Complete Guide to the ZomB Dashboard System

Team 451 "The Cat Attack"

Revision 1.1

Table Of Contents

Overview

The ZomB dashboard is aimed at providing any FRC team an easy to use customizable dashboard that they can use for debugging and competition display. This document is divided into 3 parts: installation and basic use, robot side configuration and development, and ZomB configuration and development. The first section covers downloading and "installing" the latest ZomB, and then sending data to the default dashboard. You only need to know minimal C++ or Java. This section also briefly covers ZomB Eye. The second section covers the ZomB protocol, and the features in the new sender. A moderate amount of C++ or Java knowledge is required. The third section covers creating a custom dashboard, and requires little to moderate amount of knowledge in C# or Visual Basic, depending on the dashboard.

Conventions used in this book

TODO

Notes

This document is published under the GNU Free Documentation License. This document is a work in progress, and may cover some features that are planned, but have not been committed to source, or build. It may also incorrectly cover some things that are planned to be changed soon. This document is eventually going to target the next version of ZomB (0.7), so hang tight if a feature is not available yet.

Installation and Basic Use

This chapter covers downloading, installing, and setup for the Default Dashboard. You only need to know basic C++ or Java to complete this section

Downloading

Go to the ZomB Dashboard System's FIRST Forge page at http://firstforge.wpi.edu/sf/projects/zombdashboard and click on the File Release section at the top. Click the latest release in the Nightly package (should be 0.6.1.something), and click on it. You should be presented with a list of files. Click on the Installer to download it. If you are not running Windows 7, you may need to install the .Net framework 3.5: http://msdn.microsoft.com/en-us/netframework/cc378097

Installing the Default Dashboard

Make sure the Driver Station is not running, and open My Computer and go to C:\Program Files\FRC Dashboard\ where you should see 3 files. Move these files to a different location so you have a backup. Run the installer (It may take awhile). Open C:\Program Files\ZomB (or your install path) and find Default.exe and copy it to C:\Program Files\FRC Dashboard\. If you run the Driver Station, you should see the default ZomB dashboard at the top instead of the LabVIEW dashboard. If you don't see anything, go to the setup tab and make sure Local Dashboard is checked.

Installing the Bindings

Open NetBeans or WindRiver Workbench (To be refereed to as the IDE from here on), and create a new project, or open an existing project. In that project, copy and paste the downloaded C++ or Java bindings file(s). In C++, add the underlined code at the top:

```
#include "WPILib.h"
#include "ZDashboard.h"
```

In Java, add the underlined code at the top:

```
package edu.wpi.first.wpilibj.templates;
import edu.wpi.first.wpilibj.SimpleRobot;
import org.thecatattack.System451.Communication.*;
```

