J. Soc. Korea Ind. Syst. Eng Vol. 38, No. 4 : 72-79, December 2015
http://dx.doi.org/10.11627/jkise.2015.38.4.72
ISSN : 2005-0461(print)
ISSN : 2287-7975(online)

A Real Time Temperature Monitoring System for Plating Process

Sun-Wung Jung · Tae-Lin Choi · Woosik Yoo · Byung Soo Kim†

Department of Industrial and Management Engineering, Incheon National University

도금공정 실시간 원격 온도 모니터링 시스템

정선웅 · 최태린 · 유우식 · 김병수†

인천대학교 산업경영공학과

A number of plating companies have been exposed to the risk of fire due to unexpected temperature increasing of water in a plating bath. Since the companies are not able to forecast the unexpected temperature increasing of water and most of raw materials in the plating process have low ignition temperature, it is easy to be exposed to the risk of fire. Thus, the companies have to notice the changes immediately to prevent the risk of fire from plating process. Due to this reason, an agile and systematic temperature monitoring system is required for the plating companies. Unfortunately, in case of small size companies, it is hard to purchase a systematic solution and be offered consulting from one of the risk management consulting companies due to an expensive cost. In addition, most of the companies have insufficient research and development (R&D) experts to autonomously develop the risk management solution. In this article, we developed a real time remote temperature monitoring system which is easy to operate with a lower cost. The system is constructed by using Raspberry Pi single board computer and Android application to release an economic issue for the small sized plating manufacturing companies. The derived system is able to monitor the temperature continuously with tracking the temperature in the batch in a short time and transmit a push-alarm to a target-device located in a remoted area when the temperature exceeds a certain hazardous-temperature level. Therefore, the target small plating company achieves a risk management system with a small cost.

Keywords : Plating Process, Real Time Monitoring, Risk Management, Android, Raspberry Pi

1. 서 론

2000년 이후 기업들은 설비의 자동화 및 정보화를 활발히 진행해 오면서 공정 내에서 발생하는 데이터의 수집 및 활용의 중요성을 알게 되었다. 그로 인해 Computer Integrated Manufacturing(CIM) 및 Manufacturing Information System(MIS)와 같은 Information and Communication Technology(ICT)를 이용한 공정 관리 및 개선활동의 도입은 실패비용을 줄이고 효과적인 사고예방을 가능하게 하는 중요요소로 자리잡아가고 있다.

대다수의 도금공정은 장시간 동안 연속적인 전기 · 화학적 처리과정을 거치기 때문에 항상 화재위험에 노출되어있다. 특히, 중 · 소기업의 경우 비용적인 측면에서 화재방지를 위해 상용화 된 시스템의 구입이나 솔루션 업체의 컨설팅을 받기는 현실적으로 어려우며 24시간 근로자를 배치하는 것 역시 쉽지 않다.

인천 남동공단은 본 연구의 사례기업인 J사와 같은 중 · 소기업이 다수 위치하며 화재 또한 빈번하게 발생한다. <Table 1>은 J사의 소재지인 인천광역시 남동공단의 2014년

Received 20 July 2015; Finally Revised 5 October 2015;
Accepted 27 November 2015
† Corresponding Author : bskim@inu.ac.kr

국가화재정보센터 화재통계자료이다[8]. 이에 따르면 관리 소홀 및 실수로 발생한 화재인 실화가 118건 중 111건으로 약 94.1% 차지하며 화재로 인한 피해액 또한 499,937만 원 중 370,110만 원으로 약 74.1%로 높은 퍼센트를 차지함을 알 수 있다.

본 연구의 대상인 도금공정의 경우 발화점이 낮은 용액들을 주로 사용하기 때문에 급격한 온도 변화 시 화재발생 위험이 매우 높다. 더욱이 도금에 사용하는 용액을 재고로 보유하고 있어 화재 발생 시 규모가 크고 피해금액 역시 높다. 따라서, 본 연구는 도금공정에서 화재발생 위험을 감소시킬 수 있는 화재예방 시스템을 제안하고자 한다.

