

IMDL Final Report  
Chameleon Ninja

Chien-Chih Chao

4/18/2011

# Contents

<b>1</b>	<b>Abstract</b>	<b>2</b>
<b>2</b>	<b>Executive Summary</b>	<b>3</b>
<b>3</b>	<b>Introduction</b>	<b>4</b>
<b>4</b>	<b>Integrated System</b>	<b>5</b>
4.1	Obstacle analysis system . . . . .	5
4.2	Line following system . . . . .	6
4.3	Color mimic system . . . . .	7
<b>5</b>	<b>Mobile Platform</b>	<b>8</b>
5.1	First level . . . . .	9
5.2	Second level . . . . .	9
5.3	Top level . . . . .	9
5.4	Front . . . . .	10
5.5	Back . . . . .	10
<b>6</b>	<b>Actuation</b>	<b>11</b>
6.1	DC motors . . . . .	11
6.2	Motor driver . . . . .	12
<b>7</b>	<b>Sensors</b>	<b>13</b>
7.1	Color sensor . . . . .	13
7.1.1	Sensor information . . . . .	13
7.1.2	Theory of color sensor . . . . .	14
7.1.3	Sensor gain optimization procedure . . . . .	15
7.1.4	Sensor operation procedure . . . . .	16
7.2	IR sensors . . . . .	17
7.2.1	Sensor information . . . . .	17
7.2.2	Theory of IR sensor . . . . .	18
<b>8</b>	<b>Behaviors</b>	<b>19</b>
8.1	Algorithm . . . . .	19
<b>9</b>	<b>Conclusion</b>	<b>20</b>
<b>A</b>	<b>References</b>	<b>21</b>
<b>B</b>	<b>Program Code</b>	<b>22</b>

# Chapter 1

## Abstract

The purpose of this document is to build a robot named Chameleon Ninja.

Starting with an overview of the system would cover the detailed about the scopes and types of the platform, actuation, and the sensors, followed by the work result and technical dicussion of the design.

The reference page has a list of relenavt books, websites, and electronics shops.

In the end, the appendices have all program cods, circuit diagrams ,and other supplementary material.

## Chapter 2

# Executive Summary

This is a project from the intelligent machines design laboratory course of university of Florida, department of ECE, which provides by Dr. A. Antonio Arroyo and Dr. Eric M. Schwartz. The TAs are Josh Weaver, Ryan Stevens, Tim Martin, Devin Hughes, and Sean Frucht. To get more information, go to <https://sites.google.com/site/imdlchameleoninja/>

## Chapter 3

# Introduction

What is chameleon? Chameleons are a distinctive and highly specialized clade of lizards, which have the ability to change the color of their skins. What is ninja? A Ninja was a secret agent of ancient Japan specializing in unorthodox arts of war.

What is Chameleon Ninja? Chameleon Ninja is a robot that combined the concepts of these two creatures. Chameleon Ninja is a 3-wheeled robot that can follow a path. But compare to ordinary line follow robot, Chameleon Ninja can alternate the LCD screen colors on its top by detecting the ground color under it. In addition, Chameleon Ninja also has ability to detect the obstacles on its front and has different response by the size of the obstacles.

## Chapter 4

# Integrated System

### 4.1 Obstacle analysis system

- 1 Detecting front obstacle from IR-sensor array(2X3).
- 2 Compute how much size of the obstacle.
- 3 Send command to motor system to drive the robot.

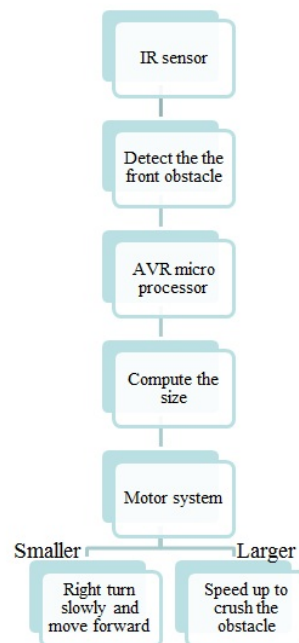


Fig. 4.1

## 4.2 Line following system

Following is the same algorithm in [3] that I used in this project.

- 1 Using IR sensor array to detect the black line.
- 2 Define L(or R) = left(or right) most sensor which reads 0, all sensor read 1 then L(or R) equal 0.

Example 1: 

L3	L2	L1	R1	R2	R3
1	0	0	1	1	1

 means  $L = 2, R = 0$

Example 2: 

L3	L2	L1	R1	R2	R3
1	1	0	0	0	0

 means  $L = 1, R = 3$

- 3 If all sensors read 1, go to step 4.

Else, if  $L > R$  Move Left; if  $L < R$  Move Right; if  $L = R$  Move Forward. Then go to step 1.

- 4 Move clockwise if line was last seen on right.

Move counter clockwise if line was last seen on left

Repeat step 4 till line is found.

- 5 Go to step 1.

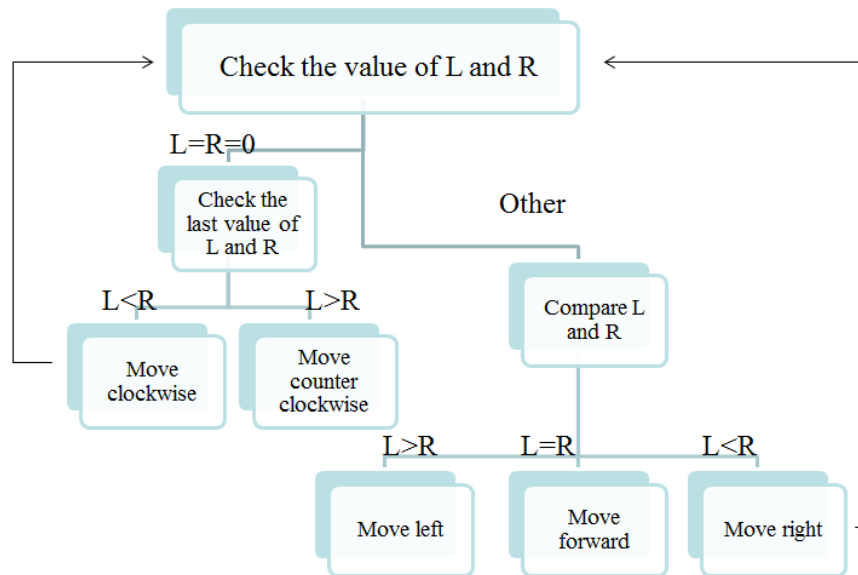


Fig. 4.2

### 4.3 Color mimic system

- 1 Detect the ground color by ADJD-S371-Q999.
- 2 Classify the color data.
- 3 Send command to LCD screen chip.
- 4 Chang the color of LCD screen.

