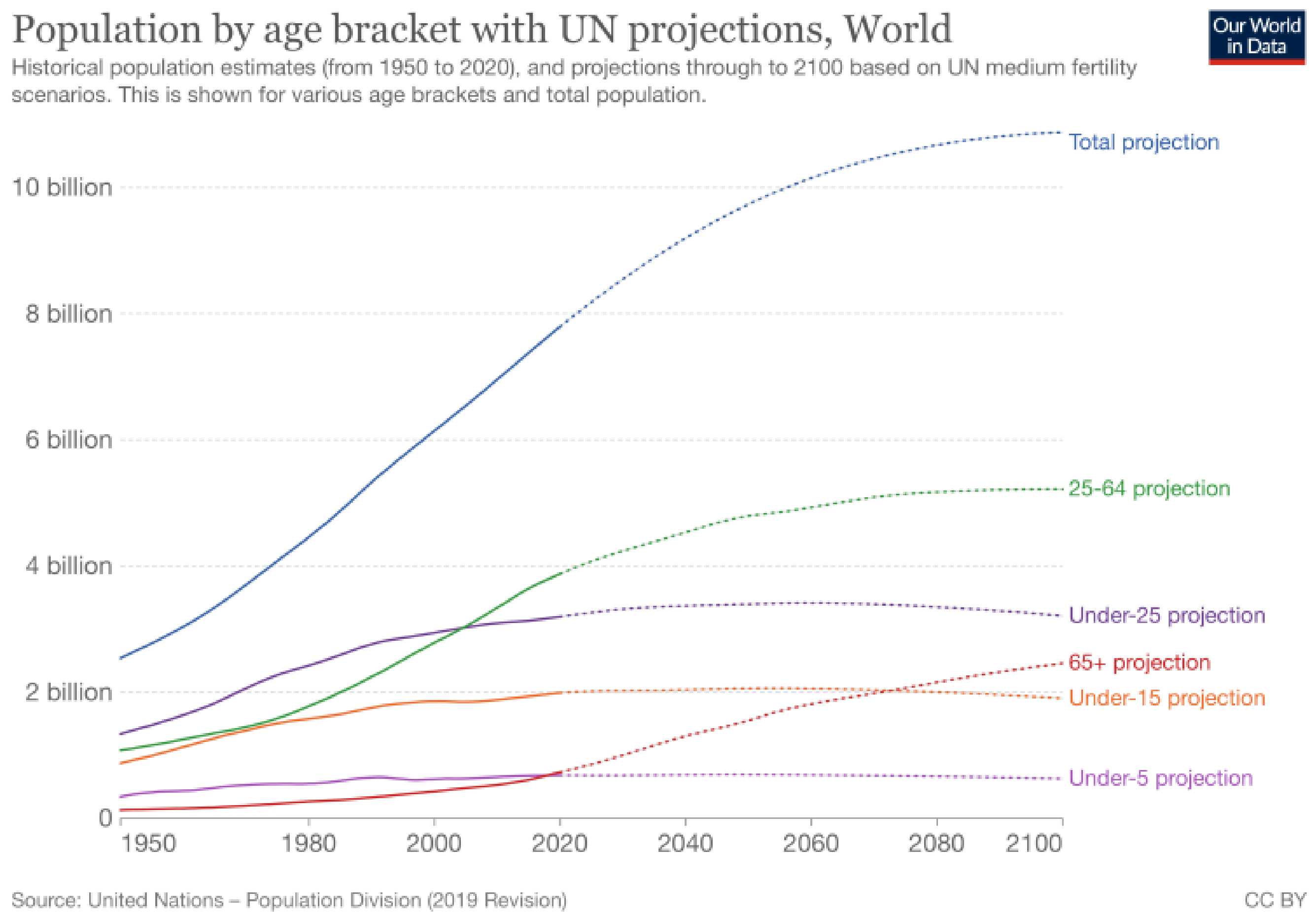
# The repository for the Sigyn robot.

## Motivation

Worldwide, the number of elderly people (65 years and older) surpassed the number of children under 5 years of age in 2019. It is likely, in my lifetime, that there will be more people needing assistance in their lives than there will people who can provide it. I don’t expect to be able to afford especially extraordinary care, nor do I expect not to need it as I age. My alternative is to create the needed technology so it will be there when I need it.



Source: https://ourworldindata.org/population-aged-65-outnumber-children

***Wimble Robotics*** is a one-person effort to create an assistive robot for my personal use.

## Organization

The various directories contain:

- Designs
  Holds various design documents. E.g. Fusion 360, LightBurn.
- Docker
  For creating docker images to run the code.
- Documentation
  Documentation artifacts, such as notes, ***wireviz*** documentation.

- Media
  Interesting pictures, movies.
- base
  ROS 2 package containing the main lunch files, configuration, maps and such.
- description
  ROS 2 package containing the URDF and Gazebo worlds.
- experiments
  Ongoing, vague experiements. Just a repository for trying out things that might become real code eventually, or gather data to guide development.
- gripper
  Teensy 4.1 code that manages the gripper elevator and extender.
- lidar
  Forked from elsewhere, contains fixes and customizations.
- msgs
  Contains messages for internal usage, not external API.
- micro_ros
  For creating a custom micro_ros with bigger message sizes and more of each resource kind.
- rviz
  rviz2 configuration files.
- scripts
  Utility scripts.
- sigyn_interfaces.
  Contains message, action and service definitions for ROS 2.
- twist_multiplexer
  Forked from elsewhere, contains fixes and customizations.
- udev
  Various rules to be placed in ***/etc/udev/rules*** to deal with the hardware. Mostly creates symbolic links to various devices, such as the ***teensy 4.1*** and ***LIDAR*** devices.

## Running the Robot

This is a standard ROS 2 package. Build it with colcon and source ***install/setup.bash***

In one window, you need to run the MicroRos agent via

```
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/teensy_sensor
```

Note that this relies on udev rules to create a symbolic link to the teensy 4.1 device in the ***/dev*** directory (see 00-teensy.rules).

In another window, run the nav2 stack, which also launches the other needed components.

```
ros2 launch base sigyn.launch.py use_sim_time:=false do_rviz:=true
```

To visualize the system, run ***rviz2***. There is a configuration file which shows things the way I like.

```
rviz2 -d ~/sigyn_ws/src/Sigyn/rviz/config/config.rviz
```

I usually begin by using the ***rviz2*** interface to set the 2D Pose Estimate, then I use the teleop module to move the robot around a bit to make sure that localization is working. Then I set a goal and watch it work.

If you don't have your own way of running teleoperation, here's what I use:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

## Running the Robot in Simulation

The robot can be run in simulation by running the following command:

```
ros2 launch base sigyn.launch.py use_sim_time:=true
```

## Recent Updates

### TeensyV2 System Enhancements

- **Modular Embedded Architecture**: Updated TeensyV2 system with improved real-time performance
- **Enhanced Battery Monitoring**: Battery messages now include location information for multi-battery setups
- **Streamlined ROS2 Bridge**: Consolidated monitoring into a single efficient bridge node
- **Updated Message Protocol**: Comprehensive documentation of communication protocol with examples

For detailed information about the TeensyV2 system and message protocol, see:

- `TeensyV2/README.md` - Complete embedded system documentation
- `TeensyV2/docs/MessageProtocol.md` - Communication protocol specification
- `sigyn_to_sensor_v2/README.md` - ROS2 bridge documentation