

Chapter 17. RTC

컴퓨터에는 배터리에 의해 동작하는 시계가 내장되어 있어 컴퓨터 전원을 내렸다가 올린 뒤에도 날짜와 시간 정보가 유지된다. 이처럼 전용 전원을 통해 날짜와 시간을 유지하는 장치를 RTC(Real Time Clock)라 한다. 하지만 아두이노 메가2560의 경우에는 RTC가 포함되어 있지 않으므로 현재 날짜와 시간을 사용하기 위해서는 타이머를 이용하는 소프트웨어적인 방법이나 별도의 모듈을 이용하는 하드웨어적인 방법을 사용하여야 한다. 소프트웨어 라이브러리를 사용하는 경우에는 아두이노의 전원을 끄면 시간 정보가 유지되지 않으므로 매번 시간을 설정하여야 하는 불편함이 있다. 별도 모듈을 이용하게 되면 이런 불편은 없지만 별도의 추가 장치가 필요하다.

1. 소프트웨어 RTC

아두이노 우노 보드에 사용된 ATmega328 마이크로컨트롤러에는 timer0, timer1, timer2의 3개 타이머가 내장되어 있다. 이 중 timer0과 timer2는 8비트 타이머이며 timer1은 16비트 타이머이다. ATmega2560 마이크로컨트롤러에는 3개의 16비트 타이머가 더 포함되어 있다. 타이머는 마이크로컨트롤러의 클럭 주파수에 영향을 받으므로 아두이노 우노와 메가2560의 타이머는 16MHz로 동작하게 된다. 아두이노의 함수 중에서 타이머를 명시적으로 사용하는 함수는 없지만 많은 아두이노의 함수들이 타이머에 의존하고 있으며 PWM 출력 역시 마찬가지다. 따라서 타이머 관련 함수를 사용하는 경우 해당 핀으로 PWM 출력은 사용할 수 없다.

	아두이노 우노 PWM 핀	아두이노 메가2560 PWM 핀	관련 함수
timer0	5, 6	4, 13	delay, millis, micros
timer1	9, 10	11, 12	서보 모터 제어
timer2	11, 3	9, 10	tone
timer3	-	2, 3, 5	
timer4	-	6, 7, 8	
timer5	-	46, 45, 44	

표 1. 타이머와 PWM 출력 핀의 연관성

이 장에서 사용할 소프트웨어 RTC 라이브러리는 swRTC 라이브러리로 timer2를 사용하므로 timer2를 사용한다. swRTC 라이브러리를 다운받아¹⁾ 설치하자. 설치가 완료되면 '스케치 → Include Library → swRTC' 메뉴를 선택하여 해당 헤더 파일을 포함시킨다.

```
#include <core_build_options.h>
#include <swRTC.h>
```

헤더 파일이 포함되면 swRTC 사용을 위해 객체를 생성한다.

```
swRTC rtc;
```

생성된 swRTC는 setTime 함수와 setDate 함수를 통해 시간과 날짜를 초기화시킨다.

```
boolean setTime(byte hourT, byte minuteT, byte secondT)
boolean setDate(byte dayT, byte monthT, int yearT)
```

초기화가 끝나면 startRTC 함수로 시계가 동작하도록 할 수 있으며, stopRTC 함수로 시계의 동작을 일시 멈출 수 있다.

```
void startRTC(void)
void stopRTC(void)
```

현재 시간과 날짜를 알아내기 위해서는 다음 함수들을 사용하면 된다.

1) <http://www.leonardomiliani.com/2011/swrtc-un-orologio-in-tempo-reale-via-software/>
swRTC 라이브러리 1.2 버전의 경우 swRTC.h 파일의 92라인에서 반환 유형을 'boolean'에서 'byte'로 변경하여야 정상적인 컴파일이 가능하다.

```
byte getSeconds(void)
byte getMinutes(void)
byte getHours(void)
byte getDay(void)
byte getMonth(void)
int getYear(void)
```

이외에도 swRTC 클래스에는 날짜와 시간 관련 함수들이 준비되어 있으므로 swRTC.h 파일에 정의된 함수들을 확인하도록 한다. 스케치 1은 swRTC 클래스에서 제공하는 메서드들을 사용하여 현재 시간을 알아내는 스케치의 예이다.

