Robot Software Architecture

A robot is a system of interconnected hardware and software subsystems. All robots have to following sub-systems:

- **joystick controller**–a way to interface a joystick controller to a set of commands.
- **drive train**–a mechanism for moving the robot about a playing field.
- **navigation**–a set of sensors and derived measurements that help to locate a robot on the playing field. This is usually a gyroscope that detects movement in three cardinal directions. It may provide magnetic measurements to determine magnetic heading angles. It may integrate the movements into a x,y,z coordinate on the playing field. This inertial data may require periodic resetting during a match to keep it accurate.

From there additional subsystems are used to provide functions and control suited for the overall capability of the robot. This may be things like:

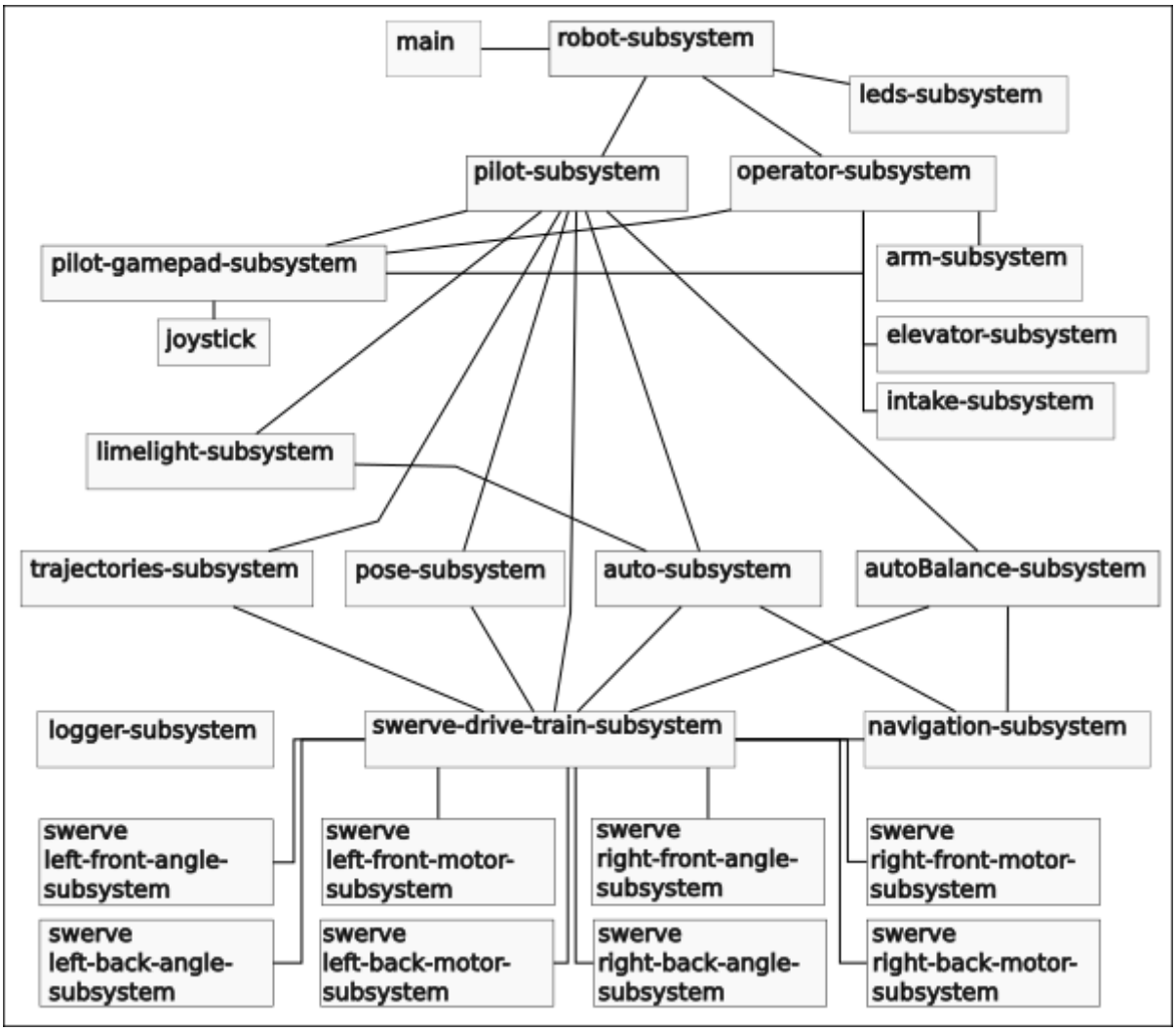
- **elevator**–a height extension mechanism of a robot.
- **arm**–a lateral extension mechanism of a robot.
- **pointing** or **pose** is the angle of the robot with respect to the field.–mechanism for propelling game pieces.
- **intake**–mechanism for loading and off-loading game pieces.
- **climber**–mechanism for climbing.
- **limelight**–a vision sensor for detecting game piece, pickup stations and scoring stations.
- **leds**–control of LED displays on robot.
- **swerve-direction**–control the angular direction of a swerve motor with respect to the frame of the swerve module
- **swerve-motion**-control the swerve motor
- etc.

