오픈소스 하드웨어에서 효율적인 임베디드 소프트웨어 개발을 위한 프레임워크

(Framework for efficient development of embedded software in open source hardware)

강기욱*, 이정환*, 홍지만**

(Kiwook Kang, Jeonghwan Lee, Jiman Hong)

요 약

무선인터넷이 보급되고 IoT 기술이 발달함에 따라 여러 종류의 센서 디바이스가 발전하였다. 그리고 IoT 환경에서 사용자들의 요구를 충족하는 다양한 서비스 개발을 위해 오픈소스 하드웨어가 도입되었다. 하지만 오픈소스 하드웨어는 개발 인력의 부족으로 인해 충분히 활용되지 못하고 있다. 따라서 본 논문에서는 오픈소스 하드웨어에서 효율적으로 임베디드 소프트웨어 개발을 교육하기 위한 소프트웨어 프레임워크를 제안한다. 제안하는 프레임워크는 비주얼 프로그래밍 언어와 빠른 결과 확인을 통해 다양한 오픈소스 하드웨어에서 빠르고 직관적으로 임베디드 소프트웨어를 개발할 수 있게 한다. 또한 제안한 프레임워크를 실제 오픈소스 하드웨어 개발 환경에 구현하여 장단점을 분석하고 개선방안을 확인하였다.

■ 중심어 : 오픈소스 하드웨어 ; IoT ; 임베디드 소프트웨어 ; 프레임워크

Abstract

Various sensor devices has been developed as the wireless Internet and IoT technology are widely used. Recently, open source hardware has evolved for providing various services in IoT environments. However, in comparison to the development of the open source hardware, the development of human resources is missing. In order to solve such a phenomenon, in this paper, we propose a software framework for the embedded software development in open source hardware. The proposed framework provides a fast and intuitive development environment by using the visual programming language and providing fast feedbacks to developers. In addition, we discuss the strengths and weaknesses of the proposed scheme based on the implement on a real board.

■ keywords : Open source hardware ; IoT ; Embedded software ; Framework

Ⅰ. 서 론

최근 무선인터넷과 스마트 기기의 보급과 함께 IoT 기술이 발달하고 있다. 그에 따라 IoT 환경을 제공하기 위한 센서 네트워크 기술이 발전하였으며, 센서 네트워크의 목적에 맞는 서비스를 구현하기 위해 많은 센서 디바이스들이 개발되었다[1][2]. 사용자의 요구에 맞추어 센서 디바이스를 빠르게 개발하고, 저작권과 비용 등의 문제를 해결하기 위해 기업들은 오픈소스 하드웨어를 도입하였다[3][4].

오픈소스 하드웨어는 해당 제품을 만드는 데 필요한 모든 것(회로도, 자재명세서, 인쇄 회로 기판 도면 등)을 대중에게 공개한 전자제품이나, 하드웨어 기술 언어가 대중에게 공개된 프로그래머블 논리 소자를 의미한다. 오픈소스 하드웨어는 손쉬운 복제가 가능하고, 기능을 추가하기 위한 모듈과 참고자료 등이 많다는 특징이 있다. 따라서 오픈소스 하드웨어는 보급이 쉽고 사용이 자유로워 교육용 및 개발용으로 자주 쓰이며, 그 중 교육용으로 자주 쓰이는 종류는 라즈베리파이[5]나 아두이노[6] 등이 있다. 하지만 보급이 쉬운 것에 비하여 오픈소스 하드웨어 신규 인력의 교육은 부족한 실적이다. 특히 라즈베리파이 등 독립적으로 작동하는 보드는 PC와의 연결이 힘들고 리눅스 기반의 OS에 개발자기 익숙하지 않은 점 등의 문제가 있다. 따라서 많은 신규인력에게 보다 효율적으로 임베디드 소프트웨어 개발을 교육하기 위한 개발 프레임워크가 필요하다.

효율적인 임베디드 소프트웨어 교육을 위한 방법으로는 비주얼 프로그래밍 언어를 사용하는 방법과 빠르게 결과를 확인하는 방법이 있다. 비주얼 프로그래밍 언어[7]는 텍스트 편집이 아닌 시각적인 조작을 통해 프로그램을 짜는 방식으로, (그림 1)

* 학생회원, 숭실대학교 컴퓨터학과
** 종신회원, 숭실대학교 컴퓨터학과
이 논문은 2016년 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2016R1D1A1B01016073)

접수일자 : 2016년 08월 18일 게재확정일 : 2016년 12월 28일
수정일자 : 2016년 12월 16일
교신저자 : 홍지만 e-mail : jiman@ssu.ac.kr

과 같은 타임라인 방식과 (그림 2)와 같은 스테이트 머신 방식이 있다. 이러한 비주얼 프로그래밍 언어는 절차적 프로그래밍 언어를 배우지 않아도 눈으로 보고 프로그램을 제작할 수 있도록 도와준다. 빠른 결과를 보여주는 방법[8]은 프로그래밍의 결과를 곧바로 보여주는 것을 의미한다. 아두이노 등의 자체 OS가 없는 하드웨어는 PC에서 프로그램을 제작하면 곧바로 프로그램이 전송되어 실행된다. 하지만 라즈베리파이 등의 자체 OS가 있는 하드웨어는 직접 연결하거나 SSH 등을 통해야하는 불편이 있고 컴파일 즉시 실행되지도 않는다. 이러한 문제는 개발 과정의 직관성을 해치고 초보자에게 불편을 야기하므로 개발과정을 간소화 하고 쉽게 반응을 볼 수 있도록 바꿀 필요가 있다.

