A Method based on the Embedded Control for the Third Order Rubik's Cube Robot

Qun Yin[1], Jianbo Zhang[2] (corresponding author), Gu Ji[1], Meisu Yin[1], Yunsheng Xu[1]

[1]Kunming University of Science and Technology, Kunming, Yunnan, China;
[2]Oxbridge College, Kunming University of Science and Technology, Kunming, Yunnan, China

Abstract

This paper mainly completes the hardware platform and software design of the three-step cube. The robot uses colour recognition, steering machine precise control and other scientific techniques. The hardware adopts STC90C516RD+ series as the main chip, and its peripheral modules include power supply and reset circuit module, 5461BS digital tube display module, TCS3200 colour sensor module, etc. Colour sensor module realized the recognition of Rubik's cube each colour piece, digital tube part of real-time display the corresponding colour of RGB values, single-chip computer analysis of the actual situation of Rubik's cube colours the corresponding reduction method, and finally by the steering gear to turn the Rubik's cube, complete the reduction process.

Keywords: Rubik's cube, STC90C516RD + series, TCS3200 colour sensor module, colour recognition sensor, steering gear control

Received: 17 June 2019

To cite this article:
Qun Yin, Jianbo Zhang, Gu Ji, Meisu Yin, Yunsheng Xu, „A method based on the embedded control for the third order Rubik's cube robot", in *Electrotehnica, Electronica, Automatica (EEA)*, 2019, vol. 68, no. 2, pp. 104-111, ISSN 1582-5175.

1. Introduction

Rubik's cube and magic square is an extreme sport of the hand. Figure 1 is the third order cube. The third order cube is composed of a central axis connecting 6 central blocks, 8 corner blocks and 12 edge blocks.

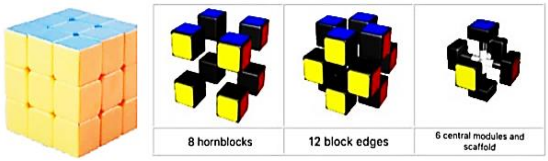

Figure 1. The third order cube

When they are joined together they form a whole, and either side can rotate horizontally without affecting the other squares.

The third order Rubik's cube robot is an intelligent competitive robot, which is characterized by strong scientific nature, rich creativity and distinct innovation points, and has good economic and social applications and education value. The Rubik's cube uses technology such as colour recognition and accurate control of steering gear. To solve the Rubik's cube robot, the control system is just like human's nerve centre, it plays the role of accepting algorithm and mechanical structure [1].

At present, the existing Rubik's cube robot still has many improvements. For example, the algorithm program is not concise enough, resulting in a long time of robot action process; the robot structure is not flexible enough, resulting in lower efficiency. The innovations of the Rubik's cube reduction robot system designed in this paper are as follows [2]:

(1) The use of toy materials for the robot bracket is conducive to the construction of the children and can better cultivate the user's hands-on ability [4].

(2) The design of the robot bracket use various simple connection structures and transmission structures to facilitate the design and debugging of users, which is more convenient and user-friendly in terms of use and can better cultivate users' logical ability.

(3) In addition to completing the reduction of Rubik's cube, the Rubik's cube reduction robot can also carry out physical teaching of the reduction steps. Compared with the reduction teaching of software, it can make learners more intuitive and profound to learn and understand the Rubik's cube, and improve learners' logical thinking ability faster [6].

This paper designs a control system based on embedded third order Rubik's cube robot. It recognizes each colour block of the Rubik's cube through the colour sensor module, and the digital tube part displays the RGB value of corresponding colour in real time. The single-chip computer analyses the actual situation of the Rubik's cube colour, and finally turns the Rubik's cube by the steering gear to complete the reduction process [7].

2. Technical introduction of the third order Rubik's cube

2.1.Introduction to the third order cube

Rubik's cube or Rubik's cube and magic square is an extreme sport of the hand. Usually, it refers to the third order Rubik's cube. The third order cube is usually a cube, made of elastic hard plastic. The game is to break up the cube and recover in the shortest time. The current official world record for the third-order cube is 4.73 seconds held by Feliks Zemdegs (Felix Zennengarth) of Australia [6].

Third-order Rubik's cube changes the overall number is 43252003274489856000. Or is approximately equal to 4.3 X 10 ^ 19. Formula of total variation of N order cube is shown in formula 1 and formula 2:

$$N=\frac{8!\times 3^{7}\times 12!\times 2^{10}\times (24!)^{\frac{(n-3)(n+1)}{4}}}{24^{\frac{3\left[(n-2)^{2}-1\right]}{2}}},n=2k+1,k\geq 1,k\in Z \tag{1}$$

$$N=\frac{7!\times 3^{6}\times (24!)^{\frac{n(n-2)}{4}}}{24^{\frac{3(n-2)^{2}}{2}}},n=2k,k\geq 1,k\in Z \tag{2}$$

And the reason of its odd and even is that even Numbers have no centre, and that centre includes centre edges and centre of surface, and if you're a Rubik's cube player, you know that. In fact, it is not difficult to derive the total change of n order as long as you master the derivation method of order 3, 4, 5. The key point is the heart block: the heart block of each surface is divided into several equivalent positions, and each group contains four heart blocks. Edge pieces, divided into edge and centre edge [3].

