ATtiny13A, 85전용개발보드 – ICSP포트에 USBtiny프로그래머 연결해 업로드 (ATtiny13A, 85 DevBoard - USBtiny programmer using ICSP ports)

PN: 23189814

차례

[1 목적 2](#_Toc522440395)

[2 코딩 시나리오 2](#_Toc522440396)

[3 결과 사진 2](#_Toc522440397)

[4 ATmel Studio 7.0 개발 환경 4](#_Toc522440398)

[5 참조 6](#_Toc522440399)

[6 코드 6](#_Toc522440400)

# 목적

ATtiny 13A 또는 85용 미니 개발보드(이후 “개발보드”) 에 ICSP포트 핀 헤더를 납땝하고, USBtiny프로그래머로 업로드 방법을 설명한다. Atmel Studio 7.0 IDE및 아두이노 IDE를 이용하였다.

# 코딩 시나리오

ATtiny 13A / 85전용 8핀 개발보드는 1달러가 채 되지 않는다. 하지만, 본 개발보드는 8핀 ATtiny MCU를 Atmel Studio IDE를 이용한 AVR C로 Power down sleep모드와 Watch dog timer를 구현하는 에너지 절감형 IoT응용 시나리오에 기반한 코딩을 실시한다. 이는 LH공사의 결로방지를 위한 모니터링 시스템에 응용될 예정이다.

# 결과 사진

## 6핀 헤더 장착한 개발보드

구매한 개발보드에 ICSP핀이 납땝되어 있지 않아서, 핀헤더를 납땜하였다. 그리고, 외부 소자 테스트를 위해 전원을 점퍼선으로 빼 두었다.

**ICSP 6 pin header**

MISO(PB1) Vcc

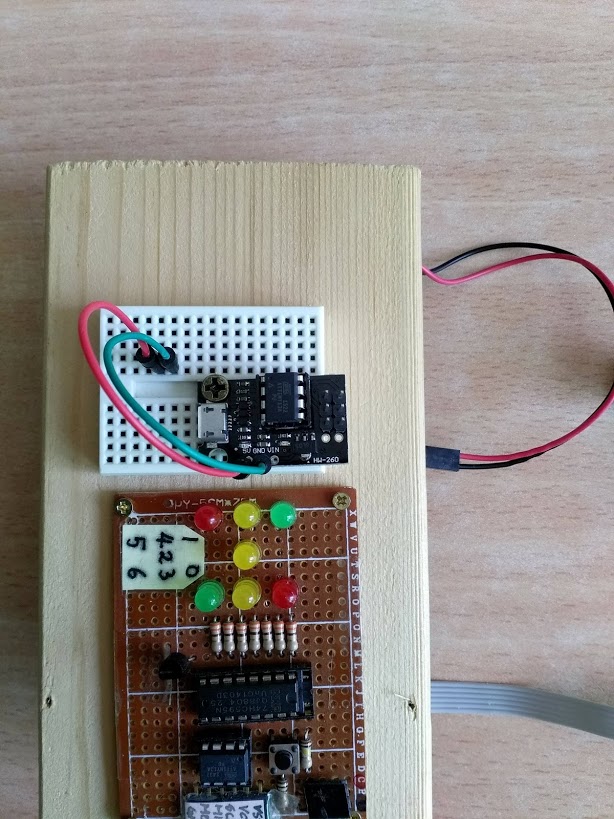
SCK(PB2) MOSI(PB0)

