

Handox

캡스톤 디자인 발표

4 팀 배달해조

2018-06-25

Contents

1. 서론	2
1.1. 설계 배경	2
1.2 설계 목적	3
1.3 팀원 소개 및 역할 분담	3
1.4 프로젝트 진행 과정	4
 2. 제품설명	 5
2.1 제품 필요성	5
2.2 시연 영상	6
2.3 시스템 구성과 기능 - 프로젝트 시험 방법 및 절차	7
2.4 제품 사용법	8
 3. 기능 및 동작	 9
3.1 기능 순서도	9
3.2 Server 및 Client의 기능	9
 4. 기대 효과.....	 11
5. 발전 동향.....	11
6. 참고 사이트	12
7. 부록	13

1. 서론

본 보고서는 Raspberry Pi와 Arduino의 Serial 통신과 PC와 Raspberry Pi의 Socket 통신을 이용한 일반적인 택배 서비스에 비해 더 안전하고 편리한 무인 택배함 시스템에 대한 것이다. 하드웨어와 소프트웨어를 적절히 사용하여 최대한 상용화 될 수 있도록 구현하였다. 아래 내용들은 Handox의 설계 목적, 구성 및 기능 등을 소개하였다.

1.1. 설계 배경

설연휴 현관 앞 택배 도난 비상... "경비실·보관함 이용해야"

조영아 | 기사입력 2018-02-12 20:49 | 최종수정 2018-02-12 21:20

택배 물 망고 도난



경제 IMF 한국 지하경제 규모 GDP 대비 19.8%

◀ 영커 ▶

설 명절을 앞두고 선물로 온 택배를 현관 앞에 두는 경우 조심하셔야겠습니다.


이걸 상습적으로 훔쳐 온 용의자가 붙잡혔습니다.

조영아 기자가 보도합니다.

[머니포커S] 택배, 경비실에 맡겨도 되나

최준신 기자 입력 : 2017-04-23 07:15

기사공유



기자포사간=이미지투데이

#. 대단지아파트와 오피스텔 위주의 주택가 지역을 담당하는 택배기사 A씨는 고객을 직접 대면하는 일이 거의 없다. 수령인에 기재된 전화번호로 연락을 하면 대부분의 고객이 '경비실에 맡겨달라'고 말한다. 이런 경우 출을 몰라 직접 배송하는 시간과 수고를 덜 수 있어 편하지만 이따금 택배물품이 분실되거나 파손된 경우 책임소재 때문에 난감해진다.

이는 공동주택 지역을 담당하는 택배기사들이 공통적으로 겪는 딜레마다. 1인가구가 보편화되며 택배가 올 시간에 집을 지키고 있는 사람이 전무할뿐더러 집에 있더라도 무거운 물건이 아닌 이상 '경비실에 맡겨달라'는 요구를 하는 사람들이 늘어나서다.

일부 고급주택의 경우 **출입증**을 필요로 하거나 컴퓨터 검색 절차가 과도히 복잡해 1, 2분이 소중한 택배 기사들이 사실상 들어갈 수 없는 경우도 허다하다. 몇몇 기사들은 택배를 모두 보안업체 직원에게 맡기고 수량을 계산해 일정수수료를 월말에 청산하기도 한다.

문제는 사고 발생시 책임소재다. 대부분의 경비실에서 주민들의 택배를 받아주고는 있지만 분실 등에 대한 책임지지 않아서다. 미리 연락을 하고 경비실에 **보관**한 증거기록을 남겼음에도 택배기사가 보상을 해야했다는 민원도 부지기수다.

일반적으로 택배를 수령할 시에 부재중인 경우가 많아 집 앞에 택배를 두고 가는 경우가 많다. 하지만 최근 기사를 살펴보면 택배를 현관앞에 두었을 시 분실 위험이 높고, 경비실에 택배를 맡길 경우에도 경비원의 업무부담을 가중시키고 택배 사고의 책임소재가 경비원에게 전가될 수 있다는 우려가 나올 수 있다. 이런 문제를 해결하기 위한 방안으로 많은 사람들이 무인택배보관함 등의 보급이 시급하다는 입을 모은다. 또한, 일반 주택에 거주하는 사람들은 경비실이 없고, 부재중일 경우 어쩔 수 없이 집 앞에 택배를 놓아야 할 수 밖에 없기에 무인택배함 같은 대책이 필요하다.

1.2 설계 목적

기존의 무인택배함은 일반적으로 공용주택에서 많이 사용 되고 있다. 다세대 주택이 아닌 반지하나 옥탑방이 있는 일반 주택에 사는 사람들은 무인택배함 시설을 이용하기에 가격이 너무 부담이 된다는 단점이 있다. 또한 최근에는 낯선 사람에게 자신의 개인 정보를 노출 시켜야 한다는 불안감이 커지고 있기 때문에, 택배 수령 주소를 공공시설로 하고 무인 택배함을 공공시설에 설치하는 등의 방법으로 택배 서비스의 효율성을 높일 수 있다.