스케치 1. swRTC 사용

```
#include <core_build_options.h>
#include <swRTC.h>

swRTC rtc;           // swRTC 인스턴스 생성

void setup() {
  rtc.stopRTC();      // RTC 정지
  rtc.setTime(15, 0, 0); // 15시 0분 0초로 설정
  rtc.setDate(6, 5, 2015); // 2015년 5월 6일로 설정
  rtc.startRTC();     // RTC 시작
  Serial.begin(9600); // 시리얼 통신 초기화
}

void loop() {
  Serial.print(rtc.getHours(), DEC); // 시
  Serial.print(":");
  Serial.print(rtc.getMinutes(), DEC); // 분
  Serial.print(":");
  Serial.print(rtc.getSeconds(), DEC); // 초
  Serial.print(" -- ");
```

```
Serial.print(rtc.getDay(), DEC);          // 일
Serial.print("/");
Serial.print(rtc.getMonth(), DEC);       // 월
Serial.print("/");
Serial.print(rtc.getYear(), DEC);         // 년

Serial.print(" (Leap year: ");           // 윤년 여부
if (rtc.isLeapYear())
    Serial.print("yes");
else
    Serial.print("no");
Serial.println("");

Serial.print("Day of week: ");
Serial.println(rtc.getDay(), DEC);       // 요일, 일요일이면 0

delay(1000);
}
```

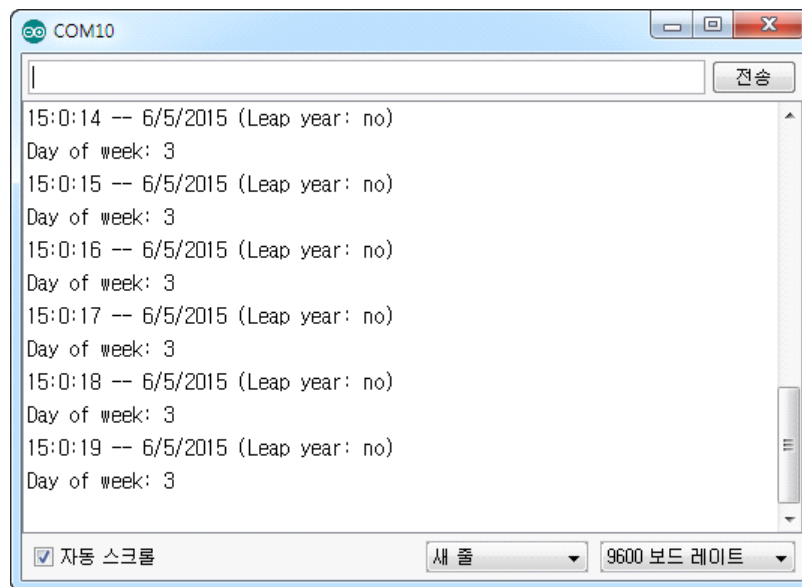


그림 1. 스케치 1 실행 결과

2. 하드웨어 RTC

RTC 하드웨어는 전용의 발진기를 통해 시간을 독립적으로 유지하며, 전용 전원을 사용하므로 마이크로컨트롤러의 전원이 꺼진 경우에도 시간을 유지할 수 있는 장점이 있다. 하드웨어 RTC는 I2C 통신을 사용하는 DS1307이 흔히 사용된다. I2C는 저속의 주변 기기를 연결하기 위한 프로토콜로 SCL(serial clock)과 SDA(serial data) 두 개의 연결을 통해 주변 장치와 통신한다. 아두이노 메가2560의 경우 SCL을 디지털 21번 핀, SDA를 디지털 20번 핀에 연결하여 사용하여야 한다. DS1307 RTC의 경우 32.768KHz 크리스털을 사용하며 3V의 전용 전원을 필요로 한다.

