ATtiny13a -시리얼모니터 (ATtiny13a-Serial monitor)

PN: 23188760

차례

[1 목적 2](#_Toc524638745)

[2 테스트 결과 2](#_Toc524638746)

[3 참조 5](#_Toc524638747)

[4 코드 5](#_Toc524638748)

# 목적

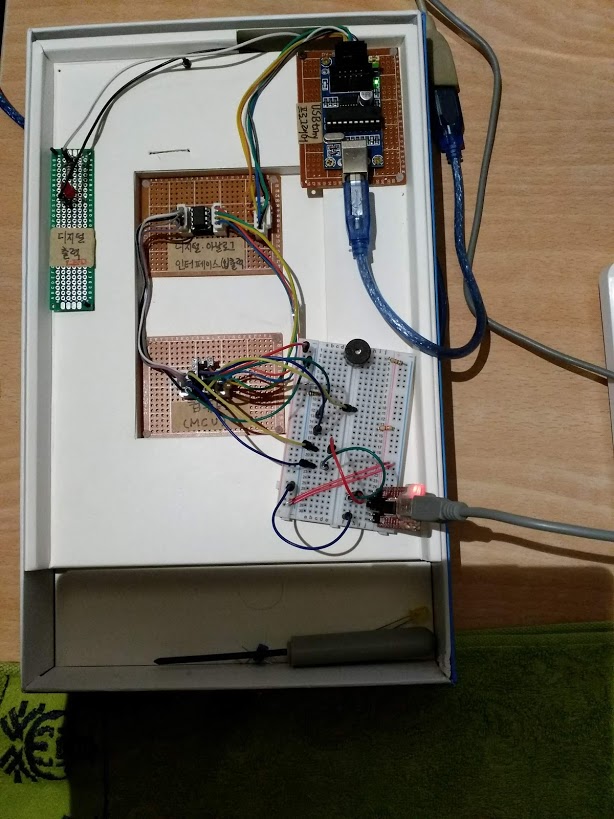
ATtiny13a MCU의 개발환경 구축할 때, 시리얼모니터가 있으면 좋겠다는 생각을 많이 한다. 252바이트의 적은 용량으로 구현할 수 있는 AVR코드를 설명한다.

# 테스트 결과-FTDi232활용

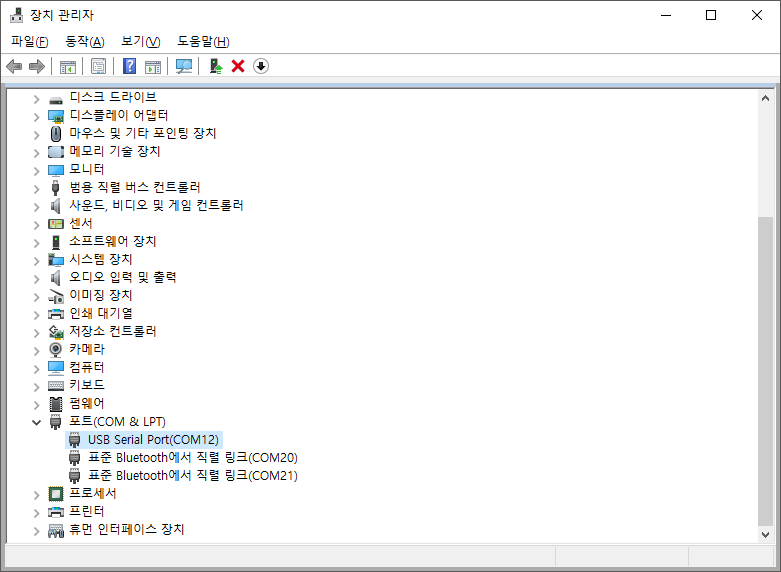
## 실험 사진

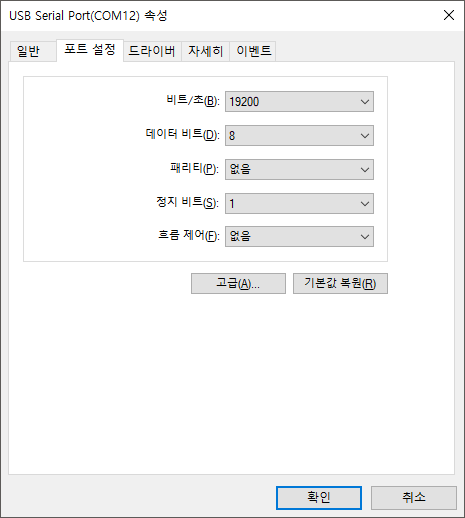
ATtiny13a MCU가 윗쪽 만능기판에 있고, USBtiny 프로그래머를 통해서 ATmel Studio에서 빌드된 hex파일을 업로드 받는다. PB3 / 4는 각각 Tx/Rx로 동작하게 코딩 하였다.