1.3 팀원 소개 및 역할 분담

역할	이름	맡은 분야
팀장	이영성	Server 및 DB 구축, 택배함 제작
팀원	이태희	
팀원	김정민	Server 함수 구현, 회의록 작성
팀원	유진솔	
팀원	배윤희	Raspberry & Arduino 코드 구현 및 통신
팀원	전은표	

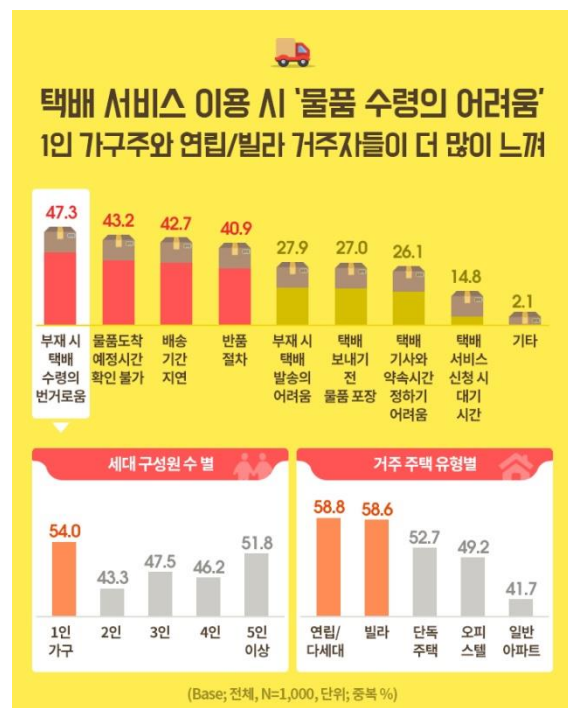
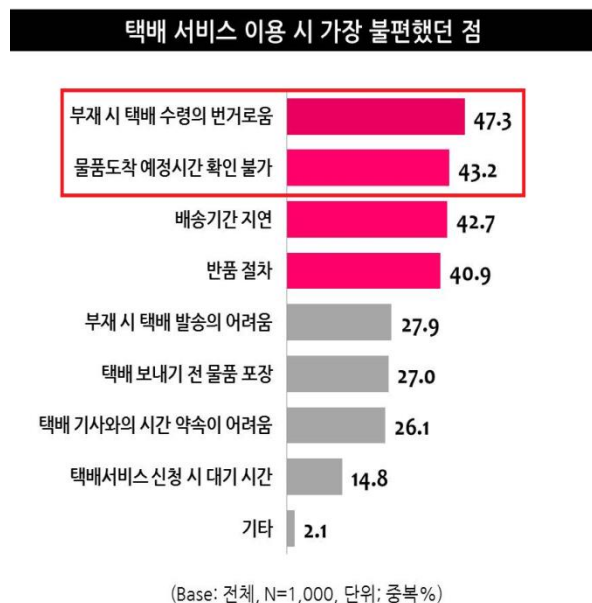
1.4 프로젝트 진행 과정

개발내용		개발자	6/1	6/2	6/4	6/7	6/8	6/9	6/11	6/22	6/23	6/24
제품 구상 및 기획	아이디어 수정 및 개발 방향	전체										
시스템 개발 및 구축	Server 구축	이영성 이태희										
	DB 구축	이영성 김정민 유진솔										
	Arduino 와 Keypad, LCD 연동	배윤희 전은표										
	Raspberry Pi (Client와 연동)	배윤희 전은표										
1차 점검	실제 시연을 위한 사물함 제작 및 1차 데모 시연	이태희 이영성										
	보고서 작성	김정민 유진솔										
최종 점검	데모 시연	전체										
	최종보고서 작성	전체										

2. 제품설명

Handox는 편리하다는 의미의 'Handy' 와 택배함의 'Box' 를 조합한 단어로, 택배함을 만듦으로써 택배 기사와 수취인 사이의 불필요한 만남을 없애고 택배 도착 시간을 정확하게 알려주는 데 도움을 주는 서비스를 제공한다.

2.1 제품 필요성



택배 서비스 이용 시 불편했던 점을 설문조사 한 결과 '부재 시 택배 수량의 번거로움' 과 '물품도착 예정시간 확인 불가' 항목의 비율이 가장 높았다. 택배 전달 시에 부재 중일 경우 택배 기사들은 택배를 어디에 두고 가야 하는지 또는 다음에 다시 와야 하는지에 대해 고민하는 상황이 발생할 수 있고 마찬가지로 수취인도 귀중품일 경우나 중요한 택배일 경우에는 더 문제가 될 수 있다.

또한, 수취인은 택배가 언제 도착하는지를 정확히 알 수 없으므로 빨리 수령해야 하는 택배일 경우에는 수령하는 데 어려움이 발생할 수 있다.

이 뿐만 아니라, 특히 1인 가구나 연립/빌라 거주자들은 따로 경비실이나 관리실 등 택배 위탁장소에 제한이 있으며 혼자 사는 경우에는 택배를 직접 집으로 배송 받을 때 택배 기사가 아닌 낯선 사람에 대한 불안함이 커질 수 있다.

이러한 문제에 대해 Handox는 부재 시에도 택배를 효과적으로 수령할 수 있도록 하고 택배 기사가 택배함에 물품을 넣고 배송을 완료할 경우 문자 메시지를 전송하여 택배 도착시간을 알 수 있게 하였다.

