Read [Solaris Core 0 Getting Started](https://docs.google.com/document/d/19XEE7bN7Zx0fPYl6cBvHnxAOXBP4H4adkOzD0CfNBKc/edit?usp=sharing) first.

# Summary

Sar stands for “Software Architecture for Robots,” a generic platform for robot autonomy. This codebase includes:

* **Core 0**–the base hardware and software for wheeled navigation
* **Core 0 Arm**–base hardware and software for manipulation
* **Lytbot (Hotbot)**–autonomous disinfection robot for hospital rooms
* **AFS**–an autonomous floor scrubber for retail stores
* **Airbot**–autonomous disinfection robot for aeroplanes and buses
* **Amro**–medicine transportation robot

The following is a summary of the software roles on our team:

* **Ajay Vishnu** (CTO)--high-level decisions
* **Harry Samson** (Hardware Manager)—AWS, docker
* **Nathan George** (Software Manager)—architect, git, docs, states, integration
* **Sabari Manohar**–algorithms, nav2, Computer Vision
* **Debanik Roy**–docker, odometry, ROS2, nav2
* **Himanshu Sahni**–documentation, external integration, ROS2
* **Ajay Kumar**–embedded firmware
* **(Web Team)**—full-stack
* (Future)—refactoring, QA, DevOps

# Development Rules

* **Pull Request (PR) System**–how to merge code into master
  + Never directly merge with master, and create a branch for each Jira task.
  + Verify all Conventions (next bullet point) are met.
  + Upload media to your Jira task that proves the “Validation” section, and change the status to REVIEW.
* **Conventions**
  + Docker Deployment
    - Separate nodes into as many Docker services as possible.
    - Every Docker service must send a preset [xstate](https://xstate.js.org/docs/guides/start.html) transition (see state-machine.js) on “/xstate/update” via [ChangeState.transition.label](https://index.ros.org/p/lifecycle_msgs/). See example at ros\_2/testing\_ws/src/lytbot/ultrasonic\_avoidance/src/identify\_obstacle.cpp.
    - Most Docker services should only start if the [GetState](https://index.ros.org/p/lifecycle_msgs/) state on “/xstate/state” matches.
  + Code
    - A header comment for every block of code
    - No commented code or printing (rosconsole, cout, etc.)
    - Ask the manager to update the README.docx if necessary.
  + Files
    - Large files must be stored in [Core 0 Software Data](https://drive.google.com/drive/folders/195srOtgVtHJoMvJ0pG3faKZHn1l9UalH?usp=sharing).
    - Each robot package for Core 0 must have the following ROS packages:
      * robot\_control–embedded, sensors, etc.
      * robot\_model–URDF, worlds, meshes, rviz, etc.
      * robot\_navigation–navigation, SLAM, etc.
* **Documentation**
  + [Testing](https://drive.google.com/drive/folders/1FAnox9_4ybDwKXXTG-ZQczsx3bhmA-iF?usp=sharing)–High-level comparisons and testing results
  + [Tutorials](https://drive.google.com/drive/folders/1EDexcC5bKPPzfST5e35IoEqDXoRX8Pl4?usp=sharing)–long explanations for running something
  + Inline–see above PR Coding Conventions
  + Jira–change the status (except DONE) and comment daily

