提供工作时钟和复位电路，就可以良好的工作于各种环境中。

（1）电源电路设计

步进电机需要 8.2~45V 的电压驱动，考虑到易用性、安全性和普遍性，采用 12V 开关电源对整个系统进行供电。由于 ATmega2560 微控制器需要采用 1.8~5.5V 的电压才能工作，其 GPIO 的电平与输入电压有关，为了使 DRV8825 步进电机驱动能有合适的脉冲电平，采用 AMS1117-5.0 作为微控制器部分的稳压芯片。

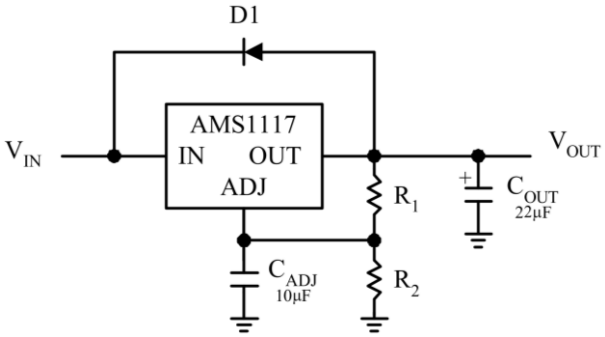


图 3.9 AMS1117 应用电路图

AMS1117-5.0 为输出电压固定为 5.0V 的低压降稳压器，其最大可以输入 15V 的电压，最大电流为 1.5A（输入输出压差为 5V），正常工作电流应小于 0.8A。为了稳定性，输出端与地之间应接 22μF 的钽电容，同时可以在输出端与输入端连接一个反向耐压值高于正常工作时输入输出电压差的二极管，当输入电压消失时使电容中的电压通过二极管泄放，保护 AMS1117-5.0 稳压器。二极管常采用整流管 1N4001。

（2）时钟电路设计

对于晶振选择 16MHz 的外部晶振来让微处理器速度达到最快，图 3.10 的 C1 和 C2 为晶振的起振电容，在此设计中 C1 与 C2 应大小相等，由于电容的大小与振荡器的稳定有关，但电容越大，起振时间越长。具体的推荐电容大小为技术手册中要求的大小范围内均可。在此选用 22pF 的电容作为晶振的起振电容。

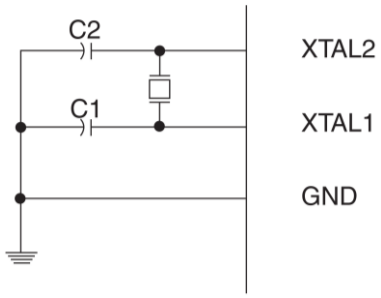


图 3.10 晶振连接方式

（3）复位电路设计

ATmega2560 内置有上电复位电路，当输入电压超过 1.0V 时微控制器进入复位状态，随后当 RESET 引脚的电平输入大于 0.9 倍的工作电压时，单片机进入复位状态，经过一段时间后复位完成，上电复位过程结束。

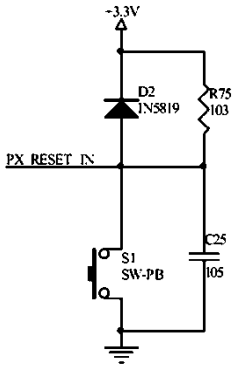


图 3.11 复位电路

如果在微控制器正常运行的过程中需要外部复位，需要设计一个外部复位电路，来达到复位的目的。图 3.11 为微控制器的复位电路。复位电路必须能降 RESET 引脚上的电平拉低到 0.2 倍的电源电压，同时还应能保持一段时间让微控制器完成复位。对于满足最小复位时间所需选用的电容充放电组合，由于不在本文讨论范围内，故不再进行计算。在此选用 10K 的电阻和 22pF 的电容，添加二极管的作用是为了防止外部静电干扰，将 RESET 钳位至电源电压还可以快速放掉电容中的电能，提高响应速度。

（4）下载电路设计

ATmega2560 支持 Boot Loader 启动，可以在用户程序启动之前执行一些程序，本设计将 ATmega2560 烧写 Arduino BootLoader，方便使用串口对用户程序进行下载。在硬件电路的设计上，将 TX、RX 和 RESET 引出，方便程序的下载。只需将 USB 转串口的 RX、TX 和 DTR 分别与 ATmega2560 的 TX、RX 和 RESET 连接，就可在下载的过程中自动进入 BootLoader，从而下载程序。DTR（Data Terminal Ready）引脚为数据终端准备就绪标志，当计算机端准备向微控制器下载程序时，此引脚会从高电平变为低电平。将此引脚与

RESET 相连，可以在计算机准备下载程序时将微控制器复位，进入 BootLoader 准备下载程序。DTR 与 RESET 串联 100nF 的电容，目的是利用电容的充放电提供足够长的时间供微控制器复位，与复位电路的设计类似。

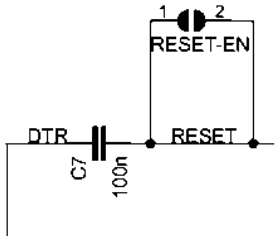


图 3.12 下载自动复位电路

（5）步进电机驱动电路设计

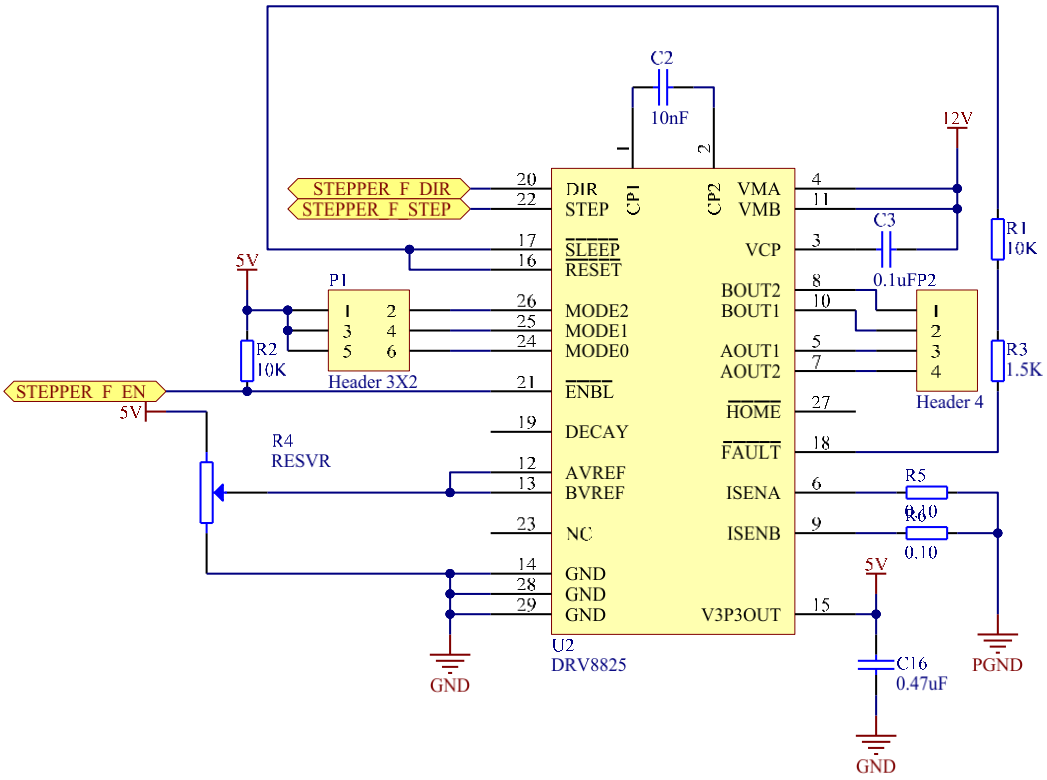


图 3.13 前方电机驱动原理图

本设计采用 6 个步进电机设计，每个步进电机需要 EN、DIR 和 STEP 三个引脚对其控制，其中 EN 控制步进电机驱动器的开启与关闭，DIR 控制步进电机旋转的方向，STEP 引脚控制步进电机的步数，STEP 引脚每发出一个脉冲步进电机旋转一步。

图 3.13 为 DRV8825 步进电机驱动芯片的外围电路图，其中 P1 为设置步进电机细分的跳线，通过跳线帽来设置细分；P2 为电机驱动输出，连接步进电机。表 3.1 为步进电机驱动模块与 ATmega2560 微控制器的引脚连接表格，每个步进电机占用 3 个 GPIO，共占用 18 个 GPIO。