한편, PC에는 FTDi232 USB-serial 모듈을 통해서 시리얼 포트가 연결되어 있고, 이는 제어판-시리얼포트에서19200 bps로 설정한다. PB3(Tx) – FTDi232 (Rx) / PB4(Rx) – FTDi232 (Tx)에 연결한다. 그리고, MobaXterm을 구동해서 해당 시리얼포트를 연다.



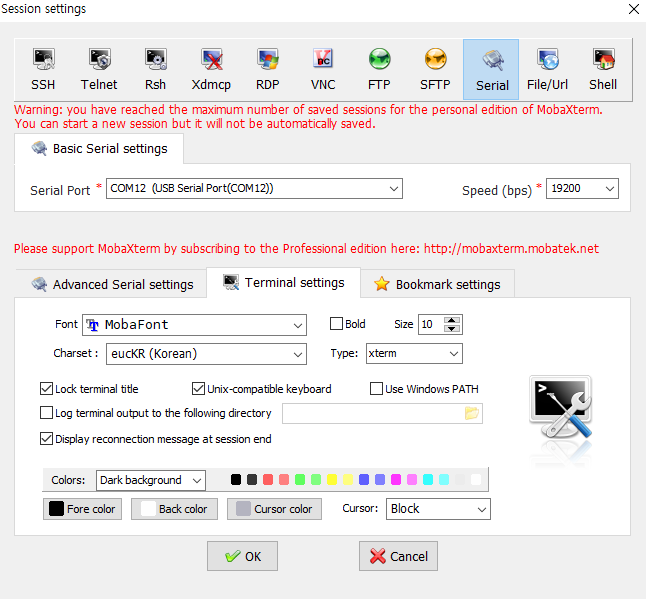
## PC제어판에서 USB-Serial포트 설정





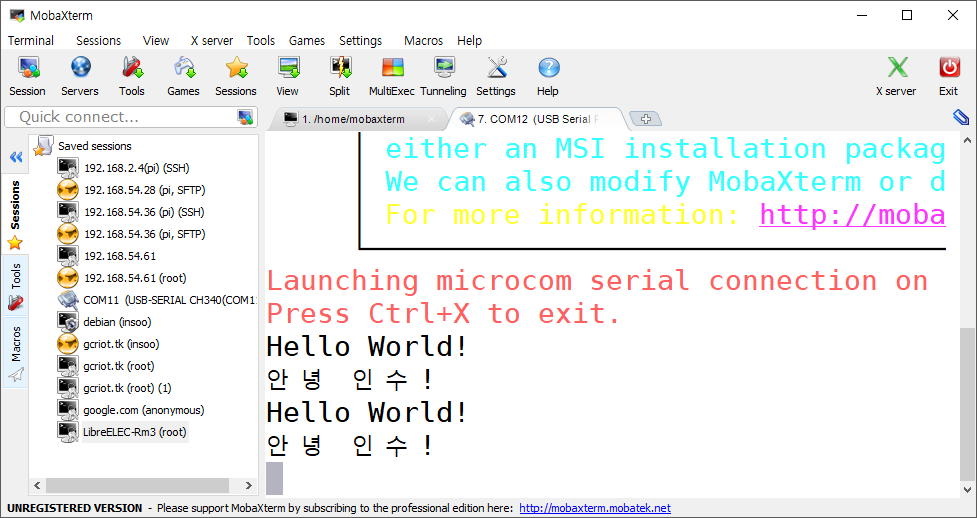
## MobaTerm시리얼 터미널 설정

한글도 시리얼통신으로 전송되는지 확인하기 위해 터미널 문자집합(charset)을 한글(eucKR)로 설정.



## PC시리얼 모니터로 ATtiny13a에서 문자를 출력한 결과

한글도 잘 전송된다.

