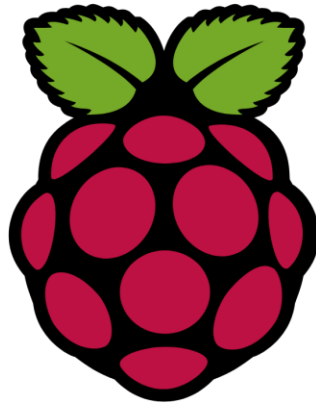


# Ezi-SERVOII-Plus-E

## < 라즈베리파이 편 >



2023년 07월 28일

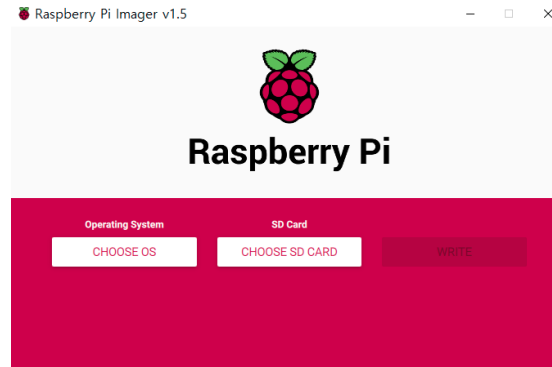
신민호

김시연

성우진



# • 라즈비안 설치



<https://www.raspberrypi.com/software/>에서 Raspberry Pi Imager 을 다운로드

Imager에서 사용할 운영체제 Raspberry PI OS를 선택  
설치할 SD카드를 선택하고 Write버튼을 눌러 라즈베리파이 OS 설치

라즈베리 파이 OS를 설치하고 컴퓨터에서 sd카드를 분리한 후

라즈베리파이에 삽입

라즈베리 파이에 전원과 모니터 키보드 마우스 등을 연결하고 전원을 켜

Welcome to Raspberry Pi 창에서 next

국가와 언어 timezone을 한국으로 선택 후 next

사용자의 이름과 비밀번호를 작성함 ex)id: fastech pw:fastech

# • 라즈베리파이 초기설정

초기에 마우스와 키보드의 반응속도가 느림

터미널에서 `sudo nano /boot/cmdline.txt`를 입력하여 `boot/cmdline.txt` 에디터 실행

*에디터가 나오면 첫번째 줄 맨뒤로 이동하여 `usbhid.mousepoll=0` 입력 후 저장  
`sudo reboot`를 입력하여 재부팅*

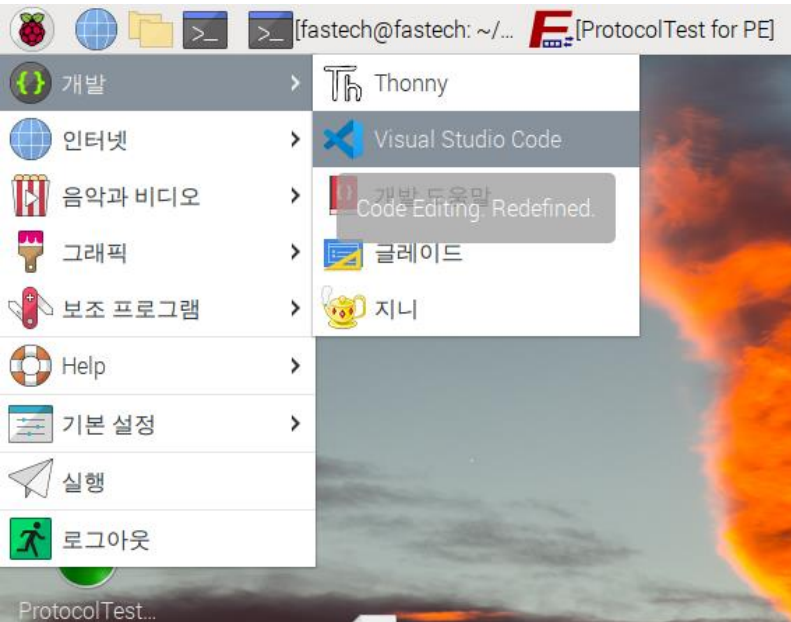
*국가를 한국으로 선택하였기 때문에 글자가 깨지는현상 이것을 한글화 함*

*터미널을 열고 `sudo apt install fonts-unfonts-core`을 입력 (폰트설치)*

*`sudo apt-get install ibus ibus-hangul` 입력(입력기 설치)*

*좌측상단 베리메뉴->기본설치->ibus입력기 설정을 hangul 로 설정*

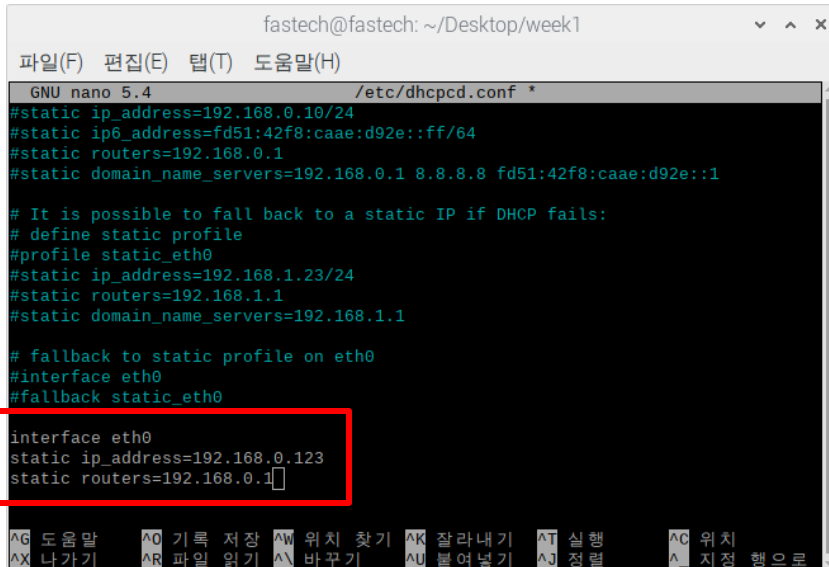
# • 개발환경



터미널에서 `sudo apt update`

`sudo apt install code`를 입력하여 vscode를 설치  
좌측상단 베리메뉴 → 개발 → vscode 실행가능

# • 라즈베리파이의 고정 IP 설정



```
fastech@fastech: ~/Desktop/week1
파일(F) 편집(E) 탭(T) 도움말(H)
GNU nano 5.4 /etc/dhcpd.conf *
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0

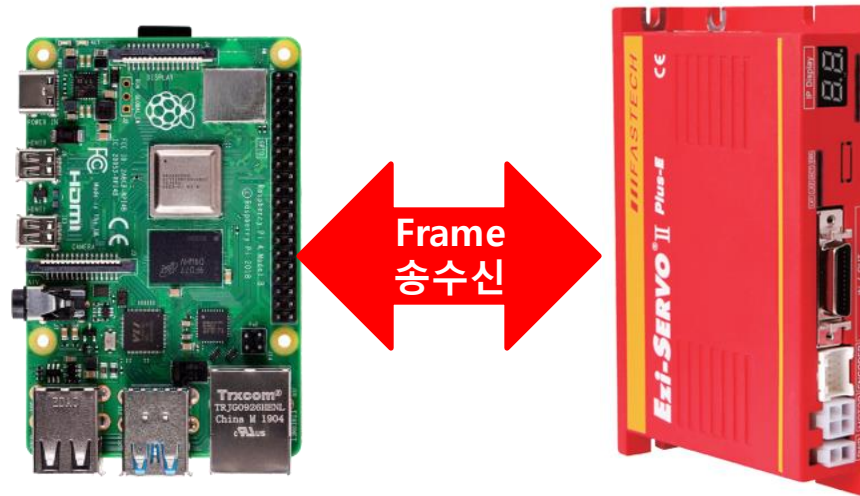
interface eth0
static ip_address=192.168.0.123
static routers=192.168.0.1
```

1. 터미널창에 **sudo nano /etc/dhcpd.conf** 입력
  1. dhcpd.conf 파일 들어가서 다음 내용 추가
2. 저장 및 파일 나오기
3. 터미널창에 **sudo /etc/init.d/networking restart** 입력
4. 터미널창에 **sudo reboot** 입력

