**마이크로 프로세서 및 실습 보고서**

**Project Hive\_rid**

**장경식 교수님**

**02 분반**

**2015136111 임세정**

**목차**

**1.개요**

**(1)선정 이유**

**(2)참고 제품**

**2.구성도**

**(1)구성도**

**3.주요기능**

**4.개발 일정**

**5.소스코드**

**6.결과**

**7.[부록]참고자료**

**8.github:** [**https://github.com/kutmicro/2015136111**](https://github.com/kutmicro/2015136111)

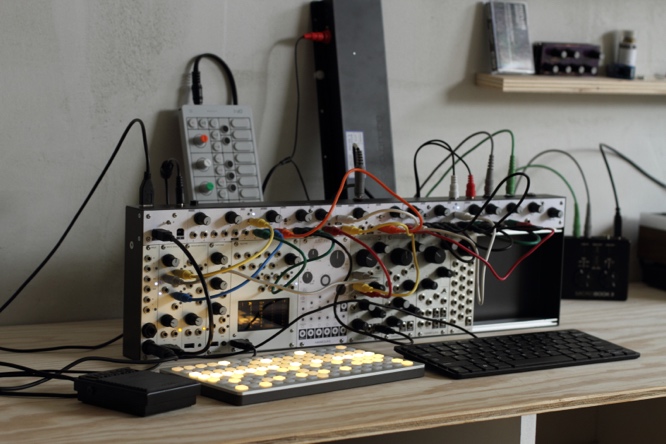
1.개요

선정이유

이전부터 미디 컨트롤러에 대해 구현하고 싶었고, 한정되어있는 구성보다는 원하는대로 구성하고 싶었습니다. 시중에 있는 미디 컨트롤러는 안정성은 뛰어나나 기능에 비해 너무나도 가격이 높기 때문에, 적당한 가격 내에서 더 우월한 기능을 할 수 있는 제품을 만들어 보고 싶었습니다.