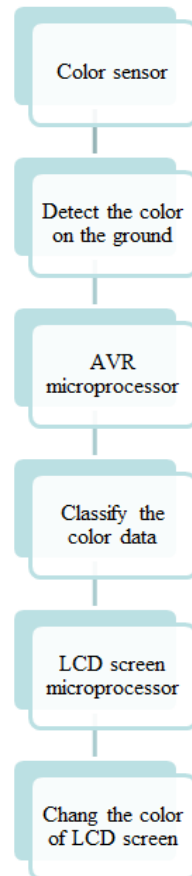


Fig. 4.3



## Chapter 5

# Mobile Platform

The main purpose of design the Chameleon Ninja is that make the LCD screen looks bigger. To achieve this goal by the finite budget, we have to make the robot as small as possible. The size limitations are the PVR board, the battery pack and the LCD screen. Since the largest size of LCD that I can afford is 3.2", the most large component is PVR board. Therefore, I design a three-level platform for Chameleon Ninja.

This was the first time for me to design my own robot, there were many drawbacks of the Chameleon Ninja's prototype. Thanks for the size of my design is small enough so I could have chance to rebuild the platform again. The second generation of Chameleon Ninja platform have several advantages as following.

- 1 Easily to assemble without using any glue
- 2 Have room to place the second LCD screen on the back
- 3 Two switches on the top

## 5.1 First level

The first level will be placed with battery pack, motors and motor driver, color sensor and IR sensor array for line following.

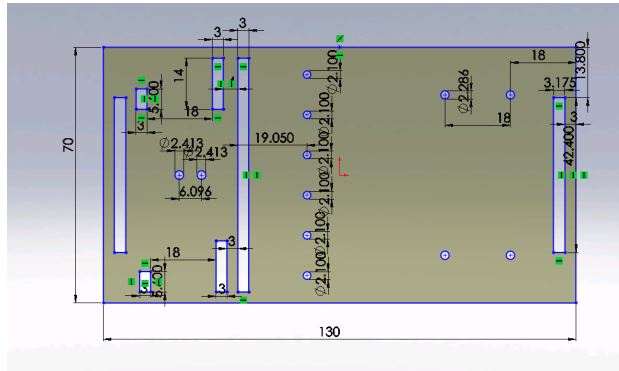


Fig. 5.1 First level (unit mm)

## 5.2 Second level

The second level is PVR board.

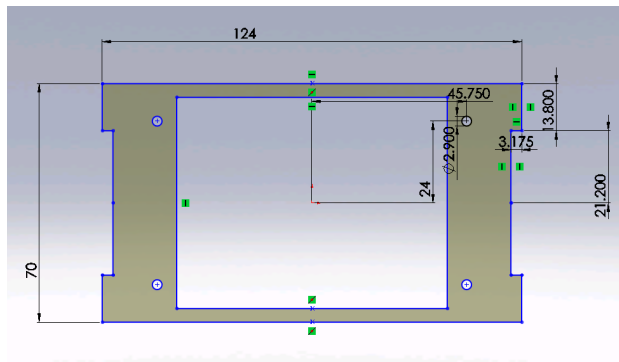


Fig. 5.2 Second level (unit mm)

## 5.3 Top level

The top level is LCD screen and switches.

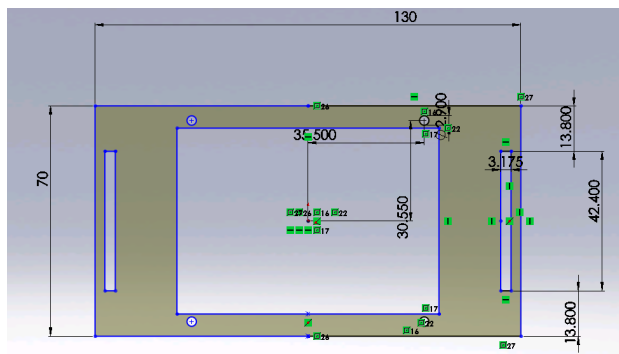


Fig. 5.3 Top level (unit mm)

## 5.4 Front

The front will have an array(2X3) IR sensors to detect the obstacle.

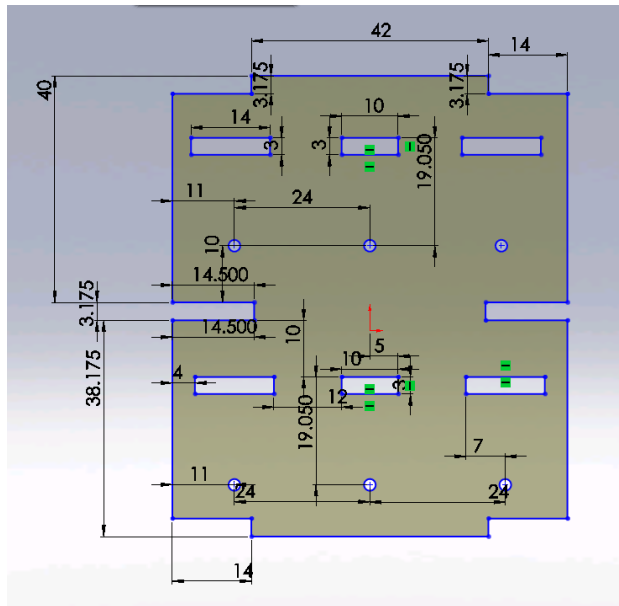


Fig. 5.4 Front (unit mm)

## 5.5 Back

The back will have the second LCD screen and the most important part of the design, which is the logo of the Chameleon Ninja.