When they are joined together they form a whole, and either side can rotate horizontally without affecting the other squares.

2.2. Third order cube reduction method

There are many methods to restore the third order Rubik's cube: layer first, Angle first, edge first, bridge method, CFOP (Four steps to restore the cube are: bottom cross -> align the first two layers at the same time -> adjust the direction of the last layer -> adjust the order of the last layer), CFOOP (Five step magic cube reduction: bottom cross - > align the first two layers at the same time - > top cross - > top colour unification - > finish the top layer), smile tiger method and so on. So, what I'm going to show you here is the layering method, the layering reduction method.

The formula for a Rubik's cube consists of an English letter, a number, and single quotation marks ['], each of which represents a movement. It's a series of rotations, it's a formula. Letters include R(Right), L(Left), U(Up), D(Down), F(Front), and B(Back). Capital letters without ['] said in the clockwise direction of the turn 90 °, the capital letters with ['] said in the counter clockwise direction of the turn 90°.

After understanding the usage of the formula, the first step of the layer-first method is to turn out a white cross, whose side is the same colour as the adjacent central block, namely red to red, blue to blue, orange to orange and green to green. Find the edge blocks with white faces (white red, white blue, white orange, white green). Ignore the edge blocks without white faces and all corner blocks. According to the different positions and directions of the white edge blocks. The white surface is adjacent to the yellow centre block.

After rotating according to the formula symbol, the edge block reaches the translucent position.

Rotation at the top, make the white edge profile and move to the same colour centre piece above, and then turn 180° complete one side of the white cross. Repeat until all four corners are restored.

The second step of the layer-first method is to restore 4 white corner blocks, after which the white surface is completed, and 4 t-shapes are formed on the side. First look for the white Angle on the top floor. Turn the top floor, and the white Angle reaches the corresponding centre block. According to the position of the white surface (the direction of the Angle block), select the corresponding formula to restore the Angle block. Repeat until all four lower corner blocks are restored [17].

If the top layer is all yellow corner blocks without white corner blocks, the green orange and white corner blocks are replaced on the top layer and then restored according to the second step.

The third step of the layer-first method is to restore 4 intermediate edges (medium edges), without white and yellow surfaces. First, look for the non-yellow edge blocks on the top floor and move them to the centre block with the same colour on the side. According to the position of the edge block, select the corresponding formula to restore, and repeat the steps until all 4 middle edges are restored.

If the top level has no suitable edge, do any of the above formula once. The brackets in the formula represent a set of coherent manipulations and grouping auxiliary memories, and the existence of the brackets does not affect the rotation of the formula. The green and orange blocks are replaced on the top layer, and then restored according to step 3.

The fourth step of the layer-first method is to restore the colour direction of four top edges, and then form a yellow cross of the top surface. This step only needs to look at the top edge of the yellow, and do not pay attention to the top edge of the side and Angle. The fifth step of the layer-first method is to restore the four top Angle blocks and complete the yellow top.

In the fourth step and the third step, it is confirmed that the first two layers have been restored, but there is something beyond the diagram listed. For example, there are 8 yellow faces on the top surface. That is, the Rubik's cube is disassembled and reassembled after it disassembles. The sixth step of the layer-first method is to restore four top corner blocks. The final step of the layering method is to recover the position of four top edges. In the top view, where the small arrow indicates the movement direction of the edge block, and the small black bar represents the colour block with the same colour on the side.

3. System design and mechanical structure

3.1. Mechanical design

The mechanical structure of the robot is the most basic carrier to realize the reduction function of the Rubik's cube. At present, the mechanical structure of Rubik's cube robot is mainly divided into two types: human-like two-arm type and four-axis rotary type. By contrast, the four-axis rotating Rubik's cube robot has more advantages in mechanical structure stability, and the four mechanical arms can be controlled together to shorten the time of reduction. Figure 2 shows the third order Rubik's cube robot control system of the symmetrical mechanical structure designed in this paper [18].

The function of the mechanical arm is to select the Rubik's cube. Th function of the chassis is to fix the mechanical arm. According to the control instruction, the robot controls the manipulator to carry out a series of movements of translation, rotation and opening and closing, so as to realize the process of reducing the Rubik's cube [19].

The mechanical structure of the robot makes the overall structure simple, saves the total time of reducing the cube, and improves the working efficiency and stability of the robot.

4. Hardware circuit system

The hardware part of the system is composed of the electronic part and the mechanical part. The electronic part mainly includes the MCU (Microcontroller Unit) minimum system main circuit with STC90C516RD+ main chip, and the peripheral circuit with TCS 3200 colour sensor module, SG90 steering engine module and 5461BS digital tube display module. The mechanical part is mainly a variety of plastic bars, screws, metal gasket, such as support and mechanical arms.

