

SmartDashboard namespace

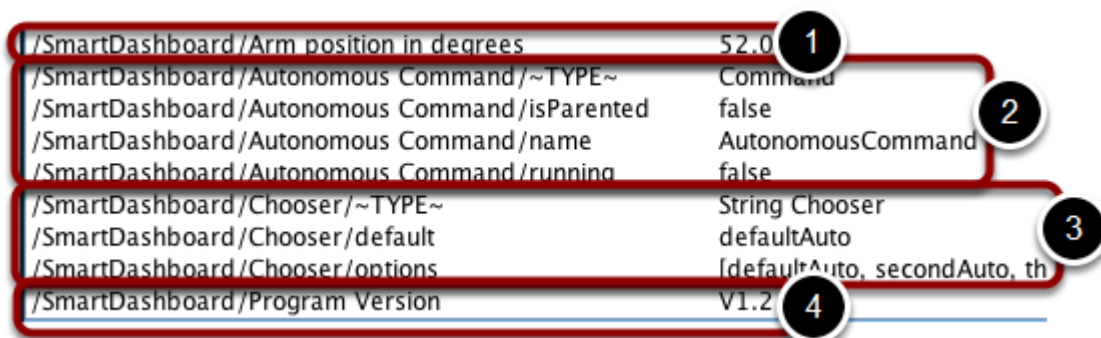
SmartDashboard namespace

SmartDashboard uses NetworkTables to send data between the robot and the Dashboard (Driver Station) computer. NetworkTables sends data as name, value pairs, like a distributed hashtable between the robot and the computer. When a value is changed in one place, its value is automatically updated in the other place. This mechanism and a standard set of name (keys) is how data is displayed on the SmartDashboard.

There is a hierarchical structure in the name space creating a set of tables and subtables. SmartDashboard data is in the SmartDashboard subtable and LiveWindow data is in the LiveWindow subtable as shown below.

For informational purposes the names and values can be displayed using the TableViewer application that is installed in the same location as the SmartDashboard. It will display all the NetworkTable keys and values as they are updated.

SmartDashboard data values



The screenshot shows a TableViewer window with a list of SmartDashboard data entries. Red boxes and numbered callouts highlight specific parts of the data:

/SmartDashboard/Arm position in degrees	52.0	1
/SmartDashboard/Autonomous Command/~TYPE~	Command	2
/SmartDashboard/Autonomous Command/isParented	false	
/SmartDashboard/Autonomous Command/name	AutonomousCommand	
/SmartDashboard/Autonomous Command/running	false	
/SmartDashboard/Chooser/~TYPE~	String Chooser	3
/SmartDashboard/Chooser/default	defaultAuto	
/SmartDashboard/Chooser/options	[defaultAuto, secondAuto, th	
/SmartDashboard/Program Version	V1.2	4

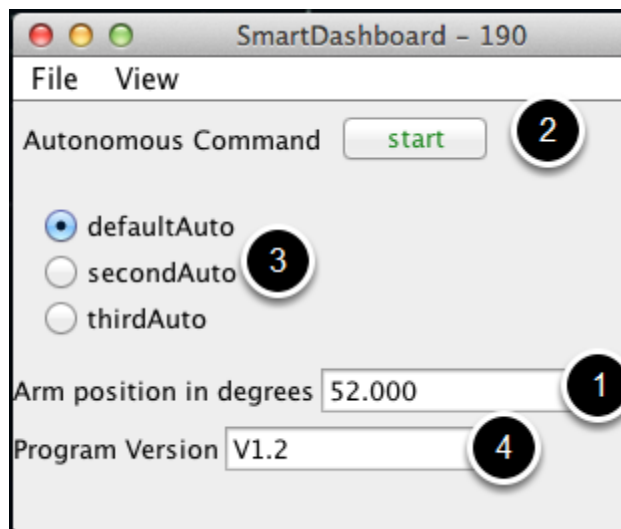
SmartDashboard values are created with key names that begin with "SmartDashboard/". The above values viewed with TableViewer correspond to data put to the SmartDashboard with the following statements:

```
chooser = new SendableChooser();
chooser.addDefault("defaultAuto", new AutonomousCommand());
chooser.addObject("secondAuto", new AutonomousCommand());
chooser.addObject("thirdAuto", new AutonomousCommand());
SmartDashboard.putData("Chooser", chooser);
SmartDashboard.putNumber("Arm position in degrees", 52.0);
SmartDashboard.putString("Program Version", "V1.2");
```

SmartDashboard namespace

The "Arm position" is created with the `putNumber()` call. The `AutonomousCommand` is written with a `putData("Autonomous Command", command)` that is not shown in the above code fragment. The chooser is created as a `SendableChooser` object and the string value, "Program Version" is created with the `putString()` call.

View of the SmartDashboard



The code from the previous step generates the table values as shown and the SmartDashboard display as shown here. The numbers correspond to the `NetworkTable` variables shown in the previous step.

LiveWindow data values

The screenshot displays the SmartDashboard software interface, which is used for controlling a robot. The interface features a title bar at the top that reads "SmartDashboard - 190". Below the title bar is a menu bar with "File" and "View" options. The main area of the interface is divided into four panels, each representing a different motor or sensor:

- Drive train:** This panel includes a slider control, a numerical display showing "0.000", and a "Zero" button.
- Ultrasonic:** This panel includes a numerical display showing "0.573".
- Left:** This panel includes a slider control, a numerical display showing "0.000", and a "Zero" button.
- Right:** This panel includes a slider control, a numerical display showing "0.000", and a "Zero" button.

Each panel also has a "PIDSubsystem Controller" section with fields for "P", "I", "D", and "F" values, all of which are currently set to "0.000". Additionally, there is a "Setpoint" field and an "Enabled" checkbox for each motor. The "Enabled" checkbox for the "Drive train" motor is currently unchecked.

```
drivetrainLeft = new Talon(1, 2);
LiveWindow.addActuator("Drive train", "Left", (Talon) drivetrainLeft);
drivetrainRight = new Talon(1, 1);
    LiveWindow.addActuator("Drive train", "Right", (Talon) drivetrainRight);
drivetrainRobotDrive = new RobotDrive(drivetrainLeft, drivetrainRight);
drivetrainRobotDrive.setSafetyEnabled(false);
drivetrainRobotDrive.setExpiration(0.1);
drivetrainRobotDrive.setSensitivity(0.5);
drivetrainRobotDrive.setMaxOutput(1.0);
drivetrainUltrasonic = new AnalogChannel(1, 3);
```

SmartDashboard namespace

```
LiveWindow.addSensor("Drive train", "Ultrasonic", drivetrainUltrasonic);
elevatorMotor = new Victor(1, 6);
LiveWindow.addActuator("Elevator", "Motor", (Victor) elevatorMotor);
elevatorPot = new AnalogChannel(1, 4);
LiveWindow.addSensor("Elevator", "Pot", elevatorPot);
wristPot = new AnalogChannel(1, 2);
LiveWindow.addSensor("Wrist", "Pot", wristPot);
wristMotor = new Victor(1, 3);
LiveWindow.addActuator("Wrist", "Motor", (Victor) wristMotor);
clawMotor = new Victor(1, 5);
LiveWindow.addActuator("Claw", "Motor", (Victor) clawMotor);
```

Values that correspond to actuators are not only displayed, but can be set using sliders created in the SmartDashboard in Test mode.