<Table 1> National Fire Data, Incheon, Korea in 2014

Section (Dong)	Total	Numbers		Cost (Ten-Thousands Won)
		Misfires	Etc.	
Jangsu	6	6	0	506
Guwol	49	45	4	12,365
Gansuk	53	48	5	12,802
Nonhyeon	49	48	1	181,002
Gojan	69	63	6	189,107
Seochang	15	14	1	5,574
Namchone	13	11	2	64,044
Susan	7	6	1	1,303
Dorim	12	10	2	19,623
Mansu	54	47	7	13,605
Total	327	298	29	499,937

Raspberry Pi는 영국 라즈베리파이 재단이 기초 컴퓨터 과학 교육을 증진시키기 위해 만든 싱글보드 컴퓨터이다. 크기는 약 85.60×53.98(mm)이고 무게는 약 45g이다. Raspberry Pi는 단일 칩 시스템을 사용하고 700Mhz Single-Core processor와 VGA Core, 512MB의 메모리를 장착하고 있다. 외부 기억 장치로는 Micro SD card를 사용하며 HDMI 포트를 이용한 VGA 출력과 4개의 USB 포트가 장착되어 있어 사용의 편의성과 범용성이 높은 장치이다. Ethernet Chipset을 이용한 네트워크를 사용할 수 있으며 가격 또한 저렴하다. 작은 크기와 저렴한 가격, 그리고 General purpose input output(GPIO)를 이용한 하드웨어 제어가 가능한 점 등을 고려해 보면 중・소 제조공장 내에서 제어 보조기기로서의 역할을 수행하기 적합하다[6]. 또한 Debian 계열의 리눅스(Linux) 운영체제인 Raspbian을 사용하기 때문에 필요한 패키지의 설치 및 유지보수가 간편하다는 장점이 있다[9].

Android는 리눅스(Linux) 2.6 Kernel을 기반으로 강력한 운영체제와 라이브러리 및 사용자 인터페이스 등을 제공한다. Android는 완전 개방형 플랫폼으로써 기반 기술인 소스코드를 완전히 공개해 개발자들의 소프트웨어의 확장 및 재사용의 자유성을 보장한다. Open-Source이므로 별도의 사용료를 지불하지 않고 사용할 수 있다[7].

최근 IOT(Internet On Things)와 관련한 많은 연구와 저서들이 출간되고 있다[18, 19]. 그 중 아날로그신호를 Raspberry Pi를 통해 계측하고 PC Server 및 Local monitor와의 연결을 통해 Home automation이나 Management System을 구축하는 연구들이 다수 발표되고 있다.

과거 IOT 개념이 생기기 전의 연구 중 하나인 Park et al.[17]은 Zigbee와 RFID를 이용해 시스템의 저비용의 통합관리시스템을 개발하는 연구를 진행했으나 웹베이스가 아니므로 한계가 있었다. Kim et al.[13]은 센서 네트워크 기반의 카메라 모듈을 이용한 실시간 공정관리가 가능한 시스템을 개발하였다. Lee et al.[16]은 조선산업 환경에서 MES (Manufacturing Execution System)개념을 기반으로 생산정보를 수집하여 실시간 공정관리가 가능한 시스템을 개발했다. 또한 Lee et al.[15]는 Relay회로에서 발생하는 신호를 감지할 수 있는 단말기를 통해 공정데이터를 Server로 전송하는 연구를 진행했다. 이와 같은 연구들은 PC에 서버와 UI를 구현하여 PC가 아닌 환경에서 공정의 현 상태를 알기 어렵다. IOT 개념이 생긴 후의 연구들 중 하나인, Goradiya and Pandya[4]는 ARM(Advanced RISC Machines) Architecture를 기반으로 Arduino에서 측정되는 온도를 Serial 통신을 이용해 Raspberry Pi에서 수신하는 연구를 진행했는데 이는 Serial 통신구축에 그쳤다. Jain et al.[10]은 Raspberry Pi를 통해 Home-Automation 구현과 Serial 통신이 아닌 E-mail로 관련정보를 수신할 수 있는 플랫폼에 관한 연구를 진행했다. Dhaval et al.[3]은 PC와 Raspberry Pi를 이용한 프로젝터(Projector) 제어를 연구하면서 Raspberry Pi와 PC를 무선통신으로 이루어진 시스템으로 구축했다. Lagu and Deshmukh[14]는 공정에 Raspberry Pi를 이용해 정수처리시설의 자동화에 관한 연구를 진행하면서 Raspberry Pi를 공정에서 활용하는 시스템을 제안했다. Jose-Luis et al.[11]은 Raspberry Pi에서 스마트센서의 데이터를 TCP/IP 통신으로 PC에 전달하고 PC가 로봇에 신호를 전달하는 3단계로 이루어진 네트워크 시스템을 제안했다. 또한 Kim et al.[12]와 Choi et al.[2]는 공정상에서 네트워크 시스템을 이용한 온도 측정 시스템을 연구하였다. 한편 수집된 온도 데이터를 활용한 상태기반보전활동(Condition-Based Maintenance)에 관련한 연구는 Gray and Watson[5]의 연구가 있다. 이 연구는 풍력발전기의 터빈(Turbine)의 상태를 실시간으로 감시하여 실패를 줄이고자 하는 연구인데, 본 연구에서는 온도 수집 이후의 활동은 다루지 않고, 온도를 수집하여 이상 온도 시 알림을 제공하는데 목적이 있다. 위의 연구들과는 달리 웹베이스가 아닌 CDMA (Code Division Multiple Access)방식을 이용한 원격 모니

