산학기술정보학회논문지
Vol. 1, No. 2, pp. 41~48. 2000

CAN을 이용한 자동화 사격장 시스템 개발

허 화 라* · 최 승 욱** · 권 구 남***

The Development of an Automatic Shooting Range System Using CAN

Hwa-Ra Hur, Sung-Yug Choi, and Gu-Nam Kwon

요 약 최근 산업현장의 모든 부분에서 자동화의 요구가 급격하게 증가되고 있다. 지금까지의 사격장 시스템은 정적인 환경을 중심으로 설계되어져 왔다. 그러나, 실제의 사격상황은 목표물이 동적이므로 다른 사격 환경 변화가 요구된다. 본 연구에서는 동적인 사격환경을 구현하기 위해 1개 사로에 4대의 타겟을 설치하여 사격순서를 순차 및 비순차로 구분하고, 사격시간을 가변적으로 조정할 수 있게 하였다. 그리고, 자동차 및 빌딩 자동화에 많이 응용되고 있는 CAN(Controller Area Network)을 이용함으로써 효율적인 통신을 가능하게 하였다. CAN은 높은 데이터 전송률과 안정성을 제공할 수 있어 다수의 ECU(Electric Control Unit)를 상호 연결하여 분산된 실시간 제어를 효율적으로 지원할 수 있다.

Abstract Recently, the requirement of automation in the every fields of the industrial places has been increased remarkably. The shooting range systems in the past have been designed on the basis of static circumstance. However, Various transformation of the shooting circumstances is required, since the target for real shooting situation not static. In this study, we have established for targets per one shooting line to realize the dynamic shooting circumstance, distributed shooting order into sequence and random ones, and accommodated the shooting time variably. And, we have done an efficient communication possible, using CAN(Controller Area Network) applied to a lot of the car and the building automation. The CAN can support real-time control efficiently by connecting multiple ECU(Electric Control Unit), since it provide the high data transfer rate and the stability of communication.

Key Words : Controller area network(CAN), Electric control unit(ECU), Pop-Up System

1. 연구 개요

현재 운영되고 있는 대부분의 사격장환경은 고정타겟에 의한 점수제 방식으로 단순한 사격 시스템을 갖추고 있다. 그러나 실제 상황은 목표물이 동적이므로 다른 사격 환경변화가 요구되어 사격환경을 Sequence Pop-Up 방식 및 Random Pop-Up 방식으로 설계하고 또한 사격허용시간을 가변적으로 조절할 수 있도록 개발한다.

본 연구에서 제안하는 자동화 사격장 Pop-Up 시스템은 사격장에서 자동으로 타겟을 움직이기 위해 기구부는 AC 모터를 사용하여 구동하고, 탄알이 타겟에 적중하였는지를 알기 위해 충격센서(Piezoelectric Shock Sensor)를 사용하여 얻어진 아날로그값을 변환하여 마이크로프로세서에 입력받아 CAN을 통하여 처리한다. CAN은 차량 내에서 센서기, 센서, 액츄에이터 등을 연결하기 위해 Bosch사에서 개발한 시리얼 통신 프로토콜이며 높은 데이터 전송률과 안정성을 제공하고 있어서 다수의 ECU(Electric Control Unit)를 상호 연결하여 분산된 실시간 제어를 효율적으로 지원할 수 있기 때문에 엔진제어 시스템, 미끄럼방지 제동시스템, 빌딩제어 시스템 등에서 사용되고 있다. 한편, 저속으로 작동되는 CAN 네트워크는 스위치 조작이나 계기 신호들을 송수신하는 데에도 사용되고 있다.[4-11]

2. 연구내용

2.1. CAN의 구성

CAN은 5 Kbps에서 1 Mbps까지 다양한 전송속도를 제공하며, 토폴로지로는 버스형(bus structure)과 스타형(star structure)을 지원한다. 링형 토폴로지는 한 스

* 송호대학 정보산업계열 **부산대학교 전자공학과
***(주)PCS 대표

본 논문은 (주)PSC(www koreashooting.com)와 공동 개발하였으며, 그 핵심내용은 CAN(Controller Area Network)을 이용한 자동화 사격장의 통합 제어시스템을 제안하였다. 이는 수입 대체효과 및 각종 산업용 통합 제어장치와 빌딩 자동화에 응용할 수 있는 통합시스템 개발을 위한 기술이다. (Tel: 033-340-1030, E-mail: hrhur@songho.ac.kr)

테이션의 고장이 전체 시스템에 영향을 주어 신뢰성이 떨어지는 반면, 버스형이나 스타형 토폴로지는 한 스테이션에서 고장이 발생하더라도 시스템의 일부는 사용할 수 있기 때문에 고장의 파급효과를 최소로 한정시킬 수 있다.

