▣ OLED LCD DRVIER SSD1306 DATASHEET SSD1306 분석

2020.11.07 강경수

```
1   import time
2   import Adafruit_SSD1306
3   from PIL import Image, ImageDraw, ImageFont
4
5   import spidev
6
7   ldr_channel = 0
8
9   spi = spidev.SpiDev()
10  spi.open(0,0)
11  spi.max_speed_hz = 100000
12
13  disp = Adafruit_SSD1306.SSD1306_128_64(rst=None,i2c_address=0x3C)
14  disp.begin()
15
16  disp.clear()
17  disp.display()
18
19  width = disp.width
20  height = disp.height
21  image = Image.new('1',(width,height))
```

→ SSD1306 CLASS에 대한 선언

H/W LEVEL RST신호 사용하므로

RST → NONE

I2C_address → 0x3C

disp class 선언시 실행되는 생성자

```
def __init__(self, width, height, rst, dc=None, sclk=None, din=None, cs=None,
             gpio=None, spi=None, i2c_bus=None, i2c_address=SSD1306_I2C_ADDRES
             i2c=None):
    self._log = logging.getLogger('Adafruit_SSD1306.SSD1306Base')
    self._spi = None
    self._i2c = None
    self.width = width
    self.height = height
    self._pages = height//8
    self._buffer = [0]*(width*self._pages)
    # Default to platform GPIO if not provided.
    self._gpio = gpio
    if self._gpio is None:
        self._gpio = GPIO.get_platform_gpio()
    # Setup reset pin.
    self._rst = rst
    if not self._rst is None:
        self._gpio.setup(self._rst, GPIO.OUT)
    # Handle hardware SPI
    if spi is not None:
        self._log.debug('Using hardware SPI')
        self._spi = spi
        self._spi.set_clock_hz(8000000)
    elif sclk is not None and din is not None and cs is not None:
        self._log.debug('Using software SPI')
        self._spi = SPI.BitBang(self._gpio, sclk, din, None, cs)
    # Handle hardware I2C
    elif i2c is not None:
        self._log.debug('Using hardware I2C with custom I2C provider.')
        self._i2c = i2c.get_i2c_device(i2c_address)
    else:
        self._log.debug('Using hardware I2C with platform I2C provider.')
        import Adafruit_GPIO.I2C as I2C
        if i2c_bus is None:
            self._i2c = I2C.get_i2c_device(i2c_address)
        else:
            self._i2c = I2C.get_i2c_device(i2c_address, busnum=i2c_bus)
    # Initialize DC pin if using SPI.
    if self._spi is not None:
        if dc is None:
            raise ValueError('DC pin must be provided when using SPI.')
        self._dc = dc
        self._gpio.setup(self._dc, GPIO.OUT)
```

```
def begin(self, vccstate=SSD1306_SWITCHCAPVCC):
    """Initialize display."""
    # Save vcc state.
    self._vccstate = vccstate
    # Reset and initialize display.
    self.reset()
    self._initialize()
    # Turn on the display.
    self.command(SSD1306_DISPLAYON)
```

disp.begin() 함수

self.reset() 관련 내용

a) Slave address bit (SA0)
SSD1306 has to recognize the slave address before transmitting or receiving any information by the I^2C-bus. The device will respond to the slave address following by the slave address bit ("SA0" bit) and the read/write select bit ("R/W#" bit) with the following byte format,

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0	1	1	1	1	0	SA0	R/W#

"SA0" bit provides an extension bit for the slave address. Either "0111100" or "0111101", can be selected as the slave address of SSD1306. D/C# pin acts as SA0 for slave address selection.
"R/W#" bit is used to determine the operation mode of the I^2C-bus interface. R/W#=1, it is in read mode. R/W#=0, it is in write mode.

b) I^2C-bus data signal (SDA)
SDA acts as a communication channel between the transmitter and the receiver. The data and the acknowledgement are sent through the SDA.
It should be noticed that the ITO track resistance and the pulled-up resistance at "SDA" pin becomes a voltage potential divider. As a result, the acknowledgement would not be possible to attain a valid logic 0 level in "SDA".
"SDA_{IN}" and "SDA_{OUT}" are tied together and serve as SDA. The "SDA_{IN}" pin must be connected to act as SDA. The "SDA_{OUT}" pin may be disconnected. When "SDA_{OUT}" pin is disconnected, the acknowledgement signal will be ignored in the I^2C-bus.

c) I^2C-bus clock signal (SCL)
The transmission of information in the I^2C-bus is following a clock signal, SCL. Each transmission of data bit is taken place during a single clock period of SCL.