터링 시스템에 관한 연구가 Bae et al.[1]에 의해 진행되었는데 이 역시 스마트폰이 아닌 PC로 데이터를 전송함으로써 실시간으로 상태를 모니터링 하기엔 역부족이다. 또한, Raspberry Pi-PC Server-스마트폰으로 이루어진 네트워크 시스템을 이용해 도금공정에서의 온도관리가 가능한 위험관리 솔루션에 대한 사례는 존재하지 않는다.

따라서, 본 연구에서는 Raspberry Pi의 GPIO를 이용해 온도를 계측 및 모니터링하고 위험온도수준을 초과할 경우 Android OS 스마트폰으로 Push-alarm을 전송하는 시스템을 개발하고자 한다. 제안된 시스템은 다수의 Raspberry Pi에서 전송되는 데이터를 처리하기 위한 Multi-thread Based Async Socket Server를 개발하여 실제 현장의 병렬 공정 상황에서 실시간으로 과열 경고 수신을 가능케 하고 화재 발생으로 인한 실패비용을 줄이고자 한다.

2. J사의 현황 및 문제점

본 장에서는 사례기업인 J사에서 위험 및 공정관리에 사용중인 도구와 화재위험에 노출될 수 밖에 없는 회사 현황 및 문제점에 대해 설명한다.

2.1 J사의 현황

본 절에서는 J사가 과열 등의 위험상황을 인지하기 위해 사용중인 경고시스템과 화재상황과 직결되는 요소인 온도를 관리하는 방법에 대해 다루고자 한다. J사의 실제 현장에서 활용중인 위험 및 공정관리 도구는 계기반과 사이렌 및 Check-sheet이다. 계기반은 현재 도금조의 온도를 디스플레이 상에 표시하는 역할을 한다. 계기반은 각 도금조별로 존재하고 모든 도금조의 온도를 볼 수 있는 주 계기반이 존재한다. 사이렌은 계기반에 표시된 온도가 위험온도를 초과할 시 소리 신호와 함께 점멸등으로 작동한다. Check-sheet은 도금조를 구성하고 있는 기계장치와 온도를 점검한다. 이러한 도구와 장치들을 통해 위험 및 공정관리를 진행하고 있지만, 화재예방을 실시간으로 감시하기엔 역부족이다.

2.2 문제점

2.2.1 계기반과 Check-sheet의 한계

계기반은 사이렌과 온도를 표시하는 장치로 구성되어 있다. 위험온도를 초과할 시 사이렌을 이용해 작업자들에게 위험상태를 인지시킬 수 있다. 그러나 현장내의 냉각기와 컨베이어벨트의 소음으로 인해 사이렌 소리를 인지하지 못하는 일이 빈번하게 발생하기 때문에 실효성이 없는 실정이다.

Check-sheet은 공정관리상태를 인지하기 용이하며 항목누락을 방지할 수 있다는 장점이 있다. 하지만 J사에서 사용 중인 Check-Sheet는 화재의 중요 관리요소인 온도에 대한 기록란이 존재하지 않고 계기반에 표시되는 온도를 기반으로 정상공정인지 판단하고 기록하는 방식이기 때문에 위험에 쉽게 노출될 수 있다. 또한 공정데이터를 활용할 수 있는 방법이 없기 때문에 화재예방 및 품질개선의 여지가 존재하지 않는다.

2.2.2 야간 감시 불가

도금공정의 특성상 장시간에 걸쳐 연속적인 전기·화학적 처리과정을 거친다. 이는 야간 시간대에도 연속적인 작업이 이루어 질 수 있음을 의미하는데, 비용적인 측면에서 영세제조업체가 야간 감시에 별도의 인력을 배치하는 것은 불가능하다.

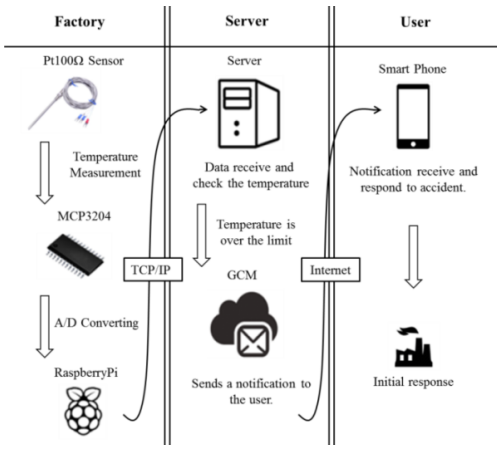
따라서 관리자가 실시간으로 화재발생여부를 인지할 수 있고 비용적인 측면을 고려한 시스템이 필요한 시점이다.