CAN의 계층 구조는 물리 계층, 트랜스퍼 계층, 오브젝트 계층의 3계층으로 나누어진다. 물리 계층은 신호들이 실제적으로 어떻게 전송되는지를 정의하는 계층으로 신호 레벨과 비트 표현방법, 전송매체 등을 정의하게 된다. 트랜스퍼 계층은 CAN 프로토콜의 핵심 부분으로 받은 메시지를 오브젝트 계층에 보내고 오브젝트 계층으로부터 전송되는 메시지를 받아들이는 일을 수행한다. 그 세부사항으로는 고장 제한(fault confinement) 기능, 에러 감지 및 시그널링, 메시지 확인, 메시지를 올바로게 수신했을 때 보내는 응답(ASK, acknowledgement) 기능, 메시지 충돌 중재 기능, 메시지 프레임화, 전송률과 타이밍 정의 등이다. 오브젝트 계층은 어떤 메시지가 전송되는지를 찾고, 트랜스퍼 계층으로부터 받은 메시지가 실제로 어떻게 사용되는지를 결정하며, 하드웨어와 관련된 응용 계층에 인터페이스를 제공하는 일을 하게 된다.[1]

2.1.1. Content-Based Addressing과 Bitwise Arbitration

CAN은 송신 스테이션과 수신 스테이션의 어드레스 대신 메시지의 내용에 따라 ID(identifier)를 부여하여 모든 메시지를 구별하고 메시지의 우선권을 정하는 내용에 의한 주소지정(content-based addressing)을 한다. 즉, 어느 한 스테이션이 메시지를 전송하기 시작하면 나머지 스테이션들은 수신상태가 되어 메시지를 받게 된다. 이때, 수신상태의 스테이션들은 수신된 메시지의 ID를 판독하여 수신된 데이터가 그들과 관련이 있는가 없는가를 확인하여 관련이 있으면 받아들이고 관련이 없으면 무시하는 메카니즘이다.

또한, 둘 이상의 스테이션이 거의 동시에 전송을 시도했을 경우, 충돌한 메시지의 ID를 한 비트씩 비교하여 수치적으로 가장 낮은 ID값을 가진 메시지(가장 높은 우선순위를 가진 메시지)만 전송을 계속하며, 나머지 메시지는 즉시 전송을 중단하게 된다. 이같은 ID에 의한 비트 비교 방식(identifier-based bitwise arbitration)은 정보와 시간의 손실이 발생하는 것을 방지하기 때문에 비파괴적 버스 액세스(non-destructive bus access)라고 불리며 파괴적 버스 액세스의 대표적인 예인 CSMA/CD(Carrier Sense Multiple Access with Collision Detection)와는 달리 네트워크의 부하가 상당히 증가하여도 효율적인 통신이 가능하다. Figure 1은 버스 중재

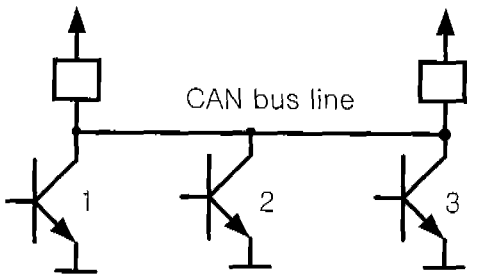

Figure 1-1. 일반적인 버스 중재 동작(1).

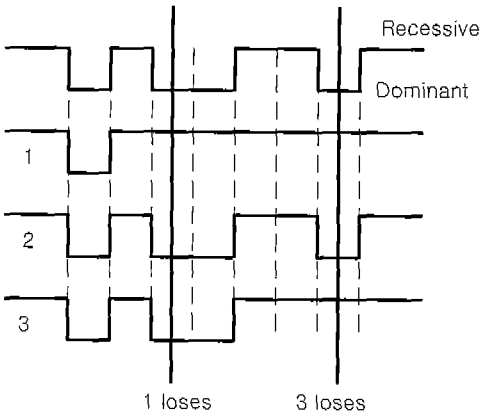

Figure 1-2. 일반적인 버스 중재 동작(2).

동작의 예를 나타낸 것으로, CAN 버스에서는 '0'을 도미넌트 비트로 사용하기 때문에 버스상에서 '0'과 '1'이 동시에 존재할 때는 '1'은 무시된다. 따라서 노드 2가 다른 노드보다 우선 순위가 높기 때문에 노드 2가 송신한 데이터 프레임이 버스상에 남고 다른 노드의 데이터 프레임의 전송은 중단된다.

2.1.2. 메시지 포맷(Message Format)

CAN에서는 4개의 다른 메시지 또는 프레임(frame) 타입이 정의되어 있다. 여기에는 데이터 프레임(data frame), 리모트 프레임(remote frame), 에러 프레임(error frame), 오버로드 프레임(overload frame) 등이 있으며 Table 1에 그 기능이 나타나 있다.

데이터 프레임은 데이터를 전송하는 프레임으로 7개의 필드(field) 형식으로 구성되어 있으며 최대 8bytes의 데이터를 전송할 수 있다. 여기서 각 비트들이 가질 수 있는 값으로 우성 (dominent)을 나타내는 'd'(0으로 표시)와 열성(recessive)을 나타내는 'r'(1로 표시) 의 두 가지가 있다. SOF(Start Of Frame)는 하나의 'd' 비트로, 메시지의 시작을 알리고 모든 스테이션을 동기화 하게 된다. 중재 필드(arbitration field)는 11비트의 ID와 1

Table 1. CAN 프로토콜 프레임 형태

형 태	기 능
Data frame	데이터 전송
Remote frame	데이터 프레임의 전송 요구
Error frame	지역적으로 감지된 에러의 전역지인 신호
Overload frame	데이터 프레임/리모트 프레임들의 앞뒤에서 메시지 간의 전송 간격 조절

버트의 RTR(Remote Transmission Request) 비트로 구성되는데, 이 영역에서는 둘 이상의 스테이션에서 동시에 메시지를 전송하고자 할 때 발생하는 메시지간의 충돌을 ID에 의한 비트 비교방식으로 해결한다.