2.2 시연 영상

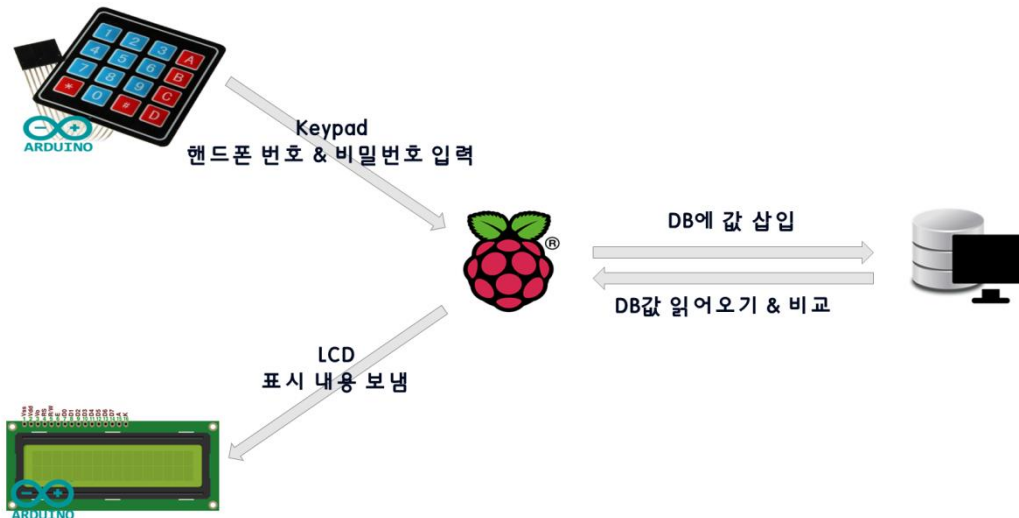
<https://youtu.be/WMnzuKIoL7A>

[DB 값]

	BOXINDEX ▾	STATUS	PASSWD	PHONE
1	1	FULL	6049	010406736521
2	2	EMPTY	0	000000000000

2.3 시스템 구성과 기능

- 프로젝트 시험 방법 및 절차



- 하드웨어 구성과 기능

① Raspberry Pi(MicroProcessor)

Raspberry Pi는 초소형, 초저가 PC로 싱글보드 컴퓨터라고 불린다. 보드의 Microprocessor는 음향, 영상, USB, Ethernet 및 HDMI까지 지원한다. Arduino와 다르게 Raspberry Pi에서는 프로그램을 써서 하드웨어를 직접 제어하지 않고, 운영체제가 깔려 있어서 운영체제 내에서 프로그래밍을 한다. 본 제품, Handox는 Raspberry Pi와 Arduino의 Serial통신을 이용하여 Raspberry Pi 내에서의 프로그래밍으로 구현하였다.

② Arduino (MicroController)

Arduino는 일반적으로 Arduino Uno를 쓰고 있다. 램은 2K, 플래시메모리는 32K 정도이고 타이머와 Serial, I2C, SPI 등이 있다. 외부에는 전압 조정기, 수동 회로 요소, 입출력 연결단자로 구성되어 있다. Arduino는 상대적으로 배우기 쉬운 디자인과 더 쉬운 소프트웨어로 구성된다. Arduino IDE에서 작성된 프로그램은 칩에서 사용하는 유일한 프로그램으로 인터프리터도 따로 필요 없고, 운영체제도, 펌웨어도 필요 없다. C코드는 기계어로 컴파일되어 칩에 실리게 되며, Arduino에서 작동된다.

Arduino, Raspberry Pi 모두 명령을 수행하는 중앙처리장치(CPU)와 타이머, 메모리(기억 장치), 입출력 핀들이 들어 있다. 하지만 가장 큰 차이점은 입출력 핀에 있는데, MicroController는 입출력에 강점을 가지고 있기 때문에 본 제품에서는 외부 기기인 키패드와 lcd, 서보 모터를 직접적으로 제어한다. 그러나 MicroController는 입출력 부분이 약한 경향이 있어서 트랜지스터가 외부 기기를 다룬다. 실제 스펙을 비교하면 Raspberry Pi가 Arduino보다 좋아 보이지만 앞서 설명한 차이점들을 놓고 보면 각각 어떤 기능에 적합한지가 다르다. 가령, 단순한 모터 동작, 글자 표시하는 LCD, 센서 제어와 같이 제어 중심의 기능만을 이용한다면 Arduino로 제어하는 것이 적합하며 비디오, 카메라, 복잡한 수치계산, 그래픽 처리와 같은 복잡한 연산이나 처리가 필요하다면 Raspberry Pi로 제어하는 것이 더 적합할 것이다.

두 가지 모두 전 세계적으로 많이 사용되는 보드이나 하나의 보드가 모든 프로젝트에 완벽하다고는 볼 수 없다. 단지, 제어 중심이라면 Arduino를, 데이터 처리에는 Raspberry Pi를 이용하는 것이 더 효율적이라고 본다.