表 3.1 步进电机驱动与 ATmega2560 连接

说明	引脚名称	连接引脚
前方步进电机	F_STEP_PIN	D54
	F_DIR_PIN	D55
	F_ENABLE_PIN	D38
后方步进电机	B_STEP_PIN	D60
	B_DIR_PIN	D61
	B_ENABLE_PIN	D56
左方步进电机	L_STEP_PIN	D26
	L_DIR_PIN	D28
	L_ENABLE_PIN	D24
右方步进电机	R_STEP_PIN	D36
	R_DIR_PIN	D34
	R_ENABLE_PIN	D30
前后松紧步进电机	FB_STEP_PIN	D46
	FB_DIR_PIN	D48
	FB_ENABLE_PIN	D62
左右松紧步进电机	LR_STEP_PIN	D47
	LR_DIR_PIN	D32
	LR_ENABLE_PIN	D35

3.2.3 解算控制器设计

解算控制器的作用为获取魔方颜色、解算还原步骤和将还原步骤输出到运动控制器。由于需要获取魔方颜色和解算还原步骤，并将魔方还原的过程直观的体现出来，采用 8 位微控制器资源以及运算速度都不能满足要求。所以本系统采用 ST 公司的 STM32F407ZGT6 作为主控芯片（以后简称 STM32）。

STM32F407ZGT6 是一款 LQFP144 封装的微控制器，具有 144 个引脚。其中有 112 个 GPIO，SRAM 为 192KB，FLASH 为 1024KB。处理器主频为 168MHz，运算速度能达到 210DMIPS，其中还有一个 DSP 核可以加速运算[14]。STM32F407ZGT6 具有 6 个串口，可以用来传输步骤信息；1 个 FSMC，可以用来传输显示数据；2 个 DMA，可以用来传输摄像头数据。

（1）电源电路设计

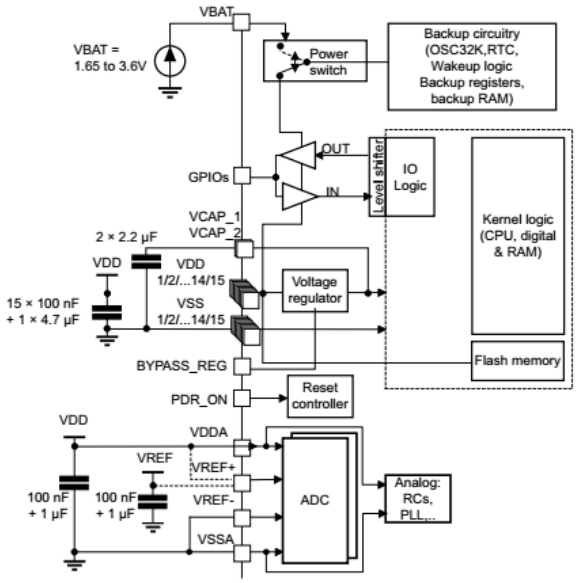


图 3.14 STM32 的供电结构框图

如图 3.14 所示，STM32 需要为其提供三个电源，分别为 VDD，VDDA 以及 VBAT。其中 VDD 为外部电源，用来给 GPIO 和内部调度器供电；VDDA 为模拟部分供电，用来给 ADC、DAC、复位模块、RC 和 PLL 供电；VBAT 是电池供电，用来给 RTC、外部 32kHZ 低速晶振和后备寄存器供电。如果想让系统能正常的工作，VDD 和 VDDA 必须都有 1.8~3.6V 的供电，并且应该同时供电。系统中 VSS 和 VSSA 为地，VREF 为 ADC 参考电平。如图 3.15 所示，必须注意 STM32 的 VCAP_1 和 VCAP_2 引脚必须接一个 22μF 的电容再接地，不能直接接地，这两个引脚是给内部调压器为了连接外部电容所设计的，如果不接或者误接，则不能给系统内核提供 1.2V 标准电压，微控制器就无法工作。

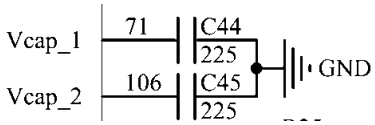


图 3.15 Vcap 的连接方式

（2）时钟电路设计

对于晶振使用 8MHz 外部晶振，晶振起振电容采用 5~25pF 的电容。本系统结合实际情况，选用 22pF 的电容作为晶振的起振电容，如图 3.16 和图 3.17 所示。

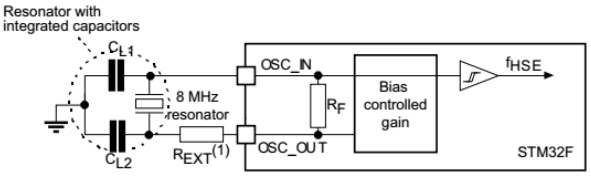


图 3.16 8Mhz 晶振的典型应用框图

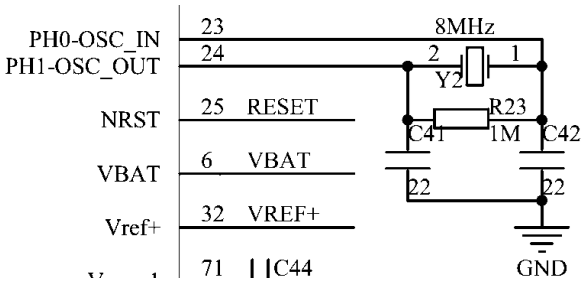


图 3.17 晶振的连接

（3）复位电路设计

复位电路微控制器内部已有上拉电阻，故只需外部接 0.1μF 的电容即可实现复位功能。但实际中一般会外接 10K 的上拉电阻，如图 3.19 所示，为了保证 NRST 引脚上的电平稳定。

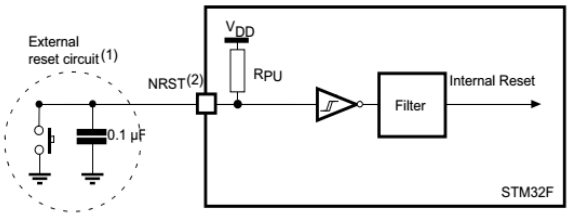


图 3.18 外部复位电路结构框图

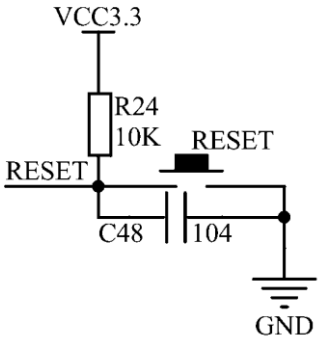


图 3.19 复位电路

（4）LCD 接口电路设计

对于 LCD 接口电路，LCD 通过排针与主板相连。本 LCD 是带触摸屏的 LCD，通过 SPI 获取触摸信息。图 3.20 为 LCD 模块的接口引脚。

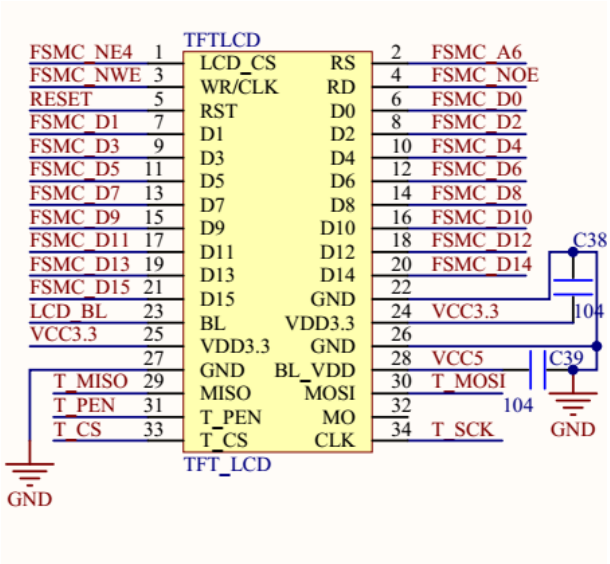


图 3.20 LCD 模块接口

表 3.2 LCD 显示模块和微控制器的连接

LCD 模块			STM32 微控制器	
引脚号	引脚名称	描述	连接名称	连接引脚
1	LCD_CS	LCD 使能信号	FSMC_NE4	PG12
2	RS	读写寄存器或数据选择	FSMC_A6	PF12
3	WR/CLK	写信号	FSMC_NWE	PD5
4	RD	读信号	FSMC_NOE	PD4
5	RST	复位信号	RESET	
6~21	DB1~DB17	并行数据线	FSMC_D0~D15	PD14~15 PD2~3 PE7~15 PD8~10
22\26\27	GND	地	GND	
23	BL	背光控制	LCD_BL	PB15
24\25	VDD3.3	电源	VCC3.3	
28	BL_VDD	背光电源	VCC5	
29	MISO	SPI 从到主机信号	T_MISO	PB2
30	MOSI	SPI 主到从机信号	T_MOSI	PF11
31	T_PEN	笔触信号	T_PEN	PB1
32	MO	未定义	NC	
33	T_CS	触摸屏使能信号	T_CS	PC13
34	CLK	SPI 同步时钟	T_SCK	PB0