버스 액세스 우선순위(bus access priority)는 11비트의 ID를 통해서 조정하게 되는데, ID의 값이 작은 쪽이 우선순위가 높아서 버스를 액세스할 수 있다. ID의 2진 비트 패턴이 1111111****인 경우, 즉, 7개의 MSB(Most Significant Bit)(ID10~ID4)가 모두 'r'인 경우는 사용불가능(CAN프로토콜에서 금지하고 있음)하므로 전체 사용 가능한 ID의 수는 2032개($2^{11}-2^4$)가 된다. 그러므로, CAN 프로토콜은 2032개까지의 우선순위와 다양한 메시지 정보를 전송할 수 있다. 이보다 더 많은 종류의 정보를 다루는 경우를 위하여 확장 버전에서는 29비트를 ID에 사용할 수 있도록 확장해 놓았다. 그리고, RTR 비트의 값에 따라 데이터 프레임인지 리모트 프레임인지를 결정하게 된다. 제어 필드(control field)는 6비트로 구성되는데 2개의 'd'비트는 예비 비트로 두고, 4비트는 DLC(Data Length Code)로서 데이터 필드 상의 데이터 길이를 정해주기 위한 코드이다. 데이터 필드는 8바이트까지 가능하고, SOF에서부터 데이터 필드까지의 메시지를 순환중복검사(Cycle Redundancy Check, CRC)하는 CRC필드는 15비트의 CRC시퀀스와 1비트의 CRC 딜리미터(delimiter)로 구성되어 에러유무를 체크하게 된다. ACK필드는 1비트의 ACK슬롯(slot)과 1비트의 ACK딜리미터로 구성되는데, 메시지를 올바르게 받은 수신 스테이션이 ACK필드를 받는 바로 그 순간에 ACK슬롯의 값을 'r'에서 'd'로 세팅하게 되며, 이 프레임이 버스 상에서 계속 전송된다. 마지막으로 EOF(End Of Frame)는 7비트의 'r'로 구성되고, 메시지의 끝을 알린다.

리모트 프레임(remote frame)은 데이터 프레임의 전송을 요구하는 프레임으로 6개의 필드(SOF, 중재필드, 제어 필드, CRC필드, ASK필드, EOF)로 구성된다. 데이터 프레임과 달리, 리모트 프레임의 RTR 비트는 'r' 값을 갖고 데이터 필드가 없으며, 제어 필드에서 DLC는 데이터 프레임에서의 DLC와 형식은 같으나 아무런 의미가 없다.

에러 프레임(error frame)은 두 개의 다른 필드로 이루어져 있는데, 첫 번째 필드는 여러 스테이션에서 전송된 에러 플래그(error flag)의 중첩으로 정의된다. 에러 플래그에는 능동 에러 플래그(active error flag)와 수동 에러 플래그(passive error flag)가 있는데. 능동 에러 플래그는 6개의 연속적인 'd' 비트로 구성되고, 수동 에러 플래그는 6개의 연속적인 'r' 비트로 구성된다. 두 번째 필드는 에러 딜리미터 필드로서 8개의 'r' 비트로 이루어져 있다.

오버로드 프레임(overload frame)은 오버로드 플래그, 오버로드 딜리미터의 두 개의 필드로 구성되어 있다. 이 프레임은 다음에 오는 데이터 프레임이나 리모트 프레임을 지연시키기 위해 사용된다. 오버로드 플래그의 전송을 이끌어 내는 두 가지의 오버로드 조건이 있는데, 첫 번째 조건은 수신 스테이션의 내부 상태가 다음 데이터 프레인이나 리모트 프레임의 지연을 필요로 할 때이고 두 번째는 데이터 프레임이나 리모트 프레임의 다음에 나오는 인터프레임 영역에서 'd' 비트를 감지할 경우이다.

데이터 프레임과 리모트 프레임은 3비트의 인터프레임 스페이스에 의해 앞의 프레임과 분리된다. 반면 오버로드 프레임과 에러 프레임 앞에는 인터프레임 스페이스가 나오지 않는다. 또, 여러 개의 오버로드 프레임은 인터프레임 스페이스(interframe space)에 의해 분리되지 않는다.

하나의 프레임이 만들어지면 이것을 코딩/디코딩(coding/decoding)하는 작업이 필요하게 되는데, 그 방법으로 영-비복귀(Non-Return-to-Zero, NRZ)를 사용하며, 이 때 SOF와 CRC시퀀스 사이에 같은 극성을 가지는 비트가 5개 이상 연속적으로 나타나면 전송 스테이션은 자동적으로 반대 극성의 비트를 삽입하여 보내는 비트 스터핑(bit stuffing)을 수행하고, 수신 스테이션은 그 비트를 다시 제거하는 디스터핑(destuffing)을 수행한다. 여기서, 나머지 비트 필드들(CRC 딜리미터에서 EOF)은 고정 포맷이므로 비트 스터핑을 수행하지 않는다.