SLAVE ADDRESS 0x3C or 0x3D
D/C# PIN 1 -> 0x3C

SDAout 과 SDA in을 한 노드로 묶어 풀업저항을 걸어줘야 함을 나타낸다. 하지만 이미 MODULE차원에서 다 묶여있음

Table 8-1 : MCU interface assignment under different bus interface mode

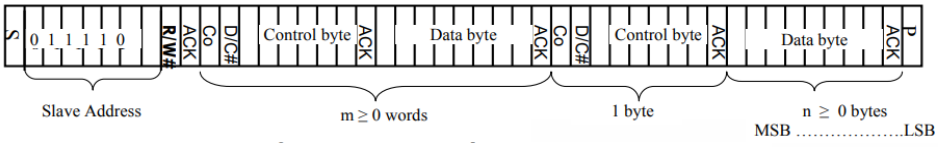
Bus Interface \ Pin Name	Data/Command Interface								Control Signal				
	D7	D6	D5	D4	D3	D2	D1	D0	E	R/W#	CS#	D/C#	RES#
8-bit 8080	D[7:0]								RD#	WR#	CS#	D/C#	RES#
8-bit 6800	D[7:0]								E	R/W#	CS#	D/C#	RES#
3-wire SPI	Tie LOW					NC	SDIN	SCLK	Tie LOW		CS#	Tie LOW	RES#
4-wire SPI	Tie LOW					NC	SDIN	SCLK	Tie LOW		CS#	D/C#	RES#
I^2C	Tie LOW					SDA_{OUT}	SDA_{IN}	SCL	Tie LOW			SA0	RES#

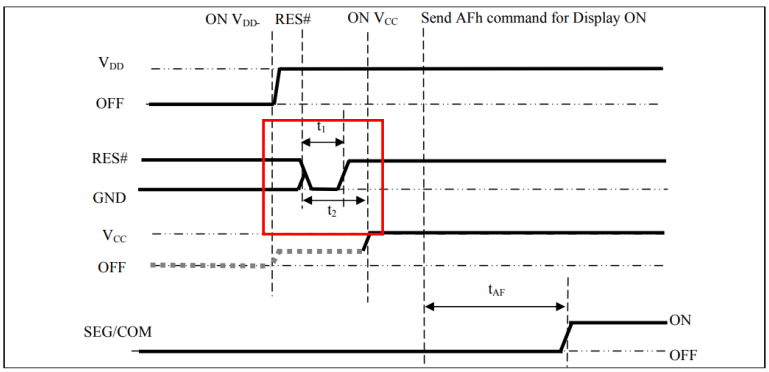
command 명령어를 통해 I2C WRITE 하는 것을 확인 할 수 있다.

Note:
Co – Continuation bit
D/C# – Data / Command Selection bit
ACK – Acknowledgement
SA0 – Slave address bit
R/W# – Read / Write Selection bit
S – Start Condition / P – Stop Condition

엮과 같은 통신 프로토콜로 제어함

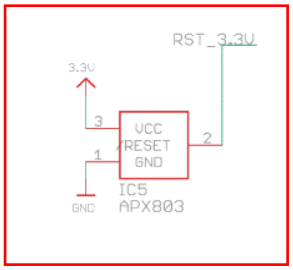
Write mode





Q.I2C 별도의 RESET 연결 안 했는데?