RST(PB5) GND

ATtiny13A

USB micro pwr inlet

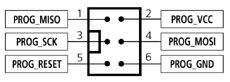
External pwr in/outlet



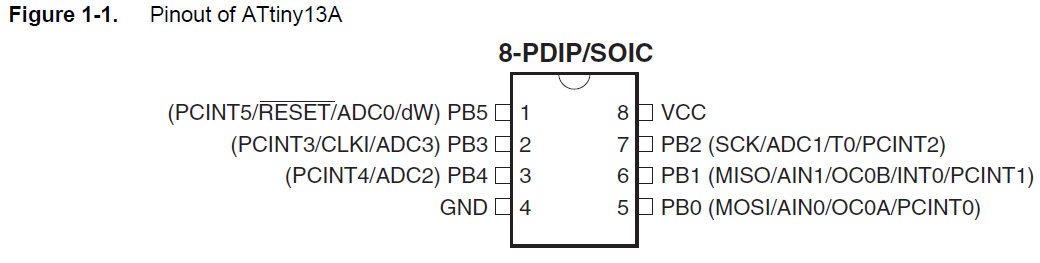
## USBtiny 프로그래머와 연결된 개발 보드

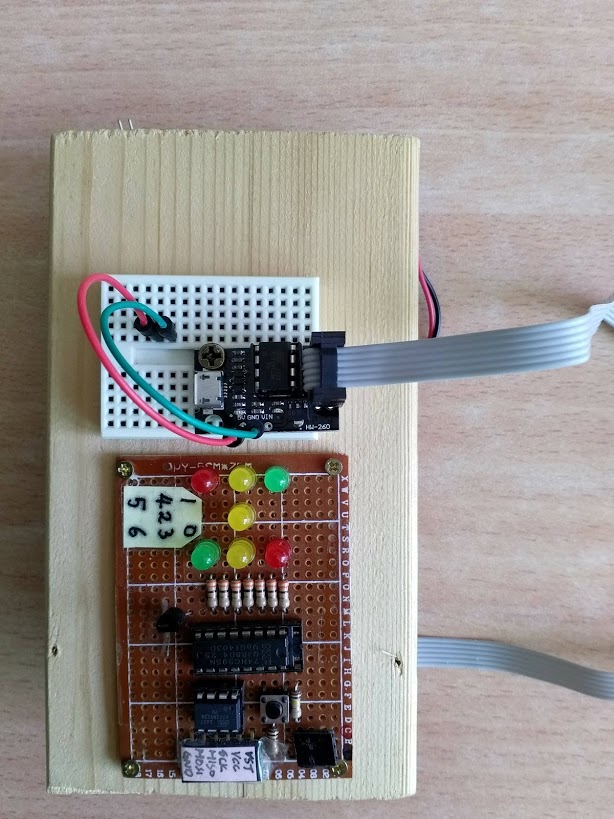
ATtiny13A 또는 ATtiny 85 MCU를 개발하기 위해 제작된 개발보드의 8핀 칩 소켓을 이용해서 ICSP 통신 선로를 연결하였다.