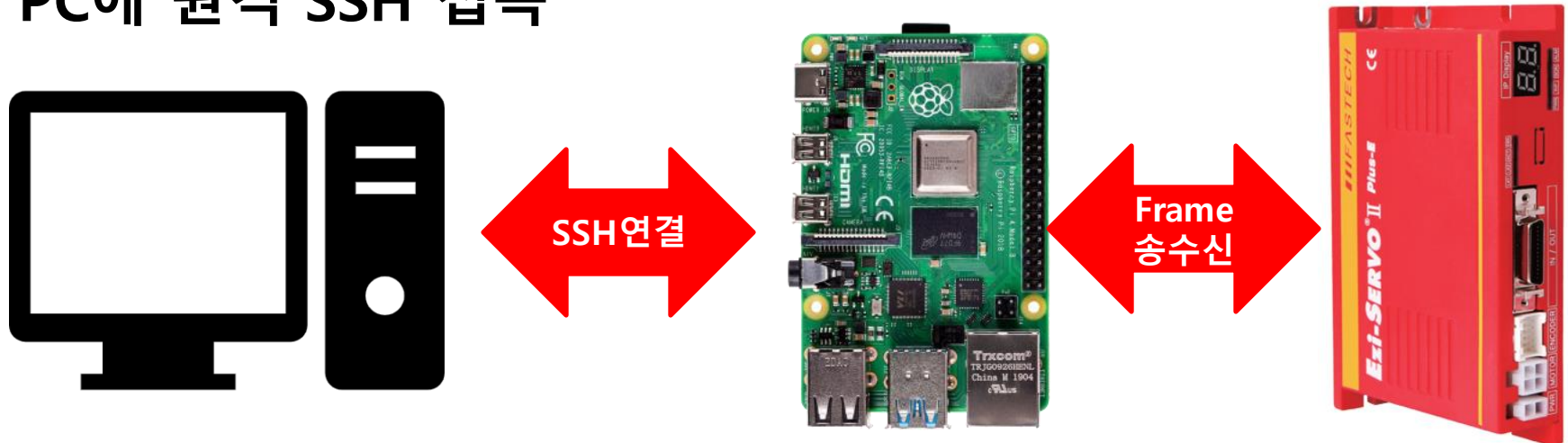
# • 터미널에서 실행되는 프로토콜 테스트

## 사용 환경

### 1. 터미널 환경



### 2. PC에 원격 SSH 접속





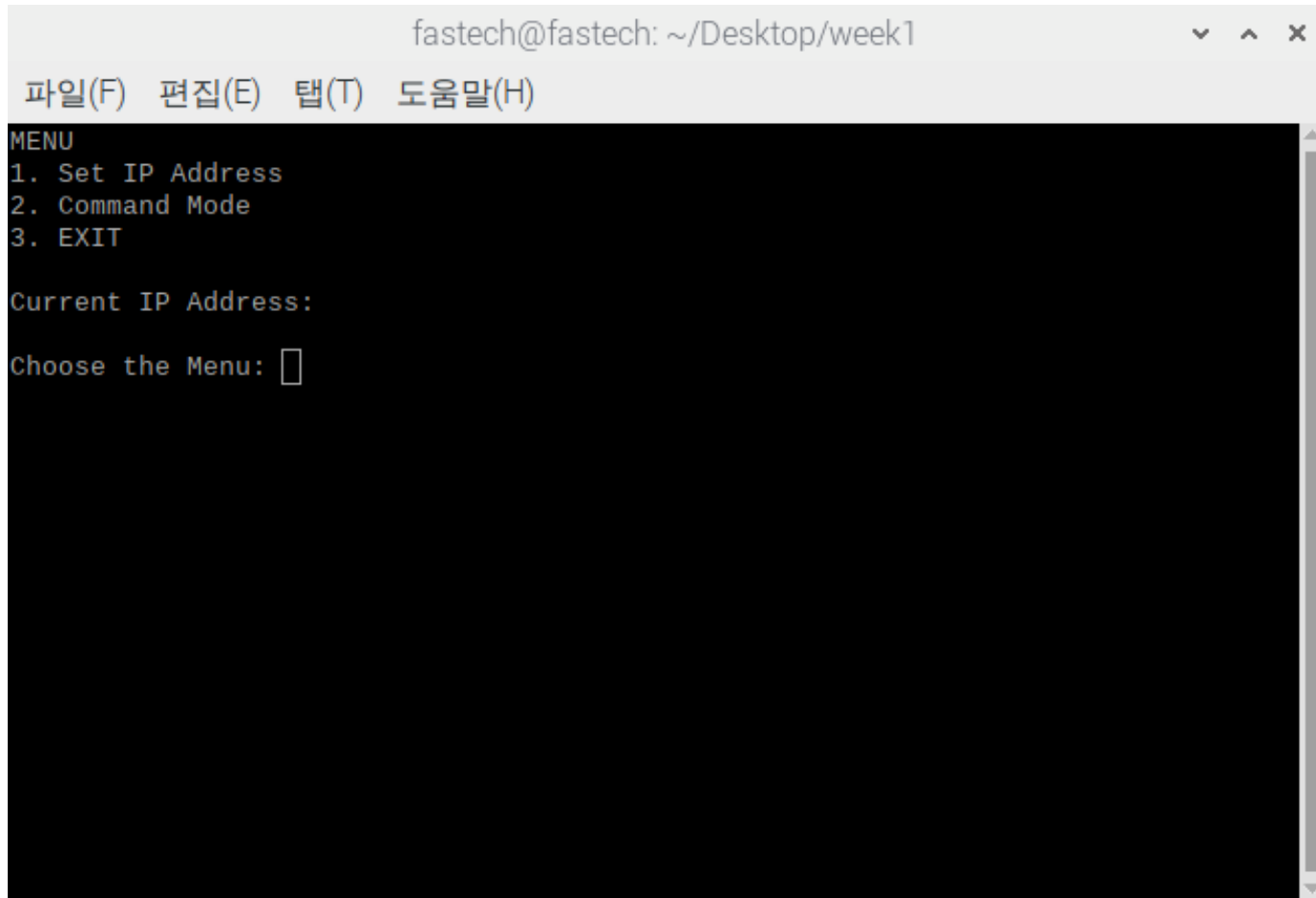
# • 터미널에서 실행되는 프로토콜 테스트

```
25 buffer_send[3] = 0x00;
26
27 // system("clear");
28 while (1)
29 {
30     Command_Buffer(FrameType); // Command_Buffer 함수를 호출하여 프레임 데이터를 구성합니다.
31
32     // 클라이언트 소켓을 통해 서버에 데이터를 전송합니다.
33     int send_result = sendto(client_socket, buffer_send, length + 2, 0, (const struct sockaddr *)&server_addr, sizeof(server_a
34     if (send_result < 0)
35     {
36         perror("sendto 실패");
37     }
38
39     // 서버로부터 데이터를 수신합니다.
40     received_bytes = recvfrom(client_socket, buffer_rcv, sizeof(buffer_rcv), 0, NULL, NULL);
41     if (received_bytes < 0)
42     {
43         perror("recvfrom 실패");
44         continue;
45     }
46
47     break; // 루프 탈출
48 }
49
50 // close(client_socket); // 주석 처리된 부분은 소켓을 연결한 후에 연결을 종료하는 코드
```

C Command\_Buffer.c  
h Command\_Buffer.h  
C Command.c  
h Command.h  
C DOS\_Test.c  
C Get\_IP\_Address.c  
h Get\_IP\_Address.h  
C Monitor.c  
h Monitor.h  
C Print\_Command\_List.c  
h Print\_Command\_List.h  
i README.txt  
C UDP\_Client\_Connect.c  
h UDP\_Client\_Connect.h  
C UDP\_Client\_Send.c  
h UDP\_Client\_Send.h

소스파일 및  
헤더파일

- 터미널에서 실행되는 프로토콜 테스트



A terminal window titled 'fastech@fastech: ~/Desktop/week1' with standard window controls. The menu text is as follows:

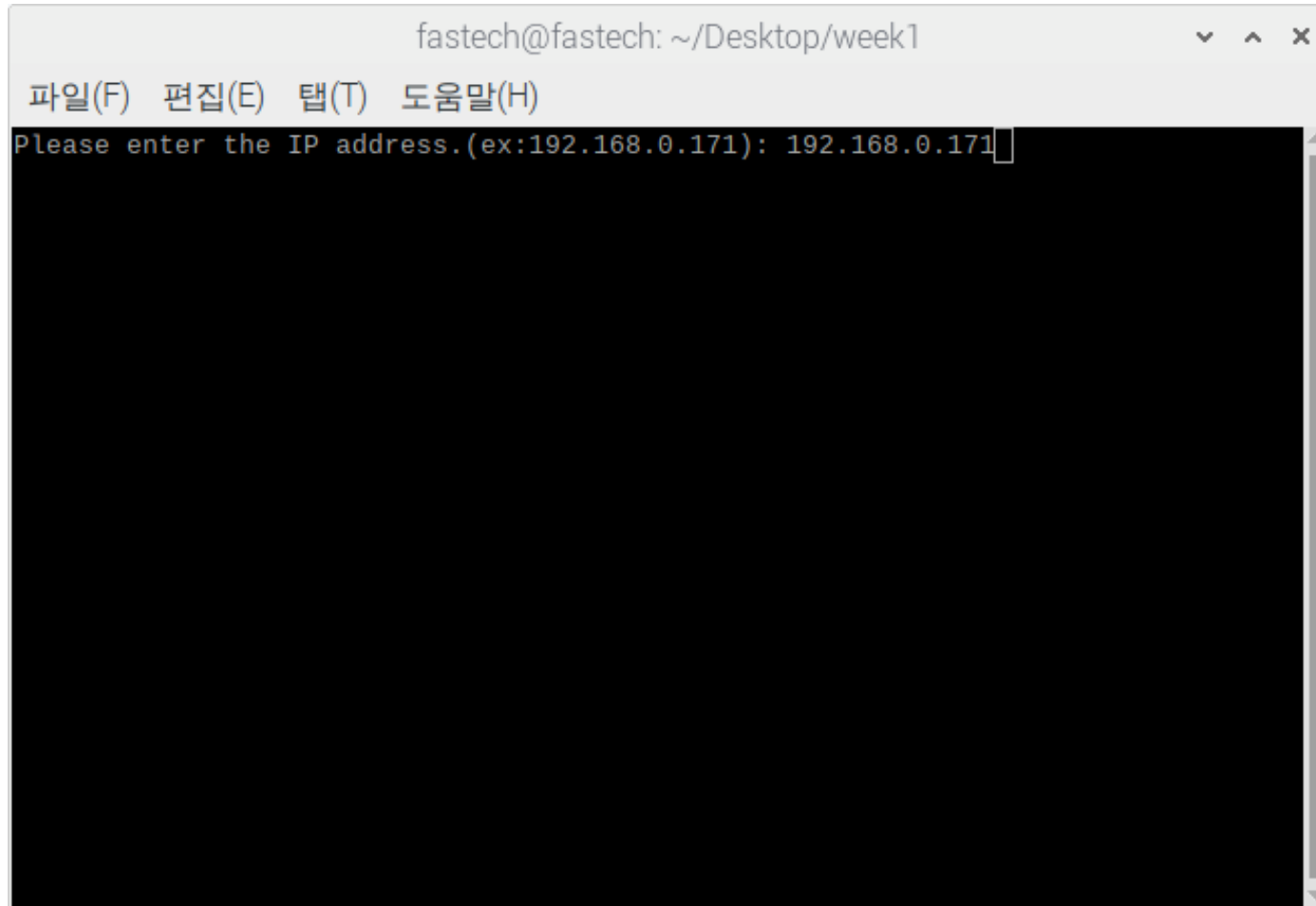
```
파일(F) 편집(E) 탭(T) 도움말(H)
MENU
1. Set IP Address
2. Command Mode
3. EXIT

Current IP Address:
Choose the Menu: 
```