4.1. System control module

The system is designed with STC90C516RD+. The characteristics of the MCU are shown in table 1.

Table 1. Characteristics of the STC90C516RD+

Working voltage(V)	5.5-3.3
Maximum clock frequency(Hz)	0-80M
FLASH	64k
SRAM(byte)	1280
Timer	3
UART	1
DPTR	2
EEPROM	-
Watchdog	Yes
Support power off to wake up external interrupts	4
Built-in reset	Yes

The extension pin interface, extension power supply interface and ISP (Internet Service Provider) interface are mainly for the convenience of connection with other modules and the convenience of program burning, and the peripheral interface of the single chip can meet the needs of information transmission and motor drive. The crystal oscillator of 11.0592 MHz is used as the oscillation source. Since the single chip has an oscillating circuit inside, it is only necessary to connect one crystal oscillator and two capacitors externally. The capacitance is generally between 15 pF and 50 pF.The pull-up resistance in the system ACTS (Ability and Competence Test System) as a current limiter and is used to solve the problem of providing current when the bus has insufficient driving capacity.

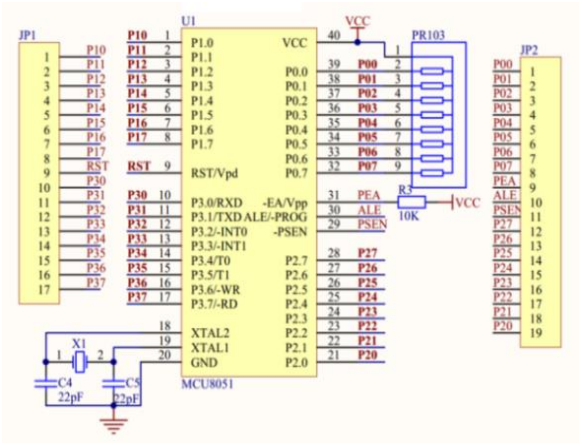

Figure 2. MCU minimum system

Figure 2 is the minimum system of single chip microcomputer, which is the control centre of hardware circuit system, responsible for processing data and controlling each motor.

4.2. Colour sensor acquisition module

According to the principle of tri-primaries induction, if we know the value of the tri-primaries that make up various colours, we can know the colour of the object under test. For TCS 3200, when selecting a colour filter, it allows only one primary colour to pass, preventing the other primary colours from passing. For example, when the red filter is selected, only red can pass through the incident light, and blue and green are blocked, so that the intensity of red light can be obtained. Similarly, by choosing other filters, you get the intensity of the blue light and the green light. With these three values, the colour of the light projected onto the TCS 3200 sensor can be analysed [20].

For the light sensor of TCS 3200, its sensitivity to the three basic colours of red, green and blue is different, resulting in that the RGB output of TCS 3200 is not equal. Therefore, white balance adjustment must be made before testing, so that TCS 3200 is equal to the three primary colours in the detected "white".

The LED control port is supposed to be high level. When TCS 3200 is powered on, four LED lights will glow. In order to make TCS 3200 get more light sources for better effects, LED lights are selected to keep on, so the LED control port cannot connect the control signal. The street of the OUT signal output port is the P3.4 port of the MCU, which is the external counting port of the timer/counter T0 (Timer/counter of single chip microcomputer) of the MCU and is responsible for receiving the signal collected by the colour sensor.

4.3. 5461BS digital tube display module

Since the design needs to use 8-bit and 8-segment LED digital tube, two 74HC573 latches are needed to control it, and the control mode is dynamic scanning.

Dynamic scanning has one drawback: if scan too fast, digital tube brightness is not enough; If the scan is too slow, the digital tube will flicker. Therefore, in the design, under the condition that the scanning speed of the digital tube is reasonable and not flickering, the use of the latch will guarantee the brightness of the digital tube will not be too dark.

The control port L of the two latches is used for selection: when the L port of a latch is at high voltage, the latch is selected, and its output end varies with the input end; when the L port is at low voltage, the latch is locked, and its output end does not change with the input end.

5. The software system

The main program flow chart of the whole system mainly judges the face recognized by the colour sensor, and makes the Rubik's cube rotate differently according to the corresponding colour. If white is detected, the cube is scrambled; if yellow, the cube is reduced; and if other colours, the cube is not turned.