USBtiny 프로그래머의 6핀 ICSP소켓 핀아웃 정보



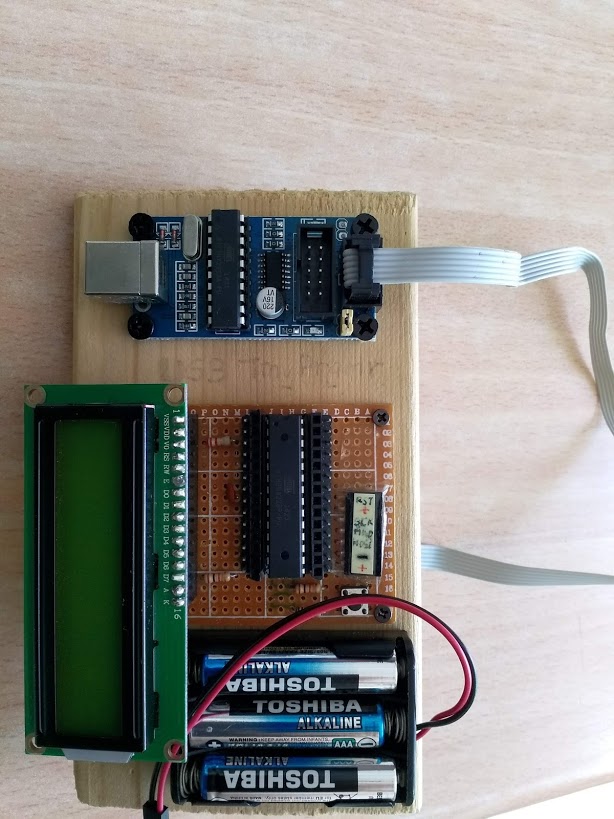
ATtiny13A 8핀 소켓 핀아웃 정보





## USBtiny 보드

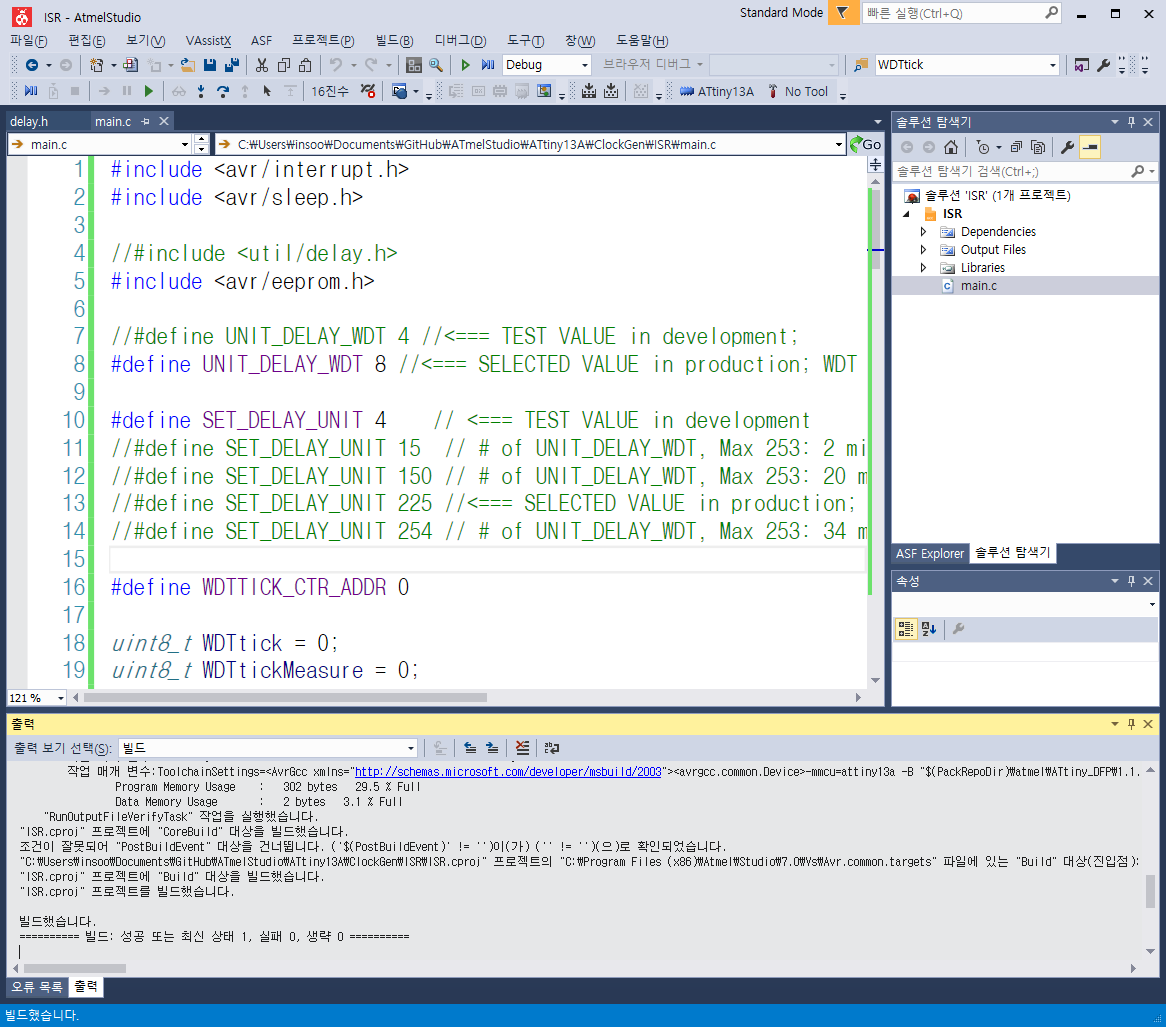
ATtiny2313 MCU를 이용해서 프로그래머로 구동된다. USB B타입(아두이노 우노용) 케이블로 호스트 컴퓨터(PC/Mac)와 연결한다.



