FPGA_AVR Communication Design

강원대학교 일반대학원
전자정보통신공학과 석사 박제창

- FPGA_AVR

- 개요

1. HBE-COMBO 또는 DE2-70 의 확장 포트에 AVR 보드를 연결하여 상호 인터페이스를 실험한다.

2. Atmega128L을 메인 디바이스로 사용하고 또한, 메인 디바이스와 각 센서 (온/습도, 거리, 조도)를 통해 주변의 환경 변화를 측정한다.

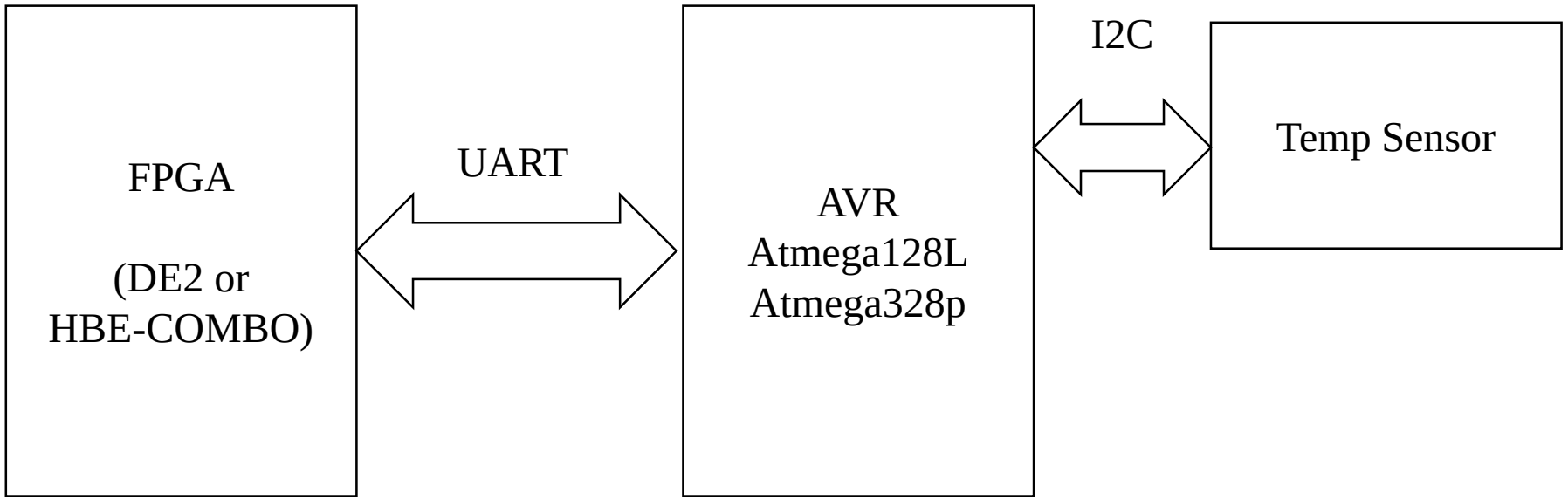
3. 상황에 적합한 알고리즘을 구현하여 Atmega128L 또는 그외의 프로세서에 프로그래밍 한다.

4. 구동 소스는 마이크로 컨트롤러에서 센서값을 1초 단위(지연을 통한)로 변환 된 값을 계속 측정하며 이를A/D 변환를 거쳐 FPGA 장치로 데이터를 전달한다.