2.1.3. 주소 지정 방식과 메시지 우선순위

내용에 의한 주소지정 방식은 어떤 한 스테이션에서 메시지를 보내고자 할 때, 데이터(또는 전송요청)와 ID를 함께 보내게 되는데, 이때 다른 모든 스테이션들은 수신 상태에서 대기하다가 메시지를 수신하면, ID로써 그 내용이 자신과 관련성이 있는가를 체크하여 관련성

이 있으면 받아들여 저장하고 그렇지 않으면 무시(message filtering)하는 기법이다.

내용에 의한 주소지정 방식은 현재의 시스템에 새로운 스테이션이 첨가되거나, 삭제되더라도 소프트웨어나 하드웨어 상의 재구성이 필요하지 않으므로 유연성 있는 시스템 구축을 가능케 한다. 또한, 버스를 할당함에 있어서도 ID를 사용하여 메시지 내용에 따라 기능적으로 중요한 메시지에 높은 우선순위를 부여한다. Figure 2는 내용에 의한 주소지정 방식을 사용하는 CAN 프로토콜의 작동원리를 보이고 있다.

어느 한 스테이션이 전송할 메시지가 있을 때, 먼저 버스의 상태를 살핀 후 버스가 'idle'하면 메시지를 전송하게 되고, 버스의 상태가 'busy'하면 수신모드로 들어가게 된다. 메시지를 전송하는 도중 충돌이 발생하게 되면 데이터 프레임의 중재 필드에서 ID영역을 비교하여 우선순위가 높은 메시지는 계속 전송하고, 우선순위가 낮은 메시지는 전송을 중단하고 수신모드로 들어갔다가 재전송을 시도하게 된다. 또, 수신노드에 있는 스테이션은 전송되는 메시지의 ID를 체크하여 관련이 있으면 받아들이고 관련이 없으면 무시하게 된다.

2.1.4. 에러 처리 방법

CAN은 전송 에러를 인식하고 재전송에 의해 자동적으로 에러를 수정해준다. 에러 타입에는 Table 2에서 나타낸 것 같이 비트 에러, 스터프(stuff)에러, CRC에러, 형식에러, 그리고 ACK에러의 5가지가 있다. 에러 감지 방법은 순환중복검사, 모니터링(monitering), 비트 스터핑, 메시지 프레임 체크 등이 있다. 순환 중복검사 방법은 전송시 그 전송값을 생성 다항식으로 나누고, 그 값을 메시지 뒤에 붙여서 함께 전송한다. 그러면 수신측에서는 받은 데이터를 동일한 생성 다항식으로 나누어 나머지가 '0'이면 에러가 없는 것으로 가정하는 방법이다.

모니터링 방법은 비트를 전송하는 스테이션이 동시에

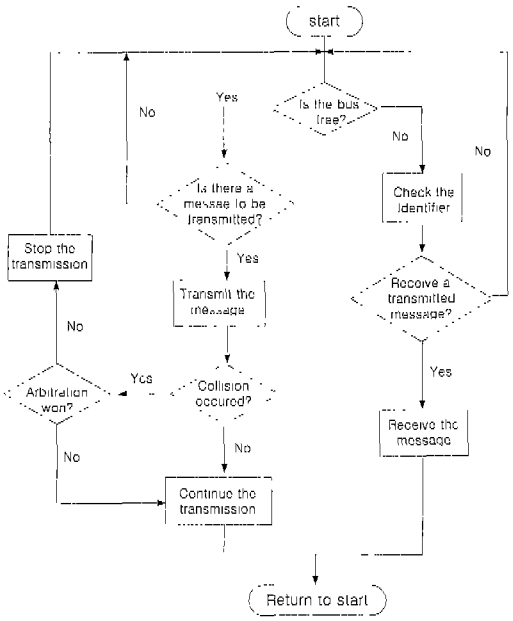

Figure 2. 내용 기반 어드레싱 및 중재 방법.

버스 레벨(bus level)을 체크하여 전송한 비트와 버스상에서 감지된 비트를 비교하는 방법이다. 이때, 스테이션에서 전송한 비트와 다른 비트가 버스 상에서 감지되면 에러 플래그를 보내게 되고 이러한 방법으로 비트 에러를 감지한다.

스터프 에러는 SOF와 CRC필드의 끝 사이에서 같은 극성을 가진 비트가 6개 이상 연속으로 나타나면 비트 스터핑이 적절히 수행되지 못한 것으로 간주하고, 스터프 에러 상태가 된다. 메시지 프레임 체크 방법은 CAN 프로토콜에 포함되어 있는 고정 포맷 비트 필드를 체크하여 메시지 프레임 형식에 어긋나는가의 여부를 알아내는 방법으로, 형식 에러를 체크한다.

