The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 18, No. 6, pp.133-137, Dec. 31, 2018. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2018.18.6.133
JIIBC 2018-6-17

아두이노 멀티 태스킹을 위한 수퍼루프 방식과 FreeRTOS 방식의 비교 분석

Comparative Analysis between Super Loop and FreeRTOS Methods for Arduino Multitasking

공동환*, 신승중**

Dong-Hwan Gong*, Seung-Jung Shin**

요 약 아두이노는 소형 마이컴으로 다양한 산업에 사용되고 있으며 특히, 오픈소스 하드웨어 IoT 디바이스로 널리 사용되고 있다. 아두이노의 멀티태스킹 방식은 크게 수퍼루프 타이밍과 RTOS 쓰레드 방식으로 나뉘며 수퍼루프 타이밍 방식은 구현이 단순하고 이해하기 쉽다는 장점이 있지만 하나의 작업이 길어지면 다음 작업의 실행에 영향을 줄 수 있다는 단점을 가진다. 또 RTOS 쓰레드 방식은 다른 작업시간에 영향을 받지 않고 실행할 수 있다는 장점을 갖지만 소형 마이컴인 아두이노는 쓰레드의 개수가 늘어나면 쓰레드의 컨텍스트 스위칭타임으로 수퍼루프 타이밍 방식에는 없는 부가 시간이 발생하는 단점이 있다. 본 논문은 이와 같은 서로 다른 특징들을 분석하기 위하여 아두이노 우노 R3와 FreeRTOS를 사용하였으며 실험을 위한 태스크는 빌트인 LED 포트에 8000번의 디지털 신호를 보내도록 작성하였다. 같은 크기의 태스크를 두 방식으로 실행하면 수퍼루프 방식이 FreeRTOS 멀티태스킹 보다 3ms 빠른 실행을 보인다. 여러 개의 태스크를 동시에 실행하면 수퍼루프 방식의 태스크는 순차 실행으로 첫 태스크와 마지막 태스크의 실행시간 차가 크게 나타나며 FreeRTOS 방식은 모두 중첩되어 동시에 실행 가능하지만 30ms 정도의 컨텍스트 스위칭타임의 실행 시간지연이 발생한다.

Abstract Arduino is a small microcomputer that is used in a variety of industry fields and especially is widely used as an open source hardware IoT device. The multi-tasking method of Arduino is divided into super loop timing and RTOS thread method. The super loop timing method is simple and easy to understand. However, when one task is long, it affects the execution of the next task. In addition, RTOS threading has the advantage of being able to run without being influenced by other work time. However, Arduino, a small microcomputer, has a disadvantage in that, when the number of threads increases, the context switching time of the thread causes additional time not included in the super loop timing method have. In this paper, we use Arduino Uno R3 and FreeRTOS to analyze these different features, and the task for the experiment is to send 8000 digital signals to the built-in LED port. If two tasks of the same size are executed, the super loop method executes 3 ms faster than FreeRTOS multitasking. If multiple tasks are executed simultaneously, superloop type task is sequential execution and difference in execution time between first task and last task is large. FreeRTOS method can be executed concurrently, but execution time delay of about 30 ms occurs in context switching time.

Key Words : IoT, Arduino, FreeRTOS, RTOS, Multitasking, Analysis.

*정회원, 한양사이버대학 기계자동차공학부
**정회원, 한세대학교 ICT디바이스학과
접수일자: 2018년 9월 7일, 수정완료: 2018년 11월 7일
게재확정일자: 2018년 12월 7일

Received: 7 September, 2018 / Revised: 7 November, 2018 / Accepted: 7 December, 2018
*Corresponding Author: expersin@hansei.ac.kr
Dept. of IT, Hansei University, Korea

