Lab Report 2 - Taylor Rainwater, Triston Luzanta

**Objectives**

The objective of this lab assignment is to create a simulated game called “Nim”. To play this game, two players take turns removing objects from rows of objects. The objects however, need to be removed from the same row. The player who is left with the last object loses the game. The game should also have an anti-cheating component, so that once a player selects a row they cannot select a different row until it is their turn again. To accomplish this, we will design the hardware of the game, and then program the functionality of the game into the hardware.

**Hardware Design**

To begin, we had to interface twelve LEDs (four of each color, blue green and red) to represent the rows of objects in the game. We also had to interface six switches (two per row) to represent the players in the game. Below are calculations to ensure that the LEDs needed for this project did not exceed the 80mA limit of our power source.

Current Calculations:

PCBV: 3.3V

Resistor (R): 220

LEDV: Data sheet

Row 1 (Blue LED) : Forward Voltage drop 3.2V

\* 4 = 1.81 mA

Row 2 (Green LED): Forward Voltage drop 2.2V

\* 4 = 20 mA

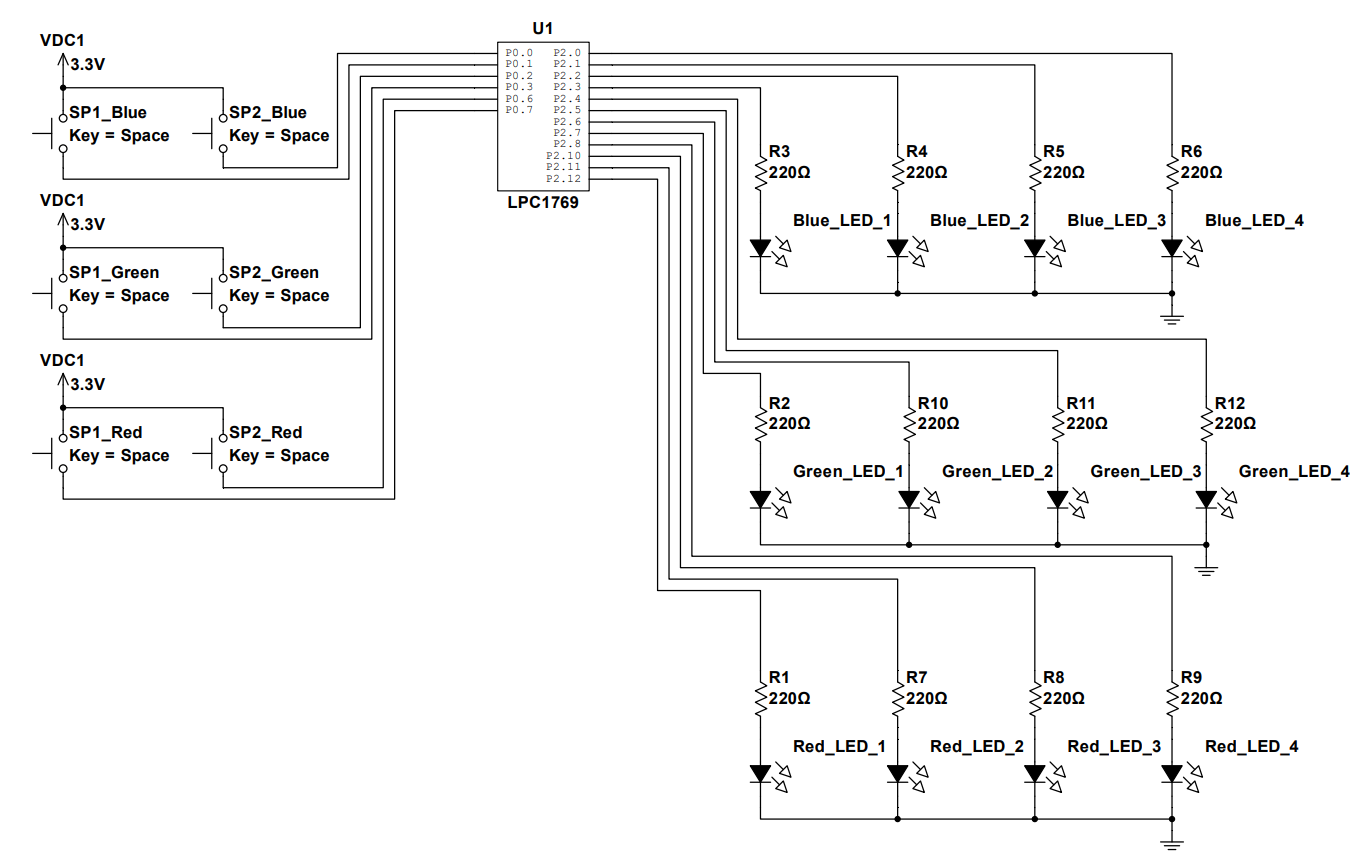
Row 3 (Red LED): Forward Voltage drop 2V