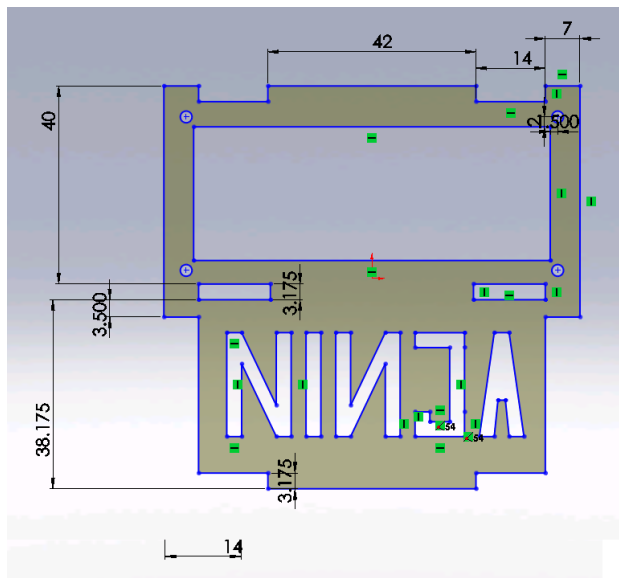


Fig. 5.5 Back (unit mm)

# Chapter 6

## Actuation

In order to build the motor system of Chameleon Ninja, we have two objectives to achieve. First, the motors should not be too large since we want to keep the robot size smaller. Second, the motor should have enough torque to drive Chameleon Ninja. These two limits make Sanyo 250:1 micro metal gearmotor HP is the best choice that I found.

### 6.1 DC motors

- 1 Model: Sanyo 250:1 Micro Metal Gearmotor HP
- 2 Pololu Robotics and Electronics
- 3 Free-run speed @ 6V:120 rpm
- 4 Free-run current @ 6V:70 mA
- 5 Stall torque @ 6V:60 oz-in
- 6 Scope of motor

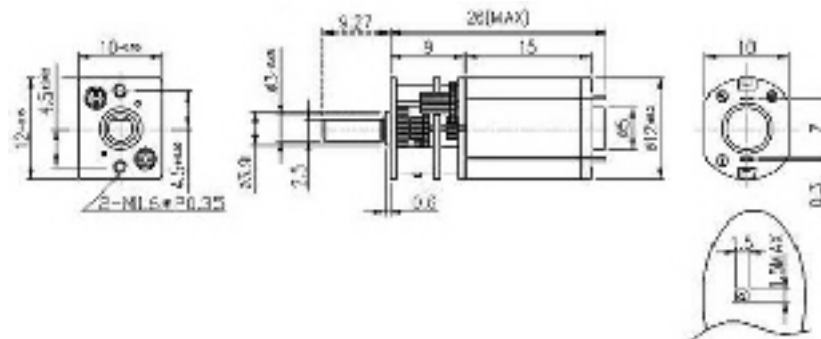


Fig. 6.1 Micro metal gearmotor dimensions (units in mm)<sup>[1]</sup>

## 6.2 Motor driver

- 1 Model: Motor Driver 1A Dual TB6612FNG
- 2 SparkFun Electronics
- 3 Power supply voltage:  $V_M=15V$  max,  $V_{CC}=2.7-5.5V$
- 4 Output current:  $I_{out}=1.2A$ (average) / 3.2A (peak)
- 5 CW/CCW/short brake/stop motor control modes
- 6 Built-in thermal shutdown circuit and low voltage detecting circuit
- 5 Scope of motor driver

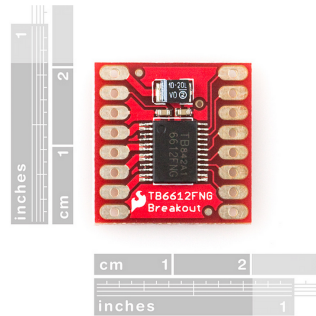


Fig. 6.2<sup>[1]</sup>

### 6 Control Function

Input				Output		
In1	In2	PWM	STBY	Out1	Out2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
L	H	L	H	L	L	Short brake
H	L	H	H	H	L	CW
H	L	L	H	L	L	Short brake
L	L	H	H	OFF	OFF	Stop
H/L	H/L	H/L	L	OFF	OFF	Standby

# Chapter 7

## Sensors

### 7.1 Color sensor

One of the main function of Chameleon Ninja is detecting the ground color. Because the sensor will be set under the bottom of the robot, it should also have a LED to light out the objective to detect.

#### 7.1.1 Sensor information

- 1 Model: Avago ADJD-S371-Q999 and Evaluation Board
- 2 SparkFun Electronics
- 3 10 bit per channel resolution
- 4 Two wire interface(I2C) communication
- 5 Scope of sensor

