

페스트가드

황 인기, 송 가람, 설 유승, 신 경임

목차

1. 개발 배경
2. 개발 목표
3. 시스템 개요
4. 하드웨어 소개
 - A. Jetson Nano
 - B. 터틀봇
 - C. 터틀봇 구동 프로세서
5. 소프트웨어 소개
 - A. ROS
 - B. OpenCV
 - C. 안드로이드 스튜디오
 - D. QtCreator
 - E. TCP Server
6. 구현 결과
 - A. 구동 시연
 - B. 애플리케이션
 - C. 시스템 통신
 - D. 해충 탐지 알고리즘
7. 구성도
 - A. 부품 리스트
 - B. 회로도
 - C. 시스템 구성도
8. 결과 및 성과
9. 향후 발전 방향
10. 참고 자료

1. 개발 배경

산업화와 도시화의 가속화로 인해 많은 인구가 도시로 집중되면서, 주거 공간은 고층 건물과 공동 주택 형태로 변화하였습니다. 이러한 밀집된 주거 환경은 위생 관리의 어려움을 증가시키며 해충의 발생을 증가시키는 결과를 가져오게 되었습니다.

한반도 아열대화에 '병원 덩어리' 바퀴벌레 급증... 4년전보다 50% ↑

활동 시기 늘고 번식 속도도 빨라져
서식지 파괴에 집안 침범 증가도 한몫

정가람 기자 2017-08-06 12:24:48 사회일반

그림 1. 급증한 바퀴벌레 분석 기사

바퀴벌레는 높은 생존력과 번식력을 가진 해충으로 알려져 있습니다. 다양한 환경에서 살아남을 수 있는 능력과 빠른 번식 속도로 인해, 한번 발생하면 개체 수가 급격히 증가할 수 있습니다. 특히 가정집에서 흔히 발견되는 독일바퀴와 미국바퀴는 실내 환경에 적응하여 서식합니다.

전통적인 해충 방제 방법은 주로 인력에 의존하며, 이는 시간과 비용이 많이 소요되는 단점이 있습니다. 또한, 사람이 직접 방제 작업을 수행하는 경우, 해충의 활동 시간과 겹치지 않아 효과적인 방제가 어려울 수 있습니다. 이러한 한계를 극복하기 위해 자동화된 시스템의 필요성이 대두되었습니다.

저희는 이미지 프로세싱과 로봇 기술을 결합하여 실시간으로 해충의 움직임을 감지하고, 해당 위치에 정확하게 약제를 분사하는 시스템을 제작하고자 하였습니다. 이 결과로 완성된 것이 페스트가드이며, 주거 공간의 위생 수준을 향상시키고 해충으로 인한 불편함을 최소화하는 것을 목표로 합니다.

2. 개발 목표

공공시설/가정집을 위한 무인 해충 방제 시스템 구축

사람이 직접 해충을 관리하기 어려운 공공시설(학교, 병원, 창고, 식당, 호텔 등)과 가정집을 대상으로 효율적인 무인 방제 시스템을 제공합니다. 기존 방제 방식과 달리 24시간 실시간 감시와 즉각적인 대응이 가능하여 보다 효과적인 해충 관리를 실현합니다.

경제적인 월 구독료 서비스 제공

고가의 방제 로봇이 초기 비용 부담으로 인해 일부 고객에게 접근성이 낮아지는 문제를 해결하기 위해, 월 구독료 기반의 서비스 모델을 도입합니다. 이를 통해 고객은 초기 투자 비용 없이 페스트가드를 사용할 수 있으며, 정기적인 유지보수 및 소프트웨어 업데이트까지 포함된 서비스를 이용할 수 있습니다.

다양한 해충 방제 기능 확장

초기 개발 단계에서는 바퀴벌레를 주요 타겟으로 하지만, 점차적으로 모기, 개미, 진드기 등 다양한 해충을 감지하고 방제할 수 있도록 기능을 확장할 계획입니다. 이를 위해 다양한 센서 및 AI 알고리즘을 개선하여 각 해충의 움직임 패턴을 분석하고 최적의 방제 전략을 적용할 것입니다.

모든 실내 시설의 해충 완전 박멸

페스트가드의 최종 목표는 사람이 머무는 모든 공간에서 해충으로 인한 불편함을 완전히 제거하는 것입니다. 단순한 방제 로봇을 넘어, 스마트 방역 시스템으로 발전시켜 AI 기반 자동 모니터링, 해충 분석 데이터 축적 및 맞춤형 방제 솔루션 제공까지 가능하도록 만들 계획입니다.

3. 시스템 개요

페스트가드는 무인 해충 방제 로봇 시스템으로, 사람이 직접 개입하지 않아도 자동으로 실내의 바퀴벌레를 탐지하고 효과적으로 박멸하는 기능을 수행합니다. 벌레의 주요 이동 경로를 파악 후 경로에 약을 분사하는 것을 목표로 합니다. ROS(Robot Operating System) 기반의 터틀봇 Burger 모델을 활용하여, Jetson Nano, Raspberry Pi 4, OpenCR 보드, OpenCV, Android 앱, Qt, MariaDB 등 다양한 기술을 접목하여 설계되었습니다.













	Main Server	Device	Camera	ROS
Board				
	Jetson nano	Raspberry pi, OpenCR	Jetson nano	Jetson nano
Language				
	C	C	C++	C++
OS	 ubuntu	 ubuntu ARDUINO	 ubuntu	 docker ubuntu
	Ubuntu 20.04	Ubuntu 20.04, Arduino	Ubuntu 20.04	Docker, Ubuntu 20.04

그림 2. 본 프로젝트에서 사용된 보드들과 언어, OS

바퀴벌레 탐지 및 좌표 전송

1. Jetson Nano에 연결된 카메라가 실시간으로 환경을 촬영
2. OpenCV를 활용한 영상 처리 기술로 바퀴벌레를 감지하고 좌표를 획득
3. 감지된 좌표 정보를 서버를 통해 ROS의 ronnClient로 전송

로봇 이동 및 방제 수행

1. ROS Client는 좌표와 명령을 ROS 토픽 통신을 통해 터틀봇에 전달
2. 명령을 수신한 터틀봇은 라즈베리 파이 4, OpenCR 보드로 구동되어 이동 수행
3. 목적지 도달 시, OpenCR 보드에서 방제 약물을 자동으로 분사
4. 이후 다시 대기 장소로 복귀하고 방제 성공 및 복귀 성공 여부를 DB에 기록

원격 조작 및 로그 기록

1. 사용자는 안드로이드 앱과 Qt 프로그램을 활용하여 로봇 원격 제어 가능
2. Maria DB를 이용한 로그 관리 시스템으로 감지된 해충 기록과 로봇의 동작 내역 저장
3. 해충 출현 패턴 및 효과적인 방제 전략 수립을 위한 데이터 확보

4. 하드웨어 소개

A. Jetson Nano



Jetson Nano는 NVIDIA에서 개발한 소형 컴퓨팅 보드로, 딥러닝, 컴퓨터 비전, 로봇틱스 등의 엣지 컴퓨팅 작업을 수행할 수 있는 강력한 성능을 갖춘 개발 플랫폼입니다. 저전력(5~10W) 환경에서도 CUDA, TensorRT, OpenCV, ROS 등의 다양한 프레임워크를 활용할 수 있어 인공지능 및 로봇 개발에 최적화된 보드입니다.