참고

1.arduinome—calarts



original: monome

2.Livid instrument-ohm64



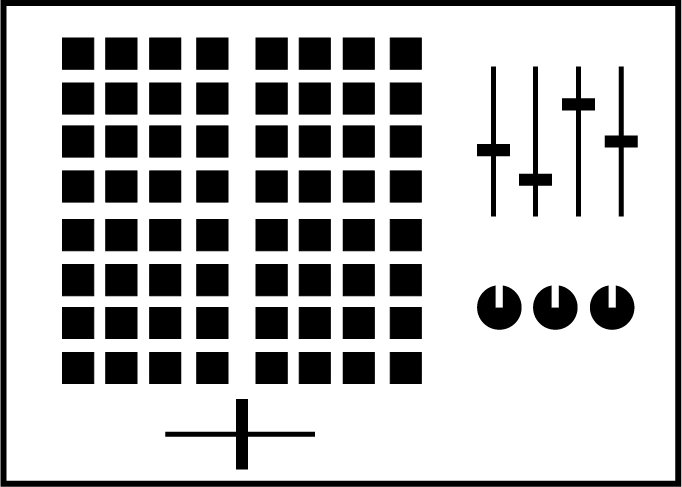
3.Akai-apc40

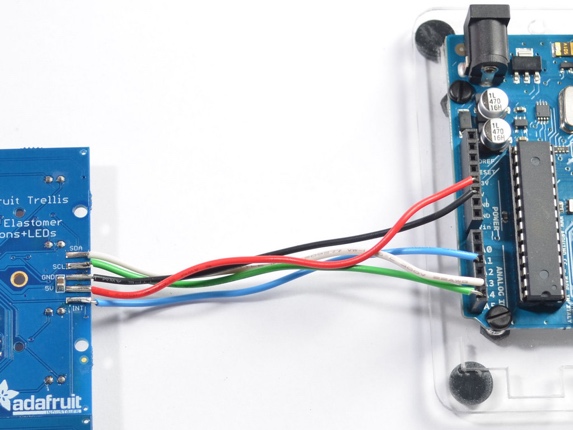
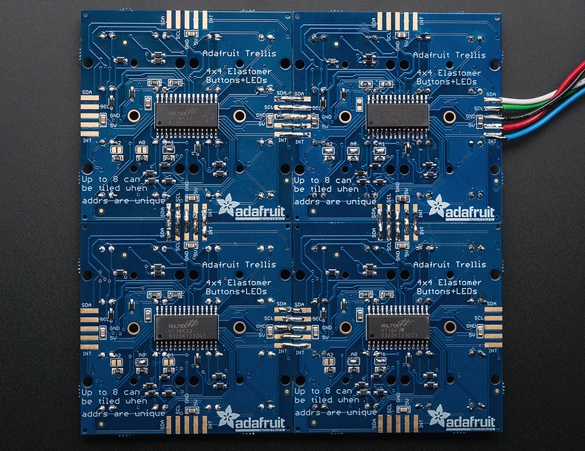


기능: 미디 신호 송출

|  |
| --- |
| **Info for BOM** |
| **Diffused Green 3mm LED (25 pack) [ADA-779]**  **68195**  🡺6,209\*3원 **\**18627  🡺http://toolparts.co.kr/front/php/product.php?product\_no=68195&main\_cate\_no=&display\_group=  **4x4 실리콘 버튼 키패드 (Silicone Elastomer 4x4 Button Keypad - for 3mm LEDs) [ADA-1611]🡺7700\*4원 \30800**  **🡺http://roboholic1.godo.co.kr/shop/goods/goods\_view.php?goodsno=329783**  **4x4 키패드 PCB 보드 (Adafruit Trellis Monochrome Driver PCB for 4x4 Keypad & 3mm LEDs)** [3714]ADA-1616 **🡺14300\*4 \57200**  **🡺http://roboholic1.godo.co.kr/shop/goods/goods\_view.php?goodsno=329784**  **아날로그 포지션 검출 슬라이드 스위치 센서 Analog Slide Position Sensor**  [DFR-DFR0053]  🡺8800\*4원 **\**35200  🡺http://roboholic1.godo.co.kr/shop/goods/goods\_view.php?goodsno=2822&category=060003  **[SEEED] Grove 슬라이드 스위치 가변저항 포텐쇼미터 (Grove - Slide Potentiometer)** [3371] SEE-101020036  🡺8250원  🡺http://roboholic1.godo.co.kr/shop/goods/goods\_view.php?goodsno=12557&category=060003  포텐셜미터+노브\*3  **10K 가변저항 (포텐쇼미터) (Rotary Potentiometer - 10k Ohm)**  [2139] ADA-1831 🡺1100\*3원 **\**3300  **http://roboholic1.godo.co.kr/shop/goods/goods\_view.php?goodsno=12280&category=044016003**  **가변저항 포텐쇼미터 노브 Potentiometer Knob - Soft Touch T18 - White**  [0744] ADA-2047 🡺770\*3원 **\**2310  http://roboholic1.godo.co.kr/shop/goods/goods\_view.php?goodsno=12331&category=063 |

디자인(수정전)





pad: GND,5V+아날로그 인 3개 사용



궁금한 점

Q. usb로 midi/osc 신호 오가는 것을 구현하는 것이 어떤 방법이 있으며,복잡한가?

2016.12.11

계획 수정 사항

trellis breakout pcb 보드가 배송이 지연되어 버튼으로 대체함에 따라 코드도 기존에 확보한 것 중 버튼에 사용가능한 것으로 대체하였습니다.

포텐셔미터의 경우 배송 문제와 아두이노의 처리 능력 문제로 인해 3개로 감소시켜 구현하였습니다.

또한 I2C가 아니라 tx 아웃풋에서 시리얼로 미디 신호를 출력시키는 방식으로 전환하였습니다.