\* 4 = 23.63 mA

Total current: 45.44 mA < 80 mA

220ohm resistors were used to limit the current on the LEDs. This value was estimated as a good standard for our LEDs, and calculations confirmed this.

**Hardware Implementation**

Below is a schematic of how the LEDs and switches were interfaced with the LPC1769. Port 2 was chosen to interface the LEDs and port 0 was chosen to interface the switches. These ports were chosen because they had the correct number of pins needed for each part of this project relatively in a row on the board, simplifying the software implementation.

**Source Code**

Below is the source code for our software implementation with comments.

/\*

===============================================================================

Name : Assignment\_2.c

Author : Taylor Rainwater and Triston Luzanta

Version :

Copyright : $(copyright)

Description : This project simulates a game called “Nim”. The game consists of two players taking turns removing objects from different rows. The objective of the game is to leave the player with the last object. In this code. In this code we are implementing the functionality of the game into our hardware design.

===============================================================================

\*/

#ifdef \_\_USE\_CMSIS

#include "LPC17xx.h"

#endif

#include <cr\_section\_macros.h>

#include <stdio.h>

// Defining registers for lights and buttons(Port 0 and Port 2)

#define FIO0DIR (\*(volatile unsigned int\*) 0x2009c000)

#define FIO2DIR (\*(volatile unsigned int\*) 0x2009c040)

#define FIO0PIN (\*(volatile unsigned int\*) 0x2009c014)

#define FIO2PIN (\*(volatile unsigned int\*) 0x2009c054)

#define PINMODE0 (\*(volatile unsigned int\*) 0x4002C040)

// Variables to track how many lights are on in each row

int bluOn;

int grnOn;

int redOn;

// Variables to disable/enable each button

int blu1;

int blu2;

int grn1;

int grn2;

int red1;

int red2;

// Wait function, uses seconds as argument instead of ticks

void wait(float sec)

{

volatile int ticks = (int)((sec - 0.00000843) / 0.00000237);

volatile int count;

for (count=0; count<ticks; count++){

}

}

// Function to set any light to on or off, pin argument is the bit number in register 2 corresponding to a light

void setPin(int pin, int val){

if(val == 1){

FIO2PIN |= (1<<pin);

}

else if(val == 0){

FIO2PIN &= ~(1<<pin);

}

}

// Decreases the number of blue lights that are on by 1

void decBlu(){

switch (bluOn)

{

case 4:

FIO2PIN &= ~(1<<0);

break;

case 3:

FIO2PIN &= ~(1<<1);

break;

case 2:

FIO2PIN &= ~(1<<2);

break;

case 1:

FIO2PIN &= ~(1<<3);

break;

default:

break;

}

bluOn--;

}

// Decreases the number of green lights that are on by 1

void decGrn(){

switch (grnOn)

{

case 4:

FIO2PIN &= ~(1<<4);

break;

case 3:

FIO2PIN &= ~(1<<5);

break;

case 2:

FIO2PIN &= ~(1<<6);

break;

case 1:

FIO2PIN &= ~(1<<7);

break;

default:

break;

}

grnOn--;

}

// Decreases the number of red lights that are on by 1

void decRed(){

switch (redOn)

{

case 4:

FIO2PIN &= ~(1<<8);

break;

case 3:

FIO2PIN &= ~(1<<10);

break;

case 2:

FIO2PIN &= ~(1<<11);

break;

case 1:

FIO2PIN &= ~(1<<12);

break;

default:

break;