5. FPGA 디바이스에서는 센서마다 데이터 값을 세그먼트에 표기한다.

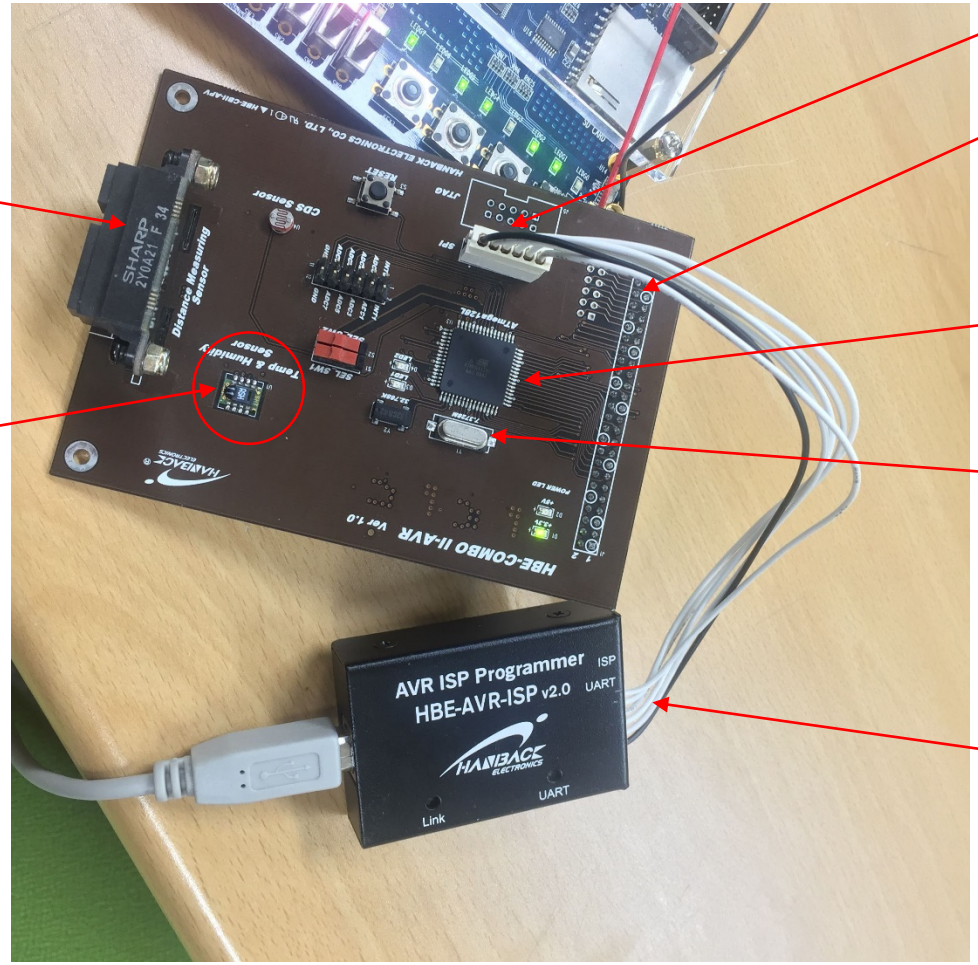
➢ FPGA_AVR

- System Block Diagram



➢ FPGA_AVR

- HanBack 전자의 AVR 보드



적외선 거리 센서

온/습도 센서

SPI Program Pin

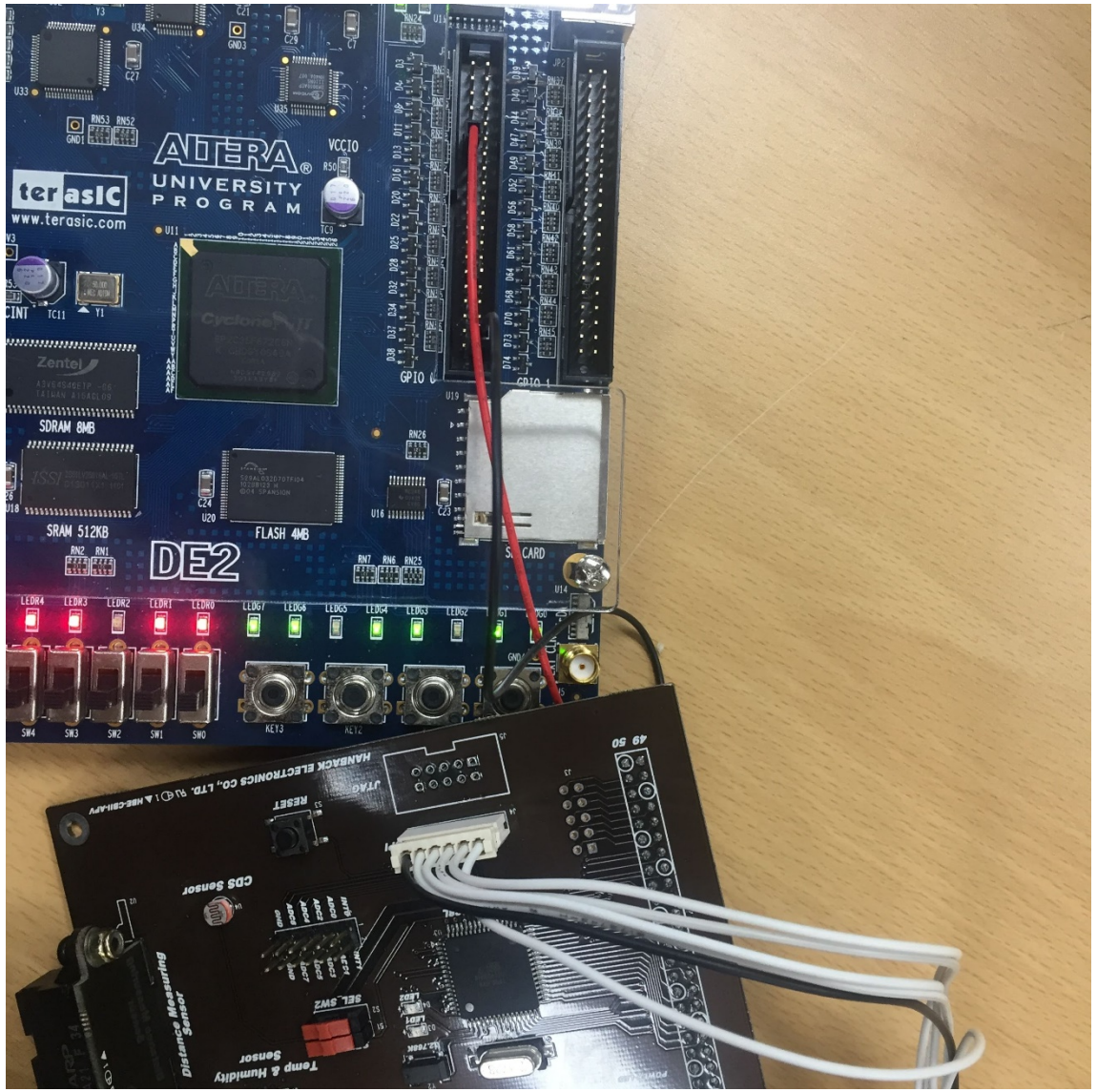
GPIO PORT

Atmega128L

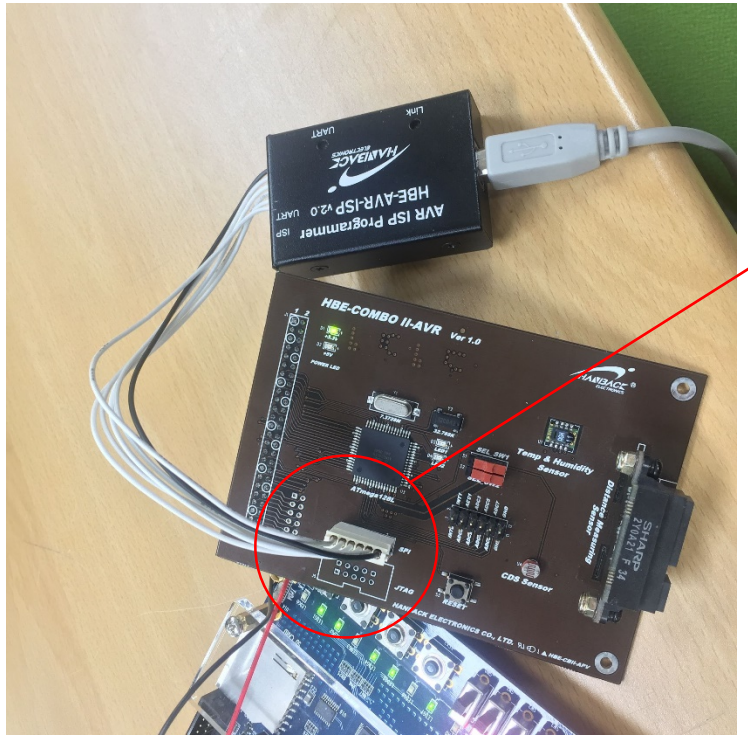
External Clock 주파수
7.3232MHz

Programmer
Debugger

➢ FPGA_AVR



한백 전자의 AVR 보드는 보드 핀에 전원을 인가해야 마이크로프로세서에 전원이 인가되며 단순히 프로그래머만 연결한다고 해서 플레시 메모리에 프로그램이 가능한 것은 아니다.



SPI통신으로 프로그래밍한다.

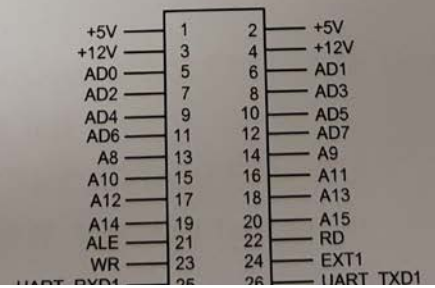
➢ FPGA_AVR

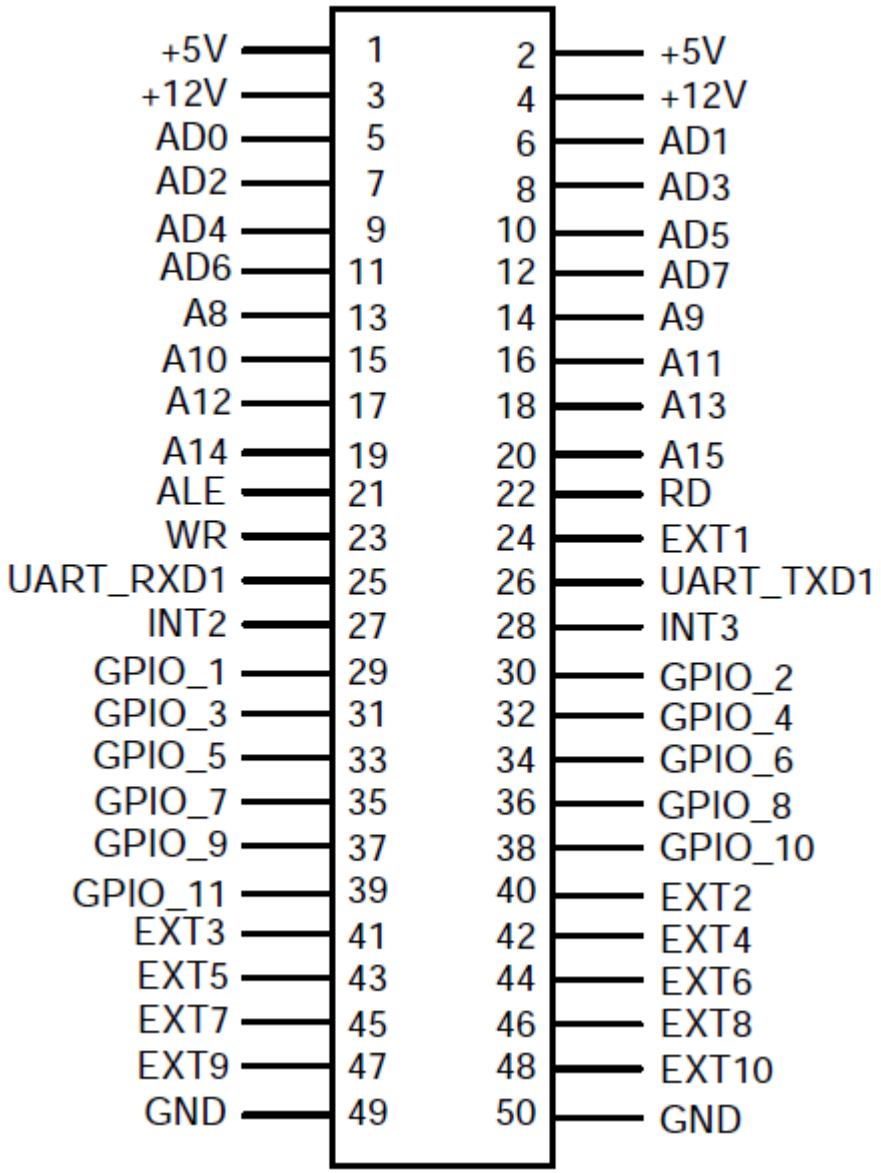
- CDS
  - Photo Resistance(10Lx) : 16 ~ 50 KΩ
  - Quick response

(다) 각 부분의 명칭 및 설명

① AVR 모듈 연결 커넥터

모듈의 50핀 확장 커넥터는 모듈에서 필요한 전원과 ATmega128과 FPGA 디바이스간의 연결을 위한 Address와 data가 연결되어 있다. 따라서 ATmega128에서 얻은 센서의 정보들을 처리하여 FPGA 디바이스로 전달하게 된다. 다음에서 50핀 확장 커넥터의 핀 정보를 확인해 볼 수 있다. 커넥터에서 핀 이름이 'AD'로 나와 있는 것은 ATmega128 디바이스의 port A로 Address의 하위 8 Byte와 data 8 Byte의 역할을 가지고 있다. 또한 핀 이름이 'A'로 정해진 것은 ATmega128 디바이스의 port C로 Address의 상위 8 Byte로 지정되어 있다. 또한 ATmega128의 기능적인 핀과 여분의 핀(GPIO)등이 FPGA 디바이스로 연결되어 있는 것을 알 수 있다.