There are also subsystems that perform an operation by controlling multiple subsystems, such as:

- **autonomous**–control the robot during the autonomous period.
- **balance**–control the robot to balance on the charge station (2023 Charge Up).
- **pickup**–control the robot to align and pickup a game piece.
- **score**–control the robot to align and score a game piece.

Coordination

Subsystems may communicate with each other. For most actions, commands originate from the joystick controller or the pilot and are passed on to the responsible subsystem.



Robot Module

The entry point for Java programs is a function called 'main.' In this architecture, the 'main' module just passes control to the robot module, which stands at the top of the robot software architecture. It creates instances of the major subsystems.

It provides methods for major operation modes such as the 'autonomous' mode where the robot is entirely controlled by the computer and the 'teleop' mode where human pilots drive the robot with computer assist. It also includes various modes for testing the robot and for simulating robot functions.

Subsystem Software Architecture

A subsystem controls a set of devices within a robot as a single process running one command at a time. Each subsystem is managed with a common set of methods:

- **setupDefaultCommand** is a method to establish a default command within the subsystem.
- **periodic** is a method that is called periodically to poll the status of various switches and sensors pertaining to the subsystem.

A **telemetry** module within a subsystem is used to convey information about the subsystem to a tab in the **shuffleboard** on the operator console.

Command Architecture

Each subsystem has a set of commands. A subsystem can execute only one command at a time. A command may run until it meets a particular criteria (like time, position, angle, count). Commands may also be interrupted or stopped by other conditions. The general architecture of each command is composed of the following methods:

- **initialize**–a method to initialize and setup the environment for the command to run
- **execute**–a method that is called every 20 milliseconds to perform the actions requested by the command
- **isFinished**–a method that is called after every execute to determine if the command is complete
- **interrupted**–a method to handle a request to interrupt the progress of a command in progress. Many times an interrupt will invoke the 'end' method of the command
- **end**–a method to stop the command either during or at the end of its execute cycle. This method brings the subsystem to a stable and known state

Subsystems

The following is a list of the current subsystems used by the robot.