시작화면

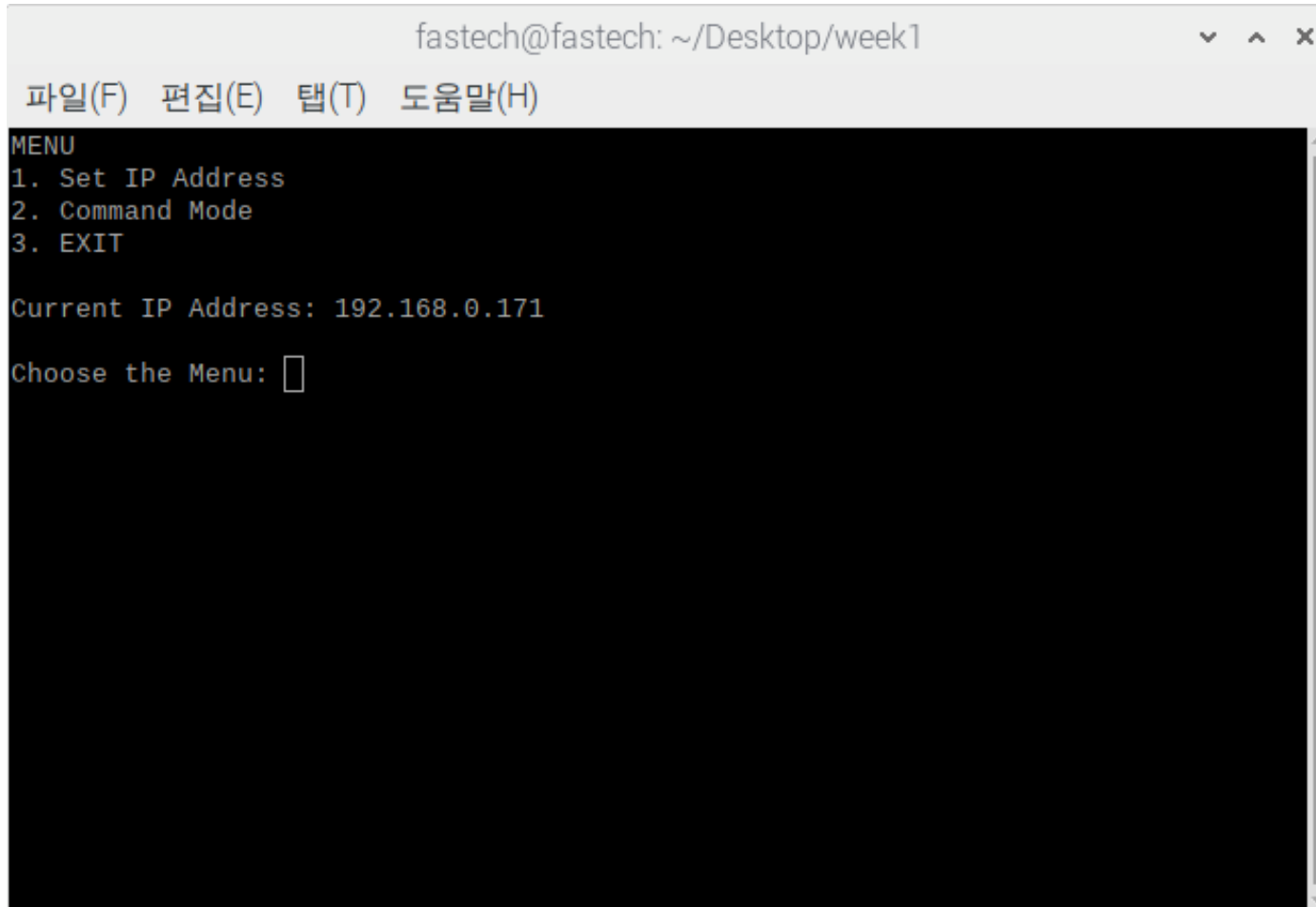


- 터미널에서 실행되는 프로토콜 테스트



IP 입력창

- 터미널에서 실행되는 프로토콜 테스트



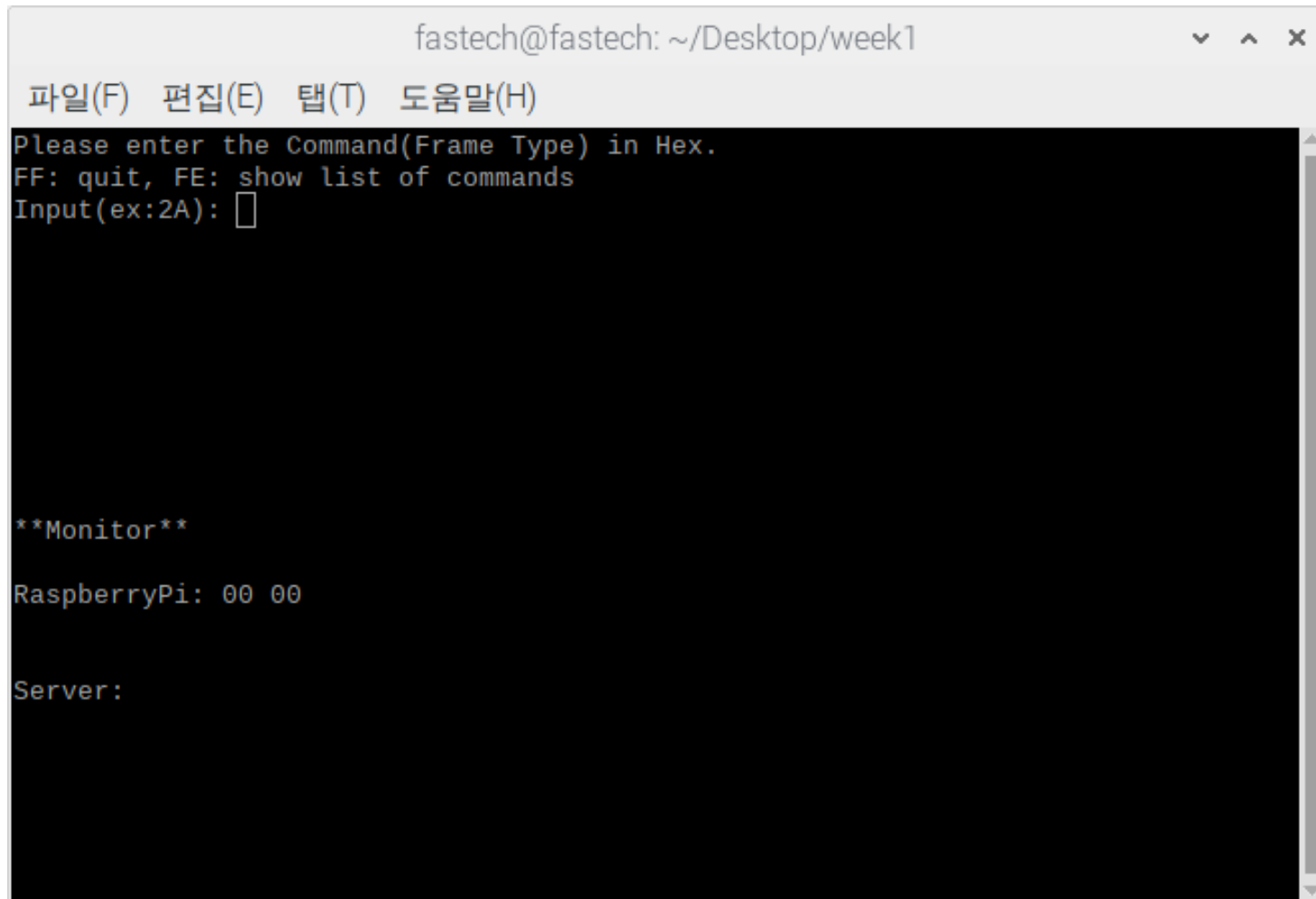
```
fastech@fastech: ~/Desktop/week1
파일(F) 편집(E) 탭(T) 도움말(H)
MENU
1. Set IP Address
2. Command Mode
3. EXIT

Current IP Address: 192.168.0.171

Choose the Menu: 
```

모터드라이브의 IP주소가 업데이트 된 시작화면

- 터미널에서 실행되는 프로토콜 테스트

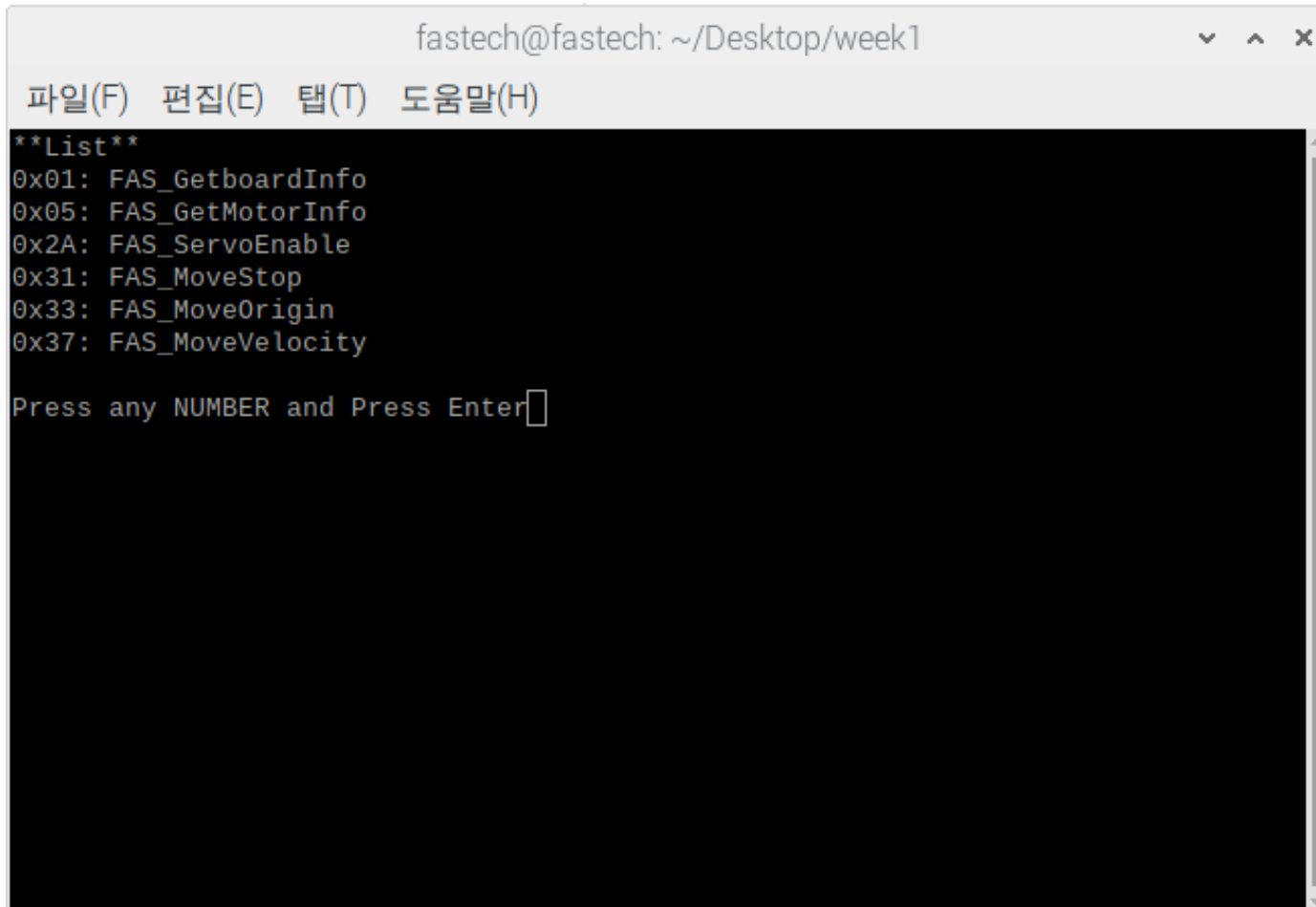


A terminal window titled 'fastech@fastech: ~/Desktop/week1' with standard window controls. The menu bar includes '파일(F)', '편집(E)', '탭(T)', and '도움말(H)'. The main text area contains the following text: 'Please enter the Command(Frame Type) in Hex.', 'FF: quit, FE: show list of commands', 'Input(ex:2A):' followed by a cursor, a large black redacted area, and then 'Server:'. Below the redacted area, the text '\*\*Monitor\*\*' and 'RaspberryPi: 00 00' are visible. A vertical scrollbar is on the right side of the terminal window.

```
fastech@fastech: ~/Desktop/week1
파일(F) 편집(E) 탭(T) 도움말(H)
Please enter the Command(Frame Type) in Hex.
FF: quit, FE: show list of commands
Input(ex:2A): 
**Monitor**
RaspberryPi: 00 00
Server:
```

