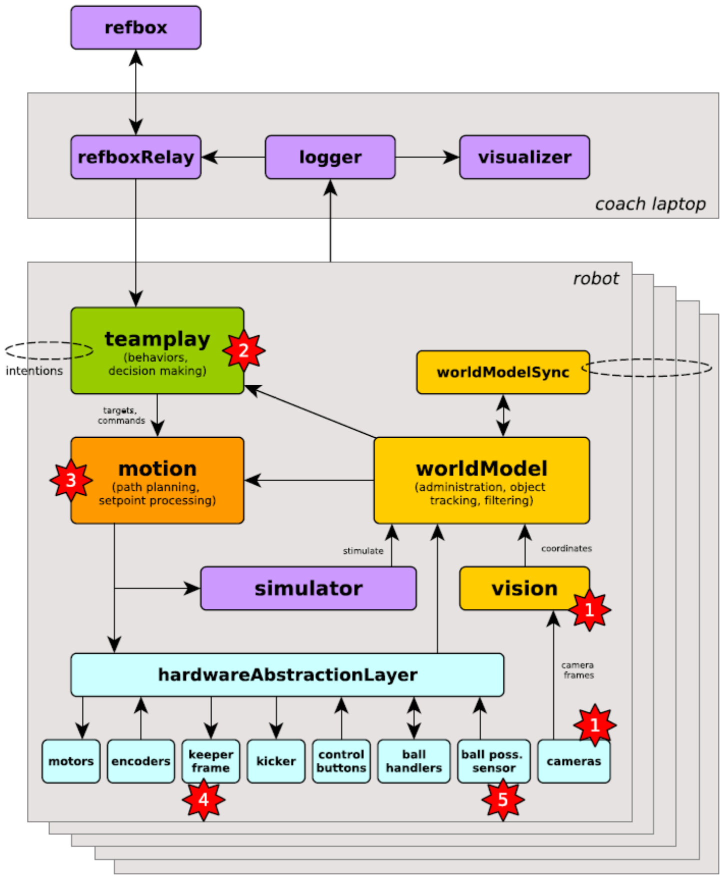


This year, the following major changes are introduced.

1. the omniVision camera module is replaced by a multiVision module
2. in teamplay the concept of heightmaps is introduced, to achieve dynamic positioning.
3. in motion, several activities are ongoing to enable faster driving:
    - the PID speed setpoint controller is replaced with a setpoint generator
    - a new processing layer 'motionPlanning' is introduced to improve software architecture and increase shoot accuracy
    - as motion board replacement the BeagleBoneBlack is considered
4. the keeper frame is now extendible
5. as a consequence of the new multiVision system, a new sensor is needed to verify ball possession

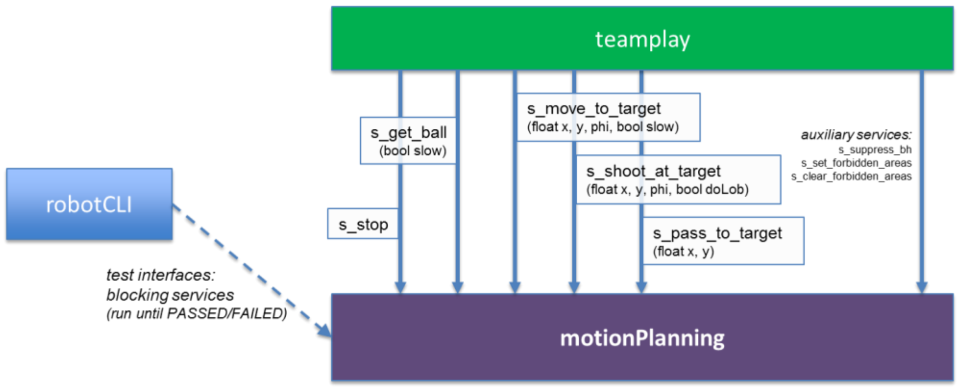
MotionPlanning

In the software architecture, a new component motionPlanning was introduced.
MotionPlanning is responsible for translating high-level teamplay commands into motion setpoints. This includes sequencing, timing and tolerances. Each software heartbeat, it reports back to teamplay what the state is of the requested command: passed, running or failed. This then gives teamplay the choice of continuing with current command, or switching to a different command.