3. 시스템 구성

본 장에서는 앞서 소개한 하드웨어와 소프트웨어를 이용해 화재여부를 신속히 인지할 수 있는 시스템의 개념도와 단계별 방법에 대해 설명하고자 한다.

3.1 시스템의 개념도

본 절에서는 실시간 원격 모니터링 시스템의 개념도와 정보의 흐름을 다루고자 한다. 아래 <Figure 1>은 실시간 원격 모니터링 시스템의 개념도를 단계별로 나타낸다.



<Figure 1> Conceptual Diagram for a Real Time Temperature Monitoring System

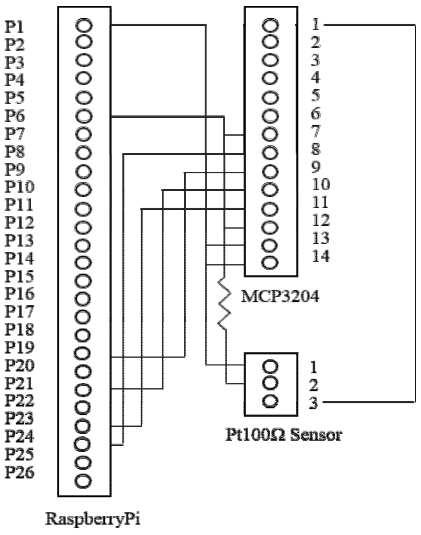
도금공정 중 실시간으로 온도를 계측하여 서버로 전송한다. 서버에서는 수신된 데이터를 기반으로 위험온도에 대한 판별을 시행한다. 이 때, 온도가 위험온도를 초과할 시 현장근무자 및 관리자에게 경고를 전달하고 정상온도이면 계속하여 진행한다.

3.2 단계별 방법

본 절에서는 <Figure 1>을 기반으로 하여 실시간 원격 모니터링 시스템의 각 단계별 세부구성에 대해 논하고자 한다. Factory 단계에서 화재예방에 필요한 온도 데이터를 측정하고 서버에 전송한다. 센서-Raspberry Pi간 통신은 직렬 통신으로 이루어지며 Raspberry Pi는 서버에 TCP/IP 통신을 이용해 데이터를 전송한다. 데이터를 수신한 서버는 전달받은 데이터가 위험온도인지 판단하여 위험온도이면 GCM 서비스를 통해 사용자의 스마트폰으로 Push-alarm을 전송한다.

3.2.1 Factory 단계

아래 <Figure 2>는 Raspberry Pi와 MCP3204 그리고 Pt100Ω 저항센서를 연결한 회로도 이다. 온도계측에 사용하는 Pt100Ω 측온저항센서는 외피가 PTFE(불소수지)로 이루어져있어 온도 변화폭이 넓은 도금공정에 사용하기 적합하다. <Figure 2>의 회로도를 따르면 MCP3204의 여유채널이 4개이기 때문에 한 개의 회로에서 4개의 도금조를 감시할 수 있다. 센서를 통해 측정된 데이터는 아날로그 형태로 출력되며 Raspberry Pi에는 별도의 AD Converter가 존재하지 않기 때문에 MCP3204를 이용해 Digital로 변환하는 과정이 필요하다.



<Figure 2> Schematic Diagram of Connection between Raspberry Pi, MCP3204, and Pt100Ω Sensor

변환 시 시스템의 주변환경 및 회로상태에 따라 측정수치에 오차가 발생할 수 있다. 따라서 오차를 보정해야 하는데, 이에 필요한 수식은 <Figure 3>에 보여지는 MCP3204의 Digital Output Code를 구하는 수식을 변형시켜 얻을 수 있다. 이 수식과 저항센서의 온도대별 이론적 저항수치를 비교하여 오차를 보정하는 온도보정상수를 결정한다.