명령 입력창

- 터미널에서 실행되는 프로토콜 테스트

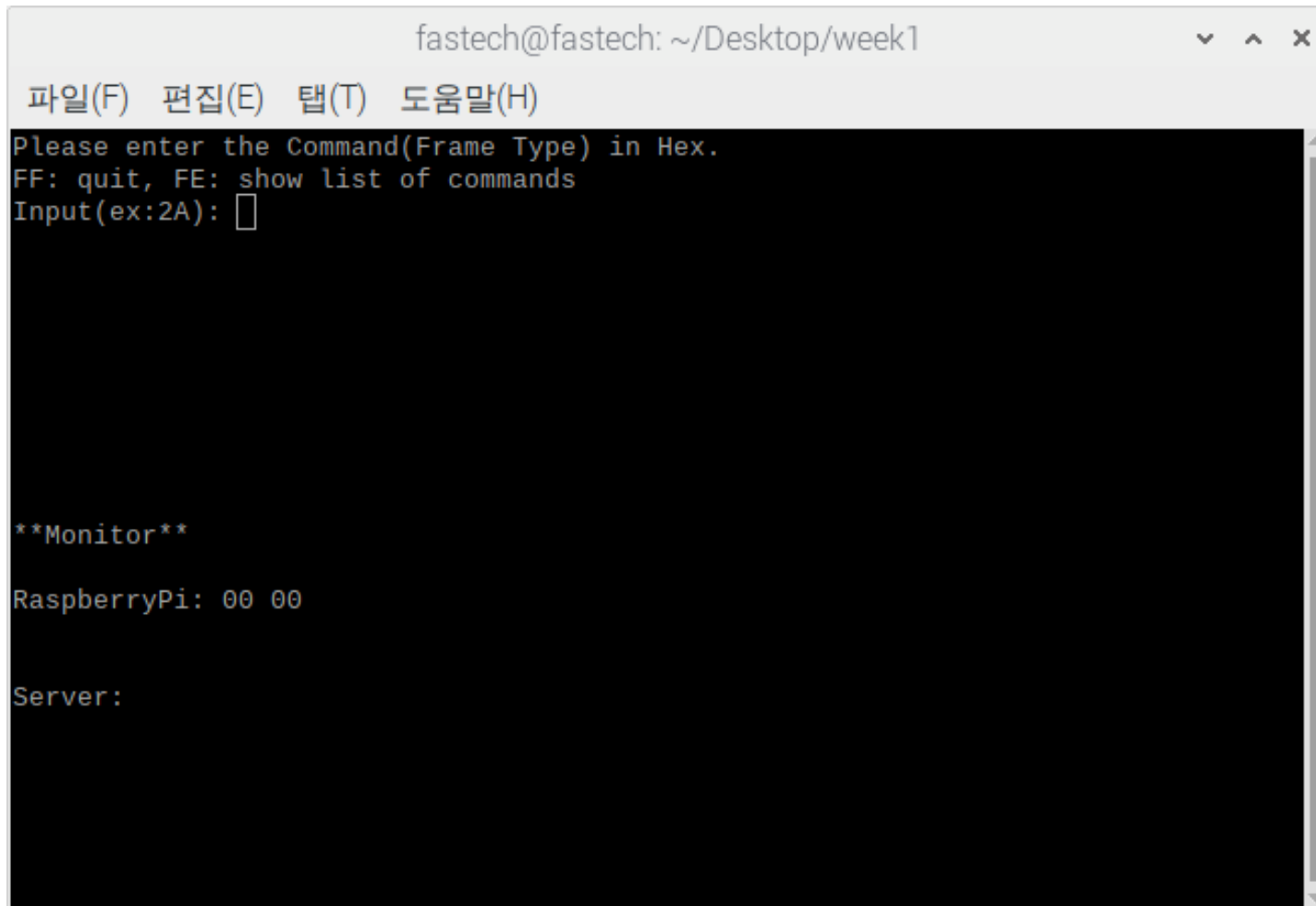


A terminal window titled 'fastech@fastech: ~/Desktop/week1' with standard window controls. The menu bar shows '파일(F) 편집(E) 탭(T) 도움말(H)'. The main content area has a black background with white text. It displays a list of commands under the heading '\*\*List\*\*'. The commands are: 0x01: FAS\_GetboardInfo, 0x05: FAS\_GetMotorInfo, 0x2A: FAS\_ServoEnable, 0x31: FAS\_MoveStop, 0x33: FAS\_MoveOrigin, and 0x37: FAS\_MoveVelocity. At the bottom, it prompts 'Press any NUMBER and Press Enter' with a cursor.

```
fastech@fastech: ~/Desktop/week1
파일(F) 편집(E) 탭(T) 도움말(H)
**List**
0x01: FAS_GetboardInfo
0x05: FAS_GetMotorInfo
0x2A: FAS_ServoEnable
0x31: FAS_MoveStop
0x33: FAS_MoveOrigin
0x37: FAS_MoveVelocity
Press any NUMBER and Press Enter
```

FE, 내재되어 있는 명령  
들을 보여줌

- 터미널에서 실행되는 프로토콜 테스트



A terminal window titled 'fastech@fastech: ~/Desktop/week1' with standard window controls. The menu bar shows '파일(F) 편집(E) 탭(T) 도움말(H)'. The terminal text is as follows:

```
Please enter the Command(Frame Type) in Hex.  
FF: quit, FE: show list of commands  
Input(ex:2A):   
  
**Monitor**  
  
RaspberryPi: 00 00  
  
Server:
```

명령 입력창

# • 터미널에서 실행되는 프로토콜 테스트

```
fastech@fastech: ~/Desktop/week1
파일(F) 편집(E) 탭(T) 도움말(H)
Please enter the Command(Frame Type) in Hex.
FF: quit, FE: show list of commands
Input(ex:2A): 2A

-Enable
ON(1), OFF(0): 1

**Monitor**
0x2A: FAS_ServoEnable
RaspberryPi: AA 04 04 00 2A 01

Server: AA 04 04 00 2A 00
```

0x2A:  
FAS\_ServoEnable

# • 터미널에서 실행되는 프로토콜 테스트

```
fastech@fastech: ~/Desktop/week1
파일(F) 편집(E) 탭(T) 도움말(H)
Please enter the Command(Frame Type) in Hex.
FF: quit, FE: show list of commands
Input(ex:2A): 37

-Speed: 1000

-Direction
+Jog(1), -Jog(0): 1

**Monitor**
0x37: FAS_MoveVelocity
RaspberryPi: AA 08 06 00 37 E8 03 00 00 01

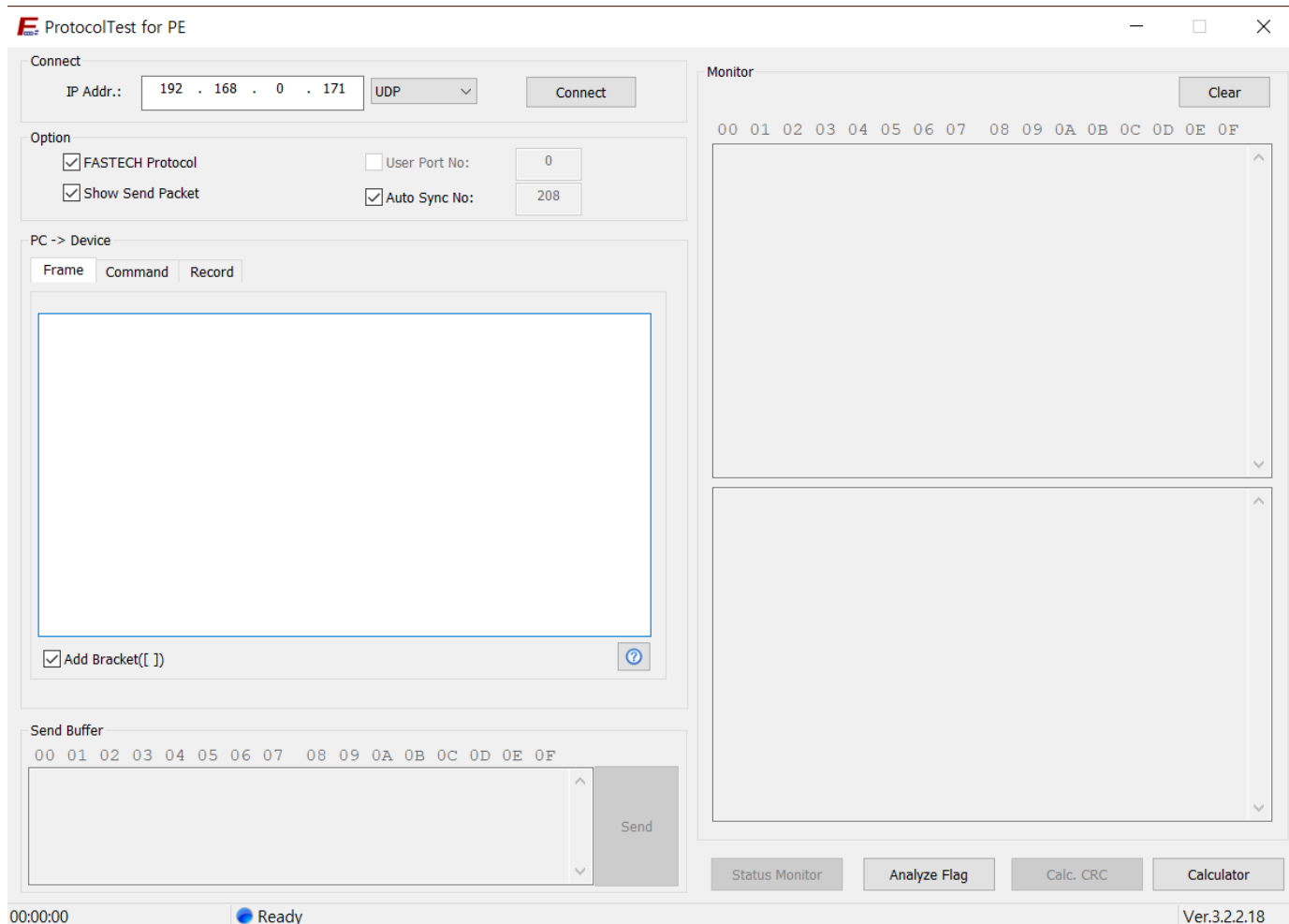
Server: AA 04 06 00 37 00
```

