//NOTES

//make button to switch between operator preferences

#include "WPILib.h"

#include "NetworkTables/NetworkTable.h"

class Robot: public IterativeRobot

{

public:

Victor \*leftRear, \*leftRearCenter, \*leftFrontCenter, \*leftFront, \*rightRear, \*rightRearCenter, \*rightFrontCenter, \*rightFront;

Joystick \*leftJoy, \*rightJoy, \*controller;

Compressor \*compressor;

DoubleSolenoid \*bertha, \*forks;

DigitalInput \*limitSwitch;

NetworkTable \*table;

Timer \*timer;

//variables

int autoStrat = 0;

int NUM\_AUTO\_MODES = 3;

int opSettings = 0;

int NUM\_OP\_MODES = 5;

//vision

const int SCREEN\_SIZE\_X = 320;

const int SCREEN\_SIZE\_Y = 240;

const int SCREEN\_CENTER\_X = SCREEN\_SIZE\_X / 2;

const int SCREEN\_CENTER\_Y = SCREEN\_SIZE\_Y / 2;

Robot()

{

//drivetrain motors

//left back wheel

leftRear = new Victor(0); //red

leftRearCenter = new Victor(1); //green

//left front wheel

leftFrontCenter = new Victor(2); //blue

leftFront = new Victor(3); //yellow

//right rear wheel

rightRear = new Victor(4); //black

rightRearCenter = new Victor(5); //white

//right front wheel

rightFrontCenter = new Victor(6); //blue-red

rightFront = new Victor(7); //black-red

//driver and operator controls

leftJoy = new Joystick(0);

rightJoy = new Joystick(1);

controller = new Joystick(2);

//compressor

compressor = new Compressor(0);

//pneumatics

bertha = new DoubleSolenoid(0, 4);

forks = new DoubleSolenoid(2, 3);

//limit switch

limitSwitch = new DigitalInput(0);

//network tables

table = NetworkTable::GetTable("SmartDashboard");

//timers

timer = new Timer();

}

void RobotInit()

{

SmartDashboard::init();

compressor->SetClosedLoopControl(true);

}

void AutonomousInit()

{

switch(autoStrat)

{

case 0:

Auto\_OneTote();

break;

case 1:

AutoFunc\_DriveBack(5);

break;

case 2:

Auto\_DoNothing();

}

}

void AutonomousPeriodic() { }

void TeleopInit() { }

void TeleopPeriodic()

{

//OPERATOR CONTROLS

switch(opSettings)

{

case 0:

opMitch();

break;

case 1:

opOli();

break;

case 2:

opAlli();

break;

case 3:

opShana();

break;

case 4:

opMatt();

break;

}

//DRIVER CONTROLS

if (leftJoy->GetRawButton(5) || rightJoy->GetRawButton(10)) //Buttons 6 and 11 on joysticks

{

TankDrive(-leftJoy->GetRawAxis(1) \* 0.5, -rightJoy->GetRawAxis(1) \* 0.5);

}

TankDrive(-leftJoy->GetRawAxis(1), -rightJoy->GetRawAxis(1));

}

void DisabledPeriodic()

{

if (controller->GetRawButton(3)) //Button Y on controller

{

Wait(0.4);

if (autoStrat < NUM\_AUTO\_MODES - 1)

{

autoStrat++;

}

else

{

autoStrat = 0;

}

if (autoStrat == 0) { SmartDashboard::PutString("Auto Mode", "1 Tote"); }

else if (autoStrat == 1) { SmartDashboard::PutString("Auto Mode", "Drive Back Only"); }

else if (autoStrat == 2) { SmartDashboard::PutString("Auto Mode", "Do Nothing"); }

}

if (controller->GetRawButton(6)) //Back button on controller

{

Wait(0.4);

if (opSettings < NUM\_AUTO\_MODES - 1)

{

opSettings++;

}

else

{

opSettings = 0;

}

if (opSettings == 0) { SmartDashboard::PutString("Op Mode", "Mitch"); }

else if (opSettings == 1) { SmartDashboard::PutString("Op Mode", "Oli"); }

else if (opSettings == 2) { SmartDashboard::PutString("Op Mode", "Alli"); }

else if (opSettings == 3) { SmartDashboard::PutString("Op Mode", "Shana"); }

else if (opSettings == 4) { SmartDashboard::PutString("Op Mode", "Matt"); }

}

}

void TestInit()

{

leftRear->Set(1);

Wait(3.0);

leftRear->Set(0);

leftRearCenter->Set(1);

Wait(3.0);

leftRearCenter->Set(0);

leftFrontCenter->Set(1);

Wait(3);

leftFrontCenter->Set(0);

leftFront->Set(1);

Wait(3);

leftFront->Set(0);

rightRear->Set(1);

Wait(3);

rightRear->Set(0);

rightRearCenter->Set(1);

Wait(3);

rightRearCenter->Set(0);

rightFrontCenter->Set(1);