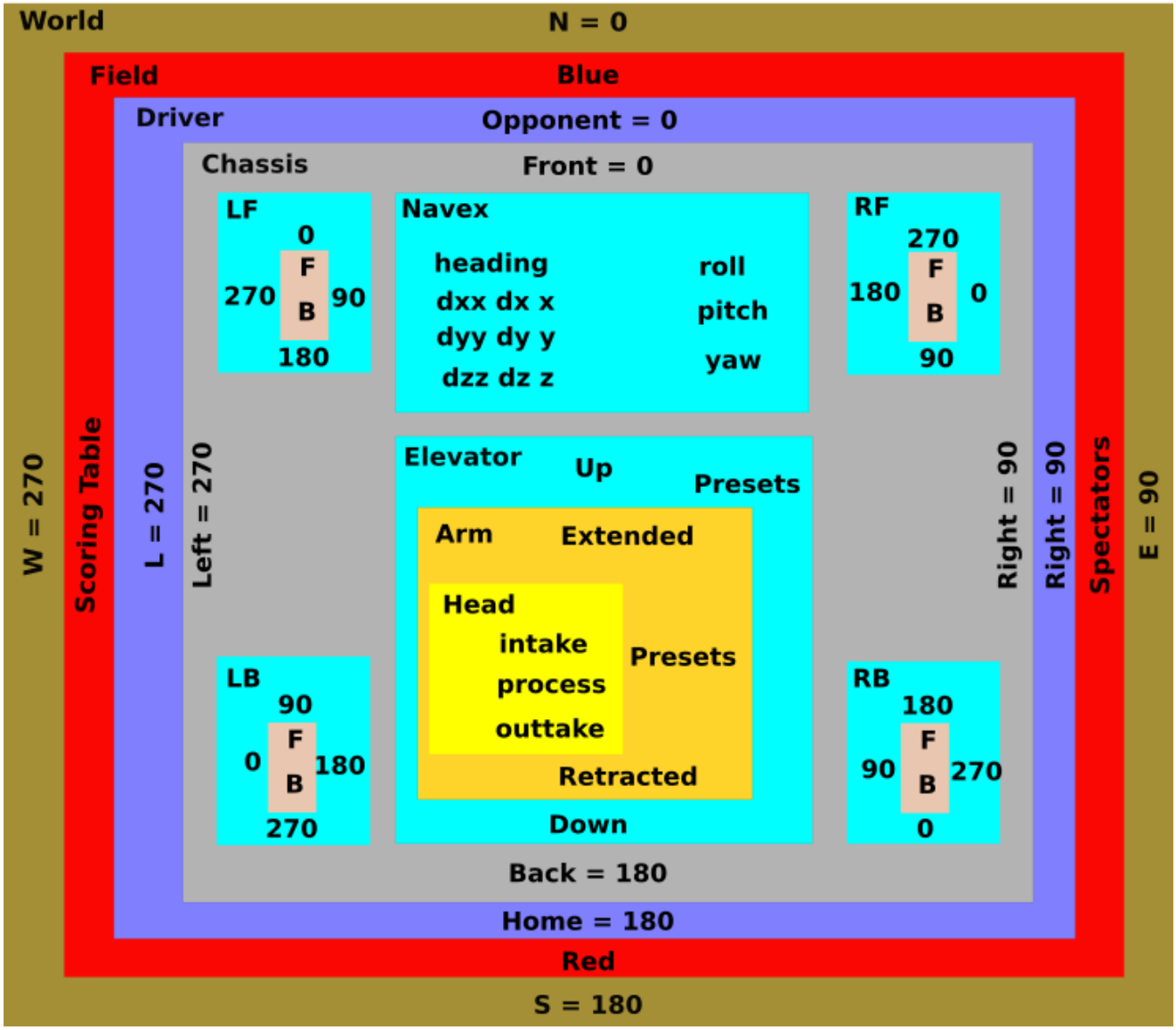
- **arm**–controls and monitors the position of the arm.
- **auto**–controls the robot during the autonomous period.
- **autoBalance**-controls the robot to balance the robot on the charge platform without human intervention. (This does not appear to be hooked up in the current 4513 code.)
- **elevator**–controls and monitors the position of the elevator.
- **intake**–controls and monitors the intake mechanism.
- **leds**–controls LEDs on the robot. (This does not appear to do anything in the current 4513 code.)
- **limelight**–controls and monitors a vision system capable of reporting certain location and game piece information. (This does not appear to be hooked up in the current 4513 code.)
- **logger**–log various events for debugging and performance evaluation. Most subsystem generate events for logging. (This appears to not be hooked up in the current 4513 code.)
- **operator**–provides control and feedback interfaces to a human co-pilot.
- **pilot**–provides control and feedback interfaces to a human driver.
- **pose**–looks like this just returns an xy position on the field. It also has an angle, but does not say what it references. It almost seems like this what I was calling pointing–angle of robot with respect to the field. (This is only used by the swerve subsystem in the current 4513 code.)
- **swerve**–controls the swerve drives to provide unified control interface for the drives.
- **trajectories**–methods for driving the robot over paths determined on the fly to minimize unnecessary movement to maximize speed and agility.