Ⅰ. 서 론

아두이노는 8비트 마이컴 보드로 오픈소스 하드웨어 기반의 하드웨어로서 누구나 특정 환경에 맞게 변형하여 개발할 수 있어 소형기기가 필요한 다양한 산업에 응용, 개발되어 지고 있다. 아두이노는 AVR을 기반으로 개발이 시작되어 지금은 ARM 계열의 제품까지 다양한 형태의 하드웨어가 존재한다. 아두이노는 하드웨어 뿐만 아니라 소프트웨어 개발도구인 통합 개발 환경(IDE)를 제공하여 아두이노 하드웨어에 맞는 빠른 소프트웨어 개발과 업데이트가 가능하도록 한다.[11],[12]

아두이노가 다양한 산업에 사용되면서 하나의 하드웨어에서 여러 가지 작업을 수행하는 동시성이 요구되며 이런 여려 작업의 동시성을 충족하기 위하여 아두이노는 비RTOS방식의 멀티태스킹과 RTOS방식의 멀티태스킹으로 구현할 수 있다. 아두이노의 비RTOS의 주요 방법으로 수퍼루프 방법이 사용되며 RTOS의 쓰레드 방식으로는 FreeRTOS 방식이 사용된다.

본 논문은 아두이노의 멀티태스킹을 위한 비RTOS 방식인 수퍼루프 방식과 RTOS 방식인 아두이노로 포팅(porting)된 AVR용 오픈소스 기반의 FreeRTOS를 사용하여 각 방식의 특징과 장단점을 비교한다. 또 각 방식의 특징들로 발생할 수 있는 소프트웨어적인 이슈를 정리하고 성능을 분석함으로서 아두이노의 멀티태스킹 구현 시 발생할 수 있는 문제들을 미연에 방지하고자 한다.[1],[4]

Ⅱ. 오픈 소스 FreeRTOS

오픈 소스(Open Source)는 전 세계 누구나 쉽게 자유롭게 소프트웨어를 개발하고 개발에 참여할 수 있도록 리차드 스톨만(Richard Stallman)이 주축이 되어 시작된 사회운동으로 오픈소스 라이선스를 통칭하는 용어이다. FreeRTOS는 오픈소스 운영체제 중 하나로 마이크로컨트롤러(MCU)에 사용하기 적합하도록 개발되었으며 실시간 운영체제의 뛰어난 이식성과 멀티태스킹 기능을 제공한다.[7],[9]

FreeRTOS는 실행(Running), 준비(Ready), 중지(Suspended), 대기(Blocked) 상태의 4가지를 가지며 그림1은 FreeRTOS의 태스크 전이 상태를 보여주는 태스크 상태머신이다.[9],[10]

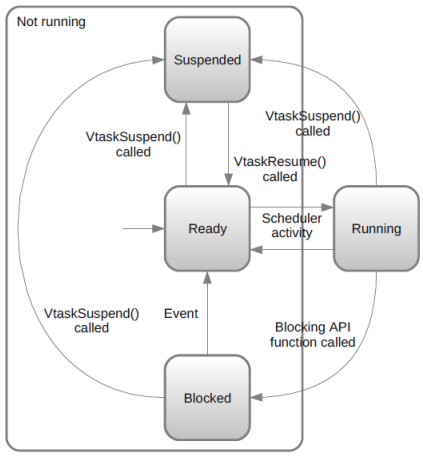

그림 1. FreeRTOS의 태스크 상태머신
Fig. 1. Task State Machine in FreeRTOS

기본적으로 태스크를 생성하기 위해서는 xTaskCreate() 함수를 호출하고 실행할 함수와 이름, 스택 사이즈, 우선순위 등을 인수로 전달한다. 생성된 태스크는 vTaskStartScheduler() 함수를 호출하기 전까지 실행되지 않는다. 주의할 점은 아두이노의 setup()함수가 종료되지 않도록 하고 loop()함수에는 어떤 작업도 없도록 빈 함수로 아두이노에 업로드 해야한다.[9]

실행(Running) 상태는 실행되고 있는 태스크의 상태로 스케줄러에 선택받은 태스크이다.

준비(Ready) 상태는 실행이 가능한 준비상태의 태스크로 일정한 스케줄링 정책에 따라 우선순위가 높은 순서대로 스케줄러에 의해 선택될 수 있는 태스크 상태를 나타낸다.