5.1. Colour sensor acquisition module design

Before the subroutine, the first is to define the relevant interface. The eight-bit digital tube is driven by two latches, so two ports, P 2.6 and P 2.7, need to be controlled respectively. In the subroutine, firstly, define the relevant variables, and then set the parameters of the relevant register TMOD (TMOD is the working mode register of timer/counter) of the timing counter. Because these three values of RGB (Red, Green, Blue) need to be detected, the "for" loop is used. After the colour sensor has been detected for three times, it starts to analyse the data. The specific method is actually very simple: test the RGB values of the six surfaces of the Rubik's cube separately (after the parameter debugging of TH1(High 8 bits of timer/counter T1) and TL1 (Lower 8 bits of timer/counter T1 has been determined), find the data characteristics that are easy to distinguish, and then the value of the variable "mian" can be determined. Start the program of digital tube display after the Rubik's cube has been recognized by the colour sensor. Similarly, this is implemented by using the "for" loop. Each byte in RGB is individually taken out and displayed on the digital tube. The output process firstly clears all the contents displayed in the original digital tube, so as to prevent the "double image" of the digital tube. The method is to lock all P0 ports after setting 1.

5.2. Steering gear module design

The control mode of steering gear is PWM (Pulse Width Modulation) mode. The key codes of the steering gear control subroutine are as follows:

```
Procedure FBLR (char i)
Begin
    If(i== 1) Then
    {DJ(0x0202); DJ(0x0101); DJ(0x0002); DJ(0x0001);/} 'F
    If(i==-1) Then
    {DJ(0x0101); DJ(0x0202); DJ(0x0001); DJ(0x0002);/} 'F'
    If(i== 2) Then
    {DJ(0x0808); DJ(0x0404); DJ(0x0008); DJ(0x0004);/} 'R
    If(i==-2) Then
    {DJ(0x0404); DJ(0x0808); DJ(0x0004); DJ(0x0008);/} 'R'
    If(i== 3) Then
    {DJ(0x2020); DJ(0x1010); DJ(0x0020); DJ(0x0010);/} 'B
    If(i==-3) Then
    {DJ(0x1010); DJ(0x2020); DJ(0x0010); DJ(0x0020);/} 'B'
    If(i== 4) Then
    {DJ(0x8080); DJ(0x4040); DJ(0x0080); DJ(0x0040);/} 'L
    If(i==-4) Then
    {DJ(0x4040); DJ(0x8080); DJ(0x0040); DJ(0x0080);/} 'L'
    If(i==13) Then
    {DJ(0x1010);DJ(0x2020);DJ(0x0010);DJ(0x0222);DJ(0x0101);
DJ(0x0002); DJ(0x0001);/} 'FB'
    If(i==-13) Then
    {DJ(0x0101);DJ(0x0202);DJ(0x0001);DJ(0x2022);DJ(0x1010);
DJ(0x0020);DJ(0x0010);/} 'F'B
    If(i==24) Then
    {DJ(0x4040);DJ(0x8080);DJ(0x0040);DJ(0x0888);DJ(0x0404);
DJ(0x0008);DJ(0x0004);/} 'RL'
    If(i==-24) Then
    {DJ(0x0404);DJ(0x0808);DJ(0x0004);DJ(0x8088);DJ(0x4040);
DJ(0x0080);DJ(0x0040);/} 'L'R
    End

Procedure DJ(uint k)
Begin
    Dim A1=k As uchar;
    Dim A2=k>>8 As uchar;
    Dim B1=(A1&0x55)|(A1&0xAA&A2) As uchar;
    Dim B2=(B1&0xAA)|(B1&0x55&A2) As uchar;
    Dim i,j As uchar;
    For(i=40 To 0)                    'Time
    {
        For(j=123 To 0);
        {
          P1←A1;    'All steering gear started
          DelayMs(1);
          DelayUs2x(28);
         P1←B1;  'Turn to steering gear starting position
          DelayUs2x(70);
         P1←B2;  'Give the steering gear an "in" position
          DelayUs2x(255);
          DelayUs2x(90);
         P1←0;  'With the end point as the benchmark, all steering
gear to stop
          DelayMs(9);         'Speed
         /}
    /}
End
```

For the steering engine driven subroutine, the first thing is to understand the formula in the Rubik's cube restoring. "F, B, L, R, U, D" is the basic letters of the Rubik's cube, it means six surfaces clockwise turn 90° at a time. If it is counter-clockwise, it is " F ', 'B, L, R', 'U', D ' ". So, if these letters are written to formula, it needs two information. For example, the letter "F" says that the first information is the front of the Rubik's

cube, and second information is that this surface needs to be clockwise turned 90°. The robot has four arms totally, respectively controlling the front, right, rear and left side of the Rubik's cube. This design is to express the front, right, rear and left side of the Rubik's cube as "1, 2, 3, 4". For example, "2" says the right side of the Rubik's cube clockwise turn 90 °, "-3" says the Rubik's cube turn back to 90°. Each arm consists of two steering engines, one controlling the forward and backward motion of the arm and the other controlling the rotation of the arm. For example, to complete "2" (the right side of the cube), the robot arm needs to move four times, as shown in figure 3.

