Jiaming Zhang

2：

**void** RGBtoHSV**(unsigned** **char** r**,** **unsigned** **char** g**,** **unsigned** **char** b**,** **unsigned** **char** **\***h**,** **unsigned** **char** **\***s**,** **unsigned** **char** **\***v**);**

**{**

**float** max**;**

**float** mid**;**

**float** min**;**

**float** r\_unit**;**

**float** g\_unit**;**

**float** b\_unit**;**

**float** h\_unit**;**

**float** s\_unit**;**

**float** v\_unit**;**

r\_unit **=** r**/**255.0**;**

g\_unit **=** g**/**255.0**;**

b\_unit **=** b**/**255.0**;**

**if(**r\_unit**>**g\_unit**)**

**{**

**if(**g\_unit**>**b\_unit**)**

**{**

max **=** **(float)**r\_unit**;**

mid **=** **(float)**g\_unit**;**

min **=** **(float)**b\_unit**;**

**}**

**else**

**{**

**if(**b\_unit**>**r\_unit**)**

**{**

max **=** **(float)**b\_unit**;**

mid **=** **(float)**r\_unit**;**

min **=** **(float)**g\_unit**;**

**}**

**else**

**{**

max **=** **(float)**r\_unit**;**

mid **=** **(float)**b\_unit**;**

min **=** **(float)**g\_unit**;**

**}**

**}**

**}**

**else**

**{**

**if(**g\_unit**<**b\_unit**)**

**{**

max **=** **(float)**b\_unit**;**

mid **=** **(float)**g\_unit**;**

min **=** **(float)**r\_unit**;**

**}**

**else**

**{**

**if(**b\_unit**<**r\_unit**)**

**{**

max **=** **(float)**g\_unit**;**

mid **=** **(float)**r\_unit**;**

min **=** **(float)**b\_unit**;**

**}**

**else**

**{**

max **=** **(float)**g\_unit**;**

mid **=** **(float)**b\_unit**;**

min **=** **(float)**r\_unit**;**

**}**

**}**

**}**

**if** **(**max**==**min**)**

**{**

h\_unit **=** **(unsigned** **char)**0**;**

**}**

**else** **if(**max**==**r\_unit **&&** g\_unit**>=**b\_unit**)**

**{**

h\_unit **=** **(unsigned** **char)(**60**\*(**g\_unit **-** b\_unit**)/(**max **-** min**));**

**}**

**else** **if(**max**==**r\_unit **&&** g**<**b**)**

**{**

h\_unit **=** **(unsigned** **char)(**60**\*(**g\_unit **-** b\_unit**)/(**max **-** min**)** **+** 360**);**

**}**

**else** **if(**max**==**g\_unit**)**

**{**

h\_unit **=** **(unsigned** **char)(**60**\*(**b\_unit **-** r\_unit**)/(**max **-** min**)** **+** 120**);**

**}**

**else** **if(**max**==**b\_unit**)**

**{**

h\_unit **=** **(unsigned** **char)(**60**\*(**g\_unit **-** b\_unit**)/(**max **-** min**)** **+** 240**);**

**}**

**if(**max**==**0**)**

**{**

s\_unit **=** **(unsigned** **char)**0**;**

**}**

**else**

**{**

s\_unit **=** **(unsigned** **char)(**1**-**min**/**max**);**

**}**

v\_unit **=** **(unsigned** **char)**max**;**

**\***h **=** h\_unit**\***255**/**360**;**

**\***s **=** s\_unit**\***255**;**

**\***v **=** v\_unit**\***255**;**

**}**

3：

A．



B.



C.



4.:

A.



B.



C.



i.







when





ii.















when



Y is positive, so it is on the left.

5.

