아두이노 작품 제작
– MARBLE MAZE, LED FRACTAL, RC Car

학생명 : 10111 박한나, 10112 백소현, 10321 양미현, 10509 더걸유리, 10514 송정섭, 10710 노창현, 10716 송기웅, 10719 양희권, 10812 박영준

Ⅰ.제작 동기

1) 마블 메이즈

수학 시간에 선생님께서 물건의 무게중심을 맞추는 것을 보여주셨다. 그것을 따라 동아리 부원들끼리 물건의 무게중심을 맞춰보았다. 무게중심을 맞추는 것은 어렵지만 재밌다는 것을 깨달았다. 그러다가 '균형을 통해 쉽게 재미를 느낄 수 있는 방법이 없을까?' 란 의문으로 제작을 시작하게 되었다.

2) LED FRACTAL

평소 수학의 프랙탈에 관심이 많던 도중 LED cube 와 프랙탈을 접목하여 시에르핀스키 사면체를 구현하고자 LED matrix라는 프로젝트를 계획하게 되었다. 이 과정에서 시프트레지스터를 이용한 멀티플랙싱 기술을 도입하여 제작을 시작하게 되었다.

3) Bluetooth RC Car

평소에 스마트폰에 있는 기능 중 하나인 블루투스에 무선연결된 헤드폰이나 스피커를 신기하게 생각하였는데 아두이노에도 블루투스 모듈을 사용할 수 있다는 것을 알게 되었다. 블루투스로 할 수 있는 작품을 고르던 중 스마트폰으로 제어할 수 있는 RC카를 선택하게 되었다.

Ⅱ.제작 목적

1) 마블 메이즈 : 무게 중심에 대한 아이들의 흥미 확대

눈척으로 미로의 균형을 맞추며 공을 도착 지점까지 도착하게 하는 활동을 통해 경사면에 따른 공의 속도와 방향을 직접 체험하게 한다. 이로 인해 균형에 대한 학생들의 흥미를 확대한다.

2) LED FRACTAL : 멀티플랙싱 기술과 프랙탈에 대한 아름다움 인식

시프트레지스터와 멀티플랙싱을 이용해 LED cube가 아닌 LED FRACTAL을 만들어 내어 학생들에게 수열이 만들어낸 작품인 프랙탈의 아름다움을 일깨워준다. 또한 LED FRACTAL을 관람하므로써 더 나아가 실생활 또는 건축 등에 이용된 프랙탈을 스스로 찾아 낼 수 있는 능력을 키우고 그 과정을 통해 수열에 대한 흥미도를 확대한다.

3) RC Car : 내가 만든 나만의 자동차

안드로이드 블루투스 앱을 통해 자동차를 조정하면서 블루투스의 원리에 대해 깨닫게 되고, 바퀴의 앞으로, 뒤로, 좌회전, 우회전의 과정을 코딩하는 과정에서 바퀴의 방향과 속도가 자동차의 움직임에 어떤 영향을 끼치는지 고민해볼 수 있다.

Ⅲ. 제작 내용 및 과정과 결과

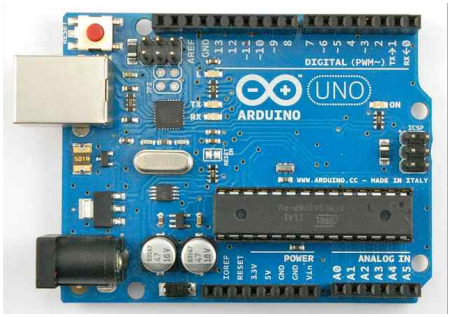
1) 이론적 배경

-아두이노란?

아두이노는 오픈 소스를 기반으로 한 단일 보드 마이크로컨트롤러로 완성 된 보드와 관련 개발 도구 및 환경을 말한다. 아두이노는 다수의 스위치나 센서로부터 값을 받아들여, LED나 모터와 같은 외부 전자 장치들을 통제함으로써 환경과 상호작용이 가능한 물건을 만들어 낼 수 있다. 임베디드 시스템 중의 하나로 쉽게 개발할 수 있는 환경을 이용하여, 장치를 제어할 수 있다.

아두이노 통합 개발 환경(IDE)을 제공하며, 소프트웨어 개발과 실행코드 업로드도 제공한다. 또한 어도비 플래시, 프로세싱, Max/MSP와 같은 소프트웨어와 연동할 수 있다.

아두이노의 가장 큰 장점은 마이크로컨트롤러를 쉽게 동작시킬 수 있다는 것이다. 일반적으로 AVR 프로그래밍이 AVR Studio(Atmel Studio로 변경, ARM 도구 추가됨)와 WinAVR(avr-gcc)의 결합으로 컴파일하거나 IAR E.W.나 코드비전(CodeVision)등으로 개발하여, 별도의 ISP 장치를 통해 업로드를 해야 하는 번거로운 과정을 거쳐야 한다. 이에 비해 아두이노는 컴파일된 펌웨어를 USB를 통해 쉽게 업로드 할 수 있다. 또한 아두이노는 다른 모듈에 비해 비교적 저렴하고, 윈도를 비롯해 맥 OS X, 리눅스와 같은 여러 OS를 모두 지원한다. 아두이노 보드의 회로도가 CCL에 따라 공개되어 있으므로, 누구나 직접 보드를 만들고 수정할 수 있다.