对于 LCD 的背光控制，通过 PB15 引脚进行控制，可以起到将 LCD 打开与关闭的作用。

对于触摸，本系统采用电阻式触摸屏，触摸驱动芯片为 XPT2046，其是一个 SPI 三线接口的芯片，但其数据发送方式与标准 SPI 接口又有些许区别。所以我们采用软件 SPI 进行通信，故将 T_CS、T_SCK、T_MOSI、T_MISO、T_PEN 分别与 PC13、PB0、PF11、PB2 和 PB1 相连。其中 T_PEN 笔接触中断信号，当有笔接触触摸屏时，触摸芯片将之拉到低电平。

LCD 显示屏采用 NT35310 主控，该主控采用 16 位 8080 并口。使用 RGB565 方式传送数据。将 LCD_CS 引脚与 PG12 即 FSMC_NE4 相连，可控制 LCD 的使能；将 LCD_WR 与 PD5 即 FSMC_NWE 相连，可控制 LCD 的写入使能；将 LCD_RD 与 PD4 即 FSMC_NOE，可控制 LCD 的读取使能；将 LCD_RS 与 PF12 相连，通过地址的控制来选择是读写命令还是读写数据；DB0~DB15 位 16 位的数据线，与 FSMC_D0~FSMC_D15 相连，可以用来传输数据；同时还应将 RESET 与复位线相连，可以在复位微控制器的同时复位 LCD。

（5）摄像头接口电路设计

本系统采用摄像头对魔方信息进行采集，摄像头采用 OV2640。该摄像头像素为 1600*1200，输出支持 RGB565 模式，与 LCD 显示屏匹配。

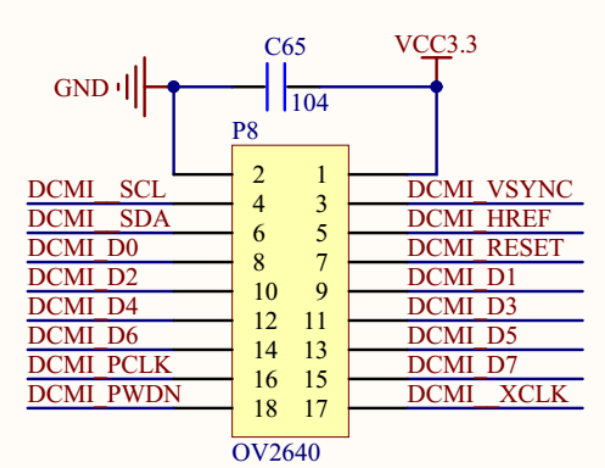


图 3.21 摄像头模块接口

其控制接口采用 SCCB 协议与微控制器通信，SCCB 是基于 I2C 方式的控制协议，专门针对摄像头设计的。DCMI_SCL 和 SCMI_SDA 与微控制器的 PD6 和 PD7 相连接。

对于数据传输，摄像头采用 8 位并行传输方式。通过像素中断、行中断和场中断来区分有效数据。其中 DCMI_PCLK、DCMI_VSYNC 和 DCMI_HREF 分别为像素中断、行中断和场中断，与微控制器的 PA6、PB7 和 PA4 相连；DCMI_RESET 为复位信号，与 PG15

相连；DCMI_PWDN 为掉电使能，与 PG9 相连；DCMI_D0~DCMI_D7 为数据传输引脚，与控制器对应引脚相连。DCMI_XCLK 没有使用。

表 3.3 摄像头模块与微控制器的连接

摄像头模块			STM32 微控制器	
引脚号	引脚名称	描述	连接名称	连接引脚
1	1	电源	VCC3.3	
2	2	地	GND	
3	3	场同步信号	DCMI_VSYNC	PB7
4	4	SCCB 时钟信号	DCMI_SCL	PD6
5	5	行同步信号	DCMI_HREF	PA4
6	6	SCCB 时钟信号	DCMI_SDA	PD7
7	7	复位信号	DCMI_RESET	PG15
8~15	8~15	并行数据线	DCMI_D0~D7	PC6~9 PC11 PB6 PE5~6
16	16	像素同步信号	DCMI_PCLK	PA6
17	17	未用	DCMI_XCLK	PA8
18	18	未用	DCMI_PWDN	PG9

第4章 系统软件设计

4.1 软件总体设计

为了满足模块化和分体化的设计需求，本系统为两个控制部分，分别为运动控制部分和解算控制部分。运动控制部分即将魔方旋转的指令（辛马斯特标记）转换为实际的机械操作，该部分采用串口接收和发送数据；解算控制部分涉及魔方各面颜色信息的采集，并通过采集到的信息计算出解决魔方的步骤，然后将之发送。

魔方还原硬件系统的总体流程图如图 4.1：

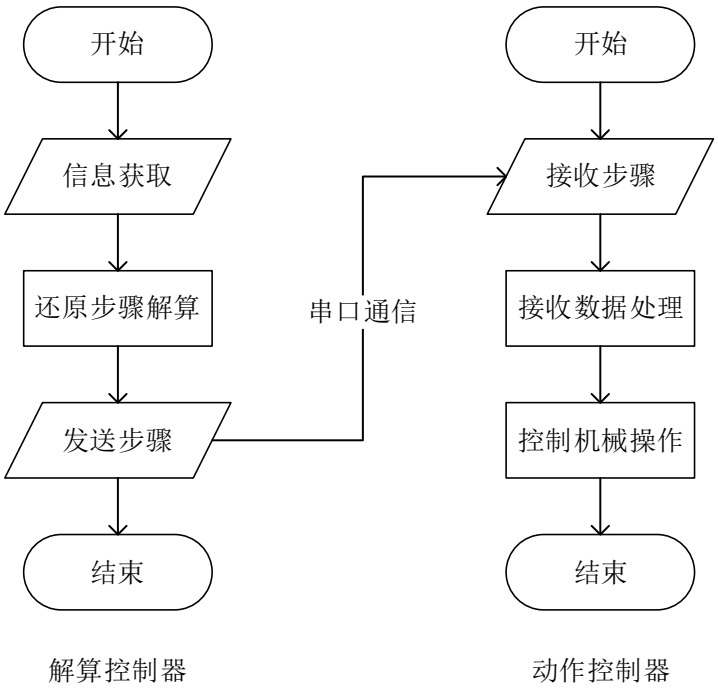


图 4.1 系统总体设计流程图

4.2 运动控制部分设计

（1）通信部分

由于本系统涉及到多个控制器之间的通信，不良的通信会造成系统的反应迟缓，甚至无法解析命令，导致还原的失败。选取一种合理的通信方式对于系统的稳定性和可靠性尤为重要。目前被广泛采用的通信方式包括有线通信方式和无线通信方式。有线通信又可以进一步划分为并行通信和串行通信，前者需要的引脚多，不利于系统的模块化，更适用于高速和大量数据的传递；后者有利于减少系统的复杂度，与并行通信方式相比更适用于本系统。由于微控制器没有内置无线通信模块，无线通信一般需要外部添加模块，本系统没有涉及一对多、多对一无线通信，无需进行无线组网通信。故采用 nRF24L01+可方便的进行无线通信。经测试，采用 nRF24L01+模块进行无线通信即使开启自动重发（ART）以及

自动应答（AA）经过一段时间的发送也会产生丢包现象。故最后采用通用同步和异步串行接收和发送器（USART）简称“串口”进行两控制器之间的通信。

波特率为每秒传输数据的位数，单位为“波特（band）”；比特率为每秒传输二进制的位数，单位为“比特每秒（bit/s）”。比特率=波特率×传输数据每一位的二进制位数。在串口通信中，传输数据的每一位的二进制位数为 1，比特率与波特率相等。动作控制器采用的为 5V 的电源供电，而解算控制器采用的是 3.3V 电源供电，如果将两者直接相连，除了添加限流电阻或电平转换电路外，被动的做法是将串口的波特率设置到 115K 以上，来保证 5V 的微控制器流入 3.3V 微控制器的电流不会过大，所以本设计将串口波特率定为 115200bps。对于 Arduino 来说，设置串口，仅需调用自带库中的“Serial.begin(115200);”函数，就可以将串口设置为 115200 波特率、8 数据位、1 停止位和无校验位模式下。

当运动控制器串口接收到数据时，程序就会跳转到中断服务程序中，程序从串口缓冲区中读取数据，并将之进行初步解析同时设置接收到数据的标志位，结束中断服务程序。串口中断服务程序中没有将数据完全解析的目的是考虑到解析的时间可能会影响串口的下次接收数据，尽快将数据处理完毕后可以准备接收下一个数据，可提高系统的响应速度和稳定性。