구성도

DAW

/VDMX

MIDI

interface

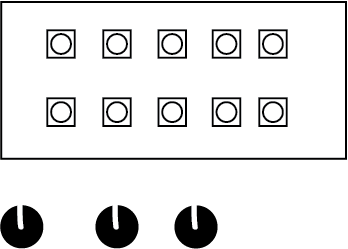
5pin socket

TX0

Arduino

Mega2560

컨트롤러 구성도



버튼 10개, 노브(포텐셔미터 3개)

버튼은 아래 차트의 규격을 따라 맨 오른쪽 아래부터 좌측으로 하나씩1~5,윗줄은6~10으로 넘버링하였으며,검은 건반을 제외하고 C4~E5까지 매핑되어 있습니다. 하지만 미디의 기능은 건반에 매핑하는 것에서 끝나지 않고, 더 넓은 범위에서 활용이 가능하기 때문에 소프트웨어 상에서 한번 더 매핑을 해주었습니다. 포텐셔미터는 특정기능을 담당하는 값을 피해 1,10,22에 매핑해 주었으며, 음악이나 영상에 걸리는 효과의 파라미터를 조절할 수 있게 매핑했습니다.



2.핀 연결

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Pin |  | --- |
| 포텐셔미터1 | A5 |  |  |
| 포텐셔미터2 | A6 |  |  |
| 포텐셔미터3 | A14 |  |  |
| 5Pin socket |  | VCC,GND |  |
| Button10 | Pwm13 | GND |  |
| Button9 | Pwm12 | GND |  |
| Button8 | Pwm11 | GND |  |
| Button7 | Pwm10 | GND |  |
| Button6 | Pwm9 | GND |  |
| Button5 | Pwm8 | GND |  |
| Button4 | Pwm7 | GND |  |
| Button3 | Pwm6 | GND |  |
| Button2 | Pwm5 | GND |  |
| Button1 | Pwm4 | GND |  |

3.기능

버튼별 매핑된 기능

* C4: 비디오 클립 랜덤 트리거
* D4: zoom blur fx
* E4: Reflected tile(만화경) fx
* F4: fade 모드
* G4: strobe 모드
* A4:Lean On (Delta heavy remix)-음악
* B4:Punish my love VIP-음악
* C5:Face Down-음악
* D5:비디오 클립 초기화
* E5: Three pound chicken wing-음악

포텐셜미터에 매핑된 기능

* 포텐셔미터 1:LPF/비디오 클립 opacity
* 포텐셔미터2:bitcrusher/글리치
* 포텐셔미터 3: ping pong delay(잔향)

위쪽에 위치한6-10에 해당하는 버튼들은 DAW(Digital Audio Workstation)을 제어하는 역할 위주로 매핑시켰으며, 아래쪽에 위치한 1~5에 해당하는 버튼들은 VDMX라는 비디오 클립 송출 소프트웨어를 조작하는 기능 위주로 매핑시켰습니다. 포텐셔미터의 경우 1,2는 유사한 기능을 음악과 영상이 동시에 조작되도록 매핑해주었고, 3의 경우는 공간계 이펙터(잔향) 값을 조절할 수 있게 매핑해주었습니다.

4.개발 일정

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 주차 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 주제 탐색 | **O** | **O** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 기획 |  | **O** | **O** |  |  |  |  |  |  |  |  |  |  |  |  |
| 라이브러리 탐색 |  |  | **O** | **O** | **O** | **O** |  |  |  |  |  |  |  |  |  |
| 라이브러리  이해 |  |  |  | **O** | **O** | **O** |  |  |  |  |  |  |  |  |  |
| 부품 배송 대기 |  |  |  |  |  | **O** | **O** | **O** | **O** | **O** | **O** | **O** |  |  |  |
| 대체 부품 확보 |  |  |  |  |  |  |  |  |  |  |  |  | **O** |  |  |
| 코드 재확보 |  |  |  |  |  |  |  |  |  |  |  |  | **O** |  |  |
| 부품 테스트 |  |  |  |  |  |  |  |  |  |  |  |  | **O** |  |  |
| 미디 연결 테스트 |  |  |  |  |  |  |  |  |  |  |  |  | **O** | **O** |  |
| DAW/VDMX  매핑 |  |  |  |  |  |  |  |  |  |  |  |  |  | **O** |  |
| 최종데모 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | **O** |