# ATmel Studio 7.0 개발 환경

## 코딩

F7을 누르면 컴파일



## 업로딩

메뉴 – 도구 – USBtiny : Alt+T, y

위 단축키는 사전에 도구메뉴의 외부도구 항목에서 설정해 주었기 때문에 가능하다.

제목: USBtin&Y

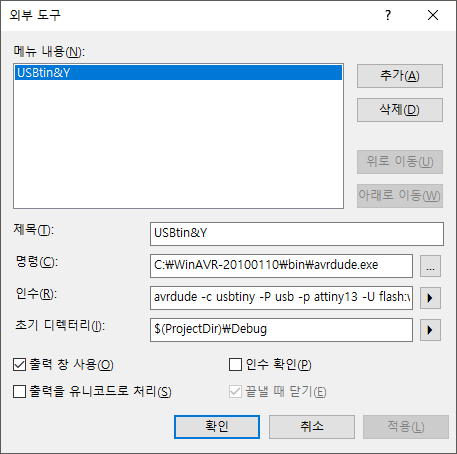
명령: C:\WinAVR-20100110\bin\avrdude.exe

인수: avrdude -c usbtiny -P usb -p attiny13 -U flash:w:ISR.hex:i

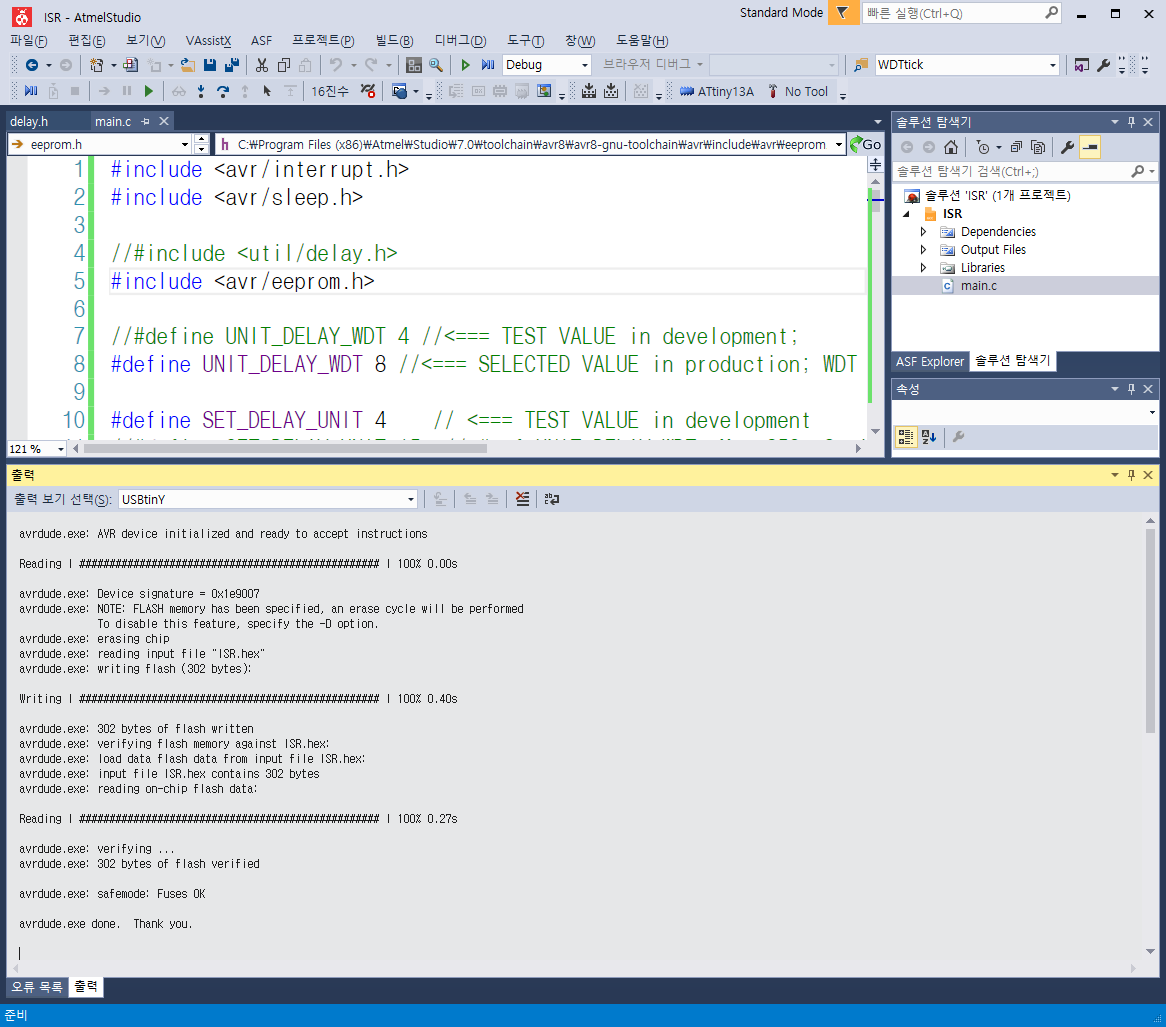
(참고) ATmega328p는 다음과 같이 인수를 지정한다.