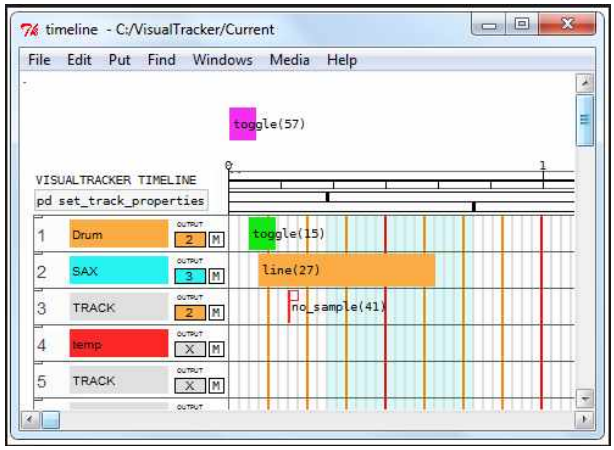
본 논문에서는 다양한 오픈소스 하드웨어에서 임베디드 소프트웨어 개발을 위한 소프트웨어 프레임워크를 제안한다. 제안하는 프레임워크는 PC에서 비주얼 프로그래밍 언어를 통하여 프로그램을 제작한 다음, 원하는 오픈소스 하드웨어에서 실행할 수 있도록 하여 손쉽게 익히고 사용 가능한 임베디드 소프트웨어 개발 환경을 제공한다. 또한 제안한 프레임워크를 라즈베리파이에 구현하여 장단점을 분석하였다.

논문의 구성은 다음과 같다. 2장에서는 관련된 연구를 소개하고 3장에서는 제안하는 방법을 설명한다. 4장에서 기존 방법과 비교를 한 후, 5장에서 개선점을 확인한다. 6장에서는 결론을 맺고 향후 연구 계획을 소개한다.

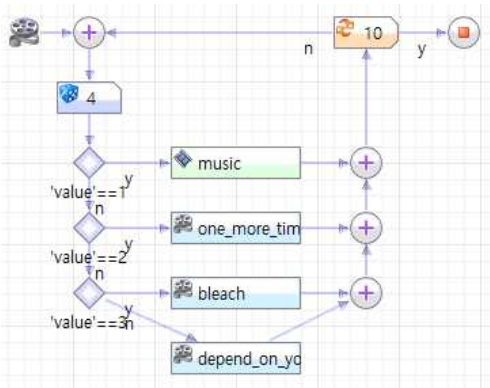
Ⅱ. 관련연구

Pearce [3]는 연구 장비로서 오픈소스 하드웨어의 장점에 대해 언급하였다. Pearce는 특히 오픈소스 3D 프린터의 가능성을 강조하였으며 이에 대한 연구는 국내외에서 활발히 진행되고 있다[9]. 그 외에도 오픈소스 하드웨어는 일상생활에서의 IoT 기기[10] 또는 이들을 총괄하는 통합 컨트롤러 개발[11] 등 다양한 용도로 연구 개발되고 있다.

비주얼 프로그래밍 언어는 시각적인 조작으로 프로그래밍의 난이도를 낮추기 위해 연구되었다[12]. 페트리 네트를 이용한 그래픽 프로그래밍을 하는 방법이 사용되었으며, 순차 제어용 프로그래밍에 활용이 되거나, 디지털 신호처리, 병렬프로그래밍에 사용되는 등 다양한 분야에서 활용되고 있다[13][14][15][16]. 특히 최근에는 교육용으로 활용하는 연구가 활발히 진행되고 있다. 류충규는 MIT에서 개발된 스크래치를 사용해 초등학생 프로그래밍 교육에 대해 연구하였다[17][18]. 임화경은 Kodu를 사용한 3D 게임 프로그래밍 교육에 대해 연구하였다[19][20]. Roboid Studio는 비주얼 프로그래밍 언어를 도입하여 펠리카노이드 로봇을 조작하는 프레임워크를 제안하였다[22].



(그림 1) Visual Code 예시(Time line 스타일)



(그림 2) Visual Code 예시
(State machine 스타일, RoboidStudio)

Ⅲ. 제안하는 프레임워크

1. 필요성

오픈소스 하드웨어를 이용한 개발을 할 때 개선이 필요한 사항이 있다. 익숙하지 않은 상황에서의 개발, 다른 분야의 개발에 비해 많은 교육의 필요성, 개발 과정의 복잡화 등이 그것이다. 본 논문이 제안하는 프레임워크는 이를 구현하고자 한다.

오픈소스 하드웨어를 처음 접하는 대부분의 신입 개발자와 학생들은 윈도우 이외의 환경에 익숙하지 않으므로, 본 논문이 제안하는 프레임워크는은 보다 익숙한 환경인 윈도우에서 임베디드 소프트웨어를 개발할 수 있도록 설계한다. 이를 위해서 윈도우가 설치된 호스트 PC에서 구동하는 통합 개발 환경(IDE)를 제공하고, 이를 타겟 보드에 전송할 수 있는 중계 프로그램을 제공한다. 이 방법은 호스트 PC에서 개발한 프로그램이 타겟 보드로 전송되어 설치므로 서로 다른 OS가 설치되어 있더라

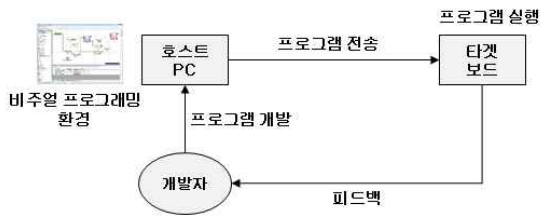
도 개발이 용이하다.

