Prerequisite

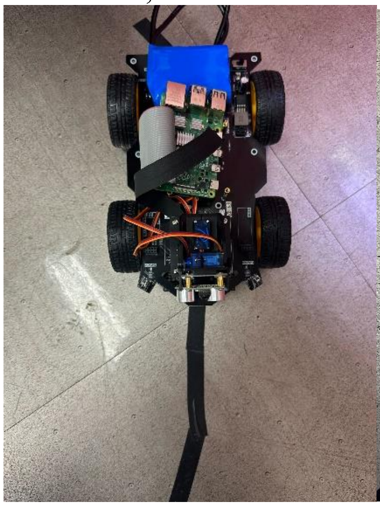
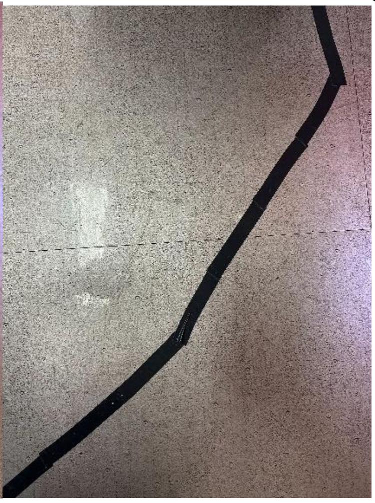
Before starting our project, you have to **change the kernel image**, as the required libraries are included in this new image.

1. Select a Raspberry Pi: Choose one unit from those available to your team. This selected Raspberry Pi will be utilized to operate a Raspbot for the duration of the semester.
2. Download the Kernel Image: Acquire a new kernel image from the following link: 0.Sytem_File (Pi4)
3. Replace the Kernel Image: Substitute the existing kernel image with the one you downloaded. You may use any software of your choice for this task, such as Etcher.
4. Test the New Image: Insert the SD card into the Raspberry Pi to verify if the new image functions correctly.

Project Part 1: Line Tracer with C

Soon, your team will start an engaging project involving a Raspbot. Before we delve into the main project, we will develop line tracing code in the C programming language, which will be integral to our upcoming project. In the first part of this project, we will use an **IR tracking sensor** to trace a **black line**. You are free to use **any C library** that suits your needs.

- Goal
  - Implementing a line tracing program with C. (not C++)
- Prerequisites
  - Set up your Raspbot following the provided guidelines.
  - Install the WiringPi library on your Raspberry Pi.
    (While not mandatory, this library is highly recommended for its utility.)
- Requirements
  - Your code must compile with the simple execution of the make command. This requires you to create a Makefile in the same directory as your source code.
  - The executable binary resulting from the compilation must be named linetracer. i.e., Your program will be executed with the command: "./linetracer."
- Notes
  - **Sensor Data Usage:** To utilize data from sensors, it is important to determine which GPIO port each sensor is connected to. This information can be obtained from the Raspbot's official website.
  - Code References: For additional insights, you may refer to Python code examples available at Yahboom's Raspbot study page: http://www.yahboom.net/study/Raspbot
  - For the final demo, we will use the common black insulation tape (절연테이프) to make a track, whose width is 19mm. Please refer to the images below.

Project Part 2: QRcode Recognition with C++

For this project part 2, you will implement QRcode Recognition using C++ language. **You will use the opencv library to implement code.** You will use the Raspbot camera to perform this assignment.

- Goal
  - Implement a QR code recognition program using C++ (not C).
  - Successfully read information from QR codes.
- Prerequisites
  - Understand the basic components of QR codes.
  - Have access to a camera sensor on the Raspbot.
  - Install OpenCV (version 4.0.0 or higher).
  - Refer to the Raspbot's official website for guidance if you encounter difficulties.
    http://www.yahboom.net/study/Raspbot
- Requirements
  - Your code must compile with the simple execution of the make command. This requires you to create a Makefile in the same directory as your source code.
  - The executable binary resulting from the compilation must be named qrrecognition. i.e.,
    Your program will be executed with the command: "./qrrecognition."
- Notes
  - After following the installation instructions for the package, you may find that your system has two versions of the OpenCV library installed: 3.2.0 and 4.x.x.
  - OpenCV versions earlier than 4.0.0 do not support QR code-related libraries.
  - Thus, to utilize QR code-related classes or functions, you should compile your .cpp files with the appropriate options to ensure the correct library version is used.
  - For example, you can use the command pkg-config --cflags --libs opencv4 to compile, as shown below:
  - `$ g++ main.c -o run.out ` + "`pkg-config --cflags --libs opencv4`"

Due & Grading

- **Important Note**: This assignment is not graded individually, i.e., there is no late penalty for this time. The provided deadline is intended to help you plan and complete the project on time.
- **Submission Requirements:**
  - **Email:**
    - Decide on your team name.
    - Send an email to the TAs with the names of all team members.
  - **GitHub Repository:** Create a GitHub repository for your source code. Include a README file that lists your team name.
  - Video Demonstration:
  - Record a video demonstrating both programs—the line tracer and the QR code recognition—functioning correctly.
- Deadline:
  - Submit all required files via the Learning Management System (LMS) by May 21, Tuesday, at 11:59:59 PM.

Getting Help

If you need any help, send us an email.
When you send an email to TAs, you should send mail to both TAs by using CC.

- 이호연: lhyzone@dgist.ac.kr
- 이준영: lolcy3205@dgist.ac.kr

If you need to discuss the project details, you can also email to the instructor.

- 김예성: yeseongkim@dgist.ac.kr