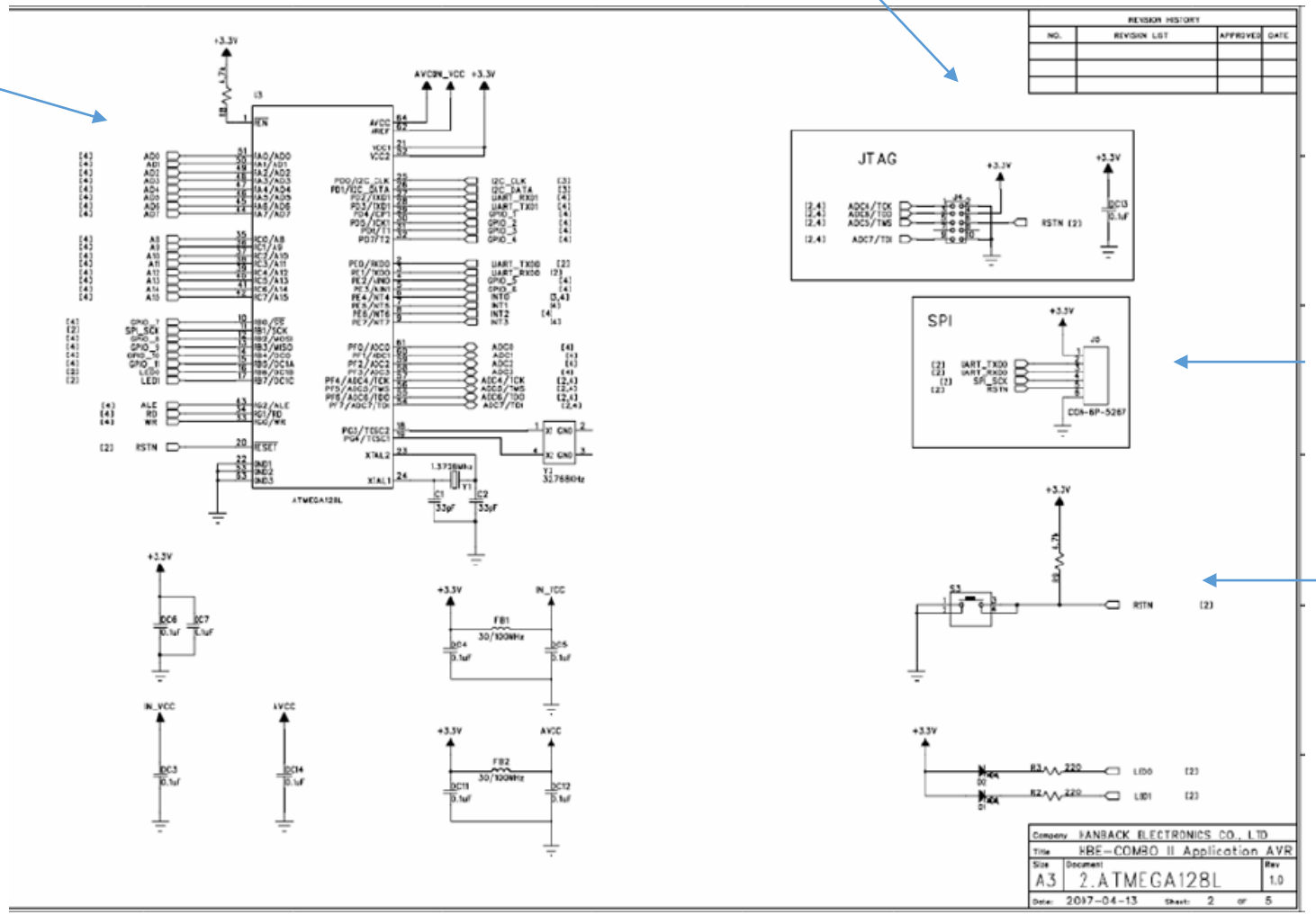


➢ FPGA_AVR

- Board Schematic

Atmega128L

JTAG



Program을 위한 SPI Port

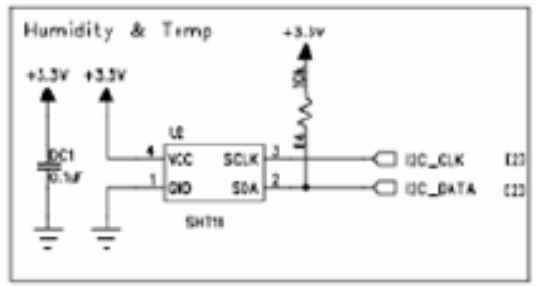
Reset SW

LED IND

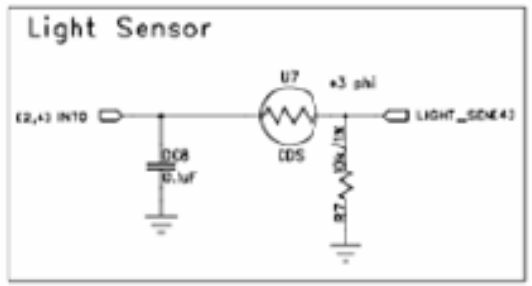
➢ FPGA_AVR

- Board Schematic

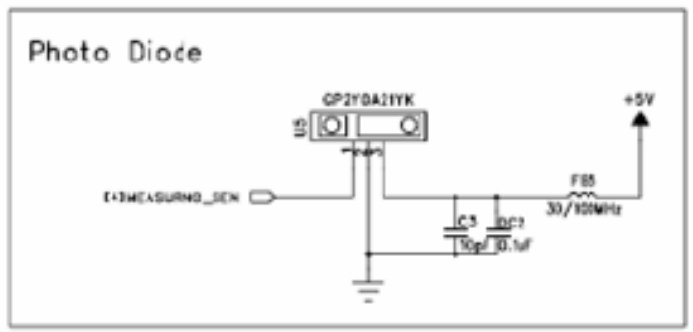
SHT 30 (온습도 센서)



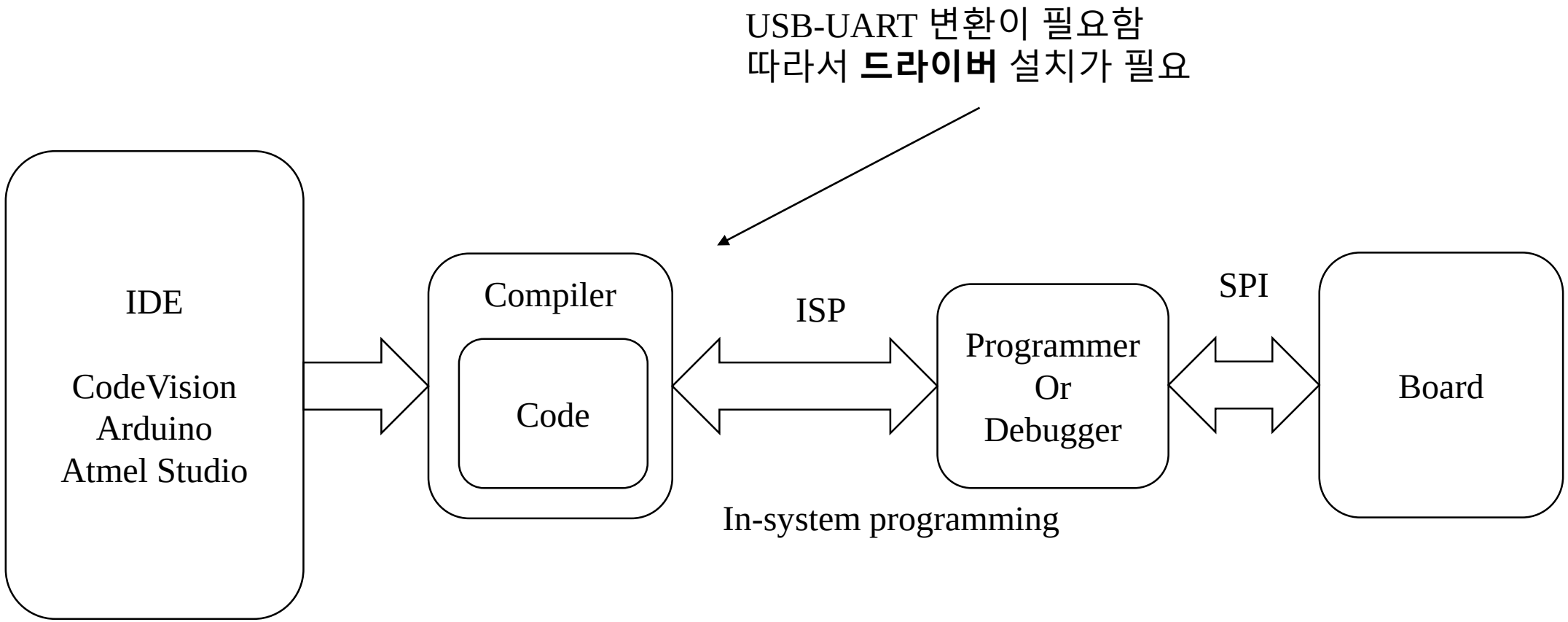
조도



거리



프로그램 방법 시스템 구조도



- FPGA_AVR

https://www.ftdichip.com/Drivers/VCP.htm

Currently Supported VCP Drivers:

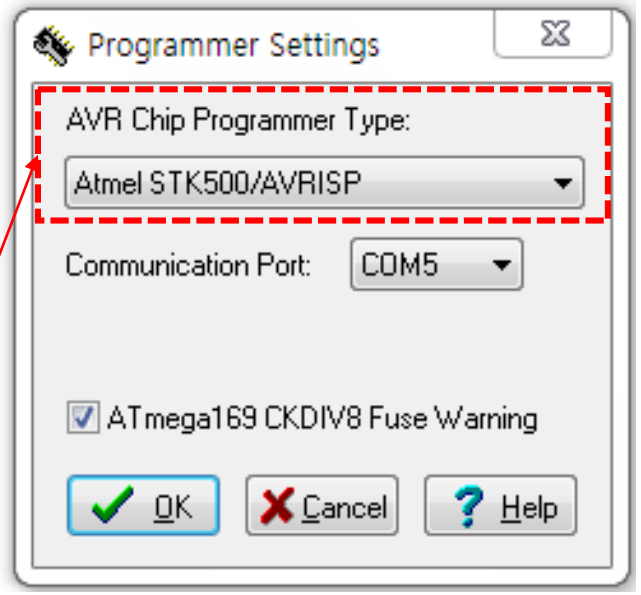
		Processor Architecture			
Operating System	Release Date	x86 (32-bit)	x64 (64-bit)	PPC	ARM
Windows*	2017-08-30	2.12.28	2.12.28	-	-
Linux	-	-	-	-	-

자신의 컴퓨터 시스템 운영체제에 적합한 Driver를 다운로드 받아 설치한다.

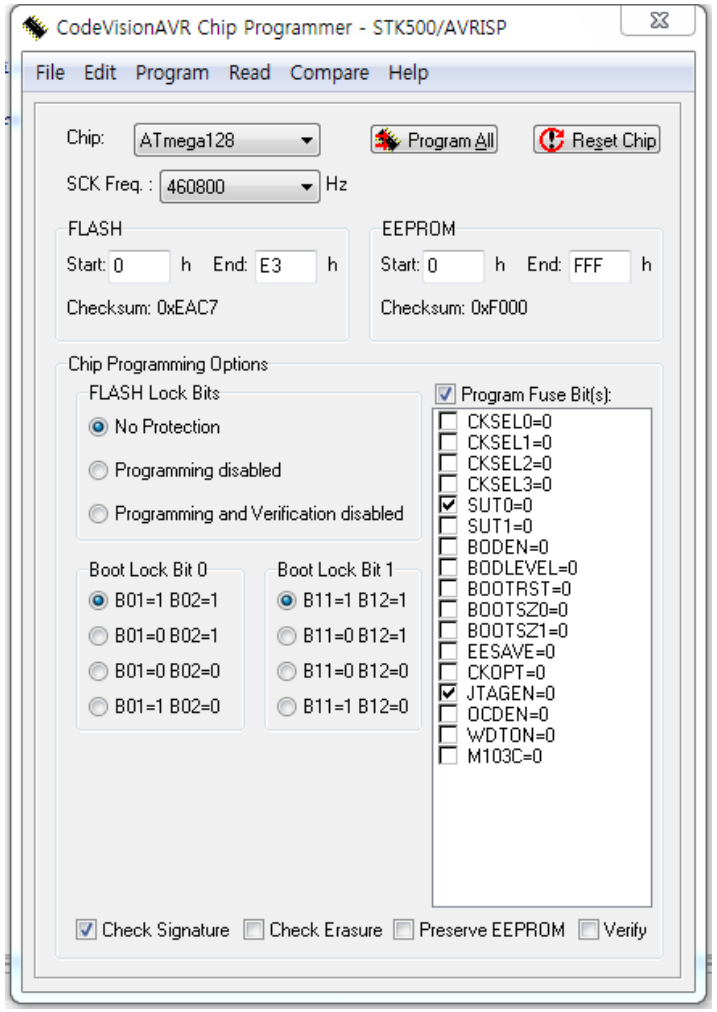
보통 UART-USB변환을 위해 FTDI 사의 FT232칩을 주로 사용하며 이외 가격이 저렴한 CH430 칩을 사용한다.