A. MODULE H/W회로 구성이 자동으로 RESET되게 구성



8.5 Reset Circuit

When RES# input is LOW, the chip is initialized with the following status:

1. Display is OFF
2. 128 x 64 Display Mode
3. Normal segment and display data column address and row address mapping (SEG0 mapped address 00h and COM0 mapped to address 00h)
4. Shift register data clear in serial interface
5. Display start line is set at display RAM address 0
6. Column address counter is set at 0
7. Normal scan direction of the COM outputs
8. Contrast control register is set at 7Fh
9. Normal display mode (Equivalent to A4h command)

-> H/W Reset시 자동 수행되는 내용들..

self.initalize() 관련 내용

Power ON sequence:

1. Power ON V_{DD}
2. After V_{DD} become stable, set RES# pin LOW (logic low) for at least 3us (t_1) [4] and then HIGH (logic high).
3. After set RES# pin LOW (logic low), wait for at least 3us (t_2). Then Power ON V_{CC}.[1]
4. After V_{CC} become stable, send command AFh for display ON. SEG/COM will be ON after 100ms (t_{AF}).

H/W Reset 이후 class 선언시 자동 실행되는 초기화 코드들

```
def _initialize(self):
    # 128x64 pixel specific initialization.
    self.command(SSD1306_DISPLAYOFF)                    # 0xAE   일단 DISPLAY OFF
    self.command(SSD1306_SETDISPLAYCLOCKDIV)            # 0xD5   DISPLAY CLOCK WRITE 명령
    self.command(0x80)                                  # the suggested ratio 0x80 ALL RESET
    self.command(SSD1306_SETMULTIPLEX)                  # 0xA8   MUX SETTING WRITE 명령
    self.command(0x3F)                                  #        64MUX
    self.command(SSD1306_SETDISPLAYOFFSET)              # 0xD3   OFFSET WRITE 명령
    self.command(0x0)                                   # no offset NO OFFSET
    self.command(SSD1306_SETSTARTLINE | 0x0)            # line #0
    self.command(SSD1306_CHARGEPUMP)                    # 0x8D
```

D/C#	Hex	D7	D6	D5	D4	D3	D2	D1	D0	Command	Description
0	D5	1	1	0	1	0	1	0	1	Set Display Clock Divide Ratio/Oscillator Frequency	A[3:0] : Define the divide ratio (D) of the display clocks (DCLK): Divide ratio= A[3:0] + 1, RESET is 0000b (divide ratio = 1) A[7:4] : Set the Oscillator Frequency, F_{OSC}. Oscillator Frequency increases with the value of A[7:4] and vice versa. RESET is 1000b Range:0000b~1111b Frequency increases as setting value increases.
0	A[7:0]	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		

```
def command(self, c):
    """Send command byte to display."""
    if self._spi is not None:
        # SPI write.
        self._gpio.set_low(self._dc)
        self._spi.write([c])
    else:
        # I2C write.
        control = 0x00   # Co = 0, DC = 0
        self._i2c.write8(control, c)
```

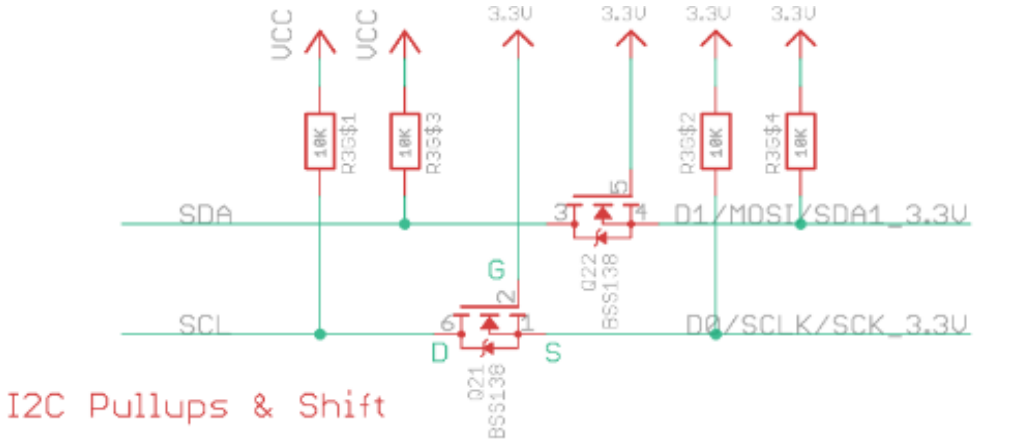
command 명령어를 통해 I2C WRITE 하는 것을 확인 할 수 있다.