2.4 제품 사용법

- 택배 기사님일 경우

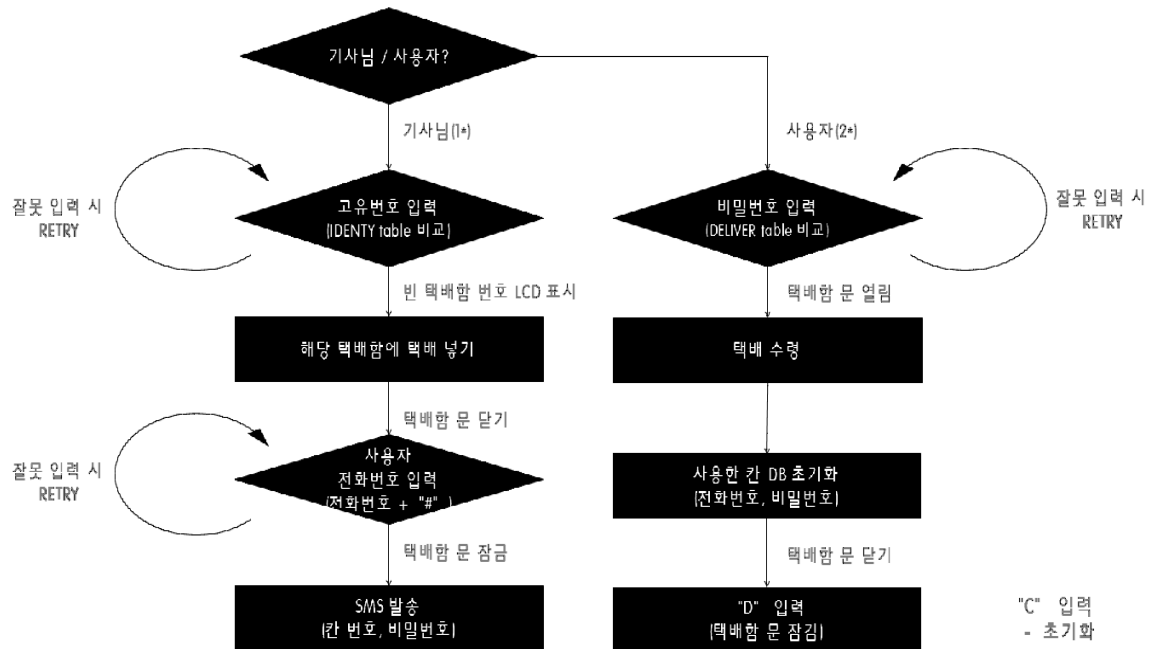
- ① 키패드에 1*을 입력한다.
- ② 기사님임을 확인하는 고유번호를 입력한다.
- ③ 택배함이 하나 열리면 안에 택배를 넣고 닫는다.
- ④ 키패드에 수령인의 핸드폰 번호 + #을 입력한다.
- ⑤ 택배함이 잠기고 수령인에게 문자가 발송된다.

- 수취인일 경우

- ① 키패드에 2*을 입력한다.
- ② 키패드에 문자로 받은 비밀번호를 입력한다.
- ③ 비밀번호에 대응되는 택배함이 열린다.
- ④ 택배를 수령한다.

3. 기능 및 동작

3.1 기능 순서도



3.2 Server 및 Client의 기능

① Server

: Raspberry Pi에서 socket 통신으로 Data를 넘겨 주면 (택배 기사님의 고유번호, 휴대폰 번호, 비밀번호 등) Server는 Data를 DB에 넣어 관리한다. 이때, 택배 기사님의 고유번호가 DB에 등록된 값과 같으면 신원이 확인된 택배 기사이기 때문에 다음 단계로 넘어 갈 수 있게 상황을 Server가 control한다. 다음 단계에서 택배 기사가 수취인의 휴대폰 번호를 입력하면 그 번호를 DB에 저장하고 마지막으로 확인 message(#) 가 들어오면 택배함 칸을 잠그는 message를 socket 통신으로 Raspberry Pi에 보내고 DB의 상태를 바꿔주며 동시에 사용자에게 문자 메시지를 전송한다. (칸 번호와 비밀번호에 관한 정보 전송)

택배 기사가 아닌 사용자가 이용할 경우 사용자가 입력한 비밀번호를 Raspberry Pi가 socket 통신으로 Server에 보내면, Server는 입력 받은 비밀번호를 DB에 있는 비밀번호 값과 비교한다. 같은 비밀번호가 있는 칸을 찾아 그 칸을 열도록 Raspberry Pi에 칸 번호를 보내준다. 칸 번호를 보내준 후 Server는 모든 절차가 완료 되었으므로 DB에 있는 사용자와 관련한 정보를 삭제한다.

②Client

: Arduino에서 화면을 띄워 택배기사인지 사용자인지 확인하는 절차를 거쳐 각각의 경우에 대해 다른 화면을 띄운다.

택배기사일 경우 고유 번호를 입력 받고 Serial 통신으로 Raspberry Pi에게 값을 보내준다. Raspberry Pi에서 신원이 확인된 택배 기사인지 아닌지 값을 받아서 확인 되었다면 다음 단계로 넘어간다. 다음 단계에서 Arduino는 Raspberry Pi로부터 받은 칸 번호를 열고 난 뒤 핸드폰 번호를 입력 받고 또 Serial 통신으로 Raspberry Pi에게 값을 보내준다. 마지막으로 확인 message (#) 을 입력 받으면 그 message를 Raspberry Pi에게 보내준다.

사용자일 경우 문자 메시지로 받은 비밀번호를 Arduino에 입력하면 그 값을 Serial 통신으로 Raspberry Pi에 보내준다. 비밀번호에 해당하는 칸 번호를 Raspberry Pi에서 Arduino로 보내주면 그 칸을 열어준다.