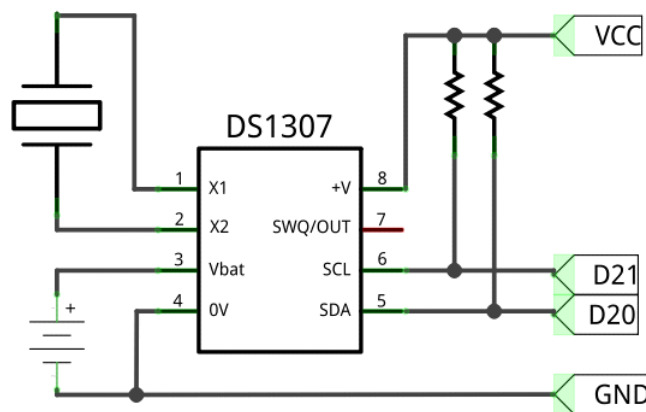


그림 2. DS1307 연결 회로도

학습보드에는 DS1307 RTC 칩과 RTC용 배터리 홀더가 포함되어 있다.

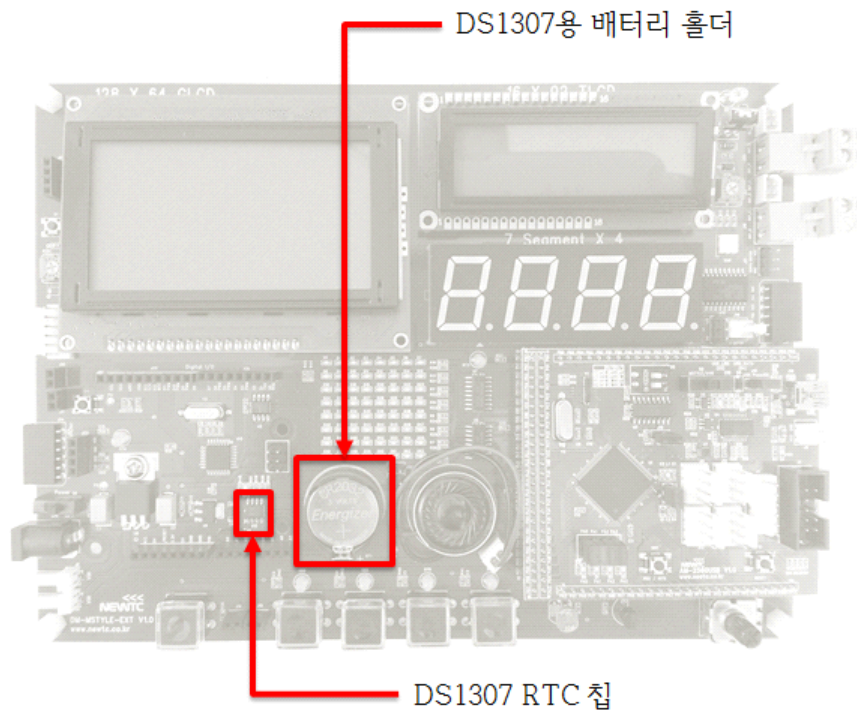


그림 3. 학습보드에 연결된 RTC와 배터리 홀더

DS1307 칩을 사용하기 위해서는 먼저 RTCLib 라이브러리를 다운받아²⁾ 설치하여야 한다. 라이브러리를 설치하고 RTC를 사용하기 위해서는 먼저 RTCLib 라이브러리 가져오기를 통해 해당 헤더 파일을 포함시킨다. RTC는 I2C 통신을 사용하므로 아두이노에서 제공하는 Wire 라이브러리도 포함시켜야 한다.

```
#include "Wire.h"
#include <RTCLib.h>
```