$$Digital\ Output\ Code = \frac{4096 \times V_{IN}}{V_{REF}}$$

V_{IN} = analog input voltage

V_{REF} = reference voltage

<Figure 3> Equation of MCP3204

<변수>

R_1 : 저항센서로부터 측정된 저항수치

V_{IN} : Input Voltage

Temp : 측정된 온도수치

R_α : 저항보정상수

T_α : 온도보정상수

<수식>

$$R_1 = \frac{100(3.3 - V_{IN})}{V_{IN}} + R_\alpha \quad (1)$$

$$\mathrm{Temp} = \frac{(R_1 - 100)}{0.39} + T_\alpha \quad (2)$$

식 (1)은 측정된 저항수치를 계산하기 위한 수식이며 이 수식의 결과값은 온도를 계산하기 위해 쓰인다. 상수 100은 저항센서의 자체 저항수치이며 상수 3.3은 Raspberry Pi GPIO의 출력 전압수치, V_{IN}은 A/D Converter를 거쳐 들어오는 전압수치이다. R_α는 저항보정상수로써 저항 측정 시 회로상태나 주변환경 등에 따라 측정수치의 변화가 심하게 일어난다. 따라서 <Table 2>에 표기된 특정온도대의 이론적 저항수치를 참고하여 저항보정상수를 결정할 필요가 있다.

식 (2)는 온도수치를 계산하기 위한 수식이다. Pt100Ω 측온저항센서는 IEC751 규격을 따르는데, 이 규격은 저항소자의 온도가 0℃일 때 저항수치가 100Ω이고 저항소자의 온도가 100℃일 때 저항수치는 139Ω으로 나타난다. 이를 따르면 온도가 1℃ 상승할 때 저항은 약 0.39Ω 상승하는 것을 알 수 있다. R_1은 식 (1)의 결과 값이며 상수 0.39는 온도 1℃당 저항의 상승폭, 상수 100은 저항센서의 자체저항수치이다. T_α는 온도보정상수로써 온도를 측정하는 저항센서의 리드선 길이 및 센서의 사용연한 등 센서 상태에 따라 측정수치의 변화가 심하게 일어난다. 따라서 <Table 2>를 참고하여 온도보정상수를 결정할 필요가 있다.

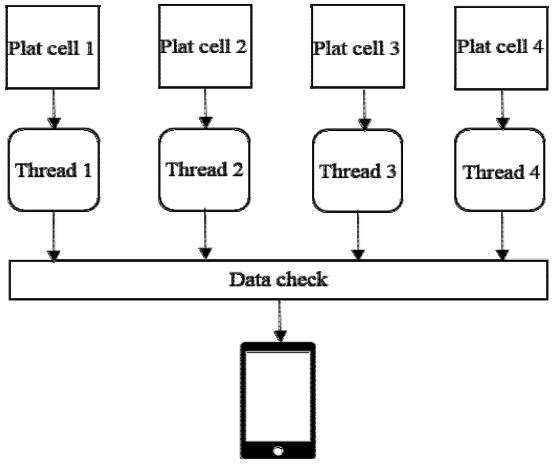
<Table 2> Theoretical Resistance Value

Temp	0	1	·····	8	9
-20	92.16	91.77		89.01	88.62
-10	96.09	95.69		92.95	92.55
0	100.00	99.61		96.87	96.48
0	100.00	100.39		103.12	103.51
10	103.90	104.29		107.02	107.40
20	107.79	108.18		110.90	111.28
30	111.67	112.06	·····	114.77	115.15
40	115.54	115.92		118.62	119.01
50	119.40	119.78		122.47	122.86
60	123.24	123.62		126.31	126.69
70	127.07	127.45		130.13	130.51
80	130.89	131.27		133.94	134.32
90	134.70	135.08		137.74	138.12
100	138.50	138.88		141.53	141.91

이를 따라서 저항 보정상수 및 온도보정상수를 Raspberry Pi 내부에 Python으로 개발된 프로그램에 저장한다. 이 과정을 통해 측정오차를 최소화시켜 데이터의 신뢰성을 증가시킬 수 있다.

3.2.2 Server 단계

Raspberry Pi와 Server간의 통신 프로토콜은 TCP/IP 통신을 사용한다. <Figure 4>는 Multi-thread based Async socket server의 개념도이다. 실제 현장의 도금조의 수는 다수이기 때문에 동시다발적인 데이터 처리를 위해 Multi-thread Server가 필요하며 통신의 동기화가 불필요 하므로 Async Socket Server를 이용한다. 데이터 수신 후 서버는 조건문을 통해 위험온도를 판별하고 위험온도를 초과할 시 GCM을 통해 User 단계로 Push-alarm을 전송한다.