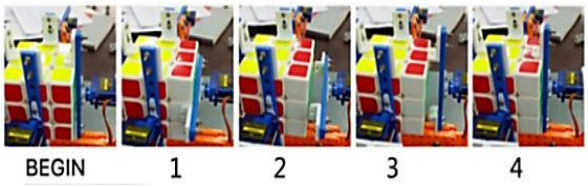

Figure 3. The right side of the Rubik's cube clockwise turn 90 °

In the subroutine "void FBLR(char I) (Procedure in pseudo code)", after the action "I = 2" is determined, four "DJ () (Procedure in pseudo code)" subroutines are used in the parentheses {}.The subroutine "DJ ()" has an input parameter of "uint k", which contains the steering gear used in this step and the steering gear's steering information. The low eight bits indicates which steering gear needs to be turned, and the high eight bits indicates the steering gear needs to be turned. For example, in "DJ(0x0888)", 0x0888 expands to the binary as "0000 1000 1000 1000", the lower octet as "1000 1000", and the fourth and eighth bytes as "1";Means that the steering gear 4 and steering gear 8 need to be rotated in this step. The eighth position is "0000 1000". The fourth word is "1", which means that steering gear 4 should rotate clockwise, and the eighth byte is "0", which means that steering gear 8 should rotate counter-clockwise.

After the parameter k (Driving variables in pseudo code) is designed, it starts to realize the rotation of the steering gear. In order to make the 8 steering gears work synchronously, the output port P1 should be controlled in parallel. If the single port is controlled one by one, the program is tedious, and the control of each steering gear will bring accumulated error. Therefore, before the real output of the control signal, it is necessary to carefully analyse the rotation angle of the steering gear, and analyse the value of the parallel control signal corresponding to different time through the parameter k.

The analysis of the rotation angle of the steering gear is as follows: for example, for the first front manipulator, the steering gear 2 that controls the clockwise and reverse rotation of the manipulator needs to swing by 90 °, and the steering gear 1 that controls the forward and backward of the manipulator needs to swing by 60 ° (it can be determined after debugging). For a single steering gear, it only needs to control by 0 ° and 60 ° And 90 °. For a 180 ° swing steering gear, the swing angle is the most stable around 90 °, so I designed the actual 45 ° position of the steering gear as the starting position of the manipulator, the actual 105 ° position after 60 ° as the second position of the manipulator, and the actual 135 ° position after 30 ° as the third position of the manipulator.

After the angle analysis of the steering gear, the parameter k needs to be analysed to get the value of the control signal at different times. First, start all the steering gear to be controlled. The control signal can be directly obtained from the low octet of k. The way to take out the low octet of k is to directly amplitude the value of k to an 8-bit VARIABLE A1 (Driving variables in pseudo code). Because the variable k is 16 bits, after "uchar A1 = k", the high octet of k will overflow and the remaining low octet will remain in A1. Of course, the way to start is "P1 = A1". After starting, after a period of time, when all the servos turn to the first position, some of the four servos (1, 3, 5, 7) that control the mechanical arm will stop here, so the control signal will be output again. The method is to first take out the high eight bits of variable k to A2 (Driving variables in pseudo code), among the eight bytes of A2. Because the even bytes control the four servos, take out the odd bytes of A1 (A1 & 0x55) first Carry out reservation, because the odd steering gear can't stop here, take out the even byte (A1 & 0XAA) of A1 and position signal A2 for "and" operation (A1 & 0XAA & A2), and carry out "or" operation between the reserved odd byte and the even byte after operation to get B1 (Driving variables in pseudo code), which is the control signal of the first position at this time. After a period of time, reach the second position, control the four steering gears (2, 4, 6, 8) of the manipulator. Some of them need to stop here. In the same way, first reserve the even byte of the previous state B1, use the odd byte of B1 and the position signal A2 for "and" operation, and then "or" operation with the even byte of B1. The final result B2 (Driving variables in pseudo code) is the second position at this time Control signal of. Finally, after a period of time to reach position 3, it is obvious that there is no steering gear to turn back, so at this time, you only need to stop all steering gear (P1 = 0).

Because the control of the steering gear is PWM mode, in the program structure, when the "P1 = 0" is executed, the control signal shall be output repeatedly after a period of time delay until the steering gear is turned to the correct position. If the delay time here is too long, the steering gear will rotate slowly. If it is too short, the steering gear will not execute PWM accurately. The specific value can only be determined after debugging. Another point is that since the control signal should be output repeatedly until the steering gear is turned to the correct position, the for cycle structure needs to be used. If the number of cycles is too small, the steering gear will not turn to the correct position. If the number of cycles is too many, the steering gear will stop for a period of time after turning to the correct position before starting the next rotation of the steering gear. The specific value can only be determined after debugging.