4. 기대 효과

본 제품은 기존 택배 서비스에 대한 이용자들의 불편한 점, 가령 부재 시에 택배를 어디에 두어야 하는지 또는 혼자 사는 거주자들의 보안 문제나 택배 도착 시간 알림 등을 고려하여 고안해낸 제품이다. 또한, 수취인이 택배를 수령할 시에 꼭 필요한 비밀번호를 무작위 하게 발생함으로써 이미 상용화 되어 있는 무인 택배 시스템의 보안 또한 더 강화하여 택배 분실 문제도 예방할 수 있을 것이다. 기존의 무인 택배 시스템은 수취인의 핸드폰 번호나 송장번호만 입력하면 누구나 택배함의 문을 열 수 있었으나, Handox는 비밀번호를 무작위 하게 발생시켜 직접 수취인의 핸드폰으로 전송해 주기 때문에 범죄나 택배 분실 위험이 더욱 감소한다.

5. 발전 동향

▶ 잠금 장치 개선

: 사용자에게 좀 더 튼튼한 잠금 장치 기능을 제공하기 위하여 도어락을 이용한 잠금 장치 서비스를 추후에 제공 할 예정이다.

▶ 원자재 교체

보안을 위해 외부 충격으로부터 안전한 원자재로 교체할 예정이다.

▶ 무인 택배함에 택배 보내기 기능 추가

: 추후에 택배 회사와 연동하여 사용자가 택배를 보낼 때 무인 택배함에 택배를 넣으면 기사가 택배를 수령하여 보낼 수 있는 기능을 추가한다.

6. 참고 사이트

기능	참고 링크	내용
SQLite	https://www.python-course.eu/sql_python.php	DB구축, 값 삽입
Server	https://docs.python.org/3/howto/sockets.html	Socket통신
Raspberry Pi & Arduino	https://www.youtube.com/watch?v=GteMrHri6r8	Keypad와 LCD의 Arduino 통신
하드웨어 구성 및 기능	http://andrew0409.tistory.com/82	Arduino, Raspberry Pi의 기능
문자 메시지	https://www.coolsms.co.kr/	문자 메시지 발송 사이트
기사 및 설문 출처	http://moneys.mt.co.kr/news/mwView.php?type=1&no=2017042019318067677&outlink=1 http://imnews.imbc.com/replay/2018/nwdesk/article/4526669_22663.html https://www.trendmonitor.co.kr/tmweb/trend/allTrend/detail.do?bidx=1425&code=0201&trendType=CKOREA (설문조사)	택배 분실 기사 및 무인 택배함 기사, 설문조사 출처

7. 부록

- server.py

```
import os
from socket import socket, AF_INET, SOCK_STREAM
import threading, logging
import random
import sqlite3

class Server:

    def __init__(self, my_port):
        self.sock = socket(AF_INET, SOCK_STREAM)
        self.sock.bind(('', my_port))
        self.sock.listen(5)
        logging.info('Server started')

        # do forever (until process killed)
        while True:
            self.conn, self.cli_addr = self.sock.accept()
            logging.info('Connected by {}'.format(self.cli_addr))
            self.handler = threading.Thread(target=self.echo_handler,
                                             args=(self.conn, self.cli_addr))

            self.handler.daemon = True
            self.handler.start()

    def echo_handler(self, conn, cli_addr):
        try:
            while True:
                self.data = self.conn.recv(1024)
                # recv next message on connected socket

                if not self.data: break
```

```

print('Server received', self.data.decode())
self.r_msg = self.data.decode()
self.msg_list = self.r_msg.split()

"""택배기사가 id 를 입력하여 보내 줬을 때"""
if self.msg_list[0] == 'type:identify':
    if self.CheckID(self.msg_list[2]):
        self.num = self.SendBoxnum()
        self.conn.send(('IDOK ' + str(self.num)).encode())

    else:
        self.conn.send("NOTOK".encode())

"""택배기사가 전화번호를 입력하여 보내줬을 때"""
if self.msg_list[0] == 'type:phone':
    # STATUS, 입력받은 전화번호와 랜덤으로 만들어진
    # 비밀번호를 DB 에 저장하고 핸드폰에 문자를 보냄
    self.pswd = self.makepasswd(self.num)
    self.Storephone(self.msg_list[2], self.num)
    self.SMS(self.msg_list[2], self.num, str(self.pswd))
    self.conn.send('phoneok'.encode())

"""사용자가 비밀번호를 입력하여 보내줬을 때"""
if self.msg_list[0] == 'type:passwd':
    self.boxnum = self.CheckWhichBox(self.msg_list[2])

    if self.boxnum != 0:
        self.conn.send(("passok " +
str(self.boxnum)).encode('utf-8'))
        self.Cleandb(self.boxnum)

    else:
        self.conn.send("passnot".encode('utf-8'))

```

```

# socket.error exception
except OSError as e:
    logging.info('socket error: {}'.format(e))

except Exception as e:
    logging.info('Exception: {}'.format(e))

else: # This client normally terminated.
    logging('Client closed {}'.format(self.cli_addr))

finally:
    self.conn.close()

# 비밀번호 만드는 함수
def makepasswd(self, boxindex):
    count = 0
    dbconn = sqlite3.connect('example.db')
    c = dbconn.cursor()
    passwd = random.randrange(1000, 10000)
    loop = True

    while loop:
        for row in c.execute('SELECT * FROM DELIVER'):
            if passwd == row[2]:
                count += 1

            if count == 0:
                loop = False

        count = 0
        passwd = random.randrange(1000, 10000)