➢ FPGA_AVR

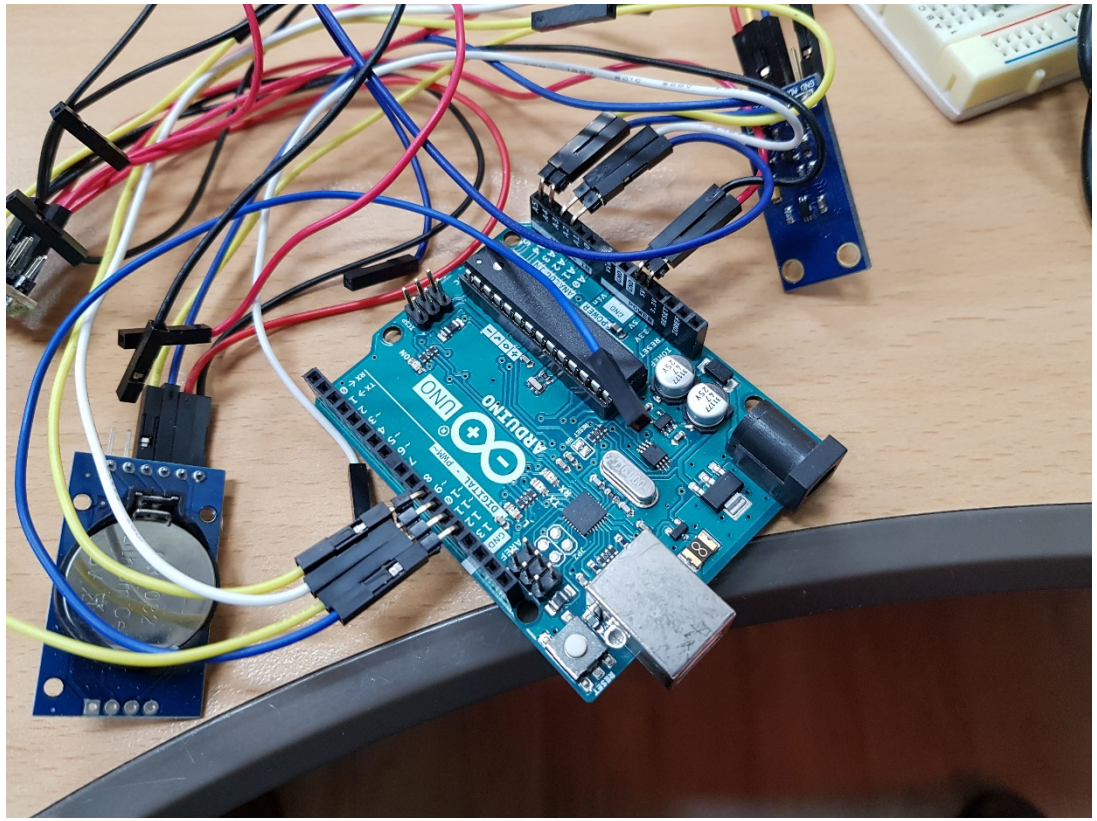
CodeVision 또는 Atmel Studio 7.0 을 통해 프로그래밍이 가능하며 프로그램전 디버거의 설정이 필수적이다.



Atmel STK500

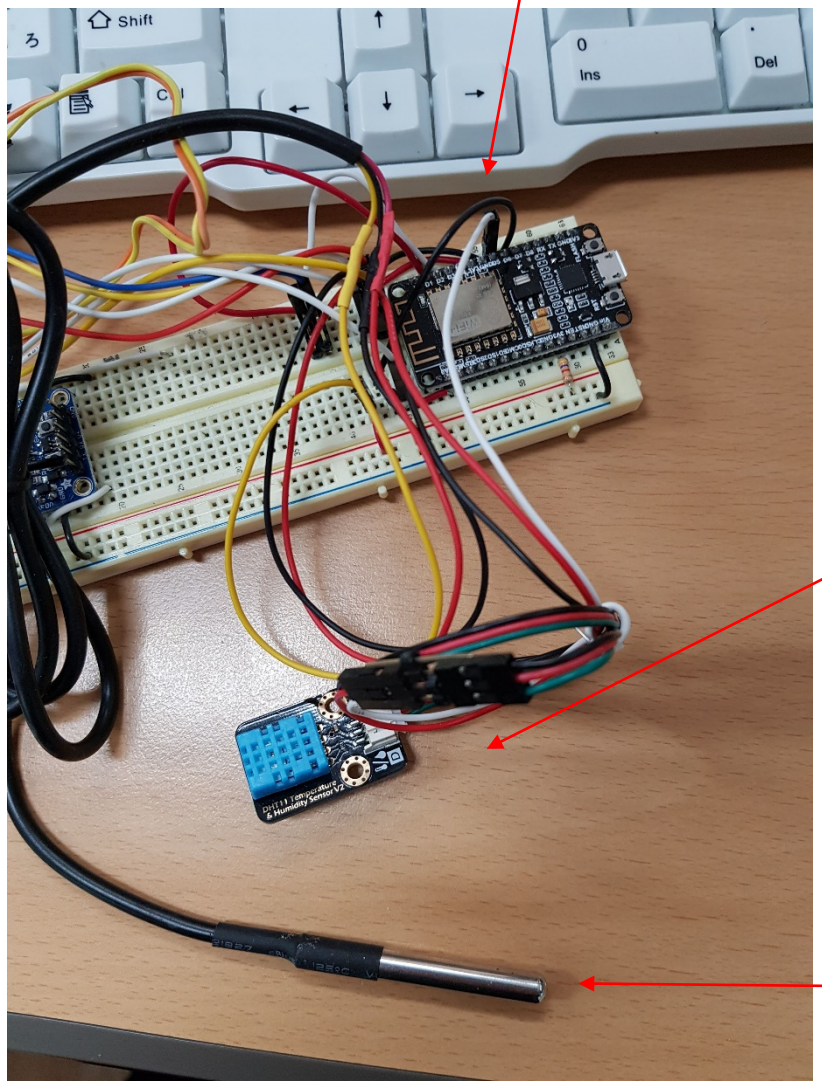


➢ FPGA_AVR



Atmega328P

Tensilica Xtensa



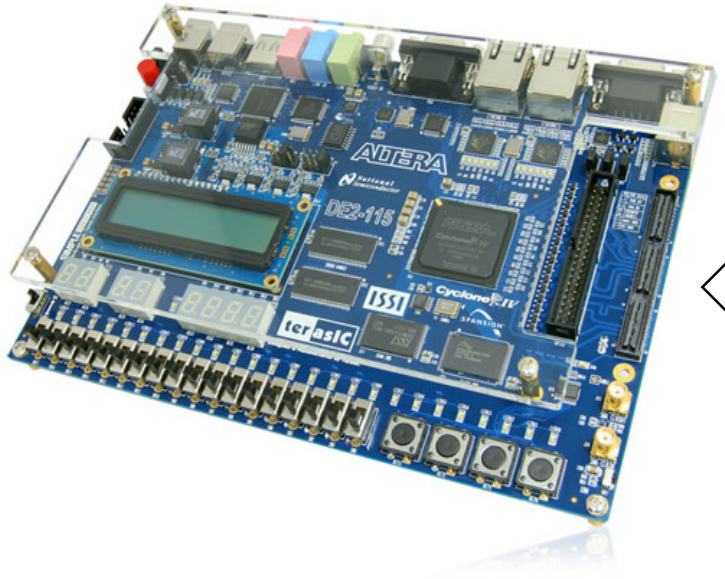
DHT 11
온습도 센서

DS
온도센서

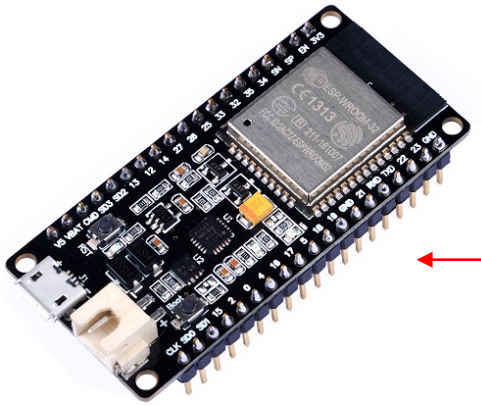
➢ FPGA_AVR

마이크로 프로세서

FPGA



UART

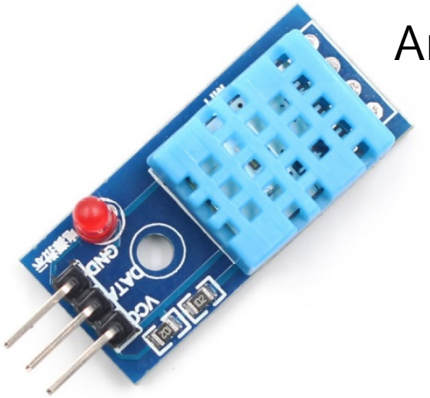


Digital

Digital

센서

Analog



➢ FPGA_AVR

SHT 30 대체 가능

전송 속도 설정

DHT11에서 온/습도 정보를
읽어오는 코드

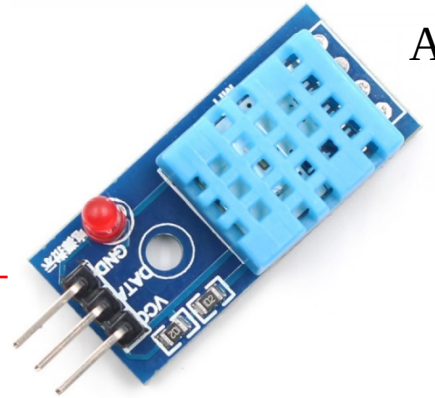
DE2로 데이터를 전송하는 부분
UART(RS232)