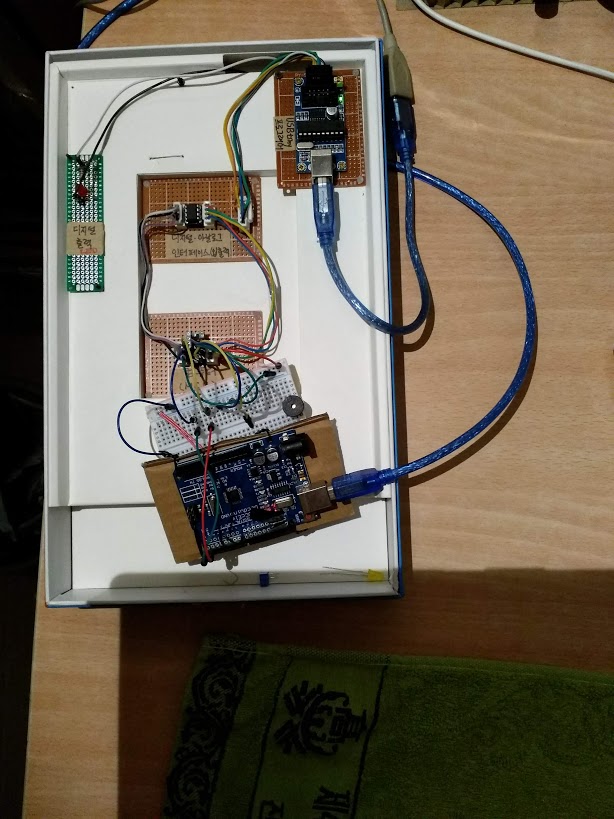


# 테스트 결과-아두이노 시리얼포트 활용

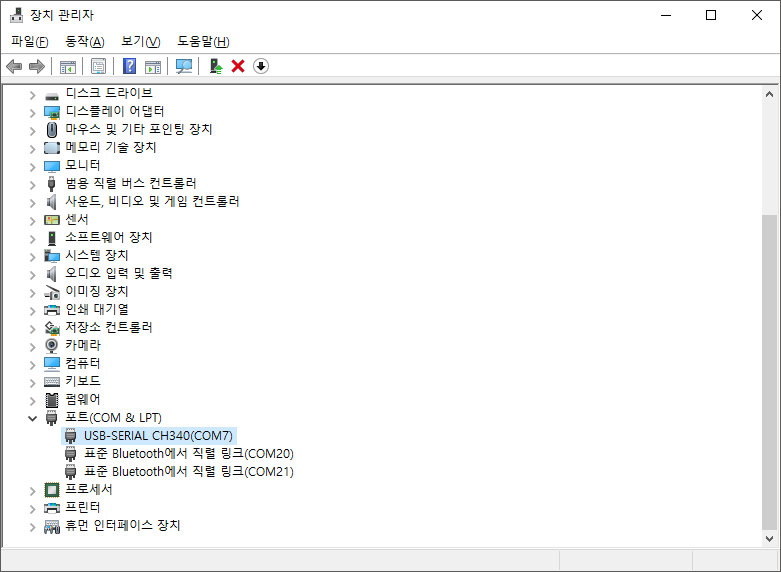
## 실험 사진

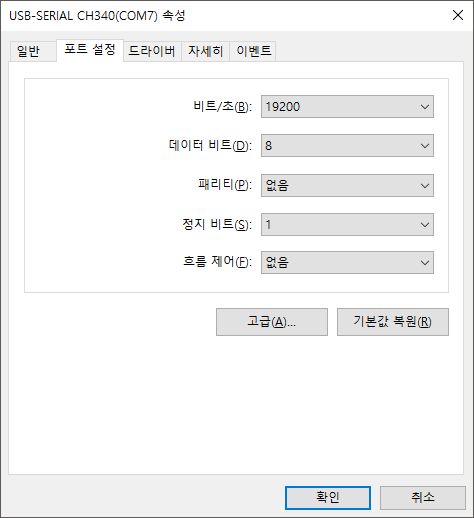
ATtiny13a MCU가 윗쪽 만능기판에 있고, USBtiny 프로그래머를 통해서 ATmel Studio에서 빌드된 hex파일을 업로드 받는다. PB3 / 4는 각각 Tx/Rx로 동작하게 코딩 하였다.

한편, PC에는 아두이노 USB-serial 모듈을 통해서 시리얼 포트가 연결되어 있고, 이는 제어판-시리얼포트에서19200 bps로 설정한다. PB3(Tx) – 아두이노 (Rx 즉, 2번핀) / PB4(Rx) – 아두이노 (Tx 즉, 1번핀)에 연결한다. 그리고, MobaXterm을 구동해서 해당 시리얼포트를 연다.



## PC제어판에서 아두이노USB-Serial포트 설정





## MobaTerm시리얼 터미널 설정

FTDi232환경에서 사용한 설정과 동일.