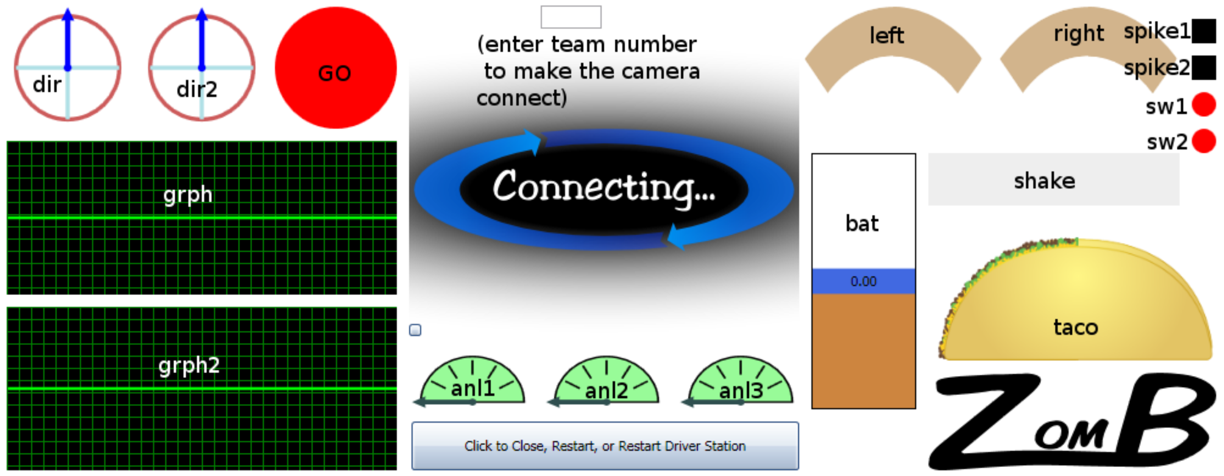
Sending Data

Now that the ZomB Dashboard bindings are in the IDE, we can write code to send data to the ZomB Dashboard. We first need to create an instance of the ZomBDashboard class:

```
class RobotDemo : public SimpleRobot
{
        RobotDrive myRobot; // robot drive system
        Joystick stick; // only joystick
        ZomBDashboard zomB;

    public:
        RobotDemo(void) :
            myRobot(1, 2), // you will most likely have
            stick(1), // more variables than this
             zomB(ZomBDashboard::GetInstance(AllTCP))
        {
            //More code goes here
```

```
public class RobotTemplate extends SimpleRobot {

   Joystick stick = new Joystick(1);
   ZDashboard zomB = new ZDashboard();

   //More code goes here
```

This code creates an instance of the ZomBDashboard called zomB, which we can use to send data. In C++, Instead of specifying *AllTCP*, you can also or together the following values: *TCP, RemoteData, DBPacket*. To send data, you need to know 2 things: the control name, and the value. If you are using the old Default Dashboard, click the "Show Names" button above the camera view to see all the controls you can control, and their names. Click anywhere to close the names. If you are using the new Default Dashboard (WPF), use the following chart:


dir
dir2
GO
(enter team number to make the camera connect)
Connecting...
left
right
spike1
spike2
sw1
sw2
grph
grph2
bat
0.00
shake
taco
anl1
anl2
anl3
Click to Close, Restart, or Restart Driver Station
ZomB

Lets say you wanted to send the joystick's y axis to the taco. This code would do it:

```
void OperatorControl()
{
   GetWatchdog().SetEnabled(true);
   while (IsOperatorControl())
   {
      GetWatchdog().Feed();

      //Put other code here

      if (zomB.CanSend())
      {
         zomB.Add("taco", stick.GetY());
         zomB.Send();
      }
      Wait(0.01);
   }
}
```

```
public void operatorControl() {
    getWatchdog().setEnabled(true);
    while (isOperatorControl()) {
        getWatchdog().feed();

        //Put other code here

        zomB.Add("taco", stick.getY());
        zomB.Send();
        Timer.delay(0.01);
    }
}
```

The CanSend function prevents you from overloading the network. The Add function takes the control name, and any value. The Send function sends it off.

Please note that the CanSend function is only available in the C++ Bindings packages.

Build and download your code to the cRIO, and run the DriverStation. Once you enable the robot, you should see the taco (or whatever controls you are using) being updated with the specified value. If you are having issues, make sure the values you are sending are in the correct range (usually -1 to 1)

Using the Camera

To use the camera, you just need to create an instance of the AxisCamera class in your robot code, and set the team number on the Default Dashboard (just above the camera view, default 451). Once all this is done, and the robot is running, click the Reset button above the camera view (old) or click the small button in the bottom right corner (new). If after 10 seconds, the camera does not show up, click it again. If at any time the camera dies, clicking Reset should clear up the error. If not, you can restart ZomB (see below, ZomB Button) If you don't want to have to keep clicking Reset, check the check box in the lower left corner of the camera view (old only).

Creating an instance of AxisCamera:

```
class RobotDemo : public SimpleRobot
{
        RobotDrive myRobot;
        Joystick stick;
        ZomBDashboard zb;
        AxisCamera& cam;

    public:
        RobotDemo(void) :
            myRobot(1, 2), stick(1),
            zb(ZomBDashboard::GetInstance(AllTCP)),
            cam(AxisCamera::GetInstance())
        {
            //More code goes here
```

```
public class RobotTemplate extends SimpleRobot {

   Joystick stick = new Joystick(1);
   ZomBDashboard zomB = ZomBDashboard.getInstance(DBPacket);
   AxisCamera x = AxisCamera.getInstance();

   //More code goes here
```

Do note that you don't need a ZomBDashboard instance to get the camera view to work.