블록(Blocked) 상태는 큐나 세마포어 이벤트를 기다리거나 vTaskDelay()함수를 호출하여 지연 상태가 된 블록 상태로 블록 시간이 완료되거나 외부 이벤트가 완료될 때 까지 대기하는 상태를 의미하며 태스크 스케줄링 대상에서 제외된다.

중지(Suspended) 상태는 명시적으로 vTaskSuspend( ) 함수가 호출되면 태스크를 중지하는 상태로 태스크 상태를 변경한다. 중지 상태를 다시 되돌리기 위해서는 명시적으로 vTaskResume() 함수를 호출하여 변경한다.

Ⅲ. 작업 환경 및 실행 구조

1. 수퍼루프 멀티태스킹 실행 구조

수퍼루프 멀티태스킹은 타이머 객체를 생성하고 setup() 함수에서 타이머 이벤트 시간과 태스크를 등록하고 loop() 함수에서 이벤트 시간을 계산하여 등록한 이벤트 시간이 완료하면 태스크가 실행되도록 구성하였다. 수퍼루프 멀티태스킹은 내부적으로 millis()함수를 사용하는 타이머를 사용하여 구현하였으며 실행하고자하는 타이밍 호출 시간에 따라 각각의 태스크를 실행하도록 구성하였다. 타이머 객체는 각 태스크가 10번 이상 반복되도록 구성하여 실험하였다.[2]

```
void Task1() 태스크1
{
   ...태스크 1번
}
void Task2() 태스크2
{
   ...태스크 2번
}

TimerObject t;

void setup() {  타이머 이벤트, 태스크 등록
   t.register(2000, Task1);
   t.register(1000, Task2);
...
}
void loop() {
   t.update(); 이벤트 시간 계산
}
```

그림 2. 수퍼루프 멀티태스킹 실행 구조
Fig. 2. Super loop multitasking execution structure

2. FreeRTOS 멀티태스킹 실행 구조

FreeRTOS는 FreeRTOS_AVR 라이브러리를 사용하였으며 setup() 함수에서 쓰레드에 해당하는 태스크를 생성하기 위해 xTaskCreate() 함수를 호출하여 실행할 태스크 함수를 등록하고 vTaskStartScheduler() 함수를 다음으로 호출하여 태스크를 실행하도록 구성하였다. 하나의 태스크는 일정한 시간동안 종료되지 않도록 루프를 실행하고 1초마다 모니터링을 위하여 시리얼모니터에 확인 값을 전송하도록 구성하였다.[12]

```
void Task1(void* arg) {
   while (1)  태스크1
   {
      ... 태스크 1번
   }
}
void Task2(void* arg) {
   while (1)  태스크2
   {
      ... 태스크 2번
   }
}
void setup()
{
      태스크1 생성, 등록
   s1 = xTaskCreate(Task1, ...);
      태스크2 생성, 등록
   s2 = xTaskCreate(Task2, ...);
      태스크 실행
   vTaskStartScheduler();
   while(1);
}
void loop()
{
}
```

그림 3. FreeRTOS 멀티태스킹 실행 구조
Fig. 3. FreeRTOS multitasking execution structure

Ⅳ. 실험 및 결과

1. 실험 목적과 환경

본 논문의 실험은 크게 두 가지로 나눌 수 있다. 하나는 같은 크기의 태스크를 수퍼루프 방식과 FreeRTOS 방식으로 나누어 실행시간 지연을 확인하는 것이고 또 하나는 FreeRTOS 멀티태스킹의 핵심인 쓰레드를 넘나들며 소모되는 부가 시간인 컨텍스트 스위칭 타임을 측정하기 위함이다. FreeRTOS 실행환경은 태스크를 독립적으로 실행하는 쓰레드를 최소 1개에서 최대 5개까지 늘려가며 실행했으며 모든 쓰레드는 같은 우선순위를 가지고 시간 측정을 위한 평균 반복 횟수는 10~15번으로 제한하였다.