5.소스 코드

테스트용 소스 코드

|  |
| --- |
| Button.ino |
| // constants won't change. They're used here to  // set pin numbers:  const int buttonPin = 4;  //const int buttonPin = 5;  //const int buttonPin = 6;  //const int buttonPin = 7;  //const int buttonPin = 8;  //const int buttonPin = 9;  //const int buttonPin = 10;  //const int buttonPin = 11;  //const int buttonPin = 12;  // the number of the pushbutton pin  const int ledPin = 13; // the number of the LED pin  // variables will change:  int buttonState = 0; // variable for reading the pushbutton status  void setup() {  // initialize the LED pin as an output:  pinMode(ledPin, OUTPUT);  // initialize the pushbutton pin as an input:  pinMode(buttonPin, INPUT);  }  void loop() {  // read the state of the pushbutton value:  buttonState = digitalRead(buttonPin);  // check if the pushbutton is pressed.  // if it is, the buttonState is HIGH:  if (buttonState == HIGH) {  // turn LED on:  digitalWrite(ledPin, HIGH);  } else {  // turn LED off:  digitalWrite(ledPin, LOW);  }  } |

버튼 테스트의 경우 기본led를 통해 잘 작동되는지 테스트해 주었습니다.

|  |
| --- |
| Pottest.ino |
| void setup() {    Serial.begin(9600);  }  void loop() {  // read the input on analog pin 0:  //int sensorValue = analogRead(A5); //pot1 test  //int sensorValue = analogRead(A6); //pot2 test  int sensorValue = analogRead(A14); //pot3 test    Serial.println(sensorValue); //print  delay(100); // delay  } |

포텐셔미터의 경우 값이 잘 전송되는지 시리얼로 내보내도록 하고 시리얼 모니터에서 확인해 주었습니다.