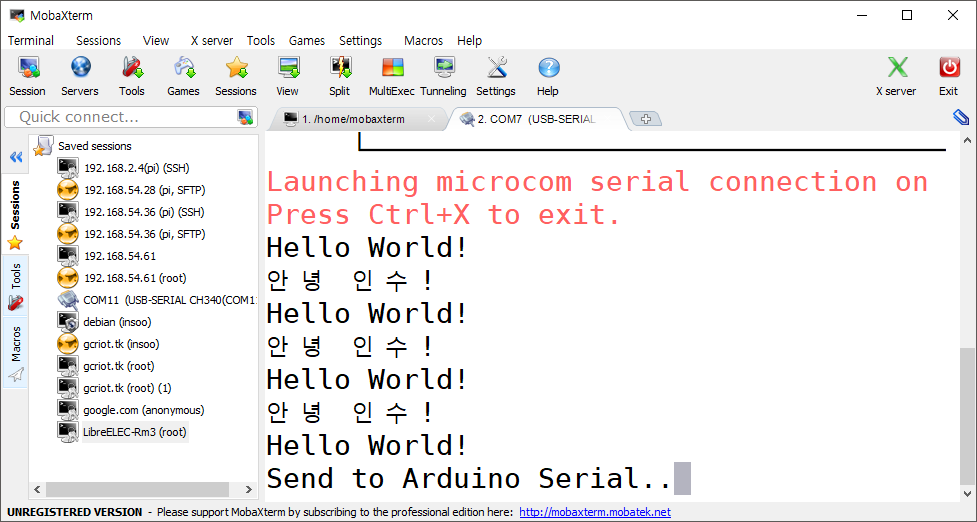
## MobaXterm시리얼 모니터로 ATtiny13a에서 문자를 출력한 결과

main()함수에서 아래와 같이 변경 후 빌드 및 업로드.

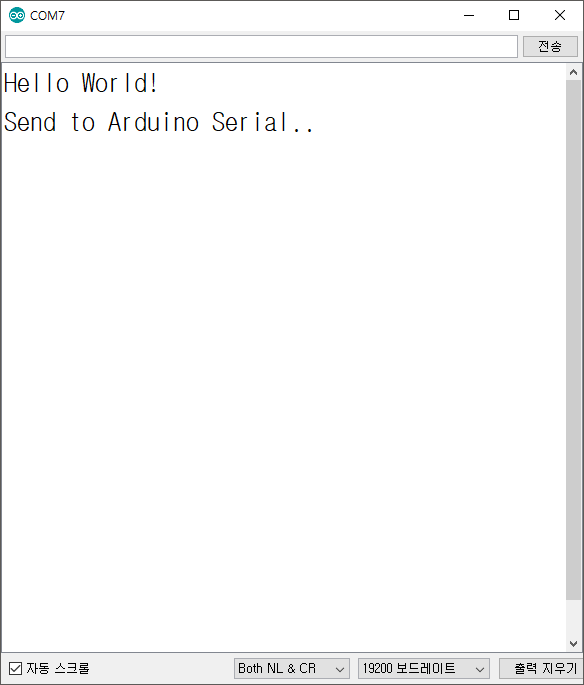
uart\_puts("Hello World! \r\n");

//uart\_puts("안녕 인수! \r\n");

uart\_puts("Send to Arduino Serial... \r\n");



## 아두이노 시리얼 모니터로 ATtiny13a에서 문자를 출력한 결과



# 참조

## 매우 큰 도움된 자료

<http://blog.podkalicki.com/attiny13-software-uart-debug-logger/>

# 코드

참조 원본코드에서 F\_CPU에 대해 #ifdef ~ #endif 부분만 수정 변경함.

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

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 : Sep 11, 2018 (On Atmel Studio 7)

Updated : Sep 13, 2018 (On Atmel Studio 7)

Description:

software serial for ATtiny13A.

HEX size[Byte]: 864 out of 1024

How to upload to the target MCU

<For Windows Atmel Studio>

Select Tool - USBtiny (USBtiny menu should be configured in the external tool menu)

<For CMD window or DOS prompt>

cd "C:\Users\insoo\Documents\GitHub\ATmelStudio\ATtiny13A\ClockGen\TMRCNT-Sound-23188757\Debug"

avrdude -c usbtiny -P usb -p attiny13 -U flash:w:TMRCNT-Sound-23188757.hex:i

Ref:

Codes:

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

/\*\*

\* Copyright (c) 2016, Łukasz Marcin Podkalicki <lpodkalicki@gmail.com>

\* ATtiny13/007

\* Simple tone generator.

