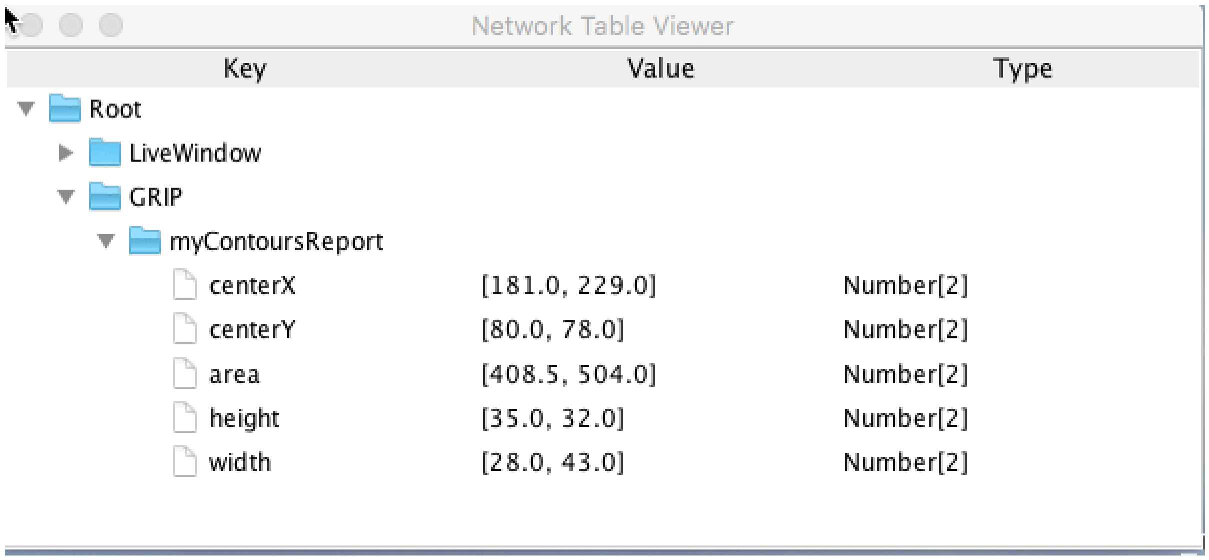
Reading array values published by NetworkTables

This article describes how to read values published by NetworkTables using a program running on the robot. This is useful when using computer vision where the images are processed on your driver station laptop and the results stored into NetworkTables possibly using a separate vision processor like a raspberry pi, or a tool on the robot like GRIP, or a python program to do the image processing.

Very often the values are for one or more areas of interest such as goals or game pieces and multiple instances are returned. In the example below, several x, y, width, height, and areas are returned by the image processor and the robot program can sort out which of the returned values are interesting through further processing.