최종 소스 코드

|  |
| --- |
| MIDI\_Controller\_v1-2mod2.ino |
| #include <MIDI.h>  #include "Controller.h"  MIDI\_CREATE\_DEFAULT\_INSTANCE();  #define B1 4  #define B2 5  …//버튼 매핑(생략)  #define B10 13  #define ptc1 A5  #define ptc2 A6  #define ptc3 A14  //pins  // button number  byte NUMBER\_BUTTONS = 10;  // potentiometers number  byte NUMBER\_POTS = 3;  //Pot (Pin Number, Command, CC Control, Channel Number)  Pot PO1(ptc1, 0, 1, 1);  Pot PO2(ptc2, 0, 10, 1);  Pot PO3(ptc3, 0, 22, 1);  //array  Pot \*POTS[] {&PO1,&PO2,&PO3};  //Button (Pin Number, Command, Note Number, Channel, Debounce Time)  //Command parameter 0=NOTE 1=CC 2=Toggle CC  Button BU1(B1, 0, 60, 1, 5 ); //60 means C4 of keyboard  Button BU2(B2, 0, 62, 1, 5 );//62 means D4 of keyboard  Button BU3(B3, 0, 64, 1, 5 );//...  Button BU4(B4, 0, 65, 1, 5 );  Button BU5(B5, 0, 67, 1, 5 );  Button BU6(B6, 0, 69, 1, 5 );//69 means A4 of keyboard  Button BU7(B7, 0, 71, 1, 5 );  Button BU8(B8, 0, 72, 1, 5 );//72 means C% of keyboard  Button BU9(B9, 0, 74, 1, 5 );  Button BU10(B10, 0, 76, 1, 5 );  //array  Button \*BUTTONS[] {&BU1, &BU2, &BU3, &BU4, &BU5, &BU6, &BU7, &BU8,&BU9,&BU10};  void setup() {  MIDI.begin(MIDI\_CHANNEL\_OFF);  }  void loop() {  if (NUMBER\_BUTTONS != 0) updateButtons();  if (NUMBER\_POTS != 0) updatePots();  }  void updateButtons() {  // Cycle through Button array  for (int i = 0; i < NUMBER\_BUTTONS; i = i + 1) {  byte message = BUTTONS[i]->getValue();  // Button is pressed  if (message == 0) {  switch (BUTTONS[i]->Bcommand) {  case 0: //Note  MIDI.sendNoteOn(BUTTONS[i]->Bvalue, 127, BUTTONS[i]->Bchannel);  break;  case 1: //CC  MIDI.sendControlChange(BUTTONS[i]->Bvalue, 127, BUTTONS[i]->Bchannel);  break;  case 2: //Toggle  if (BUTTONS[i]->Btoggle == 0) {  MIDI.sendControlChange(BUTTONS[i]->Bvalue, 127, BUTTONS[i]->Bchannel);  BUTTONS[i]->Btoggle = 1;  }  else if (BUTTONS[i]->Btoggle == 1) {  MIDI.sendControlChange(BUTTONS[i]->Bvalue, 0, BUTTONS[i]->Bchannel);  BUTTONS[i]->Btoggle = 0;  }  break;  }  }  // Button is not pressed  if (message == 1) {  switch (BUTTONS[i]->Bcommand) {  case 0:  MIDI.sendNoteOff(BUTTONS[i]->Bvalue, 0, BUTTONS[i]->Bchannel);  break;  case 1:  MIDI.sendControlChange(BUTTONS[i]->Bvalue, 0, BUTTONS[i]->Bchannel);  break;  }  }  }  }  void updatePots() {  for (int i = 0; i < NUMBER\_POTS; i = i + 1) {  byte potmessage = POTS[i]->getValue();  if (potmessage != 255) MIDI.sendControlChange(POTS[i]->Pcontrol, potmessage, POTS[i]->Pchannel);  }  } |