아두이노(UNO)

아두이노(ZERO)

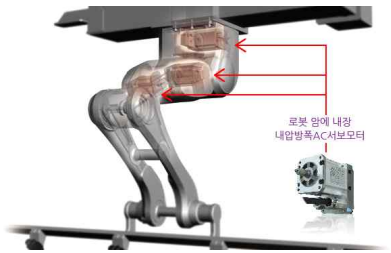
아두이노가 인기를 끌면서 이를 비즈니스에 활용하는 기업들도 늘어나고 있다. 장난감 회사 레고는 자사의 로봇 장난감과 아두이노를 활용한 로봇 교육 프로그램을 학생과 성인을 대상으로 북미 지역에서 운영하고 있다. 자동차회사 포드는 아두이노를 이용해 차량용 하드웨어와 소프트웨어를 만들어 차량과 상호작용을 할 수 있는 오픈XC라는 프로그램을 선보이기도 했다.

-서보모터란?

서보모터(Servo Motor)는 모터와 제어구동보드(적당한 제어 회로와 알고리즘)를 포함하는 것으로, 모터자체만 가지고 서보모터라 하지 않는다. 즉 서보모터라는 개념이 모터의 구동시스템까지 포함하는 것이기에 모터 자체만 가지고 이게 서보모터냐 아니냐를 따지는 것은 아니다.

서보(Servo)라는 용어는 '추종한다' 혹은 '따른다'는 의미로서 명령을 따르는 모터를 서보모터라고 한다. 공작기계, CCTV 카메라, 캠코더, DVD, 프린터 등에 사용되는 모터처럼 명령에 따라 정확한 위치와 속도를 맞출 수 있는 모터를 서보모터라고 한다.

서보 모터



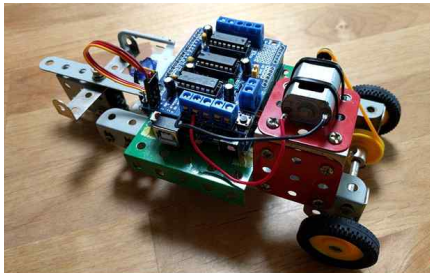
서보 모터의 사례(로봇 팔)

서보모터는 일반 제어 기구의 조절부와 자동 평형 계기의 섭동 저항부를 움직이는데 이용된다. 또한 산업용 로봇, 금속 공작기계 등 각종가공기계, 반도체 제조·가공장치, 포장 기계, 인쇄 기계, 목공 기계, 방전 가공기, 전용기, 연삭반, 볼반, 자동절단기 등 다양한 분야에서 이용이 되고 있다.

-DC 모터란?

DC 모터란, 고정자로 영구자석을 사용하고, 회전자(전기자)로 코일을 사용하여 구성한 것으로, 전기자에 흐르는 전류의 방향을 전환함으로써 자력의 반발, 흡인력으로 회전력을 생성시키는 모터이다.
모형 자동차, 무선조정용 장난감 등을 비롯하여 여러 방면에서 가장 널리 사용되고 있는 모터이다.

DC 모터



DC 모터의 사례

-블루투스란?

블루투스는 근거리 무선통신 규격의 하나로, 2.45GHz 주파수를 이용하여 반경 10~100m 범위 안에서 각종 전자, 정보통신 기기를 무선으로 연결·제어하는 기술규격을 말한다. 가정이나 사무실 내에 있는 컴퓨터, 프린터, 휴대폰, PDA 등 정보통신기기는 물론 각종 디지털 가전제품 간의 통신에 물리적인 케이블 없이 무선 주파수를 이용하여 고속으로 데이터를 주고받을 수 있다.

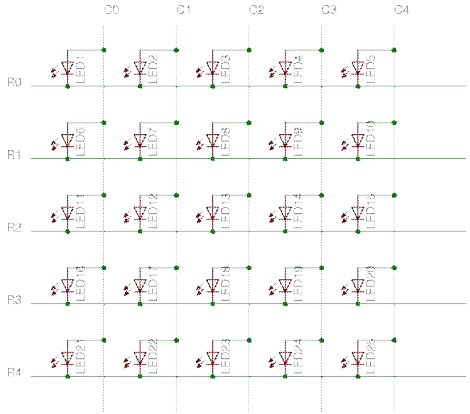
블루투스는 전 세계적으로 2,000여 개 업체가 가담할 정도로 무선 네트워킹의 세계적인 표준으로 떠오르고 있다. 이 기술을 적용하는 데 관심 있는 조직은 bluetooth adopter 합의서에 사인함으로서 참여할 수 있다. 이 조직의 회원사(adopter 또는 non-founding SIG회원사)는 블루투스 기술을 기반으로 하는 제품 개발에 로열티 없는 라이선스의 자격을 갖게 된다.

-멀티플랙싱(Multiplexing)이란?

하나의 통신로를 통하여 여러개의 독립된 신호를 전송하는 방식으로서 주파수 대역으로 구별하여 다중화를 행하는 주파수 분할 다중방식(FDM)과 고주파 펄스에 의해 각각의 신호를 일정 간격으로 표준화하여 이를 정해진 시간축상에 순서적으로 배열하여 전송하는 것에 의해 다중화를 행하는 시분할 다중방식(TDM)이 있다.

우리가 만드는 LED FRACTAL에서는 멀티플랙싱이 LEd의 음전하를 묶고, 양전하도 특정한 방법으로 묶어서 output을 줄이는 것으로 사용된다. 멀티플랙싱은 한 개씩 키려면 멀티플랙싱은 필요 없지만, 여러 개를 한 번에 키려고 하면 필수이다.

예를 들자면 LED12를 켜고 싶다면 C1와 R2만 키면 되고, LED25를 켜고 싶다면 C4와 R4를 키면 된다. 하지만 LED12와 LED25를 한 번에 켜고 싶다면 C1와 R2, C4와 R4를 켜야 하는데 이 과정에서 필요 없는 LED15와 LED22가 켜진다. 이것을 방지하기 위해 멀티플랙싱을 사용한다.