Running ZomB on a Remote Computer

It is easy to run the ZomB Dashboard on a computer not running the Driver Station software, just run the exe on any computer you want. However, the Driver Station does not know where we are, so we need to tell it where we are. Find the IP address of the computer you are on by going to command prompt (run cmd.exe) and entering "*ipconfig*". Once you know the IP address, go to the Driver Station, click the Setup tab, and click on remote dashboard. Enter the IP address of the computer the dashboard is running on, and hit enter. The ZomB Dashboard should now be able to receive data. Note that at this point (0.7a5) you cannot use TCP connections on a remote computer, but that will change in the near future.

Saving Video

Want to record a video of what your robot sees through its camera? Make sure the camera view is working, then right click on it to start and stop it. Videos are saved in C:\Program Files\ZomB\Data as AVI's by the old dashboard (***WARNING! WARNING! WARNING!*** The videos are UNCOMPRESSED and 2 minutes of recording makes a 400MB file! Use the new Dashboard!) and compressed WebM files in the new dashboard.

ZomB Button

The ZomB Button is the button at the center bottom of the Default Dashboard with the text "Exit ZomB, Restart ZomB, or Restart DS" and when you click it, a menu pops up with those options. This is useful if the dashboard is acting up, or to clear FMS Locked (Restart DS) (NOTE: TCP on C++ does not like a restarting as of 0.7a5)

Other Features

ZomB's old Default Dashboard (WinForms) has a number of features. The check box in the corner of the camera view auto-resets the camera. Clicking the "See Names" button displays the names of all the controls, their range, and a brief how to. Right clicking on the camera view lets you change the source between robot (default) and webcam. You can even record webcam if you want to!

Robot Side Code and the ZomB Protocol

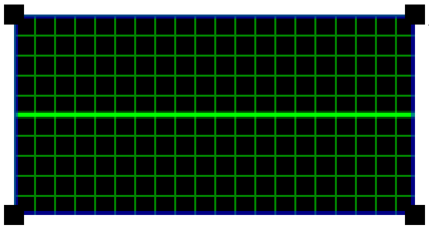
Note this section only applies to the C++ sender

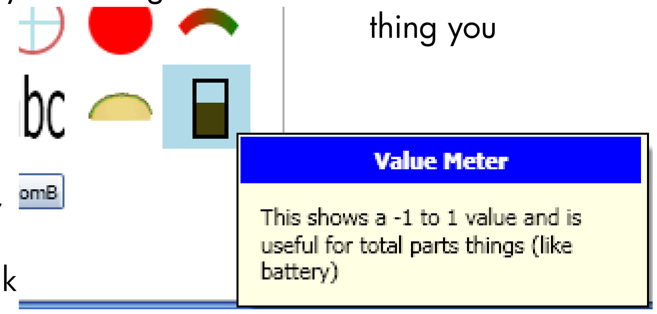
There are 3 main modes of communication, *TCP*, *DBPacket*, and *RemoteData*. There are also some pre-build combinations, *AllTCP*, *Both*, and *All*. The 3 you will most likely use most often are *TCP*, *AllTCP*, and *DBPacket*. Once you are initialized, you can use the Add(*name*, *value*) function, the var(*name*, *value*) (or the longer AddDebugVariable) function, where you are sending *value* to *name*. Also, if you are using *RemoteData*, you can also use GetString(*name*), GetDouble(*name*), etc... Sample:

```
void OperatorControl()
{
   GetWatchdog().SetEnabled(true);
   while (IsOperatorControl())
   {
      GetWatchdog().Feed();

      if (zomB.CanSend())
      {
         zomB.Add("taco", stick.GetY());
         zomB.var("taco", stick.GetY());//use a DebugGrid to see
         myRobotDrive.Drive(stick.GetY(), zomB.GetFloat("in"));
         zomB.Send();
      }
      Wait(0.01);
   }
}
```

For the ZomB Protocol, refer to http://firstforge.wpi.edu/sf/wiki/do/viewPage/projects.zombdashboard/wiki/ZomBProtocol.