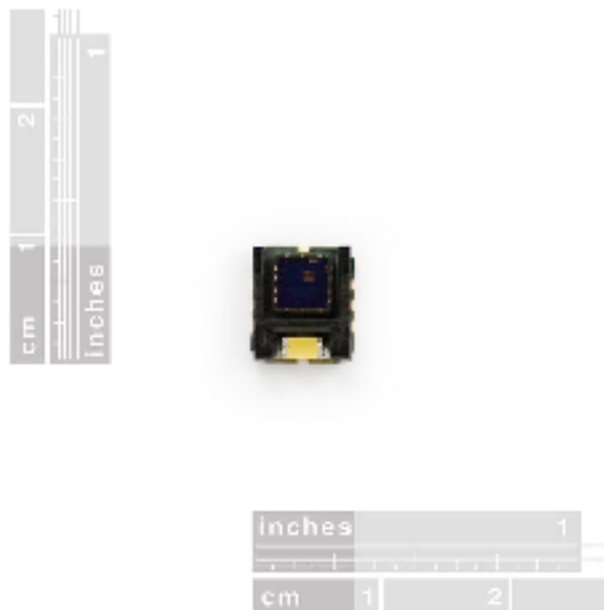


Fig. 7.1<sup>[1]</sup>

## 6 Schematic of breakout board

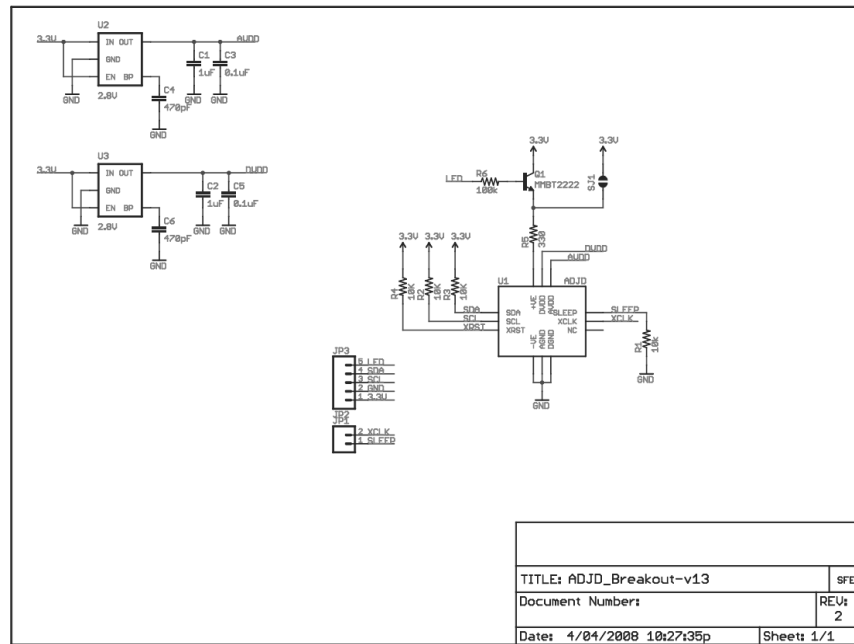


Fig. 7.2<sup>[1]</sup>

### 7.1.2 Theory of color sensor

Every color can be combined as three different intensity of red, green and blue. The sensor will detect the color of objective by the light reflected from the white LED light on the breakoutboard. The LED lamps are lighted one by one and grouped according to its color. The filter-coated photodiode array of the color sensor converts the incident light into digital R, G and B reading. Since all three outputs increase linearly with increasing light intensity, the sensor can measure both color and total intensity of the light.

### 7.1.3 Sensor gain optimization procedure

- 1 First perform an external power on reset. Wait 10 $\mu$ s for the reset sequence to be completed.
- 2 Write sensor gain registers, CAP\_RED, CAP\_GREEN, CAP\_BLUE and CAP\_CLEAR to select the number of capacitor. The values must range from 00H to 0FH.
- 3 Write sensor gain registers, INT\_RED, INT\_GREEN, INT\_BLUE and INT\_CLEAR to select the integration time. The integration time registers is a 12-bit registers, the values is range from 0 to 4095. A higher value in integration time will generally result in higher sensor digital value if the capacitance gain registers have the same value.
- 4 Read the data from sensor. If these sensor digital values are not optimum, do another iteration loop consisting of step 2, 3 and 4
- 5 If the sensor digital values obtained in step 4 are optimum, a check is done to ensure that the sensor digital values will never exceed 1000. This is done by selecting running condition or any operating case which the sensor will give maximum value in
  - a) red, green and blue channel, or
  - b) any of the red, green or blue channels.

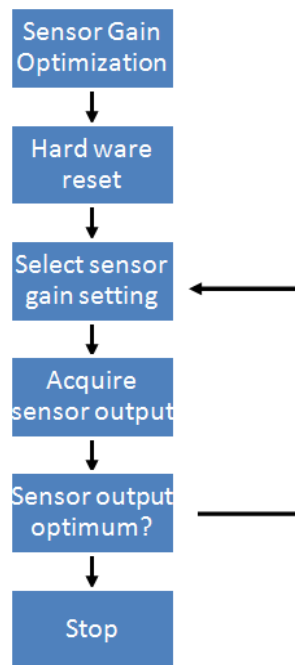


Fig. 7.3



#### 7.1.4 Sensor operation procedure

- 1 First perform an external power on reset. Wait 10 $\mu$ s for the reset sequence to be completed.
- 2 Write sensor gain registers, CAP\_RED, CAP\_GREEN, CAP\_BLUE and CAP\_CLEAR with values obtained from the previous sensor gain optimization procedure.
- 3 Write sensor gain registers, INT\_RED, INT\_GREEN, INT\_BLUE and INT\_CLEAR with values obtained from the previous sensor gain optimization procedure.
- 4 Acquire offsets in normal operating environment. The light source for color sensing must be turned off. The offsets are acquired by writing 02H to CTRL register. By writing 01H to CONFIG register (address 01H), all digital values of the sensor will automatically trim the offset
- 5 Now the sensor is ready to read colors. Sensor digital values can be acquired by writing 01H to CTRL register (address 00H). Read CTRL register. When the value in CTRL register is 00H, sensor digital values are acquired in sensor sample data registers.

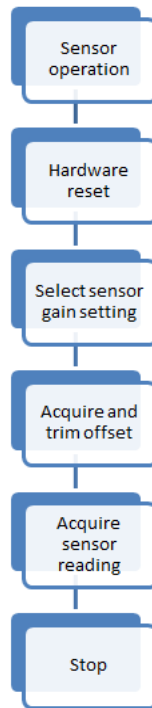


Fig. 7.3

## 7.2 IR sensors

The other functions of Chameleon Ninja is detecting and analysing the size of the obstacle in front of it, and also following the black line on the ground. We can simply apply the IR sensor to achieve both goal.

### 7.2.1 Sensor information

- 1 Model: Sharp GP2Y0D810Z0F Digital Distance Sensor
- 2 Pololu Carrier Breakout Board
- 3 Sensing distance to 5 cm or 10 cm
- 4 Simple digital output, high for nothing or black, other is low.
- 5 Scope of sensor breakout board

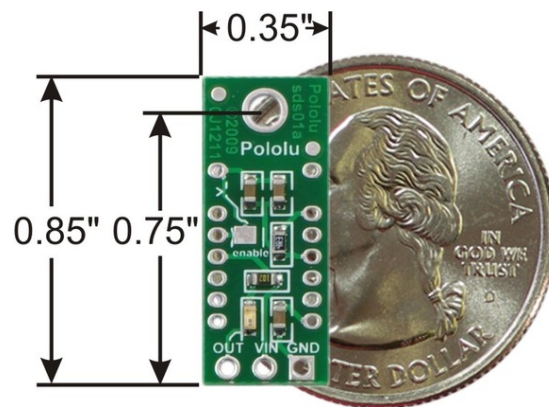


Fig. 7.4<sup>[2]</sup>

- 6 Schematic of breakout board

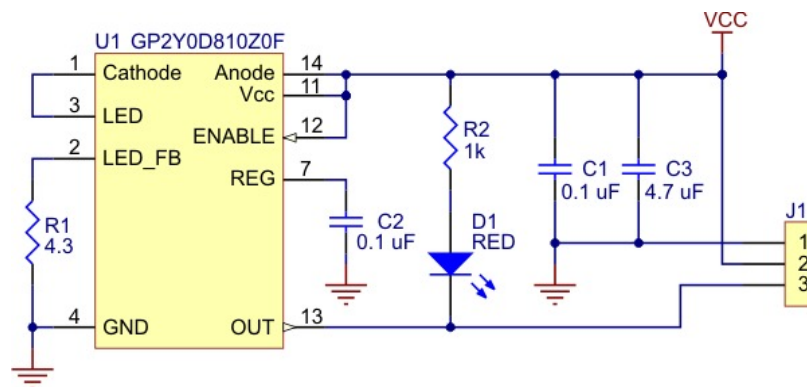


Fig. 7.5<sup>[2]</sup>

### 7.2.2 Theory of IR sensor

GP2Y0D810Z0F is distance measuring sensor unit, composed of an integrated combination of PD(photo diode) , IRED(infrared emitting diode) and signal processing circuit. The sensor data base can be shown in Fig. 7.6.

