작성(개발)자 : 조상호

작성(개발) 내용 : 숫자야구 프로그램.

키패드를 통하여 숫자 입력 구현

LED를 통하여 스트라이크 / 볼 수 표시

LCD를 통하여 입력된 수 / 스트라이크, 볼 / 게임 성공여부 확인

입력 모듈

4X4 키패드

출력 모듈

LCD (1602A, I2C)

LED 5개

```
#include <iostream>
#include <wiringPi.h>
#include <stdlib.h>
#include <time.h>

#include "Baseball.h"
#include "I2C_LCD.h"
#include <mqueue.h>


mqd_t mfd;

struct mq_attr attr = {
      .mq_maxmsg = 10,
      .mq_msgsize = 4,
};




PI_THREAD(BTN_MAP)
{
      int iPin_Col[] = {25,24,23,22};
      int iPin_Row[] = {26,27,28,29};

      int iMAPPING[] =
      {
            1,2,3,12,
            4,5,6,12,
            7,8,9,12,
            12,0,20,30
      };

      for(int i = 0 ; i < 4; i++)
      {
            pinMode(iPin_Col[i], OUTPUT);
            pinMode(iPin_Row[i], INPUT);
            digitalWrite(iPin_Col[i], LOW);
      }

      int iColSelect = 0;

      while(1)
      {
            digitalWrite(iPin_Col[iColSelect], HIGH);
            int iKeyMap[4]={0,};
            for(int i = 0; i<4; i++)
            {
                  iKeyMap[i] =  digitalRead (iPin_Row[i]);
                  int iTemp =  i+(iColSelect*4);
                  if(iKeyMap[i] == HIGH)
                  {
                        //메시지큐에 입력
                        printf("%d₩n", iMAPPING[iTemp]);
                         mq_send(mfd, (char*) &iMAPPING[iTemp], attr.mq_msgsize,1 );
                  }
            }

            delay(65);
            digitalWrite(iPin_Col[iColSelect], LOW);
            iColSelect ++;
```

```
            iColSelect %= 4;
      }
}



void init( int* iPin_Strike, int *iPin_Ball)
{
      wiringPiSetup();


      mq_unlink("/msg_q");
      mfd = mq_open("/msg_q", O_RDWR | O_CREAT, 666, &attr) ;

      if( mfd == 1)
      {
            perror("open error") ;
            exit(-1);
      }

      piThreadCreate(BTN_MAP);


      for(int i = 0 ; i < 2; i++)
      {
            pinMode(iPin_Strike[i], OUTPUT);
      }
      for(int i = 0 ; i < 3; i++)
      {
            pinMode(iPin_Ball[i], OUTPUT);
      }

      srand((unsigned) time(NULL)) ;
}

int inCheck(int num, int input)
{
      for (int i = 0 ; input != 0 ; i++ )
      {
            if((input%10) == num)
                  return 1;
            input /=10;
      }
      return 0;
}


int InputNumber(I2C_LCD& lcd )
{
      int iInput = 0;

      int iMsg = 12;
      lcd.setXY(0,0);
      lcd.print("Input Num : ___");
      lcd.setXY(12,0);

      do
      {
            mq_receive(mfd, (char *) &iMsg, attr.mq_msgsize, NULL) ;
            if( iMsg < 10 && iInput < 100 && (inCheck(iMsg, iInput) == 0))
            {
```

```
123                   lcd.putchar(iMsg + 48);
124                   iInput *= 10;
125                   iInput += iMsg;
126             }
127             else if(iMsg == 20)
128             {
129                   iInput /= 10;
130                   lcd.setXY(12,0);
131                   lcd.print("___ ");
132                         lcd.setXY(12,0);
133
134                   if(iInput!=0)
135                   {
136                         char * cNumberToString = new char[4] ;
137                         sprintf(cNumberToString,"%d",iInput);
138                         lcd.print(cNumberToString);
139                   }
140             }
141       }while( iMsg != 30);
142
143
144       if( iInput < 100 || iInput > 999)
145             return InputNumber(lcd);
146
147       return iInput;
148 }
149
150
151 void DisplaySnB(I2C_LCD& lcd, int iStrike, int iBall)
152 {
153        lcd.clear();
154       lcd.setXY(1,1);
155       lcd.putchar(((char)iStrike)+48);
156       lcd.print(" S");
157       lcd.setXY(8,1);
158       lcd.putchar(((char)iBall)+48);
159       lcd.print(" B");
160 }
161
162 void LedSnB(int* iPin_Strike, int* iPin_Ball , int iStrike, int iBall)
163 {
164
165       for(int i = 0; i< MAX_LENGTH-1; i++)
166       {
167             digitalWrite(iPin_Strike[i], LOW );
168             if( i <= iStrike-1)
169                   digitalWrite(iPin_Strike[i], HIGH );
170       }
171
172       for(int i = 0; i< MAX_LENGTH; i++)
173       {
174             digitalWrite(iPin_Ball[i], LOW );
175             if( i < iBall)
176                   digitalWrite(iPin_Ball[i], HIGH );
177       }
178 }
179
180 void LED_OFF(int* iPin_Strike, int* iPin_Ball)
181 {
182       for(int i = 0; i< MAX_LENGTH-1; i++)
183                   digitalWrite(iPin_Strike[i], LOW );
```

```
      for(int i = 0; i< MAX_LENGTH; i++)
                  digitalWrite(iPin_Ball[i], LOW );

}

int EndCheck(I2C_LCD& lcd , ComNumber cnComputer )
{
       lcd.clear();
      int iDone = 1;
      char cEndCheck;
      do{
            lcd.setXY(0,0);
            lcd.print("Success");
            lcd.setXY(0,1);
            lcd.print("Continue?(1/2)");
            mq_receive(mfd, (char *) &cEndCheck, attr.mq_msgsize, NULL) ;

            if( cEndCheck == 1){
                  cnComputer.Reset();
                  iDone = 1;
            }
            else  if( cEndCheck == 2)
                  iDone = 0;
            lcd.clear();
      }while( !( cEndCheck ==1 || cEndCheck== 2) );

      return iDone;
}



int main()
{

      I2C_LCD lcd;
      lcd.init(0x27);
      lcd.clear();

      int iPin_Strike[] = {4,5};
      int iPin_Ball[] = {0,2,3};

      init(iPin_Strike, iPin_Ball);
      LED_OFF(iPin_Strike, iPin_Ball);

      ComNumber cnComputer = ComNumber() ;
      int iDone = 1;
      printf("%dWn", cnComputer.iGetNumber());
      lcd.print("Input Num : ___");
      while(iDone)
      {
            int iInput = InputNumber(lcd);

            int iStrike =  cnComputer.iStrikeCount(iInput);
            int iBall = cnComputer.iBallCount(iInput);

            if( iStrike != 3)
            {
                DisplaySnB(lcd, iStrike, iBall);
                LedSnB(iPin_Strike, iPin_Ball, iStrike, iBall);
            }
            else
```

```
245             {
246                   LED_OFF (iPin_Strike, iPin_Ball);
247                  iDone = EndCheck(lcd, cnComputer);
248             }
249       }
250 }
251
```