\*/

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define UART\_RX\_ENABLED (1) // Enable UART RX

#define UART\_TX\_ENABLED (1) // Enable UART TX

#define F\_CPU (1200000UL) // 1.2 MHz

#if defined(UART\_TX\_ENABLED) && !defined(UART\_TX)

#define UART\_TX PB3 // Use PB3 as TX pin

#endif /\* !UART\_TX \*/

#if defined(UART\_RX\_ENABLED) && !defined(UART\_RX)

#define UART\_RX PB4 // Use PB4 as RX pin

#endif /\* !UART\_RX \*/

#if (defined(UART\_TX\_ENABLED) || defined(UART\_RX\_ENABLED)) && !defined(UART\_BAUDRATE)

#define UART\_BAUDRATE (19200)

#endif /\* !UART\_BAUDRATE \*/

#define TXDELAY (int)(((F\_CPU/UART\_BAUDRATE)-7 +1.5)/3)

#define RXDELAY (int)(((F\_CPU/UART\_BAUDRATE)-5 +1.5)/3)

#define RXDELAY2 (int)((RXDELAY\*1.5)-2.5)

#define RXROUNDED (((F\_CPU/UART\_BAUDRATE)-5 +2)/3)

#if RXROUNDED > 127

# error Low baud rates unsupported - use higher UART\_BAUDRATE

#endif

static char uart\_getc();

static void uart\_putc(char c);

static void uart\_puts(const char \*s);

int

main(void)

{

char c, \*p, buff[16];

uart\_puts("Hello World! \r\n");

uart\_puts("안녕 인수! \r\n");

/\*

uart\_putc("K");

uart\_putc("I");

uart\_putc("M");

uart\_putc("\n");

\*/

/\* loop \*/

while (1) {

p = buff;

while((c = uart\_getc()) != '\n' && (p - buff) < 16) {

\*(p++) = c;

}

\*p = 0;

\_delay\_ms(10);

uart\_puts(buff);

}

}

char

uart\_getc(void)

{

#ifdef UART\_RX\_ENABLED

char c;

uint8\_t sreg;

sreg = SREG;

cli();

PORTB &= ~(1 << UART\_RX);

DDRB &= ~(1 << UART\_RX);

\_\_asm volatile(

" ldi r18, %[rxdelay2] \n\t" // 1.5 bit delay

" ldi %0, 0x80 \n\t" // bit shift counter

"WaitStart: \n\t"

" sbic %[uart\_port]-2, %[uart\_pin] \n\t" // wait for start edge

" rjmp WaitStart \n\t"

"RxBit: \n\t"

// 6 cycle loop + delay - total = 5 + 3\*r22

// delay (3 cycle \* r18) -1 and clear carry with subi

" subi r18, 1 \n\t"

" brne RxBit \n\t"

" ldi r18, %[rxdelay] \n\t"

" sbic %[uart\_port]-2, %[uart\_pin] \n\t" // check UART PIN

" sec \n\t"

" ror %0 \n\t"

" brcc RxBit \n\t"

"StopBit: \n\t"

" dec r18 \n\t"

" brne StopBit \n\t"

: "=r" (c)

: [uart\_port] "I" (\_SFR\_IO\_ADDR(PORTB)),

[uart\_pin] "I" (UART\_RX),

[rxdelay] "I" (RXDELAY),

[rxdelay2] "I" (RXDELAY2)

: "r0","r18","r19"

);

SREG = sreg;

return c;

#else

return (-1);

#endif /\* !UART\_RX\_ENABLED \*/

}

void

uart\_putc(char c)

{

#ifdef UART\_TX\_ENABLED

uint8\_t sreg;

sreg = SREG;

cli();

PORTB |= 1 << UART\_TX;

DDRB |= 1 << UART\_TX;

\_\_asm volatile(

" cbi %[uart\_port], %[uart\_pin] \n\t" // start bit

" in r0, %[uart\_port] \n\t"

" ldi r30, 3 \n\t" // stop bit + idle state

" ldi r28, %[txdelay] \n\t"

"TxLoop: \n\t"

// 8 cycle loop + delay - total = 7 + 3\*r22

" mov r29, r28 \n\t"

"TxDelay: \n\t"

// delay (3 cycle \* delayCount) - 1

" dec r29 \n\t"

" brne TxDelay \n\t"

" bst %[ch], 0 \n\t"

" bld r0, %[uart\_pin] \n\t"

" lsr r30 \n\t"

" ror %[ch] \n\t"

" out %[uart\_port], r0 \n\t"

" brne TxLoop \n\t"

:

: [uart\_port] "I" (\_SFR\_IO\_ADDR(PORTB)),

[uart\_pin] "I" (UART\_TX),

[txdelay] "I" (TXDELAY),

[ch] "r" (c)

: "r0","r28","r29","r30"

);

SREG = sreg;

#endif /\* !UART\_TX\_ENABLED \*/

}

void

uart\_puts(const char \*s)

{

while (\*s) uart\_putc(\*(s++));

}