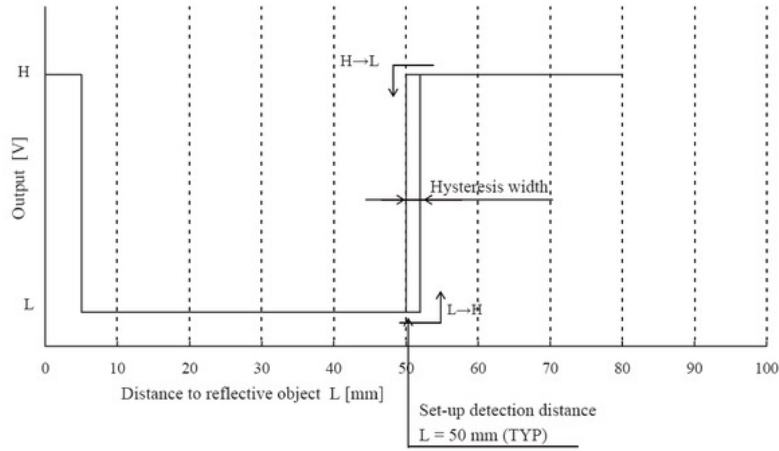


Fig. 7.6<sup>[2]</sup>

# Chapter 8

## Behaviors

### 8.1 Algorithm

- 1 Detect the color of ground.
- 2 Imitate the color on the LCD on top.
- 3 If the color is red, left turn about 90 degree.
- 4 Detect the front obstacle.
- 5 If there are no obstacle, go to step 6.  
If the obstacle is larger, stop and avoid slowly.  
If the obstacle is smaller, speed up to crush the obstacle.
- 6 Detect the line on the ground.
- 7 Follow the line or move forward to find the line.

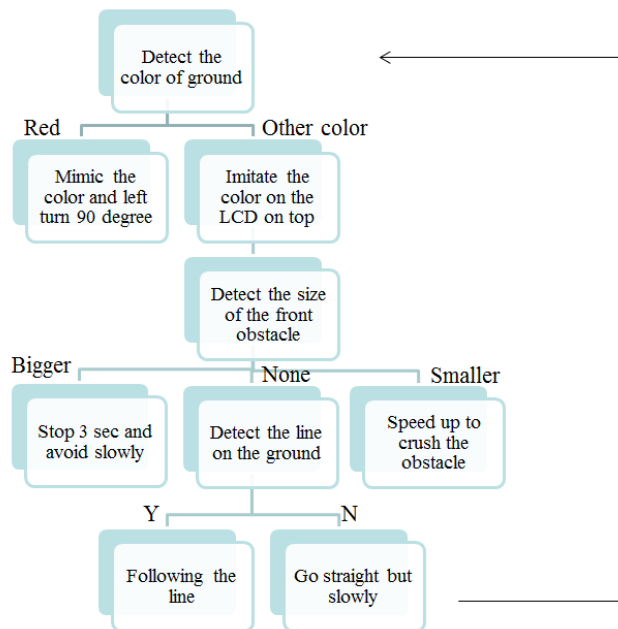


Fig. 8.1

## Chapter 9

# Conclusion

The most challenge in this project is to get the color sensor working. First, I need to enable the communication between the color sensor and the AVR xmega128A1 micro processor. Second, I have to build the protection of external lights for the color sensor in order to have more accuracy color data.

This is also the first time for me to design a platform for robot. The T-Tech machine is very accurate that I can assemble my platform without using any glue. However, I found a lot of disadvantage of the first design and need to redesign the new platform. Fortunately, the size of my robot was small enough to have two different platform.

I didn't get very good performance of line following at first. The reason was that the control algorithm just changed the speed of turning for different angle of curve. I designed the new algorithm is that both wheels are move forward in different speed, and will able to have better performance of following the curve smoothly.

In the USART command to the LCD screen, the AVR processor needs a feedback signal of LCD screen to notify the color changing completely. And it tooks more than one second to changing the back ground color of LCD. So I couldn't perform the line following and color changing in the same time. In the final demo day, Chameleon Ninja have to stop when it detect a diffent color on the ground. In order to get these two functions working simultaneously, I add the line following command in the while loop of waiting the feed back signal of LCD screen. Finally, I can demo the color changing and the line following in the same time.

I really had a wonderful experience in this course. Thanks for all effort from teachers and TAs. And I also learned a lot from the classmate also.

# Appendix A

## References

- [1] The website of Sparkfun Electronics <http://www.sparkfun.com/>
- [2] The website of Pololu Robotics & Electronics. <http://www.pololu.com/>
- [3] Priyank Patil, "Line Following Robot"
- [4] The website of Atmel of ATxmega 128A1  
[http://www.atmel.com/dyn/products/product\\_card.asp?part\\_id=4298](http://www.atmel.com/dyn/products/product_card.asp?part_id=4298)  
including `avr_compiler.h`, `twi_master_driver.h`, `usart_driver.h`, `twi_master_driver.c`, and `usart_driver.c`
- [5] PVR manual and PVR example code.