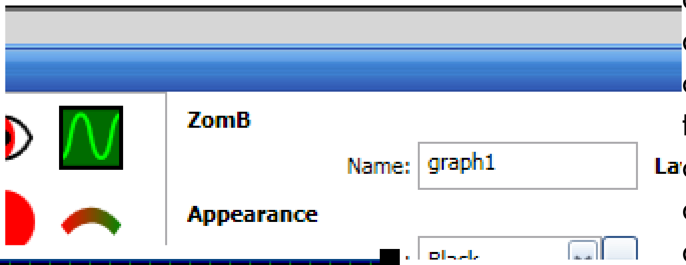
Creating A Custom ZomB Dashboard



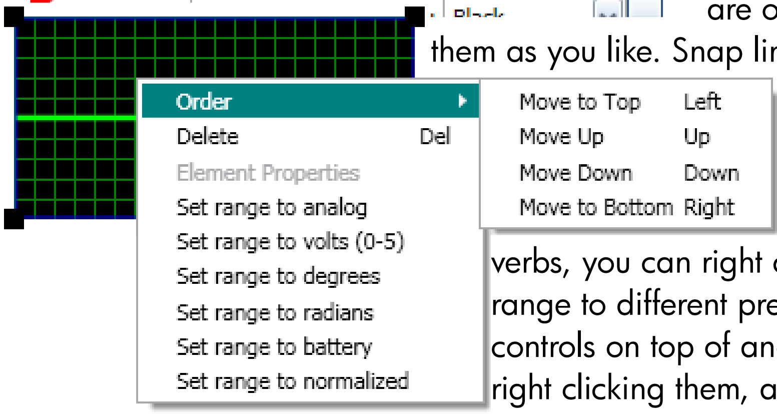


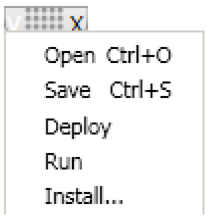
Value Meter

This shows a -1 to 1 value and is useful for total parts things (like battery)