헤더 파일이 포함되면 먼저 RTC_DS1307 클래스의 객체를 생성한다.

```
RTC_DS1307 RTC;
```

생성된 객체는 setup 함수에서 RTC의 초기화한다. 이 때 Wire 라이브러리 역시 초기화시켜

2) <https://github.com/adafruit/RTCLib>

주어야 한다.

```
Wire.begin();
RTC.begin();
```

시간 설정은 초기화 함수인 begin에서도 실행 가능하지만 시간 설정을 위한 별도의 adjust 함수를 제공하고 있다. 이 때 매개변수는 DateTime 클래스 형식으로 DateTime 클래스는 날짜와 시간을 다루기 위한 보조 클래스이다. DateTime 클래스의 생성자는 다음과 같다.

```
// 2000년 1월 1일 0시 0분 0초로 초기화
DateTime (uint32_t t = 0);
// 지정한 날짜와 시간으로 초기화. 시분초를 지정하지 않으면 0시 0분 0초로 초기화
DateTime (uint16_t year, uint8_t month, uint8_t day,
          uint8_t hour = 0, uint8_t min = 0, uint8_t sec = 0);
// date = "Dec 26 2009", time = "12:34:56" 형식으로 초기화
// RTC.adjust(DateTime(__DATE__, __TIME__)); 으로 지정하여
// 컴퓨터 시간으로 초기화 할 수 있다.
DateTime (const char* date, const char* time);
```

시작 시간이 설정되면 현재 날짜와 시간은 RTC_DS1307 클래스의 now 함수로 얻어올 수 있으며 반환값은 DateTime 클래스 형식이다. 반환값을 통해 연월일시분초를 알아내는 방법을 보여주는 예가 스케치 2이다.

스케치 2. 하드웨어 RTC 사용

```
#include <Wire.h>           // I2C 라이브러리
#include <RTCLib.h>          // RTC 라이브러리

RTC_DS1307 RTC;             // RTC 객체

void setup () {
  Serial.begin(9600);        // 직렬 통신 초기화
```

```
Wire.begin();           // I2C 라이브러리 초기화
RTC.begin();            // RTC 초기화

// 컴퓨터의 컴파일 시간으로 RTC를 초기화
// RTC.adjust(DateTime(__DATE__, __TIME__));
// 2015년 4월 15일 0시 0분 0초로 초기화
RTC.adjust(DateTime(2015, 4, 6, 15, 0, 0));
}

void loop () {
  if(!RTC.isrunning()){
    // RTC가 동작하지 않으면 오류 메시지 출력
    Serial.println("RTC is NOT running!");
  }
  else{
    DateTime now = RTC.now();           // 현재 날짜와 시간 얻기
    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');

    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
  }
  delay(1000);
}
```

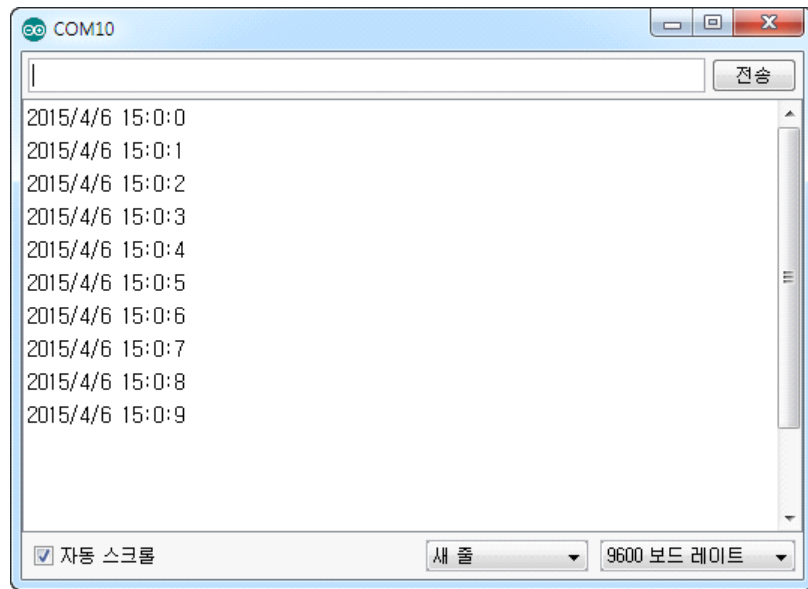


그림 4. 스케치 2 실행 결과