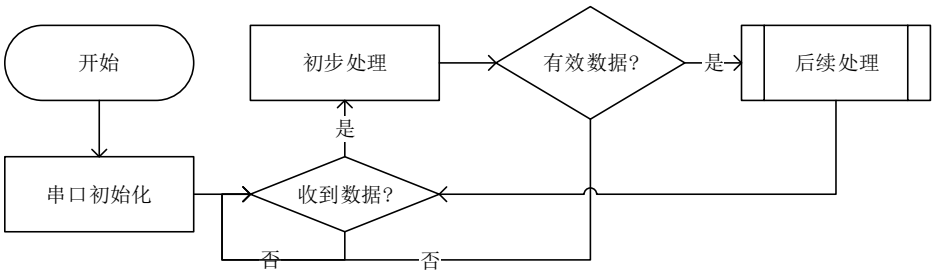


图 4.2 通信部分流程图

（2）控制部分

DRV8825 是一个性能优良的步进电机驱动器，对其的控制也极为简单：通过 DIR 引脚可以控制步进电机的正反转，STEP 引脚发送一个脉冲即可实现一步的运动。必须在 SLEEP 引脚置高电平 1.7ms 后，或者 ENBL 引脚置低电平 650ns 后，或者 DIR 引脚变化 650ns 后，发送脉冲信号。脉冲必须持续 1.9μs 后，步进电机驱动器才可以控制步进电机旋转 1 步。STEP 引脚上的波形频率反映在步进电机上即为电机的旋转速度。程序中通过设置时间间隔来控制电机的旋转速度。

图 4.3 为步进电机驱动芯片的时序图，表 4.1 为步进电机的时序图中的各个时间规定的说明。

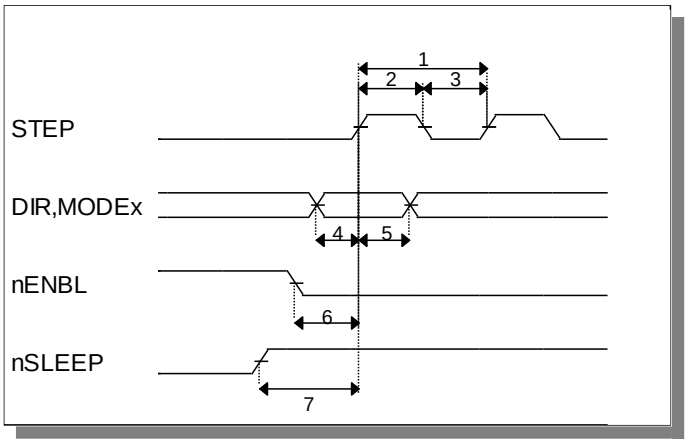


图 4.3 DRV8825 时序图

表 4.1 DRV8825 时序

序号	说明	最小值	最大值	单位
1	f_{step} 步进脉冲频率		250	kHz
2	$t_{WH(STEP)}$ 脉冲高电平持续时间	1.9		μs
3	$t_{WL(STEP)}$ 脉冲低电平持续时间	1.9		μs
4	$t_{SU(STEP)}$ 设置时间	650		ns
5	$t_{H(STEP)}$ 保持时间	650		ns
6	t_{ENBL} 使能时间	650		ns
7	t_{WAKE} 唤醒时长		1.7	ms

对于电机而言，缓加速和缓减速可以避免电机的损坏和瞬时过大的电流对电路的冲击，也可以给机械结构充分的准备时间。控制部分通过对发送脉冲时间间隔的逐渐缩短，而达到缓加速的目的；对发送脉冲时间间隔的逐渐加长，而达到缓减速的目的。电机停止后将EN 置高电平，停止驱动对电机的输出，减小待机电流的同时可以减少发热。

Atmega2560 中有 6 个定时器，定时器为自减并且自动重装型定时器。其中定时器 0 和定时器 2 与延时等函数有关，故修改定时器 0 或定时器 2 的参数可能造成程序时序的错误。本设计中使用定时器 1 作为中断源，当定时器 1 中的值自减为零时触发定时器中断，进入中断服务程序。中断服务程序中处理串口初步处理的步骤信息，将第一个字符进行选择，使对应电机进行转动；第二个字符如果为数字“2”则旋转相应的面 180°；如果最后一个字符为“’”则为逆时针旋转，没有表示为顺时针旋转。例如串口接收到“F2’”则将魔方前面的步进电机逆时针旋转 180°，从而可以带动魔方的前面旋转到相应的位置。

如果接收到的是旋转顶面的命令，由于四个爪子只可以旋转前后左右四个面，无法直接旋转魔方顶面。通过控制左右两个爪子松开，前后两个爪子同时逆时针旋转，将魔方整

体逆时针旋转 90°，然后左右两个爪子夹紧，这样旋转左右两个爪子就可以相对的旋转魔方的顶面和底面；当前状态下需要旋转魔方的左右两个面时，将上述过程的逆过程进行一遍，就可以恢复魔方的默认姿态，从而旋转魔方的左面或右面。通过上述过程，可以使用 4 个面控制步进电机对魔方 6 个面进行旋转，克服了传统机械结构只能将魔方改装将魔方与电机轴固定才可以旋转六个面的缺点。

图 4.4 动作控制流程图

4.3 解算控制部分设计

（1）魔方信息的采集

为了实现魔方各面信息的方便和快捷录入，本系统采用摄像头采集，按键或触摸屏辅助修正魔方颜色信息的方法。由于摄像头受光照和环境影响大，一次将 64（=8*6）个块的

颜色信息全部采集正确的几率不大。加入了修正录入数据的功能，即弥补了手工输入耗时长的缺点，又可以实现颜色的快速录入。

对于摄像头的采集，采用 OV2640 的 RGB565 输出格式，每一个像素为 16 位数据，接口宽度为 8 位，因此每两个字节可以取得一个像素的数据。VSYNC 为摄像头的帧同步信号，该信号由高电平变为低电平（下降沿）表示一帧已经准备好发送；HREF 为行同步信号，该信号由低电平变为高电平（上升沿）即为一行准备发送；PCLK 为像素同步信号，当由低电平变为高电平时（上升沿）读取到的值就为有效像素。当 VSYNC 产生下降沿之后，等待 HREF 产生上升沿之后再在 PCLK 的上升沿读取数据接口的信息，即为所需要的 8 位像素值，如图 4.5。

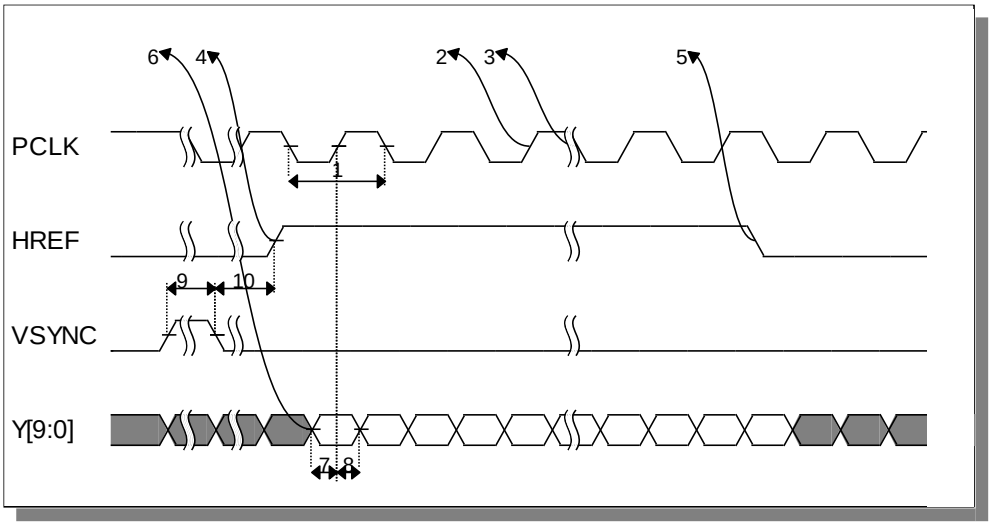


图 4.5 OV2640 输出时序图

表 4.2 为时序图中各个时间的说明，可以看到，摄像头在像素同步信号 5ns 后输出像素信息，场同步信号高电平持续时间和场同步信号下将到行同步信号上升时间这两段时间为 OV2640 的内部处理时长，没有产生有效数据，所占据的时间为 968994.18（=（4*1922+27193）*27.78）ns。由于摄像头采用 UXGA 分辨率输出，所以期间共输出了 1920000（=1600*1200）个像素信号，但由于显示屏为 480*320 分辨率，故只使用了其中的一部分。