# Appendix B

## Program Code

```
#include <avr/io.h>
#include "PVR.h"
#include "usart_driver.h"
#include "avr_compiler.h"
#include "twi_master_driver.h"
/*! Define that selects the Usart used in this project. */
#define USART USARTE1
USART_data_t USART_data;
/*! Success variable, used to test driver. */
bool success;
/*! Number of bytes to send. */
#define NUM_BYTES 3
/*! Color data to send. */
uint8_t sendStart[1] = {0x55};
uint8_t sendClear[1] = {0x45};
uint8_t sendWhiteBG[NUM_BYTES] = {0x42, 0xff, 0xff};
uint8_t sendBlueBG[NUM_BYTES] = {0x42, 0x00, 0x1f};
uint8_t sendGreenBG[NUM_BYTES] = {0x42, 0x07, 0xE0};
uint8_t sendYellowBG[NUM_BYTES] = {0x42, 0xff, 0xE0};
uint8_t sendRedBG[NUM_BYTES] = {0x42, 0xf8, 0x00};
uint8_t sendPurpleBG[NUM_BYTES] = {0x42, 0xf8, 0x1f};
uint8_t sendCyanBG[NUM_BYTES] = {0x42, 0x07, 0x1f};
uint8_t sendOrangeBG[NUM_BYTES] = {0x42, 0xfc, 0x02};
uint8_t sendBlackBG[NUM_BYTES] = {0x42, 0x00, 0x00};
/*! Defining an ADJD slave address. */
#define ADJD_ADDRESS 0x74
/*! CPU speed 32MHz, BAUDRATE 9600Hz and Baudrate Register Settings */
#define CPU_SPEED 32000000
#define BAUDRATE 100000
#define TWI_BAUDSETTING TWI_BAUD(CPU_SPEED, BAUDRATE)
/* Global variables */
TWI_Master_t twiMaster; /*!< TWI master module. */
/* ADJD capacitor address */
uint8_t CAP_RED[2] = {0x06, 0x02};
uint8_t CAP_GREEN[2] = {0x07, 0x02};
uint8_t CAP_BLUE[2] = {0x08, 0x02};
uint8_t CAP_CLEAR[2] = {0x09, 0x02};
uint8_t INT_RED_LO[2] = {0x0A, 0xC4};
uint8_t INT_RED_HI[2] = {0x0B, 0x09};
```

```

uint8_t INT_GREEN_LO[2] = {0x0C, 0xC4};
uint8_t INT_GREEN_HI[2] = {0x0D, 0x09};
uint8_t INT_BLUE_LO[2]   = {0x0E, 0xC4};
uint8_t INT_BLUE_HI[2]   = {0x0F, 0x09};
uint8_t INT_CLEAR_LO[2] = {0x10, 0xC4};
uint8_t INT_CLEAR_HI[2] = {0x11, 0x09};
uint8_t rh,rl,gh,gl,bh,bl; //define the data from ADJD
uint8_t    ColorFlag,LcdFlag =0;
void adjd_init(void);
void adjd_read(void);
void adjd_data(void);
void Color_recognize(void);
void LcdColor(void);
void USARTInit(void);
void USARTcmd(uint8_t NUM, uint8_t sendData[]);
void USARTcmd_line(uint8_t NUM, uint8_t sendData[]);
#define MAX 3
#define HMAX 1
uint8_t i,history[MAX],dirL,hcount=0;
static uint8_t Rdev,Ldev,Final_dev;
static uint8_t FflowStatus=0;
void ObstacleDetect(void);
void LineDetect(void);
void LineFollow(void);
void MotorInit(void);
void Forward(int value);
void Right(int value);
void Left(int value);
void Back(int value);
void CW(int value);
void CCW(int value);
void Standby();
char up =1;
char down =1;
void main(void)
{
    xmegaInit();                //setup XMega
    delayInit();                //setup delay functions
    ADCAINit();                 //setup PORTA analog readings
    lcdInit();                  //setup LCD on PORTK
    MotorInit();                //setup Motor driver and speed
    PORTC_DIR |= 0xFF           //set portC as input
    PORTH_DIR |= 0x03;         //set portH pin1 and pin2 as input
    USARTInit();
    PORTCFG.MPCMASK = 0xFF;
    TWI_MasterInit(&twiMaster,&TWIF,TWI_MASTER_INTLVL_LO_gc,    TWI_BAUDSETTING);//setup
TWI
    delay_ms(1000);
    USARTcmd(1, sendStart);//send start to LCD
    USARTcmd(1, sendClear);//send clear screen cmd to LCD
    adjd_init();                //set up color sensor
    lcdGoto(0,0);
    lcdString("IMDL Final Demo");
    LineDetect();                //Detect line

```



```

while(Final_dev==0) //if all sensor are not read 0
{
    adjd_read();
//    adjd_data();
    Color_recognize();
    LcdColor();
    ObstacleDetect();

    if(up>2) //upper front sensor
    {
        delay_ms(3);
        ObstacleDetect(); //check again
        if(up==4) //Larger obstacle
        {
            Standby();
            delay_ms(3000);
            CW(450);
            delay_ms(2200);
            FlowStatus=0;
        }
    }
    else if(down==3) //smaller obstacle
    {
        Forward(1900);
        delay_ms(500);
    }
    else
    {
        if ((PORTH_IN & 0b00000001) == 0b00000001) //switch
        {
            LineDetect();
            LineFollow();
        }
        else
        {
            Standby();
        }
    }
    delay_ms(1);
}
Standby();

lcdGoto(0,0);
lcdString(" Reach the End ");
lcdGoto(1,0);
lcdString(" GOOD JOB!! ");

}
void ObstacleDetect(void)
{

    int up_left=0,up_mid=0,up_right=0,down_left=0,down_mid=0,down_right=0;
    int i=0;
    while(i<7) //read 7 times

```

```

    {
    if (ADCA0()<2000)
    up_right++;
    if (ADCA1()<2000)
    up_mid++;
    if (ADCA2()<2000)
    up_left++;
    if (ADCA3()<2000)
    down_right++;
    if (ADCA4()<2000)
    down_mid++;
    if (ADCA5()<2000)
    down_left++;

    i++;
    }

    up =1;
    down =1;
    LineDetect();
    if (down_left==7)
        down++;
    if (down_mid==7)
        down++;
    if (down_right==7)
        down++;
    if (up_left==7)
        up++;
    if (up_mid==7)
        up++;
    if (up_right==7)
        up++;
    }
void LineDetect(void)
{
    Rdev =0;
    Ldev =0;
    Final_dev=0;

    if ((PORTC_IN & 0b00111111) == 0b00111111)// all sensor read zero meaning found end
    Final_dev=1;
    if ((PORTC_IN & 0b00000100) == 0b00000100)
        Rdev=1;
    if ((PORTC_IN & 0b00000010) == 0b00000010)
        Rdev=2;
    if ((PORTC_IN & 0b00000001) == 0b00000001)
        Rdev=3;
    if ((PORTC_IN & 0b00100000) == 0b00100000)
        Ldev=1;
    if ((PORTC_IN & 0b00010000) == 0b00010000)
        Ldev=2;
    if ((PORTC_IN & 0b00001000) == 0b00001000)
        Ldev=3;
}

