# Robotics platform

## Architecture 3, Test 2:

* This design uses the databases (masterControl and status) as the primary data exchange medium on the server side. I think if possible, the use of zMQ from the human interface to the controller may reduce/eliminate lag. Since I have made the decision to hard code play modes (masterControl commands), the need for a database to archive all commands and used to ‘’select’’ the active command would not be necessary.
* The start/stop commands are working.
* Is there a need for user controlled commands (other than game mode). Such as : robot 1 slow down. Or is it okay to let the server controller micro-manage specific actions (caused by hall effect switches and rfid tags – through the “special1” and “special2” fields).
* Only basic functionality has been implemented to demonstrate the viability of this architecture. So far tests have been positive. Delays are minimal (even with heavy db use on server side).

Data format from server to robot:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | drive | aux | Special1 | Special2 | report | request |

* ID: This needs to be larger than “0” (zero) for the controller scrip to read the masterControl database. By default, use “1”.
* Drive: If drive is “1” and request is “4”, the robot is enabled to drive normally.
  + If drive is “0” and request is “4”, the robot is stopped.
* Aux: This selects if the auxiliary motors are enables (“1”) or disabled (“0”). Not implemented.
* Special1, special2: special commands where RFID tag/hall effect switch responses can be sent back to the robot. Depending on the play mode selected by the control user, these can mean different things. The controller has to implement special1&special2 capabilities in the future.
* Report: Refers to the robot number (1..16). Use “60” for all robots.
  + This selects either a specific robot to which this command applies or selects to send the command to all robots.
* Request: Details what type of request the command line has.
  + Request = “1” : return command values to null (default driving mode)
  + Request = “2”: Send robot status to the server.
  + Request = “3”: special 1 is used to be a number between “0” and “1”. This will be multiplied with the drive speed limit to slow a robot by a proportional factor of that value.
  + Request = “4”: will either enable the robot to drive normally (drive = “1”) or stop the robot (drive = “0”)
  + FUTURE DEVELOPMENT EXPECTED.

CAUTION: Programmers should take note that: the robot script MUST have the sleep delays in the main loop. This is to reduce CPU load and decrease heat on the RPI. Removing the delays may cause overheating and/or catastrophic failure.