0x37:  
FAS\_MoveVelocity



# • ProtocolTest

Fastech에서 제공하는 윈도우 환경 ProtocolTest 프로그램

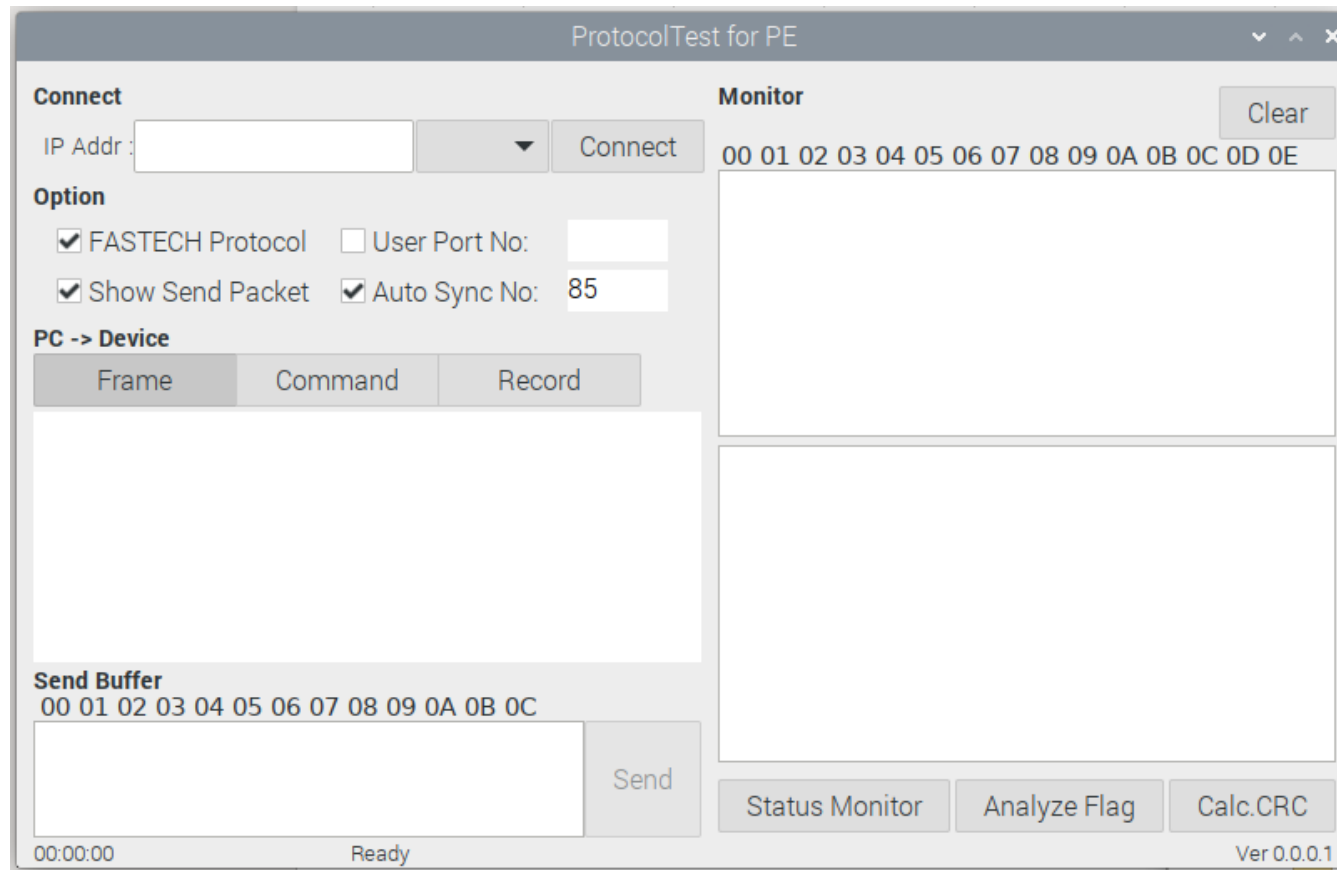


# • ProtocolTest Linux Version

라즈베리파이로 이식 중인 ProtocolTest 프로그램

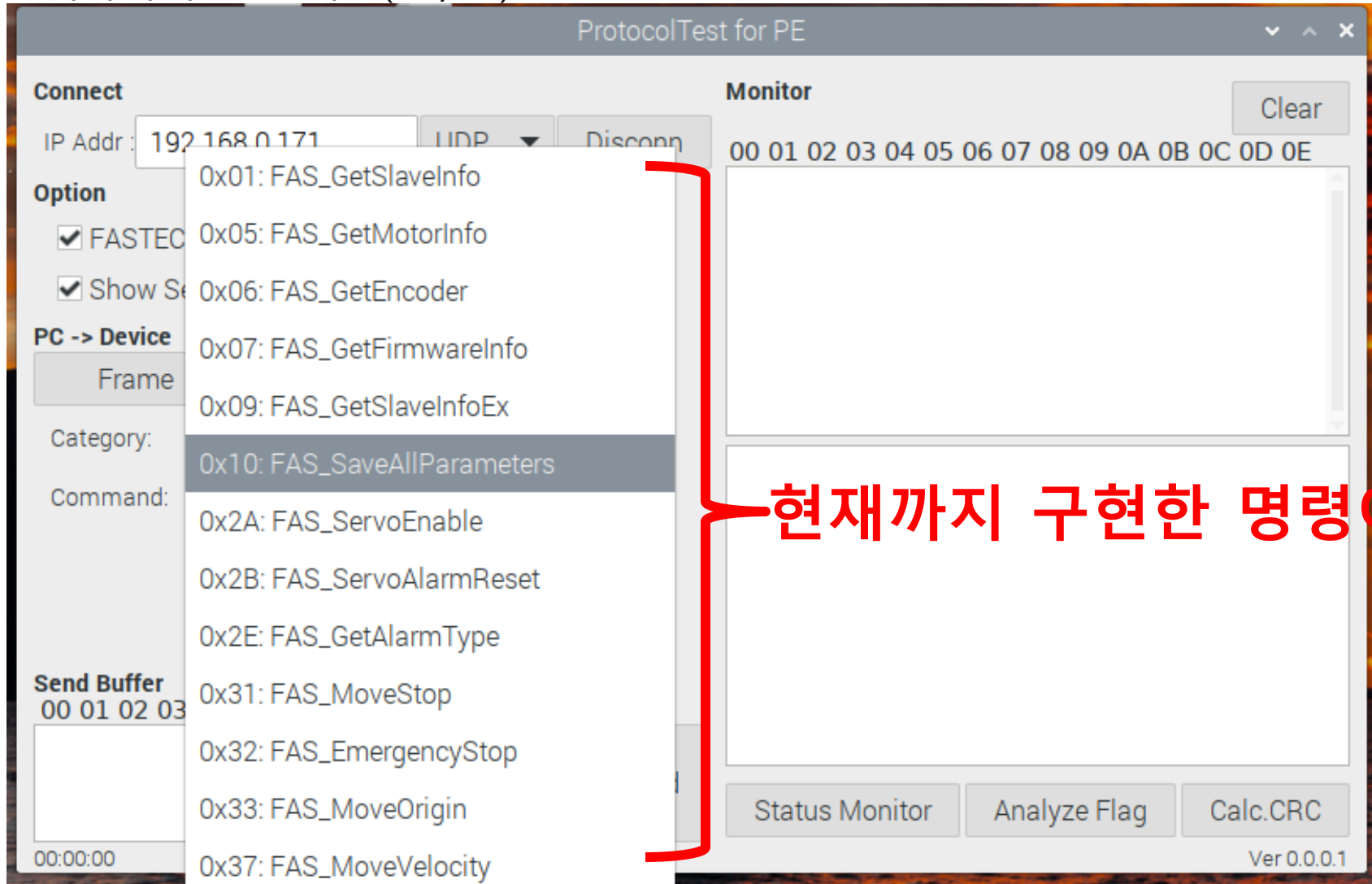
```
fastech@fastech:~/Desktop $ gcc ProtocolTest.c -o ProtocolTest `pkg-config --cflags --libs gtk+-3.0`  
fastech@fastech:~/Desktop $ ./ProtocolTest
```

터미널에서 파일을 같은 디렉토리에 두고, 위의 명령어로 실행파일 생성, 실행



# • ProtocolTest Linux Version

현재까지 구현한 기능(~7/27)



# • ProtocolTest Linux Version

현재까지 구현한 기능(~7/27)

The screenshot shows the 'ProtocolTest for PE' application window. It features a 'Connect' section with an IP address field (192.168.0.171), a protocol dropdown (UDP), and a 'Disconn' button. An 'Option' section includes checkboxes for 'FASTECH Protocol' and 'Show Send Packet', and input fields for 'User Port No.' and 'Auto Sync No.' (86). Below this is a 'PC -> Device' section with tabs for 'Frame', 'Command', and 'Record'. The 'Command' tab is active, showing a 'Category' dropdown (ALL), a 'Command' dropdown (0x2A: FAS\_ServoEnable), and an 'Enable' dropdown (1: ON). A 'Send Buffer' section at the bottom left contains a hex data field (AA 04 56 00 2A 01) and a 'Send' button. On the right, a 'Monitor' section displays hex data (AA 04 56 00 2A 01) and text labels for '[SEND] FAS\_ServoEnable' and '[RECEIVE] FAS\_ServoEnable RESPONSE : FMM\_OK'. At the bottom right are buttons for 'Status Monitor', 'Analyze Flag', and 'Calc.CRC'. The status bar at the bottom shows '00:00:00' and 'Ready'.

**IP입력, UDP프로토콜 연결/연결해제**

**전송한 Frame**

**수신한 Frame**

**랜덤 Auto Sync No 생성&표시**

**일부 명령어 선택**

**보내는 Buffer**

**주고받는 명령 통신상태**

# • ProtocolTest Linux Version

ProtocolTest for PE

**Connect**

IP Addr: 192.168.0.171 UDP Disconn

**Option**

☒ FASTECH Protocol ☐ User Port No:

☒ Show Send Packet ☒ Auto Sync No: 87

**PC -> Device**

Frame Command Record

Category: ALL ☐ Use List

Command: 0x37: FAS\_MoveVelocity

Speed: 5000

Direction: 1: +Jog

**Send Buffer**

00 01 02 03 04 05 06 07 08 09 0A 0B 0C

AA 08 57 00 37 88 13 00 00 01

Send

**Monitor**

Clear

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E

AA 08 57 00 37 88 13 00 00 01

AA 04 57 00 37 00

[SEND]

FAS\_MoveVelocity

[RECEIVE]

FAS\_MoveVelocity

RESPONSE : FMM\_OK

Status Monitor Analyze Flag Calc.CRC

00:00:00 Ready Ver 0.0.0.1

# • 사용한 Framework

## GTK

GTK 42개 언어

문서 토론

위키백과, 우리 모두의 백과사전.

**GTK**는 킴프 툴킷(GIMP Toolkit)의 준말로, 초기에 **킴프**를 위해서 만든 툴킷이었으며 **X 윈도 시스템**을 위한 위젯 툴킷 가운데 하나이다. GTK와 Qt는 **모티프**에 대한 좋은 대안이 되어 주었다. GTK는 1997년 **스펜서 킴볼**(Spencer Kimball), **피터 마티스**(Peter Mattis), **조시 맥도널드**(Josh MacDonald)가 함께 만든 것이다. 그들은 모두 **UC 버클리**에 있는 **eXperimental Computing Facility (XCF)** 소속이었다. **LGPL**로 라이선스되었기 때문에, GTK는 자유 소프트웨어이자 오픈 소스 소프트웨어이고, GNU 프로젝트의 일 부분이다.

### 설계

GTK는 C언어로 작성된 객체지향 위젯 툴킷이다. X11 디스플레이 서버 상에서, GTK는 위젯들을 그리는 데 Xlib를 사용한다. Xlib는 유연하고 X 윈도 시스템이 작동하지 않는 플랫폼에서도 GTK가 사용될 수 있도록 한다.

GTK는 Qt와 마찬가지로 (다른 많은 위젯 툴킷들과 달리) **Xt**에 기반을 두지 않는다. 그래서 GTK를 많은 다른 환경으로 이식할 수 있었다. 하지만 전통적인 X11 응용 프로그램의 사용자 설정 방식인 X 리소스 데이터베이스에 접근할 수 없다는 단점이 있다.

### 언어 바인딩

C++, 펄, 루비, 자바, 파이썬 등으로의 바인딩을 제공한다. 많은 사람들이 **에이다**, **D**, **하스켈**, **파스칼**, **PHP**, **닷넷 프레임워크**로의 바인딩을 작성하였다.

### Hello World 프로그램

#### C

- 소스 코드

위키백과, 우리 모두의 백과사전.

문서 편집 역사 보기 도구

위키백과, 우리 모두의 백과사전.