```

```

void LineFollow(void)
{
    if ((PORTC_IN & 0b00111111) != 0b00000000) //found line
        FllowStatus=1;
    if(FllowStatus ==0) // no line was found
    {
        Forward(500);
    }
    else
    {
        if((PORTC_IN & 0b00111111) != 0b00000000)
        {
            if(Rdev>Ldev)
                Right(Rdev);
            if(Rdev<Ldev)
                Left(Ldev);
            if(Rdev==Ldev)
                Forward(500);
        }
        else
        {
            for(i=0,dirL=0;i<MAX;i++)
            {
                if(history[i]==1)
                    dirL++;
            }
            if(dirL>HMAX)
                CCW(300*dirL);
            else
                CW(300*(3-dirL));
        }
    }
}

void MotorInit(void)
{
    ServoDInit(2000); //setup PORTD PWM signal
    PORTJ_DIR = 0xff; //set portJ as output
}

void Forward(int value)
{
    ServoD0(value*0.982); //two motor had a little diffenent speed
    ServoD1(value*1.018);
    PORTJ_OUT = 0b00010101;
}

void Right(int value)
{
    value=4-value;
    ServoD0(800);
    ServoD1(50+(value-1)*300);
    PORTJ_OUT = 0b00010101;
    hcount=(hcount+1)%MAX;
    history[hcount]=0;
}

```

```

}
void Left(int value)
{
    value=4-value;
    ServoD0(50+(value-1)*300);
    ServoD1(800);
    PORTJ_OUT = 0b00010101;
    hcount=(hcount+1)%MAX;
    history[hcount]=1;
}
void CW(int value)
{
    ServoD0(value);
    ServoD1(value);
    PORTJ_OUT = 0b00010110;
}
void CCW(int value)
{
    ServoD0(value);
    ServoD1(value);
    PORTJ_OUT = 0b00001101;
}
void Back(int value)
{
    ServoD0(value);
    ServoD1(value);
    PORTJ_OUT = 0b00001110;
}
void Standby()
{
    ServoD0(200);
    ServoD1(200);
    PORTJ_OUT = 0b00000000;
}

ISR(USARTE1_RXC_vect)
{
    USART_RXComplete(&USART_data);
}
ISR(USARTE1_DRE_vect)
{
    USART_DataRegEmpty(&USART_data);
}
/*! TWIF Master Interrupt vector. */
ISR(TWIF_TWIM_vect)
{
    TWI_MasterInterruptHandler(&twiMaster);
}
void USARTInit(void)
{
    //setting USARTE1
    /* PE7 (TXD0) as output. */
    PORTE.DIRSET = PIN7_bm;
    /* PE6 (RXD0) as input. */

```

```

    PORTE.DIRCLR = PIN6_bm;
    /* Use USARTE1 and initialize buffers. */
    USART_InterruptDriver_Initialize(&USART_data, &USART, USART_DREINTLVL_LO_gc);
    /* USARTC0, 8 Data bits, No Parity, 1 Stop bit. */
    USART_Format_Set(USART_data.usart, USART_CHSIZE_8BIT_gc,
USART_PMODE_DISABLED_gc, false);
    /* Enable RXC interrupt. */
    USART_RxdInterruptLevel_Set(USART_data.usart, USART_RXCINTLVL_LO_gc);
//    Set Baudrate to 9600 bps:
    USART_Baudrate_Set(&USART, 1659, -3);
//    USART_Baudrate_Set(&USART, 207, 0);
    /* Enable both RX and TX. */
    USART_Rx_Enable(USART_data.usart);
    USART_Tx_Enable(USART_data.usart);

    PMIC_CTRL |= PMIC_LOLVLEN_bm;
    sei();
}
void USARTcmd_line(uint8_t NUM, uint8_t sendData[]) //follow line and chang color in the same
time
{
    uint8_t i=0;
    while(1)
    {
        uint8_t GG = 0;
        while (GG < NUM) {
            LineDetect();
            LineFollow();
            bool byteToBuffer;
            byteToBuffer = USART_TXBuffer_PutByte(&USART_data, sendData[GG]);
            if(byteToBuffer){
                GG++;
            }

        }
        uint8_t RRR =0;
        uint8_t receiveData;
        while (RRR < 1)
        {
            if (USART_RXBufferData_Available(&USART_data))
            {
                receiveData = USART_RXBuffer_GetByte(&USART_data);
                RRR++;
            }
        }
        ObstacleDetect();

        if(up>2)
        {
            delay_ms(3);
            ObstacleDetect();
            if(up==4)
            {
                Standby();
                delay_ms(3000);
            }
        }
    }
}

```

```

        CW(450);
        delay_ms(2200);
        FllowStatus=0;
    }
}
else if(down==3)
{
    Forward(1900);
    delay_ms(500);
}
else
{
    if ((PORTH_IN & 0b00000001) == 0b00000001)
    {
        LineDetect();
        LineFollow();
    }
    else
    {
        Standby();
    }
}
}
}
i++;

if (receiveData == 0x06 | i>20)
break;
}
}
void USARTcmd(uint8_t NUM, uint8_t sendData[])
{
    while(1)
    {
        uint8_t GG = 0;
        while (GG < NUM) {
            bool byteToBuffer;
            byteToBuffer = USART_TXBuffer_PutByte(&USART_data, sendData[GG]);
            if(byteToBuffer){
                GG++;
            }
        }
        uint8_t RRR =0;
        uint8_t receiveData;
        while (RRR < 1)
        {
            if (USART_RXBufferData_Available(&USART_data))
            {
                receiveData = USART_RXBuffer_GetByte(&USART_data);
                RRR++;
            }
        }
    }
    if (receiveData == 0x06)
    break;
}
}