STM32F407 微控制器自带 DCMI（数字摄像头）接口，是一个同步并行接口，用来接收外部 8 位、10 位、12 位或 14 位 CMOS 摄像头模块发送的数据，其可以自发的处理摄像头产生的场、行和像素中断，从而将摄像头数据读取到 DCMI 的寄存器中。为了实现通过三个同步信号接收摄像头数据，需要将 DCMI 配置为 8 位数据格式、HSYNC 高电平时、VSYNC 高电平是和 PCLK 上升沿时获取数据。

表 4.2 OV2640 输出时序图说明

序号	说明	最小值	典型值	最大值	单位
1	t_p PCLK 周期		27.78		ns
2	t_{pr} 像素同步信号上升时间		3.5		ns
3	t_{pf} 像素同步信号下降时间		2.2		ns
4	t_{dphr} 行同步信号上升时间	0		5	ns
5	t_{dphf} 行同步信号下降时间	0		5	ns
6	t_{dpd} 像素同步信号低电平到数据输出延时	0		5	ns
7	t_{su} 数据总线设置时间	15			ns
8	t_{hd} 数据总线保持时间	8			ns
9	场同步信号高电平持续时间		4x1922		t_p
10	场同步信号下降到行同步信号上升时间		27193		t_p

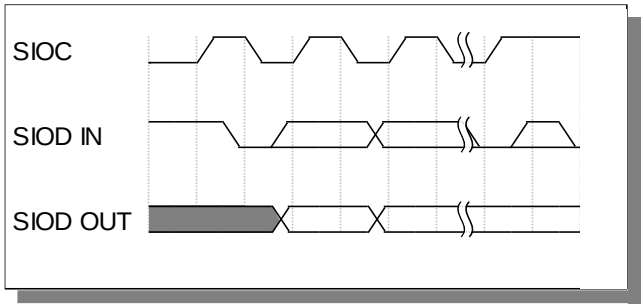


图 4.6 SCCB 接口读写时序

摄像头的内部寄存器的配置采用 SCCB 接口进行配置，SCCB 是串行摄像机控制总线（Serial Camera Control Bus）的简称，是和 I2C 类似的一种协议，也是由一条时钟线 SCL 和一条数据线 SDA 构成。由于不需要进行多次、高速的配置，SCCB 接口采用软件模拟进行。

图 4.7 为摄像头读取单个像素有效数据的流程图，对于其他像素数据的读取，以此类推。

通过将魔方放置到固定的位置，将魔方各个块的大致范围限定。摄像头采集完毕后，统计每个块中每个像素的颜色信息，并将这个块中所有像素的颜色信息进行统计，将这个块最可能的颜色作为这个块的颜色，以此类推确定 48 个块的颜色。

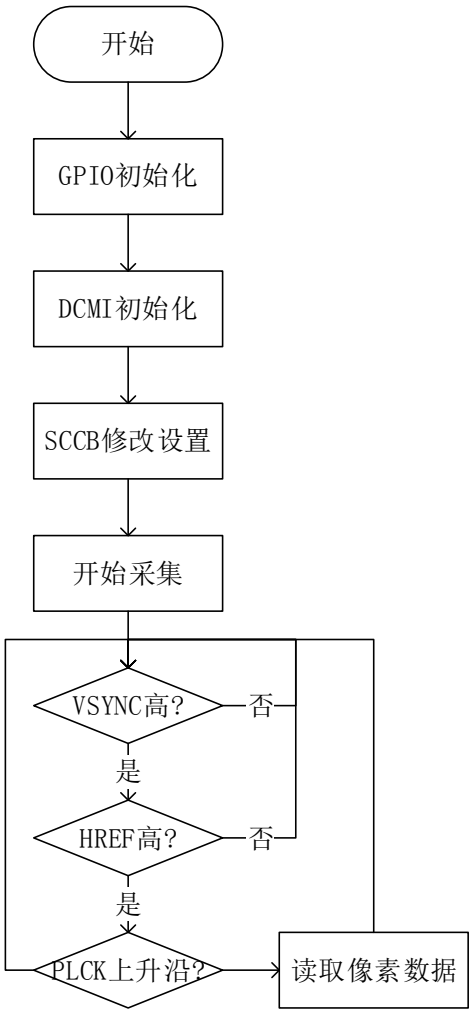


图 4.7 摄像头读取像素数据流程图

（2）显示部分

为了直观的体现魔方各面的颜色和魔方的还原过程，本设计采用 LCD 显示屏显示各种信息。

对于 LCD 显示屏，其接口采用 8080 并口。8080 并口是一种通用接口，与 SPI 或 I2C 类似，都有自己的协议。8080 并口是一种并行接口，对于 LCD 来说，有 16 条数据线，一次可以传输 16 位数据，如果微控制器要从设备中读取数据，将 CS 即片选信号置低电平，WR 即写使能信号置高电平，RD 即读使能先置低电平，然后在 RD 的上升沿读取引脚上的数据，即为有效数据；如果微控制器要向设备中写入数据，将 CS 置低电平，RD 置低电平，WR 先置为低电平，然后在 WR 的上升沿，就会将引脚上的数据写入设备，如图 4.8 为读写单个字节的时序图。LCD 还有一个 RS 引脚，该引脚控制着是要读写数据还是要读写命令，对 LCD 进行完整的读写操作涉及到许多字节，其中首先应发送的为命令字节，然后 LCD 根据微控制器发送的命令字节，来做出相应的读写多个字节数据的操作，如图 4.9 和图 4.10。

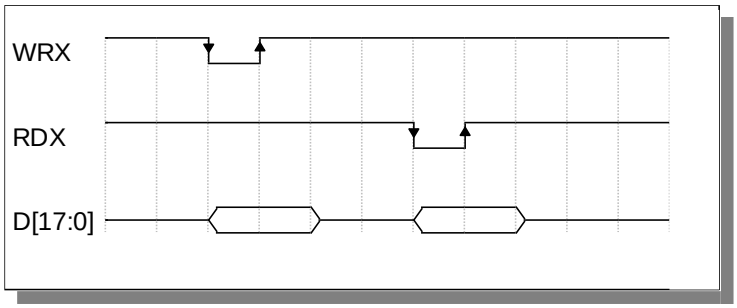


图 4.8 写、读单个字节时序图

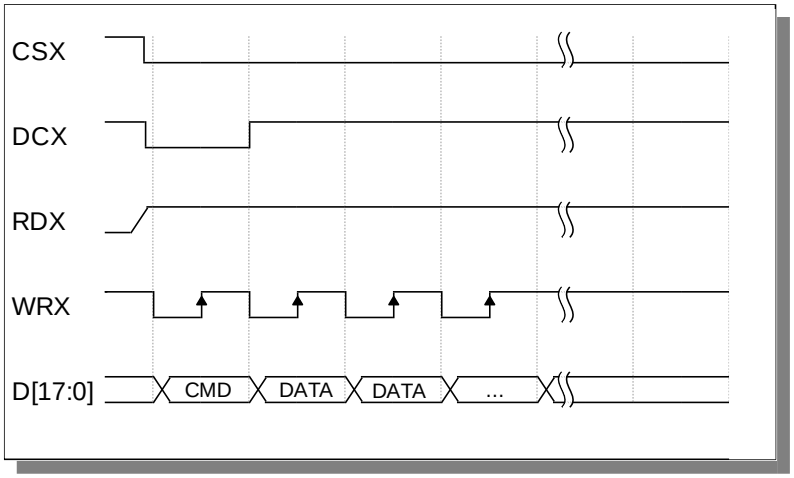


图 4.9 写操作时序图

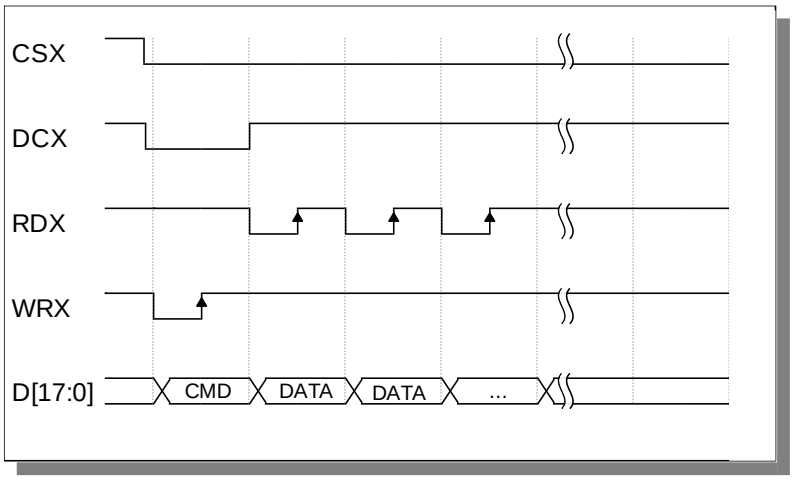


图 4.10 读操作时序图

STM32F407 带有一个 FSMC（灵活的静态存储控制器），该控制器可以用来读写外部 NOR/PSRAM 部件或者 NAND/PC 卡部件。由于上述部件都是由地址线、数据线、读使能、写使能和片选信号构成的，而 LCD 的时序与这些存储部件有相似之处，故可以将 FSMC 用来控制 LCD，大大的节省 CPU 的资源。将 LCD_CS 与 FSMC_NE4 相连，即为存储块 1 的第 4 个区域的片选信号，该区的地址范围为 0x6C00 0000~0x6FFF FFFF。LCD_RS 与 FSMC_A6 相连。在 STM32 内部，由于需要对 8 位设备进行兼容，所以最后一位地址对于