Jetson Nano는 페스트가드 시스템에서 해충 탐지를 담당하는 핵심 하드웨어로서, 이미지 프로세싱을 수행하여 해충을 탐지하고 ROS 기반 로봇과의 연계를 통해 완전 자동화된 무인 방제 시스템을 구현하는 데 중요한 역할을 합니다. 추가로 CUDA Core가 내장되어 있어, 추후 딥러닝을 통한 해충 인식 정확도 개선 작업 수행시 큰 도움이 될 것입니다.

표 1. Jetson Nano 주요 사양

CPU	ARM A57 쿼드 코어 @ 1.43GHz
GPU	128-코어 Maxwell CUDA
메모리	4GB LPDDR4
저장 공간	microSD 카드 슬롯
인터페이스	HDMI, DisplayPort, USB 3.0, USB 2.0, GPIO, I2C, SPI
소비 전력 (W)	5~10
크기 (W D H, mm)	100, 80, 29

B. 터틀봇

TURTLEBOT3



TurtleBot3 Burger는 ROBOTIS에서 개발한 오픈소스 로봇 플랫폼으로, ROS(Robot Operating System) 기반으로 동작하는 대표적인 연구용 및 교육용 로봇입니다. 소형, 경량이면서도 확장성이 뛰어나 SLAM, 내비게이션, AI 기반 자율주행 등의 다양한 로봇 연구에 활용이 가능합니다.

터틀봇은 페스트가드 프로젝트에서 바퀴벌레를 탐지한 후 해당 위치로 이동하여 약을 분사하는 핵심 이동 플랫폼 역할을 수행합니다. ROS 내비게이션을 활용하여 경로를 계산하고 자동으로 이동하며, LDS-02 라이다 센서를 활용해 실내 환경을 매핑(SLAM)하여 장애물을 피하면서 이동합니다. 사용자는 안드로이드 앱과 Qt 기반 인터페이스를 통해 필요시 수동 조작도 가능합니다. 이러한 기능들을 통해 Jetson Nano와 연계하여 완전한 무인방제 시스템을 구현할 수 있었습니다.

표 2. 터틀봇 주요 사양

메인보드	라즈베리 파이 4 / OpenCR 1.0
전원	11.1V 리튬 폴리머 배터리
센서	LDS-02 라이다 센서, IMU 6 축 관성 센서
모터	Dynamixel XL-430-W250-T
주행 방식	2륜 차동 구동
OS	Ubuntu 20.04
크기 (W D H, mm)	138, 178, 192

C. 터틀봇 구동 보드

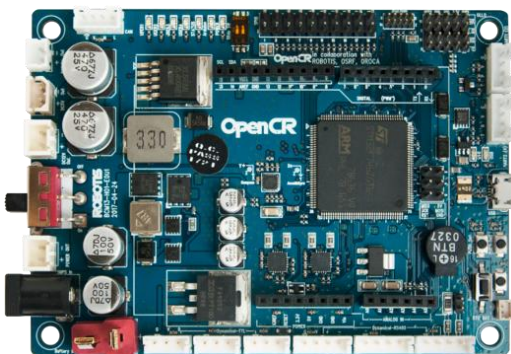
라즈베리 파이 4



라즈베리 파이 4는 ARM 기반의 소형 싱글 보드 컴퓨터로, 강력한 성능과 다양한 입출력 포트를 제공합니다. 이러한 장점에 힘입어 로봇 베어, IoT, 임베디드 시스템 개발 등에 널리 사용됩니다.

터틀봇에서 라즈베리 파이는 두뇌와 같습니다. Ubuntu 20.04 환경에서 ROS Noetic을 구동하며, Jetson Nano 및 OpenCR과 연동하여 터틀봇을 제어합니다. ROS 토픽을 통해 수신된 바퀴벌레의 좌표로 이동 명령을 수행하면서 라이다 센서와 관성 센서 등으로 터틀봇의 이동 경로를 계산합니다.

OpenCR 보드



OpenCR 보드는 터틀봇3에 최적화된 임베디드 제어 보드입니다. 터틀봇의 모터와 센서를 직접 제어하는 역할을 맡으며, UART/USB를 통해 상단의 라즈베리 파이와 통신합니다. 지정된 좌표로 터틀봇이 이동할 수 있도록 속도와 방향을 조절합니다.

각종 주변 장치를 직접 제어할 수 있으며, 이를 활용하여 약제 분사 등을 자동화할 수 있습니다.

5. 소프트웨어 소개

A. ROS (Robot operating System)



ROS(Robot Operating System)는 로봇 소프트웨어 개발을 위한 오픈소스 프레임워크로, 모듈화된 구조와 강력한 통신 시스템을 제공하여 다양한 로봇 애플리케이션을 개발할 수 있도록 지원합니다. 페스트가드 프로젝트에서는 TurtleBot3의 핵심 운영체제로 사용되며, 로봇의 자율 이동, 센서 데이터 처리, 약제 분사 기능을 구현하는 데 필수적인 역할을 합니다.

페스트가드에서는 ROS Noetic 버전을 사용하였으며, C++ 기반의 roscpp를 활용하여 개발하였습니다. ROS의 토픽 통신과 서비스 통신을 활용하여 바퀴벌레 탐지, 로봇 이동, 약제 분사 기능을 구현하였으며, RViz와 Gazebo를 이용하여 로봇의 동작을 시뮬레이션하고 시각화하여 개발을 효율적으로 진행할 수 있었습니다. ROS의 강력한 모듈화 기능과 통신 시스템은 페스트가드와 같은 복잡한 로봇 시스템을 효율적으로 개발할 수 있게 해주는 핵심 요소입니다. 또한 도커를 활용하여 어느 시스템에서나 간단하게 구동할 수 있도록 하였습니다.

코드 참고: https://github.com/Azruine/ros_docker

B. OpenCV (Open Computer Vision Library)



OpenCV는 컴퓨터 비전 및 영상 처리에 특화된 라이브러리로, 다양한 이미지 처리 알고리즘을 지원하며 이미지 변환, 객체 검출, 머신 러닝 등 다방면으로 활용할 수 있습니다.