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>

#define LCD_CHR  1 // Mode - Sending data
#define LCD_CMD  0 // Mode - Sending command

#define LINE1  0x80 // 1st line
#define LINE2  0xC0 // 2nd line

#define LCD_BACKLIGHT_ON   0x08  // On
#define LCD_BACKLIGHT_OFF  0x00  # Off

#define ENABLE  0b00000100 // Enable bit

class I2C_LCD
{
private :

      int fd;

      void lcd_byte(int bits, int mode);
      void lcd_toggle_enable(int bits);

      // added by Lewis
      void typeInt(int i);
      void typeFloat(float myFloat);

public:
      I2C_LCD();
      I2C_LCD(int fd);

      void init(int fd);

      void print( const char* str);
      void setXY(int x, int y);
      void clear();
      void putchar(char str);

};
```

```
1
2 #include "I2C_LCD.h"
3
4 I2C_LCD :: I2C_LCD()
5 {
6                   fd = 0;
7 }
8
9 I2C_LCD :: I2C_LCD(int fd)
10 {
11              init(fd);
12 }
13
14 void I2C_LCD::init(int fd)
15 {
16       this->fd = wiringPiI2CSetup(fd);
17       // Initialise display
18       lcd_byte(0x32, LCD_CMD); // Initialise
19       lcd_byte(0x06, LCD_CMD); // Cursor move direction
20       lcd_byte(0x0C, LCD_CMD); // 0x0F On, Blink Off
21       lcd_byte(0x28, LCD_CMD); // Data length, number of lines, font size
22       lcd_byte(0x01, LCD_CMD); // Clear display
23       delayMicroseconds(500);
24 }
25
26 void I2C_LCD::print( const char* str)
27 {
28       while ( *str ) lcd_byte(*(str++), LCD_CHR);
29 }
30
31 void I2C_LCD::putchar(char str)
32 {
33       lcd_byte(str, LCD_CHR);
34 }
35
36
37 void I2C_LCD::setXY(int x, int y)
38 {
39       //0,0
40             lcd_byte((LINE1      | (0x40 * y) )      + x, LCD_CMD);
41 }
42 void I2C_LCD::clear()
43 {
44       lcd_byte(0x01, LCD_CMD);
45       lcd_byte(0x02, LCD_CMD);
46 }
47
48
49 // float to string
50 void      I2C_LCD::typeFloat(float myFloat)       {
51       char buffer[20];
52       sprintf(buffer, "%4.2f",      myFloat);
53       print(buffer);
54 }
55
56 // int to string
57 void      I2C_LCD::typeInt(int i)       {
58       char array1[20];
59       sprintf(array1, "%d",      i);
60       print(array1);
61 }
```