**GTK+**

<b>개발자</b>	그놈 재단
<b>발표일</b>	1998년 4월 14일(25년 전)
<b>안정화 버전</b>	4.6.0 / 2021년 12월 30일(18개월 전) <sup>[1]</sup>
<b>미리보기 버전</b>	4.3.1 / 2021년 6월 9일(2년 전) <sup>[2]</sup>
<b>저장소</b>	<a href="https://gitlab.gnome.org/GNOME/gtk">gitlab.gnome.org/GNOME/gtk</a>
<b>운영 체제</b>	크로스 플랫폼
<b>종류</b>	개발 라이브러리
<b>라이선스</b>	LGPL
<b>웹사이트</b>	<a href="http://www.gtk.org">www.gtk.org</a>

## QT

Qt (소프트웨어)

문서 토론

위키백과, 우리 모두의 백과사전.

**Qt**는 컴퓨터 프로그래밍에서 **GUI** 프로그램 개발에 널리 쓰이는 크로스 플랫폼 프레임워크이다. 서버 용 콘솔과 명령 줄 도구와 같은 비GUI 프로그램 개발에도 사용된다. 그래픽 사용자 인터페이스를 사용하는 경우에는 Qt를 **위젯 툴킷**으로 분류한다. 회사 내부에서는 Qt를 "cute"로 발음하고 있으며 비공식적으로는 "큐티"로 발음한다. Qt는 **KDE**, **Qtopia**, **OPIE**에 이용되고 있다,

노르웨이 회사 **트를텍**에 의해서 개발되었다. 2008년 1월에는 **노키아**에 인수되었다.<sup>[3]</sup> 이후, 2012년 8월에 핀란드 회사 **Digia**에 인수되었다.<sup>[4]</sup>

Qt는 C++를 주로 사용하지만, **파이썬**, **루비**, **C**, **펄**, **파스칼**과도 연동된다. 수많은 플랫폼에서 동작하며, 상당히 좋은 국제화를 지원한다. **SQL** 데이터베이스 접근, **XML** 처리, **스레드** 관리, 단일 크로스 플랫폼 파일 관리 **API**를 제공한다.

### 종류

Qt는 다음 플랫폼으로 개발된다.

- 리눅스/X11 — **X 윈도 시스템**(유닉스 / 리눅스)을 위한 Qt
- 맥 OS X — OS X을 위한 Qt
- 윈도 — **마이크로소프트 윈도우**를 위한 Qt
- 임베디드 리눅스 — **PDA**, **스마트폰** 등의 임베디드 플랫폼을 위한 Qt
- 윈도 CE — **윈도용 CE** 등을 위한 Qt
- 심비안 - 심비안을 위한 Qt
- 마에모 - Maemo를 위한 Qt

각각 플랫폼에는 세 종류의 에디션이 있다.

- GUI 프레임워크 — 네트워킹과 데이터베이스를 제외한 순수 GUI 개발 에디션(데스크톱 라이트-Desktop Light-라고도 불린다.)
- 동 프레임워크 — 상업용 개발을 위한 완전한 에디션.

Qt 42개 언어

문서 편집 역사 보기 도구

위키백과, 우리 모두의 백과사전.

**Qt**

임베디드 Qt를 이용한 Qt 크리에이터의 GUI 디자인

<b>개발자</b>	Trolltech (1991–2008) 노키아 (2008–2011) Digia (2012–2014) Qt 프로젝트 (2011–현재) Qt 컴퍼니 (2014–현재)
<b>발표일</b>	1995년 5월 20일(28년 전) <sup>[1]</sup>
<b>안정화 버전</b>	6.6 Beta <sup>[2]</sup> / 2023년 6월 15일; 2023년 7월 19일(35일 전)
<b>저장소</b>	<a href="https://code.qt.io/cgit/qt/qtbase.git/">code.qt.io/cgit/qt/qtbase.git/</a>
<b>운영 체제</b>	크로스 플랫폼
<b>종류</b>	개발 라이브러리
<b>라이선스</b>	GPL, LGPL, 상용 버전
<b>웹사이트</b>	<a href="http://www.qt.io">www.qt.io</a>

**LGPL 라이선스로 상업적 이용  
소스코드 비공개 가능**

# • 개발환경

## Glade

### 글레이드 인터페이스 디자이너

🗨️ 19개 언어 ▾

문서 토론

읽기 편집 역사 보기 도구 ▾

위키백과, 우리 모두의 백과사전.

**글레이드 인터페이스 디자이너**(Glade Interface Designer)는 **그놈**의 추가 구성요소가 포함된 **GTK**용 그래픽 사용자 인터페이스 빌더이다. 3번째 버전의 글레이드는 **프로그래밍 언어** 의존적이며 이벤트를 위한 코드를 생성하지는 않고 적절한 바인딩에 사용할 수 있는 **XML** 파일을 생성한다.(**에이다** 프로그래밍 언어를 사용하는 GtkAda 등)

글레이드는 **GNU 일반 공중 사용 허가서**로 배포되는 **자유-오픈 소스 소프트웨어**이다.

### 같이 보기 [ 편집 ]

- **인터페이스 빌더**
- **마이크로소프트 블렌드**

### 각주 [ 편집 ]

- ↑ Reed, David (2004년 7월 1일). "Rapid Application Development with Python and Glade" . *Linux Journal*. "Damon Chaplin wrote the Glade program"
- ↑ Chaplin, Damon (2000). "Glade FAQ version 1.0" .
- ↑ Welsh, Matt; Kalle Dalheimer, Matthias; Kaufman, Lar (August 1999). *Running Linux*  3판. Appendix B The GNOME Project > B.5.3 Programming Tools > ..."Of particular interest is Damon Chaplin's Glade..."
- ↑ "Damon Chaplin (author of the original Glade tool)" . 3 November 2006에 **원본 문서**에서 보존된 문서. 18 February 2013에 확인함.
- ↑ "Historical Glade website" . 23 April 1999에 **원본 문서**에서 보존된 문서. 18 February 2013에 확인함.
- ↑ "GLADE GTK+ User Interface Builder > History > The first release, Version 0.1, was on 18. Apr 1998" . 8 October 1999에 **원본 문서**에서 보존된 문서. 18 February 2013에 확인함.
- ↑ "Files · master · GNOME / Glade" .

### 글레이드



글레이드에서 환경 설정 대화 상자를 디자인하는 모습

원저자	데몬 채플린(Damon Chaplin) <sup>[1][2][3][4][5]</sup>
개발자	그놈 프로젝트
발표일	1998년 4월 18일(25년 전) <sup>[6]</sup>
저장소	<a href="https://gitlab.gnome.org/GNOME/glade.git">gitlab.gnome.org/GNOME/glade.git</a> <sup>[7]</sup>
프로그래밍 언어	C, XML
운영 체제	유닉스 계열, 윈도우 <sup>[7]</sup>
종류	GUI 빌더 리눅스 온 더 데스크톱 휴먼 인터페이스 가이드라인
라이선스	GNU 일반 공중 사용 허가서
웹사이트	<a href="https://glade.gnome.org">glade.gnome.org</a> <sup>[8]</sup>



# • GTK3 & Glade 설치

## 공식 문서

GTK3 : <https://docs.gtk.org/gtk3/>

Glade : <https://gitlab.gnome.org/GNOME/glade>



## GTK 설치

### Installing GTK

```
sudo apt-get install libgtk-3-dev
```

### Using In Your Project

Update your makefile to use the gtk libs and cflags:

```
LIBS = `pkg-config --libs gtk+-3.0`  
CFLAGS = `pkg-config --cflags gtk+-3.0`
```

Include gtk and get programming - see [here](#).

```
#include <gtk/gtk.h>
```

## Glade 설치

To install Glade on Raspberry Pi, enter the command:  
**\$ sudo apt-get install glade**