페스트가드에서는 OpenCV 4.8.0 버전을 활용하였으며 노이즈 제거, 배경 추출, 이진화, 객체 검출 등의 다양한 알고리즘을 활용하였습니다.

코드 참고: https://github.com/Azruine/cpp_camera_project

C. 안드로이드 스튜디오



안드로이드 스튜디오는 구글에서 제공하는 공식 안드로이드 앱 개발 환경(IDE)으로, Java 및 Kotlin을 사용하여 앱을 개발할 수 있습니다. Java는 안드로이드 앱 개발에서 가장 많이 사용되는 언어 중 하나로, 객체 지향 프로그래밍(OOP)개념을 기반으로 합니다.

코드 참고: <https://github.com/GaramSong-95/TcpTelnet>

D. QtCreator



QtCreator는 Qt 프레임워크 기반의 크로스 플랫폼 애플리케이션 개발을 위한 IDE입니다. C++ 및 QML 등의 언어를 사용하여 앱을 개발할 수 있습니다. 페스트가드 로봇의 조작과 관리를 위한 데스크톱 애플리케이션을 개발하기 위해 QtCreator를 활용하였습니다.

코드 참고: https://github.com/GaramSong-95/AiotClient_tab5

안드로이드 앱과 Qt 프로그램은 모두 TCP 소켓 통신을 통해 젯슨 나노 서버와 연결되며, 서버로부터 데이터를 수신하고 명령을 전송합니다. 이를 통해 사용자는 언제 어디서든 로봇의 상태를 확인하고 제어할 수 있으며, 해충 탐지 및 방제 기록을 효율적으로 관리할 수 있습니다. 또한 카메라가 촬영중인 화면이나 해충 경로 감지 화면을 손쉽게 열람할 수 있습니다.

E. TCP 서버

TCP(Transmission Control Protocol)는 인터넷 프로토콜 스위트의 핵심 프로토콜 중 하나로, 두 장치 간 신뢰성 있는 데이터 전송을 보장합니다. TCP는 연결 지향적이며, 데이터 전송 전 연결을 설정하고, 데이터 손실이 없도록 패킷을 추적하며, 순서를 보장하는 특징이 있습니다.

페스트가드 시스템에서는 젯슨 나노에 TCP 서버를 구축하여 다양한 구성 요소 간의 통신을 관리합니다. 이 서버는 다음과 같은 핵심 역할을 수행합니다:

- 터틀봇 제어 명령 수신 및 전달: 사용자가 앱이나 Qt 애플리케이션을 통해 보낸 이동 명령을 수신하여 터틀봇에 전달합니다.

- 센서 데이터 수집 및 배포: 터틀봇의 각종 센서로부터 수집된 데이터를 처리하고 클라이언트 애플리케이션에 실시간으로 전송합니다.
- 영상 스트리밍 중계: OpenCV로 처리된 카메라 영상을 앱과 Qt 프로그램으로 전송하여 사용자가 원격으로 상황을 모니터링할 수 있게 합니다.
- 해충 탐지 정보 관리: OpenCV를 통해 탐지된 해충 정보를 저장하고, 이를 클라이언트 애플리케이션에 통보합니다.

서버는 멀티스레딩을 사용하여 여러 클라이언트의 동시 접속을 처리하며, JSON 형식의 데이터 패킷을 주고받아 명령과 상태 정보를 교환합니다. 이러한 TCP 서버 구조를 통해 페스트가드 시스템의 각 구성 요소가 유기적으로 연결되어 원활한 해충 탐지 및 방제 작업이 가능해집니다.

The image shows two terminal windows side-by-side. The left window shows the server starting: 'jetson@nano13:~/workspace/iot_server\$.', followed by 'IoT Server Start!!', and then a message from the client: '[USER] New connected! (ip:10.10.141.133 msg : [USER->ALLMSG] Hello, World!'. The right window shows the client's perspective: 'jetson@nano13:~/workspace/iot_server\$ b', followed by 'Input a message! [ID]msg (Default ID:AL [USER] New connected! (ip:10.10.141.133 Hello, World! [USER]Hello, World!'. Both windows have a terminal icon and the IP address '10.10.141.133' in the title bar.

그림 3. TCP 서버 구동 및 접속

코드 참고: https://github.com/Azruine/iot_server

6. 구현 결과

A. 구동 시연

젯슨 나노에서 roscore를 실행합니다. 이는 ROS 마스터 역할을 수행합니다.

다음으로 터틀봇에서 bringup을 시행하여 로봇의 기본 기능을 활성화합니다.

The image shows a terminal window with the output of the 'bringup' command. It starts with '[INFO] [1743141030.319089]: Setup publisher on battery_state [sensor_msgs/BatteryState]', followed by '[INFO] [1743141030.335735]: Setup publisher on magnetic_field [sensor_msgs/MagneticField]', and '[INFO] [1743141030.420720]: Setup service server on bot3_gpio [bot3_kccisc_serv...ice/bot3gpio]'. It then shows '[INFO] [1743141030.564189]: Setup publisher on /tf [tf/tfMessage]', '[INFO] [1743141030.585630]: Note: subscribe buffer size is 1024 bytes', '[INFO] [1743141030.597432]: Setup subscriber on cmd_vel [geometry_msgs/Twist]', '[INFO] [1743141030.617522]: Setup subscriber on sound [turtlebot3_msgs/Sound]', '[INFO] [1743141030.639978]: Setup subscriber on motor_power [std_msgs/Bool]', '[INFO] [1743141030.672350]: Setup subscriber on reset [std_msgs/Empty]', '[INFO] [1743141032.837622]: Setup TF on Odometry [odom]', '[INFO] [1743141032.850750]: Setup TF on IMU [imu_link]', '[INFO] [1743141032.859024]: Setup TF on MagneticField [mag_link]', '[INFO] [1743141032.868100]: Setup TF on JointState [base_link]', '[INFO] [1743141032.895709]: -----', '[INFO] [1743141032.908886]: Connected to OpenCR board!', '[INFO] [1743141032.915829]: This core(v1.2.6) is compatible with T33 Burger', '[INFO] [1743141032.925309]: -----', '[INFO] [1743141032.939373]: Start Calibration of Gyro', and finally '[INFO] [1743141035.400584]: Calibration End'.

그림 4. Bringup 수행

다음으로 젯슨 나노에서 로봇 네비게이션을 수행합니다.