센서

Analog

Digital

```
4  #include "DHT.h"
5  #define DHTPIN 2
6  #define DHTTYPE DHT11
7  DHT dht(DHTPIN, DHTTYPE);
8
9  void setup() {
10   Serial.begin(9600);
11 }
12 void loop() {
13   delay(2000);
14   int h = dht.readHumidity();
15   int t = dht.readTemperature();
16   Serial.print("Humidity: ");
17   Serial.print(h);
18   Serial.print(" %₩t");
19   Serial.print("Temperature: ");
20   Serial.print(t);
21   Serial.println(" C");
22 }
```



- FPGA_AVR

Atmega128 설정

프로그래머(다운로더) 종류 설정하기

GCC를 통해 컴파일 하거나 Python을 통해 컴파일 한 후 프로그래머로 전송하기 때문에 Arduino IDE에서도 CodeVision에서 설정한 것과 동일한 디버거를 설정해야 함.

➢ FPGA_AVR CodeVision 또는 Atmel Studio(AVR Studio) 사용 시

1

```
//*****************************************************************************
// File Name : cmdlinetest.c
//
// Title        : example usage of cmdline (command line) functions
// Revision     : 1.0
// Notes        :
// Target MCU   : Atmel AVR series
// Editor Tabs  : 4
//
// Revision History:
// When                 Who          Description of change
// -----------  -----------  -----------------------
// 21-Jul-2003  pstang       Created the program
//*****************************************************************************


//----- Include Files ---------------------------------------------------------
#include <stdlib.h>
#include <avr/io.h>             // include I/O definitions (port names, pin names, etc)
//#include <avr/signal.h>       // include "signal" names (interrupt names)
#include <avr/interrupt.h>  // include interrupt support
#include <math.h>

#include "global.h"         // include our global settings
#include "uart2.h"          // include uart function library
```

2

```
#include "rprintf.h"    // include printf function library
#include "timer128.h"       // include timer function library (timing, PWM, etc)
#include "vt100.h"          // include vt100 terminal support
#include "a2d.h"
#include "adc_sht11_hanback.h"


// global variables
u08 Run;
u08 disp_mode;
s16 Temp;
u16 Humi;
u16 Photo;
u16 Dist;
double Dist2;
double Dist3;
double Dist4;
double Dist5;
double Dist_fit;
char TempBuf[4];
char HumiBuf[4];
char PhotoBuf[4];
char DistBuf[4];
u08 segBuf[4];
s08 Bin2FND[16]={0x0,0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9};
//setting of external SRAM interface emulated PLD
#define MEMORY_BASE_ADDRESS 0x1100
volatile unsigned char * const pldLCDline1Address = (volatile unsigned char *
const)(MEMORY_BASE_ADDRESS);
//address of rfid send data
volatile unsigned char * const pldLCDline2Address = (volatile unsigned char *
const)(MEMORY_BASE_ADDRESS+0x10);

//----- Begin Code ------------------------------------------------------------
int main(void)
{
    //external ram interface enable
  sbi(MCUCR,SRE);
  outb(XMCRB,0x07); //upper address pin set to PORTC
  DDRB=0xc0;
  DDRE=0x10;
  //external interrupt
  EIMSK = 0x40;
  EICRB = 0x20;
  EIFR = 0x40;
  // initialize the UART (serial port)
  uartInit();
  uartSetBaudRate(0,57600);
  // make all rprintf statements use uart for output
  rprintfInit(uart0SendByte);
  //! initialize a2d converter
    a2dInit();
  // initialize vt100 terminal
  //vt100Init();
  //init timer
  timerInit();
  // sht11 init
  _initialize_sht11_hanback();
  start_SHT11_sensor();
  //init cds light sensor power on
  PORTE |= 0x10;
  Run=1;
  disp_mode=0;
```

➢ FPGA_AVR CodeVision 또는 Atmel Studio(AVR Studio) 사용 시

3

```
sei();
timerPause(100);
*(pldLCDline1Address)='H';
*(pldLCDline1Address+0x01)='B';
*(pldLCDline1Address+0x02)='E';
*(pldLCDline1Address+0x03)='-';
*(pldLCDline1Address+0x04)='C';
*(pldLCDline1Address+0x05)='o';
*(pldLCDline1Address+0x06)='m';
*(pldLCDline1Address+0x07)='b';
*(pldLCDline1Address+0x08)='o';
*(pldLCDline1Address+0x09)=0xa0;
*(pldLCDline1Address+0x0a)=' ';
*(pldLCDline1Address+0x0b)='A';
*(pldLCDline1Address+0x0c)='V';
*(pldLCDline1Address+0x0d)='R';
*(pldLCDline1Address+0x0e)=' ';
*(pldLCDline1Address+0x0f)=' ';
while(Run)
{
    EIMSK |= 0x40;
    //온도값을 가져온다.
    Temp=get_sht11_hanback_data(TEMP);
    itoa(Temp,&TempBuf[1],10);
    if(Temp<0)TempBuf[0]='-';
    //시리얼로 데이터를 뿌려준다.
    if(Temp<0)TempBuf[0]='-';
    else TempBuf[0]='+';
    if(Temp==0)TempBuf[0]=' ';
    rprintfStr("sht11 Temperature is ");
    rprintfChar(TempBuf[0]);
    rprintfChar(TempBuf[1]);
    rprintfChar(TempBuf[2]);
    rprintfChar(TempBuf[3]);
    rprintfStr("\r\n");
    PORTB |= (0xc0);
    //timerPause(20);
    //습도값 가져온다.
    Humi=get_sht11_hanback_data(HUMI);
    if(Humi>999)    itoa(Humi,&HumiBuf[1],10);
    else itoa(Humi,&HumiBuf[1],10);
    //시리얼로 뿌려준다.
    rprintfStr("sht11 Humidity is ");
      if(Humi>999)rprintfChar('1');
    else rprintfChar(' ');
    rprintfChar(HumiBuf[1]);
    rprintfChar(HumiBuf[2]);
    rprintfChar(HumiBuf[3]);
    rprintfStr("\r\n");
    PORTB &= ~(0x40);
    //timerPause(20);
    //조도값 가져온다.
    Photo=a2dConvert10bit(ADC_CH_ADC0);
    if(Photo<99){
       itoa(Photo,(char*)&PhotoBuf[2],10);
       PhotoBuf[1]=' ';
       PhotoBuf[0]=' ';
      }
    else if(Photo<999){
       itoa(Photo,(char*)&PhotoBuf[1],10);
       PhotoBuf[0]=' ';
      }
```

온도 읽기

시리얼 전송

습도 읽기

시리얼 전송

조도 읽기

4

```
else{
   itoa(Photo,(char*)&PhotoBuf[0],10);
  }
//시리얼로 뿌려준다.
rprintfStr("Photo value is ");
rprintfChar(PhotoBuf[0]);
rprintfChar(PhotoBuf[1]);
rprintfChar(PhotoBuf[2]);
rprintfChar(PhotoBuf[3]);
rprintfStr("\r\n");
PORTB &= ~(0x80);
PORTB |= (0x40);
//timerPause(100);
//거리값 가져온다.
Dist=a2dConvert10bit(ADC_CH_ADC1);
Dist2=(double)(Dist*Dist);
Dist3=(double)(Dist2*Dist);
Dist4=(double)(Dist3*Dist);
Dist5=(double)(Dist4*Dist);

//Equation: y = A1*exp(-x/t1) + y0
//Parameter     Value       Error
//------------------------------------------
//y0  109.91634      6.48522
//A1  1205.19866     30.45446
//t1  15.30898 0.59908
//------------------------------------------

  //Dist_fit=100.89465+(1205.08623*exp(-0.06531*(double)Dist));
Dist_fit=exp((double)Dist/(double)-15.30898);
Dist=(u16)Dist_fit;
itoa(Dist,(char*)DistBuf,10);
if(Dist<99){
   itoa(Dist,(char*)&DistBuf[2],10);
   DistBuf[1]=' ';
   DistBuf[0]=' ';
  }
else if(Dist<999){
   itoa(Dist,(char*)&DistBuf[1],10);
   DistBuf[0]=' ';
  }
else{
   itoa(Dist,(char*)&DistBuf[0],10);
  }
//시리얼로 뿌려준다.
rprintfStr("Distance value is ");
rprintfChar(DistBuf[0]);
rprintfChar(DistBuf[1]);
rprintfChar(DistBuf[2]);
rprintfChar(DistBuf[3]);
rprintfStr("\r\n");
PORTB &= ~(0xc0);
//disp_mode값에 따라 PLD에 저장한다.
switch(disp_mode)
{
   case 0:
      *(pldLCDline2Address)='T';
      *(pldLCDline2Address+0x01)='e';
      *(pldLCDline2Address+0x02)='m';
      *(pldLCDline2Address+0x03)='p';
      *(pldLCDline2Address+0x04)=' ';
      *(pldLCDline2Address+0x05)=':';
```

시리얼 전송

거리 읽기

거리 연산

시리얼 전송

FPGA로 VHDL을 사용해 UART 통신을 하는 기본 코드는 다음과 같다.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity rs_232 is
         port (
                  reset : in std_logic;
                  clk : in std_logic;  -- 25MHz input clock

                  parity_en : in std_logic;
                  parity_mode : in std_logic;

                  rxd : in std_logic;
                  txd : out std_logic;

                  rxd_en : out std_logic;
                  txd_en : in std_logic;
                  uart_data_in : out std_logic_vector(7 downto 0);
                  uart_data_out : in std_logic_vector(7 downto 0)
         );
end rs_232;
```