```



```

        c.execute("UPDATE DELIVER SET PASSWD={0} WHERE
BOXINDEX={1}".format(passwd, boxindex))

        dbconn.commit()

        dbconn.close()

    return passwd

"""기사님 일 때 사용하는 함수들"""
# 클라이언트가 기사님일 때 빈 택배함 번호 알려주는 함수
def SendBoxnum(self):
    dbconn = sqlite3.connect('example.db')
    c = dbconn.cursor()
    start = True

    for row in c.execute("SELECT * FROM DELIVER"):
        if row[1] == 'EMPTY' and start == True:
            status = 'FULL'
            start = False
            c.execute("UPDATE DELIVER SET STATUS=? WHERE
BOXINDEX=?", (status, row[0]))
            dbconn.commit()
            dbconn.close()

            return row[0]

    else:
        return "FULL"

# 택배기사가 ID 를 입력할 경우 DB 에 있는지 비교
def CheckID(self, id):
    dbconn = sqlite3.connect('example.db')
    c = dbconn.cursor()
    for row in c.execute('SELECT * FROM IDENTITY'):

```

```

        if row[0] == id:
            dbconn.close()
            return True

    dbconn.close()

    return False

# 택배기사가 전화번호를 입력하면 DB 에 전화번호 저장
def Storephone(self, phonenumber, num):
    dbconn = sqlite3.connect('example.db')
    c = dbconn.cursor()

    for row in c.execute('SELECT * FROM DELIVER'):
        if row[0] == num:
            c.execute("UPDATE DELIVER SET PHONE=? WHERE
BOXINDEX=?", (phonenumber, row[0]))
            dbconn.commit()

    dbconn.close()

# SMS 보내기
def SMS(self, phonenumber, num, pswd):
    os.system("python sendsms.py {0} {1} {2}".format(phonenumber, num,
pswd))

"""사용자 일 때 사용하는 함수들"""
# 사용자가 비밀번호 입력하면 비밀번호에 맞는 택배함 번호 찾기
def CheckWhichBox(self,passwd):
    dbconn = sqlite3.connect('example.db')
    c = dbconn.cursor()
    passwd = int(passwd)
    for row in c.execute('SELECT * FROM DELIVER'):

```

```

        if row[2] == passwd:
            return row[0]

    return 0

# 사용자가 택배함을 사용후 택배함 칸에 맞는 DB 값 초기화
def Cleandb(self, boxnum):
    dbconn = sqlite3.connect('example.db')
    c = dbconn.cursor()

    for row in c.execute('SELECT * FROM DELIVER'):
        if row[0] == boxnum :
            originphone = '000000000000'
            originpasswd = 0000
            originstatus = 'EMPTY'
            c.execute("UPDATE DELIVER SET PHONE=? WHERE
BOXINDEX=?", (originphone, row[0]))
            c.execute("UPDATE DELIVER SET PASSWD=? WHERE
BOXINDEX=?", (originpasswd, row[0]))
            c.execute("UPDATE DELIVER SET STATUS=? WHERE
BOXINDEX=?", (originstatus, row[0]))
            dbconn.commit()

    dbconn.close()

if __name__ == '__main__':
    server = Server(50013)

```

- client.py

```
import socket
import serial
import time

class Client:

    def __init__(self, server_addr):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #
        make TCP/IP object
        self.sock.connect(server_addr)
        self.server_addr = server_addr

        self.port = '/dev/ttyACM1' # keypadlcd
        self.port2 = '/dev/ttyACM2' # box 1
        self.port3 = '/dev/ttyACM0' # box 2

        """시리얼 지정"""
        self.ser = serial.Serial(self.port, 9600)
        self.ser1 = serial.Serial(self.port2, 9600)
        self.ser2 = serial.Serial(self.port3, 9600)
        self.ser.flushInput()
        self.ser1.flushInput()
        self.ser2.flushInput()

        """택배함 번호 지정"""
        self.ser1.write('Z'.encode('utf-8'))
        self.ser2.write('Z'.encode('utf-8'))
        self.ser1index = self.ser1.readline().decode()[0]
        self.ser2index = self.ser2.readline().decode()[0]

        if self.ser1index > self.ser2index:
```

```

        self.ser1, self.ser2 = self.ser2, self.ser1
    self.echo_client()

def echo_client(self):
    """Serial Port"""
    self.keypad_port = self.port
    self.serial_keypad_duino = serial.Serial(self.keypad_port, 9700)
    self.serial_keypad_duino.flushInput()
    self.message = ""

    """variable"""
    self.postman = False
    self.user = False
    self.phone_postman = False
    self.passwd_user = False
    self.phone_check = False

    print('conneted: ', self.sock.getpeername())

    while True:
        self.serialmsg = self.serial_keypad_duino.readline().decode()[0] #
        키패드에서 입력받은 캐릭터를 저장
        self.message = self.message + self.serialmsg # 키패드에서 입력받은
        캐릭터를 문자열에 저장

        if self.serialmsg == '*':
            print(self.message[:-1])
            self.content = self.message[:-1]
            self.message = ""

            # 택배기사
            if self.content == '1':

