시스템프로그래밍 프로젝트 보고서

(스마트 주문 관리 & 무인 서빙 시스템)

4horsemen (6조)

조장 : 강동훈 (201220987)

조원 : 최정용 (201221764)

조원 : 전동훈 (201320994)

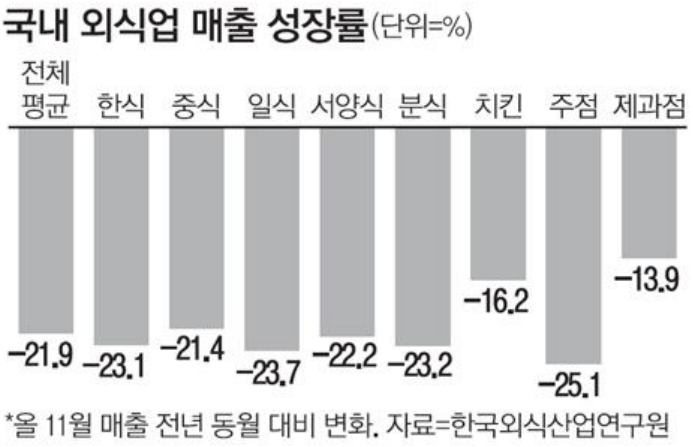
조원 : 송용승 (201120899)

목차

1. 프로젝트 개요

이 프로젝트는 크게 두가지로 나눠진다. 서빙하는 인원을 줄이기 위해 만든 무인 서빙 시스템과 주문을 한곳에서 통합하여 관리 할 수 있게 만든 스마트 주문 관리 시스템이다. 점장은 임베디드 보드를 이용하여 만들어진 스마트 주문관리 시스템으로 손님이 주문하는 음식을 기록한다. 손님이 음식 주문을 완료하면 임베디드 보드를 이용하여 무선으로 무인 서빙 시스템을 담당하고 있는 RC카에 신호를 보내 RC카를 손님이 있는 테이블까지 보낸다. 이것이 우리 프로젝트의 메인 시나리오 이다.

2. 프로젝트 동기 및 목표



출처 : MK News : 불황에 외식 안한다

현재 많은 자영업자들은 장기적인 불황에 허덕이며 제한된 금전적 상황에서 수익을 창출하기 위해 많은 노력을 쏟고 있다. 그 중에서도 요식업에서는 좋은 재료를 사용해야만 좋은 음식을 내놓을 수 있기에 음식 재료비를 제외한 다른 요소들에서 지출을 최소화 하려고 하고 있다. 이런 현상으로 인해 점점 요식업에서 재료비가 차지하는 비중이 점점 커지고 있다.

외식업체 매출 대비 식재료비 비율 자료: 농림축산식품부, 2015 외식업체 식재료 구매 현황

	2010년	2011년	2012년	2013년	2014년
식재료비 비율	35.7%	35.7%	35.7%	35.7%	40.6%

출처 :2015 외식업체 경영실태 조사 보고서

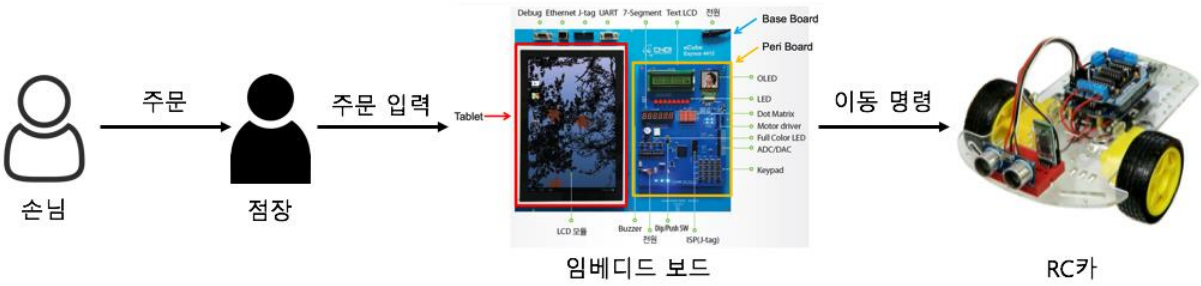
이에 우리 4horsemen은 소형 요식업을 하는 자영업자들을 위하여, 서빙을 담당하는 종업원의 수를 줄이기 위한 무인 서빙 카트, 그리고 이 무인 카트와 음식 주문을 한곳에서 관리할 수 있는 스마트 주문 관리 시스템을 만들기로 하였다.