또한 신입 개발자들은 타겟 보드를 이용한 임베디드 프로그래밍에 익숙하지 않다. 제안하는 프레임워크의 IDE는 그래픽 기반의 비주얼 프로그래밍 언어를 제공하여, 유한상태머신 형태의 비주얼 프로그래밍 언어를 제공한다. 그것에서 상태머신에서 상태의 집합은 각 모듈이나 핀을 나타내고, 각 상태 하나는 모듈이나 핀의 한 상태를 의미한다. 그리고 각 상태의 연결은 모듈 혹은 핀의 입력과 출력의 변화이고, 이를 통해서 모듈은 상태가 변화한다. 비주얼 프로그래밍 언어는 원시 언어보다 이해하기 쉽고 프로그램의 구동 과정을 쉽게 파악할 수 있으며, 마우스 드래그 및 간단한 수식 입력을 통해 프로그램을 제작할 수 있으므로 교육 및 개발 과정을 단축할 수 있다.

마지막으로 제안하는 프레임워크는 개발과정을 간소화하는 기능을 제공한다. 라즈베리파이 등의 오픈소스 하드웨어는 보드 자체가 자신의 OS를 가지므로 이를 직접 PC에 연결하여 개발하기가 힘들다. 실제 개발을 하고 곧바로 결과를 확인할 수 있는 PC 환경과 달리, 임베디드 소프트웨어는 개발 후 결과물을 옮기고 실행하여 모듈들의 변화를 확인하는데 걸리는 과정이 복잡하다. 이를 위해 OS에서 SSH 등의 기능을 제공하지만 이는 신입 개발자가 익숙해지고 사용하는데 오래 걸린다. 제안하는 프레임워크는 PC와 타겟 보드를 연결하여 개발 후 전송 및 실행하는 과정을 간소화 하는 모듈을 제공한다.

2. 설계

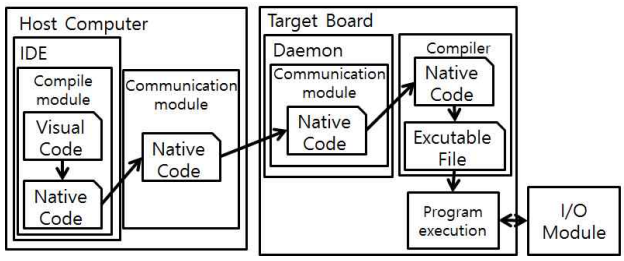
제안하는 프레임워크는 호스트 PC와 타겟 보드로 구성된다. 호스트 PC에서는 비주얼 프로그래밍 언어를 사용하는 통합 개발 환경(IDE)을 제공하며 작성된 프로그램은 타겟 보드로 전송되어 실행된다. 일반적인 개발 환경에서는 개발자가 타겟 보드에 연결하여 원시 언어로 직접 프로그램을 작성해야 하므로 임베디드 소프트웨어 개발 교육이 쉽지 않은 반면, 제안하는 프레임워크에서는 비주얼 언어를 사용하여 간편한 조작으로 프로그램을 쉽게 작성할 수 있고, 타겟 보드에서 프로그램을 실행한 후 빠르게 결과를 확인하고 오류를 수정할 수 있다(그림 3).



(그림 3) 제안하는 프레임워크 구성

나. 소프트웨어 구조

제안하는 프레임워크의 소프트웨어 구조는 그림 4와 같다. 호스트 PC에서는 IDE와 통신 모듈이 탑재되며, 타겟 보드에는 통신 모듈, 컴파일러, I/O모듈이 탑재된다.



(그림 4) 제안하는 프레임워크의 소프트웨어 구조

나. 호스트 PC

호스트 PC는 비주얼 프로그래밍 언어를 이용하여 개발을 하고 완성된 코드를 타겟 보드로 전송하는 PC이다. 타겟 보드와는 OS나 성능이 동일할 필요가 없으며, 시리얼 포트나 무선 인터넷 등으로 통신할 수 있어야 한다.

호스트 PC는 먼저 그래픽 기반의 개발 모듈을 이용하여 프로그램의 코드를 생성한다. 비주얼 언어를 이용하여 생성되는 비주얼 코드(Visual code)는 그림 2와 같이 상태 머신 형태로 표현이 되며, 각 상태는 드래그 앤 드롭 등으로 생성한다. 그리고 파일로 저장될 때는 각 상태와 변수, 관계들이 XML 형태로 저장된다.

번역 모듈에서는 개발모듈에서 완성된 비주얼 코드가 C언어 등 타겟 보드에서 컴파일하고 받아들일 수 있는 소스 코드(Source Code)로 번역된다. 소스 코드는 대부분의 임베디드 보드에서 컴파일 가능한 C, C++나 플랫폼에 상관없는 JAVA 등의 원시 소스 코드로 번역한다. 번역이 끝난 후 필요에 따라 전송 속도를 위해서 변수 명을 바꾸는 등의 처리를 한 후 통신을 통해 전송될 준비를 한다.

통신 모듈에서는 완성된 소스 코드를 타겟 보드로 전송한다. 호스트 PC는 타겟 보드에게 소스 코드를 전송한다.