에러의 발생을 알리는 방법으로 에러를 감지한 스테이션에서 에러 플래그를 전송하게 되는데, 비트 에러,

Table 2. 에러 형태

Error 구분		검출 위치	에러 프래그의 전송시점
Bit error	전송한 비트가 버스상에서 모니터된 비트와 다른 경우	Transmitter	에러를 감지한 비트의 다음 비트
Stuff error	같은 극성을 가진 비트가 6개 이상 연속으로 감지된 경우	Transmitter/Receiver	
Form error	정해진 형식의 비트 패턴이 나타나지 않은 경우	Receiver	
ACK error	ACK 슬롯이 'd'로 셋팅 되지 않은 경우	Transmitter	
CRC error	나눗셈에서 나머지가 0이 아닌 경우	Receiver	ACK Delimiter의 다음 비트

스터프 에러, 형식 에러, ACK 에러의 경우에는 에러를 감지한 비트의 바로 다음에 전송하고, CRC 에러의 경우에는 ACK딜리미터 바로 다음에 에러 플래그를 전송한다. 이 에러 플래그는 비트 스터핑 법칙이나 프레임의 고정된 형식을 깨뜨리게 되고, 이것이 다시 다른 스테이션으로 하여금 에러 플래그를 전송하게 한다. 예를 들어, 임의의 스테이션 A가 데이터 프레임을 수신하다가 에러를 감지하면 A는 6개의 'd'비트(에러 플래그)와 8개의 'r' 비트(에러 딜리미터)로 구성된 에러 프레임(능동 에러 플래그)을 전송하게 된다. 이 에러 프레임을 받은 다른 스테이션들은 6개의 'd'비트를 수신하여 에러가 발생했을 때 인식한다. 그래서 이들 스테이션들도 또 6개의 'd'비트(에러 플래그)와 8개의 'r'비트(에러 딜리미터)로 구성된 에러 프레임을 전송하게 되고, A가 전송한 에러 프레임의 에러 딜리미터 영역에 6개의 'd'비트(에러 딜리미터)로 구성된 에러 프레임을 전송하게 되고, A가 전송한 에러 프레임의 에러 딜리미터 영역에 6개의 'd'비트를 오버라이트(overwrite)하여 에러 플래그의 중첩 영역(6~12bits)을 만들게 된다.

또, CAN에는 전송 에러 카운터와 수신 에러 카운터가 있는데, 메시지가 올바로 송수신된 경우에는 에러 카운터가 감소하고 에러가 발생한 경우에는 에러 카운터가 증가한다. 만일 이 두 에러 카운터가 127 에러 포인트를 넘지 않으면 능동 에러(error active) 상태로 동작하고, 두 에러 카운터 중 어느 하나가 127 에러 포인트를 넘으면 수동 에러(error passive) 상태로 바뀌게 된다. 한편, 두 255 에러 포인트를 초과한 전송 에러는 시스템을 'bus off' 상태로 바뀌게 한다.

2.2. Pop-Up 시스템 구성

2.2.1. 전체 시스템 구성

자동화 사격장 Pop-Up 시스템은 Figure 3과 같이 4대의 엑츄에이터가 하나의 단위로 작동하고 이러한 단위들을 CAN 컨트롤러가 내장된 87C196CA를 통하여 제어하고, 4개의 단위를 하나의 PC로 모니터링 및 관리하는 시스템으로 구성되어 있다. 각각의 엑츄에이터에서는 컨트롤러로부터 명령을 받고 센서를 검사하여 타겟에 탄알의 명중 여부를 컨트롤러에 보내주게 된다. 컨트롤러에서는 사격자의 초기 입력에 따라 순차적/비순차적 및 초/중/고급에 대한 명령에 따라 다음 엑츄에이터에 대한 명령을 수행하게 된다. 이러한 데이터는 컨트롤러와 PC와의 CAN 통신을 통하여 전달되게 되고, PC에서는 C++Builder를 이용한 모니터링 프로그램으로 관리자가 시각적으로 사격상황을 인식할 수 있게 하고, 결과를 프린트하여 사격자에게 제공할 수 있도록 한다.[2]

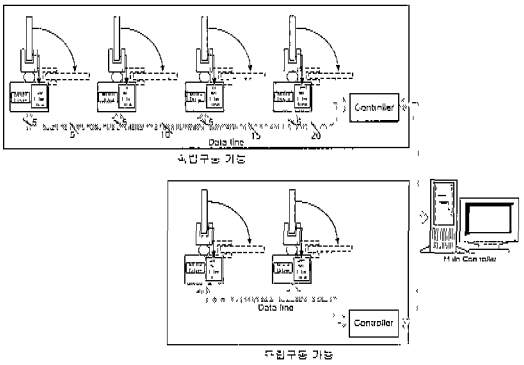

Figure 3. 전체 시스템 구성.