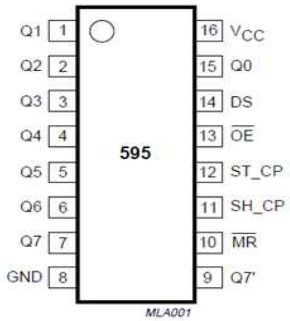
왼쪽 사진은 5*5 LED matrix를 기호로 표현한 그림이다. 멀티플랙싱 설명이 가장 쉬운 형태이다. 우리가할 LED cube는 행과 열 대신에 층과 기둥으로 하면 된다.

멀티플랙싱을 하는 방법은 C(0~4)또는 R(0~4)를 매우 빠른 속도(약 20밀리세컨드)로 순환하면서 해당하는 열이 켜질 때 필부분들만 키는 것이다. 이렇게 하면 LED15와 LED22가 켜지지 않는다.

-시프트 레지스터(Shift Register, 74HC595)란?

시프트 레지스터(shift register)는 디지털 회로에서 형 방식으로 설치된 프로세서 레지스터의 집합이며, 회로가 활성화 되었을 때 데이터를 줄 아래로 이동시키는 것과 같은 방법으로 입출력을 서로 연결하고 있다.

시프트 레지스터의 종류에는 시프트 레지스터는 직렬 입력, 병렬 출력(SIPO)과 병렬 입력, 직렬 출력(PISO) 형태를 포함하여 직렬과 병렬로 입출력을 결합할 수 있다. 또한 직병렬 입력을 가진 형태와 직병렬 출력을 가진 형태가 있다. 또한 시프트 레지스터의 방향을 다르게 할 수 있는 양방향 시프트 레지스터도 있다. 그리고 레지스터의 직렬 입력과 출력은 원형 시프트 레지스터를 만들기 위해 서로 연결할 수도 있다. 하나의 시프트 레지스터는 더 복잡한 연산을 수행할 수 있는 다차원 시프트 레지스터를 만들 수 있다.



PINNING

PIN	SYMBOL	DESCRIPTION
1	Q1	parallel data output
2	Q2	parallel data output
3	Q3	parallel data output
4	Q4	parallel data output
5	Q5	parallel data output
6	Q6	parallel data output
7	Q7	parallel data output
8	GND	ground (0 V)
9	Q7'	serial data output
10	$\overline{MR}$	master reset (active LOW)
11	SH_CP	shift register clock input
12	ST_CP	storage register clock input
13	$\overline{OE}$	output enable (active LOW)
14	DS	serial data input
15	Q0	parallel data output
16	V_{CC}	positive supply voltage

시프트 레지스터는 직렬과 병렬 인터페이스를 전환하는 데 가장 일반적으로 사용된다. 이것은 많은 회로가 병렬 비트의 집합으로 동작하기 때문에 유용하지만, 직렬 인터페이스의 구성이 더 간단하다. 시프트 레지스터는 간단한 지연 회로처럼 사용될 수 있다. 몇몇 양방향 시프트 레지스터는 스택의 하드웨어 구현을 위해서 병렬로 연결할 수도 있다.

시프트 레지스터는 펄스 확장기로 사용할 수도 있다. 단안정 멀티바이브레이터와 비교해서 타이밍은 구성요소의 값에 의존하지 않으나 외부 클럭이 요구되고 타이밍 정확성은 클럭의 입상도(granularity)에 의해 제한된다. 예시 - 로냐 트위스터는 다섯 개의 74164 시프트 레지스터가 이 방법으로 타이밍 논리의 코어를 생성한다.(회로도).

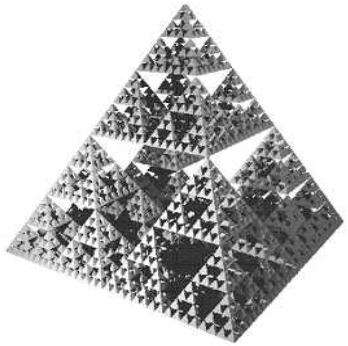
-프랙탈이란?

부분과 전체가 똑같은 모양을 하고 있다는 자기 유사성 개념을 기하학적으로 푼 구조를 말한다. 프랙탈은 단순한 구조가 끊임없이 반복되면서 복잡하고 묘한 전체 구조를 만드는 것으로, 즉 '자기 유사성(self-similarity)'과 '순환성(recursiveness)'이라는 특징을 가지고 있다. 자연계의 리아스식 해안선, 동물혈관 분포형태, 나뭇가지 모양, 창문에 성에가 자라는 모습, 산맥의 모습도 모두 프랙탈이며, 우주의 모든 것이 결국은 프랙탈 구조로 되어 있다. 프랙털은 수학적 도형으로도 연구되고 있다. 프랙털 도형은 종종 컴퓨터 소프트웨어를 이용한 재귀적이거나 반복적인 작업에 의한 반복되는 패턴으로 만들어진다. 대표적인 프랙털 도형에는 망델브로 집합, 칸토어 집합, 시에르핀스키 삼각형, 페아노 곡선, 코흐 곡선 등이 있다.

그 중 시에르핀스키 삼각형(Sierpiński triangle)은 바츠와프 시에르핀스키의 이름이 붙은 프랙탈 도형이다. 시에르핀스키 가스켓(Sierpiński gasket)으로도 불린다.

시에르핀스키 삼각형은 다음과 같은 방법을 통해 얻을 수 있다.

① 정삼각형 하나에서 시작한다.
② 정삼각형의 세 변의 중점을 이으면 원래의 정삼각형 안에 작은 정삼각형이 만들어진다. 이 작은 정삼각형을 제거한다.
③ 남은 정삼각형들에 대해서도 ②를 실행한다.
④ ③의 과정을 무한히 반복한다.