|  |
| --- |
| Controller.cpp |
| #include "Controller.h"  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Mux::Mux(byte outpin\_, byte numPins\_, bool analog\_)  {  outpin = outpin\_;  //enablepin = enablepin\_;  numPins = numPins\_;  analog = analog\_;  if (analog == false) pinMode(outpin, INPUT\_PULLUP);  //pinMode(enablepin, OUTPUT);  pinMode(2, OUTPUT);  pinMode(3, OUTPUT);  pinMode(4, OUTPUT);  if (numPins > 8) pinMode(5, OUTPUT);  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //Button (Pin Number, Command, Note Number, Channel, Debounce Time)  Button::Button(byte pin, byte command, byte value, byte channel, byte debounce)  {  \_pin = pin;  pinMode(\_pin, INPUT\_PULLUP);  \_value = value;  \_command = command;  \_debounce = debounce;  \_time = 0;  \_busy = false;  \_status = 0b00000010;  \_last = 1;  Bcommand = command;  Bvalue = value;  Bchannel = channel;  Btoggle = 0;  }  Button::Button(Mux mux, byte muxpin, byte command, byte value, byte channel, byte debounce)  {  \_pin = mux.outpin;  \_numMuxPins = mux.numPins;  \_muxpin = muxpin;  \_value = value;  \_command = command;  \_debounce = debounce;  \_time = 0;  \_busy = false;  \_status = 0b00000010;  \_last = 1;  Bcommand = command;  Bvalue = value;  Bchannel = channel;  Btoggle = 0;  }  void Button::muxUpdate()  {  byte temp = \_muxpin;  temp = temp << 2;  if (\_numMuxPins > 8) PORTD = PORTD & B11000011;  else PORTD = PORTD & B11100011;  PORTD = PORTD | temp;  }  byte Button::getValue()  {  // If BUSY bit not set - read button  if (bitRead(\_status, 0) == false) { // If busy false  if (digitalRead(\_pin) == \_last) return 2; // If same as last state - exit  }  // If NEW Bit set - Key just pressed, record time  if (bitRead(\_status, 1) == true) { // If new is true  bitSet(\_status, 0); // Set busy TRUE  bitClear(\_status, 1); // Set New FALSE  \_time = millis();  return 255;  }  // Check if debounce time has passed - If no, exit  if (millis() - \_time < \_debounce) return 255;  // Debounce time has passed. Read pin to see if still set the same  // If it has changed back - assume false alarm  if (digitalRead(\_pin) == \_last) {  bitClear(\_status, 0); // Set busy false  bitSet(\_status, 1); // Set new true  return 255;  }  // If this point is reached, event is valid. return event type  else {  bitClear(\_status, 0); // Set busy false  bitSet(\_status, 1); // Set new true  \_last = ((~\_last) & 0b00000001); // invert \_last  return \_last;  }  }  void Button::newValue(byte command, byte value, byte channel)  {  Bvalue = value;  Bcommand = command;  Bchannel = channel;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Pot::Pot(byte pin, byte command, byte control, byte channel)  {  \_pin = pin;  \_control = control;  \_value = analogRead(\_pin);  \_value = \_value >> 3;  \_oldValue = \_value << 3;  \_value = \_value << 3;  Pcommand = command;  Pcontrol = control;  Pchannel = channel;  }  void Pot::muxUpdate()  {  byte temp = \_muxpin;  temp = temp << 2;  if (\_numMuxPins > 8) PORTD = PORTD & B11000011;  else PORTD = PORTD & B11100011;  //PORTD = PORTD & B11000011;  PORTD = PORTD | temp;  }  Pot::Pot(Mux mux, byte muxpin, byte command, byte control, byte channel)  {  \_pin = mux.outpin;  \_numMuxPins = mux.numPins;  \_muxpin = muxpin;  \_control = control;  muxUpdate();  \_value = analogRead(\_pin);  \_value = \_value >> 3;  \_oldValue = \_value << 3;  \_value = \_value << 3;  Pcommand = command;  Pcontrol = control;  Pchannel = channel;  }  byte Pot::getValue()  {  \_value = analogRead(\_pin);  int tmp = (\_oldValue - \_value);  if (tmp >= 8 || tmp <= -8) {  \_oldValue = \_value >> 3;  \_oldValue = \_oldValue << 3;  return \_value >> 3;  }  return 255;  }  void Pot::newValue(byte command, byte value, byte channel) {  Pcommand = command;  Pcontrol = value;  Pchannel = channel;  }  //Controller.h 생략 |

처음 코드를 아두이노에 올리고 케이블을 연결했을때 처리속도의 문제인지 셧다운되는 문제가 있었습니다. 그래서 영향을 주지 않는 선에서 필요없는 코드들은 정리를 했고, 잘 구동된다는 것을 확인하였습니다. 포텐셔미터의 값은 새로운 값이 읽혀지는 경우에만 전달되도록 했고 버튼의 경우 노트를 순서대로 매핑해 주었으나 변경될 수 있도록 하였습니다.

6.결과 및 동영상

|  |
| --- |
| ../Downloads/KakaoTalk_Photo_2016-12-17-02-38-31_74.jpeg  <https://www.youtube.com/watch?v=Q83aIwn69yk> :시연 동영상  <https://www.youtube.com/watch?v=lgkfeq8VbKE> :미디 테스트  <https://www.youtube.com/watch?v=1XfN2XwowE0> :버튼 테스트  <https://www.youtube.com/watch?v=76slEKvYhzI> :포텐셔미터 테스트 |

7.[부록] 참고 자료

<http://www.instructables.com/id/Easy-3-Pot-Potentiometer-Arduino-Uno-Effects-Midi-/>

<http://www.maximporges.com/arduinome/>

<http://www.notesandvolts.com/2016/03/arduino-midi-controller-potentiometers.html>