16 位外设并不选用，因此最后一位默认为 0，对应到地址引脚时默认为内部地址右移一位对应，即内部地址第 1 位到第 7 位对应外部地址引脚第 0 位到第 6 位。则 LCD 的基址为 0x6C00 007E，对应到地址引脚第六位为 0，即 FSMC_A6 为低电平，对应到 LCD 为 LCD_RS 为低电平，即读写命令；当对下一个地址进行操作时，STM32 默认将地址右移一位进行增加，经过计算为 0x6C00 0080，经过上述对应可以得到地址线第 6 位变为 1，对应到 LCD 为 LCD_RS 为高电平，即读写数据。通过上述操作可以实现将对 LCD 的命令和数据操作映射为对其所对应地址中的数据的操作。

图 4.11 为 LCD 显示屏读写操作的流程图。通过微控制器内部的 FSMC 接口，LCD 的内置 RAM 就被映射成为一段地址，对这段地址的操作就为对 LCD 的 RAM 的操作。

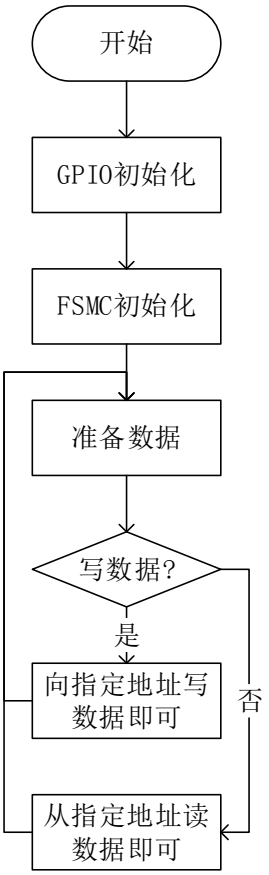


图 4.11 LCD 显示屏读写数据流程图

（3）触摸控制部分

前文已经介绍了魔方信息的输入过程，但通过摄像头输入的魔方信息并不一定准确，外界灯光和环境的干扰很大。当摄像头采集到的颜色信息与实际有不同时，本设计采用按键或者触摸控制，更改已经录入好的颜色，来为后续的完成作必要的完善。

按键输入采用扫描方式读取引脚的电平信息，本文就不再以赘述。

电阻式触摸屏利用压力感应来进行位置感测，当屏幕被按下时，屏幕的电阻发生变化。对于电阻屏的触摸输入，本系统采用 XPT2046 芯片作为驱动芯片，该芯片内置 AD 转换器，可以读取按下时 X 和 Y 不同的电阻值，并将之解算成为 X 和 Y 坐标的数字量。

该芯片采用 SPI 协议通信，CS 为芯片的使能信号，低电平为选中该芯片；DCLK 为时钟信号，在上升沿或者高电平时芯片获取 DIN 上的数据或者为微控制器读取 DOUT 上的数据；BUSY 信号为低电平表示芯片正在执行转换或者输出。同时芯片还有一个 PENIRQ 引脚，当屏幕上有触摸动作时，PENIRQ 就会产生一个下降沿，微控制器可以以此下降沿获取按下信息。

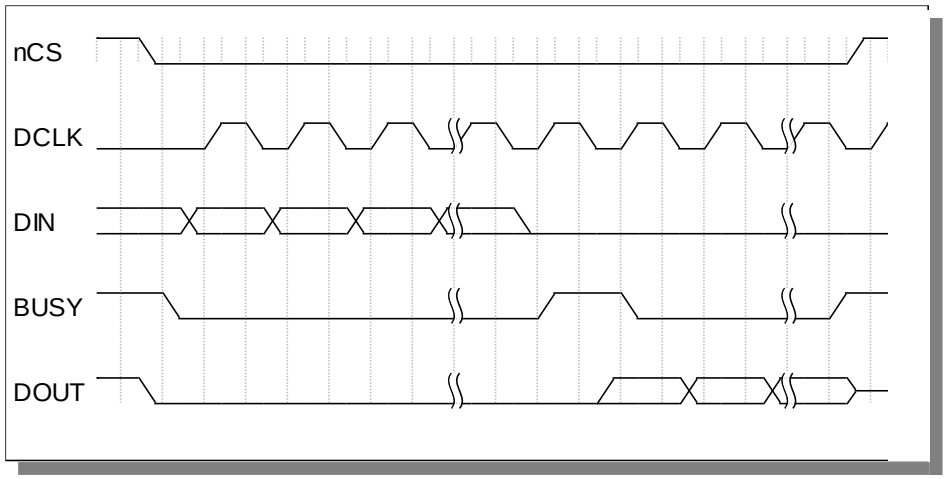


图 4.12 XPT2046 写、读数据时序

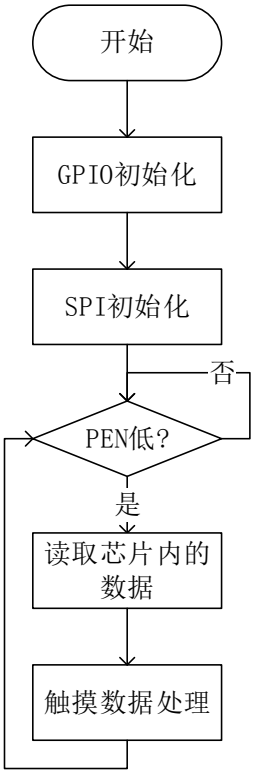


图 4.13 获取触摸数据流程图

图 4.13 为读取触摸数据的流程图，通过判断 PEN 引脚上的电平来判断是否有笔操作，当 PEN 为电平时通过 SPI 接口读取芯片内的数据，就为触摸坐标。

（4）解算魔方算法

获取到正确的魔方各面的颜色信息后，系统进入魔方解算过程。魔方的还原采用降群法还原方法。

将魔方的黄色作为底面，红色面作为前面，魔方的其他面的位置也就唯一确定下来了。

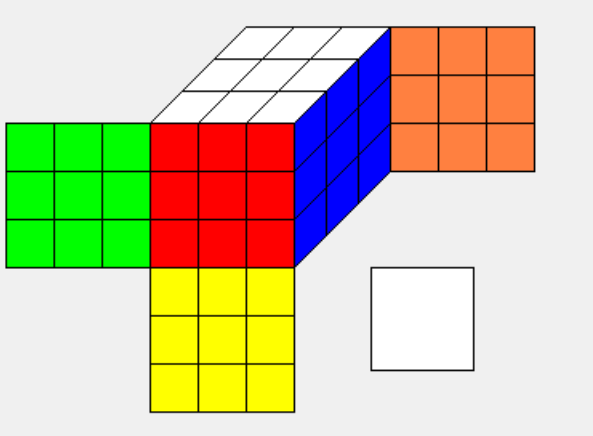


图 4.14 标准的魔方摆放位置

首先，把魔方的表示方法加以规范。在一般的理解中，魔方是以面为基础的，每一面除了中心块以外有 8 个块，每一个块都可以是 6 种颜色中的任意一种。为了表示每个面每个块的颜色，需要有一个二维的数组，记录着每个块的颜色信息。这样做虽然直观，但不方便处理数据，并且不方便对各种算法进行解析。本设计采用一种特殊的记录方式，此方式由 Mike Reid 发明，采用一个前后有固定顺序的序列，对每个位置进行记录[15]。由于魔方除了中心块以外仅由棱和角组成，还原状态下的魔方从棱开始，从顶面的前方的棱开始，这个位置的上面记为第一个“U”，前面记为“F”，将这两个字母记为一组“UF”，既为可以表示这个棱的位置信息；对于顶面右侧的棱，上面的记为“U”，右面的记为“R”，两个合为一组即为“UR”。以此类推将所有的记录下来，组成一组字符串，为：