◀ 왼쪽 사진은 우리가 만드는 3차원 시에르핀스키 삼각형 사진이다. LED의 수가 적은 관계로 우리는 축소된 모형을 만들었다.

이것을 반복하면 다음과 같은 도형이 얻어진다.(무한반복)

2) 제작 과정 및 결과

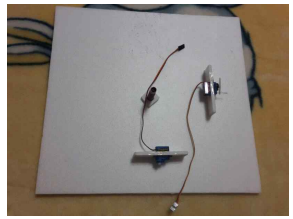
❶ 제작 과정

㉠ 아두이노 마블 메이즈(MARBLE MAZE)

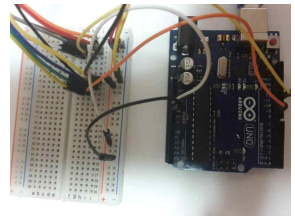
-준비물 : 아두이노(UNO), 서보모터 2개, 우드락, 스프링, 브레드 보드, 전선, 중심축(볼펜 심대), 아크릴, 구슬, 철사

-제작과정

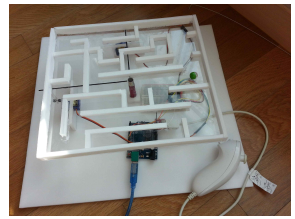
1. 마블 메이즈 기본받침 틀(우드락)과 미로(우드락+아크릴)를 디자인 및 제작한다.
2. 우드락으로 만든 기본 받침 틀에 중심축(용수철+볼펜)과 서보모터를 연결한다.
3. 회로도에 맞게 아두이노 (UNO)에 연결, 나머지를 배치한다.
4. 프로그래밍한 코드를 아두이노(UNO)에 입력한 후 컴파일 한다.
5. 제작 결과를 관찰한다.



서보모터와 중심축을 연결한 받침틀



회로도 연결



마블 메이즈 완성본

-아두이노 MARBLE MAZE 코드

```
#include <Wire.h>
#include <Servo.h>

Servo servoLeft;          // Define left servo
Servo servoRight;         // Define right servo

static uint8_t nunchuck_buf[6];   // array to
store nunchuck data,

void setup()
{
  Serial.begin(19200);

  servoLeft.attach(11);  // Set left servo to
digital pin 10
  servoRight.attach(9);  // Set right servo to
digital pin 9

  nunchuck_setpowerpins(); // use analog pins
2&3 as fake gnd & pwr
  nunchuck_init(); // send the initilization
handshake
  Serial.print ("Finished setup\n");
}

void loop()
{
  nunchuck_get_data();

// map nunchuk data to a servo data point
  int x_axis = map(nunchuck_buf[0], 23, 222,
180, 0);
  int y_axis = map(nunchuck_buf[1], 32, 231,
0, 180);

//move servo to desired position based on Wii
nunchuk reading
  servoLeft.write(x_axis);
  servoRight.write(y_axis);

//  un-comment next line to print data to
serial monitor
//  nunchuck_print_data();

}
}

//
// Nunchuck functions
//

// Uses port C (analog in) pins as power &
ground for Nunchuck
static void nunchuck_setpowerpins()
{
#define pwrpin PORTC3
#define gndpin PORTC2
    DDRC |= _BV(pwrpin) | _BV(gndpin);
    PORTC &=~ _BV(gndpin);
    PORTC |= _BV(pwrpin);
    delay(100);     //  wait  for  things  to
stabilize
}

// initialize the I2C system, join the I2C
bus,
// and tell the nunchuck we're talking to it
void nunchuck_init()
{
  Wire.begin();                    // join i2c
bus as master
  Wire.beginTransmission(0x52);             //
transmit to device 0x52
  Wire.write(0x40);            // sends memory
address
  Wire.write(0x00);            // sends sent a
zero.
  Wire.endTransmission();             //  stop
transmitting
}

// Send a request for data to the nunchuck
// was "send_zero()"
void nunchuck_send_request()
{
  Wire.beginTransmission(0x52);             //
transmit to device 0x52
  Wire.write(0x00);              // sends one
byte
  Wire.endTransmission();             //  stop
transmitting
}
```