I2C HW구성

SDA,SCL OPEN DRAIN 채널

10k 저항을 통해 PULL UP

※풀업저항 계산 : SLVA689 참조



```python
def display(self):
    """Write display buffer to physical display."""
    self.command(SSD1306_COLUMNADDR)
    self.command(0)              # Column start address. (0 = reset)
    self.command(self.width-1)   # Column end address.
    self.command(SSD1306_PAGEADDR)
    self.command(0)              # Page start address. (0 = reset)
    self.command(self._pages-1)  # Page end address.
    # Write buffer data.
    if self._spi is not None:
        # Set DC high for data.
        self._gpio.set_high(self._dc)
        # Write buffer.
        self._spi.write(self._buffer)
    else:
        for i in range(0, len(self._buffer), 16):
            control = 0x40   # Co = 0, DC = 0
            self._i2c.writeList(control, self._buffer[i:i+16])
```

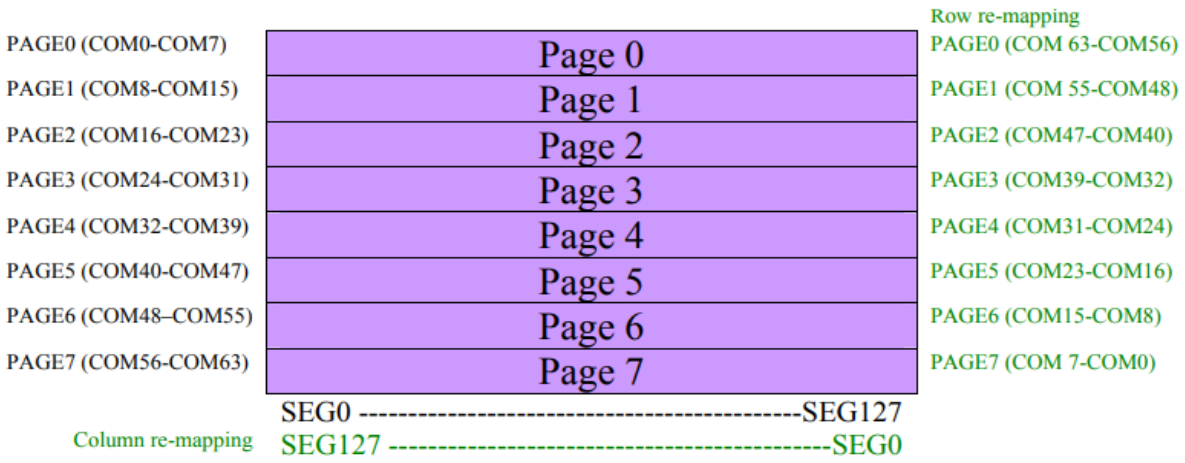
이미지를 그리는
여러가지 모드중
PAGEADDRESS모드 사용

class.dislplay() 함수 내용.

COLUMADDR지정하고
DISPLAY 방법
PAGEADDRES 사용

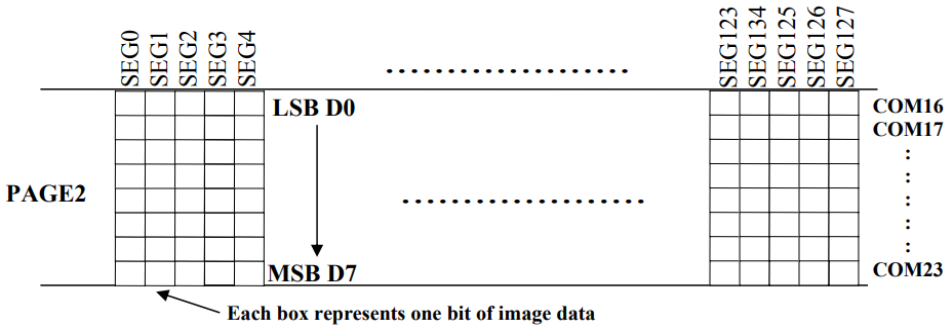
spi가 정의 돼있지 않으면
자동으로 i2c protocol통신 함.