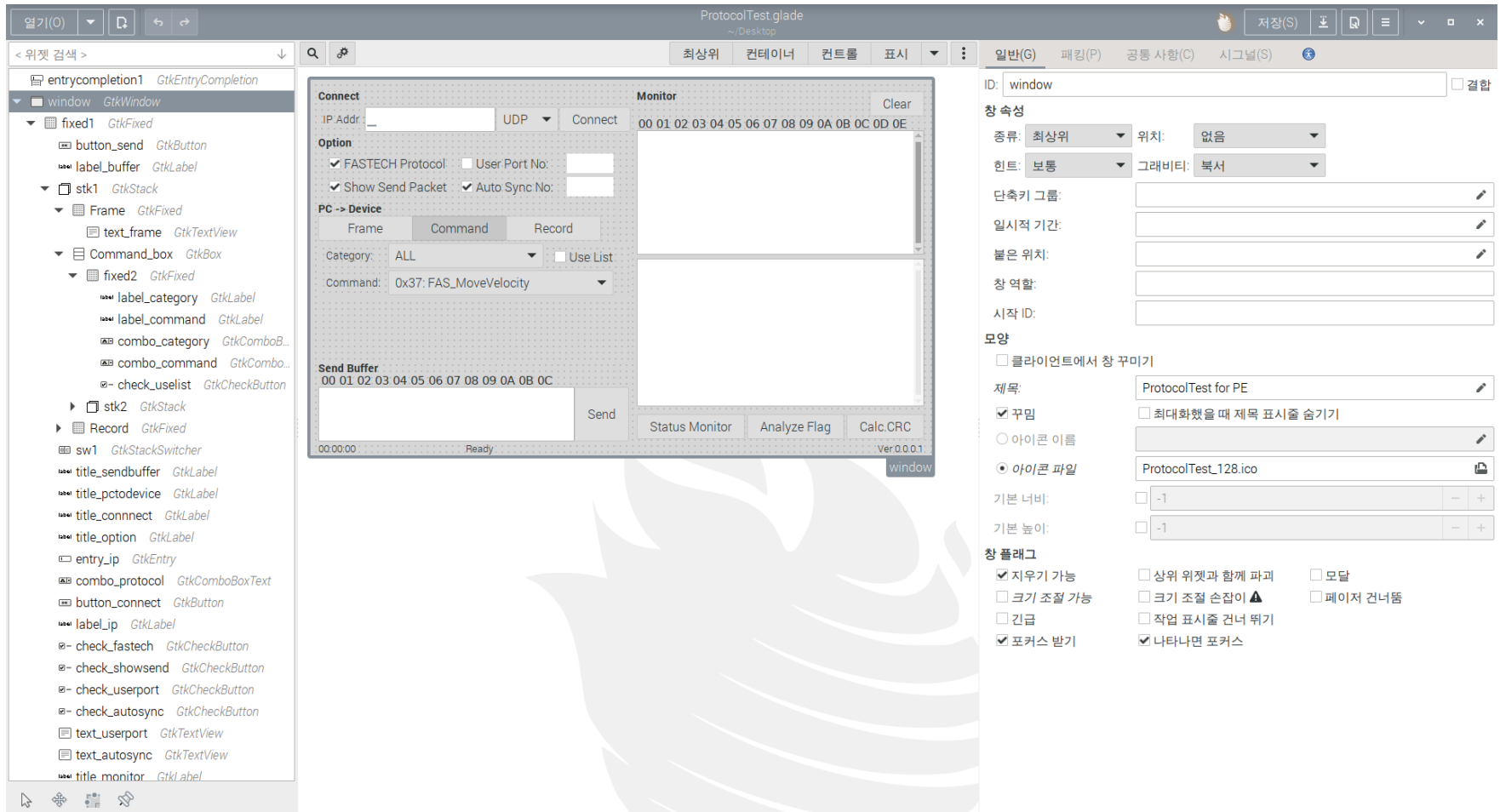
# • Glade

## GTK에서 제공하는 방식대로 XML 작성

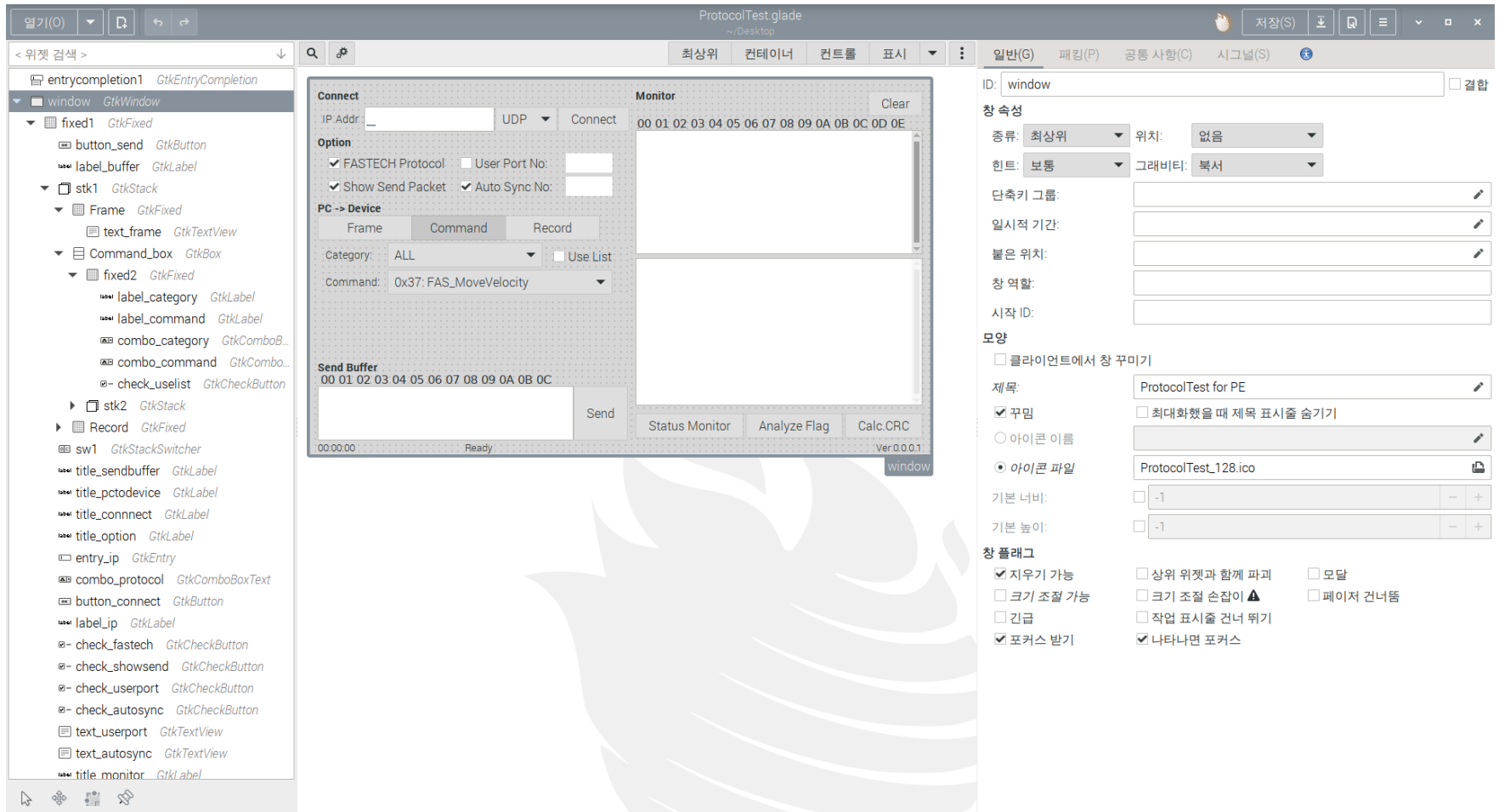
```
~/Desktop/ProtocolTest glade - Mousepad
파일(F) 편집(E) 검색(S) 보기(V) 문서(D) 도움말(H)
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.38.2 -->
<interface>
  <requires lib="gtk+" version="3.24"/>
  <object class="GtkEntryCompletion" id="entrycompletion1"/>
  <object class="GtkWindow" id="window">
    <property name="width-request">800</property>
    <property name="height-request">480</property>
    <property name="can-focus">False</property>
    <property name="title" translatable="yes">ProtocolTest for PE</property>
    <property name="resizable">False</property>
    <property name="icon">ProtocolTest_128.ico</property>
    <child>
      <object class="GtkFixed" id="fixed1">
        <property name="visible">True</property>
        <property name="can-focus">False</property>
        <child>
          <object class="GtkButton" id="button_send">
            <property name="label" translatable="yes">Send</property>
            <property name="width-request">50</property>
            <property name="height-request">70</property>
            <property name="visible">True</property>
            <property name="can-focus">True</property>
            <property name="receives-default">True</property>
            <signal name="clicked" handler="on_button_send_clicked" swapped="no">
              <callback function="on_button_send_clicked">
                <property name="x">340</property>
                <property name="y">395</property>
              </callback>
            </signal>
          </object>
          <packing>
            <property name="x">340</property>
            <property name="y">395</property>
          </packing>
        </child>
      </object>
      <child>
        <object class="GtkLabel" id="label_buffer">
          <property name="width-request">320</property>
          <property name="height-request">20</property>
          <property name="visible">True</property>
          <property name="can-focus">False</property>
          <property name="label" translatable="yes">00 01 02 03 04 05 06 07</property>
          <attributes>
            <attribute name="font-desc" value="Sans 10"/>
            <attribute name="scale" value="1.1020000000000001"/>
          </attributes>
        </object>
        <packing>
          <property name="x">10</property>
          <property name="y">377</property>
        </packing>
      </child>
    </child>
  </object>
  <object class="GtkStack" id="stk1">
    <property name="width-request">400</property>
    <property name="height-request">400</property>
    <property name="visible">True</property>
    <property name="can-focus">False</property>
    <child>
      <object class="GtkFixed" id="Frame">
        <property name="visible">True</property>
        <property name="can-focus">False</property>
        <child>
          <object class="GtkTextView" id="text_frame">
            <property name="width-request">400</property>
            <property name="height-request">150</property>
            <property name="visible">True</property>
            <property name="can-focus">True</property>
          </object>
        </child>
      </object>
      <packing>
        <property name="name">Frame</property>
        <property name="title" translatable="yes">Frame</property>
      </packing>
    </child>
    <child>
      <object class="GtkBox" id="Command_box">
        <property name="visible">True</property>
        <property name="can-focus">False</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkFixed" id="fixed2">
            <property name="width-request">400</property>
            <property name="height-request">70</property>
            <property name="visible">True</property>
            <property name="can-focus">False</property>
            <child>
              <object class="GtkLabel" id="label_category">
                <property name="visible">True</property>
                <property name="can-focus">False</property>
                <property name="label" translatable="yes">Category:</property>
                <attributes>
                  <attribute name="scale" value="0.90000000000000002">
                    <property name="x">10</property>
                    <property name="y">5</property>
                  </attribute>
                </attributes>
              </object>
            </child>
          </object>
          <packing>
            <property name="x">10</property>
            <property name="y">5</property>
          </packing>
        </child>
      </object>
    </child>
  </object>
</interface>
```

# • Glade

## WYSIWYG방식(What You See Is What You Get)으로 UI디자인 가능



# • Glade - UI디자인

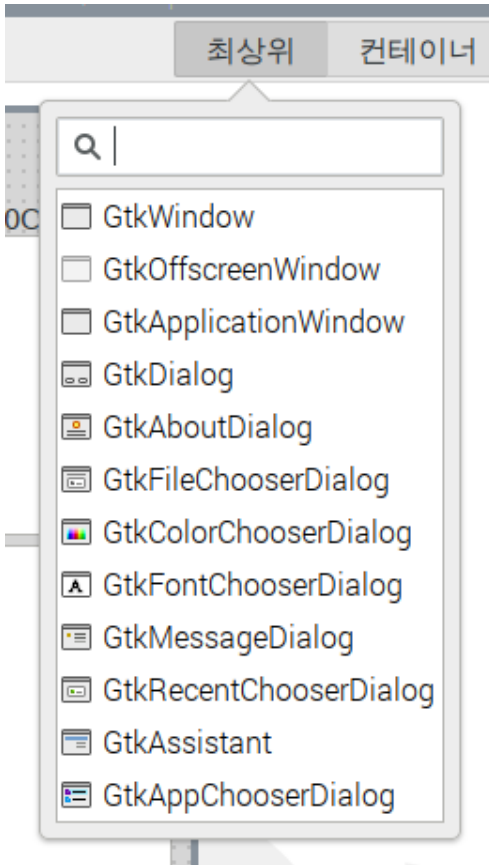


GTK3 : <https://docs.gtk.org/gtk3/>

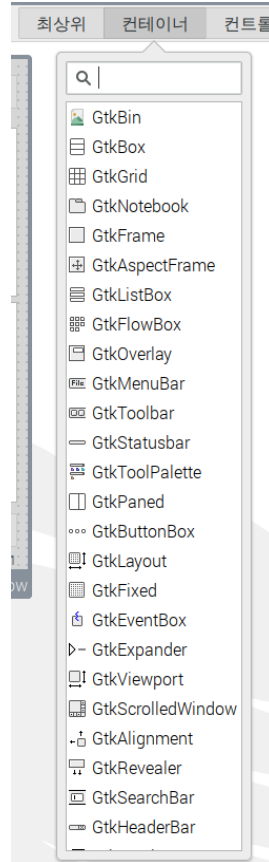
Glade : <https://gitlab.gnome.org/GNOME/glade>

# • Glade - UI디자인

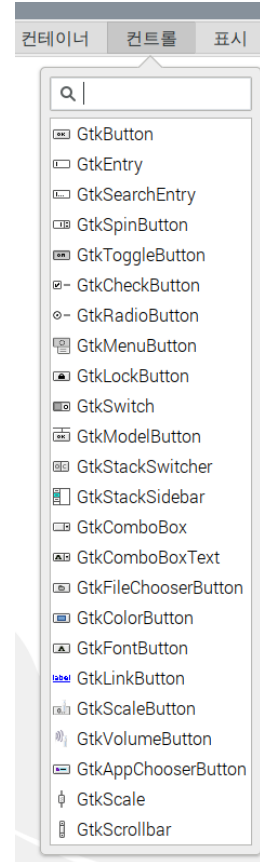
**최상위**  
(프로그램의 틀)