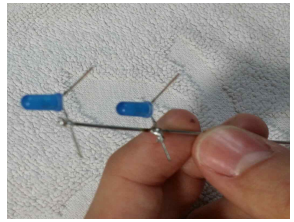
```
// Receive data back from the nunchuck,
int nunchuck_get_data()
{
    int cnt=0;
    Wire.requestFrom (0x52, 6);     // request
data from nunchuck
    while (Wire.available ()) {
      // receive byte as an integer
      nunchuck_buf[cnt]                      =
nunchuk_decode_byte(Wire.read());
      cnt++;
    }
    nunchuck_send_request();  // send request
for next data payload
    // If we recieved the 6 bytes, then go
print them
    if (cnt >= 5) {
     return 1;   // success
    }
    return 0; //failure
}

// Print the input data we have recieved
// accel data is 10 bits long
// so we read 8 bits, then we have to add
// on the last 2 bits.  That is why I
// multiply them by 2 * 2
void nunchuck_print_data()
{
  static int i=0;
  int joy_x_axis = nunchuck_buf[0];
  int joy_y_axis = nunchuck_buf[1];

  int accel_x_axis = nunchuck_buf[2]; // * 2 *
2;
  int accel_y_axis = nunchuck_buf[3]; // * 2 *
2;
  int accel_z_axis = nunchuck_buf[4]; // * 2 *
2;


  int z_button = 0;
  int c_button = 0;

  // byte nunchuck_buf[5] contains bits for z
and c buttons
  // it also contains the least significant
bits for the accelerometer data
  // so we have to check each bit of byte
outbuf[5]
  if ((nunchuck_buf[5] >> 0) & 1)
    z_button = 1;
  if ((nunchuck_buf[5] >> 1) & 1)
    c_button = 1;

  if ((nunchuck_buf[5] >> 2) & 1)
    accel_x_axis += 2;
  if ((nunchuck_buf[5] >> 3) & 1)
    accel_x_axis += 1;

  if ((nunchuck_buf[5] >> 4) & 1)
    accel_y_axis += 2;
  if ((nunchuck_buf[5] >> 5) & 1)
    accel_y_axis += 1;

  if ((nunchuck_buf[5] >> 6) & 1)
    accel_z_axis += 2;
  if ((nunchuck_buf[5] >> 7) & 1)
    accel_z_axis += 1;

  Serial.print(i,DEC);
  Serial.print("\t");

  Serial.print("joy:");
  Serial.print(joy_x_axis,DEC);
  Serial.print(",");
  Serial.print(joy_y_axis, DEC);
  Serial.print("  \t");

  Serial.print("acc:");
  Serial.print(accel_x_axis, DEC);
  Serial.print(",");
  Serial.print(accel_y_axis, DEC);
  Serial.print(",");
  Serial.print(accel_z_axis, DEC);
  Serial.print("\t");

  Serial.print("but:");
  Serial.print(z_button, DEC);
  Serial.print(",");
  Serial.print(c_button, DEC);

  Serial.print("\r\n");  // newline
  i++;
}

// Encode data to format that most wiimote
drivers except
// only needed if you use one of the regular
wiimote drivers
char nunchuk_decode_byte (char x)
{
  x = (x ^ 0x17) + 0x17;
  return x;
}
```


㉡ 아두이노 LED FRACTAL

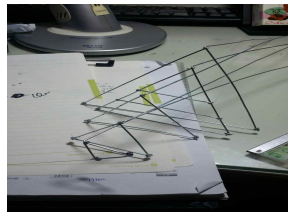
-준비물 : 아두이노(UNO), LED 130개, 저항, 브레드 보드, 전선, 철사, 납, 우드락, 절연테이프

-제작과정

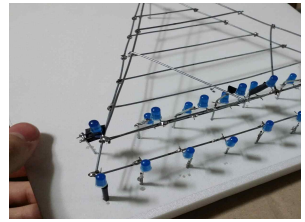
1. 철사를 용접과 납땜의 과정을 거쳐 시에르핀스키 삼각형의 틀을 만든다.
2. 철사를 구부려서 음전하를 위한 삼각형 틀을 만든다.
3. 철사에 2.5cm 간격으로 표시하고 필요한 길이를 자른다.
3. 설계도에 맞게 led의 양전하 다리를 철사에 납땜한다.
4. 우드락에 삼각형틀과 철사를 설치한 뒤 음전하 다리를 삼각형틀에 납땜한다.
4. 설계도에 맞게 위 과정을 반복한다.
5. 시프트 레지스터를 음전하는 음전하끼리, 양전하는 양전하끼리 연결하여 시프트 레지스터를 두 줄로 만든다.
6. 시프트 레지스터를 데이지 체인하고 순서대로 아웃풋에 연결한 후 적절한 클락 조절 인풋과 아두이노를 연결한다.
7. 멀티플렉싱을 함수로 코딩한 후 표시할 것을 디자인하고 함수를 사용해 코딩, 업로드한다.
8. 제작 결과를 관찰한다.