Figure 8-13 : GDDRAM pages structure of SSD1306



PAGE의 구조

Figure 8-14 : Enlargement of GDDRAM (No row re-mapping and column-remapping)



1개의 PAGE는
8개의 COM과
128개의 SEEG로 구성

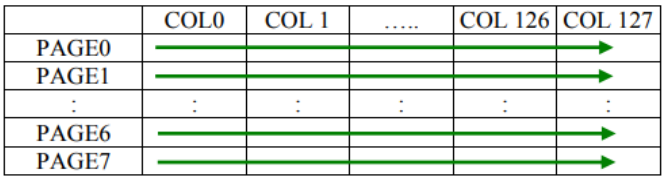
따라서 원하는
이미지를 띄우기
위해서는 SEG, COM
PAGE를 조정해가며
이미지를 DISPLAY

※ PAGE ADDRESS MODE에 대한 요약

Page addressing mode (A[1:0]=10xb)

In page addressing mode, after the display RAM is read/written, the column address pointer is increased automatically by 1. If the column address pointer reaches column end address, the column address pointer is reset to column start address and page address pointer is not changed. Users have to set the new page and column addresses in order to access the next page RAM content. The sequence of movement of the PAGE and column address point for page addressing mode is shown in Figure 10-1.

Figure 10-1 : Address Pointer Movement of Page addressing mode



In normal display data RAM read or write and page addressing mode, the following steps are required to define the starting RAM access pointer location:

- Set the page start address of the target display location by command B0h to B7h.
- Set the lower start column address of pointer by command 00h~0Fh.
- Set the upper start column address of pointer by command 10h~1Fh.

For example, if the page address is set to B2h, lower column address is 03h and upper column address is 10h, then that means the starting column is SEG3 of PAGE2. The RAM access pointer is located as shown in Figure 10-2. The input data byte will be written into RAM position of column 3.

(1) COL은 자동적으로 1씩 증가함

(2) COL이 127에 도달시 0으로 RESET

(3) PAGE는 증가하지 않으므로 사용자가 PAGE를 바꿔줘야 함

```
draw = ImageDraw.Draw(image)
draw.rectangle((0,0,width,height),outline = 0, fill = 0)
```

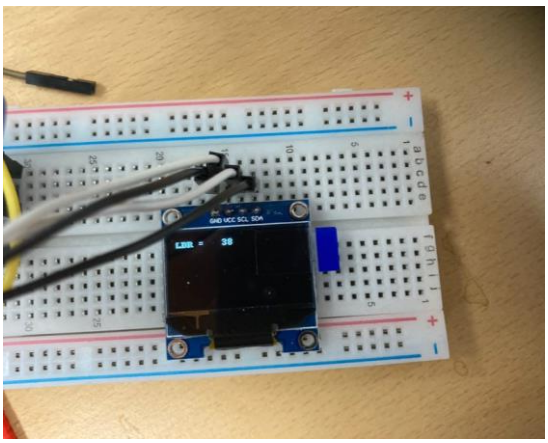
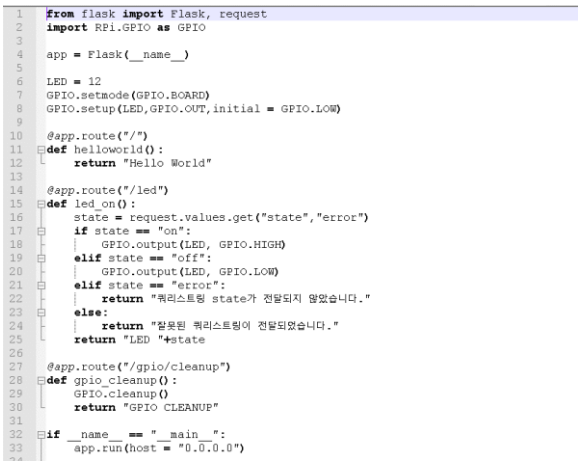
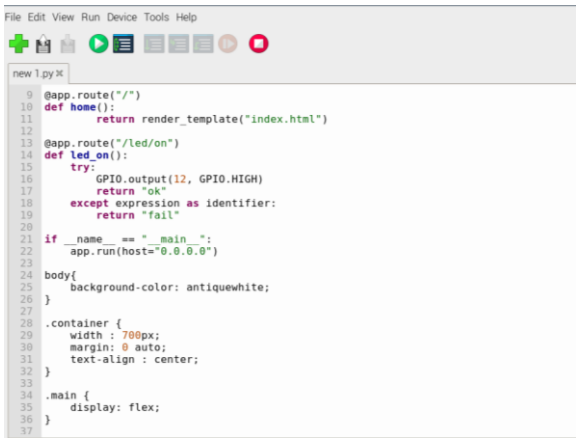
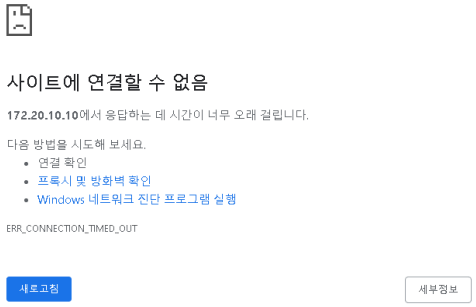
PIL