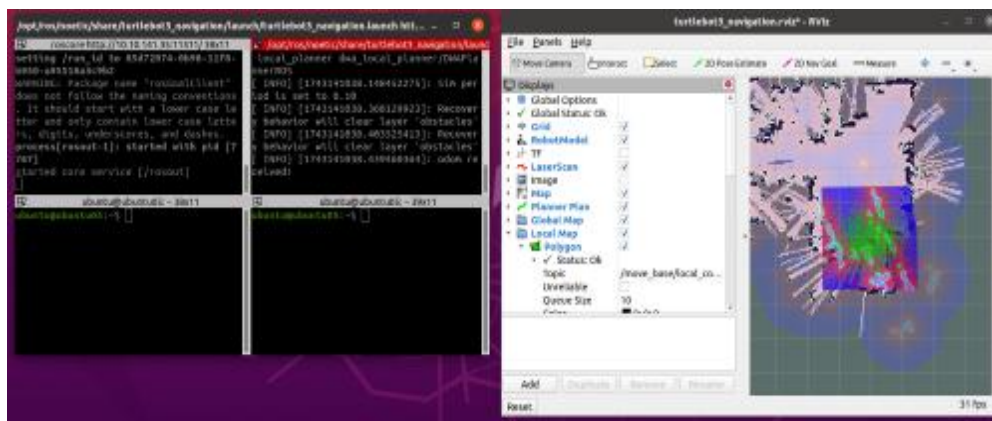


그림 5. roscore 구동(좌측), 네비게이션 구동(우측)

위 모든 과정이 완료된 후, 로봇에서 rosrunc를 수행하여 ROS 클라이언트를 서버에 접속하게끔 합니다.

```
ubuntu@ubuntu05:~/catkin_ws$ rosrunc rosGoalClient rosGoalClient 10.10.141.35 5000 ROS
Input a message! [ID]msg (Default ID:ALLMSG)
MYSQL startup
Connection Successful!

[ROS] New connected! (ip:10.10.141.35,fd:4,sockcnt:1)
```

그림 6. ROS 클라이언트 접속

젯슨 나노는 실시간으로 화면을 분석하여 해충으로 의심되는 객체를 탐지하고, 좌표를 획득하여 로봇에게 전송합니다.

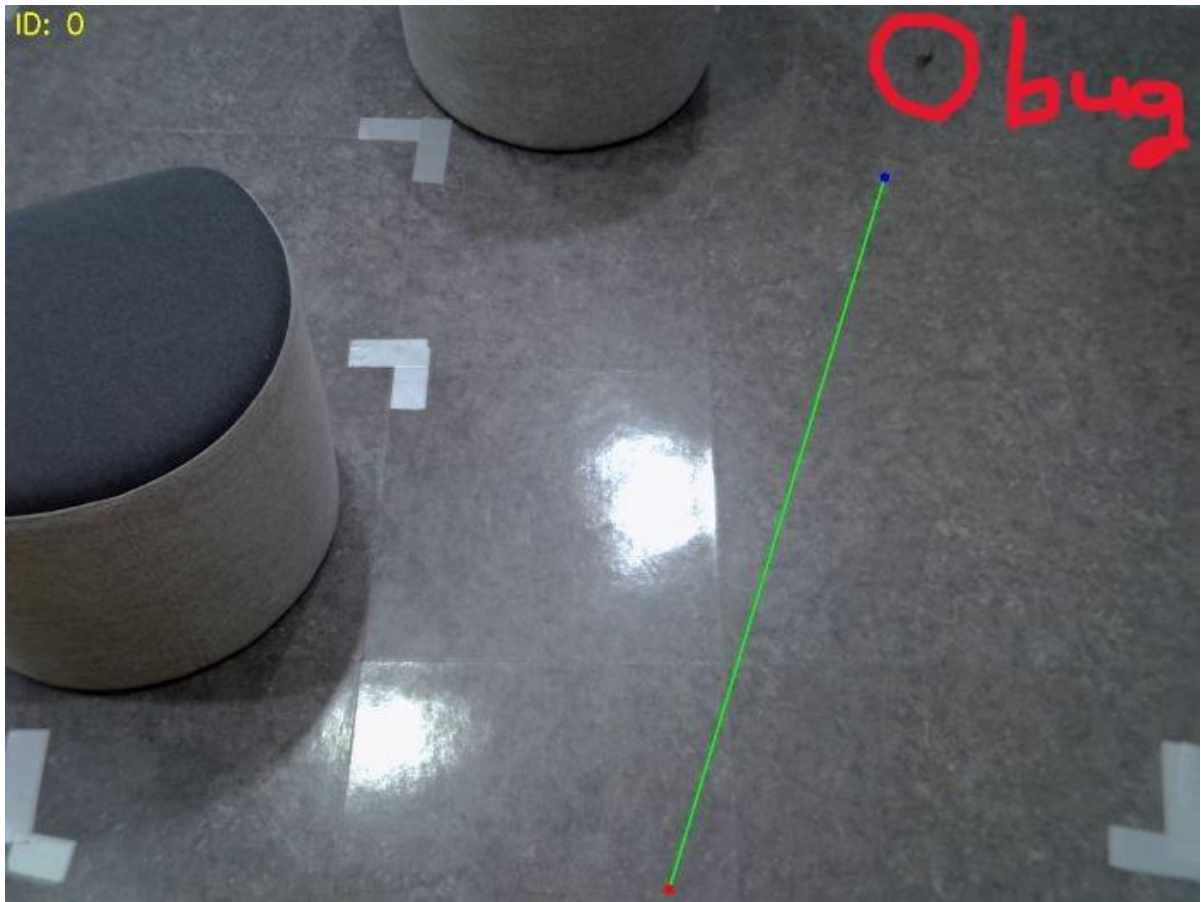


그림 7. 젯슨 나노에서 감지된 해충 및 그 경로

```
jetson@nano13:~$ ./workspace/cpp_camera_project/build/release/bin/camera_app 50 10000
Using min area: 50, max area: 10000, min path distance: 300
Connecting to 10.10.141.132:5000 as 'JETSON'...
Connected to server successfully!
Server: [JETSON] New connected! (ip:10.10.141.132,fd:8,sockcnt:4)

[ WARN:001.131] global cap_gstreamer.cpp:1728 open OpenCV | GStreamer warning: Cannot query video position: status=0, values=-1, durations=-1
Camera opened successfully.
Press 'q' to quit.
Camera resolution: 1280x960
TCP client using resolution: 1280x960
HTTP server started on port 8080
Initial image saved: images/initial_20250328_072627.jpg
Object 0 ignored (distance: 0 < 300)
Object 1 path: Start(771,955) -> End(1081,667) Distance: 423.136
Motion detection image saved: images/motion_1_20250328_072712.jpg
Sent to server: [ROS]GOGOL@2.2001.0000.40

Server: [ROS]MISSIONCOMPLETE

Get response
Object 2 path: Start(1246,515) -> End(1191,170) Distance: 341.459
Motion detection image saved: images/motion_2_20250328_072820.jpg
Sent to server: [ROS]GOGOL@2.4101.2000.70

Server: [ROS]MISSIONCOMPLETE

Get response
Object 4 ignored (distance: 142.006 < 300)
Object 5 ignored (distance: 20.2485 < 300)
Object 6 ignored (distance: 0 < 300)
```