다. 타겟 보드

타겟 보드는 개발한 프로그램이 올라가고 실제로 동작하게 될 임베디드 보드이다. CPU 등 컴퓨터의 필수 부품이 포함되어 있을 뿐 아니라 다양한 I/O 디바이스 및 센서와 연결되어 있어 IoT 환경에서 다양한 역할을 수행한다.

타겟 보드는 데몬을 실행하고 프로그램을 실행하는 역할을 한다. 데몬은 개발보드에서 다운로드, 컴파일, 프로그램 실행을

담당한다. 먼저 통신 모듈에서 호스트로부터 온 소스 코드를 읽은 후, 특정 경로에 소스 코드를 저장한다. 이후 데몬은 컴파일러를 실행하여 해당 소스 코드를 컴파일하고, 만들어진 실행파일을 실행하여 I/O 모듈을 작동하게 한다.

데몬은 이 외에도 I/O 모듈의 충돌을 막고 개발의 편의를 위하여 작동중인 프로그램을 하나만 작동하도록 유지한다. 또한 디버깅을 위한 로그를 유지하고, 타겟 보드가 재부팅 되었을 때 이전의 프로그램을 곧바로 작동하도록 한다.

3. 구현

본 절에서는 제안하는 프레임워크의 목표를 위한 기능을 어떻게 구현하는지 설명한다. 먼저 전체 프레임워크의 구조를 설명한 다음, 각 구성 하드웨어의 기능을 설명한다.

가. 사양

제안하는 프레임워크는 타겟 보드와 호스트 PC에서 구현된다. 타겟 보드와 호스트 PC의 사양은 표 1과 같다.

표 1. 구현하는 프레임워크의 테스트 환경

항목		사양	기타
PC	CPU	Intel i5-4690	
	RAM	8 GB	
	OS	Windows 10 Home	
타겟 보드	보드	raspberry Pi 2 1GB	핀 40개
	시리얼	PL 2303	
	모터	Micro servo SG90	

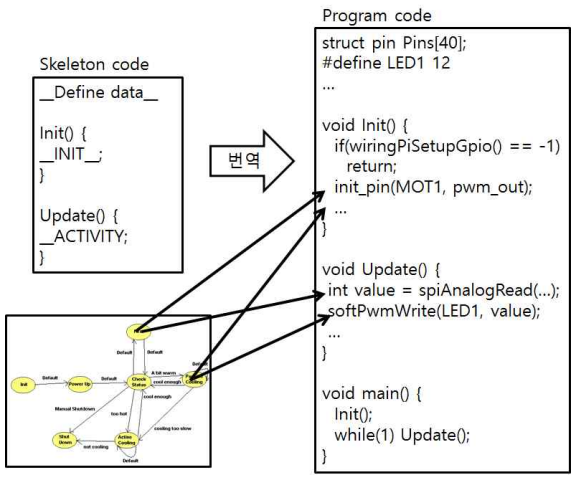
나. 호스트 PC

먼저 비주얼 프로그래밍을 위한 IDE를 구현하였다. 비주얼 프로그래밍 언어의 구현 방식은 타임라인 방식과 스테이트 머신 방식 중, 많은 개발자에게 보다 익숙한 방식인 스테이트 머신 방식을 선택하였다. 제안하는 프레임워크의 IDE는 많은 사람들이 사용하고, 내용을 수정할 수 있는 오픈 소스 IDE인 이클립스를 기반으로 프로그래밍 하였다. 또한 기존에 펠리컨 로봇을 이용하여 비주얼 프로그래밍을 가능하도록 만든 Roboid Studio 역시 이클립스를 수정하여 만들었으므로, 이를 참고하였다.

IDE에서 다루는 비주얼 프로그래밍 언어에서, 입출력은 각각 하나의 I/O 모듈을 사용한다. 타겟 보드인 라즈베리파이는 전용 모듈이 존재하지 않고 핀을 기반으로 I/O 모듈을 조종하므로, 실제로 사용할 수 있는 모듈은 무수히 많다. 하지만 간단한 구현을 위하여 각각 디지털 입력, 디지털 출력, 아날로그 입력, 아날로그 출력에 대응하는 버튼, LED, 가변 저항, Servo 모터에 대한 모듈만 사용 가능하도록 제한하였다.

또한 사용자 입력을 이용하여 각 변수의 상태를 변경하고 이를 출력한다. 각 상태는 변수의 할당, 변화, 판단 등에서 사용되며, 모듈의 입력 혹은 상태의 결과나 프로그램의 흐름에 따라서 다른 상태로 이전되거나 모듈로 출력을 한다. 이 상태의 이전이나 입출력과 상태의 관계는 상태 머신에서 상태 간의 화살표로 표시되며, 이는 그림 2와 같은 모습이다.

번역 모듈은 스테이트 머신을 C언어로 번역한다. 번역 모듈은 우선 스테이트 머신에 논리적 오류가 있는지 확인하고, 오류가 없다면 XML 형태로 저장한다. 스테이트 머신의 입출력과 상태, 그리고 상태의 이전은 위치와 그 연결이 저장된다. 그 다음, 저장된 파일에서 위치, 주석, 색깔 등 필요 없는 부분을 제외하고 C언어의 절차적 프로그램으로 필요한 부분만 잘라낸다. 마지막으로 간단한 내용이 담겨있는 템플릿인 Skeleton code를 라즈베리파이에 맞도록 수정한 다음, 잘라낸 코드를 수정된 스켈레톤 코드의 필요한 부분에 번역하여 추가한다. 번역이 완료되면 IDE는 통신 모듈을 실행한다(그림 5).