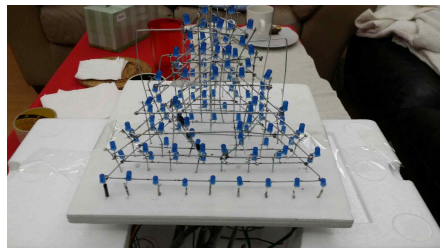
양전하 기둥



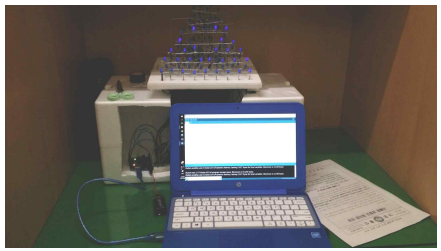
음전하 프레임



첫 두 음전하 프레임과 양전하 기둥을 연결한 모습



시에르핀스키 사면체 구조물



프로그램이 실행된 모습

-아두이노 LED FRACTAL 코드

```
int pos_shcp = 10; // shift register clock input
int pos_stcp = 12; // storage register clock input
int pos_ds = 11; // serial data input
int neg_shcp = 5;
int neg_stcp = 7;
int neg_ds = 6;

int pos_tick(int input)
{
if (input==1)//if pos-rod is on
{
digitalWrite(pos_ds,HIGH);
digitalWrite(pos_shcp,HIGH);
digitalWrite(pos_ds,LOW);
digitalWrite(pos_shcp,LOW);//output = pos, shift once
}
else
{
digitalWrite(pos_shcp,HIGH);
digitalWrite(pos_shcp,LOW);//only shift
}
}

int neg_tick(int input)
{
if (input==1)//if neg-rod is off
{
digitalWrite(neg_ds,HIGH);
digitalWrite(neg_shcp,HIGH);
digitalWrite(neg_ds,LOW);
digitalWrite(neg_shcp,LOW);//output = pos, shift once
}
else
{
digitalWrite(neg_shcp,HIGH);
digitalWrite(neg_shcp,LOW);//only shift
}
}

int pos_upload()
{
digitalWrite(pos_stcp,HIGH);//latch for pos-rod
digitalWrite(pos_stcp,LOW);
}

int neg_upload()
{
digitalWrite(neg_stcp,HIGH);//latch for neg-rod
digitalWrite(neg_stcp,LOW);
}
int pos_shift()
{
digitalWrite(pos_shcp,HIGH);
digitalWrite(pos_shcp,LOW);//only shift
}

int neg_shift()
{
digitalWrite(neg_shcp,HIGH);
digitalWrite(neg_shcp,LOW);//only shift
}

int up_floor(int num)
{
digitalWrite(neg_ds,HIGH);
for(int i =1 ; i <= num-1 ; i++)
{
digitalWrite(neg_shcp,HIGH);
digitalWrite(neg_shcp,LOW);
}
digitalWrite(neg_ds,LOW);//output = gnd

digitalWrite(neg_shcp,HIGH);//only shift.
digitalWrite(neg_shcp,LOW);

digitalWrite(neg_ds,HIGH);//output = pos
for(int i=1; i < 9-num; i++)//shift (8-num) times
{
digitalWrite(neg_shcp,HIGH);
digitalWrite(neg_shcp,LOW);
}
digitalWrite(neg_ds,LOW);

digitalWrite(neg_stcp,HIGH);//latch
digitalWrite(neg_stcp,LOW);
}

int flush_floor()
{
for(int i = 0; i<=9; i++)
{
digitalWrite(neg_shcp,HIGH);
digitalWrite(neg_shcp,LOW);//flush floors
}
}
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(pos_shcp, OUTPUT);
pinMode(pos_stcp, OUTPUT);
pinMode(pos_ds, OUTPUT);

pinMode(neg_shcp, OUTPUT);
pinMode(neg_stcp, OUTPUT);
pinMode(neg_ds, OUTPUT);
}


void loop() {

//1start
```



```
int pos_1[9][42]={/*1*/0,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/1,1,1,1,1,1,1,1,1,

/*1*/0,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,/*8*/1,1,1,1,1,1,1,1,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/0,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/1,1,1,1,1,1,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/0,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/1,1,1,1,1,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/0,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/1,1,1,1,1,/*6*/0,0,0,0,/*7*/0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/0,/*2*/0,0,/*3*/0,0,0,/*4*/1,1,1,1,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/0,/*2*/0,0,/*3*/1,1,1,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/0,/*2*/1,1,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,};

int neg_1[9]={0,0,0,0,0,0,0,0,0};


for(int k=0; k <= 1000 ; k++){
//Simple front display(1)
neg_tick(0);

for(int j=0; j < 9; j++)
{
neg_upload();

for(int i =0 ;i<42 ; i++)
{
pos_tick(pos_1[j][i]);
}
pos_upload();

delay(10);
neg_tick(1);
}
}
//1end,2start
```

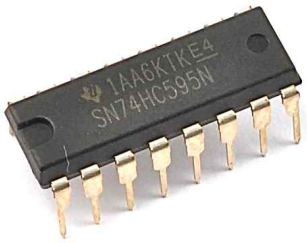
```
int pos_2_1[9][42]={/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/1,0,0,1,/*5*/1,0,0,0,1,/*6*/1,0,0,1,/*7*/0,0,0,0,0,0,/*8*/1,0,0,1,1,0,0,1,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/1,0,1,/*4*/0,0,0,0,/*5*/1,0,0,0,1,/*6*/0,0,0,0,/*7*/1,0,1,1,0,1,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/1,1,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/1,0,0,0,1,/*6*/1,1,1,1,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/1,0,0,0,1,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/1,0,0,1,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/1,0,1,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/1,1,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,};

int pos_2_2[9][42]={/*1*/1,/*2*/1,1,/*3*/1,0,1,/*4*/1,0,0,1,/*5*/1,0,0,0,1,/*6*/1,0,0,1,/*7*/1,0,0,0,1,/*8*/1,0,0,0,0,0,0,1,/*9*/1,1,1,1,1,1,1,1,1,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/1,0,0,0,0,0,0,1,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/1,0,0,0,0,1,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/0,0,0,0,0,/*6*/1,0,0,1,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/1,0,0,0,1,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/1,0,0,1,/*5*/0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,
```

```
/*1*/1,/*2*/0,0,/*3*/1,0,1,/*4*/0,0,0,0,/*5*/
0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/
0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/1,1,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/
0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/
0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,0,

/*1*/1,/*2*/0,0,/*3*/0,0,0,/*4*/0,0,0,0,/*5*/
0,0,0,0,0,/*6*/0,0,0,0,/*7*/0,0,0,0,0,0,/*8*/
0,0,0,0,0,0,0,0,/*9*/0,0,0,0,0,0,0,0,0,0,};



//simple triangle display
for(int num=0; num<=1000;num++)
{
neg_tick(0);

for(int j=0; j < 9; j++)
{

neg_upload();

for(int i =0 ;i<42 ; i++)
{
pos_tick(pos_2_2[j][i]);
}
pos_upload();

delay(10);
neg_tick(1);
}
}
```

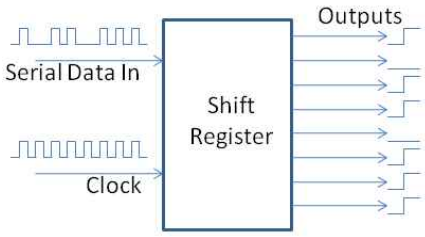
```
for(int num=0;num<=1000;num++)
{
neg_tick(0);

for(int j=0; j < 9; j++)
{

neg_upload();

for(int i =0 ;i<42 ; i++)
{
pos_tick(pos_1[j][i]);

}
pos_upload();

delay(10);
neg_tick(1);
}
}


}
```

- LED Fractal 설계도 및 원리

아두이노에 직접 LED를 연결해도 되는데 Shift Register로 구현하는 이유는 아두이노의 핀의 개수(OUTPUT)의 개수가 D0 ~ D13까지 제한이 있기 때문에 130개의 LED를 켜는 것은 불가능하기 때문이다. 이를 위해 Data를 분산하고 제어하는 Shift Register를 도입하여 130개의 LED를 한번에 제어할 수 있게 되었다.