```

```

}
void adjd_init(void)
{
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&CAP_RED[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&CAP_GREEN[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&CAP_BLUE[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&CAP_CLEAR[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&INT_RED_LO[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&INT_RED_HI[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&INT_GREEN_LO[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&INT_GREEN_HI[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&INT_BLUE_LO[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&INT_BLUE_HI[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&INT_CLEAR_LO[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&INT_CLEAR_HI[0],2);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
}
void adjd_read(void)
{

```

```

uint8_t setup[2] = {0x00, 0x01};
TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,&setup[0],2); //setsensor read
while (twiMaster.status != TWIM_STATUS_READY) {
    /* Wait until transaction is complete. */
}
uint8_t i=0;
uint8_t response,green,blue;

while(1)
{
    TWI_MasterWrite(&twiMaster,ADJD_ADDRESS,0x00,1);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    TWI_MasterRead(&twiMaster,ADJD_ADDRESS,1);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    response = twiMaster.readData[0];
    if(response == 0) break;
    i++;
}
uint8_t DATA_RED_LO = 0x40;
uint8_t DATA_RED_HI = 0x41;
uint8_t DATA_GREEN_LO = 0x42;
uint8_t DATA_GREEN_HI = 0x43;
uint8_t DATA_BLUE_LO = 0x44;
uint8_t DATA_BLUE_HI = 0x45;
uint8_t DATA_CLEAR_LO = 0x46;
uint8_t DATA_CLEAR_HI = 0x47;
TWI_MasterWriteRead(&twiMaster,ADJD_ADDRESS,&DATA_RED_HI,1,1);
while (twiMaster.status != TWIM_STATUS_READY) {
    /* Wait until transaction is complete. */
}
rh = twiMaster.readData[0];
TWI_MasterWriteRead(&twiMaster,ADJD_ADDRESS,&DATA_RED_LO,1,1);
while (twiMaster.status != TWIM_STATUS_READY) {
    /* Wait until transaction is complete. */
}
rl = twiMaster.readData[0];
TWI_MasterWriteRead(&twiMaster,ADJD_ADDRESS,&DATA_GREEN_HI,1,1);
while (twiMaster.status != TWIM_STATUS_READY) {
    /* Wait until transaction is complete. */
}
gh = twiMaster.readData[0];
TWI_MasterWriteRead(&twiMaster,ADJD_ADDRESS,&DATA_GREEN_LO,1,1);
while (twiMaster.status != TWIM_STATUS_READY) {
    /* Wait until transaction is complete. */
}
gl = twiMaster.readData[0];
TWI_MasterWriteRead(&twiMaster,ADJD_ADDRESS,&DATA_BLUE_HI,1,1);
while (twiMaster.status != TWIM_STATUS_READY) {
    /* Wait until transaction is complete. */
}
}

```



```

    bh = twiMaster.readData[0];
    TWI_MasterWriteRead(&twiMaster,ADJD_ADDRESS,&DATA_BLUE_LO,1,1);
    while (twiMaster.status != TWIM_STATUS_READY) {
        /* Wait until transaction is complete. */
    }
    bl = twiMaster.readData[0];
}
void adjd_data(void)
{
    char rl_data[3];
    sprintf(rl_data,"%02x",rl);
    lcdGoto(1,0);
    lcdString("rl=");
    lcdGoto(1,3);
    lcdString(rl_data) ;
    char gl_data[3];
    sprintf(gl_data,"%02x",gl);
    lcdGoto(1,5);
    lcdString("gl=");
    lcdGoto(1,8);
    lcdString(gl_data) ;
    char bl_data[3];
    sprintf(bl_data,"%02x",bl);
    lcdGoto(1,10);
    lcdString("bl=");
    lcdGoto(1,13);
    lcdString(bl_data) ;
}
void Color_recognize(void)
{
    lcdGoto(1,7);
    if((rl>0x82 && rl<0xA4)&&(gl>0x39 && gl<0x4E)&&(bl>0x1E && bl<0x2F))
    {
        lcdString("Red ");
        ColorFlag =1;
    }
    if((rl>0x85 && rl<0xA7)&&(gl>0x51 && gl<0x67)&&(bl>0x1E && bl<0x2E))
    {
        lcdString("Orange ");
        ColorFlag =2;
    }
    if((rl>0x93 && rl<0xB0)&&(gl>0x8D && gl<0xAA)&&(bl>0x25 && bl<0x37))
    {
        lcdString("Yellow ");
        ColorFlag =3;
    }
    if((rl>0x30 && rl<0x45)&&(gl>0x4D && gl<0x60)&&(bl>0x17 && bl<0x25))
    {
        lcdString("Green ");
        ColorFlag =4;
    }
    if((rl>0x2E && rl<0x40)&&(gl>0x3C && gl<0x50)&&(bl>0x45 && bl<0x59))
    {
        lcdString("Blue ");
    }
}

```

```

    ColorFlag =5;
}
if((r1>0x56 && r1<0x72)&&(g1>0x72 && g1<0x90)&&(b1>0x6C && b1<0x85))
{
    lcdString("Cyan ");
    ColorFlag =6;
}
if((r1>0x77 && r1<0x83)&&(g1>0x4C && g1<0x63)&&(b1>0x37 && b1<0x4F))
{
    lcdString("Purple ");
    ColorFlag =7;
}
if((r1>0x02 && r1<0x0F)&&(g1>0x02 && g1<0x0F)&&(b1>0x00 && b1<0x0B))
{
    lcdString("Black ");
}
}
void LcdColor(void)
{
    if(ColorFlag!=LcdFlag)
    {
        LcdFlag = ColorFlag;
        if(LcdFlag ==1)
        {
            Standby();
            USARTcmd(3, sendRedBG);
            CCW(700);
            delay_ms(1300);
            FlowStatus=0;
            LcdFlag = 0;
        }
        if(LcdFlag ==2)
        {
            Standby();
            USARTcmd_line(3, sendOrangeBG);
        }
        if(LcdFlag ==3)
        {
            Standby();
            USARTcmd_line(3, sendYellowBG);
        }
        if(LcdFlag ==4)
        {
            Standby();
            USARTcmd_line(3, sendGreenBG);
        }
        if(LcdFlag ==5)
        {
            Standby();
            USARTcmd_line(3, sendBlueBG);
        }
        if(LcdFlag ==6)
        {
            Standby();
        }
    }
}

```

```
        USARTcmd_line(3, sendCyanBG);
    }
    if(LcdFlag ==7)
    {
        Standby();
        USARTcmd_line(3, sendPurpleBG);
    }
}
}
```