# Installation

## Main Software

1. Either install Ubuntu 20.04 or for Jetson boards…
   1. Make sure a WiFi chip is installed.
   2. Make sure the power source is officially supported; off-brands don’t always work.
   3. Install Jetson firmware.
      1. [Jetson Nano 4GB](https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write) ([verify](https://forums.developer.nvidia.com/t/checking-jetpack-version-on-jetson-nano/110901/2))
2. Install ROS
   1. Install [ROS2 Foxy](https://docs.ros.org/en/foxy/Installation/Ubuntu-Install-Debians.html)
   2. Install [colcon](https://colcon.readthedocs.io/en/released/user/installation.html)
3. Install Docker (the apt or snap options might fail)
   1. Docker
      1. pip3 install --upgrade pip
      2. pip install -U setuptools
      3. pip3 install -U setuptools
      4. sudo apt-get update -y
      5. sudo apt-get upgrade -y
      6. sudo apt-get install curl python3-pip libffi-dev python-openssl libssl-dev zlib1g-dev gcc g++ make -y
      7. curl -sSL https://get.docker.com/ | sh
      8. sudo pip3 install docker-compose
   2. Update Daemon
      1. sudo apt install nvidia-container-runtime
      2. sudo cp ~/sar/ros\_2/config/daemon.json ~/etc/docker
      3. sudo systemctl restart docker
   3. Connect with AWS for Docker Images
      1. Install [AWS Client V2](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html)
      2. aws configure # using keys from Harry
      3. Go to [Amazon ECR](https://us-east-2.console.aws.amazon.com/ecr/get-started?region=us-east-2) > [Repositories](https://us-east-2.console.aws.amazon.com/ecr/repositories?region=us-east-2) > [sar](https://us-east-2.console.aws.amazon.com/ecr/repositories/private/153479249734/sar?region=us-east-2) > “View push commands”
      4. Run the first command; it should look like “aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 153479249734.dkr.ecr.us-east-2.amazonaws.com”
4. Install Node.js and npm
   1. sudo apt install nodejs
   2. For x86\_64, “sudo apt install npm”
   3. For arm64, follow the [arm npm instructions](https://github.com/Jaronets/JETSON-npm-Install).
   4. cd ros\_2/docker/packages/state\_machine
   5. npm init
   6. npm install xstate
   7. npm install roslib
5. Build code for real robot
   1. git clone <https://git-codecommit.us-east-2.amazonaws.com/v1/repos/sar>
   2. cd /home/dev/sar/ros2\_ws/docker
   3. sudo docker-compose -f <docker-compose…> build
      1. choose the yaml file based on architecture
6. Build code for simulated robot
   1. cd ros\_2/testing\_ws
   2. Download the robot meshes from the following link directly: [urdf\_meshes](https://drive.google.com/drive/folders/1FYxd2MWQxU8EtELyaB7vnNBj2VY_P2mo).
   3. Move them to “jetbrain\_hotbot/meshes”.
   4. colcon build # testing\_ws

## Deprecated Software

If you need to use ROS1

* Install Ubuntu 18.04 and ROS Melodic
* git clone <https://git-codecommit.us-east-2.amazonaws.com/v1/repos/sar>
* cd /home/dev/sar/catkin\_ws
* rosdep install --from-paths src --ignore-src -r -y && catkin\_make

# Running

## Simulation

1. ros2 launch jetbrain\_hotbot jetbrain\_hotbot.launch.py

## Real Robot (Main Software)

1. cd ros\_2/docker/arm64
2. sudo docker-compose -f <yaml> up # see docker-compose.yaml for profiles
   1. replace <yaml> with the docker-compose of your architecture
3. docker-compose -f <yaml> down
   1. to destroy containers

## Real Robot (Deprecated Software)

1. Always
   1. Connect to the wifi of the robot.
   2. Export ROS\_MASTER\_URI and ROS\_HOSTNAME
   3. To check the ip use the command **hostname -I**
   4. ./sar/utilities/scripts/run.sh <name of the module --default Core0> <Name of the SBC(onboard computer) --default Odroid XU4>
2. Teleop
   1. rosrun teleop\_twist\_keyboard teleop\_twist\_keyboard.py
3. Mapping
   1. Run: ./sar/utilities/scripts/run.sh <name of the module --default Core0> <Name of the SBC --default Odroid XU4> -mode M
   2. Save: rosrun map\_server map\_saver -f map
4. Navigation
   1. ./sar/utilities/scripts/run.sh <name of the module --default Core0> <Name of the SBC(onboard computer) --default Odroid XU4>
   2. rosservice call /StartLocalization
5. Killing
   1. ./sar/utilities/scripts/killall.sh

# Files

The folder structure of the repo is as follows:

* config
  + motors - motor configurations
  + params - navigation parameters
* models
  + modules - specific robot setup for running the robot
  + robots\_urdf - urdf and xacro files of robots
* ros\_1- ROS1 code
  + Catkin\_ws
    - src/afs–[see further documentation](https://drive.google.com/drive/folders/1ugdxB-jbL3td8wJXsVBISAU4BLwARfDL)
    - src/airbot–[airbot\_documentation](https://docs.google.com/document/d/1Es3F6OA5XEp8w-eWGW5DduKbZEqPQz__/edit)
    - src/core0\_arm
    - src/dreamvu\_pal\_camera–[360 degree camera driver](https://drive.google.com/drive/folders/1ygvQFIDsK0o0in6Z0K-CKX_8_h6e7oIC)
  + control - ROS1 software for the robots
  + launch - ros launch files for run.sh
* ros\_2 - ROS2 code
  + src—old simulation (TODO remove)
  + docker—new simulation and real world deployment
* tools - reset the rosserial in case of arduino failure
* web
  + db - web database
  + logs - web logs
  + maps - real world maps for robots to navigate
  + server - app and web based server code