Verify the network table keys being published



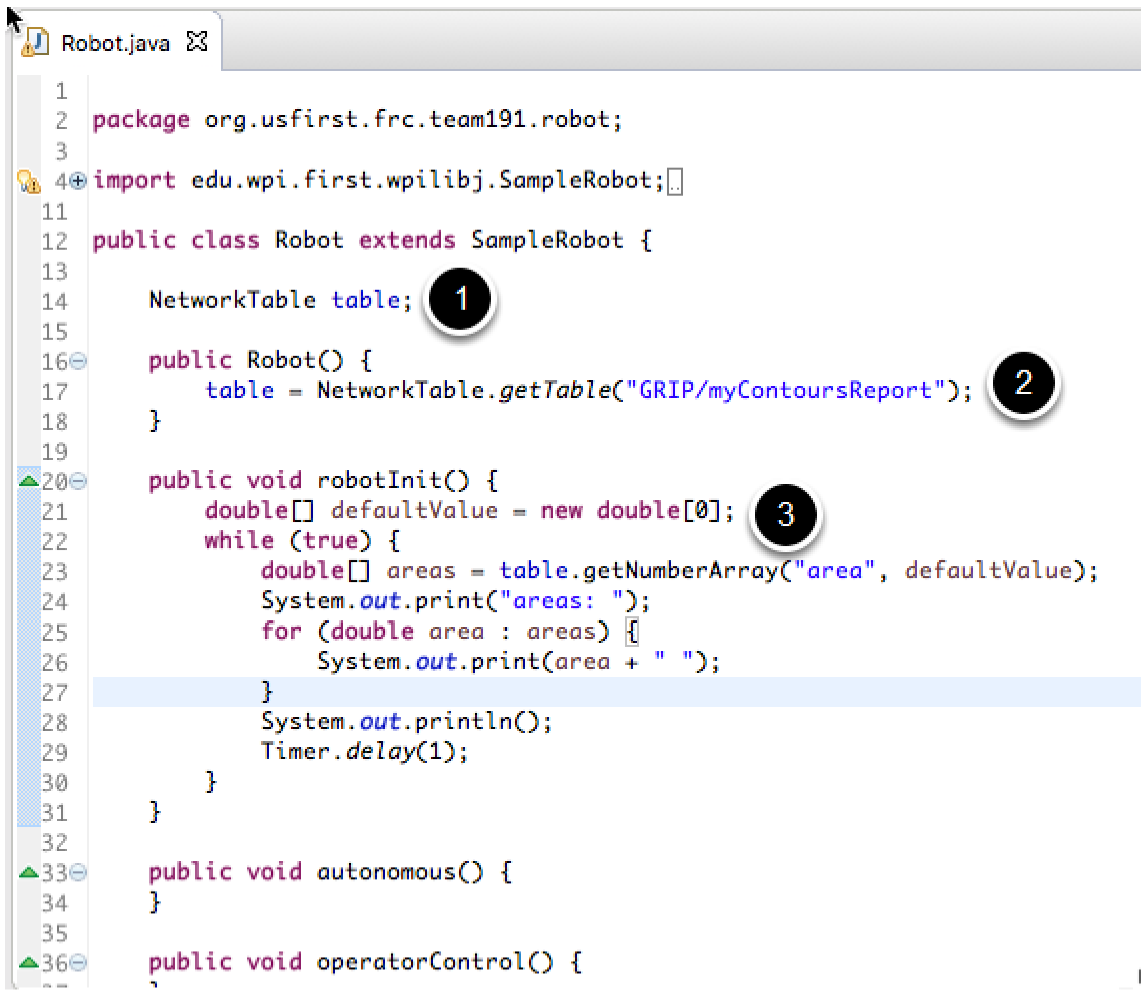
Key	Value	Type
centerX	[181.0, 229.0]	Number[2]
centerY	[80.0, 78.0]	Number[2]
area	[408.5, 504.0]	Number[2]
height	[35.0, 32.0]	Number[2]
width	[28.0, 43.0]	Number[2]

You can verify the names of the network table keys used for publishing the values by using the Network Table Viewer application. It is a java program in your user directory in the wpilib/tools folder. The application is started by selecting the "WPILib" menu in eclipse then "OutlineViewer". In this example, wth the image processing program running (GRIP) you can see the values being put into NetworkTables.

In this case the values are stored in a table called GRIP and a sub-table called myContoursReport. You can see that the values are in brackets and there are 2 values in this case for each key. The network table key names are centerX, centerY, area, height and width.

Both of the following examples are extremely simplified programs that just illustrate the use of NetworkTables. All the code is in the robotInit() method so it's only run when the program starts up. In your programs, you would more likely get the values in code that is evaluating which direction to aim the robot in a command or a control loop during the autonomous or teleop periods.