2. 수퍼루프와 FreeRTOS 멀티태스킹 방식의 태스크 실행 비교

하나의 태스크를 싱행하면 수퍼루프 멀티태스킹 방식이 FreeRTOS 멀티태스킹 방식보다 3ms정도 빠른 속도를 보였으며 그림4와 같이 FreeRTOS 멀티태스킹 방식이 조금 더 고른 실행시간 속도를 보였다.

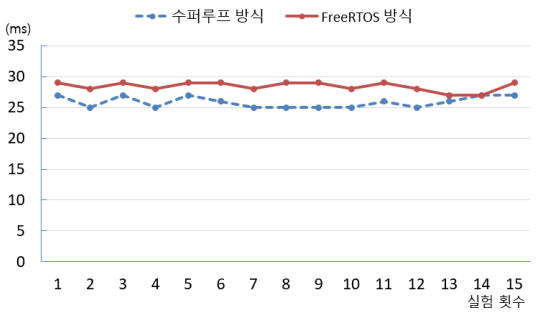

그림 4. 수퍼루프 방식과 FreeRTOS 멀티태스킹 방식의 태스크 실행시간 비교

Fig. 4. Comparing Task Execution Times of Super Loop and FreeRTOS Multitasking

그림 5를 보면 수퍼루프 멀티태스킹 방식은 순차실행으로 하나의 태스크가 실행될 때마다 실행 시간이 누적되어 큰 작업의 태스크를 실행하면 마지막 태스크는 점차 늦게 실행되어 지정시간에 실행되지 않는 실행 오류가 발생할 수 있다. FreeRTOS 멀티태스킹 방식은 각각의 태스크가 쓰레드에 의해 독립적으로 실행되며 모든 태스크가 균일한 시간에 동시에 실행된다는 장점이 있지만 쓰레드의 개수에 따라 일정한 실행 시간지연이 발생한다.

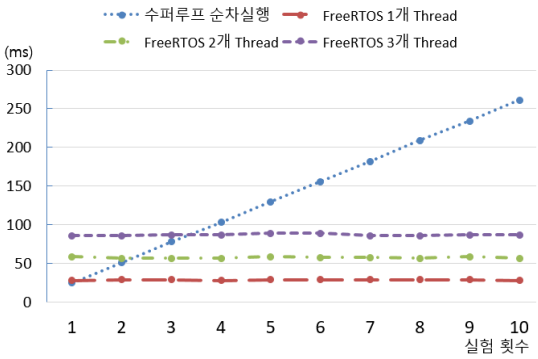

그림 5. 여러 태스크를 실행하는 수퍼루프 방식과 FreeRTOS 멀티태스킹 방식의 실행시간 비교

Fig. 5. Comparing the execution time of super loop method and FreeRTOS multitasking method that execute several tasks

그림 6은 FreeRTOS 멀티태스킹 방식으로 독립적인 태스크를 5개까지 쓰레드를 늘려가며 태스크의 실행시간을 비교한 그래프이다. 태스크의 실행시간은 일정한 시간에 실행되는 것을 확인할 수 있으며 쓰레드의 개수가 늘어나면 쓰레드의 컨텍스트 스위칭 타임으로 30ms정도의 실행 시간지연이 부가적으로 발생하는 것을 알 수 있다.

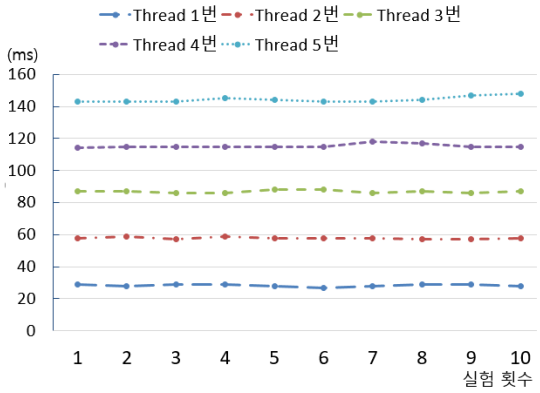

그림 6. FreeRTOS 멀티태스킹 방식의 쓰레드 개수에 따른 태스크 실행시간 비교

Fig. 6. Comparison of task execution time according to the number of FreeRTOS multitasking threads