3. 분석

우리 시스템이 기존 시스템과 비교하여 갖는 차별점은 바로 무인 서빙 카트이다. 앞서 말했듯이 무인 서빙 카트는 사람의 도움을 전혀 필요로 하지 않는다. 따라서 점주가 고용하는 점원의 수가 현저히 줄어들 수 있고, 점원이 뜨거운 음식을 서빙을 하다 일어날 수 있는 사고에도 예방을 할 수 있어 기존의 시스템과 확실히 다른 차별점을 갖고 있다.

4. 설계 및 구현

- 전체 시스템 개요도

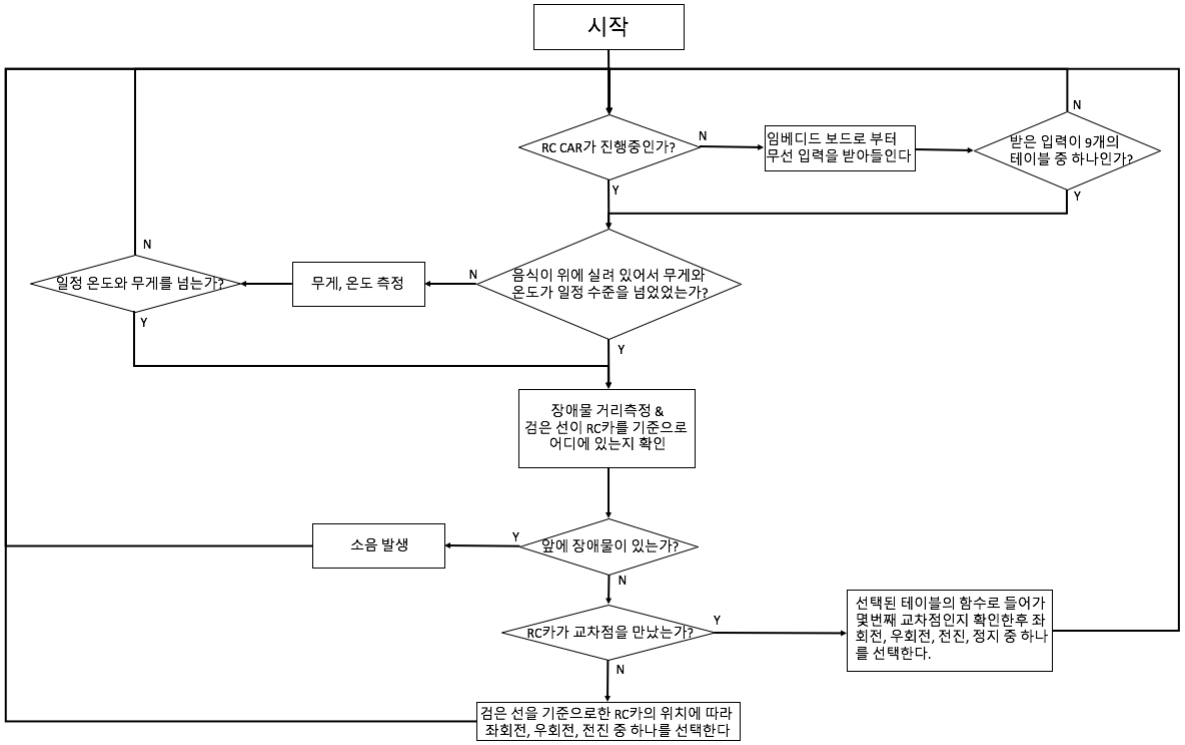


손님이 점장에게 말로 주문을 하면 점장이 이를 임베디드 보드에 기록을 하고 주문이 끝나면 RC카에게 음식을 서빙하라고 이동 명령을 내린다. 이렇듯 전체 시스템은 크게 임베디드 보드와 아두이노 RC카로 나뉘어 진다. 따라서 전체

알고리즘과 세부 알고리즘은 아두이노와 임베디드 보드로 나누어 설명하도록 하겠다.

4.1 아두이노

- 아두이노 전체 알고리즘의 개요



위 그림은 전체 알고리즘의 개요이다. 교차점을 만났을때 어떻게 행동하는지에 대한 설명은 아래 세부 알고리즘에서 설명되어진다.