avrdude -c usbtiny -P usb -p atmega328p -U flash:w:LCD.hex:i

초기디렉토리: $(ProjectDir)\Debug



F7으로 컴파일 후 문제가 없음을 확인 한 후, Alt+T, y를 눌러서 USBtiny프로그래머를 통해 ATmega328p로 업로드 하면, Atmel Studio의 출력창에 업로드 진행 경과가 표시된다. 업로드 중에는 USBtiny프로그래머의 빨간 LED가 켜지고, 업로드 완료되면 꺼진다.



# 참조

## DHT11 AVR C코딩

<http://www.electronicwings.com/avr-atmega/dht11-sensor-interfacing-with-atmega16-32>

# 코드

## Main.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<For Windows Atmel Studio>

cd " C:\Users\insoo\Documents\GitHub\ATmelStudio\ATtiny13A\ClockGen\ISR\Debug "

avrdude -c usbtiny -P usb -p attiny13 -U flash:w:LCD.hex:i

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Target MCU & clock speed: ATtiny13A @ 1.2Mhz internal

Name : main.c

C modules of this project, ISR:

main.c

Custom Headers:

Nothing

Author : Insoo Kim (insoo@hotmail.com)

Created : May 15, 2015

Updated : Aug 18, 2018 (On Atmel Studio 7)

Description:

ATtiny13A controls power up or down to ESP-01 module by 2n2222 NPN transistor.

ATtiny13A sleeps in most of operation time and wake up periodically to measure temperature and humidity by DHT22 attached to ESP-01.

HEX size[Byte]: 302 out of 1024

How to upload to the target MCU

<For Windows Atmel Studio>

Slect Tool – USBtiny (USBtiny memu should be configured in the external tool memu)

<For CMD window or DOS prompt>

cd " C:\Users\insoo\Documents\GitHub\ATmelStudio\ATtiny13A\ClockGen\ISR\Debug "

avrdude -c usbtiny -P usb -p attiny13 -U flash:w:ISR.hex:i

Ref:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <avr/interrupt.h>

#include <avr/sleep.h>

//#include <util/delay.h>

#include <avr/eeprom.h>

//#define UNIT\_DELAY\_WDT 4 //<=== TEST VALUE in development;

#define UNIT\_DELAY\_WDT 8 //<=== SELECTED VALUE in production; WDT period in seconds