```
architecture a of rs_232 is

signal resetn : std_logic := '0';

constant bit_rate : std_logic_vector(7 downto 0) := "11011001";  -- 25M / 115200 = 217
constant bit_rate_half : std_logic_vector(7 downto 0) := "01101100";  -- 108

type rx_state is (rx_idle, rx_start, data_rx, rx_parity, rx_stop);
signal rx_routine : rx_state;

type tx_state is (tx_idle, tx_start, data_tx, tx_parity, tx_stop);
signal tx_routine : tx_state;

signal rx_bit_add : std_logic_vector(7 downto 0);

signal cnt_rx_bit_rate : std_logic_vector(7 downto 0);
signal cnt_rx_data : std_logic_vector(2 downto 0);
signal cnt_tx_bit_rate : std_logic_vector(7 downto 0);
signal cnt_tx_data : std_logic_vector(2 downto 0);

signal rx_data : std_logic_vector(7 downto 0);
signal tx_data : std_logic_vector(7 downto 0);

signal reg_rxd : std_logic_vector(15 downto 0);

signal tx_parity_chk : std_logic;
```

```
begin
resetn <= not reset;
process (resetn, clk)
begin
        if resetn = '0' then
                reg_rxd <= (others => '0');
        elsif clk'event and clk = '1' then
                reg_rxd <= rxd & reg_rxd(15 downto 1);
        end if;
end process;
process (resetn, clk)
begin
        if resetn = '0' then
                rx_bit_add <= (others => '0');
        elsif clk'event and clk = '1' then
                if rx_routine = data_rx then
                        if cnt_rx_bit_rate = bit_rate then
                                rx_bit_add <= (others => '0');
                        else
                                rx_bit_add <= rx_bit_add + reg_rxd(0);
                        end if;
                else
                        rx_bit_add <= (others => '0');
                end if;
        end if;
end process;
```

```
process (resetn, clk)
begin
        if resetn = '0' then
                rx_routine <= rx_idle;
        elsif clk'event and clk = '1' then
                case rx_routine is
                        when rx_idle =>
                                if reg_rxd = "0000000000000000" then
                                        rx_routine <= rx_start;
                                end if;
                        when rx_start =>
                                if cnt_rx_bit_rate = bit_rate then
                                        rx_routine <= data_rx;
                                elsif cnt_rx_bit_rate = bit_rate_half then
                                        if reg_rxd(0) = '1' then
                                                rx_routine <= rx_idle;
                                        end if;
                                end if;
                        when data_rx =>
                                if cnt_rx_data = "111" then
                                        if cnt_rx_bit_rate = bit_rate then
                                                if parity_en = '1' then
                                                        rx_routine <= rx_parity;
                                                else
                                                        rx_routine <= rx_stop;
                                                end if;
                                        end if;
                                end if;
                        when rx_parity =>
                                if cnt_rx_bit_rate = bit_rate then
                                        rx_routine <= rx_stop;
                                end if;
                        when rx_stop =>
                                if cnt_rx_bit_rate(5 downto 0) = bit_rate(7 downto 2) then
                                        rx_routine <= rx_idle;
                                end if;
                end case;
        end if;
end process;
```

```
process (resetn, clk)
begin
		if resetn = '0' then
			cnt_rx_bit_rate <= (others => '0');
		elsif clk'event and clk = '1' then
			if rx_routine = rx_idle then
				cnt_rx_bit_rate <= (others => '0');
			elsif cnt_rx_bit_rate = bit_rate then
				cnt_rx_bit_rate <= (others => '0');
			else
				cnt_rx_bit_rate <= cnt_rx_bit_rate + '1';
			end if;
		end if;
end process;

process (resetn, clk)
begin
		if resetn = '0' then
			cnt_rx_data <= (others => '0');
		elsif clk'event and clk = '1' then
			if rx_routine = data_rx then
				if cnt_rx_bit_rate = bit_rate then
					if cnt_rx_data = "111" then
						cnt_rx_data <= (others => '0');
					else
						cnt_rx_data <= cnt_rx_data + '1';
					end if;
				end if;
			else
				cnt_rx_data <= (others => '0');
			end if;
		end if;
end process;
```

```
process (resetn, clk)
begin
		if resetn = '0' then
			rx_data <= (others => '0');
		elsif clk'event and clk = '1' then
			if rx_routine = data_rx then
				if cnt_rx_bit_rate = bit_rate - 1 then
					if rx_bit_add >= bit_rate_half then
						rx_data <= '1' & rx_data(7 downto 1);
					else
						rx_data <= '0' & rx_data(7 downto 1);
					end if;
				end if;
			elsif rx_routine = rx_idle then
				rx_data <= (others => '0');
			end if;
		end if;
end process;

process (resetn, clk)
begin
		if resetn = '0' then
			rxd_en <= '0';
		elsif clk'event and clk = '1' then
			if rx_routine = rx_stop then
				if cnt_rx_bit_rate >=  10 then
					rxd_en <= '1';
				else
					rxd_en <= '0';
				end if;
			else
				rxd_en <= '0';
			end if;
		end if;
end process;
```

```
process (resetn, clk)
begin
		if resetn = '0' then
			uart_data_in <= (others => '0');
		elsif clk'event and clk = '1' then
			if rx_routine = rx_stop then
				uart_data_in <= rx_data;
			end if;
		end if;
end process;
```

```
process (resetn, clk)
begin
            if resetn = '0' then
                        tx_routine <= tx_idle;
            elsif clk'event and clk = '1' then
                        case tx_routine is
                                    when tx_idle =>
                                                if txd_en = '1' then
                                                            tx_routine <= tx_start;
                                                end if;
                                    when tx_start =>
                                                if cnt_tx_bit_rate = bit_rate then
                                                            tx_routine <= data_tx;
                                                end if;
                                    when data_tx =>
                                                if cnt_tx_data = "111" then
                                                            if cnt_tx_bit_rate = bit_rate then
                                                                        if parity_en = '1' then
                                                                                    tx_routine <= tx_parity;
                                                                        else
                                                                                    tx_routine <= tx_stop;
                                                                        end if;
                                                            end if;
                                                end if;
                                    when tx_parity =>
                                                if cnt_tx_bit_rate = bit_rate then
                                                            tx_routine <= tx_stop;
                                                end if;
                                    when tx_stop =>
                                                if cnt_tx_bit_rate = bit_rate - 10 then
                                                            tx_routine <= tx_idle;
                                                end if;
                                    when others =>
                                                tx_routine <= tx_idle;
                        end case;
            end if;
end process;
```

```
process (resetn, clk)
begin
        if resetn = '0' then
                cnt_tx_bit_rate <= (others => '0');
        elsif clk'event and clk = '1' then
                if tx_routine = tx_idle then
                        cnt_tx_bit_rate <= (others =>
'0');
                elsif cnt_tx_bit_rate = bit_rate then
                        cnt_tx_bit_rate <= (others =>
'0');
                else
                        cnt_tx_bit_rate <=
cnt_tx_bit_rate + '1';
                end if;
        end if;
end process;
```

```
process (resetn, clk)
begin
        if resetn = '0' then
                cnt_tx_data <= (others => '0');
        elsif clk'event and clk = '1' then
                if tx_routine = data_tx then
                        if cnt_tx_bit_rate = bit_rate then
                                if cnt_tx_data = "111" then
                                        cnt_tx_data <= (others => '0');
                                else
                                        cnt_tx_data <= cnt_tx_data + '1';
                                end if;
                        end if;
                else
                        cnt_tx_data <= (others => '0');
                end if;
        end if;
end process;
```

```
process (resetn, clk)
begin
        if resetn = '0' then
                tx_data <= (others => '0');
        elsif clk'event and clk = '1' then
                if tx_routine = tx_idle then
                        if txd_en = '1' then
                                tx_data <= uart_data_out;
                        end if;
                end if;
        end if;
end process;

process (resetn, clk)
begin
        if resetn = '0' then
                tx_parity_chk <= '0';
        elsif clk'event and clk = '1' then
                if tx_routine = tx_idle then
                        if txd_en = '1' then
                                tx_parity_chk <= uart_data_out(7) xor uart_data_out(6) xor
                                                 uart_data_out(5) xor uart_data_out(4) xor
                                                 uart_data_out(3) xor uart_data_out(2) xor
                                                 uart_data_out(1) xor uart_data_out(0) xor
                                                 parity_mode;
                        end if;
                end if;
        end if;
end process;
```

FPGA UART(RS232)

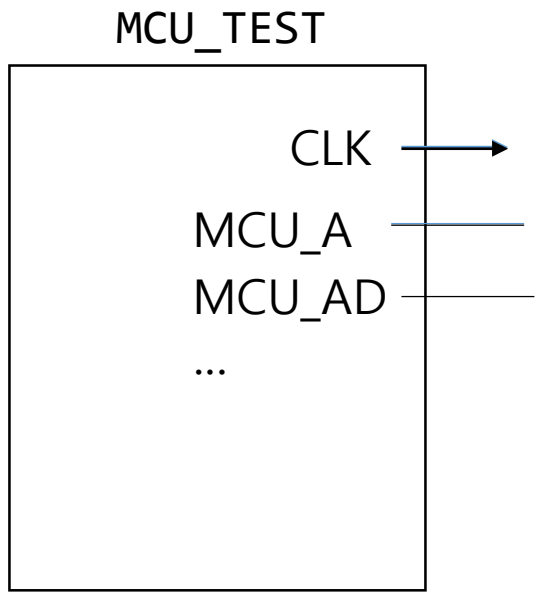
```
process (resetn, clk)
begin
		if resetn = '0' then
			txd <= '1';
		elsif clk'event and clk = '1' then
			case tx_routine is
				when tx_idle =>
					txd <= '1';
				when tx_start =>
					txd <= '0';
				when data_tx =>
					if cnt_tx_data = "000" then
						txd <= tx_data(0);
					elsif cnt_tx_data = "001" then
						txd <= tx_data(1);
					elsif cnt_tx_data = "010" then
						txd <= tx_data(2);
					elsif cnt_tx_data = "011" then
						txd <= tx_data(3);
					elsif cnt_tx_data = "100" then
						txd <= tx_data(4);
					elsif cnt_tx_data = "101" then
						txd <= tx_data(5);
					elsif cnt_tx_data = "110" then
						txd <= tx_data(6);
					elsif cnt_tx_data = "111" then
						txd <= tx_data(7);
					else
						txd <= '1';
					end if;
				when tx_parity =>
					txd <= tx_parity_chk;
				when tx_stop =>
					txd <= '1';
				when others =>
					txd <= '1';
			end case;
		end if;
end process;