5.3 The key code of the colour recognition subroutine

The key code of colour recognition subroutine is as follows:

```
Procedure YanSe()
Begin
    Dim i,d,R,G,B As uchar;
    Dim j,k,RGB As ulong;
    TMOD←0x1E;
    'TMOD=(GATE,C/!T,M1,M0,GATE,C/!T,M1,M0,)=(0001 1110)-
T0 count, T1 timing
    EA←1;
    S0←S1←TCON←0x00;
    For(i=0 To 2)
    {
       TH1←250; TL1←93;        TL0←0;
       If(i==0) Then
       {S0←1; S1←0; S2←0; S3←0;/}
       '20%-Open R
       If(i==1) Then
       {S0←1; S1←0; S2←1; S3←1;/}
       '20%--Open G
       If(i==2) Then
       {S0←1; S1←0; S2←0; S3←1;/}
       '20%--Open B
       TCON←0x50;
       'TCON=(TF1,TR1,TF0,TR0,IE1,IT1,IE0,IT0)=(0101 0000)
       While(!TF1) Do ;
       S0←S1←TCON:=0x00;
       If(i==0) Then R←TL0;
       If(i==1) Then G←TL0;
       If(i==2) Then B←TL0;
       RGB←i<1?(TL0):(RGB*256+TL0);
    /}
    If(R>G && R>B) Then
    mian←R>0x20?4:3;
    If(G>R && G>B) Then mian←R>0x12?2:6;
    If(B>R && B>G) Then mian←R>0x0b?1:5;
    RGB←RGB*16+mian;
    For(i=0 To 7, j=16, k=1, d=RGB%16 To
d=RGB%(j*=16)/(k*=16))
    {
       LED_OUT←0xFF;SuoCun_d←0;
       If(i==1) Then i←2;
       LED_OUT←Wei[7-i];
       SuoCun_w←1;
       SuoCun_w←0;    'Bit selection
       LED_OUT←LED[d];          SuoCun_d←1;
       'Segment selection
    /}
    LED_OUT←0xFF;
    SuoCun_d←0;
    EA←0;
End
```

Before the subroutine, the first is to define the relevant interface, the eight-bit digital tube is driven by two latches, so it needs two ports P2.6 and P2.7 for control. The former controls which section of each digital tube to light, a digital tube has eight segments, the eight ports of the single-chip microcomputer P0 port are exactly one-to-one corresponding, the digital tube needs to display 0~F, the corresponding bright value is stored in the array "LED[16]".The latter control which digital tube to display, so it also requires eight port control, because there is a latch, so still use P0 port control.

In a subroutine, you first define the associated variables, and then set the parameters of the associated register TMOD for the timing counter. TMOD is an eight-bit register, with the high four-bit setting for the timing counter T1((Timer / counter of single chip microcomputer)) and the low four-bit setting for the timing counter T0.All four of them are "GATE,C/!"T,M1,M0(Working mode setting) "for example, in the high four," GATE "is the control mode. If" GATE==1 ", it is controlled by the external port P 3.5; otherwise, it is not. "C/!T "is the working mode, if" C/!T==1 ", then the mode of timing counter T1 is counting, otherwise it is timing;"M1, M0" is the working mode. You can choose four working modes. For example, "M1,M0==01()" means the working mode is one. The same goes for the lower four bits. According to the principle of the colour sensor, my processing method is to count and calculate the time needed for the colour sensor to receive the signal when it is full of 255, so we need to use T0 and T1 at the same time. "TMOD=0x1E" means that T1 is timed in working mode and not externally controlled, and T0 is counted in working mode two counts and is externally controlled.

It is allowed to use T0 and T1,namely "EA (Total interrupt allowed bit) == 1", close these two timing counters and clear the related flag bits, i.e. "TCON (TCON is the control register, which controls the start and stop of T0 and T1 and sets the overflow flag) == 0x00". Of course, the colour sensor cannot work at this time, so turn off its sending signal. The RGB values need to be detected, the for loop is used. Set the values T1 and T0 in the loop. T1 needs to set TH1 and TL1. Because the timing is by counting, the two values can be written at will and determined after debugging. T0 is a count, and only needs to be counted to 255, while the full value of working mode 2 is exactly 255, so you only need to set "TL0 (Lower 8 bits of timer / counter T0) = 0", that is, counting from zero. After that, the red, green and blue channels of the colour sensor will be selected in turn for counting, and each time one is recorded, it will be saved in the variable "RGB". Each counting result needs to be saved in two bytes of the same hexadecimal, so it is not saved once, and only two bits of the variable RGB need to be moved to the left, that is, "RGB * 256".

After the colour sensor detects three times, it starts to analyse the data. The specific method is very simple. It is to test the RGB value of the colour of the six faces of the cube (after the parameters of TH1 and TL1 are determined, of course), find the data characteristics that are easy to distinguish, and then determine the value of the variable "mian". What we need to understand first is that the value of the variable "mian" corresponding to the colour "white, yellow, red, orange, blue and green" of the cube is "1, 2, 3, 4, 5 and 6". The data characteristics are specified in later debugging. After the value of the variable "mian" is determined, it is also stored in RGB. This is for the convenience of nixie tube display. Since only one byte is needed to determine 1-6, the original RGB value only needs to move one bit to the left.