- 세부 알고리즘의 개요

세부 알고리즘의 개요에서는 아두이노에서 가장 중요한 교차점을 만났을 때를 기준으로 설명하도록 하겠다. 교차점을 만난 경우 선택된 테이블의 번호에 따라 들어가는 함수가 달라진다. 예를들어 a를 입력받은 경우 table 1번을 담당하고 있는 table_go1 함수로 들어간다. 이 함수에 들어간후 현재까지 만난 교차점의 개수를 담당하고 있는 check_tran을 확인하여 그에 알맞는 if문을 찾아간다. 예를들어 table_go1에서 첫번째 교차점을 만난경우 check_tran==0을 가진 if문으로 들어가 교차점의 개수를 담당하는 check_tran을 하나 증가시키고 교차점을 완전히 지날때까지 check_duple에 1을 넣어서 LOCK을 걸어준다. 이렇게 하면 table_go1의 if문은

check_duple이 0일때만 수행이 가능하기 때문에 같은 교차점을 연속으로 읽는다 해도 if문을 실행하지 않아 오류가 발생하지 않는다. 그리고 현재까지 만난 교차점의 개수에 따른 함수를 수행하게 되는데 여기에서는 forward 함수를 수행하게 된다. 그 후 교차점을 지날 수 있도록 delay를 걸어 일정 시간동안 forward를 수행하게 한후 다시 적외선 센서값을 읽어들여 RC카의 행동을 결정하는데 여기서 RC카가 검은선의 정 가운데에 있으면 forward문이 실행되고 여기서 check_duple을 다시 0으로 만들어 교차점의 LOCK을 풀어준다. 이런 방식으로 아두이노 RC카는 교차점을 처리한다.

좌회전, 우회전, 전진, 후진, 정지를 수행하는 left, right, forward, backward, brake 함수들은 모두 L298N 드라이버를 이용하여 수행된다. 여기서 IN1, IN2, ENA는 오른쪽 바퀴를 담당하고 IN3, IN4, ENB는 왼쪽 바퀴를 담당한다.

ENA (or ENB)	IN1 (or IN3)	IN2 (or IN4)	모터 A (or B)
High	High	Low	정방향 회전
High	Low	High	역방향 회전
High	High	High	정지
High	Low	Low	정지
Low	상관없음	상관없음	정지

출처 : L298N 설정 방법

위의 표와 같이 작동하게 되는데 반드시 일정 시간 동안 동작할 수 있도록 마지막에는 delay를 걸어준다. right 함수는 RC카가 선을 기준으로 왼쪽으로 쏠렸을 때 작동하게 되며, left는 그 반대이다. 이런 방식으로 RC카가 선을 벗어나지 않게 하였다.

4.2 임베디드 보드

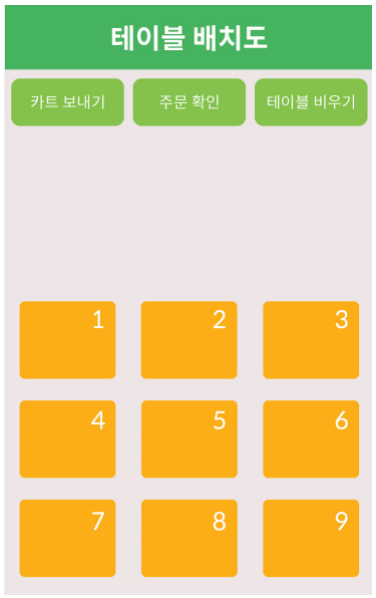
- 임베디드 보드 전체 알고리즘의 개요

임베디드 보드에서 구현은 크게 페리보드, LCD 터치, LCD 화면, 아두이노와 통신 이렇게 4가지로 나눌 수 있다. 각 부분은 무한루프로 프로그램 실행동안 계속 실행하게 된다. 쓰레드를 사용하여 다중사용이 가능하도록, 그리고 락을 이용하여 한 쓰레드에서 특정 기능을 수행할시 다른 쓰레드는 사용을 못하도록 구현하였다. 주문 받기는 페리보드의 기능만을 사용하는데, 모드에 상관없이 항상 사용이 가능하다. 주문 받는 도중에 음식이 나올 수 있으니 주문받기와 카트 보내기 기능은

동시에 사용할 수 있도록 하여 주문을 받는 도중에도 카트를 보낼 수 있도록 구현하였다.