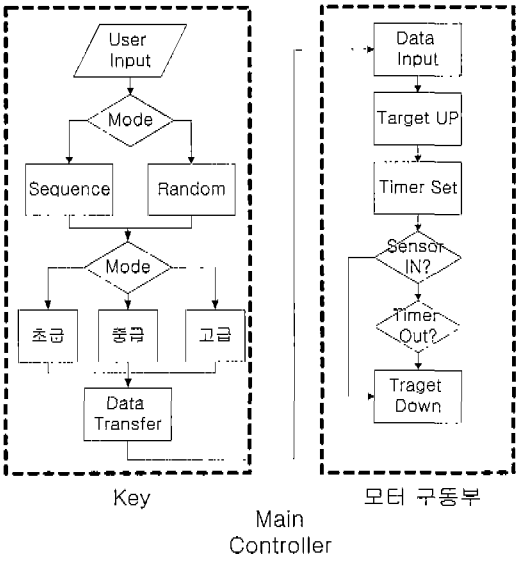

Figure 4. 시스템 흐름도.

Figure 4는 전체적인 시스템의 흐름도를 나타내고 있다. 관리자의 버튼 입력에 따라 사격 모드를 결정하고 그에 따른 신호가 컨트롤러부에서 각각의 구동부로 전달되게 된다. 구동부의 타겟에 부착되어 있는 센서 신호로부터 사격 명중여부를 컨트롤러에 전달하고, 명중되었을 때는 즉시 타겟을 넘어뜨리고, 그렇지 않을 때는 일정한 시간이 경과한 후에 넘어뜨린다. 그 결과는 PC로 보내어진다.

2.2.2. 센서 인터페이스부

압전소자는 충격이나 진동에 비례하여 전압을 발생시킨다. 위의 충격센서는 충격 에너지를 그에 상응하는 전기 신호로 바꾸는 압전 세라믹을 이용한다. 본 연구에 사용된 충격센서는 무라타사의 PKS1-4A1으로써 특징은 다음과 같다.

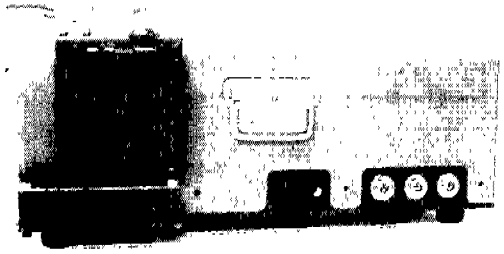

Figure 5. 모터 구동부의 외형.

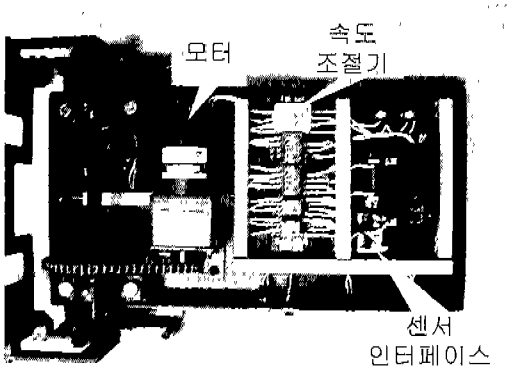

Figure 6. 모터 구동부 내부 구성.

① 작고 가볍게 설계되었다.

② 마이크로 레벨의 충격이나 진동까지도 감지할 정도의 고감도이다.

③ 바이어스 전압을 필요로 하지 않기 때문에 주변회로가 간단하다.

2.2.3. 모터구동부

본 연구에서 개발된 모터구동부의 외형과 내부구성은 Figure 5, 6과 같다.

모터 구동부는 센서의 감도를 고려하여 외부의 충격에 강인하도록 설계되었고 모터는 220V용 AC모터를 사용하였다.

2.2.4. 컨트롤러부

Figure 7은 컨트롤러의 내부 회로 구성을 나타내고 있다. 컨트롤러부는 크게 전원부(220V AC 입력, 5V DC 출력)와 CPU 모듈, 4대의 엑츄에이터 구동과 버튼 입력을 담당하는 부분으로 구성되어 있다.

CPU 모듈은 크게 마이크로 컨트롤러(MCU), 메모리, 네트워크, I/O 부분으로 구성되어 있으며, 마이크로 컨트롤러는 모듈의 모든 동작을 프로그램에 의해 총괄하고 각 부분을 통제하며, 메모리는 마이크로 컨트롤러에

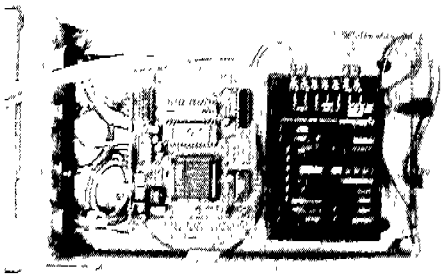

Figure 7. 컨트롤러 내부구성.

프로그램 메모리와 데이터 메모리를 공유한다. 네트워크 부분은 CAN 컨트롤러와 버스 드라이버로 구성되어 있고, I/O 부분은 키 입력 및 램프 구동을 담당한다.