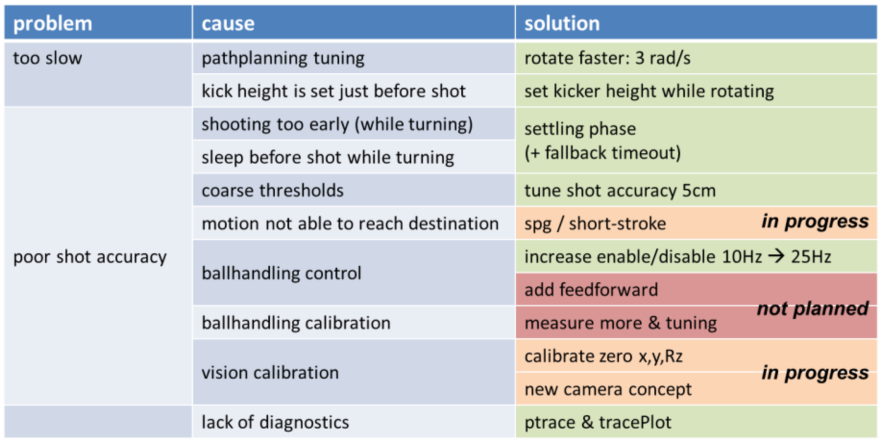
Introduction of this component solves a number of issues:

- wildgrowth: teamplay contained too much motion logic and motion thresholds in its actions layer and even in the behavior trees
  - ➔ motion logic, thresholds and tricks are now applied in motionPlanning, not in teamplay
- interface spaghetti: teamplay had direct connections with several lower-level motion components, even with hardwareAbstractionLayer
  - ➔ teamplay now only communicates in a high-level abstraction with motionPlanning, which then will communicate with the lower components (shootPlanning, pathPlanning, etc.)
- poor extendibility: it was relatively hard to apply seemingly simple functional extensions, such as kicker height adjustment during aiming for shot, or sprints for the ball
  - ➔ these extensions can now simply be introduced in the respective command implementation in motionPlanning
- code duplication in python scripts due to limited test interfaces
  - ➔ motionPlanning offers straightforward test interface wrappers around the commands it provides to teamplay; these test interfaces block until done and are now called from python test/scripting suite during e.g. technical challenge, or calibration / tuning.

The interface between teamplay and motionPlanning now is:



As first major functional use case for this component, we improved shoot accuracy while also increasing rotation speed with ball.
The following table shows a collection of problems, causes and solutions we identified and (partially) addressed.



problem	cause	solution
too slow	pathplanning tuning	rotate faster: 3 rad/s
	kick height is set just before shot	set kicker height while rotating
poor shot accuracy	shooting too early (while turning)	settling phase (+ fallback timeout)
	sleep before shot while turning	
	coarse thresholds	tune shot accuracy 5cm
	motion not able to reach destination	spg / short-stroke ***in progress***
	ballhandling control	increase enable/disable 10Hz → 25Hz
		add feedforward ***not planned***
	ballhandling calibration	measure more & tuning
	vision calibration	calibrate zero x,y,Rz ***in progress***
		new camera concept
	lack of diagnostics	ptrace & tracePlot

To summarize, our shooting speed and accuracy was improved by

- setting kicker height during rotation,
- waiting for the robot to settle on target, to decouple motion (tuning) issues like overshoot from shoot accuracy,
- using a distance metric instead of a static angle threshold for shoot accuracy,
- increasing rotation speed limit with ball from 1.8rad/s to 3.0rad/s.