#include "msp430g2553.h"

#include "UART.h"

**void** print\_every**(int** rate**);**

**char** newprint **=** 0**;**

**long** NumOn **=** 0**;**

**long** NumOff **=** 0**;**

**int** statevar **=** 1**;**

**int** timecheck1 **=** 0**;**

**int** timecheck2 **=** 0**;**

**int** state **=** 0**;**

**char** count2\_6 **=** 0**;**

**char** count2\_7 **=** 0**;**

**char** flag2\_6 **=** 0**;**

**char** flag2\_7 **=** 0**;**

**void** main**(void)** **{**

WDTCTL **=** WDTPW **+** WDTHOLD**;** *// Stop watchdog timer*

**if** **(**CALBC1\_16MHZ **==**0xFF **||** CALDCO\_16MHZ **==** 0xFF**)** **while(**1**);**

DCOCTL **=** CALDCO\_16MHZ**;** *// Set uC to run at approximately 16 Mhz*

BCSCTL1 **=** CALBC1\_16MHZ**;**

*// Initialize Port 1*

P1SEL **&=** **~**0x01**;** *//Set P1.0 GPIO*

P1SEL2 **&=** **~**0x01**;**

P2SEL **&=** **~**0xc0**;** *//Set P2.6 P2.7 GPIO*

P2SEL2 **&=** **~**0xc0**;**

P1REN **=** 0x0**;** *// No resistors enabled for Port 1*

P1DIR **|=** 0x10**;** *// Set P1.4 to output*

P2DIR **|=** 0x04**;** *// Set P2.2 to output*

P2DIR **&=** **~**0xc0**;** *//Set P2.6 and P2.7 to input*

P2REN **|=** 0xc0**;** *// P2.6 and P2.7 Resistor enabled*

P2OUT **|=** 0xc0**;** *// P2.6 and P2.7 Pullup Resistor selected*

*// Port 2 Interrupts*

P2IE **|=** 0xc0**;** *// P2.6 and P2.7 interrupt enabled*

P2IES **|=** 0xc0**;** *// P2.6 and P2.7 H/L edge*

P2IFG **&=** **~**0xc0**;** *// P2.6 and P2.7 IFG cleared*

*// Timer A Config*

TACCTL0 **=** CCIE**;** *// Enable Periodic interrupt*

TACCR0 **=** 16000**;** *// period = 1ms*

TACTL **=** TASSEL\_2 **+** MC\_1**;** *// source SMCLK, up mode*

Init\_UART**(**115200**,**1**);** *// Initialize UART for 115200 baud serial communication*

\_BIS\_SR**(**GIE**);** *// Enable global interrupt*

**while(**1**)** **{** *// Low priority Slow computation items go inside this while loop. Very few (if anyt) items in the HWs will go inside this while loop*

*// for use if you want to use a method of receiving a string of chars over the UART see USCI0RX\_ISR below*

*// if(newmsg) {*

*// newmsg = 0;*

*// }*

*// The newprint variable is set to 1 inside the function "print\_every(rate)" at the given rate*

**if** **(** **(**newprint **==** 1**)** **&&** **(**senddone **==** 1**)** **)** **{** *// senddone is set to 1 after UART transmission is complete*