IMAGE LIBRARY 함수

```
padding = -2
top = padding
bottom = height - padding

x = 0
font = ImageFont.load_default()

def readadc(adcnum):
    if adcnum>7 or adcnum<0:
        return -1
    r = spi.xfer2([1,8 + adcnum<<4,0])
    data = ((r[1] & 3) <<8)+r[2]
    return data

while True:
    ldr_value = readadc(ldr_channel)
    print('ldr value = %d' % ldr_value)

    draw.rectangle((0,0,width,height), outline = 0, fill = 0)
    draw.text((x,top), 'LDR = {0000:4d}'.format(ldr_value), font = font, fill = 255)

    disp.image(image)
    disp.display()
    time.sleep(2)

draw = ImageDraw.Draw(image)

draw.rectangle((0,0,width,height), outline = 0, fill = 0)
```

▣ Raspberrypi 실습

2020.11.10 강경수

사 진	사 진
	
	
	

```
padding = -2
top = padding
bottom = height - padding

x = 0

font = ImageFont.load_default()

def readadc(adcnum):
    if adcnum>7 or adcnum<0:
        return -1
    r = spi.xfer2([1,8 + adcnum<<4,0])
    data = ((r[1] & 3) <<8)+r[2]
    return data

while True:
    ldr_value = readadc(ldr_channel)
    print('ldr value = %d' % ldr_value)

    draw.rectangle((0,0,width,height), outline = 0, fill = 0)
    draw.text((x,top), 'LDR = {0000:4d}'.format(ldr_value), font = font, fill = 255)

    disp.image(image)
    disp.display()
    time.sleep(2)

draw = ImageDraw.Draw(image)
```

```
from flask import Flask, request
import RPi.GPIO as GPIO

app = Flask(__name__)

LED = 12
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED,GPIO.OUT,initial = GPIO.LOW)

@app.route("/")
def helloworld():
    return "Hello World"

@app.route("/led")
def led_on():
    state = request.values.get("state","error")
    if state == "on":
        GPIO.output(LED, GPIO.HIGH)
    elif state == "off":
        GPIO.output(LED, GPIO.LOW)
    elif state == "error":
        return "쿼리스트링 state가 전달되지 않았습니다."
    else:
        return "잘못된 쿼리스트링이 전달되었습니다."
    return "LED "+state

@app.route("/gpio/cleanup")
def gpio_cleanup():
    GPIO.cleanup()
    return "GPIO CLEANUP"

if __name__ == "__main__":
    app.run(host = "0.0.0.0")
```

사이트에 연결할 수 없음

172.20.10.10에서 응답하는 데 시간이 너무 오래 걸립니다.

다음 방법을 시도해 보세요.

- 연결 확인
- 프록시 및 방화벽 확인
- Windows 네트워크 진단 프로그램 실행

ERR_CONNECTION_TIMED_OUT

새로고침 세부정보

```
@app.route("/")
def home():
        return render_template("index.html")

@app.route("/led/on")
def led_on():
    try:
        GPIO.output(12, GPIO.HIGH)
        return "ok"
    except expression as identifier:
        return "fail"

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