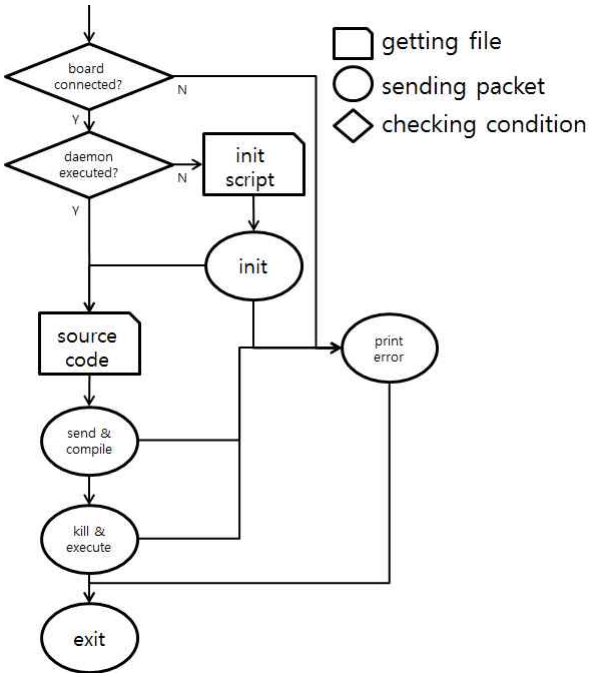


(그림 5) 번역 모듈의 번역 과정

통신 모듈은 호스트 PC와 타겟 보드를 연결해주는 미들웨어이다. 통신 모듈은 타겟 보드를 검색하고, 프레임워크의 작동을 위한 초기화를 하고, 번역된 코드를 전송한다. 통신 모듈은 이를 위해 초기화를 위한 코드를 포함하고 있다. 통신 모듈이 작동하는 전체 과정은 (그림 6)과 같다.

통신 모듈은 우선 코드 전송을 위한 장치를 검색한다. 검색을 위한 선행 조건으로 호스트 PC는 타겟 보드와 시리얼로 연결되어 있고, 해당 시리얼이 디바이스 드라이버가 필요하다면 먼저 설치되어 있어야한다. 제안하는 프레임워크는 PL 2302로 테스트 했으므로 드라이버를 먼저 설치했다. 준비가 완료되면 윈도

우 API를 이용하여 시리얼 장치를 검색한다. 연결된 장치 중 “COM”으로 시작하는 모든 장치를 검사한다. 초기화 확인 패킷을 보내서 초기화가 되었다면 확인 패킷이 돌아오고, 아니라면 반응에 따라 라즈베리 파이인지 아닌지 판정한다. 확인 패킷은 라즈베리파이의 기본 로그인 네임과 같이 설정하여, 라즈베리파이라면 패스워드를 입력하라고 하거나 쉘 오류가 나고, 라즈베리파이가 아니라면 정상적인 반응이 오지 않는다. 통신 모듈은 라즈베리파이를 찾을 때 까지 확인 패킷을 보내는 과정을 반복한다.



(그림 6) 통신 모듈의 전체 흐름도

통신 모듈이 라즈베리 파이를 찾았고, 초기화 되지 않았다면 통신 모듈은 초기화 과정을 실행한다. 제안하는 프레임워크의 타겟 보드인 라즈베리파이의 기본설정은 시리얼 연결 시 쉘이 작동하는 것이다. 따라서 통신 모듈은 필요한 파일을 쉘을 통해서 보내고, 다음 과정을 진행한다. 초기화를 위해 필요한 파일은 파일 전송을 위해 작성된 프로그램, 타겟 보드 위에서 작동할 데몬 프로그램 코드, 설정을 변경할 쉘 스크립트, 파일을 복사하고 컴파일 할 쉘 스크립트 등이 있다. 통신 모듈은 확인 패킷의 결과에 따라서, 시리얼로 연결된 타겟 보드에서 실행되고 있는 쉘 스크립트에 로그인을 하고 홈 디렉터리로 진입한다. 먼저 파일 전송을 위한 간단한 프로그램 코드를 vi 에디터를 통해서 전송하고, 컴파일 하여 실행한다. 해당 프로그램은 입력을 곧바로 파일로 쓰는 방식으로, vi 등의 파일 에디터보다 더 직관적이고 빠르게 수행되도록 제작한다. 파일이 전송되면 컴파일 후 실행하여 다른 파일을 전송한다. 이후 전송된 쉘 스크립트를 실행하여 설정 변경 스크립트 등을 실행하고 데몬 프로그램을 설치한 다음, 재부팅을 한다. 설정 스크립트가 하는일은 SPI를 사용 가능하도록 설정하는 것, 시리얼에서 자동으로 쉘을 켜지 않도록 적용하는 것, 타겟 보드가 부팅시 데몬 프로그램이 자동으로 실행되도록 설정하는 것 3가지가 있다. 모든 설정을 마친 후 재부팅을 하면 자동으로 데몬 프로그램이 실행된다.

초기화가 된 장치를 확인하거나, 장치의 초기화가 끝나면 통신 모듈은 타겟 모듈에 프로그램 코드를 보내고 실행하라는 패킷을 보낸다. 그리고 결과물을 IDE로 보내서 컴파일 에러, 실행 에러 등을 알려준다.