*// only one UART\_printf can be called every 15ms*

UART\_printf**("count2\_6: %d count2\_7: %d\n\r",**count2\_6**,**count2\_7**);**

newprint **=** 0**;**

**}**

**}**

**}**

#pragma vector=PORT2\_VECTOR

\_\_interrupt **void** Port\_2**(void){**

**if((**P2IFG **&** 0x40**)** **==** 0x40**)** **{**

P1OUT **^=** 0x10**;** *// P1.4 toggled*

P2IFG **&=** **~**0x40**;** *// Clear P2.6 interrupt bit*

P2IE **&=** **~**0x40**;** *// Disable P2.6 interrupt*

flag2\_6 **=** 1**;** *// P2.6 interrupt has been disabled*

count2\_6**++;**

**}**

**if((**P2IFG **&** 0x80**)** **==** 0x80**)** **{**

P2OUT **^=** 0x04**;** *// P2.2 toggled*

P2IFG **&=** **~**0x80**;** *// Clear P2.7 interrupt bit*

P2IE **&=** **~**0x80**;** *// Disable P2.7 interrupt*

flag2\_7 **=** 1**;** *// P2.7 interrupt has been disabled*

count2\_7**++;**

**}**

**}**

*// Timer A0 interrupt service routine*

#pragma vector=TIMER0\_A0\_VECTOR

\_\_interrupt **void** Timer\_A **(void)**

**{**

print\_every**(**500**);** *//print in TeraTerm*

**if** **(**flag2\_6 **==** 1**){**

timecheck1**++;**

**if** **(**timecheck1 **==** 300**){**

timecheck1 **=** 0**;**

P2IFG **&=** **~**0x40**;** *// Clear P2.6 interrupt bit*

P2IE **|=** 0x40**;** *// Enable P2.6 interrupt*

flag2\_6 **=** 0**;**

**}**

**}**

**if** **(**flag2\_7 **==** 1**){**

timecheck2**++;**

**if** **(**timecheck2 **==** 300**){**

timecheck2 **=** 0**;**

P2IFG **&=** **~**0x80**;** *// Clear P2.7 interrupt bit*

P2IE **|=** 0x80**;** *// Enable P2.7 interrupt*

flag2\_7 **=** 0**;**

**}**

**}**

**}**

*/\**

*// ADC 10 ISR - Called when a sequence of conversions (A7-A0) have completed*

*#pragma vector=ADC10\_VECTOR*

*\_\_interrupt void ADC10\_ISR(void) {*

*}*

*\*/*

*// USCI Transmit ISR - Called when TXBUF is empty (ready to accept another character)*

#pragma vector=USCIAB0TX\_VECTOR

\_\_interrupt **void** USCI0TX\_ISR**(void)** **{**

**if(**IFG2**&**UCA0TXIFG**)** **{** *// USCI\_A0 requested TX interrupt*

**if(**printf\_flag**)** **{**

**if** **(**currentindex **==** txcount**)** **{**

senddone **=** 1**;**

printf\_flag **=** 0**;**

IFG2 **&=** **~**UCA0TXIFG**;**

**}** **else** **{**

UCA0TXBUF **=** printbuff**[**currentindex**];**

currentindex**++;**

**}**

**}** **else** **if(**UART\_flag**)** **{**

**if(!**donesending**)** **{**

UCA0TXBUF **=** txbuff**[**txindex**];**

**if(**txbuff**[**txindex**]** **==** 255**)** **{**

donesending **=** 1**;**

txindex **=** 0**;**

**}**

**else** txindex**++;**

**}**

**}** **else** **{** *// interrupt after sendchar call so just set senddone flag since only one char is sent*

senddone **=** 1**;**

**}**

IFG2 **&=** **~**UCA0TXIFG**;**

**}**

**if(**IFG2**&**UCB0TXIFG**)** **{** *// USCI\_B0 requested TX interrupt (UCB0TXBUF is empty)*

IFG2 **&=** **~**UCB0TXIFG**;** *// clear IFG*

**}**

**}**

*// USCI Receive ISR - Called when shift register has been transferred to RXBUF*

*// Indicates completion of TX/RX operation*

#pragma vector=USCIAB0RX\_VECTOR

\_\_interrupt **void** USCI0RX\_ISR**(void)** **{**

**if(**IFG2**&**UCB0RXIFG**)** **{** *// USCI\_B0 requested RX interrupt (UCB0RXBUF is full)*

IFG2 **&=** **~**UCB0RXIFG**;** *// clear IFG*

**}**

**if(**IFG2**&**UCA0RXIFG**)** **{** *// USCI\_A0 requested RX interrupt (UCA0RXBUF is full)*

*// Uncomment this block of code if you would like to use this COM protocol that uses 253 as STARTCHAR and 255 as STOPCHAR*