Frames of Reference

Knowing where a robot is and where it is pointing is important for automated controls within the robot. Nearly every device on the robot has at least one angle or position that it is concerned about. In simple robots, everything can reference the robot chassis. Its front is at 0 degrees and angles increase clockwise looking down from above. Driving the robot is much easier when looking at the field from the driver perspective. Down field is 0 degrees, right is 90 degrees and left is 270 degrees. The driver may be on a red alliance or a blue alliance. The operation of the robot really does not change, but the placement of field pieces and game pieces does. This has an impact to automated sequences during both the autonomous period and the teleop period. Beyond the field setup is the world. This can be found with a magnetic compass found in some navigation sensors, again with north being 0, degrees, east being 90 degrees, south 180 degrees and west 270 degrees. This could be useful in some situations to determine the bearing of the robot, but the relation ship of the magnetic compass and the field must be known or determined.

The components within a robot also have frames of reference in their relationship to the robot chassis. In a swerve drive robot there are four drive assemblies. These each have a different angular relationship with the chassis. it is also important to know their position with the robot for determining how to turn. This can be expressed as a x, y measurement from the center of the robot or an angle from the front of the robot and the radian distance between the center and the wheel of the motor assembly. It is also important to know at what angle forward is within the assembly. The drive motor can turn in either direction, so the assembly only needs to be turn +/- 90 degrees to reach any desired angle. Having a wheel turning backward in relation to other wheels is counter productive.

Robot Frames of Reference



The navigation system provides two more frames of reference. It provides a magnetometer that shows the roll, pitch and yaw of the robot with respect to magnetic north. This could help align the robot with respect to the world frame of reference as long as the correlation between the magnetic direction and the field direction are known. The inertial measurements provide a roll, pitch and yaw angles of the robot with respect to its start up position (or subsequent resets). The system also provides a displacement estimate that is good to about 5-6 inches during a competition (unless accurately reset). For the inertial measurements to be useful for field displacement estimates and angular measurements, they must be aligned to the field frame of reference. Since robots may be started or reset from various points on the field, the point where the reset is occurring must be supplied manually or determined by a vision system against field land marks or inferred by displacement estimate and commands being performed.

The difference between the magnetic frame of reference and the inertial frame of reference may be a way to measure the drift of the inertial frame of reference.

Accessory attachments to the robot chassis such as a elevator and or arm mechanisms have a (hopefully) fixed relationship to the chassis. This allows the chassis to be oriented in a way that the elevator and arm can effectively interact with the game elements.