그림 8. 젯슨 나노로부터 ROS 클라이언트에게 좌표 전송



그림 9. 전체 구성도 및 시연 시작

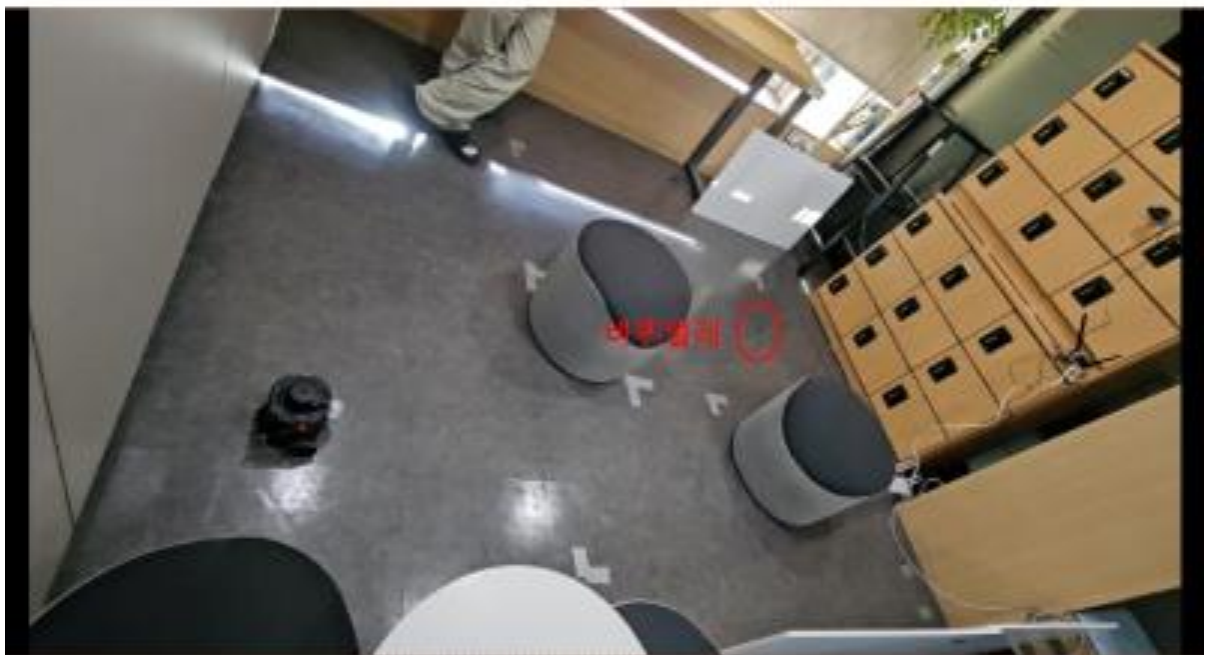


그림 10. 바퀴벌레 이동 감지



그림 11. 좌표 전송 후 목표 지점으로 이동 시작



그림 12. 방제 시작



그림 13. 방제 완료 후 초기 위치로 복귀

B. 애플리케이션

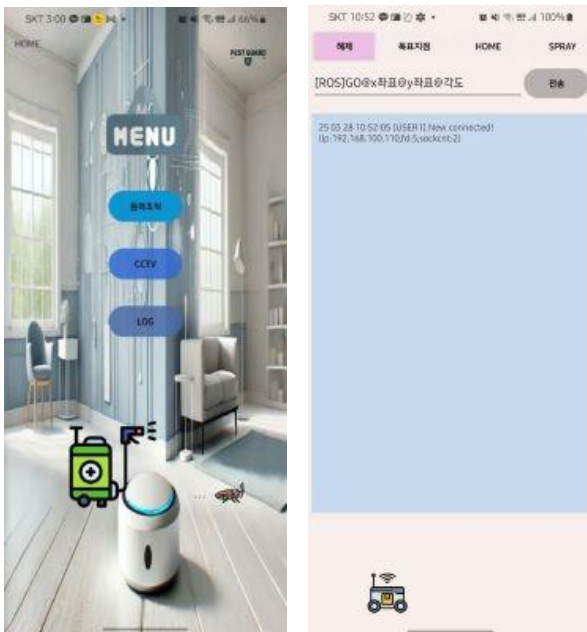


그림 14. 안드로이드 앱 메인 메뉴

앱 실행 후, 총 3개의 버튼으로 구성된 메인 화면이 나타납니다. 각 버튼을 통해 로봇 원격 조작, CCTV 영상 확인, 로그 데이터 조회기능을 수행할 수 있습니다.

원격 조작 기능

"원격 조작" 버튼을 클릭하여 서버 로그인 화면으로 이동합니다. "연결" 버튼을 클릭하면 자동으로

저장된 아이디로 로그인합니다. 서버 연결 후, 터틀봇을 조작할 수 있는 다양한 기능을 제공합니다.

목표 좌표 이동

"목표지점" 버튼 클릭→ 명령 입력창 자동 생성

명령 입력창에는 [ROS]GO@x좌표@y좌표@각도형식으로 명령어 입력

x좌표: 목표 위치의 X 좌표, y좌표: 목표 위치의 Y 좌표, 각도: 터틀봇이 이동 후 회전할 각도, "전송" 버튼 클릭→ 터틀봇 클라이언트에 전송 (TCP 통신), 터틀봇은 받은 좌표를 기반으로 해당 위치로 이동

원점 복귀 (HOME 기능)

"HOME" 버튼 클릭→ 터틀봇이 초기 위치로 복귀

스프레이 작동

"SPRAY" 버튼 클릭→ 터틀봇에 장착된 OpenCR 보드를 통해 약 분사 (ROS 서비스 통신)

CCTV 화면 기능 (해충 탐지 이미지 확인)

메인 화면에서 "CCTV" 버튼을 클릭하여 바퀴벌레 탐지 화면으로 이동합니다.

기본적으로 초기 화면은 빈 상태이며, 바퀴벌레가 탐지되면 자동으로 화면이 업데이트됩니다. 젯슨 나노에서 해충 탐지 시 캡처된 이미지를 HTTP 서버로 업로드하여 사용자가 확인할 수 있습니다.