```

```

        self.ser.write('B'.encode('utf-8'))
        self.postman = True

    # 사용자
    elif self.content == '2':
        self.ser.write('C'.encode('utf-8'))
        self.user = True

    # 그 외에 입력
    elif self.content in list(range(3,10)):
        self.ser.write('A'.encode('utf-8'))

    # 택배기사 ID 입력
    elif self.postman == True:
        self.msg = 'type:identify\n' + 'id: ' + self.content
        self.sock.send(self.msg.encode('utf-8'))
        self.data = self.sock.recv(1024).decode('utf-8')
        print(self.data)
        self.data_list = self.data.split()

    # DB 에 ID 가 있을경우
    if self.data_list[0] == "IDOK":

        # LCD 에 핸드폰 입력 화면 출력
        self.ser.write("E".encode('utf-8'))

        # BOXINDEX 가 1 일경우
        if self.data_list[1] == '1':
            self.ser.write(self.data_list[1].encode('utf-8'))
            self.serialindex = self.ser1
            self.serialindex.write('O'.encode('utf-8'))
            time.sleep(5)
            self.ser.write('E'.encode('utf-8'))

```

```

        self.phone_postman = True

        # BOXINDEX 가 2 일경우
        elif self.data_list[1] == '2':
            self.ser.write(self.data_list[1].encode('utf-8'))
            self.serialindex = self.ser2
            self.serialindex.write('O'.encode('utf-8'))
            time.sleep(5)
            self.ser.write('E'.encode('utf-8'))
            self.phone_postman = True

        # 택배함을 모두 사용 중일 경우
        elif self.data_list[1] == 'FULL':
            self.ser.write('I'.encode('utf-8'))
            self.postman = False

        # DB 에 ID 가 없을 경우
        elif self.data == "NOTOK":
            self.ser.write("D".encode())

    elif self.serialmsg == '#':
        print(self.message[:-1])
        self.content = self.message[:-1]
        self.message = ""

        # 사용자 일 때 비밀번호 입력
        if self.user == True:
            self.msg = 'type:passwd\n' + "passwd: " + self.content
            self.sock.send(self.msg.encode('utf-8'))
            self.data = self.sock.recv(1024).decode('utf-8')
            self.data_list = self.data.split()

        # 패스워드가 DB 에 있을 경우

```

```

if self.data_list[0] == 'passok' :

    # BOXINDEX 가 1 일경우
    if self.data_list[1] == '1' :
        self.serialindex = self.ser1
        self.serialindex.write('O'.encode('utf-8'))
        self.ser.write('F'.encode('utf-8'))
        self.passwd_user = True

    # BOXINDEX 가 2 일경우
    elif self.data_list[1] == '2':
        self.serialindex=self.ser2
        self.serialindex.write('O'.encode('utf-8'))
        self.ser.write('F'.encode('utf-8'))
        self.passwd_user = True

    # DB 에 패스워드가 없을 경우
    elif self.data == 'passnot':
        self.ser.write('G'.encode('utf-8'))

    # 번호의 자리수가 11 개가 아닐 경우
    elif self.postman == True and self.phone_postman == True:

        # 번호의 자리수가 11 개가 아닐 경우
        if len(self.content) != 11:
            print(self.content)
            self.ser.write('H'.encode('utf-8'))

        # 번호의 자리수가 11 개일 경우
        elif len(self.content) == 11:
            print('content is ' + self.content)
            self.phone_postman = False
            self.postman = False

```



```

        self.msg = 'type:phone\n' + "phone: " + self.content
        self.sock.send(self.msg.encode('utf-8'))
        self.data = self.sock.recv(1024).decode('utf-8')

        # 서버에서 DB 에 전화번호를 저장했다는 신호가 올 경우
        if self.data == 'phoneok':
            self.ser.write('A'.encode('utf-8'))
            self.serialindex.write('S'.encode('utf-8'))
            self.phone_check = False

    # 키패드 입력을 C 를 누를경우 (초기화)
    elif self.serialmsg == 'C':
        self.message = ""
        self.ser.write('A'.encode('utf-8'))
        self.user = False
        self.postman = False
        self.phone_postman = False
        self.phone_check = False
        self.passwd_user = False

    # 사용자가 택배함에서 물건을 수령후 잠글 경우
    elif self.serialmsg == 'D' and self.passwd_user == True:
        self.message = ""
        self.ser.write('A'.encode('utf-8'))
        self.serialindex.write('S'.encode('utf-8'))
        self.user = False
        self.phone_postman = False

    self.sock.close() # close socket to send eof to server

if __name__ == '__main__':
    client = Client(('35.194.97.11', 50013))

```

- sendsms.py

```
# -*- coding: utf8 -*-

"""
vi:set et ts=4 fenc=utf8:
Copyright (C) 2008-2010 D&SOFT
http://open.coolsms.co.kr
"""

import sys
import coolsms

def main():
    cs = coolsms.sms()
    var1 = sys.argv[1]
    var2 = str(sys.argv[2])
    var3 = str(sys.argv[3])

    # 프로그램명과 버전을 입력합니다. (생략가능)
    cs.appversion("TEST/1.0")

    # 자원 인코딩: euckr, utf8
    cs.charset("utf8")

    # 아이디와 패스워드를 입력합니다.
    cs.setuser("lyst95", "tzc12yt34")

    cs.addsms(var1, "01035419130", "{0} 칸 비밀번호는 {1}
입니다.".format(var2,var3))

    nsent = 0
```

```

if cs.connect():
    # add 된 모든 메시지를 서버로 보냅니다.
    nsent = cs.send()
else:
    # 오류처리
    print "서버에 접속할 수 없습니다. 네트워크 상태를 확인하세요."

# 연결 해제
cs.disconnect()

# 결과를 출력합니다.
print "%d 개를 전송한 결과입니다." % nsent
cs.printr()

# 메모리 초기화
cs.emptyall()

if __name__ == "__main__":
    main()
    sys.exit(0)