- 세부 알고리즘의 개요

보드에서 LCD 화면의 모드는 int형 변수인 mode 를 통해 **카트 보내기** 모드, **주문확인** 모드, **테이블 비우기** 모드의 3가지로 나뉜다. 매장 관리자 입장에서 가장 많이 사용하게 될 기능이 **카트 보내기**이므로 카트 보내기 모드가 기본 mode 로 설정되어있다. LCD의 1~9 숫자중 원하는 테이블을 클릭하면, 1280x800 픽셀 크기의 LCD 상에 미리 설정해둔 구획에 따라 터치한 테이블의 번호가 ArduinoStat에 저장되고, 어떤 모드인지 mode 변수를 통해 확인 후 lockScreen 변수에 +1을 해줘 while(lockScreen==0) 문장에서(락) 빠져나올수 있도록 해준다. 그후 해당 테이블이 선택되었다는 화면을 2초간 띄워준후 다시 lockScreen 변수에 -1을 하여 락을 걸어 기능을 제한함과 동시에 오른쪽의 사진과 같은 초기 화면을 띄운다. 예를 들어 mode =1일시 카트보내기 모드인데, mode =1, ArduinoStat =9 이면 아두이노에게 문자 'i'를 보내게 된다. 문자 'i'를 받은 아두이노는 9번 테이블을 향해 움직일 것이다. 밑에 그림은 테이블 매핑 표이다.



테이블	1	2	3	4	5	6	7	8	9	평소
문자	a	b	c	d	e	f	g	h	i	x

5. 변동사항

5.1 아두이노

온도센서, 무게 센서 : 원래 계획은 온도가 너무 뜨겁거나 RC카가 견디지 못하는 무게의 음식을 올릴경우 경고음을 울리게 하였는데 이렇게 하면 계속 음식을 RC카에 올렸다 내렸다 하면서 음식의 온도와 무게를 테스트 해야하는 문제점이 생겼다. 이렇게 된다면 오히려 RC카에 올렸다 내렸다 하면서 점원이 더 힘들어질

것을 생각하여 이 계획은 폐기 하였다. 그래서 음식의 온도와 무게는 주인이 따로 온도계와 무게 측정기를 사용하여 측정하는 것으로 생각였다. 대신 온도 센서와 무게 센서가 아두이노 위에 음식이 올려지는 것을 감지하여 음식이 올려졌을때만 RC카를 움직일 수 있게 하였다.

5.2 임베디드 보드

NONE

6. 시험평가

6.1 아두이노

6.1.1 작성한 프로그램의 컴파일 환경 및 방법

아두이노 프로그래밍은 MacOS 운영체제에서 Arduino 편집기를 이용하여 편집되어졌다. 쓰레드는 따로 사용되어지지 않았다.

6.1.2 테스트 결과에 대한 설명, Bug 및 개선할 사항

모든 동작들이 정상적으로 수행되는 것을 볼 수 있다. 개선할 사항으로는 실수로 사람이 RC카를 검은 선에서 떨어뜨릴 경우 RC카가 제자리를 찾아 돌아가지 못하는데 이를 추가적으로 구현할 수 있다면 매우 안정적인 서빙 카트가 될 것이다.

6.1.3 구현한 사항과 구현하지 못한 사항

- RC카 모든 동작 구현 완료
- 모든 센서 동작 구현 완료
- 구현하지 못한 사항 없음

6.2 임베디드 보드

6.2.1 작성한 프로그램의 컴파일 환경 및 방법

실습용으로 배포한 bitmap.h 헤더파일이 있는곳에서 크로스컴파일을 실시하며 쓰레드 사용을 하였기 때문에 맨 뒤에 -lpthread 옵션을 넣어 컴파일을 한다. 그후 tftp를 이용하여 임베디드 보드로 전송 후, 실행 권한을 준 다음 실행한다. 실행시 위에서 설명한 4개의 쓰레드가 무한루프로 실행된다.

6.2.2 테스트 결과에 대한 설명, Bug 및 개선할 사항

4개의 쓰레드가 독립적으로 실행되는 것을 확인 할 수 있다. 와이파이 통신 환경이 원활한 경우엔 문제 없이 잘 실행되지만, 와이파이 환경이 불안하면 아두이노와 통신하는데 시간이 꽤 오래 걸린다. 너무 많은 기능이 들어가서, CPU 사용률이 너무 높기 때문인지 실행하는 중간에 가끔씩 CPU가 shutdown이 된다. CPU가 Shutdown 되지 않도록 최적화를 진행할 필요가 있을 것 같다.

6.2.3 구현한 사항과 구현하지 못한 사항

- 주문 & 관리 구현 완료
- 서빙 카트와 통신기능 구현 완료
- 구현하지 못한 사항 없음

7. Challenge Issue

강동훈 : 아두이노 부분을 주로 담당하였는데 생각보다 RC카의 속도가 너무 빨라서 고생하였다. 우리 RC카는 교차점의 처리때문에 속도가 느려야했는데 우리가 가진 모터는 감속비가 작아서 천천히 가면 힘이 약해 바퀴가 굴러가지 않았다. 이 문제는 더 큰 감속비를 가진 모터를 구입하여 해결하였다.

