연구발표회

201911148 소프트웨어학과 강현우

목표 했던 것

- GPU 사용을 위한 기본적 베이스 마련(HW)
- 터틀봇 모니터링 시스템 구축 & 군집 운행 건.

ROS1에 적응

작업 목록	자세한 내용(ETC)
ROS1 패키지 구조 파악(Catkin WS)	초기 오류로 인터넷 검색 / 직접 해결
Teleop/Navigation + SLAM 주행	공식 문서 참고

하드웨어 교체

작업 목록	자세한 내용(ETC)
SBC 교체	Raspberry Pi3B+ → Jetson Nano

ROS2로 이동

- 교수님의 추가적인 조언과 개인적인 생각이 일치해 ROS2로 Migration 진행

  → ROS를 어차피 새로 배우는 상황과, ROS1보다 최신인 ROS2를 배우는 것이 더 좋다고 판단.

ROS2의 문제 (Main)

문제점	자세한 내용(ETC)
RViz2의 Crash / Segfault	Stack Memory Violation으로, SIGABRT시그널을 보내고 로그 없이 프로그램이 죽어버리는 현상이 발생함.
데이터 통신 에러	Host-Slave가 서로 통신을 못하는 상황이 발생함.

ROS2의 문제 (Debugging)

문제점	자세한 내용(ETC)
Host-Slave 간 데이터 통신 에러	Host-Slave ROS 버전이 서로 다르면 데이터 송/수신이 원활하지 않음.(Ref)
RViz2의 Crash / Segfault	없는 정보를 참조하려고 해 에러 발생.(Opened PR)

ROS2의 문제 (추가적인)

문제점	자세한 내용(ETC)
자율주행 패키지 호환성	ROS2는 Catkin구조가 아닌, Colcon 구조로 기본 컴파일 진행 불가
내부 라이브러리 미정립	Migration문서의 부재, 핵심 기능 누락 등 (밑의 비교 표 참고)

내용	ROS1	ROS2
Python 내부 라이브러리	ROSPY	RCLPY
패키지 구성 (구조 및 컴파일 방식)	Catkin Workspace	Colcon Build
네트워크 구성	Master – Slave	Multiple Master & Multiple Slave
호환성	N/A	ROS1 일부 기능만 가능 (일부는 아직 개발 안됨)

ROS2의 문제 (실질적 문제)

- ROS1 드라이버(자율주행 패키지)를 포팅하려면 기본적으로 같은 기능을 그대로 ROS2로 가져오는 것이 일반적.
- 노드 등록 방식, Host-SBC간 통신 방식 등 모두 갈아 엎어져 포팅이 실질적으로 불가능한 상황에 놓이게 됨.
- 따라서 Migration가이드를 참조했지만, ROS플랫폼 자체적으로 ROS1에서는 지원하고, ROS2는 지원하지 않는 Backward-Compatibility가 존재하지 않았음. 따라서 드라이버 포팅도 실질적으로 매우 어려움을 알게 됨.

ROS - Jetson

문제점	자세한 내용(ETC)
카메라 접속 일부 불가	Topic이 만들어졌지만, Publishing이 제대로 되질 않음

내용	Raspberry Pi	Jetson Nano
지원 방식	MMAL Supported → Cam	GStreamer → Cam

- Gstreamer에서 Locking/Free문제가 의심됨
  → 한번 작동되고, 재부팅 전까지는 캠이 나오지 않음.
- 정확하게 디버깅 하지 못해 100% 확실하지는 않지만, 추가적으로 다음 링크에 명시된 사항이 의심됩니다.(링크 **CSI-Issue 항목 참조**)

결론

- 따라서 실질적인 결과는 도출하지 못했음
- 그러나 이번 연구 프로젝트를 진행하며 ROS에 대한 기본적인 내용들을 경험할 수 있었음
- 추가적으로 문서 작성이 일반 사용자에게 미치는 영향을 가장 실질적으로 느끼게 되었음

2020
Summer - Winter

- Linux-Based Cloud Operating System

  → AWS/MS Azure에서 제공하는 클라우드 시스템의 구조적 내용을 공부해 보고, 이를 구현합니다.

- 대표적으로 자원 모니터링 / 자원 배분이 있으며, 현재는 이 두가지를 목표로 삼고 있습니다.

- 프로젝트의 이름은(가칭) CL-OS이며, 이에 대한 사이드 프로젝트, Jetson_Monitoring(Service), CLMonitoring은 미리 살짝 개발해 두었습니다.

2020
~ Summer

- 현재 일부 개발을 마친 Jetson_Monitoring, CLMonitoring을 조금 더 가꾸며, TODO 리스트 및 이에 관한 내용을 철저히 기록해 누적합니다.
- 기록 내용 중, 실제 구현에 있어서 어려운 점, 그리고 필요한 HW/기본 지식들을 모두 추가 기록해 계획 절차를 반복합니다.
- 마지막으로 순서를 정해 실제 구현 시에는 계획대로 최대한 흘러가게 계획합니다.

2020
~ Summer Break

- 필요 시, 시스템 프로그래밍에 대한 이론적 내용을 공부합니다.

  → Kernel Module Programming

- 이는 현재 Jetson_Monitoring 프로그램에 쓰일 것이고, sysfs로 따로 등록할 목적입니다.

2020
~ Winter(Break)

- 모니터링 시스템을 커널단으로 올리고, Node-Master화시켜 마스터 서버에서 바로 접속할 수 있도록 서버 구성;
- GPU자원 분배가 한 하드웨어에서 가능한지에 대한 여부 조사 및 계획 절차(이는 여름 계획과 동일합니다.)
- 모니터링 시스템을 일부 정립하고, 유저마다 할당된 노드에 SSH(Shell 명령)으로 컨트롤 할 수 있는 시스템을 구축합니다.