MCU는 87C186CA를 사용하였다. 87C196CA는 인텔사의 MCS96 계열의 16비트 MCU로서 'CAN 2.0 specification'이 내장되어 있어 각 모듈에 가장 적합한 MCU로 선정되었다. 87C196CA는 최고 20MHz의 클럭 주파수를 사용할 수 있으며, 7개의 I/O 포트, 6채널의 AD 변환기, 1채널의 시리얼 포트, 10 채널의 이벤트 입/출력, 1채널의 CAN 포트 등이 내장되어 있다. 특히 CAN 기능은 87C196CA의 가장 특징적인 기능으로 내부 메모리 중 1F00H~1FDFH가 CAN 기능에 관련된 SFR로 할당되어 있다. 내장된 CAN 포트를 이용하여 네트워크를 통한 다른 모듈과의 정보의 교류를 가능하게 할 수 있다.[3]

메모리는 프로그램 메모리와 데이터 메모리로서 나눌 수 있다. 프로그램 메모리는 32KB ROM인 27C256를 사용하였고 데이터 메모리는 8KB NVRAM(Autostore non-volatile RAM)인 STK12C68를 사용하였다. STK12C68은 전원이 꺼지면 자동으로 데이터의 내용을 저장하는 기능이 있어 사용자가 셋팅해 놓은 값을 저장할 수 있다. 프로그램 메모리 영역은 0000H~7FFFH 이며, 데이터 메모리 영역은 F800H~FFFFH이다.

네트워크 부분은 크게 CAN 컨트롤러와 버스 드라이버로 구성되어진다. CAN 컨트롤러는 87C196CA에 내장되어 있으므로 별도의 CAN 컨트롤러는 필요하지 않고, 데이터의 송수신 및 에러의 처리 등을 담당하여 통신부분에서 MCU의 부담을 현저히 줄여준다. 버스 드라이버는 82C250을 사용했는데, TTL 신호 레벨을 CAN에 맞는 레벨로 변환하는 역할을 한다.

전체적인 컨트롤러의 배선 위치는 Figure 8과 같다. 순차적은 10 m, 15 m, 20 m, 25 m순으로 사격을 실시하며, 비순차적은 임의(Random)의 순서로 사격을 실시

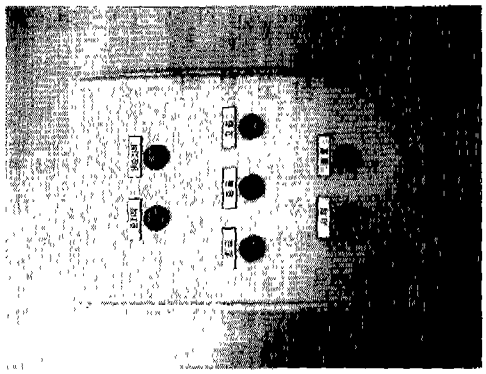

Figure 8. 컨트롤러 버튼 구성.

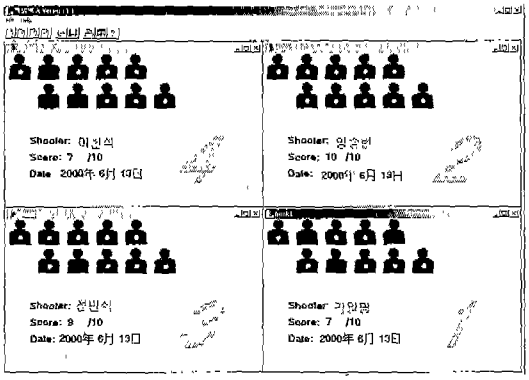

Figure 9. 모니터링 프로그램.

한다. 초/중/고급에서 사격의 난이도를 결정하며, 초급은 타겟이 올라온 후 7초, 중급은 5초, 고급은 3초동안 기다린다.

2.3. 모니터링 프로그램

관리자가 사격장 외부에서 사격과정을 인지할 수 있도록 모니터링 프로그램을 개발하였다. 본 연구에서는 C++Builder 5.0에서 개발하여 사용자 위주의 그래픽 인터페이스(GUI)를 Figure 9와 같이 구현하였으며, 사격자에게 사격결과를 제공하기 위해 출력폼을 구성하였다. 출력폼에서는 과거 사격 성적과 사격장에 대한 자료를 포함하고 있다.

2.4. 가상 사격 시나리오

최초 프로그램 실행화면은 Figure 10과 같다.

"창분할" 버튼을 누르면 Figure 11과 같은 창분할 모드가 나타난다.

컨트롤러와 통신을 초기화하기 위해서 "통신초기화" 버튼을 누른다.

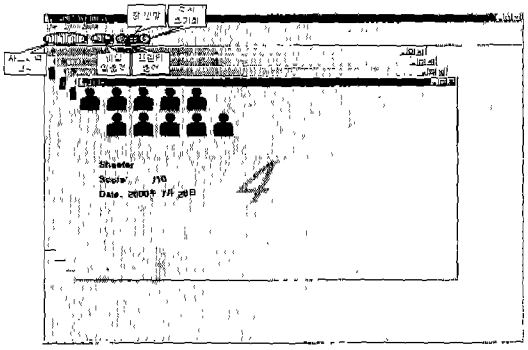

Figure 10. 초기화면.