```
body{
    background-color: antiquewhite;
}

.container {
    width : 700px;
    margin: 0 auto;
    text-align : center;
}

.main {
    display: flex;
}
```

사이트에 연결할 수 없음

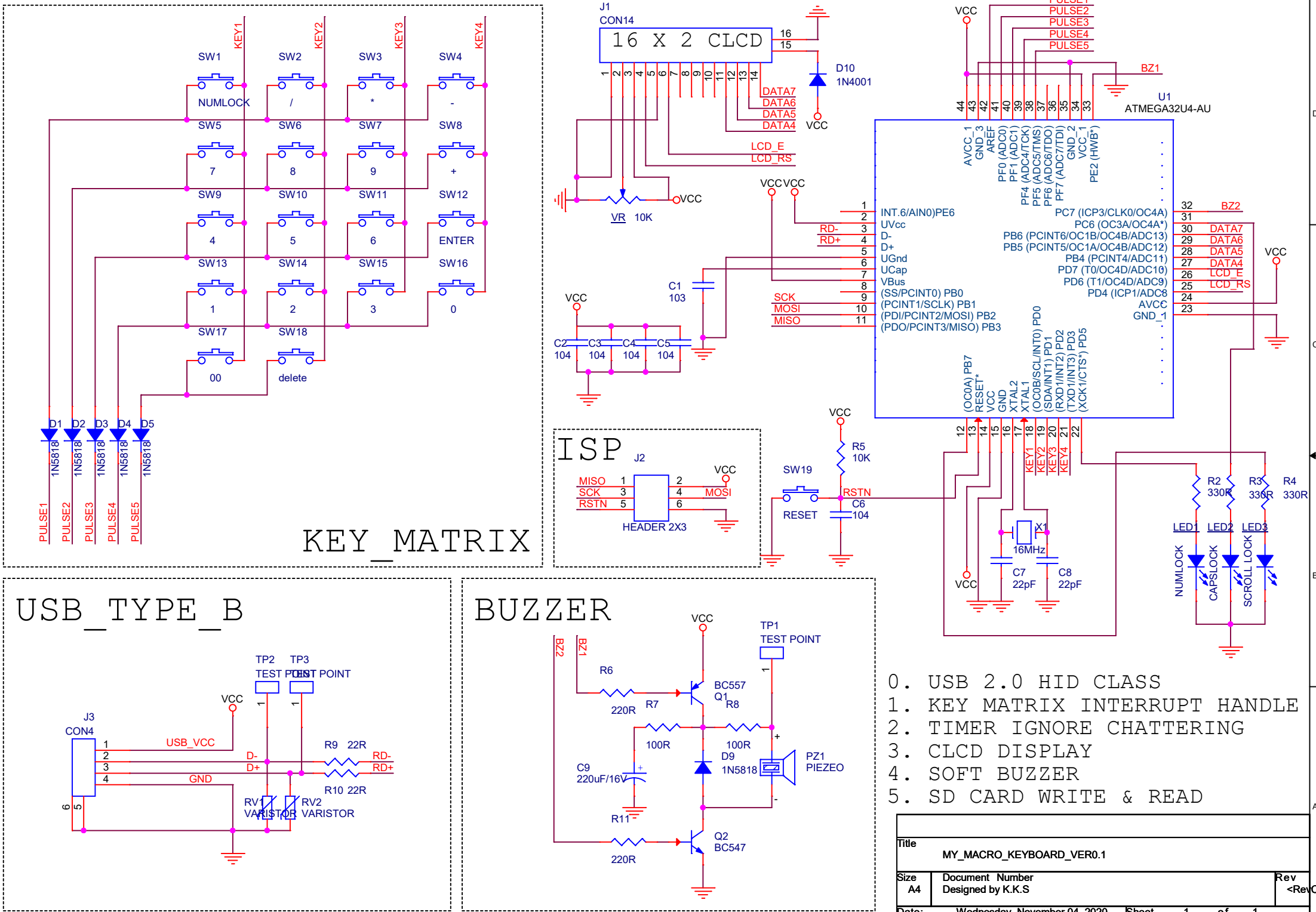
172.20.10.10에서 응답하는 데 시간이 너무 오래 걸립니다.

다음 방법을 시도해 보세요.

- 연결 확인
- 프록시 및 방화벽 확인
- Windows 네트워크 진단 프로그램 실행

ERR_CONNECTION_TIMED_OUT

새로고침 세부정보


KEY_MATRIX
SW1
SW2
SW3
SW4
KEY1
KEY2
KEY3
KEY4
NUMLOCK
/
*
-
7
8
9
+
4
5
6
ENTER
1
2
3
0
00
delete
D1 D2 D3 D4 D5
1N5818
PULSE1
PULSE2
PULSE3
PULSE4
PULSE5
J1
CON14
16 X 2 CLCD
DATA7
DATA6
DATA5
DATA4
LCD_E
LCD_RS
VR 10K
D10
1N4001
U1
ATMEGA32U4-AU
BZ1
BZ2
C1 103
C2 104
C3 104
C4 104
C5 104
SCK
MOSI
MISO
ISP
J2
HEADER 2X3
SW19
RESET
R5 10K
C6 104
X1
16MHz
C7 22pF
C8 22pF
R2 330R
R3 330R
R4 330R
LED1
LED2
LED3
NUMLOCK
CAPSLOCK
SCROLL LOCK
USB_TYPE_B
J3
CON4
USB_VCC
D-
D+
GND
R9 22R
R10 22R
RD-
RD+
RV1 VARISTOR
RV2 VARISTOR
TP2 TEST POINT
TP3 TEST POINT
BUZZER
TP1 TEST POINT
R6
R7
R8
220R
100R
BC557
Q1
Q2
BC547
C9 220uF/16V
R11
D9 1N5818
PZ1 PIEZEO
0. USB 2.0 HID CLASS
1. KEY MATRIX INTERRUPT HANDLE
2. TIMER IGNORE CHATTERING
3. CLCD DISPLAY
4. SOFT BUZZER
5. SD CARD WRITE & READ
Title
MY_MACRO_KEYBOARD_VER0.1
Size
A4
Document Number
Designed by K.K.S