Writing a Java program to access the keys



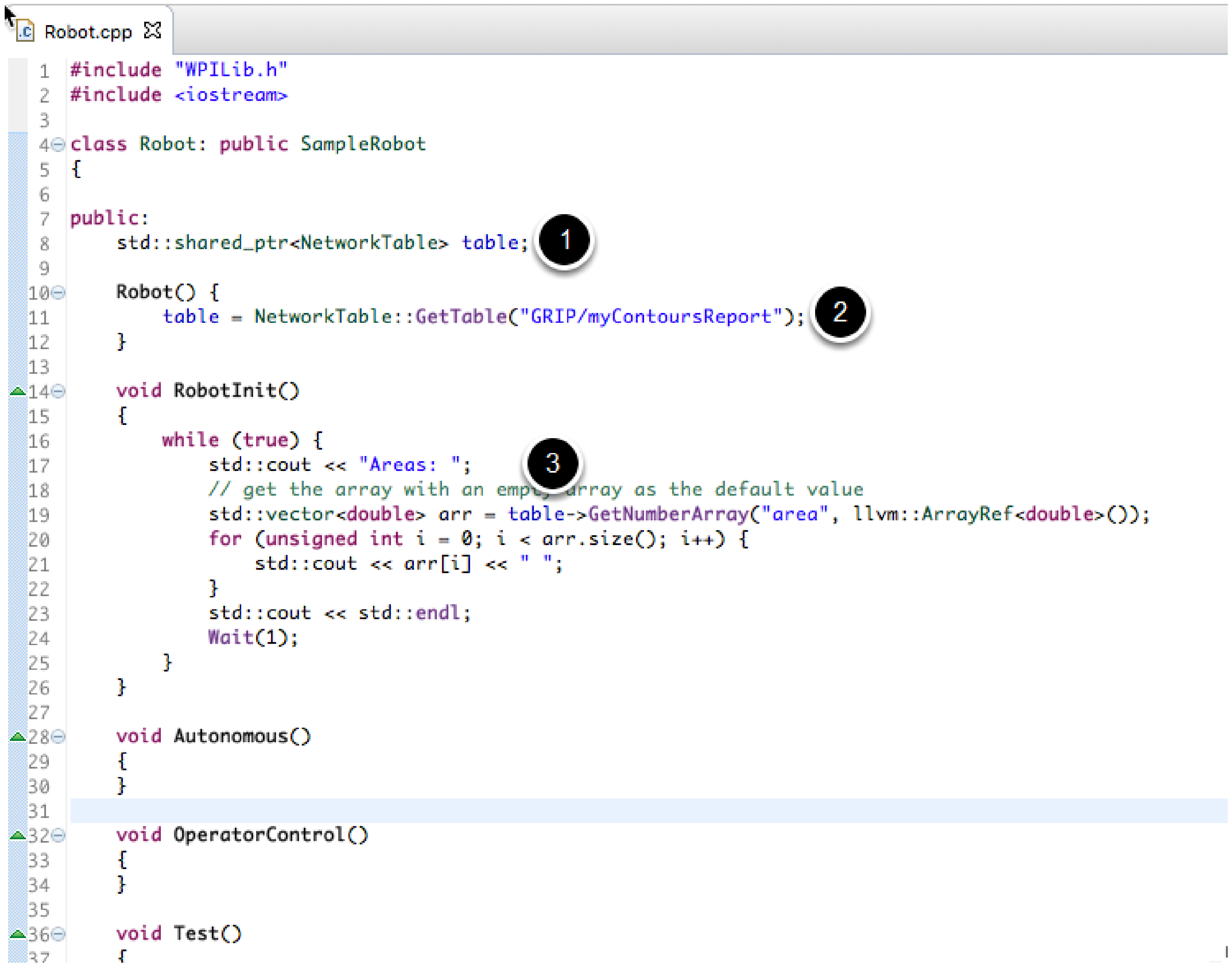
```
package org.usfirst.frc.team191.robot;

import edu.wpi.first.wpilibj.SampleRobot;

public class Robot extends SampleRobot {

    NetworkTable table;

    public Robot() {
        table = NetworkTable.getTable("GRIP/myContoursReport");
    }

    public void robotInit() {
        double[] defaultValue = new double[0];
        while (true) {
            double[] areas = table.getNumberArray("area", defaultValue);
            System.out.print("areas: ");
            for (double area : areas) {
                System.out.print(area + " ");
            }
            System.out.println();
            Timer.delay(1);
        }
    }

    public void autonomous() {
    }

    public void operatorControl() {
```