After the colour sensor is identified, the digital tube display program will be executed. Don't write the digital tube display program as a subroutine because it can reduce the call of the display subroutine and make the real-time display effect of the digital tube better (of course, this is only known after debugging). Because there are 7 bytes of data in RGB, if the 8-bit digital tube displays, the 6-bit data of the extra colour sensor is separated from the data of the variable "mian" to facilitate data observation.

This is also done using a for loop, where each byte in RGB is taken out separately and displayed on a digital tube. So, the way to get the bytes is, instead of taking out the 3 out of the 12345, you can take 12345 divided by 1000 and take the remainder is 123, and then you take 123 divided by 100 and you get 3. After the bytes are fetched, the corresponding digital tube value needs to be output to the P0 port. Output process first of all the original digital tube display content all clear, this can prevent digital tube "double". The purpose of writing the parameters and changing conditions to speed up the byte output and reduce the "flicker" effect caused by the dynamic display.

6. The system implementation

The system has a four sides symmetrical structure. The four mechanical arms can work independently, which structure makes the operation stable and reliable. Use colour sensor to collect colour information.

In the process of debugging, the adjustment of the Rubik's cube is very important, because the torque of SG90 steering engine is not very big. If the Rubik's cube is too tight, it will not turn; if it is too loose, the Rubik's cube will easily fall apart. On the choice of Rubik's cube, we should choose a Rubik's cube with good smoothness and fault-tolerance.

In the process of testing the colour sensor module, we tested the RGB values of the six colours of the Rubik's cube and recorded the corresponding RGB values. Then leave the light source around 0.5cm and 1cm, and test all colour RGB values again, as shown in table 2. According to the data, the farther the colour sensor is from the cube, the smaller the RGB value is. However, when the colour sensor is within 0.5 cm from the cube, the largest of its RGB does not change, indicating that the size relation of each RGB value of the six faces is unchanged within the distance. So, using this property, we can tell the difference between the six sides.

Table 2. The detection of RGB value (decimal H)

	From the light source 0.0 cm			From the light source 0.5cm			From the light source 1.0cm		
	R	G	B	R	G	B	R	G	B
White face	33	33	43	14	13	1a	0a	09	0d
Yellow face	44	48	32	14	15	10	0a	09	09
Red face	20	0b	10	0c	07	0a	07	05	08
Orange face	79	19	22	1d	0a	0e	0f	07	09
Blue face	0b	13	25	08	09	0f	06	06	09
Green face	12	28	19	0a	0d	0d	06	07	08

The adjustment of the mechanical arm is also very important, especially the adjustment of the mechanical arm, because it is directly in contact with the Rubik's cube. If the adjustment is not good, it will cause that the Rubik's cube rotation is not in place or stuck. The buttons on both ends of the manipulator should be just enough to hold the cube, not too tight or too loose. The front of the button should be cut a chamfer with a knife or other tools, to facilitate the Rubik's cube get stuck.

For the mechanical arm, it is mainly responsible for the transmission mechanism, so it is necessary to ensure that the ground should be loose and the place should not be tight. The sliding groove of the mechanical arm is required to ensure that the sliding shaft in the middle can slide easily and not deviate from the axis.

Then, the rotary shaft of the rear end of the mechanical arm needs to be adjusted properly to ensure the best transmission effect. The rotating shaft mainly includes the rotating shaft A between the steering gear and the driving bar and B between the driving bar and the sliding shaft. The outer nut is the lock nut. Its main function is to ensure that the transmission bar runs smoothly on the rotating shaft and does not move from side to side.

The whole process of restoring Rubik's cube in the system is as follows: the colour sensor module recognizes each colour block of the Rubik's cube, and the digital tube part displays RGB value of corresponding colour in real time. The micro controller analyses the actual situation of Rubik's cube colour and the reduction algorithm of Rubik's cube. Finally, turn the cube by the steering gear, completing the reduction process.

After many tests, there have been some mistakes in the occasional time, such as mechanical arm card does not reach the designated position, or the mechanical arm pulls back the Rubik's cube. After testing and continuous improvement, the reliability of the Rubik's cube is about 80 %, which reaches the design goal of the Rubik robot.

7. Conclusions

In this paper, an embedded third-order Rubik robot is designed and implemented, and a control mechanism with four mechanical arms working independently is proposed, which can cooperate to complete the reduction process of the Rubik's cube. As the "eyes" of the robot, the colour sensor module is used to collect the colour information of the cube's independent blocks. Through this information and the corresponding reduction algorithm program, the micro controller analyses the corresponding reduction methods and controls the steering gear. Experiments show that the third-order Rubik's cube robot based on embedded can be quickly and stability to restore the Rubik's cube.

8. Bibliographic References

[1] Jian Hu. *Microcontroller principle and interface technology*. Beijing: Mechanical Industry Press, 2004.

[2] Jian Hu. *Practical course of single chip microcomputer*. Beijing: Ordnance Press, 2001.