}

redOn--;

}

// Returns the current state of the button indicated by the parameter

int readSwitches(int val)

{

return (FIO0PIN>>val) & 1;

}

// Driver that executes the game

int main(void) {

// Initializes outputs on register 2 for lights

for(int i = 0; i < 13; i++){

if(i != 9){

FIO2DIR |= (1<<i);

}

}

// Initializes inputs on register 0 for buttons

for(int i = 0; i < 8; i++){

if(i != 4 && i !=5 ){

FIO0DIR &= ~(1<<i);

PINMODE0 |= (1<<(i\*2)) | (1<<((i\*2)+1));

}

}

//Game loop

while(1){

// Always put this at the start of the game loop so all the lights are turned on

for(int i = 0; i < 13; i++){

if(i != 9){

setPin(i, 1);

}

}

// Always put these at the start of the game loop so light on count is accurate

bluOn = 4;

grnOn = 4;

redOn = 4;

// Prevents cheating in other words players can only press one button at a time

blu1 = 1;

blu2 = 1;

grn1 = 1;

grn2 = 1;

red1 = 1;

red2 = 1;

//This loop checks if a button was pressed by each player. The loop also forces the player to press one button.

while(bluOn > 0 || grnOn > 0 || redOn > 0){

if(readSwitches(2) && blu1){

grn1 = 0;

red1 = 0;

red2 = 1;

grn2 = 1;

blu2 = 1;

decBlu();

wait(0.2);

}

else if(readSwitches(3) && blu2){

grn2 = 0;

red2 = 0;

red1 = 1;

grn1 = 1;

blu1 = 1;

decBlu();

wait(0.2);

}

else if(readSwitches(6) && grn1){

blu1 = 0;

red1 = 0;

red2 = 1;

grn2 = 1;

blu2 = 1;

decGrn();

wait(0.2);

}

else if(readSwitches(7) && grn2){

blu2 = 0;

red2 = 0;

red1 = 1;

grn1 = 1;

blu1 = 1;

decGrn();

wait(0.2);

}

else if(readSwitches(1) && red1){

blu1 = 0;

grn1 = 0;

red2 = 1;

grn2 = 1;

blu2 = 1;

decRed();

wait(0.2);

}

else if(readSwitches(0) && red2){

blu2 = 0;

grn2 = 0;

red1 = 1;

grn1 = 1;

blu1 = 1;

decRed();

wait(0.2);

}

else{

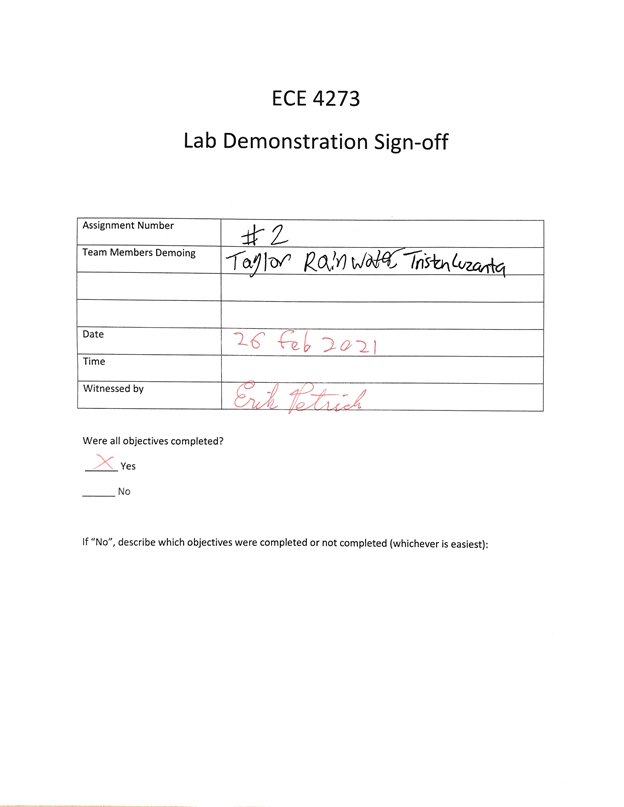
}

}

wait(2); // After the game is finished, wait 2 seconds for another round to be set

}

}

****