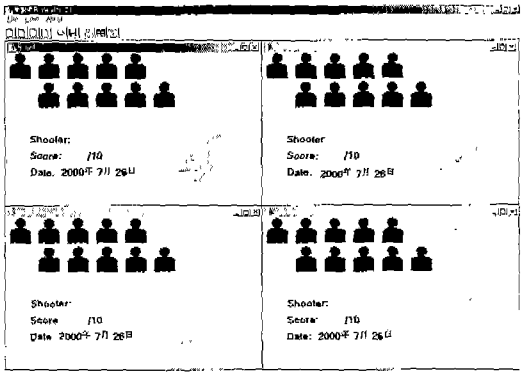

Figure 11. 창분할.

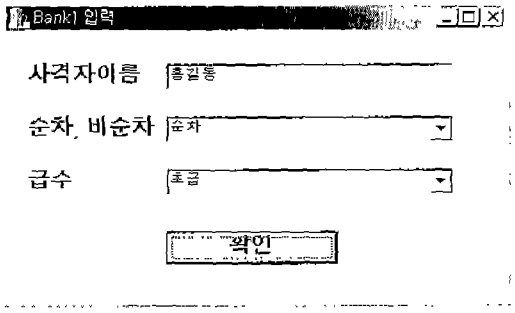

Figure 12. 신상 및 사격모드 입력.

사격자가 입장하면 사로를 선택하고 사격자의 신상 및 사격모드를 Figure 12와 같이 입력하여 확인버튼을 누른다.

사격 관리자는 사격자의 요구에 따라 컨트롤러의 버튼을 입력함과 동시에 Pop-Up 시스템이 동작을 시작한다. 10발 사격이 완료되면 자동적으로 Pop-Up 시스템이 종료되며 사격자가 퇴장하면 관리자는 사격결과를 출력하여 사격자에게 제공한다.

결과출력 방법은 사격사로를 선택하여 "프린터출력"

Figure 13. 사격자용 출력폼.

버튼을 누른다.

사격결과 출력 화면은 Figure 13과 같다.

사격결과 파일을 열고 파일입출력 버튼 중 "저장"버튼을 누른다. 사격일시 및 사격자 신상, 사격결과가 파일로 저장된다.

3. 연구 개발결과

본 연구에서 개발된 사격장 자동 타겟 Pop-Up 시스템은 고감도의 센서 선정, PC를 제외한 독립구동 가능, CAN의 도입으로 인한 통신의 유용성, 모니터링 프로그램을 통한 사용자 편의의 그래픽 인터페이스(GUI)구현 등으로 연구의 최종목표를 달성하였다. 제안한 시스템은 모니터링 프로그램 및 통신을 담당하는 PC, 타겟보드가 설치되는 엑츄에이터, 사격적중 여부를 판단하는 센서부, 타겟보드를 제어하는 제어기부로 구성되어 있다. 제어기에서는 각 엑츄에이터에 대한 제어를 담당하고 타겟보드에 있는 센서신호를 받아 적중여부를 판단하게 되고, 이를 CAN을 통하여 PC에 전달한다. PC에서는 모니터링 프로그램으로 현재 진행중인 사로에 대한 사격현황을 나타내고, 사격후 결과를 사격자에게 제공한다. 본 연구에서 제안된 시스템으로 사격수의 능력향상 및 사격에 대한 흥미유발의 효과를 기대할 수 있다. 그러나, 센서의 설치위치 선정과 복잡한 신호선으로 인한 설치시의 문제점이 야기되었다. 향후 연구과제로는 신호선을 무선화(RF)함으로써 간편한 설치로 자동화 사격라인을 구성할 수 있는 시스템 구축이 요구된다.

참 고 문 헌

[1] 최치권, 조영조, 유범재, 오상록, 윤태웅, "고속전철을 위한 CAN의 응용층 설계 및 성능 해석," 97 추계 KACC 학술회의논문집, pp. 883~886, 1997.

[2] 허화라, "CAN을 이용한 자동차용 Network 구현," 대한경영정보학회 논문지, 제2권 1호, pp. 335~354, 6월, 1998.

[3] 이장명, 주진화, 허화라, "80C196KC를 이용한 제어시스템 설계," 진영사, 1999.

[4] R. B. Gmbh, "CAN Specification 2.0 Part A&B," Stuttgart, 1991.

[5] CAN-the technical introduction, "CAN in Automation", http://www.can-cia.de/ican.htm.

[6] A. Burns and A. Wellings, "Real-Time System and Programming languages," Addison-Wesley, pp. 357~490, 1989.

[7] W. Lawrenz, "CAN System Engineering From Theory to Pactical Applications," Springer, 1997.

[8] K. Tindell and A. Burns, "Guaranteeing Message Latencies on Controller Area Network(CAN)," Proceeding 1st International CAN conference, Mainz, Germany, September, 1994.

[9] M. G. Rodd, K. Dimyati, and L. Motus, "The design and analysis of low-cost real-time fieldbus systems," IFAC Workshop Proceedings, 1997.

[10] M. Gergeleit and H. Streich, "Implementing a Distributed High-Resolution Real-Time Clock using the CAN-Bus," Proceedings of the 1st international CAN-Conference 94, Mainz, 13~14, September, 1994.

[11] J. W. Park, D. G. Roh, and J. M. Lee, "Design of CAN-based System for Distributed Control," Proceeding of ITC-CSCC, pp. 600~603, July, 2000.