[3] Yikun Zhang. *Principle and application of single-chip microcomputer*. Xi 'an: University Of Electronic Science And Technology Press, 1998, 1st edition.

[4] Tianxiang Guo. *Introduction to C language course of 51 microcontroller* [M]. Beijing: Electronic Industry Press, 2009.
[5] Lina Zhou. *Protel99SE circuit design technology*. Beijing: China Railway Publishing House, 2009.
[6] Haoqiang Tan. *C language programming*. Beijing: tsinghua university press, 2006.
[7] Shibai Tong and Chengying Hua. *Foundation of analog electronic technology*. Beijing: Advanced Education Press, 2001.
[8] Shi Yan. *Basic course of digital electronic technology*. Beijing: Tsinghua University Press, 2007.
[9] Guili Tan and Shangqing Wu. *Configuration software control technology* [M]. Beijing: Polytechnic University Press, 2007.
[10] Xiuying Yuan. *Configuration control technology* [M]. Beijing: Electronic Industry Press, 2003.
[11] Limin He. *McS-51 series MCU application system design system configuration and interface technology* [M], 1999.
[12] Qingming Lai. *Sensor and chip interface*. Beijing: University Of Aeronautics And Astronautics Press, 2008
[13] Xianwu Huang. *Practical application circuit design of sensors*. Sichuan: University Of Electronic Science And Technology Press, 1991.
[14] Yan Zhao. *Sensor theory and application*. Beijing: Peking University Press., 2009.
[15] Zhongkui Wang, Yuting Hao and Yan Sun. Design and implementation of temperature sensor based on WIFI technology. *Electronic World*, 2014, 10.
[16] J C Candy; G C Temes, *Oversampling Methods for Data Conversion*, 1991.
[17] [1] Jianbo Zhang, Qun Yin, "Design of echo cancellation based on FM1188". in *Electrotehnica Electronica Automatica (EEA)*. 2013, vol. 61 (no. 4): pp. 41-47.
[18] [2] Zhang JIANBO,Yin QUN, "Design of Public Regional Emergency Communication System based on RS485". in *Electrotehnica Electronica Automatica (EEA)*. 2014, vol. 62 (no. 3): pp. 126-134.
[19] [3] Yin Qun, Zhang Jianbo, "Design of control platform systems based on object-oriented". in *International Review on Computers and Software*. 2012, vol. 7 (no. 1): pp. 438-442.
[20] [4] Yin QUN, Zhang JIANBO, "Design of Power Line Carrier Communication System based on FSK-KQ330 Module". in *Electrotehnica Electronica Automatica (EEA)*. 2014, vol. 62 (no. 3): pp. 135-142.

Funding Sources

It is a pleasure to acknowledgment the contributions of prof. Yin Qun, Zhang Jianbo, students. Gu Ji, Yunsheng Xu, both of whom participated in a number of fruitful discussions. This work was financially supported by the City College, Kunming University of Science and Technology, Kunming Yunnan, China.

Authors' Biographies

Yin Qun was born in 1978.
She is a professor and is graduated from Kunming University ,the computer information engineering institute in China, in 2008.
She received her Bachelor Degree in Computer Science and Technology from Kunming University of Science and Technology, China in 2003, her Master's degree in Management Engineering and Science from Kunming University of Science and Technology, China in 2008.
She is currently a lecture in Kunming University of Science and Technology.
Her research interests include Embedded systems, single-chip measurement and control field, the computer control system, the interface control system.
e-mail: exnet@vip.qq.com

Zhang Jianbo was born in 1984.
He is a Senior engineer and is graduated the University of Yunnan, Software Institute in China, in 2011.
He received his Bachelor Degree in Computer Software Engineering from Oxbridge College, Kunming University of Science and Technology, China in 2007, his Master's degree in Computer Software Engineering from Yunnan University China in 2011.
He is currently a lecture in Oxbridge College, Kunming University of Science and Technology.
His research interests include Computer software engineering, information management systems, and embedded systems.
e-mail: 455601875@qq.com

Gu Ji was born in 1999.
He is a sophomore student in Kunming university of science and technology, majoring in electronic information engineering.
Gu Ji now as a sophomore student in Kunming university of science and technology, majoring in electronic information engineering.
His research interest are Internet of things(IOT)
e-mail: *819238610*@qq.com

Meisu Yin was born in 1999.
She is a sophomore student in Kunming university of science and technology, majoring in electronic information engineering.
Meisu Yin now as a sophomore student in Kunming university of science and technology, majoring in electronic information engineering.
Her research interest are Internet of things(IOT)
e-mail: *2498034856*@qq.com

Yunsheng Xu was born in 1998.
He is a sophomore student in Kunming university of science and technology, majoring in electronic information engineering.
Yunsheng Xu now as a sophomore student in Kunming university of science and technology, majoring in electronic information engineering.
His research interest are Internet of things(IOT)
e-mail: *1738133544*@qq.com