초기화 후 다시 설정을 변경하거나 시리얼을 이용하기 위해서는 역 초기화 과정이 필요하다. 초기화 과정의 반대로 SPI 등의 장치 설정을 원래대로 되돌리고, 시리얼 포트를 다시 쉘과 연결한다. 그리고 데몬 프로그램이 자동으로 실행되지 않도록 시작프로그램을 설정한다. 모든 과정이 끝나면 타겟 보드를 재부팅해서 다시 시리얼로 쉘 작업을 할 수 있도록 한다. 역초기화 과정은 개발 과정 오류에 대응하거나, 개발 후 보드를 다른 용도로 사용하기 위해서 사용한다.

그 외에 통신 모듈은 디버깅이나 특정 상황에 대비하여 한 패킷에 모든 기능을 하는 것이 아닌, 타겟 보드에서의 프로그램 종료, 실행, 파일전송, 컴파일 등의 각 기능에 관한 패킷을 따로 제작한다. 또한 이를 활용하기 위해 각 패킷을 임의로 보낼 수 있는 커맨드 모드를 따로 구현한다.

다. 타겟 보드

타겟 보드는 데몬 프로그램을 통해서 호스트 PC의 통신 모듈과 통신하여 프로그램을 실행한다. 타겟 보드의 역할은 프로그램 코드 수신, 컴파일, 프로그램 실행 및 관리, 초기화 패킷에 반응 등이 있다.

데몬 프로그램은 우선 확인 패킷에 답변을 보내야한다. 이 패킷을 이용하여 통신 모듈이 디바이스가 사용 가능한지 확인하므로, 초기화 되지 않은 타겟 보드나, 라즈베리파이가 아닌 다른 연결된 디바이스와 구분되는 메시지를 설정하여 호스트 PC로 보낸다.

이후 타겟 보드는 소스 코드를 수신한다. 먼저 호스트 PC는 파일을 보낸다는 패킷에 파일 길이를 포함하여 송신한다. 타겟 보드는 파일 길이를 패킷 길이로 나누어, 몇 개의 패킷이 올지 예측한다. 이후 호스트 PC에서 보내는 패킷은 패킷 순서와 소스 코드의 일부를 포함한다. 타겟 보드는 이 패킷을 받아서 순서가 맞는지 확인하고, 지정된 파일에 패킷 내용인 소스 코드를 쓴다. 지정된 길이까지 입력이 완료되면 호스트 PC는 전송 종료 패킷을 보내고, 타겟 보드는 이를 확인했다는 패킷을 돌려준다. 과정 중에 순서가 잘못되었다면 타겟 보드는 오류 코드와

정상적인 패킷 순서를 포함한 패킷을 돌려주고, 그 패킷부터 다시 전송을 시작한다. 전송이 완료되면 타겟 보드의 데몬 프로그램은 파일을 닫고 컴파일 준비를 한다.

파일이 전송 된 다음은 호스트 PC는 컴파일 패킷을 보낸다. 타겟 보드는 전송이 완료된 파일을 컴파일 한 후, 그 결과를 호스트 PC에 답변한다.

이후 호스트PC는 타겟 보드에 프로그램 실행 패킷을 보낸다. 이미 이전에 데몬 프로그램이 실행한 프로그램이 있다면, 타겟 보드는 해당 프로그램을 종료시킨다. 그 다음 완성된 프로그램을 실행하고 나중에 종료하기 위해 해당 pid를 저장한다. 타겟 보드인 라즈베리파이의 특성상, 외부 입출력인 GPIO를 이용하기 위해 슈퍼유저 권한으로 실행한다. 실행 파일이 없거나 SPI 등 옵션이 초기화 되지 않아서 오류가 생긴다면 해당 결과를 호스트 PC에 알려줘서 다시 초기화를 하도록 한다.

표 2는 타겟 보드를 초기화하고, 파일을 전송 및 실행하는 과정에서 시리얼 통신에 사용되는 패킷의 종류이다. 모든 패킷은 호스트 PC에서 먼저 보내면 타겟 보드에서 답변하는 형식이다. 개발의 편의와 오류 메시지의 분류, 그리고 추후 발전을 위하여 하나의 메시지로 모든 과정을 처리하지 않고, 기능별로 나누어 메시지를 보낸다.

표 2. 시리얼 통신에 사용하는 패킷 목록

이름	기능	기타
INIT	초기화	
UNINIT	역 초기화	
COMP	소스 코드 컴파일	
EXEC	프로그램 실행	
KILL	프로그램 종료	
SNDS	파일 전송 시작	파일 길이 포함
SNDN	파일 전송	패킷 순서 포함
SNDF	파일 전송 종료	
CHECK	초기화 확인	

Ⅳ. 기존 방식과 장단점 비교

본 장에서는 제안하는 프레임워크와 다른 방식을 비교하여 장단점을 분석해본다. 기존에 사용하는 방식은 타겟 보드에서 개발하거나 인터넷 연결 후 SSH에서 개발하는 2가지 방법이 있다. 각 방식과 제안한 프레임워크를 컴퓨터학과 대학생 3학년 중 임베디드 초보 개발자와, 동일 학과 대학원생 중 리눅스에 익숙한 숙련된 개발자 각 5명의 시점에서 편의성과 개발시간을 비교할 것이다.