그림 15. 안드로이드 앱 CCTV 화면

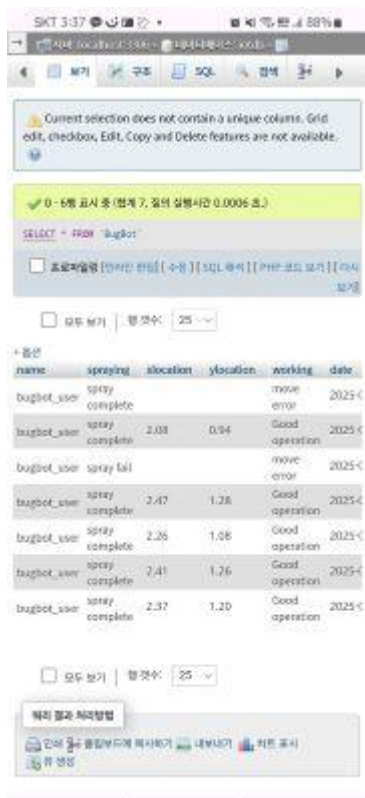


그림 16. 안드로이드 앱 로그 화면

메인 화면에서 "LOG" 버튼 클릭→ PHP 기반의 로그 화면으로 이동, MariaDB에 저장된 데이터를 조회하여, 터틀봇의 상태 및 약 분사 이력을 확인 가능합니다.

Qt 앱 화면



그림 17. Qt 서버 화면

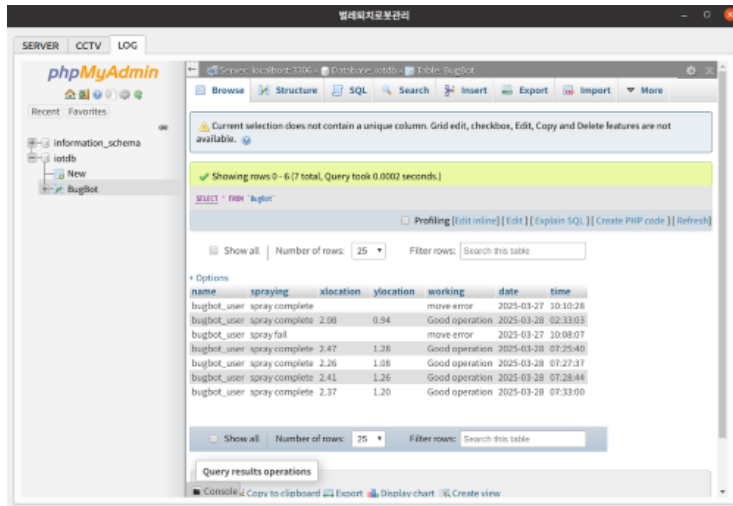


그림 18. Qt 로그 화면

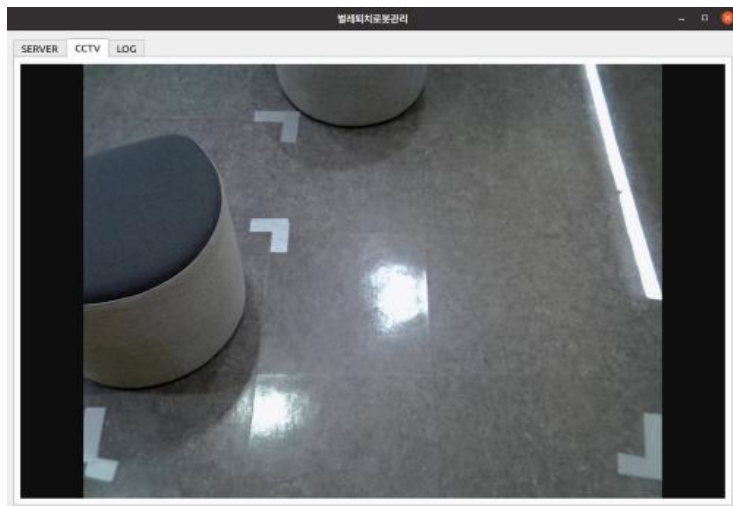


그림 19. Qt CCTV 화면

C. 해충 탐지 알고리즘

해충 탐지는 다음과 같은 과정을 거칩니다.

1. 원본 프레임 획득 및 GaussianBlur 적용, KNN 배경 제거

노이즈 등을 제거하기 위해 전체 프레임에 가우시안 블러를 적용합니다. 다음, KNN(K-Nearest Neighbor) 알고리즘을 활용하여 배경을 제거하고 객체의 경계선을 추출합니다.

```
1 // Apply background subtraction
2 cv::Mat processed_frame;
3 cv::GaussianBlur(frame, processed_frame, cv::Size(5, 5), 0);
4 bg_subtractor->apply(processed_frame, foreground_mask);
```

그림 20. 노이즈 제거 및 배경 제거

2. 모폴로지 연산 및 이진화

배경이 제거된 경계선 이미지에서 작은 잡음과 구멍을 제거합니다. 이진화를 거쳐 객체를 강조한 최종 마스크를 생성합니다.

```
1 // Apply morphological operations to remove noise
2 cv::Mat kernel =
3     cv::getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(5, 5));
4 cv::morphologyEx(foreground_mask, foreground_mask, cv::MORPH_OPEN, kernel);
5 cv::morphologyEx(foreground_mask, foreground_mask, cv::MORPH_CLOSE, kernel);
6
7 cv::threshold(foreground_mask, foreground_mask, 200, 255,
8               cv::THRESH_BINARY);
```

그림 21. 모폴로지 연산과 이진화

3. 컨투어 검출

이진화된 마스크에서 컨투어를 검출하고, 설정된 면적 조건에 따라 객체를 필터링합니다. 다음으로, 객체들을 적절히 병합하여 최종 결과를 얻습니다.

```
1 // Find contours in the foreground mask
2 std::vector<std::vector<cv::Point>> contours;
3 cv::findContours(foreground_mask, contours, cv::RETR_EXTERNAL,
4                 cv::CHAIN_APPROX_SIMPLE);
5
6 // Filter contours by area and create bounding rectangles
7 std::vector<cv::Rect> motion_regions;
8 for (const auto& contour : contours) {
9     double area = cv::contourArea(contour);
10
11     // Filter objects by size using min_area_ and max_area_ parameters
12     if (area ≥ min_area_ && area ≤ max_area_) {
13         cv::Rect bbox = cv::boundingRect(contour);
14         motion_regions.push_back(bbox);
15     }
16 }
17
18 // merge overlapping boxes
19 motion_regions = mergeOverlappingBoxes(motion_regions);
20
```

그림 22. 컨투어 검출 및 객체 병합

4. 기존 객체와의 비교 및 사라짐 감지

기존 객체들과 비교하여 이동 경로를 추적하고, 새로이 생성된 객체인지 구별합니다.