UF UR UB UL DF DR DB DL FR FL BR BL UFR URB UBL ULF DRF DFL DLB DBR

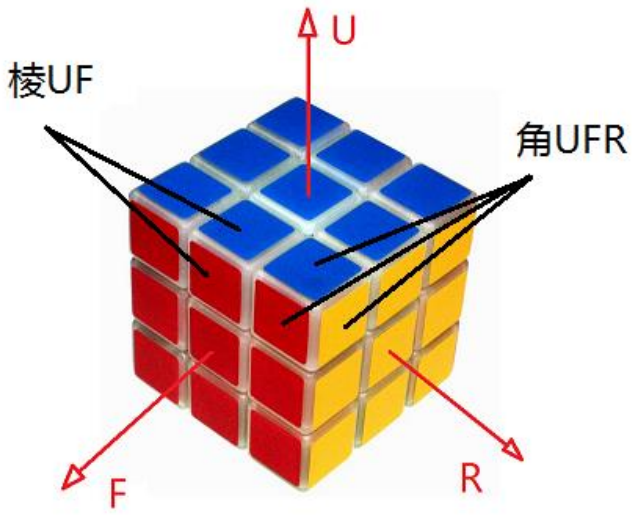


图 4.15 魔方的棱和角

上面的字符串就为还原状态下的魔方的全部棱、角块信息。注意如果将“UF”与“UR”的位置进行互换，或者将“UF”字母的顺序相互变更，表示的都不是一个还原了的魔方。

例如，将还原好的魔方进行如下旋转操作：

L+ D+ L2 R2 U2 F- R- D+ F2 U2 B- R2 D- U- L+ B2 R+ D2 F- D2 R+ D- F+ U+ L+

和如下的表示方法等价，都为将打乱为图 4.16 所示的魔方。

DL DR FL BU BR UL DB FU RU FR LB DF ULF RBU BDL DFL DBR RFD UFR LUB

我们从第一个“DL”开始看起，表示的原先位于下面和左面的棱现在位于上面和前面。且原先位于下面的白色现在在上面。其他的可以类推。这样表示有两个优点，一是可以方便的用一串字符表示整个魔方，另一个是表示的方法唯一，即使魔方各面的颜色变化，但相对位置不变，最后的表达式不变。

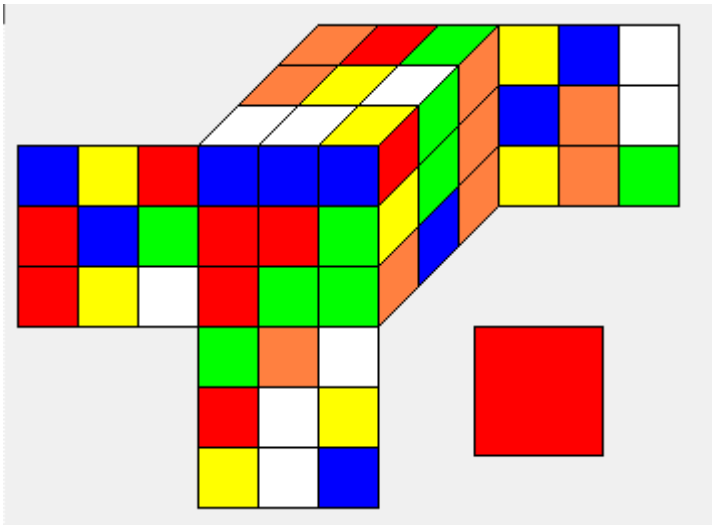


图 4.16 打乱的魔方

接下来将准备降群法还原魔方方式。降群法将魔方的还原分为几个步骤，目标是将打乱的棱、角块恢复到相应的位置上。由于魔方的总的变化数约为 4.33x10^19，找到当前变化的状态并且找到相应的旋转步骤就可以将魔方还原。但是将这么多的变化全部存入微控制器既不现实，也不符合程序精简的原则，更受限于微处理器的速度不可能在短时间内查找到相应的变化状态。降群法即提供了一种方法将打乱的魔方分为几种不同混乱程度的状态，每一种状态称之为一种“魔方群”，并将各个状态的变化操作记录到微控制器的内存中，通过由复杂到简单的魔方群进行降群复原。该方法将复杂的所有打乱情况进行了精简分类，可以大大节约所占用的存储空间，满足微控制器运算的需求。同时本方法因为可以找到较少的还原步骤，被广泛采用于各种还原系统中。

对于魔方群的分类，降群法将魔方分为 5 个群，群名称分别为“G0”、“G1”、“G2”、“G3”和“G4”。其中“G0”为可以通过上、下、左、右、前或后转一次这个基本动作打乱的魔方群；“G1”为可以通过上、下、左或右转一次或者前或后转两次这个基本动作打乱的魔方群；“G2”为可以通过上或下转一次或者左、右、前或后转两次这个基本动作打乱的魔方群；“G3”为可以通过上、下、左、右、前或后转两次这个基本动作打乱的魔方群；G4 为魔方已经完全还原的状态。

关于魔方的还原方法，不在本文的重点讨论范围内，这里就不在赘述。一般微控制器采用的都是占用资源较少的层先法或 CFOP 法，这些方法公式较少，便于记忆，因此在人类还原魔方时常采用这些方法。但公式少意味着重复率高，必须经过数百步才可还原，大大降低了还原速度。本设计将用于计算机的降群法移植到 STM32 微控制器上，利用微控制器进行运算，可以大大提高整套系统的精简程度和降低成本。

（5）通信部分

将魔方的步骤解算出来后，通过串口向动作控制器发送。为了使解算控制器和动作控制器的进度匹配，解算控制器会等待动作控制器完成相应动作并返回完成信号后才发送下一个控制信号。

STM32F407 有 4 个 USART 和 2 个 UART，其中 USART 除了与 UART 都需要 TX 和 RX 信号以外，还需要 CK 同步时钟信号，使用同步模式进行收发数据。同步模式有主从机之分，主机发出时钟信号，从机接收时钟信号，用来获取波特率信息。串口一般采用异步模式，即 UART。将微控制器波特率设置为 115200，字长为 8 位，停止位为 1 位和无奇偶校验位模式，以满足动作控制器的接收需求。

第5章 调试与测试

5.1 还原步数测试

不同的解算算法的步骤不尽相同。针对不同的打乱状态，使用不同的解算算法的步骤均不相同。本测试旨在测试几种不同打乱情况下的不同算法的还原步骤。

表 5.1 还原步数测试

序号	层先法	降群法	二阶段法
1	105	20	10
2	120	34	22
3	118	30	18

由此可得，二阶段法的还原步骤最少。但由于需要更多的空间存储表格，微控制器不能将表格完全存储，故选用降群法比较好。

5.2 动作时间测试

对于魔方还原硬件系统而言，每个动作的执行时间不相同。由于旋转 90°需要让爪子恢复的默认的状态，动作时间比旋转 180°的时间长。旋转顶面的动作需要变换魔方的姿态，时间会进一步加长。动作“F’”是“F”的逆运动，理论上花费时间相同，故不再测试。

表 5.2 动作执行时间测试

动作	时间	单位
F	1030	ms
F2	221	ms
B	1000	ms
B2	220	ms
L	1050	ms
L2	218	ms
R	1020	ms
R2	217	ms
U	2061	ms
U2	2001	ms
D	2530	ms
D2	2110	ms