초보 개발자에게 있어서 편의성은 제안하는 프레임워크가 더 좋다. SSH에 연결하거나 직접 개발하는 것은 초기 설정이 필요하고, 리눅스 환경과 임베디드 프로그래밍에 익숙해질 필요가 있다. 하지만 제안하는 프레임워크는 자동으로 초기 설정을 해주고 직관적인 개발 과정을 가지므로 편의성을 가진다. 숙련된 개발자의 경우 제안하는 프레임워크를 쉽게 다룰 수 있지만, 초기 설정이나 임베디드 프로그래밍이 익숙할 것이다. 오히려 사용가능한 기능이 제한되므로 더 불편할 가능성이 있다.

시간의 경우 개발 시간, 초기 설정 시간, 전송시간으로 나누었다. 대부분의 사람들이 TUI보다는 GUI에 익숙하며[23], 초보 개발자는 임베디드 프로그래밍에 대한 이해를 한 후 개발을 시작하는 것 보다 비주얼 프로그래밍 언어를 이용하는 것이 개발 시간은 더 적을 것이다. 피험자를 상대로 개발을 진행해본 결과, 일반적인 프로그래밍 언어를 이용할 때는 개발시간의 차이가 컸지만, 비주얼 프로그래밍 언어를 이용할 때는 툴에 익숙해지면 피험자들의 개발시간 차이는 적었다.

초기 설정 시간은 ssh나 ftp등을 설치하고 spi 등의 I/O 모듈을 위한 설정을 하는 시간이다. 시리얼 연결을 하는 경우는 초기 설정 시간은 약 3분 정도이고, 네트워크 연결을 통해 사용할 경우 무선 네트워크와 ssh 설치 등을 포함하여 약 5분이 걸린다. 하지만 제안한 프레임워크는 모든 설정을 미리 지정해놓고, 네트워크 설정을 필요로 하지 않으므로 약 30초 정도의 시간이 걸린다.

전송 시간은 파일을 타겟보드로 전송하는 시간이다. 제안한 방법을 제외한 다른 방법들은 5초 이하의 짧은 시간이 걸린다. 제안한 방법은 시리얼 통신의 오류를 해결하기 위해, 1kb당 1초씩 추가 시간을 필요로 하므로 걸리므로 전송하려는 크기가 커질수록 시간이 더 걸린다. 소스코드 자체가 커지지 않는다면 크지 않은 차이이다.

전체 과정에서 제안한 프레임워크는 보다 초보 개발자에게 개발에 있어 편의성과 직관성을 제공하고, 적은 시간을 소모하도록 함을 확인하였다. 다만 숙련된 개발자는 기존 개발방식과 다른 개발방식과 부족한 기능으로 인해 불편함을 느낄 수 있다.

Ⅴ . 개선점

다른 방식과 달리 제안하는 프레임워크는 파일 전송이나 초기화 과정에서 생기는 오버헤드가 존재한다. 따라서 오버헤드나 발견된 문제점을 해결하면 보다 개선된 성능을 낼 수 있을 것이다. 본 절은 제안된 프레임워크의 개발 도중 알려진 문제점과 해결 방법을 제시한다.

먼저 프로그램 코드의 전송 속도가 느리다는 점이 있다. 스켈레톤 코드의 형식을 맞추기 위해서는 프로그램 코드는 충분히 큰 크기를 가지고, 큰 파일을 옮길수록 속도는 느려진다. 이러한 문재를 해결하는 방법을 프로그램 코드를 수정하거나 압축하는 것이다. 프로그램 코드는 이미 완성된 코드이고 사용자가 그 코

드의 변형에 영향을 받지 않으므로, 해당 코드를 압축 파일로 압축하여 보내거나, 혹은 변수명 등을 짧게 고쳐서 코드 자체의 크기를 줄인다면 코드 크기가 작아지고 전송 속도도 빨라질 것이다.

다른 문제점으로 초기화 과정의 오류처리가 부실하다는 점이 있다. 개발 과정중 라즈베리파이에서 일어날 수 있는 오류에 대해 조사가 부족하여 오류에 대한 대비가 부족했고, 라즈베리파이 자체적인 문제점으로 부팅 도중 부팅보다 타이머 이벤트가 먼저 일어나서 멈추는 문제점도 있다. 충분한 조사와 라즈베리파이의 타이머 이벤트에 대한 이해를 통해 초기화 과정의 오류를 줄일 수 있다.

데이터 전송 중 생기는 오류에 대해서도 취약하다. 시리얼 전송 자체는 데이터 손실이나 변형에 대한 복구 루틴이 없다. 따라서 호스트 PC의 통신 모듈에서 데이터가 손실되거나 유실될 경우에 대한 오류 처리 체크섬을 데이터에 포함시키고, 손실됐을 때의 처리를 추가한다면 이러한 오류에 대해서 대처할 수 있다.

Ⅵ. 결론 및 향후 연구

본 논문에서는 비주얼 프로그래밍 언어를 사용하여 일반 PC에서 오픈소스 하드웨어의 임베디드 보드 프로그래밍을 할 수 있는 프레임워크를 제안하하고 실제로 구현하였다. 본 논문의 방법을 통하여 개발을 할 경우 일반적인 프로그래밍 언어를 사용할 때 보다 이해하기 쉽고 빠르게 개발할 수 있으며, 또한 개발 과정 또는 결과를 곧바로 타겟 보드에서 실행할 수 있기 때문에 빠르게 결과를 확인하고 디버깅 및 수정을 할 수 있다.

앞으로는 구현된 프레임워크의 단점을 보완하고 작동 중 일어나는 문제점을 해결하는 방법을 찾고 적용해 볼 예정이다. 또한 교육용 개발 프레임워크라는 개발 취지에 맞추어 다른 입출력 모듈과 다른 타겟 보드를 프레임워크에 적용하여, 여러 종류의 모듈과 타겟 보드에 대한 개발을 가능하게 만들 예정이다.