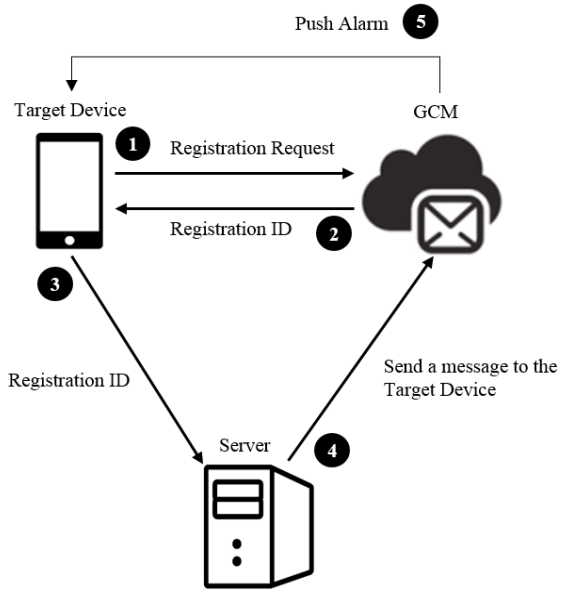


<Figure 4> Multi-Thread Based Async Socket Server

3.2.3 User 단계

Server 단계에서 데이터에 대한 판별 후 User 단계로 Push-alarm을 전송할 때 GCM(Google Cloud message)를 사용한다. GCM은 Google에서 제공하는 서비스인데, 이는 Server-target device간에 메시지를 간단히 주고받을 수 있는 서비스이다. 서비스 사용과 구축에는 <Figure 5>와 같은 간단한 절차가 필요하다.

Push-alarm의 전송을 위해선 GCM 서버에 등록하는 과정이 필요하다. 먼저 Target device에서 GCM 서버로 Target device 등록을 요청한다. GCM 서버는 요청을 받은 후 Target device의 등록ID(Registration ID) 정보를 반납하고 Target device는 등록ID를 수신함과 동시에 Server에 등록ID 정보를 전송한다. Server는 Raspberry Pi에서 전송 받은 데이터가 위험온도를 초과할 시 등록ID 정보에 해당하는 Target device로 Push-alarm을 보낼 것을 GCM 서버에 요청한다. GCM 서버는 Server의 요청을 받으면 등록ID 정보를 검사 후 Target device로 Push-alarm을 전송한다.



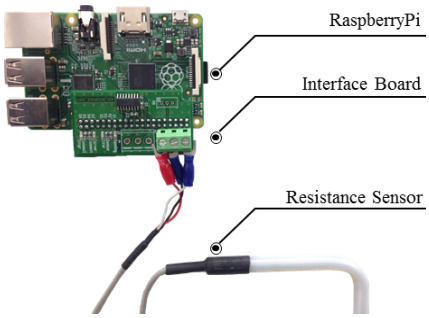
<Figure 5> System Process Procedure for Google Cloud Message

4. 구성 결과

본 장에서는 앞서 설명한 시스템 구성의 결과를 Hardware part와 Software part, 그리고 보정상수 적용의 유효성을 보이고자 한다. Hardware part는 <Figure 2>의 회로도를 기반으로 구성되었으며 Software part는 Raspberry Pi 프로그램의 실행화면과 Push-alarm의 수신화면을 보인다. 마지막으로 보정상수적용의 유효성을 확인하기 위해 기존 온도계와 비교실험을 진행하고 경제성을 분석한다.

4.1 Hardware Part

앞서 설계한 <Figure 2>를 기반으로 한 회로의 안정성을 높이기 위해 Interface board를 제작하였다. Interface board는 저항센서와 연결할 수 있고 Raspberry Pi의 GPIO에 직접 장착되어 저항센서에 의해 측정된 데이터를 전송한다. <Figure 6>은 Raspberry Pi-저항센서-Interface board를 연결한 그림이다.



<Figure 6> Hardware Part

4.2 Software Part

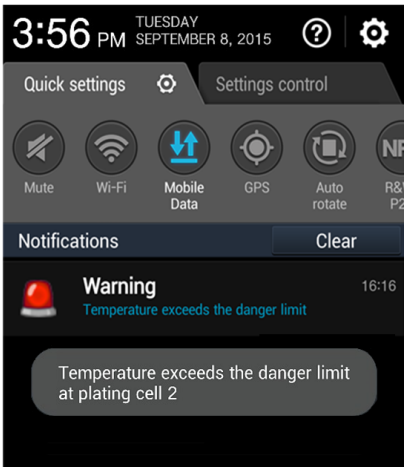
<Figure 7>은 Raspberry Pi에서 실행되는 프로그램이다. 저항센서로부터 계측된 저항값을 계산하고 저항보정상수와 온도보정상수를 적용시켜 최종적으로 Server에 전송하는 Client 역할을 한다.

```
Function CalculateTemp(self, channel)
 While
   ReadResister ← self.read(channel)
   Vout ← Vin × (ReadResister)/4096
   Resister ← 100 × (Vin - Vout) )/Vout + Ralpha
   Temperature ← (Resister - 100)/0.39 + Talpha
 Return Temperature

Function SendData(self)
 While True
   Socket ← socket.(SOCK_STREAM)
   Packet ← self.MakePacket()
   Socket.connect()
   Socket.send(Packet)
   Socket.close()
```

<Figure 7> Pseud Code of Client Program

<Figure 8>은 GCM 서버로부터 Push-alarm을 정상적으로 수신한 Android Application User interface(UI)이다. 수신 시 팝업 알림과 동시에 상단 알림 바에 메시지가 표시된다. 어떤 도금조의 온도가 위험온도를 초과했는지 실시간으로 인지할 수 있게 한다.