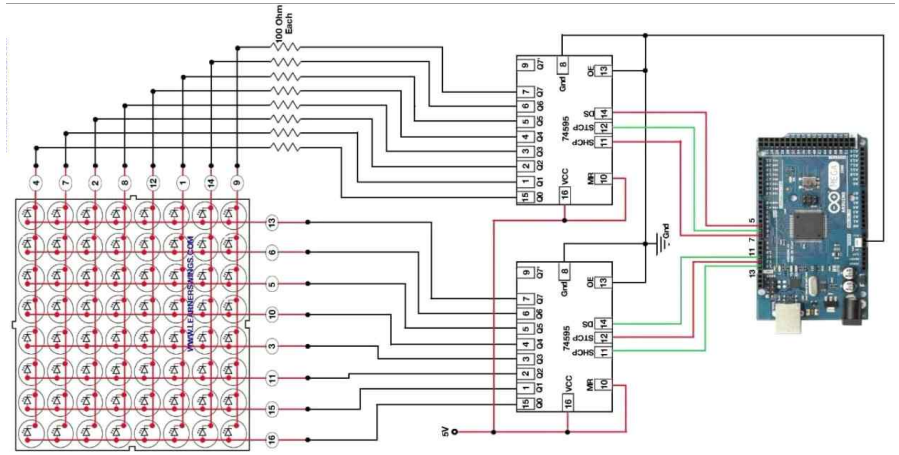


Shift Register(SN74HC595N)



Shift Register의 데이터 분산 원리

아래의 설계도는 2차원 평면상에 구현한 LED이므로 현재 우리의 LED Fractal과는 다르다. 하지만 현재 시에르핀스키 사면체를 설계도로 구현하기 어려우므로 위의 설계도로 대체한다. 여기서 8개의 양전하 설계를 42개로 바꾸고 8개의 음전하 설계를 2개로 바꾸면 우리의 설계도와 같다.



Shift Register를 이용해 130개의 LED를 연결하기 위해서는 가로 층 9개를 음전하로 연결하여 두 개의 Shift Register로 연결하고, 42개의 양전하 세로선을 양전하로 연결하여 6개의 Shift Register에 연결한 후 아웃풋의 개수를 줄인다. 양전하와 음전하 철사를 위의 그림과 같이 교차하여 만나게 한 후 양전하와 음전하가 통하게 하여 130개의 LED를 켤 수 있다.

㉢ RC Car

-준비물 : 아두이노(UNO), DC모터 2개, 브레드 보드, 전선, 바퀴 2개, 나무판자, 모터쉴드, 스마트폰, 납, 블루투스 모듈

-제작과정

1. RC카의 밑면이 될 나무판자를 자른다.
2. 아두이노와 모터쉴드를 합친 후에 나무판자에 붙인다.
3. 모터와 전선을 연결하고 모터쉴드에 모터와 같은 극을 연결한다.
4. 모터에 바퀴를 연결하고 나무판자 밑에 테이프로 고정한다.
5. 모터쉴드에 블루투스 모듈을 납땜해서 연결한다.
6. 스마트폰 어플을 이용해서 어플과 블루투스 모듈을 연결한다.
7. 프로그래밍한 코드를 아두이노(UNO)에 입력한 후 컴파일 한다.
8. 제작결과를 관찰한다.

◀ RC Car 제작 과정

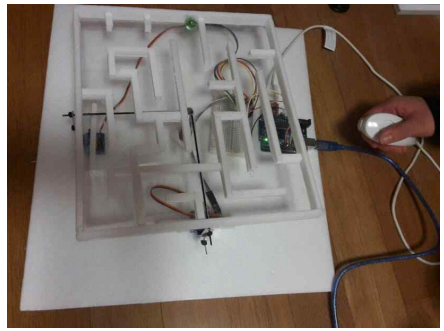
-아두이노 RC Car 코드

```
#include <AFMotor.h>
#include <SoftwareSerial.h> //시리얼 통신 라이
브러리 호출

AF_DCMotor motor1(1, MOTOR12_64KHZ);
AF_DCMotor motor2(2, MOTOR12_64KHZ);
//AF_DCMotor motor3(3, MOTOR34_64KHZ);
//AF_DCMotor motor4(4, MOTOR34_64KHZ);

//블루투스 PIN 설정
#define BLUE_TX A0 //블루투스 TX
#define BLUE_RX A1 //블루투스 RX

SoftwareSerial mySerial(BLUE_TX, BLUE_RX); //
시리얼 통신을 위한 객체선언
char value; //블루투스에서 받는 값

int speed = 200;

//바퀴 속도 일괄 세팅
int set_speed(int sp){

motor1.setSpeed(sp);
motor2.setSpeed(sp);
//motor3.setSpeed(sp);
//motor4.setSpeed(sp);
}
int forward(){
motor1.run(FORWARD);
motor2.run(FORWARD);
//motor3.run(FORWARD);
//motor4.run(FORWARD);
}

int backward(){
motor1.run(BACKWARD);
motor2.run(BACKWARD);
//motor3.run(BACKWARD);
//motor4.run(BACKWARD);
}

int left(){
motor1.run(FORWARD);
motor2.run(BACKWARD);
//motor3.run(BACKWARD);
//motor4.run(FORWARD);
}

int right(){
motor1.run(BACKWARD);
motor2.run(FORWARD);
//motor3.run(FORWARD);
//motor4.run(BACKWARD);
}

int stop(){
motor1.run(RELEASE);
motor2.run(RELEASE);
//motor3.run(RELEASE);
//motor4.run(RELEASE);

void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
Serial.println("Motor test!");

set_speed(100);
}


void loop() {
while(mySerial.available())  //mySerial에  전송
된 값이 있으면
{
value = mySerial.read(); //전송값 읽음
Serial.print(value);
if(value == 'l'){ //좌회전
Serial.println("left");
left();
}else if(value == 'r'){ //우회전
Serial.println("right");
right();
}else if(value == 'f'){ //전진
Serial.println("forward");
forward();
}else if(value == 'b'){ //후진
Serial.println("backward");
backward();
}else if(value == 's'){ //정지
Serial.println("stop");
stop();
}
}

}
```