7. 구성도

A. 부품 리스트

번호	품명	이미지	링크
1	Nvidia Jetson Nano Development Kit-B01		https://www.devicemart.co.kr/goods/view?no=12513656
2	Raspberry Pi 4 Model B 2GB		https://www.devicemart.co.kr/goods/view?no=12234533
3	OPENCN 1.0		https://www.devicemart.co.kr/goods/view?no=13145676
4	Turtlebot Burger RPi4 2GB		
5	로지텍 HD Webcam C270		
6	분사기		

B. 회로도

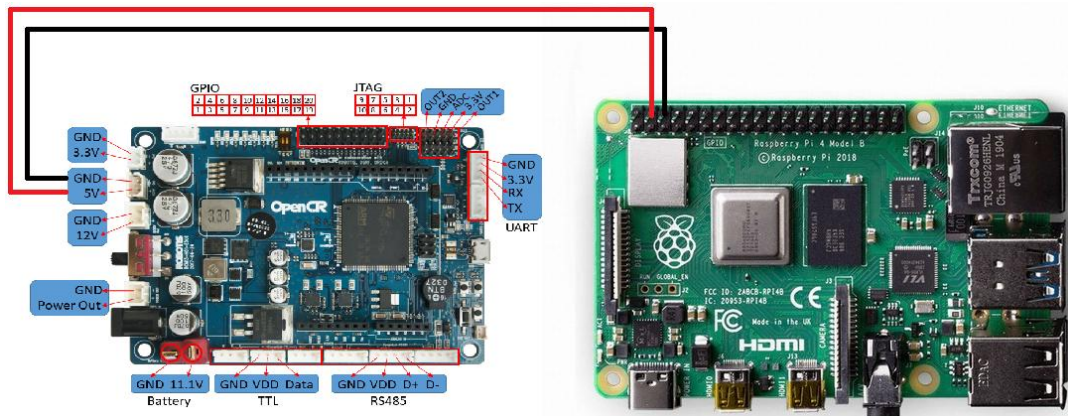


그림 23. 터틀봇의 OpenCR, 라즈베리 파이 4 회로도

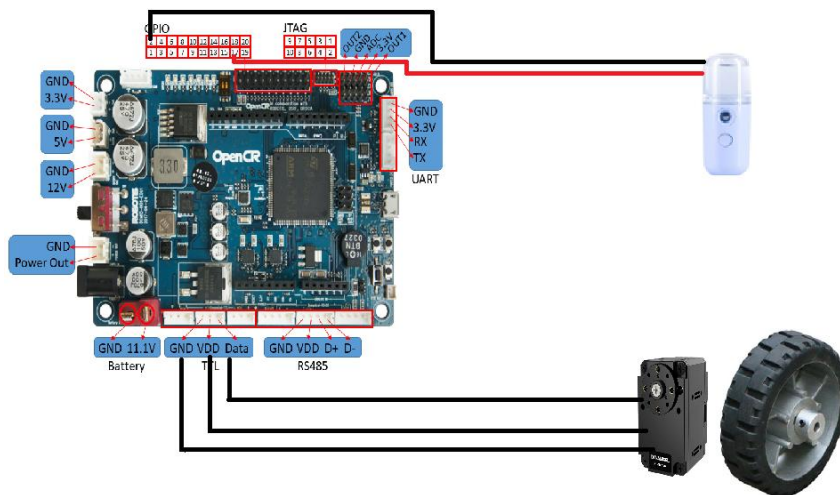


그림 24. OpenCR 회로도

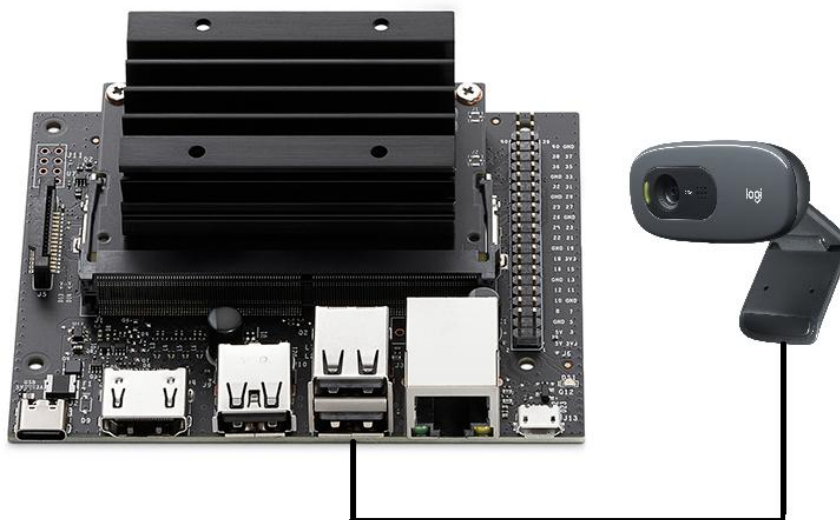


그림 25. 젯슨 나노 회로도

C. 시스템 구성도

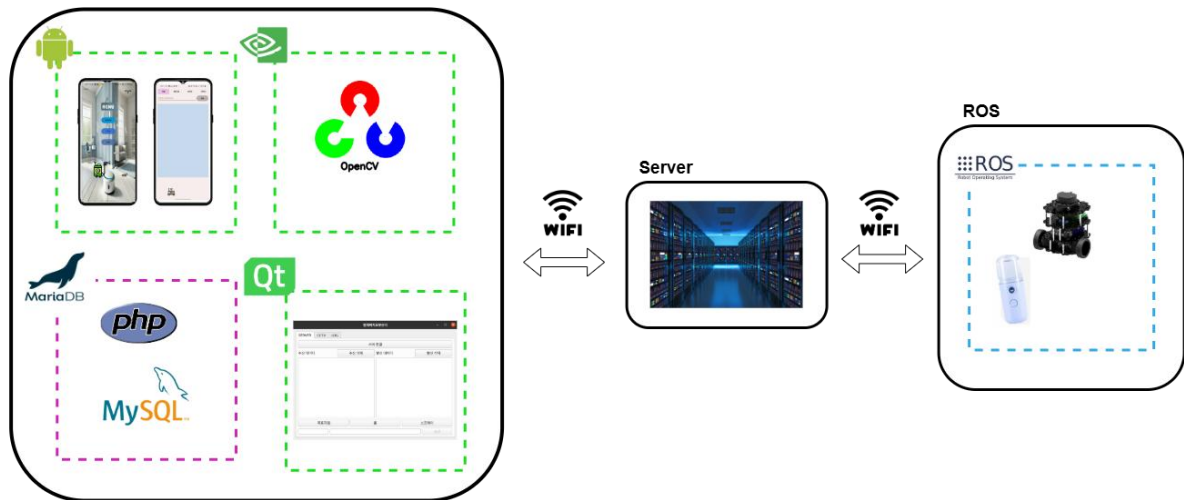


그림 26. 전체 시스템 구성도

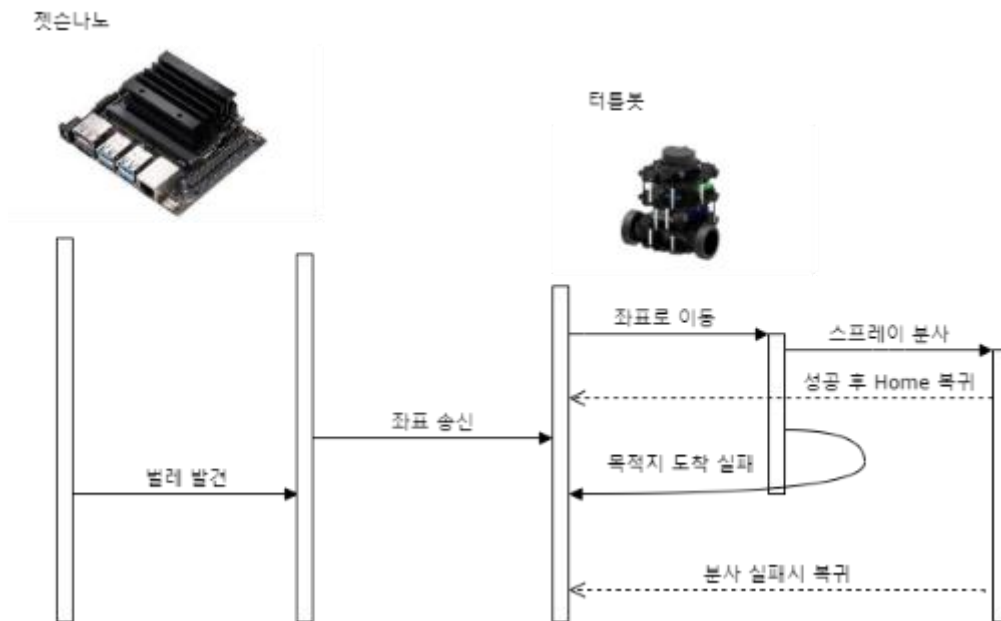


그림 27. 구동 과정 모식도

8. 결과 및 성과

자동 해충 탐지 및 퇴치 기능 구현

OpenCV를 활용하여 배경 제거 후 해충 객체만 인식하는 알고리즘을 개발하였으며, ROS 기반으로 터틀봇을 제어하여 해충이 발견된 위치로 이동하고 OpenCR을 활용해 정밀한 약 분사 기능을 적용하였습니다. 특히 벌레의 주요 이동경로를 파악한 후 해당 경로에 약을 분사하는 방식으로 효율적인 방제가 가능해졌습니다.

실시간 원격 조작 및 데이터 기록 시스템 구축

Qt 기반 PC 프로그램과 Android 앱을 활용하여 사용자가 원격으로 로봇을 조작할 수 있도록 하였으며, MariaDB와 연동하여 해충 탐지 이력, 약 분사 기록, 로봇 상태 정보를 로그로 저장하고 조회할 수 있는 시스템을 구현하였습니다. 이를 통해 해충 발생 패턴 분석과 방제 효과 평가가 가능해졌습니다.

경제적인 서비스 모델 확립

가정, 공공시설에서 활용 가능하도록 월 구독형 서비스 모델을 제안하였습니다. 초기 고가의 로봇 구매 비용 대신 월 구독료를 통해 페스트가드 서비스를 이용할 수 있도록 하여 비용 부담을 줄이고 접근성을 높였습니다. 이는 특히 다중이용시설이나 기업체에서 효율적인 해충 관리를 위한 경제적인 대안이 될 수 있습니다.

결론