V. 결론 및 향후 계획

본 논문에서는 아두이노의 수퍼루프 멀티태스킹과 FreeRTOS 멀티태스킹 방식의 실행시간 비교, 분석을 위해 임의의 태스크를 만들고 실험하였다. 결과로 하나의 태스크 실행시간은 수퍼루프 멀티태스킹 방식이 FreeRTOS 멀티태스킹 방식보다 3ms 정도의 빠른 실행시간을 보였으며 FreeRTOS 멀티태스킹은 독립적인 태스크를 실행하기 위해 쓰레드가 늘어날 때 마다 일정한 크기의 컨텍스트 스위칭 타임의 증가로 실행 시간지연이 누적되며 소요되는 것을 확인하였다.

향후 계획으로는 소형 컴퓨터와 오픈소스 하드웨어에서 멀티태스킹만큼 중요한 이슈 중 하나가 효율적인 네트워크 데이터 전송이다. 소형 컴퓨터에서 대용량 네트워크 전송 시 발생되는 부하테스트를 통해 전송 속도를 확인하고 효율적인 전송을 위한 설계 연구를 진행하고자 한다.

References

[1] Z. Cheng, Y. Li, and R. West. Qduino: A

multithreaded arduino system for embedded computing. In Proceedings of the 36th IEEE Real-Time Systems Symposium (RTSS 2015), San Antonio, Texas, December 2015.

[2] Dong-Hwan Gong, “A Study on Open Source Hardware Performance for IoT Devices”, Hansei Univ, 2017.

[3] T. P. Baker. Stack-based scheduling for realtime processes. Real-Time Systems, 3(1):67 - 99, April 1991.

[4] Galadima and A. Adamu, "Arduino as a learning tool", 2014 11th International Conf. on Electronics, Computer and Computation, pp. 1-4, Sep. 2014.

[5] OSHA(Open Source Hardware Association), https://www.oshwa.org/

[6] Y. Chemin, N. Sanjaya, P. K. N. C. Liyanage, “An Open Source Hardware & Software online raingauge for real-time monitoring of rainwater harvesting in Sri Lanka.” Symposium on Mainstreaming Rainwater Harvesting as a Water Supply Option, 2014.

[7] Freertos port for arduino. https://github.com/greiman/FreeRTOS-Arduino

[8] Arduino scheduler library. https://www.arduino.cc/en/Reference/Scheduler

[9] Nicolas Melot, “Study of an operating system : FreeRTOS”, http://wiki.csie.ncku.edu.tw/embedded/FreeRTOS_Melot.pdf

[10] FreeRTOS, http://www.freertos.org

[11] Arduino Playground. http://playground.arduino.cc/

[12] Wikipedia, https://www.wikipedia.org/

저자 소개

공 동 환(정회원)

- 2001년도 조선대학교 대학원 컴퓨터공학 졸업(석사)
- 2018년도 한세대학교 대학원 IT융합 졸업(박사)
- 2001년도 ~ 현재 비트컴퓨터 비트교육센터 강사
- 2017년도 ~ 현재 한양사이버대학 기계자동차공학부 겸임교수

<주관심분야> : IoT, 인공지능

신 승 중(종신회원, 교신저자)

- 1988년도 세종대학교 대학원 경영학과 졸업(석사)
- 1994년도 건국대학교 대학원 전자계산학과 졸업(석사)
- 1999년도 국민대학교 대학원 정보관리학과 졸업(박사)
- 1988~1995 중경공업전문대 전자과 겸임교수
- 1995년~2003 중부대학교 정보보호학과 부교수
- 2003~현재 한세대학교 ICT디바이스학과 부교수

<주관심분야> : 정보보호, 이동통신, 통신공학