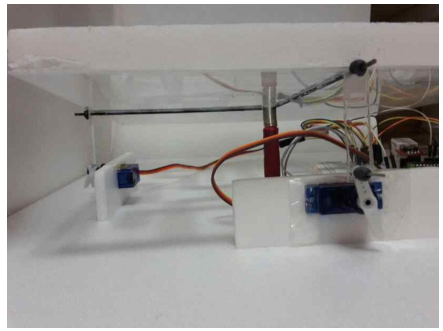

❸제작 결과

㉠ 아두이노 MARBLE MAZE

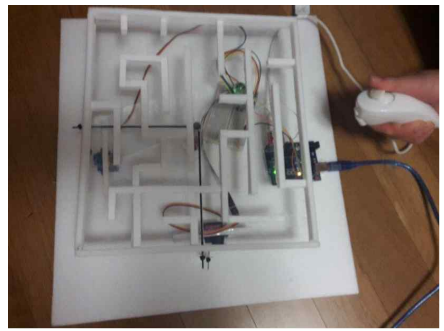
계획한 대로 눈척의 움직임에 따라 서보모터가 작동해 경사면을 만듦으로써 경사면에 따라 구슬이 움직이는 게임인 마블메이즈를 제작 완료했다.



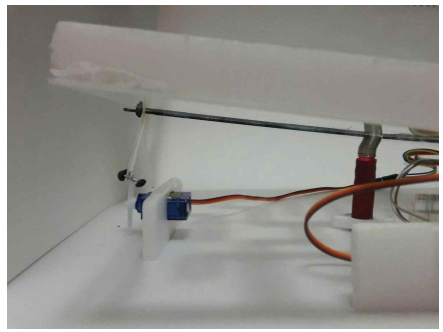
wii 눈척으로 마블메이즈를 실행하는 모습



앞의 모터의 날개가 일자로 곧게 세워지는 모습



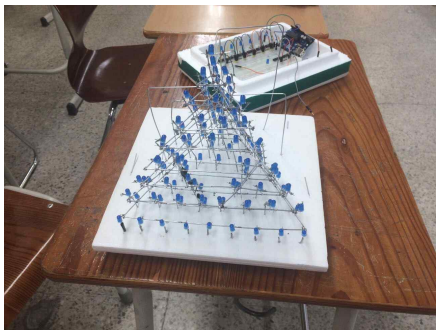
wii 눈척으로 마블메이즈를 실행하는 모습



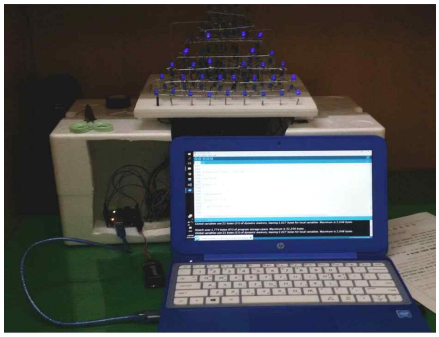
왼쪽 모터의 날개가 일자로 곧게 세워져
미로가 기울어지는 모습

㉡ 아두이노 LED FRACTAL

시에르핀스키 삼각형의 모양으로 쌓은 LED FRACTAL에 코딩한대로 (순식간이지만) LED가 층별로 켜졌으며, 가장자리부터 반짝이기 등 여러 가지 Display가 실행되어 뿌듯함을 느꼈다.



완성된 시에르핀스키의 구조물



LED 에 불이 들어온 모습

V.참고 자료

- 선행 아두이노 작품들

◎ Arduino Labyrinth Maze Game Solver

https://www.youtube.com/watch?v=RRKcLv8gxkU

◎ Arduino LED

https://www.youtube.com/watch?v=GUcX41pokZY

- 용어 정의

◎ 위키 백과

https://ko.wikipedia.org

◎ 네이버 지식 백과

http://terms.naver.com

- 이미지

http://wiki.vctec.co.kr/opensource/arduino/arduinotutorial_led (아두이노 UNO)

http://studymake.tistory.com/152 (아두이노 ZERO)

http://kocoafab.cc/tutorial/view/62 (서보모터)

http://www.takubo.co.jp/k/product/others/servo_motor.html (서보모터의 예 - 로봇 팔)

http://blog.cheetahking.net/79 (3차원 프렉탈 삼각형)

https://argonblue.wordpress.com/tag/multiplexing (멀티플랙싱)

http://www.electronics-lab.com/project/3-wire-serial-lcd-using-a-shift-register (시프트 레지스터)