end a;
```

위의 기준 코드를 활용하여 AVR에서 수신 받은 값을 표기할 수 있도록 해본다.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY MCU_TEST IS
PORT (
SW_RESET : IN STD_LOGIC;
CLK : IN STD_LOGIC;
MCU_A : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
MCU_AD : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0);
MCU_ALE : IN STD_LOGIC;
MCU_WRN : IN STD_LOGIC;
MCU_RDN : IN STD_LOGIC;
MCU_TXD : IN STD_LOGIC;
MCU_RXD : OUT STD_LOGIC;
MCU_INT2 : OUT STD_LOGIC;
MCU_INT3 : OUT STD_LOGIC;
MCU_GPIO : INOUT STD_LOGIC_VECTOR(10 DOWNTO 0);
SW_PUSH : IN STD_LOGIC;
VFD_E : OUT STD_LOGIC;
VFD_RS : OUT STD_LOGIC;
VFD_RW : OUT STD_LOGIC;
VFD_DATA : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
);
END MCU_TEST;
```

라이브러리

블록 핀 설정



```
ARCHITECTURE A OF MCU_TEST IS

SIGNAL RESETN : STD_LOGIC;
SIGNAL MCU_ADDR : STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL MCU_READ_DATA : STD_LOGIC_VECTOR(7 DOWNTO 0);
TYPE AR_MEM IS ARRAY (0 TO 31) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL MCU_MEM : AR_MEM;

TYPE VFD_STATE IS (
DELAY_50M,
FUNCTION_SET,
ENTRY_MODE,
DISP_ON,
DISP_LINE1,
DISP_LINE2,
BRIGHT_SET,
DELAY_100M
);
SIGNAL VFD_ROUTINE : VFD_STATE;
TYPE AR_VFD_DATA IS ARRAY (0 TO 16) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL REG_VFD_DATA1 : AR_VFD_DATA;
SIGNAL REG_VFD_DATA2 : AR_VFD_DATA;
SIGNAL CNT_CLK : INTEGER RANGE 0 TO 4999;
SIGNAL CNT_DELAY_50M : INTEGER RANGE 0 TO 49;
SIGNAL CNT_DELAY_100M : INTEGER RANGE 0 TO 99;
SIGNAL CNT_LINE : INTEGER RANGE 0 TO 16;
```

ARCHITECTURE 안에서 사용할 신호 및 변수 설정

➢ FPGA_AVR FPGA UART(RS232)

```
BEGIN
RESETN <= NOT SW_RESET;
MCU_INT2 <= NOT SW_PUSH;
MCU_INT3 <= '1';
MCU_GPIO <= (OTHERS => 'Z');
MCU_RXD <= '0';
PROCESS (RESETN, MCU_ALE)
BEGIN
IF RESETN = '0' THEN
MCU_ADDR <= (OTHERS => '0');
ELSIF MCU_ALE'EVENT AND MCU_ALE = '0' THEN
MCU_ADDR <= MCU_A & MCU_AD;
END IF;
END PROCESS;
PROCESS(RESETN, MCU_WRN)
BEGIN
IF RESETN = '0' THEN
MCU_MEM(0) <= "01001000";
MCU_MEM(1) <= "01000010";
MCU_MEM(2) <= "01000101";
MCU_MEM(3) <= "00101101";
MCU_MEM(4) <= "01000011";
MCU_MEM(5) <= "01101111";
MCU_MEM(6) <= "01101101";
MCU_MEM(7) <= "01100010";
MCU_MEM(8) <= "01101111";
MCU_MEM(9) <= "10100000";
MCU_MEM(10) <= "00100000";
MCU_MEM(11) <= "01000001";
MCU_MEM(12) <= "01010110";
MCU_MEM(13) <= "01010010";
MCU_MEM(14) <= "00100000";
MCU_MEM(15) <= "00100000";
MCU_MEM(16) <= "10011101";
MCU_MEM(17) <= "10011101";
MCU_MEM(18) <= "10011101";
MCU_MEM(19) <= "10011101";
MCU_MEM(20) <= "10011101";
MCU_MEM(21) <= "10011101";
MCU_MEM(22) <= "10011101";
MCU_MEM(23) <= "10011101";
MCU_MEM(24) <= "10011101";
MCU_MEM(25) <= "10011101";
MCU_MEM(26) <= "10011101";
MCU_MEM(27) <= "10011101";
MCU_MEM(28) <= "10011101";
MCU_MEM(29) <= "10011101";
MCU_MEM(30) <= "10011101";
MCU_MEM(31) <= "10011101";
ELSIF MCU_WRN'EVENT AND MCU_WRN = '1' THEN
MCU_MEM(CONV_INTEGER(UNSIGNED(MCU_ADDR))) <= MCU_AD;
END IF;
END PROCESS;
```

스위치를 누르면 프로세서에 External Interrupt를 발생시킨다.

➢ FPGA_AVR FPGA UART(RS232)

```
PROCESS (RESETN, MCU_RDN)
BEGIN
IF RESETN = '0' THEN
MCU_READ_DATA <= (OTHERS => '0');
ELSIF MCU_RDN'EVENT AND MCU_RDN = '0' THEN
MCU_READ_DATA <= MCU_MEM(CONV_INTEGER(UNSIGNED
(MCU_ADDR)));
END IF;
END PROCESS;
MCU_AD <= MCU_READ_DATA WHEN MCU_RDN = '0' ELSE (OTHERS => 'Z');
REG_VFD_DATA1(0) <= "10000000"; -- LINE1 ADDRESS
REG_VFD_DATA1(1) <= MCU_MEM(0);
REG_VFD_DATA1(2) <= MCU_MEM(1);
REG_VFD_DATA1(3) <= MCU_MEM(2);
REG_VFD_DATA1(4) <= MCU_MEM(3);
REG_VFD_DATA1(5) <= MCU_MEM(4);
REG_VFD_DATA1(6) <= MCU_MEM(5);
REG_VFD_DATA1(7) <= MCU_MEM(6);
REG_VFD_DATA1(8) <= MCU_MEM(7);
REG_VFD_DATA1(9) <= MCU_MEM(8);
REG_VFD_DATA1(10) <= MCU_MEM(9);
REG_VFD_DATA1(11) <= MCU_MEM(10);
REG_VFD_DATA1(12) <= MCU_MEM(11);
REG_VFD_DATA1(13) <= MCU_MEM(12);
REG_VFD_DATA1(14) <= MCU_MEM(13);
REG_VFD_DATA1(15) <= MCU_MEM(14);
REG_VFD_DATA1(16) <= MCU_MEM(15);
REG_VFD_DATA2(0) <= "11000000"; -- LINE2 ADDRESS
REG_VFD_DATA2(1) <= MCU_MEM(16);
REG_VFD_DATA2(2) <= MCU_MEM(17);
REG_VFD_DATA2(3) <= MCU_MEM(18);
REG_VFD_DATA2(4) <= MCU_MEM(19);
REG_VFD_DATA2(5) <= MCU_MEM(20);
REG_VFD_DATA2(6) <= MCU_MEM(21);
REG_VFD_DATA2(7) <= MCU_MEM(22);
REG_VFD_DATA2(8) <= MCU_MEM(23);
REG_VFD_DATA2(9) <= MCU_MEM(24);
REG_VFD_DATA2(10) <= MCU_MEM(25);
REG_VFD_DATA2(11) <= MCU_MEM(26);
REG_VFD_DATA2(12) <= MCU_MEM(27);
REG_VFD_DATA2(13) <= MCU_MEM(28);
REG_VFD_DATA2(14) <= MCU_MEM(29);
REG_VFD_DATA2(15) <= MCU_MEM(30);
REG_VFD_DATA2(16) <= MCU_MEM(31);
VFD_RW <= '0';
```

- FPGA_AVR FPGA UART(RS232)

```
PROCESS (RESETN, CLK)
BEGIN
IF RESETN = '0' THEN
          CNT_CLK <= 0;
ELSIF CLK'EVENT AND CLK = '1' THEN
          IF CNT_CLK = 4999 THEN
          CNT_CLK <= 0;
          ELSE
          CNT_CLK <= CNT_CLK + 1;
          END IF;
END IF;
END PROCESS;

PROCESS (RESETN, CLK)
BEGIN
IF RESETN = '0' THEN
CNT_DELAY_50M <= 0;
ELSIF CLK'EVENT AND CLK = '1' THEN
IF VFD_ROUTINE = DELAY_50M THEN
IF CNT_CLK = 4999 THEN
IF CNT_DELAY_50M = 50 THEN
CNT_DELAY_50M <= 0;
ELSE
CNT_DELAY_50M <= CNT_DELAY_50M + 1;
END IF;
END IF;
ELSE
CNT_DELAY_50M <= 0;
END IF;
END IF;
END PROCESS;
```

```
PROCESS (RESETN, CLK)
BEGIN
IF RESETN = '0' THEN
CNT_DELAY_100M <= 0;
ELSIF CLK'EVENT AND CLK = '1' THEN
IF VFD_ROUTINE = DELAY_100M THEN
IF CNT_CLK = 4999 THEN
IF CNT_DELAY_100M = 99 THEN
CNT_DELAY_100M <= 0;
ELSE
CNT_DELAY_100M <= CNT_DELAY_100M + 1;
END IF;
END IF;
ELSE
CNT_DELAY_100M <= 0;
END IF;
END IF;
END PROCESS;
BEGIN
IF RESETN = '0' THEN
CNT_LINE <= 0;
ELSIF CLK'EVENT AND CLK = '1' THEN
IF VFD_ROUTINE = DISP_LINE1 OR VFD_ROUTINE = DISP_LINE2 THEN
IF CNT_CLK = 4999 THEN
IF CNT_LINE = 16 THEN
CNT_LINE <= 0;
ELSE
CNT_LINE <= CNT_LINE + 1;
END IF;
END IF;
ELSE
CNT_LINE <= 0;
END IF;
END IF;
END PROCESS;
```