<Figure 8> Push-Alarm UI Using Android

4.3 보정상수적용의 유효성 확인 실험

<Figure 3>을 통해 산출된 보정상수를 적용시킨 온도측정시스템과 기존 온도계 간의 측정수치 비교실험을 진행한다. 이는 제안한 시스템의 온도측정정밀도를 입증하기 위해 필요하다. 사례기업인 J사에서 현재 도금공정 온도측정에 사용중인 A사의 T제품과 제안된 시스템 간의 측정온도수치를 비교한다. 실험방법은 3개의 온도구간을 설정 후 약 5초의 간격으로 데이터를 기록한다. 제안한 시스템과 T제품에서 측정된 온도의 평균을 비교하여 그 차이를 산출하여 판단한다. 산출 수식은 다음과 같다.

<변수>

Avg_T : 제품T 측정치의 평균

Avg_PS : 제안시스템 측정치의 평균

<수식>

$$\mathrm{GAP}(\%) = \frac{|Avg\_T - Avg\_PS|}{Avg\_T} \times 100$$

편의상 30℃~39℃를 구간 (1), 40℃~49℃를 구간 (2), 50℃~59℃를 구간 (3)으로 칭한다. <Table 3>은 각 구간의 측정된 온도 데이터이다.

<Table 3> Average Temperature Gap between Product T and Proposed System

Temperature Class (℃)	Average Temperature (Proposed System) (℃)	Average Temperature (Product T) (℃)	Gap (%)
(1) 30~39	37.4	37.3	0.26
(2) 40~49	43.7	43.8	0.22
(3) 50~60	54.8	54.8	0

구간 (1)의 각각의 평균값은 37.4와 37.3로 산출되었으며 평균의 차이는 0.1이며 약 0.26%의 차이가 발생했다. 구간 (2)의 각각의 평균값은 43.7과 43.8로 산출되었으며 평균의 차이는 0.1이며 약 0.22%의 차이가 발생했다. 구간 (3)의 각각의 평균값은 54.8과 54.8로 산출되었으며 평균의 차이는 0이며 0%의 차이가 발생했다. 이 결과로 미루어 볼 때, 제안한 시스템의 측정오차는 약 ±0.1℃ 수준이며 액체를 주로 사용하는 도금공정에 사용하기 적합할 것으로 판단된다.

4.4 제안된 시스템과 기존제품간의 경제성 분석

제안된 시스템의 경제성을 분석하기 위해 시스템을 구성하고 있는 요소들의 비용과 기존 제품 *T*의 비용을 비교하여 진행했다. 제안된 시스템을 구성하는 요소들의 총 비용과 기존 제품 *T*의 비용을 비교해 보면 제안된 시스템을 사용하는 것이 약 16.9%의 비용절감 효과를 가져오는 것을 알 수 있다.

5. 결 론

본 연구는 도금공정 상황에서 발생할 수 있는 화재위험을 예방하고 중·소기업의 상황에 적합한 낮은 비용의 실시간 원격 모니터링 시스템을 제안하였다.

제안된 시스템은 도금공정에서 연속적으로 발생하는 공정데이터를 활용하여 위험온도 초과 시 관리자나 근무자에게 실시간으로 경고를 할 수 있다. 기존에 사용하던 경보 시스템은 사고현장에 근로자가 배치되어 있지 않거나 경보 사이렌의 소리를 청취하지 못할 시 신속하게 대응하기 어렵다는 치명적인 단점이 존재하였다. 하지만 제안된 시스템은 일반적으로 상시 지니고 다니는 스마트폰에 경고를 전달함으로써 신속한 사고인지와 즉각적인 조치를 가능하게 하였다. 또한 비용적인 측면에서도 제안된 시스템이 우수하므로 공정관리 측면에서 효율적이다. 도금공정 실시간 원격 모니터링 시스템의 개발은 개념적으로 유사한 모든 업종에 적용할 수 있을 것으로 보이며 낮은 도입비용으로 인한 중·소기업의 정보화를 도모할 수 있다. 또한 수집된 온도 데이터를 기반으로 상태기반보전(Condition-Based Maintenance) 활동을 할 수 있는 초석이 될 것으로 예상된다.