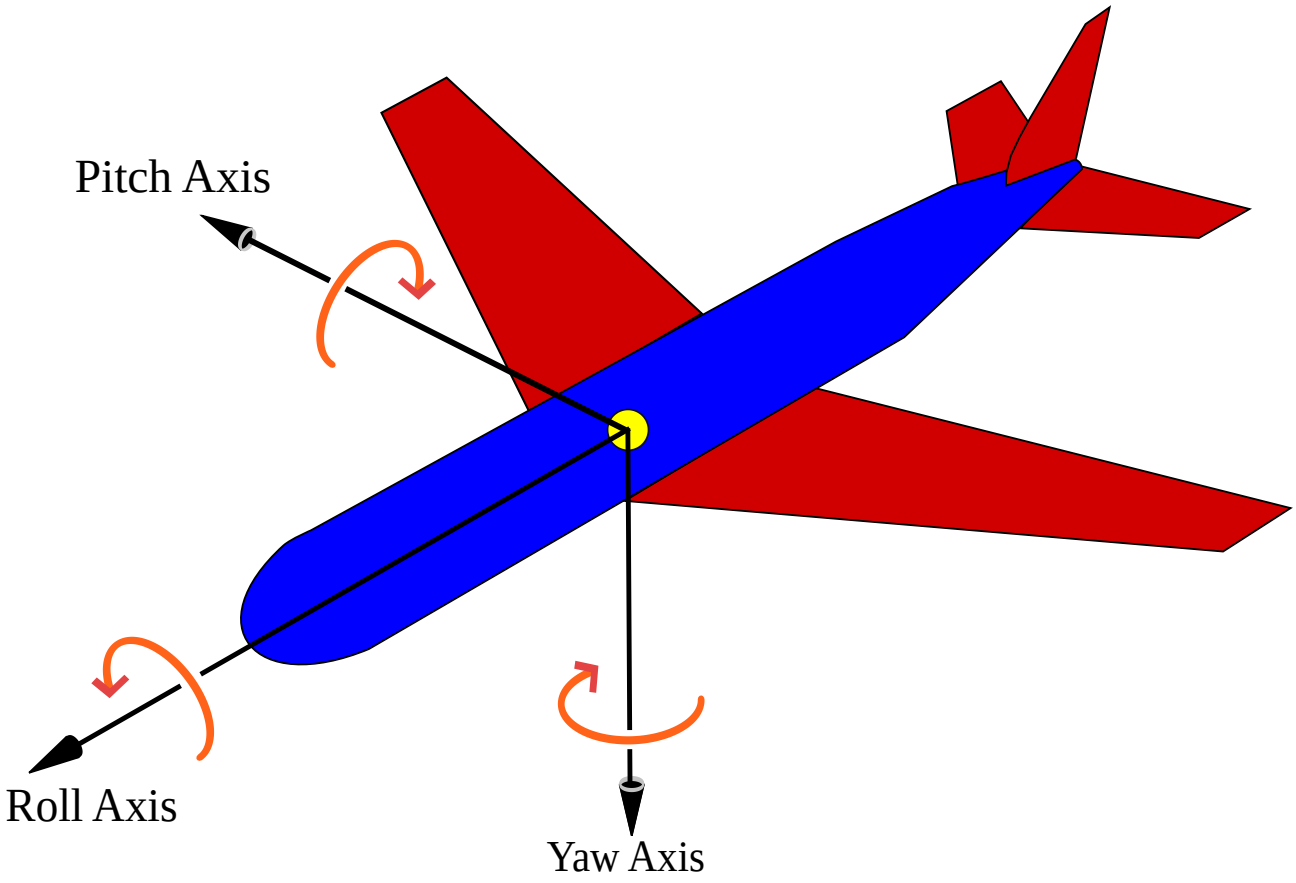
Vocabulary

heading is the angle that the robot is moving toward with respect to the field.

pitch is the (front-to-vertical) angle of rotation of the robot about the x-axis with respect to initial inertial vertical angle.

pointing or **pose** is the angle of the robot with respect to the field.

roll is the (side-to-vertical) angle of rotation of the robot about the y-axis with respect to the initial inertial vertical angle.