```
62
63 void      I2C_LCD::lcd_byte(int bits, int mode)       {
64
65       //Send byte to data pins
66       // bits = the data
67       // mode = 1 for data, 0 for command
68       int bits_high;
69       int bits_low;
70       // uses the two half byte writes to LCD
71       bits_high = mode | (bits & 0xF0) | LCD_BACKLIGHT_ON ;
72       bits_low = mode | ((bits << 4) & 0xF0) | LCD_BACKLIGHT_ON ;
73
74       // High bits
75       wiringPiI2CReadReg8(fd, bits_high);
76       lcd_toggle_enable(bits_high);
77
78       // Low bits
79       wiringPiI2CReadReg8(fd, bits_low);
80       lcd_toggle_enable(bits_low);
81 }
82
83 void      I2C_LCD::lcd_toggle_enable(int bits)       {
84       // Toggle enable pin on LCD display
85       delayMicroseconds(500);
86       wiringPiI2CReadReg8(fd, (bits | ENABLE));
87       delayMicroseconds(500);
88       wiringPiI2CReadReg8(fd, (bits & ~ENABLE));
89       delayMicroseconds(500);
90 }
91
92
```

```cpp
#include <iostream>
#include <wiringPi.h>
#include <stdlib.h>
#include <time.h>

#define MAX_LENGTH           3

using namespace std;

class ComNumber
{
      private :
            int* iNumberArray;
      public :
            int* iCreateNumber();
            ComNumber();
            int iBallCount(int iUserInput);
            int iStrikeCount(int iUserInput);
            int iGetNumber();
            int* SplitNumber(int iNumber);
            void Reset();

};

```

```cpp
#include "Baseball.h"

ComNumber::ComNumber()
{
	iNumberArray = iCreateNumber();
}

int* ComNumber::iCreateNumber()
{
	int iNum =0;
	int* iArray= new int[MAX_LENGTH];
	int   iDone = 0;

	do{
		iDone = 0;
		iNum = rand() %900 +100;

		for ( int i = 0; i < MAX_LENGTH; i++)
		{
			iArray[i] = iNum %10;
			iNum = iNum / 10;
		}

		for (int i  =  0 ; i <MAX_LENGTH ; i++)
			if (iArray[i] == iArray[ (MAX_LENGTH+i-1)%MAX_LENGTH] ||
				iArray[i] == iArray[ (MAX_LENGTH+i-2)%MAX_LENGTH]  )
			{
				iDone =1;
				break;
			}

	} while(iDone != 0);

	return iArray;
}

int ComNumber::iBallCount(int iUserInput)
{
	int* iUserInputArray = SplitNumber(iUserInput);

	int iBallCnt = 0;
	for (int i  = 0 ; i <MAX_LENGTH; i++ )
		if(iUserInputArray[i] ==  iNumberArray[( i + MAX_LENGTH-1)%MAX_LENGTH] ||
		iUserInputArray[i] ==  iNumberArray[(i + MAX_LENGTH-2)%MAX_LENGTH] )
			iBallCnt ++;
	return iBallCnt;
}

int* ComNumber::SplitNumber(int iNumber)
{
	int* iUserInputArray = new int[MAX_LENGTH] ;

	for (int i  = 0 ; i <MAX_LENGTH; i++ )
	{
		iUserInputArray[i] = iNumber %10;
		iNumber /=10;
	}
	return iUserInputArray;

}

```

```
int ComNumber::iStrikeCount(int iUserInput)
{
	int* iUserInputArray= SplitNumber(iUserInput);

	int iStrikeCnt = 0;
	for (int i  = 0 ; i <MAX_LENGTH; i++ )
		if(iUserInputArray[i] ==  iNumberArray[i] )
			iStrikeCnt ++;
	return iStrikeCnt;
}

void ComNumber::Reset()
{
	iNumberArray = iCreateNumber();
}

int ComNumber::iGetNumber()
{
	return iNumberArray[0]+iNumberArray[1]*10+iNumberArray[2]*100;
}

```