아두이노 자체의 성능이 부족하여 제대로 프로그램이 동작하지 않아 매우 고생하였었다. 원래는 아두이노 우노 위에 와이파이 쉴드를 올려서 RC카를 동작하게 하였었다. 그때는 RC카 코드와 와이파이 코드를 완전히 파일로 분리하여서 사용했기에 별 문제없이 작동 했었다. 하지만 두 코드를 합치자 아두이노 우노의 성능이 부족해지면서 우리가 원했던 동작이 일어나지 않았었다. 하지만 이게 하드웨어 적인 문제인지 알 방법이 없었던 우리는 전시회 당일 새벽까지 코드를 열심히 수정하였었고 새벽 4시까지 시도한 결과 우리는 거의 전시회 포기 직전까지 갔었다. 하지만 그때 마지막으로 내가 조교님께 자문을 구해보자는 제안을 해서 조교님께 자문을 구했는데 조교님은 이게 아두이노 자체의 성능 때문일 가능성이 있다고 하셨다. 그래서 바로 아두이노 우노 대신 아두이노 메가를 달아서 테스트한 결과 바로 성공하였다. 이번 프로젝트에 있어서 이 하드웨어의 성능을 파악하는 것이 우리에게는 가장 힘든 일 중 하나였다.

최정용 : 보드부분은 나포함 팀원 두명이 주로 진행하였는데, 서로 부족한 점이 보완이 되어서 구현하는데 딱히 어려움은 없었던 것 같다. 굳이 어려웠던 부분을 뽑자면 LCD화면을 구성하는 것이였는데, 화면이 삐뚤어져서 화면을 교정시키는 것에 대한 어려움이 있었다. 해결 방법은 강의노트도 찾아보고, 책을 찾아 보기도 하고, 조교에게 물어도 보고 해서 결국엔 잘 구현해낸 것 같다.

송용승 : 가장 어려웠던 부분은 역시 Bus error를 해결하는 것이었다. 여러 개의 디바이스가 동시에 작동하는것이 우리가 제안한 시스템의 특징인데, 페리보드가 이런 부하를 견디지 못하는 것이었다. 이같은 문제를 해결하기 위해 쓰레딩으로 버스에 가해지는 부하를 조절하려고 해 봤지만 우리가 사용할 수 있는 리소스가

얼마나 되는지 모르는 상태에서 스케줄링을 하는 것은 다소 답답한 작업이었다. 그래도 팀원들이 모두 실습 수업을 열심히 진행한 덕분에 디바이스에 대한 이해도가 높아서 좋은 아이디어가 많이 나왔고 일정에 큰 차질 없이 문제를 해결할 수 있었다.

전동훈 : 아두이노와 임베디드 보드간의 통신을 담당하였는데 처음에 wifi모듈로 사용한 ESP8266이 가장 큰 어려움을 주었다. 펌웨어 설치까지는 진행하였으나 그 뒤 예제파일 실행에서 1주일 이상을 매달렸으나 실패하였고 ESP8266대신 wifi shield를 사용하는게 되었다. wifi sheild를 이용하여서 아두이노에서 wifi에 접속하는 과정은 순탄하였다. 그 뒤에 임베디드 보드를 서버로 사용하는것과 아두이노를 장착한 RC카를 클라이언트로한 둘 사이의 통신은 수월하게 진행되었다. 다만 시연 당일 아두이노 보드와 RC카의 통신이 원활이 진행되지 못했던 것이 아쉽다.

8. 조원 별 역할 분담 내용, 일정

- 조원 별 역할 분담

구분	임베디드 보드	RC카(아두이노)	RC카<->보드 통신
강동훈	보조	메인	보조
최정용	메인	보조	보조
송용승	메인	보조	보조
전동훈	보조	보조	메인

- 일정

구분		9월	10월				11월				12월		
		준비기간	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주
RC카	센서 테스트												
	센서 통합 구현												
	RC카 제작												
	RC카 보드 연동												
보드	디바이스 드라이버 제작 및 테스트												
	페리보드 기능 통합 구현												
	LCD 모듈 구현												
	LCD 터치 구현												
	RC카 보드 연동												
통신	통신 기능 구현 및 테스트												
	RC카 보드 연동												
통합	최종 연동 및 디버깅												