```

- Arduino

[keypad_LCD]

```
#include <LiquidCrystal.h> // include LCD library (standard library)
#include <Keypad.h> // include keypad library

// number of the keypad's rows and columns
const byte rows = 4;
const byte cols = 4;

// define the symbols on the buttons of the keypad
char keyMap [rows] [cols] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'+', '0', '#', 'D'}
};

//pins of the keypad
byte rowPins [rows] = {0, 2, 3, 4};
byte colPins [cols] = {5, 6, 7, 8};

Keypad myKeypad = Keypad(makeKeymap(keyMap), rowPins, colPins, rows, cols);

// pins of the LCD. (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd (A0, A1, A2, A3, A4, A5);

void setup(){
  lcd.begin(16, 2); // 16 * 2 LCD module
  Serial.begin(9700);
}

// request only once open
char condition = 'A';
boolean view = false;

void loop(){
  char presskey = myKeypad.getKey(); //define which key is pressed with getKey

  if(presskey != 0){
    Serial.println(presskey);
  }

  // LCD 초기 화면
  if(condition == 'A'){
    lcd.setCursor(0, 0); // fisrt line
    lcd.print("Delivery -> 1*");
    lcd.setCursor(0, 1); // second line
    lcd.print("Recipient -> 2*");
  }

  if(Serial.available()){
    char in_data;
    in_data = Serial.read();
  }
}
```

```

// 택배기사(1*)를 선택했을 때의 화면
if(in_data == 'B'){
    condition = 'B';
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" Input your ID ");
    lcd.setCursor(0, 1);
    lcd.print(" then Enter :+ ");
    view = true;
}

// 사용자(2*)를 선택했을 때의 화면
else if(in_data == 'C'){
    condition = 'C';
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" Input your passwd ");
    lcd.setCursor(0, 1);
    lcd.print(" then Enter :# ");
}

// DB에 택배기사 고유번호(ID)가 없을 경우
else if(in_data == 'D'){
    condition = 'D';
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("NO ID RETRY");
    lcd.setCursor(0, 1);
    lcd.print(" then Enter :# ");
}

// 초기화면
else if(in_data == 'A'){
    condition = 'A';
    lcd.clear();
    lcd.setCursor(0, 0); // fisrt line
    lcd.print("Delivery -> 1*");
    lcd.setCursor(0, 1); // second line
    lcd.print("Recipient -> 2*");
}

// 택배기사 ID가 확인 되었을 때 사용자의 핸드폰 번호 입력
else if(in_data == 'E'){
    condition = 'E';
    lcd.clear();
    lcd.setCursor(0, 0); // fisrt line
    lcd.print("input phone num");
    lcd.setCursor(0, 1); // second line
    lcd.print("then enter : #");
}

// 사용자가 입력한 비밀번호가 DB에 존재했을 경우
else if(in_data == 'F'){
    condition = 'F';
    lcd.clear();
    lcd.setCursor(0, 0); // fisrt line
    lcd.print("AFTER TAKE");
    lcd.setCursor(0, 1); // second line
    lcd.print("then enter : D");
}

```

```

// 전화번호 확인
else if(in_data == 'H'){
    condition = 'H';
    lcd.clear();
    lcd.setCursor(0, 0); // fisrt line
    lcd.print("RETRY PHONE NUM");
    lcd.setCursor(0, 1); // second line
    lcd.print("then enter : #");
}

// 모든 BOX STATUS가 FULL일 경우
else if(in_data == 'I'){
    condition = 'I';
    lcd.clear();
    lcd.setCursor(0, 0); // fisrt line
    lcd.print("ALL BOXES USED");
    lcd.setCursor(0, 1); // second line
    lcd.print("then enter : C");
}

// 택배기사 ID가 확인 되었을 때 이용할 택배 칸
else{
    condition = 'J';
    lcd.clear();
    lcd.setCursor(0, 0); // fisrt line
    lcd.print("USE BOX NUM:");
    lcd.setCursor(13,0);
    lcd.print(in_data);
    lcd.setCursor(0, 1); // second line
    lcd.print("WAIT 5 SECS");
    view = false;
}
}
}

```

[servo_motor]

```
servo_motor$  
#include<Servo.h>  
Servo servo;  
int boxindex = 1;  
  
void setup(){  
  servo.attach(7);  
  Serial.begin(9600);  
  servo.write(100);  
}  
  
void loop(){  
  if(Serial.available()){  
    char in_data;  
    in_data = Serial.read();  
  
    // 택배함 번호 할당  
    if(in_data=='Z'){  
      Serial.println(boxindex);  
    }  
  
    // 잠금  
    if(in_data=='S'){  
      servo.write(100);  
    }  
  
    // 열림  
    if(in_data == 'O'){  
      servo.write(0);  
    }  
  }  
}
```