Rev
Date:
Wednesday, November 04, 2020
Sheet
1
of
1

MACRO KEYBOARD PARTLIST

※ LEONARDO BOARD에 회로 검증 후 부품 변경 될 수 있음 **작성자 : 강경수**

※ SD카드 추가 필요 **작성일 : 201102**

ITEM	PARTNAME	DESCRITPION	QTY	LOCATION	VENDOR
1	CERAMIC CAPACITOR	103/50V	1	C1	ANY
2	CERAMIC CAPACITOR	104/50V	5	C2,C3,C4,C5,C6	ANY
3	CERAMIC CAPACITOR	22pF/50V	2	C7,C8	ANY
4	ELEC. CAPACITOR	220uF/16V	1	C9	ANY
5	SWITCHING DIODE	1N5818	6	D1,D2,D3,D4,D5,D9	ANY
6	SWITCHING DIODE	1N4001	1	D10	ANY
7	14X1 HEADER	P 2.54mm	1	J1	ANY
8	2X3 HEADER	P 2.54mm	1	J2	ANY
9	670688000	USB - B CON	1	J3	MOLEX
10	SZH - SW038	SWITCH	18	SW1~SW18	CHERRY
11	NW3-A06-B3	TACK SWITCH	1	SW19	NW3
12	3BCLSW02	WHITE 3PI	3	LED1,LED2,LED3	DAKWANG
13	FQ - 030	PIEZEO BUZZER	1	PZ1	SMG
14	BC557	NPN TR	1	Q1	ONSEMI
15	BC547	PNP TR	1	Q2	ONSEMI
16	CG0603MLC-05E	VARISTOR	2	RV1,RV2	BOURNS
17	CHIP RESISTOR	330R,J,1608,1/4W	3	R2,R3,R4	TA-I
18	CHIP RESISTOR	10K,J,1608,1/4W	1	R5	TA-I
19	CHIP RESISTOR	22R,J,1608,1/4W	2	R7,R8	TA-I
20	CHIP RESISTOR	220R,J,1608,1/4W	2	R6,R11	TA-I
21	CHIP RESISTOR	100R,J,1608,1/4W	2	R7,R8	TA-I
22	CHIP RESISTOR	22R,J,1608,1/4W	2	R9,R10	TA-I
23	CRYSTAL	16Mhz, X-TAL	1	X1	caltron
24	MCU	ATMEGA32U4-AU	1	U1	Microchip
25	16X2 CLCD (LEFT UP)	LC1621-SMLYH6	1	LCD1	-