/\*----------------------------------------------------------------------------\*/

/\* Copyright (c) FIRST 2008. All Rights Reserved. \*/

/\* Open Source Software - may be modified and shared by FRC teams. The code \*/

/\* must be accompanied by the FIRST BSD license file in the root directory of \*/

/\* the project. \*/

/\*----------------------------------------------------------------------------\*/

//<<// FRC Team 313 2013 Code - Programmed By: Kyle Spurlock, Captain of Information Technology \\>>\\

package edu.wpi.first.wpilibj.templates;

import edu.wpi.first.wpilibj.Compressor;

import edu.wpi.first.wpilibj.IterativeRobot;

import edu.wpi.first.wpilibj.Joystick;

import edu.wpi.first.wpilibj.Relay;

import edu.wpi.first.wpilibj.RobotDrive;

import edu.wpi.first.wpilibj.Timer;

import edu.wpi.first.wpilibj.Victor;

import edu.wpi.first.wpilibj.smartdashboard.SmartDashboard;

public class RobotTemplate extends IterativeRobot {

//THIS IS WHERE ALL OF THE VARIABLES ARE DECLARED\\

//These objects control physical things on the robot.

RobotDrive robotDrive = new RobotDrive(2, 1, 4, 3); //Drive Motors

Joystick leftJoystick = new Joystick(1); //First Joystick

Joystick rightJoystick = new Joystick(2); //Second Joystick

Joystick controller = new Joystick(3); //Operator Controller

Victor shooter1 = new Victor(9); //First Shooter Motor

Victor shooter2 = new Victor(10); //Second Shooter Motor

Compressor compressor = new Compressor(1,1); //Compressor

//Relay hang = new Relay(3); //Hanging Pistons

Relay discIn = new Relay(5); //Piston Finger

Relay shooterAngle = new Relay(4); //Angle Piston

//These variables are used later in the code to tell the robot whether something is happening or not.

int currentShots; //Number of Shots Already Fired

int desiredShots; //Number of Shots Wanted

boolean prevHang;

boolean prevPush;

public void robotInit() {

//THIS METHOD IS CALLED ONCE ON ROBOT STARTUP\\

compressor.start(); //Starts Compressor

}

public void autonomousInit() {

//THIS METHOD IS CALLED ONCE ON AUTONOMOUS STARTUP\\

robotDrive.setSafetyEnabled(false);

discIn.set(Relay.Value.kReverse); //Retracts Finger Piston

shooter1.set(1); //Sets First Shooter Motor to 100% Power

shooter2.set(1); //Sets Second Shooter Motor to 100% Power

currentShots = 0; //Sets the Starting Value of Shots to 0

//desiredShots = 4; //Sets the Number of Shots Wanted to 3

Timer.delay(3.0); //Waits 3 Seconds for Shooter Motors to Spin Up

}

public void autonomousPeriodic() {

if (currentShots < 4) {

discIn.set(Relay.Value.kForward);

Timer.delay(.5);

discIn.set(Relay.Value.kReverse);

robotDrive.drive(0, 0.7);

Timer.delay(.25);

robotDrive.drive(0, 0);

Timer.delay(2);

currentShots++;

}

else if (currentShots < 12) {

discIn.set(Relay.Value.kForward);

Timer.delay(.5);

discIn.set(Relay.Value.kReverse);

robotDrive.drive(0, -0.7);

Timer.delay(.25);

robotDrive.drive(0, 0);

Timer.delay(2);

currentShots++;

}

else if (currentShots == 12) {

shooter1.set(0);

shooter2.set(0);

}

}

/\*

public void autonomousPeriodic() {

//THIS METHOD IS CALLED PERIODICALLY, SO LONG AS AUTONOMOUS IS ENABLED\\

if (currentShots < desiredShots) { //Checks to See if Number of Shots Already Fired is Less Than 3, The Desired Amount of Shots

discIn.set(Relay.Value.kReverse); //Retracts Finger Piston

Timer.delay(1.5); //Waits 1.5 Seconds for Shooter Motors to Spin Up

discIn.set(Relay.Value.kForward); //Extends Finger Piston and Shoots Frisbee

Timer.delay(1.5); //Waits 1.5 Seconds for Shooter Motors to Spin Up

currentShots++; //Adds One to the Number of Shots Taken

} else { //If currentShots is not Less Than desiredShots, then...

Timer.delay(1.0); //Waits 1 Second

shooter1.set(0.0); //Stops the First Shooter Motor

shooter2.set(0.0); //Stops the Second Shooter Motor

discIn.set(Relay.Value.kReverse); //Retracts Finger Piston

}

}

\*/

public void teleopInit() {

//THIS METHOD IS CALLED ONCE ON TELEOP STARTUP\\

prevHang = false;

prevPush = false;

robotDrive.setSafetyEnabled(true);

//robotDrive.tankDrive(leftJoystick, rightJoystick); //Sets up the Drive Motors to be used for Tank Drive (Left joystick controls left wheels, while right joystick controls right wheels.)

}

public void teleopPeriodic() {

//THIS METHOD IS CALLED PERIODICALLY, SO LONG AS TELEOP IS ENABLED//

if (controller.getAxis(Joystick.AxisType.kY) > 0.25) { //Checks the Controller's Analog Stick's Value along its Y Axis: If it's being pushed up.

shooterAngle.set(Relay.Value.kForward); //Inclines the Shooter

}

else if (controller.getAxis(Joystick.AxisType.kY) < -0.25) { //Checks the Controller's Analog Stick's Value along its Y Axis: If it's being pushed down.

shooterAngle.set(Relay.Value.kReverse); //Declines the Shooter

} else { //If the Controller's Analog Stick isn't being moved either Forwards or Backwards...

shooterAngle.set(Relay.Value.kOff); //Prevents the Shooter from changing Angle

}

/\*if (controller.getRawButton(2)) {

if (!prevHang) {

hang.set(Relay.Value.kForward);

} else {

hang.set(Relay.Value.kOff);

}

prevHang = true;

} else {

if (prevHang) {

hang.set(Relay.Value.kReverse);

} else {

hang.set(Relay.Value.kOff);

}

prevHang = false;

}\*/

if (controller.getRawButton(4)) {

shooter1.set(1);

shooter2.set(1);

} else {

shooter1.set(0);

shooter2.set(0);

}

if (controller.getRawButton(6)) {

if (!prevPush) {

discIn.set(Relay.Value.kForward);

} else {

discIn.set(Relay.Value.kOff);

}

prevPush = true;

} else {

if (prevPush) {

discIn.set(Relay.Value.kReverse);

} else {

discIn.set(Relay.Value.kOff);

}

prevPush = false;

}

//robotDrive.tankDrive(-leftJoystick.getRawAxis(2), rightJoystick.getRawAxis(2), false);

robotDrive.tankDrive(leftJoystick, rightJoystick);

//robotDrive.tankDrive(-leftJoystick.getAxis(Joystick.AxisType.kX), rightJoystick.getAxis(Joystick.AxisType.kX), false);

//if (leftJoystick.getAxis(Joystick.AxisType.kY))

SmartDashboard.putNumber("Left Wheels", leftJoystick.getAxis(Joystick.AxisType.kY));

SmartDashboard.putNumber("Right Wheels", rightJoystick.getAxis(Joystick.AxisType.kY));

SmartDashboard.putNumber("Shooter", (shooter1.get() + shooter2.get()) / 2);

SmartDashboard.putNumber("Shooter Angle", controller.getAxis(Joystick.AxisType.kY));

SmartDashboard.putBoolean("Pushing", prevPush);

SmartDashboard.putBoolean("Hanging", prevHang);

}

}