#define SET\_DELAY\_UNIT 4 // <=== TEST VALUE in development

//#define SET\_DELAY\_UNIT 15 // # of UNIT\_DELAY\_WDT, Max 253: 2 min when UNIT\_DELAY\_WDT is 8

//#define SET\_DELAY\_UNIT 150 // # of UNIT\_DELAY\_WDT, Max 253: 20 min

//#define SET\_DELAY\_UNIT 225 //<=== SELECTED VALUE in production; # of UNIT\_DELAY\_WDT, Max 253: 30 min

//#define SET\_DELAY\_UNIT 254 // # of UNIT\_DELAY\_WDT, Max 253: 34 min - 8 sec

#define WDTTICK\_CTR\_ADDR 0

uint8\_t WDTtick = 0;

uint8\_t WDTtickMeasure = 0;

ISR(WDT\_vect)

{

// On every watch dog timer interrupt,

// get the WDTtick counter value every UNIT\_DELAY\_WDT sec

// from the designated EEPROM address

WDTtick = eeprom\_read\_byte((uint8\_t\*)WDTTICK\_CTR\_ADDR);

// increase WDTtick every UNIT\_DELAY\_WDT sec

// and update it at the designated EEPROM address

eeprom\_update\_byte((uint8\_t\*)WDTTICK\_CTR\_ADDR, ++WDTtick);

// When the accumulated WDT reaches every SET\_DELAY\_UNIT

if (WDTtick % SET\_DELAY\_UNIT == 0) //every UNIT\_DELAY\_WDT \* SET\_DELAY\_UNIT sec

//if (WDTtick == SET\_DELAY\_UNIT) //every UNIT\_DELAY\_WDT \* SET\_DELAY\_UNIT sec

{

// Give logic HIGH to port 4 to turn ON NPN transistor(2n2222),

// so let the GND of ESP-01 module CONNECT to system GND.

// This will power ON ESP-01 and measure temperature & humidity via DHT22

PORTB = 1<<PB4;

// save current WDT counter number to WDTtickMeasure

WDTtickMeasure = WDTtick;

}//if (WDTtick % SET\_DELAY\_UNIT == 0)

// When meeting the next tick after turning on ESP-01

//if (WDTtick == SET\_DELAY\_UNIT + 1)

if (WDTtick == WDTtickMeasure + 2)

{

// Give logic LOW to port 4 to turn OFF NPN transistor(2n2222),

// so let the GND of ESP-01 module DISCONNECT to system GND.

// This will power OFF ESP-01 and don't measure temperature & humidity via DHT22

PORTB = 0<<PB4;

// Reset WDT counter value of the designated address in the EEPROM of ATtiny13A

eeprom\_update\_byte((uint8\_t\*)WDTTICK\_CTR\_ADDR, 0);

// Reset relevant variables

WDTtickMeasure = 255;

WDTtick = 1;

}// if (WDTtick == WDTtickMeasure + 1)

}//ISR(WDT\_vect)

int main(void) {

// Set up Port B pin 4 mode to output

DDRB = 1<<DDB4;

// temporarily prescale timer to UNIT\_DELAY\_WDT seconds so we can measure current

switch (UNIT\_DELAY\_WDT)

{

case 4:

WDTCR |= (1<<WDP3); // 4s

break;

case 8:

WDTCR |= (1<<WDP3) | (1<<WDP0); // 8s

break;

default:

WDTCR |= (1<<WDP3) | (1<<WDP0); // 8s

}

// (1<<WDP2) | (1<<WDP0);

// Enable watchdog timer interrupts

WDTCR |= (1<<WDTIE);

sei(); // Enable global interrupts

// Reset the WDTtick at the designated EEPROM address

eeprom\_update\_byte((uint8\_t\*)WDTTICK\_CTR\_ADDR, 0);

// Use the Power Down sleep mode

set\_sleep\_mode(SLEEP\_MODE\_PWR\_DOWN);

for (;;) {

sleep\_mode(); // go to sleep and wait for interrupt...

}

}//main