페스트가드 프로젝트를 통해 자동화된 해충 방제 시스템의 가능성을 실험하고, 실질적인 적용이 가능한 수준까지 시스템을 구축하였습니다. OpenCV 기반 움직임 탐지를 활용하여 최대 초당 15 프레임의 처리 속도를 확보하였고, ROS 기반의 터틀봇과 다양한 소프트웨어 기술을 결합하여 완전한 무인 해충 방제 솔루션을 개발하였습니다. 이 시스템은 공공시설, 가정, 음식점, 창고 등 해충 관리가 어려운 공간에서 효과적으로 활용될 수 있을 것으로 기대됩니다.

9. 향후 발전 방향

페스트가드 시스템의 향후 발전 방향은 크게 네 가지 영역에서 추진될 예정입니다:

다양한 해충 탐지 확장 - 바퀴벌레, 모기, 개미, 진드기 등 인식

우선, 현재의 바퀴벌레 탐지 기능을 넘어 모기, 개미, 쥐, 진드기 등 다양한 해충을 인식하고 방제할 수 있는 기능을 개발할 계획입니다. 이를 위해 딥러닝 기반의 AI 탐지 모델을 도입하여 더 높은 정확도로 다양한 해충을 구분할 수 있도록 할 것입니다.

자율 탐색 및 매핑 고도화 - SLAM 기반 효율적 실내 정찰

두 번째로, SLAM(Simultaneous Localization and Mapping) 기술을 고도화하여 자율 탐색 및 매핑 기능을 강화할 예정입니다. 현재는 해충이 발견된 좌표로만 이동하는 방식이지만, 향후에는 로봇이 스스로 실내 공간을 탐색하며 해충을 탐지하고 방제하는 완전 자율 시스템으로 발전시킬 계획입니다.

클라우드 기반 데이터 분석 - 해충 패턴 인식 및 예측 모델

세 번째로, 클라우드 기반의 데이터 분석 시스템을 구축하여 해충 출현 패턴을 분석하고 예측하는 모델을 개발할 예정입니다. 수집된 데이터를 AI로 분석하여 해충 발생 가능성이 높은 시간과 장소를 예측하고, 선제적인 방제 전략을 수립할 수 있을 것입니다.

완전 자율 방제 시스템 - AI 기반 해충 분석 및 자동 최적 방제 전략

마지막으로, 이 모든 기술을 통합하여 완전 자율적인 방제 시스템으로 발전시키는 것이 목표입니다. AI 기반으로 해충을 분석하고, 최적의 방제 전략을 자동으로 수립하여 적용하는 스마트 방역 시스템으로 발전시켜 나갈 것입니다. 이를 통해 페스트가드는 단순한 로봇이 아닌, 해충 문제를 근본적으로 해결하는 혁신적인 솔루션으로 자리매김할 것입니다.

10. 참고 자료

- 정재곤, "안드로이드 앱 프로그래밍" 이지스퍼블리싱, 2021
- 서영진, "사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리파이" 제이펍, 2020
- 표윤석, 임태훈, "ROS 2로 시작하는 로봇 프로그래밍" 루비페이퍼, 2021
- 김성호, "Linux 쉘 스크립트 및 서버 구축 Embedded Linux 이해 및 활용" 월텍
- 김성호, "라즈베리파이 기반 Qt 프로그래밍" 월텍
- [TurtleBot 3 공식 Documents](#)
- 프로젝트 깃허브 링크: https://github.com/Azruine/project_pestguard