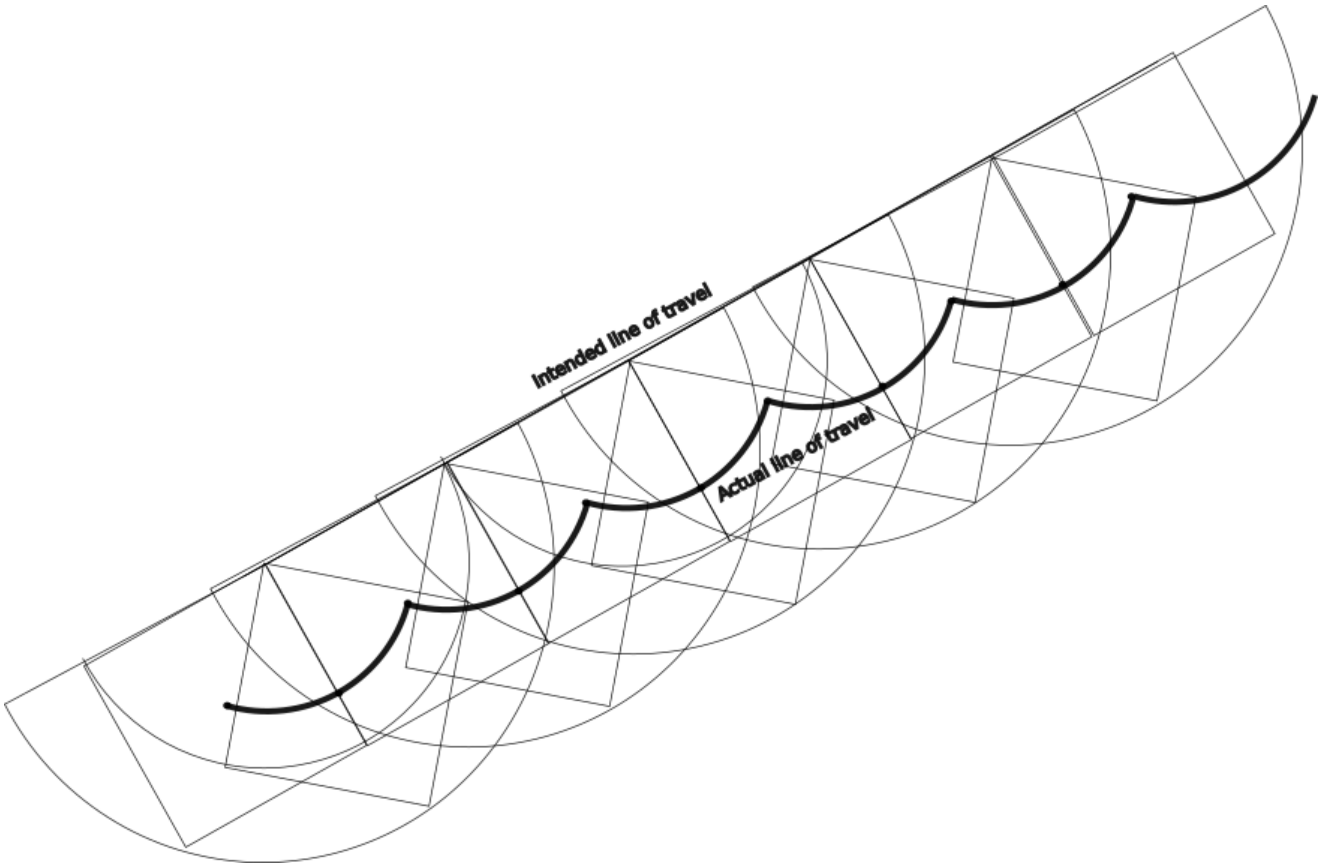


yaw is the angle of the robot (really inertial measurement device) about the z-axis and the initial inertial angle.

Ideas

The navigation subsystem will detect collisions. You can take the impact vector and subtract the last known robot momentum (E = mvv) vector to find the direction of the colliding robot. For automatic bump and run...check to see if the robot is clear of game structures, if it is, roll away from the colliding robot.

The bump and run roll holds a corner of the robot against the other robot. The center of the robot will trace a scallop curve with the radius of the center of the robot rotating around a corner until another corner makes contact with the opposing robot. If the movement is to the left of the other robot, the spin is clockwise, otherwise it is counter clockwise. The rate of the spin is to keep the corners of the robot stationary.



Should turn on the logger code. This should be used to output timestamped pose estimates which can be used to create a heat map to point out where too much time is being spent.