- FPGA_AVR FPGA UART(RS232)

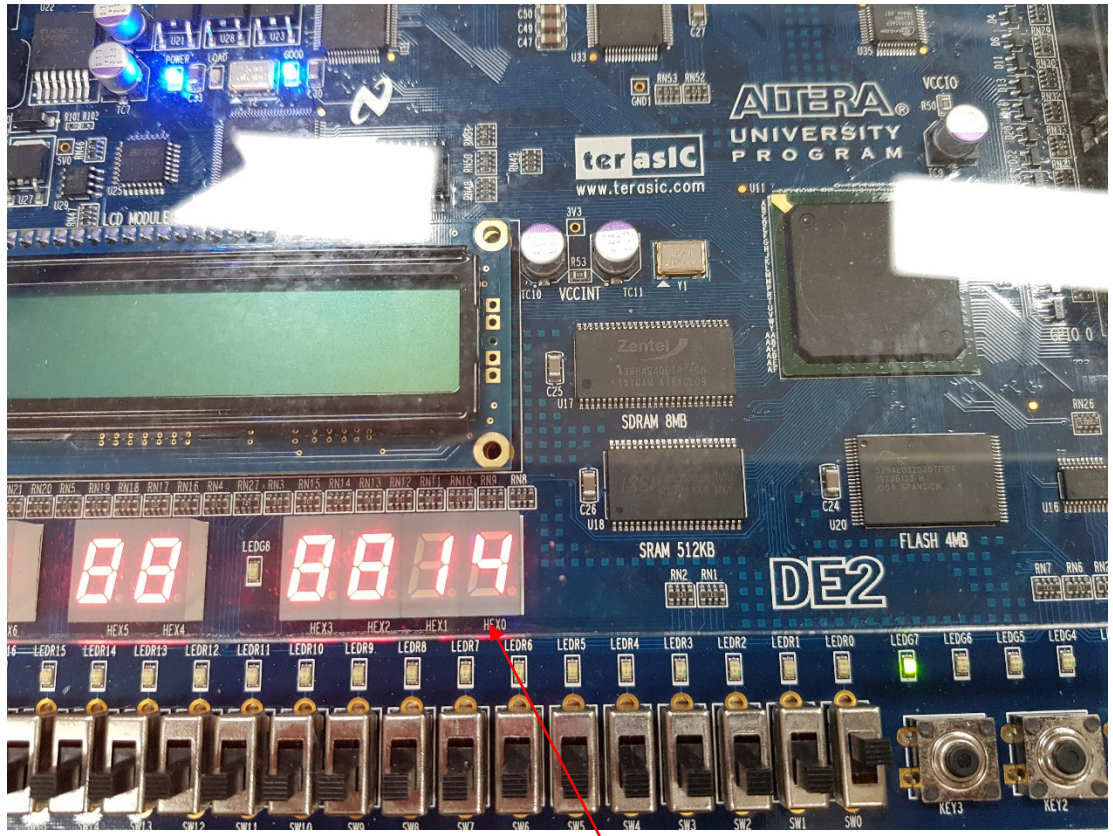
```
PROCESS (RESETN, CLK)
BEGIN
IF RESETN = '0' THEN
VFD_ROUTINE <= DELAY_50M;
ELSIF CLK'EVENT AND CLK = '1' THEN
IF CNT_CLK = 4999 THEN
CASE VFD_ROUTINE IS
WHEN DELAY_50M =>
IF CNT_DELAY_50M = 49 THEN
VFD_ROUTINE <= FUNCTION_SET;
END IF;
WHEN FUNCTION_SET =>
VFD_ROUTINE <= ENTRY_MODE;
WHEN ENTRY_MODE =>
VFD_ROUTINE <= DISP_ON;
WHEN DISP_ON =>
VFD_ROUTINE <= BRIGHT_SET;
WHEN BRIGHT_SET =>
VFD_ROUTINE <= DISP_LINE1;
WHEN DISP_LINE1 =>
IF CNT_LINE = 16 THEN
VFD_ROUTINE <= DISP_LINE2;
END IF;
WHEN DISP_LINE2 =>
IF CNT_LINE = 16 THEN
VFD_ROUTINE <= DELAY_100M;
END IF;
WHEN DELAY_100M =>
IF CNT_DELAY_100M = 99 THEN
VFD_ROUTINE <= DISP_LINE1;
END IF;
END CASE;
END IF;
END IF;
END PROCESS;
```

```
PROCESS (RESETN, CLK)
BEGIN
IF RESETN = '0' THEN
VFD_E <= '0';
ELSIF CLK'EVENT AND CLK = '1' THEN
CASE VFD_ROUTINE IS
WHEN DELAY_50M => VFD_E <= '0';
WHEN FUNCTION_SET TO BRIGHT_SET =>
IF CNT_CLK >= 200 AND CNT_CLK <= 2499 THEN
VFD_E <= '1';
ELSE
VFD_E <= '0';
END IF;
WHEN DELAY_100M =>
VFD_E <= '0';
END CASE;
END IF;
END PROCESS;
PROCESS (RESETN, CLK)
BEGIN
IF RESETN = '0' THEN
VFD_RS <= '0';
ELSIF CLK'EVENT AND CLK = '1' THEN
IF VFD_ROUTINE = DISP_LINE1 OR VFD_ROUTINE = DISP_LINE2 THEN
IF CNT_LINE = 0 THEN
VFD_RS <= '0';
ELSE
VFD_RS <= '1';
END IF;
ELSE
VFD_RS <= '0';
END IF;
END IF;
END PROCESS;
```

- FPGA_AVR FPGA UART(RS232)

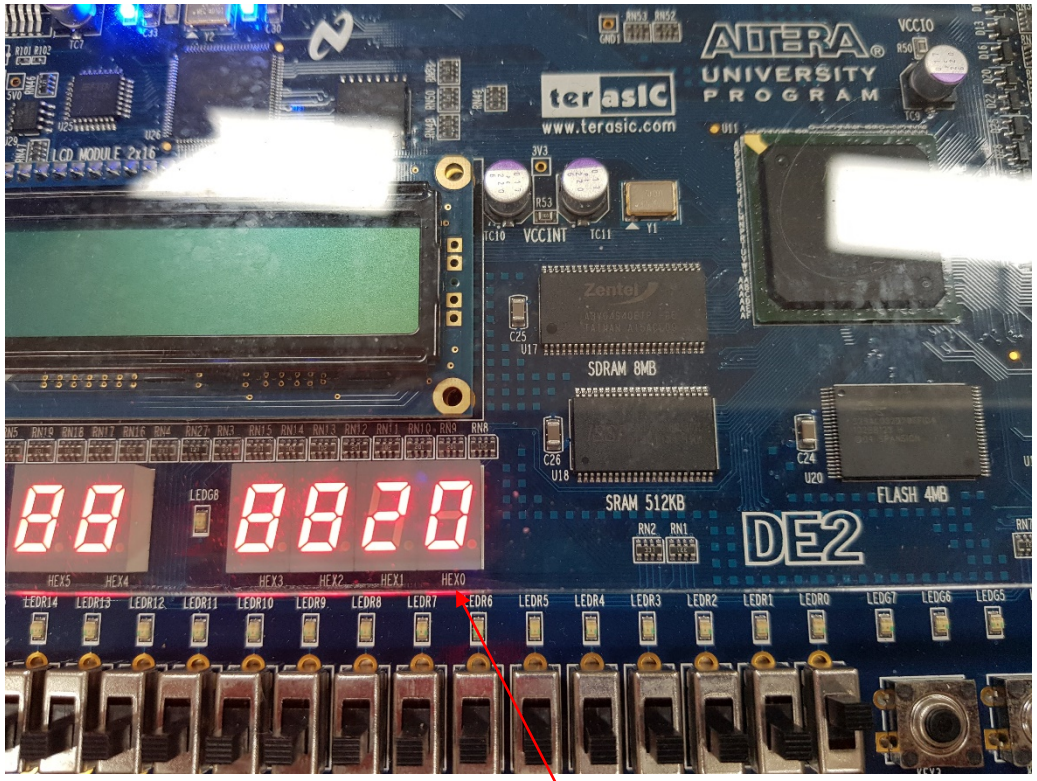
```
PROCESS (RESETN, CLK)
BEGIN
IF RESETN = '0' THEN
VFD_DATA <= "00000000";
ELSIF CLK'EVENT AND CLK = '1' THEN
CASE VFD_ROUTINE IS
WHEN DELAY_50M =>
VFD_DATA <= "00000000";
WHEN FUNCTION_SET =>
VFD_DATA <= "00111100";
WHEN ENTRY_MODE =>
VFD_DATA <= "00000110";
WHEN DISP_ON =>
VFD_DATA <= "00001100";
WHEN DISP_LINE1 =>
VFD_DATA <= REG_VFD_DATA1(CNT_LINE);
WHEN DISP_LINE2 =>
VFD_DATA <= REG_VFD_DATA2(CNT_LINE);
WHEN BRIGHT_SET =>
VFD_DATA <= "00111100";
WHEN DELAY_100M =>
VFD_DATA <= "00000000";
END CASE;
END IF;
END PROCESS;
END A;
```

- FPGA_AVR
  - Result



Laboratory Temperature
The Segment Show 14

- FPGA_AVR
  - Result



When I touched temperature Sensor
, the temperature was increased.
14 -> 20