References

[1] Qian Zhu, Ruicong Wang, Qi Chen, Yan Liu and Weijun Qin. “IOT Gateway: Bridging Wireless Sensor Networks into Internet of Things”, IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2010.12, pp. 347-352.

[2] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic and Marimuthu Palaniswami. “Internet of Things (IoT): A vision, architectural elements, and future directions”, Future Generation Computer Systems, vol. 29, no. 7, 2013.9, pp. 1645-1660.

[3] Joshua M. Pearce. “Building Research Equipment with Free, Open-Source Hardware”, Science, vol. 337, no. 6100, 2012.9. pp. 1303-1304.

[4] Clive Thompson. “Build It. Share It. Profit. Can Open Source Hardware Work?”, Wired Magazine, vol. 16, no. 11, 2008.10, pp. 1-7.

[5] https://www.raspberrypi.org/

[6] https://www.arduino.cc/

[7] Brad A. Myers. “Taxonomies of visual programming and program visualization”, Journal of Visual Languages and Computing, vol. 1, no. 1, 1990.3, pp. 97-123.

[8] James Grenning. “Applying test driven development to embedded software”, IEEE Instrumentation and Measurement Magazine, vol. 10, no. 6, 2007.12, pp. 20-25.

[9] 김성수. “오픈소스 하드웨어 기술 관점에서의 3D 프린팅”, 건축시공 제 15권 제 4호(통권 70호), 2015.12, pp. 32-41.

[10] 조봉언, 박영상, 서숙길, 김진걸, 이영삼. “오픈소스 하드웨어를 이용한 침상머리각도 측정 시스템의 래피드 프로토타이핑”, 제어로봇시스템학회 논문지 제 21권 제 11호, 2015.11, pp. 1038-1043.

[11] 이세훈, 김주봉, 고희창. “오픈소스 기반의 IoT 통합 컨트롤러 설계”, 2016년 한국컴퓨터정보학회 동계학술대회 논문집 제 24권 제 1호, 2016.1, pp. 15-18.

[12] Brad A. Myers. “Visual Programming, Programming by Example, and Proram Visualization: A Taxonomy.”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, vol. 17, no. 4, 1986.4, pp. 59-66.

[13] 이광형, 김창범, 이건명, 장형규, 김형신. “페트리 네트를 이용한 그래픽 프로그래밍”, 한국정보과학회 1990년도 봄 학술발표논문집 제 17권 제 1호, 1990.4, pp. 57-60.

[14] 국금환. “그래픽 조직 언어를 이용한 순차 제어용 프로그래밍 시스템 개발”, 한국정밀공학회지 13(4), 1996.4, pp. 24-33.

[15] 박진호, 정훈, 홍선기. “그래픽언어를 이용한 DSP 제어 기법 연구”, 대한전기학회 학술대회 논문집 , 2010.7, pp. 1758-1759.

[16] 이원용, 박두순. “그래픽 사용자 인터페이스를 이용한 병렬 프로그래밍 환경”, 한국인터넷정보학회 학술발표대회 논문집 2(2), 2001.11, pp. 408-413.

[17] 류충규, 이철현, “스크래치 프로그래밍이 초등 영재학생들의 창의적 문제해결력에 미치는 효과”, 한국실과교육학회지 25(1), 2012.3, pp. 149-169.

[18] John Maloney et al. “The Scratch Programming

Language and Environment", ACM Transactions on Computing Education, vol. 10, no. 4, 2010.11, article no. 16.
[19] 임화경, 조용남. "Kodu 비주얼 프로그래밍 언어를 사용한 초등학생의 창의적 3D 게임프로그래밍 학습", 한국컴퓨터정보학회 논문지 제 17권 제 11호, 2012.11, pp. 53-61.
[20] Matthew B. MacLaurin. "The design of kodu: a tiny visual programming language for children on the Xbox 360", ACM SIGPLAN Notices, vol. 46, no. 1, 2011.1, pp. 241-246.
[21] 한인규, 임성수. "가상화 환경에서 임베디드 시스템을 위한 모니터링 프레임워크와 디버깅 시스템", 정보과학회 컴퓨팅의 실제 논문지 제 21권 제 12호, 2015.12, pp. 792-797.
[22] http://www.roboidstudio.org/
[23] Jung-Wei Chen, DDS, MS, MS, Jiajie Zhang. "Comparing Text-based and Graphic User Interfaces for Novice and Expert Users", AMIA Annu Symp Proc, 2007, pp. 125 - 129.

저 자 소 개

강기욱(학생회원)

2015년 숭실대학교 컴퓨터 학과 학사 졸업.
2016년 숭실대학교 컴퓨터 학과 석사 재학.
<주관심분야 : 시스템 소프트웨어, 운영체제>

이정환(학생회원)

2016년 숭실대학교 컴퓨터 학과 학사 졸업.
2016년 숭실대학교 컴퓨터 학과 석사 재학.
<주관심분야 : 시스템 소프트웨어, 운영체제>

홍지만(종신회원)

1999년 서울대학교 컴퓨터학과 박사 졸업.
2004년~2007년 광운대 컴퓨터학과 교수.
2007년~ 숭실대학교 컴퓨터학과 교수 재직중.
<주관심분야 : 시스템 소프트웨어, 운영체제>