*/\* if(!started) { // Haven't started a message yet*

*if(UCA0RXBUF == 253) {*

*started = 1;*

*newmsg = 0;*

*}*

*}*

*else { // In process of receiving a message*

*if((UCA0RXBUF != 255) && (msgindex < (MAX\_NUM\_FLOATS\*5))) {*

*rxbuff[msgindex] = UCA0RXBUF;*

*msgindex++;*

*} else { // Stop char received or too much data received*

*if(UCA0RXBUF == 255) { // Message completed*

*newmsg = 1;*

*rxbuff[msgindex] = 255; // "Null"-terminate the array*

*}*

*started = 0;*

*msgindex = 0;*

*}*

*}*

*\*/*

IFG2 **&=** **~**UCA0RXIFG**;**

**}**

**}**

*// This function takes care of all the timing for printing to UART*

*// Rate determined by how often the function is called in Timer ISR*

**int** print\_timecheck **=** 0**;**

**void** print\_every**(int** rate**)** **{**

**if** **(**rate **<** 15**)** **{**

rate **=** 15**;**

**}**

**if** **(**rate **>** 10000**)** **{**

rate **=** 10000**;**

**}**

print\_timecheck**++;**

**if** **(**print\_timecheck **==** rate**)** **{**

print\_timecheck **=** 0**;**

newprint **=** 1**;**

**}**

**}**

6

Solder two resistors as R\_B. Connect the two input to P2.0 and P2.1. Two TIP122 chips are soldered at the right side on the board with three pins B C E. For figure1, connect COLLECTOR to negative pole of ultra-bright LED, and add a 47Ω resistor between positive pole of ultra-bright LED and +5V source. For figure2, connect COLLECTOR to Pin6 of DIP05-1A57 and also positive pole of 1N4003. The negative pole of 1N4003 is connected to Pin2 and +5V source. Connect pin14 to the 5V source. Add a 470Ω resistor between Pin 8 and positive pole of standard LED. The negative pole of standard LED is connected to ground

7

**switch(**statevar**)**

**{**

**case** 1**:** *//move to goal*

**if** **(**no\_pink\_is\_seen**)**

**{**

move\_towards\_x**,**y**;**

**if(**sensor**[**113**]** **<** set\_value**)**

**{**

statevar **=** 2**;**

**}**

**if(**sensor on the right **<** set\_value**)**

**{**

statevar **=** 3**;**

**}**

**if(**sensor on the left **<** set\_value**)**

**{**

statevar **=** 4**;**

**}**

**if((**sensor**[**112**]<=**set\_value**)&&(**sensor**[**112**]>**set\_value**)&&(**sensor**[**114**]<=**set\_value**))**

**{**

statevar **=** 5**;**

**}**

**}**

**else**

**{**

statevar **=** 6**;**

**}**

**break;**

**case** 2**:** *// front obstacle avoidance*

**if** **(**no\_pink\_is\_seen**)**

**{**

left **or** right\_wall\_follows**;**

**if(**front\_distance **>** set\_value**)**

**{**

statevar **=** 1**;**

**}**

**}**

**else**

**{**

statevar **=** 3**;**

**}**

**break;**

**case** 3**:** *// right obstacle avoidance*

**if** **(**no\_pink\_is\_seen**)**

**{**

right\_wall\_follows**;**

**if(**front\_distance **>** set\_value**)**

**{**

statevar **=** 1**;**

**}**

**}**

**else**

**{**

statevar **=** 3**;**

**}**

**break;**

**case** 4**:** *// left obstacle avoidance*

**if** **(**no\_pink\_is\_seen**)**

**{**

left\_wall\_follows**;**

**if(**front\_distance **>** set\_value**)**

**{**

statevar **=** 1**;**

**}**

**}**

**else**

**{**

statevar **=** 3**;**

**}**

**break;**

**case** 5**:** *// find legs*

brake the car**;**

**if** **(**no obstacle is found**)**

**{**

statevar **=** 1**;**

**}**

**break;**

**case** 6**:** *//move to pink*

move\_towards\_pink**;**

**if** **(**can't\_see\_pink)

**{**

statevar **=** 1**;**

**}**

**break;**

**}**