可以看出，对于旋转 180° 可以比旋转 90° 减少一半以上的动作时间，在选取魔方算法时，应尽可能的选取各面旋转 180° 多的方法，来使系统的动作达到最快。

5.3 系统稳定性测试

魔方还原的每一步必须精准无误，否则会影响后面的动作，最终导致无法还原。通过进行长时间，多次运动来测试系统的稳定性。

表 5.3 系统稳定性测试

序号	执行时间	是否还原	是否出错
1	3’	是	否
2	1’30”	是	否
3	1’	是	否

通过多次的系统测试，本系统性能优良，可以满足高速多次状态下的还原需求。

5.4 系统展示

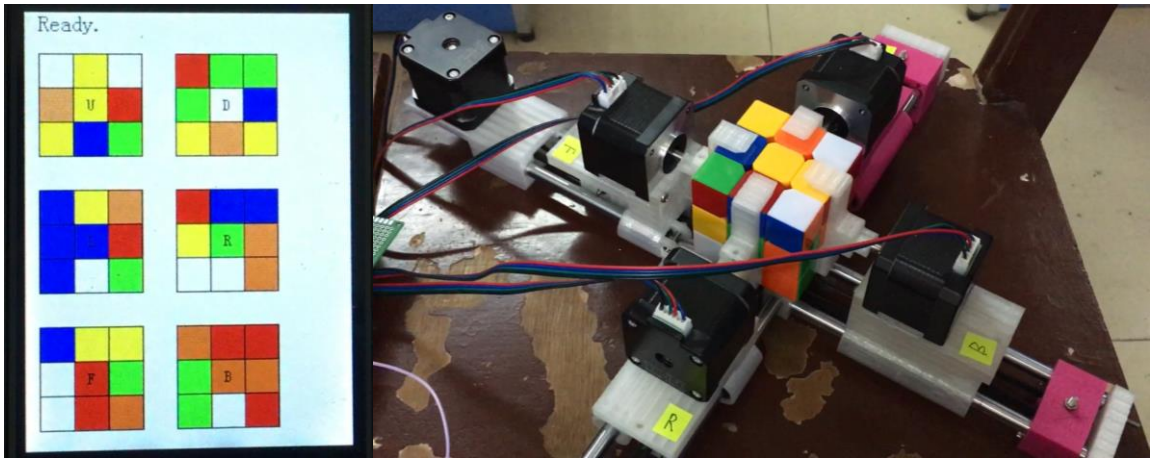


图 5.1 开始还原

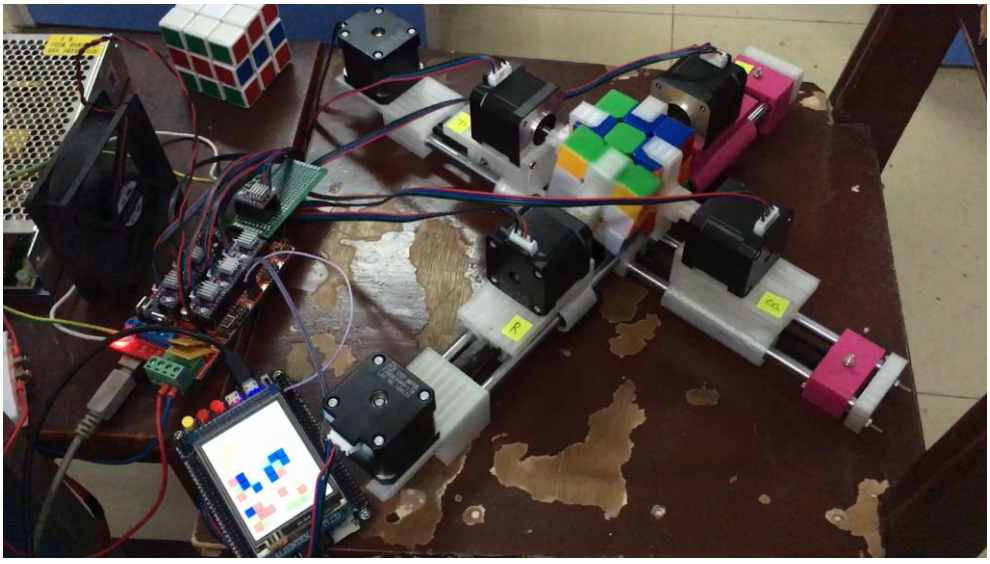


图 5.2 还原过程中

图 5.1 为采集完魔方各面信息后，经过触摸屏和按键修正，得到的最终魔方信息。可以看到，显示屏上的魔方各面颜色信息与实际情况相同。在还原过程中，显示屏上同步显示当前状态和步骤，如图 5.2。

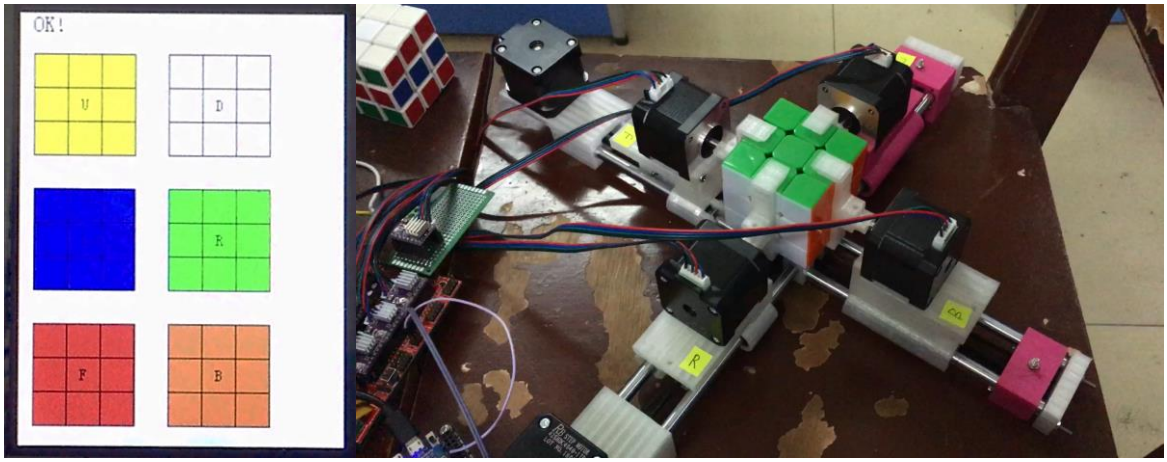


图 5.3 还原完成

总　　结

基于嵌入式微控制器的魔方还原硬件系统，是一套完整的从输入魔方信息、解算魔方还原步骤到实际的机械结构对魔方进行操作的完整系统。该系统具有结构精巧，装配简单和模块化的优点，能完整的展现魔方还原的每个步骤，并且可以实时的动态显示。该系统可用于展示、教育等场景，还原成功率高。

本文从机械结构设计、硬件结构设计和软件设计三个方面详细讲解了整个系统的实施方案。其中硬件设计为本文的重点，通过几种方案的相互比较，选取了一种满足系统需求的方案并将之加以论述。对于电路的设计，本文从微控制器的最小系统开始，由小到大，由简单到复杂的将整个设计加以阐述。

对于机械结构设计，本文介绍了一种可以在不破坏魔方结构的情况下对魔方各个面进行旋转的机械结构，通过 6 个步进电机就可以对魔方各面进行操作。该结构简单，操作速度快，且不易发生故障，能够适应展示和教学的应用环境。

软件设计部分，对步进电机驱动 DRV8825、摄像头 OV2640、LCD 显示屏和触摸屏的时序以及原理进行了分析，对摄像头检测颜色算法、机械控制算法和结算魔方算法都有分析和研究。

本设计从理论到实践，最终制作出了一套可以将魔方在一分钟左右还原的硬件系统，该系统遵从模块化的设计，在方便维护的情况下还可将之进行拓展，方便以后的算法研究等更多的实验和研究。

参考文献

[1]郑瑜.魔方原理及其应用[D].浙江大学,2009.

[2]李海龙.基于 LEGO 平台的智能解魔方机器人系统研究与实现[D].东北大学,2015.

[3]Rokicki T. Twenty-Two Moves Suffice for Rubik's Cube [J]. Mathematical Intelligencer, 2010, 32(1):33-40.

[4]卢昌海.魔方与"上帝之数"[J].金融博览,2014(11):26-27.

[5]董海阳,魏巍.类人四轴解魔方机器人的设计[J].电子技术与软件工程,2013(08):62.

[6]李国军,钟志强,张毅宁.LEGO 机器人魔方还原基本方法与过程[J].鞍山师范学院学报,2014,16(02):71-74.

[7]维基百科编者.直流电动机[G/OL].维基百科,2018(20180519)[2018-05-19].https://zh.wikipedia.org/w/index.php?title=%E7%9B%B4%E6%B5%81%E7%94%B5%E5%8A%A8%E6%9C%BA&oldid=49621688.

[8]维基百科编者.步进马达[G/OL].维基百科,2018(20180606)[2018-06-06].https://zh.wikipedia.org/w/index.php?title=%E6%AD%A5%E9%80%B2%E9%A6%AC%E9%81%94&oldid=49872263.

[9]维基百科编者.伺服马达[G/OL].维基百科,2018(20180201)[2018-02-01].https://zh.wikipedia.org/w/index.php?title=%E4%BC%BA%E6%9C%8D%E9%A6%AC%E9%81%94&oldid=48111424.

[10]郑振杰,江衍煊,游德智.单片机结合 ULN2003 驱动步进电机的应用[J].电机技术,2010,54(9):77-78.

[11]郭力峰,揭宗昌,蔡泽辉.基于 XC886 单片机的魔方机器人设计[J].电脑知识与技术,2011,24(8):25-28.

[12]蓝杰,张浩然.基于 STM32 的微型步进电机驱动控制器设计[J].微型机与应用,2015,34(01):43-46.

[13]吴迪.基于 DRV8825 的打印机电机驱动电路设计[J].电子技术与软件工程,2013(11):62.

[14]李栋.基于 ARM 嵌入式的图像处理研究[D].石家庄:河北科技大学,2010.

[15]梁小龙.解魔方算法的研究和系统实现[D].东北大学,2013.

致　　谢

感谢 Thingiverse3D 模型开源分享网站上的 Jaime Ferrís 提供硬件结构的参考。和 Jaap Scherphuis 的降群还原魔方算法。在论文的写作过程中，邹连英老师对于论文题目的选择、写作思路和行文结构，也提出了许多宝贵的意见，同时对我进行了悉心的指导。我的同学也对我进行了指点和帮助。本文借鉴了诸多学者的观点和资料，已列入参考文献。在此对他们表达由衷的感谢！