The steps to getting the values and, in this program, printing them are:

1. Declare the table variable that will hold the instance of the subtable that have the values.

2. Initialize the subtable instance so that it can be used later for retrieving the values.
3. Read the array of values from NetworkTables. In the case of a communicating programs, it's possible that the program producing the output being read here might not yet be available when the robot program starts up. To avoid issues of the data not being ready, a default array of values is supplied. This default value will be returned if the network table key hasn't yet been published. This code just loops forever and reads values and prints them to the console.

Writing a C++ program to access the keys

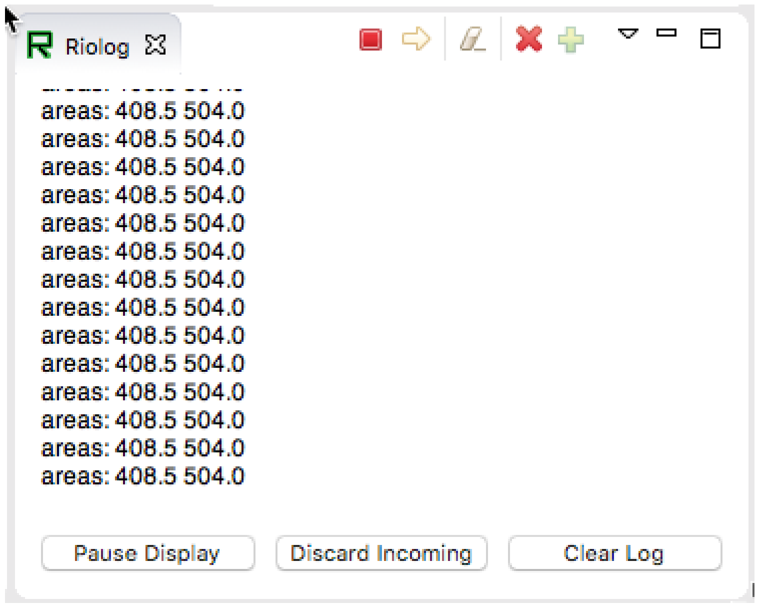


```
#include "WPILib.h"
#include <iostream>

class Robot: public SampleRobot
{

public:
    std::shared_ptr<NetworkTable> table;

    Robot() {
        table = NetworkTable::GetTable("GRIP/myContoursReport");
    }

    void RobotInit()
    {
        while (true) {
            std::cout << "Areas: ";
            // get the array with an empty array as the default value
            std::vector<double> arr = table->GetNumberArray("area", llvm::ArrayRef<double>());
            for (unsigned int i = 0; i < arr.size(); i++) {
                std::cout << arr[i] << " ";
            }
            std::cout << std::endl;
            Wait(1);
        }
    }

    void Autonomous()
    {
    }

    void OperatorControl()
    {
    }

    void Test()
    {
```

The steps to getting the values and, in this program, printing them are:

1. Declare the table variable that will hold the instance of the subtable that have the values. It is a shared pointer where the library takes care of allocation and deallocation automatically.
2. Initialize the subtable instance so that it can be used later for retrieving the values.
3. Read the array of values from NetworkTables. In the case of a communicating programs, it's possible that the program producing the output being read here might not yet be available when the robot program starts up. To avoid issues of the data not being ready, a default array of values is supplied. llvm::ArrayRef<double> creates this temporary array reference of zero length that would be returned if the network table key hasn't yet been published. This code just loops forever and reads values and prints them to the console.

Program output



```
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
areas: 408.5 504.0
```

In this case the program is only looking at the array of areas, but in a real example all the values would more likely be used. Using the Riolog in eclipse or the DriverStation log you can see the values as they are retrieved. This program is using a sample static image so they areas don't change, but you can imagine with a camera on your robot, the values would be changing constantly.