Wait(3);

rightFrontCenter->Set(0);

rightFront->Set(1);

Wait(3);

rightFront->Set(0);

}

void TankDrive(double left, double right)

{

leftFront->Set(left);

leftFrontCenter->Set(left);

leftRear->Set(left);

leftRearCenter->Set(left);

rightFront->Set(-right);

rightFrontCenter->Set(-right);

rightRear->Set(-right);

rightRearCenter->Set(-right);

SmartDashboard::PutNumber("Left Drivetrain", left);

SmartDashboard::PutNumber("Right Drivetrain", -right);

}

void Wait(double time)

{

timer->Reset();

timer->Start();

while (timer->Get() < time) { }

timer->Stop();

}

///

//////

//OPERATOR SWITCH

//////

///

void opMitch()

{

if (controller->GetRawAxis(1) > 0.25) //Left Thumbstick on controller

{

bertha->Set(DoubleSolenoid::kForward);

}

else if (controller->GetRawAxis(1) < -0.25)

{

bertha->Set(DoubleSolenoid::kReverse);

}

else

{

bertha->Set(DoubleSolenoid::kOff);

}

if (controller->GetRawButton(0)) //A Button on controller

{

forks->Set(DoubleSolenoid::kForward);

}

else

{

forks->Set(DoubleSolenoid::kReverse);

}

}

void opOli()

{

if (controller->GetRawAxis(1) > 0.25) //Left Thumbstick on controller

{

bertha->Set(DoubleSolenoid::kReverse);

}

else if (controller->GetRawAxis(1) < -0.25)

{

bertha->Set(DoubleSolenoid::kForward);

}

else

{

bertha->Set(DoubleSolenoid::kOff);

}

if (controller->GetRawButton(0)) //A Button on controller

{

forks->Set(DoubleSolenoid::kForward);

}

else

{

forks->Set(DoubleSolenoid::kReverse);

}

}

void opAlli()

{

if (controller->GetRawAxis(1) > 0.25) //Left Thumbstick on controller

{

bertha->Set(DoubleSolenoid::kReverse);

}

else if (controller->GetRawAxis(1) < -0.25)

{

bertha->Set(DoubleSolenoid::kForward);

}

else

{

bertha->Set(DoubleSolenoid::kOff);

}

if (controller->GetRawButton(0)) //A Button on controller

{

forks->Set(DoubleSolenoid::kForward);

}

else

{

forks->Set(DoubleSolenoid::kReverse);

}

}

void opShana()

{

if (controller->GetRawAxis(5) > 0.25) //Left Thumbstick on controller

{

bertha->Set(DoubleSolenoid::kReverse);

}

else if (controller->GetRawAxis(5) < -0.25)

{

bertha->Set(DoubleSolenoid::kForward);

}

else

{

bertha->Set(DoubleSolenoid::kOff);

}

if (controller->GetRawAxis(3) > 0.25) //Right Trigger on controller

{

forks->Set(DoubleSolenoid::kForward);

}

else

{

forks->Set(DoubleSolenoid::kReverse);

}

}

void opMatt()

{

if (controller->GetRawAxis(1) > 0.25) //Left Thumbstick on controller

{

bertha->Set(DoubleSolenoid::kReverse);

}

else if (controller->GetRawAxis(1) < -0.25)

{

bertha->Set(DoubleSolenoid::kForward);

}

else

{

bertha->Set(DoubleSolenoid::kOff);

}

if (controller->GetRawAxis(3) > 0.25) //Right Trigger on controller

{

forks->Set(DoubleSolenoid::kForward);

}

else

{

forks->Set(DoubleSolenoid::kReverse);

}

}

///

//////

//AUTONOMOUS FUNCTIONS

//////

///

void AutoFunc\_GetFirstTote()

{

//open forks

forks->Set(DoubleSolenoid::kReverse);

Wait(1);

//drive small amount forward until tote locks in place and then stop moving

/\*

while (!limitSwitch->Get())

{

TankDrive(0.5, 0.5);

}

TankDrive(0, 0);

\*/

TankDrive(0.5, 0.5);

Wait(2);

TankDrive(0, 0);

//retract forks

forks->Set(DoubleSolenoid::kForward);

}

void AutoFunc\_DriveBack(double time)

{

TankDrive(-1.0, -1.0);

Wait(time);

TankDrive(0, 0);

}

void AutoFunc\_DropTotes()

{

//open forks to drop totes

forks->Set(DoubleSolenoid::kForward);

}

///

//////

//AUTONOMOUS MODES

//////

///

void Auto\_OneTote()

{

AutoFunc\_GetFirstTote();

AutoFunc\_DriveBack(3);

AutoFunc\_DropTotes();

AutoFunc\_DriveBack(1);

}

void Auto\_DoNothing()

{

//Do Nothing

}

};

START\_ROBOT\_CLASS(Robot);