Name: graph1



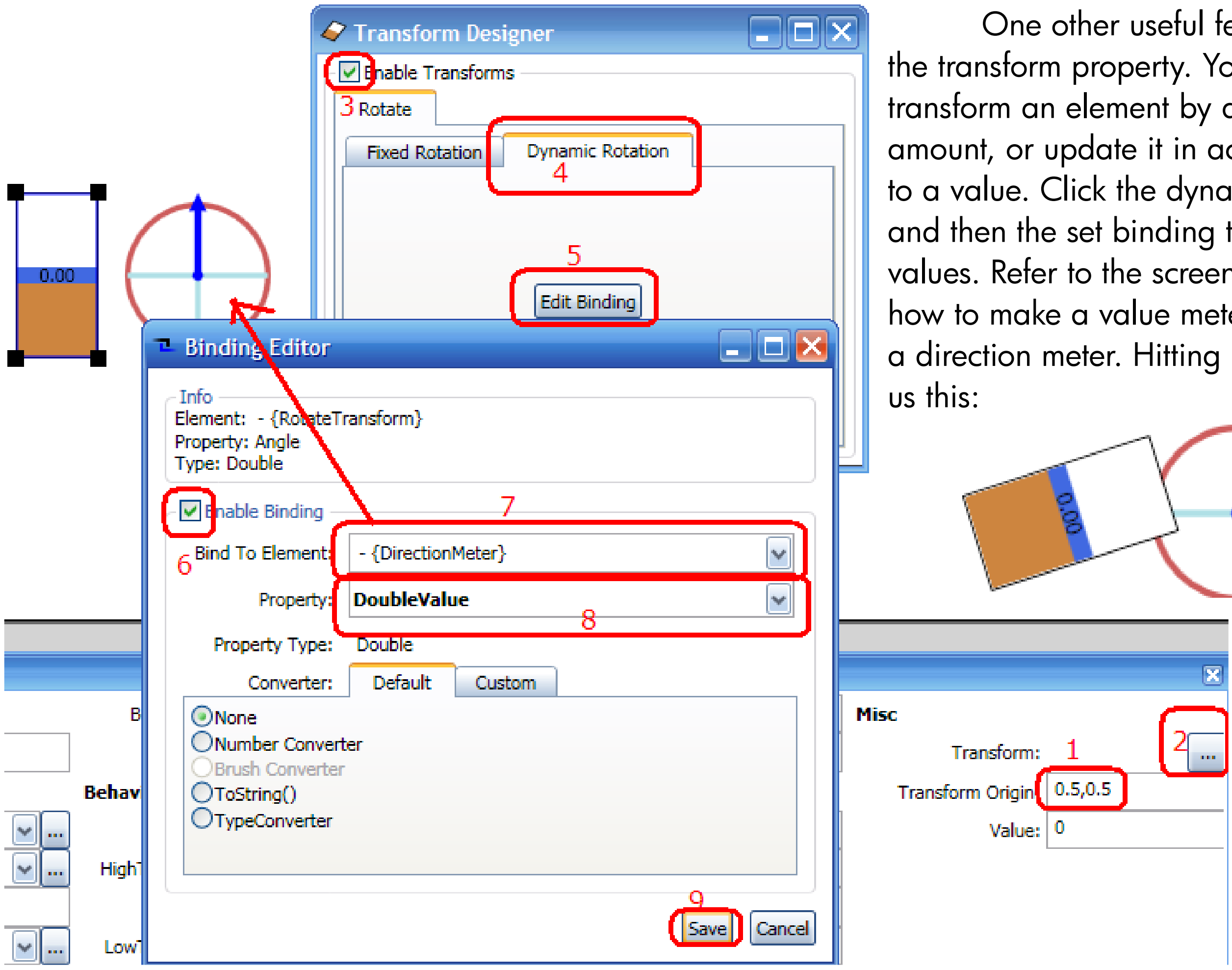


Creating a custom dashboard is easy, start by launching Visual ZomB from *Start>All Programs>ZomB>Visual ZomB*. The first thing you should do is to set your team number in the toolbox, and then what you will be receiving (dashboard packet, tcp, etc...) You can change these later by clicking a blank area of the designer surface. Start designing by dragging controls from the toolbox on the bottom left onto the design surface. If you don't know what a control is, hover over it, and useful tooltip will appear with a small explanation. Once controls are on the surface, drag them around and arrange them as you like. Snap lines will help with precision arranging. Make sure to name all your controls so they will be able to be referenced by the robot. You can also edit any other property of the control, like the colors. Many controls also come with verbs, you can right click them, and change their range to different preset values. You can arrange controls on top of and below other controls by right clicking them, and going under Order. Save a file by clicking the menu at the top of the surface, and going to save. Deploy will build the current dashboard to a single exe and save it in the FRC Dashboard folder so the Driver station will auto load it. Run will run the current dashboard, and can also be accessed by pressing F5. Install can be safely ignored. When your screen is small (width less than 1024, aka the classmate), the standard window chrome will disappear, and to move the window, use the grip area of the menu. The x will close it. Closing the toolbox will also close the window.

Advanced Editing

There are several more advanced aspects of Visual ZomB. One is the gradient editor. Start by clicking the ... button to the right of a color. This will open the brush designer. Switch to the gradient tab. You should see 2 stops, both the same color. Slide a

RGB slider at the top to change one of the colors. You can move the positions around, add stops, and delete stops. You can also set the gradient to be horizontal or vertical. All is updated in real time.



Transform Designer

[✓] Enable Transforms

Rotate

Fixed Rotation | Dynamic Rotation

Edit Binding

Binding Editor

Info
Element: - {RotateTransform}
Property: Angle
Type: Double

[✓] Enable Binding

Bind To Element:	- {DirectionMeter}
Property:	**DoubleValue**
Property Type:	Double
Converter:	Default | Custom

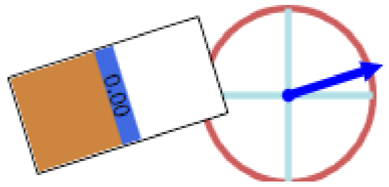
[✓ None]
Number Converter
Brush Converter
ToString()
TypeConverter

Save Cancel

Misc

Transform:	...
Transform Origin	0.5,0.5
Value:	0

One other useful feature is the transform property. You can transform an element by a set amount, or update it in accordance to a value. Click the dynamic tab, and then the set binding to use the values. Refer to the screen shot on how to make a value meter rotate to a direction meter. Hitting F5 gives us this:



More To Come Soon...