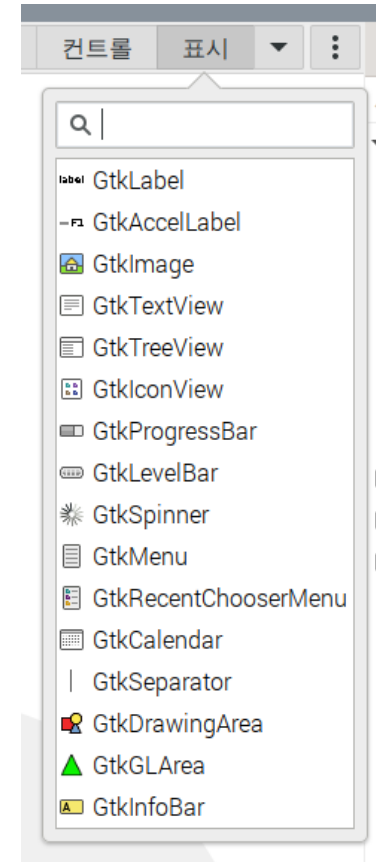
**컨테이너**  
(구성요소 담는 공간)



**컨트롤**  
(주로 입력을 받는 요소)



**표시**  
(주로 정보 표시)



GTK3 : <https://docs.gtk.org/gtk3/>

Glade : <https://gitlab.gnome.org/GNOME/glade>

# • Glade - UI디자인

일반 속성(구성요소마다 다름) 패킹->레이아웃 상 위치

공통사항 속성

일반(G) 패킹(P) 공통 사항(C) 시그널(S) ⓘ

ID: text\_monitor1

고정폭: ☐

텍스트 보기 속성

버퍼:

용도: 자유 형식

입력 힌트: 없음

☐ 편집 가능 ☐ 커서 보여주 기 ☐ 덮어 쓰기 모드

☐ 탭 입력 ☐ 채우기

텍스트 포매팅

줄바꿈 모드: 단어

행 맞춤: 왼쪽

들어쓰기: 0

간격

왼쪽: 0 위 줄: 0

오른쪽: 0 아래 줄: 0

위: 0 내부 줄 바꿈: 0

아래: 0

공백

스크롤 속성

가로:  정책: 최소

세로:  정책: 최소

일반(G) 패킹(P) 공통 사항(C) 시그널(S) ⓘ

가로 위치: 420

세로 위치: 47

일반(G) 패킹(P) 공통 사항(C) 시그널(S) ⓘ

위젯 속성

위젯 이름:

스타일 클래스:

풍선 도움말: ☐ 사용자 설정 ☐ 마크업 사용

불투명 위젯: 1.00

바로그가기:

이벤트: 구성

크기 변경 모드: 상위 위젯

위젯 플래그

☒ 보임 ☐ show all 안 하기 ☒ 반응

☒ 포커스 가능 ☐ 포커스 받음 ☐ 포커스인가

☐ 기본 가능 ☐ 기본 사용 ☐ 기본 받음

☐ 응용프로그램 그리기 가능 ☒ 더블 버퍼링 ⚠

위젯 공백

늘리기

가로: ☐ 세로: ☐

맞춤

가로: 채우기 세로: 채우기

간격

위: 0 시작: 0

아래: 0 끝: 0

왼쪽: ⚠ 0 오른쪽: ⚠ 0

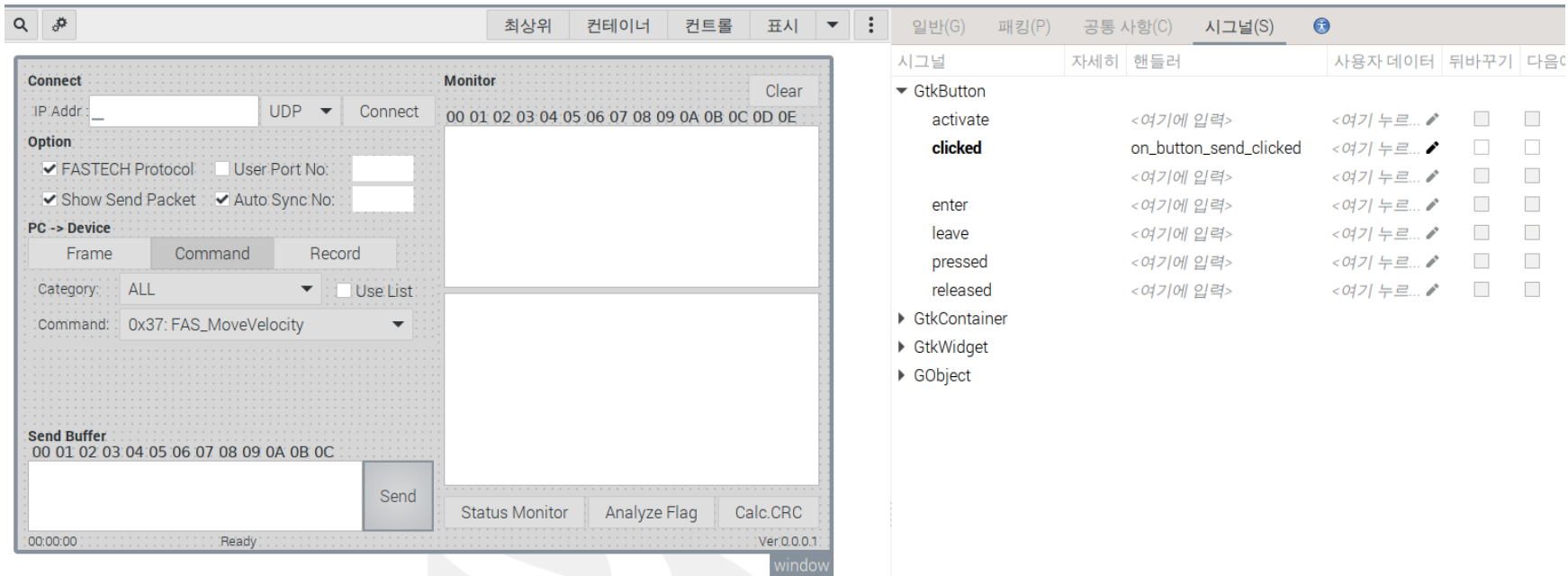
GTK3 : <https://docs.gtk.org/gtk3/>

Glade : <https://gitlab.gnome.org/GNOME/glade>



# • Glade - 상호작용

## 각 구성요소(버튼, 콤보박스, 체크박스 등)의 시그널과 Callback함수와 연동



The screenshot shows the Glade GUI designer interface. On the left, a window titled 'Connect' contains several controls: an 'IP Addr' text entry, a 'UDP' dropdown, a 'Connect' button, and checkboxes for 'FASTTECH Protocol', 'Show Send Packet', 'User Port No.', and 'Auto Sync No.'. Below these are tabs for 'Frame', 'Command', and 'Record', with 'Command' selected. A 'Category' dropdown is set to 'ALL', and a 'Command' dropdown is set to '0x37: FAS\_MoveVelocity'. At the bottom left is a 'Send Buffer' section with a text area and a 'Send' button. On the right, a 'Monitor' section displays a hex dump. The right-hand pane shows the 'Signals' tab, listing signals for various widget classes like GtkButton, GtkContainer, GtkWidget, and GObject. The 'clicked' signal for GtkButton is highlighted, showing its handler as 'on\_button\_send\_clicked'.

## 시스템 실행 단계에서 구성요소 생성, g\_signal\_connect로 callback함수와 연결

```
button = gtk_builder_get_object(builder, "button_send");
g_signal_connect(button, "clicked", G_CALLBACK(on_button_send_clicked), builder);
```

## Callback함수에 각 구성요소 별 기능 작성

```
/**@brief Send버튼의 callback*/
static void on_button_send_clicked(GtkButton *button, gpointer user_data){
    sync_no++;
    char sync_str[4];
    sprintf(sync_str, "%u", sync_no);

    gtk_text_buffer_set_text(autosync_buffer, sync_str, -1);
    library_interface();

    int send_result = sendto(client_socket, buffer, buffer[1] + 2, 0, (const struct sockaddr *)&server_addr, sizeof(server_addr));
    if (send_result < 0) {
        perror("sendto failed");
    }
}
```



# • C프로그램 구조

## Main 함수

```
int main(int argc, char *argv[]) {  
  
    GTK 구성요소 선언  
    GTK 메인루프 시작  
    등 초기설정  
  
}
```

## Callback 함수

```
static void on_button_connect_clicked(GtkButton *button, gpointer user_data);  
static void on_button_send_clicked(GtkButton *button, gpointer user_data);  
  
static void on_combo_protocol_changed(GtkComboBoxText *combo_text, gpointer user_data);  
static void on_combo_command_changed(GtkComboBox *combo_id, gpointer user_data);  
static void on_combo_data1_changed(GtkComboBox *combo_id, gpointer user_data);  
static void on_combo_direction_changed(GtkComboBox *combo_id, gpointer user_data);  
  
static void on_check_autosync_toggled(GtkToggleButton *togglebutton, gpointer user_data);  
static void on_check_fastech_toggled(GtkToggleButton *togglebutton, gpointer user_data);
```

## Interface 함수

```
void library_interface();  
char *command_interface();  
char *FMM_interface(FMM_ERROR error);
```

명령어 구분  
데이터 처리

명령어 함수 호출

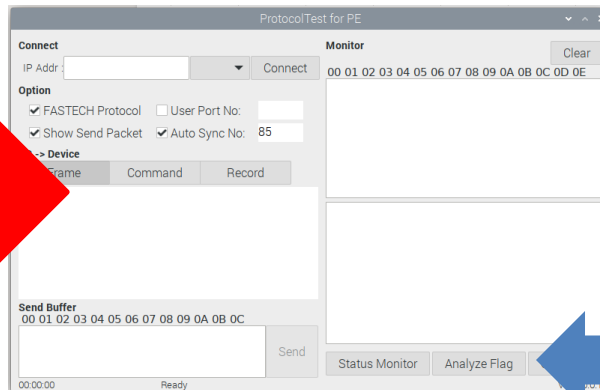
## 라이브러리(명령어) 함수

```
bool FAS_Connect(BYTE sb1, BYTE sb2, BYTE sb3, BYTE sb4, int iBdID);  
bool FAS_ConnectTCP(BYTE sb1, BYTE sb2, BYTE sb3, BYTE sb4, int iBdID);  
void FAS_Close(int iBdID);  
int FAS_ServoEnable(int iBdID, bool bOnOff);  
int FAS_MoveOriginSingleAxis(int iBdID);  
int FAS_MoveStop(int iBdID);  
int FAS_MoveVelocity(int iBdID, DWORD IVelocity, int iVelDir);  
int FAS_GetboardInfo(int iBdID, BYTE pType, LPSTR LpBuff, int nBuffSize);  
int FAS_GetMotorInfo(int iBdID, BYTE pType, LPSTR LpBuff, int nBuffSize);  
int FAS_GetEncoder(int iBdID, BYTE pType, LPSTR LpBuff, int nBuffSize);  
int FAS_GetFirmwareInfo(int iBdID, BYTE pType, LPSTR LpBuff, int nBuffSize);  
int FAS_GetSlaveInfoEx(int iBdID, BYTE pType, LPSTR LpBuff, int nBuffSize);  
int FAS_SaveAllParameters(int iBdID);  
int FAS_ServoAlarmReset(int iBdID);  
int FAS_EmergencyStop(int iBdID);  
int FAS_GetAlarmType(int iBdID);
```

GUI 실행

연결된  
Callback 호출

Frame  
송수신



송신 Frame 생성