향후 연구로, 가동시간 및 작업시간과 같은 공정데이터를 적극적으로 활용하여 낮은 도입비용과 간편한 활용이 가능한 통합 위험관리 솔루션으로 발전시킬 계획이다.

Acknowledgement

이 논문은 2015년도 정부(교육부)의 재원으로 한국산업기술진흥원의 산업단지캠퍼스조성사업지원을 받아 수행된 것임(과제번호 : 201500490001).

References

[1] Bae, Y.H. and Kim, H.C., A Study on real time remote monitoring system for equipment. *Journal of the Korean Institute of Plant Engineering*, 2014, Vol. 19, No. 4, pp. 39-47.

[2] Choi, J.M., Park, K.W., Kim, J.H., Ahn, D.S., Oh, S.Y., and Kim, Y.J., Context-aware Temperature Control System Using sensor Network. *The korea Society of Digital industry and Information Management*, 2007, Vol. 3, No. 1, pp. 27-34.

[3] Dhaval, C.D., Divyesh, D.D., and Shraddha, C.T., Smart Projectors using Remote Controlled Raspberry Pi. *International Journal of Computer Applications*, 2013, Vol. 82, No. 16, pp. 6-11.

[4] Goradiya, B.C. and Pandya, D.H.N., Real time Monitoring and Data logging System using ARM architecture of Raspberry Pi and Ardiuno UNO. *International Journal of VLSI and Embedded Systems-IJVES*, 2013, Vol. 4, No. 06118, pp. 513-517.

[5] GRAY, C.S. and WATSON, S.J., Physics of failure approach to wind turbine condition based maintenance. *Wind Energy*, 2010, Vol. 13, No. 5, pp. 395-405.

[6] http://en.wikipedia.org/wiki/Raspberry_Pi.

[7] http://www.android.com.

[8] http://www.nfds.go.kr.

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

[10] Jain, S., Vaibhav, A., and Goyal, L., Raspberry Pi based interactive home automation system through E-mail. *Optimization, Reliability, and Information Technology(ICROIT)*, International Conference, 2014, pp. 277-280.

[11] Jose-Luis, J.G., Jose-Luis, P.L., Juan-Luis, P.Y., David B.M., and Jose-Enrique., Performance and Results of the Triple Buffering Built-In in a Raspberry PI to Optimize the Distribution of Information from a Smart Sensor. *Distributed Computing and Artificial Intelligence, 11th International Conference Advances in Intelligent Systems and Computing*, 2014, Vol. 8, pp. 279-286.

[12] Kim, Y.D., Kim, H.R., Kim, D.S., and Kim, G.I., Development of the temperature monitoring system for in-

sulated phase bus. *Research Institute of Industrial Science and Technology*, 2005, Vol. 19, No. 1, pp. 66-72.

[13] Kim, Y.G., Kim, D.S., Lee, W.J., and Heo, S.Y., Development of camera module production process management system based on a sensor network. *Journal of Society of Korean Institute of Information Technology*, 2010, Vol. 8, No. 5, pp. 93-101.

[14] Lagu, S.S. and Deshmukh, S.B., Raspberry Pi for automation of water treatment plant, Advances in Computing. *Communications and Informatics(ICACCI)*, International Conference, 2014, pp. 1999-2003.

[15] Lee, J.H., Lee, S.H., and Oh, H.O., Development of a Process Monitoring System for Real-Time Process Control. *Journal of the Society of Korea Industrial and System Engineering*, 2008, Vol. 31, No. 1, pp. 92-100.

[16] Lee, K.H., Yoon, T.H., and Oh, J.T., Development of Real-Time Process Monitoring System in Shipbuilding Industry. *Society of Korea Institute of Industrial, Autumn Conference*, 2013, pp. 607-611.

[17] Park, J.J. and Cho, J.H., A Research on the Design of an Automatic Manufacturing System and the Real-time Process Monitoring System using ZigBee. *Journal of the Korean Institute of Plant Engineering*, 2009, Vol. 14, No. 4, pp. 125-131.

[18] Simon, M., Raspberry Pi Cookbook. 1st ed., O'Reilly Media, 2014.

[19] Steven, G.W., Smart Home Automation with Linux and Raspberry Pi. 2nd ed., Apress, 2013.

ORCID

Woosik Yoo | http://orcid.org/0000-0002-0967-9051

Byung Soo Kim | http://orcid.org/0000-0002-8340-1306