#include "WPILib.h"

#include <math.h>

class RobotDemo : public IterativeRobot

{

public:

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

Joystick \*joy, \*controller;

Compressor \*compressor;

Relay \*bertha, \*forks;

Relay \*ledRing;

DigitalInput \*limitSwitch;

Timer \*timer;

bool direction;

int autoStrat;

RobotDemo()

{

leftRear = new Victor(1);

leftRearCenter = new Victor(2); //left back wheel

leftFrontCenter = new Victor(3);

leftFront = new Victor(4); //left front wheel

rightRear = new Victor(5);

rightRearCenter = new Victor(6); //right rear wheel

rightFrontCenter = new Victor(7);

rightFront = new Victor(9); //right front wheel, Bad port 8 on Digital sidecar (Breadboard)

joy = new Joystick(1);

controller = new Joystick(2);

compressor = new Compressor(1, 8);

bertha = new Relay(2);

forks = new Relay(3);

ledRing = new Relay(7);

limitSwitch = new DigitalInput(2);

timer = new Timer();

direction = true; //True = Right, False = Left

autoStrat = 0;

}

void RobotDemo::RobotInit()

{

compressor->Start();

}

void RobotDemo::AutonomousInit()

{

switch(autoStrat)

{

case 0:

Auto\_GetToteMoveBack(); //get first tote, then move back

break;

case 1:

direction = true; //strafe right

Auto\_Strafe();

break;

case 2:

direction = false; //strafe left

Auto\_Strafe();

break;

case 3:

Auto\_MoveBack(); //move back, but don't get first tote

break;

case 4:

//////////insert mode

}

}

void RobotDemo::AutonomousPeriodic()

{

}

void RobotDemo::TeleopInit()

{

}

void RobotDemo::TeleopPeriodic()

{

ArcadeShift(joy->GetRawAxis(1), -joy->GetRawAxis(2), joy->GetRawAxis(3));

if (controller->GetRawAxis(2) > 0.25)

{

bertha->Set(Relay::kReverse);

}

else if (controller->GetRawAxis(2) < -0.25)

{

bertha->Set(Relay::kForward);

}

else

{

bertha->Set(Relay::kOff);

}

if (controller->GetRawButton(1))

{

forks->Set(Relay::kForward);

}

else

{

forks->Set(Relay::kReverse);

}

if (limitSwitch->Get())

{

ledRing->Set(Relay::kOn);

}

else

{

ledRing->Set(Relay::kOff);

}

DriverStationLCD \*dsLCD = DriverStationLCD::GetInstance();

dsLCD->Printf(DriverStationLCD::kUser\_Line1, 1, "X: %f", joy->GetRawAxis(1));

dsLCD->Printf(DriverStationLCD::kUser\_Line2, 1, "Y: %f", -joy->GetRawAxis(2));

dsLCD->Printf(DriverStationLCD::kUser\_Line3, 1, "Z: %f", joy->GetRawAxis(3));

if (limitSwitch->Get())

{

dsLCD->Printf(DriverStationLCD::kUser\_Line4, 1, "Limit Switch: %d", 1);

}

else

{

dsLCD->Printf(DriverStationLCD::kUser\_Line4, 1, "Limit Switch: %d", 0);

}

dsLCD->UpdateLCD();

}

void RobotDemo::TestPeriodic()

{

double y = joy->GetRawAxis(2);

leftRear->Set(-y);

}

void RobotDemo::DisabledPeriodic()

{

if (controller->GetRawButton(4))

{

Wait(0.4);

if (autoStrat < 4)

{

autoStrat++;

}

else

{

autoStrat = 0;

}

DriverStationLCD \*dsLCD = DriverStationLCD::GetInstance();

if (autoStrat == 0) { dsLCD->Printf(DriverStationLCD::kUser\_Line6, 1, "1,Back\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"); }

else if (autoStrat == 1) { dsLCD->Printf(DriverStationLCD::kUser\_Line6, 1, "1,Strafe\_R,2,Back\_\_\_\_\_\_\_\_\_\_\_\_"); }

else if (autoStrat == 2) { dsLCD->Printf(DriverStationLCD::kUser\_Line6, 1, "1,Strafe\_L,2,Back\_\_\_\_\_\_\_\_\_\_\_\_"); }

else if (autoStrat == 3) { dsLCD->Printf(DriverStationLCD::kUser\_Line6, 1, "Only\_Move\_Back\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"); }

else if (autoStrat == 4) { dsLCD->Printf(DriverStationLCD::kUser\_Line6, 1, "1,L2,L3,Back\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"); }

else if (autoStrat == 5) { dsLCD->Printf(DriverStationLCD::kUser\_Line6, 1, "1,R2,R3,Back\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"); }

dsLCD->Printf(DriverStationLCD::kUser\_Line5, 1, "Autonomous Mode: %d", autoStrat);

dsLCD->UpdateLCD();

}

}

void AutoFunc\_GetFirstTote()

{

//open forks

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

//retract forks

}

void AutoFunc\_Strafe(bool direction) //right is true, left is false

{

if (direction)

{

//strafe right

}

else

{

//strafe left

}

//vision tracking

}

void AutoFunc\_GetTote()

{

//move arm up

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

//lower arm to set top tote on bottom tote

//open forks to drop top tote onto bottom tote

//lower arm to grab bottom tote

//lift bottom tote up

}

void AutoFunc\_DriveBackShort()

{

//drive back short amount

}

void AutoFunc\_DriveBackLong()

{

//drove back long amount

}

void Auto\_Strafe\_TwoTotes(bool direction)

{

AutoFunc\_GetFirstTote();

AutoFunc\_DriveBackShort();

AutoFunc\_Strafe(direction);

AutoFunc\_GetTote();

AutoFunc\_DriveBackLong();

}

void Auto\_OneTote()

{

AutoFunc\_GetFirstTote();

AutoFunc\_DriveBackLong();

}

void Auto\_ThreeTotes(bool direction)

{

AutoFunc\_GetFirstTote();

AutoFunc\_DriveBackShort();

AutoFunc\_Strafe(direction);

AutoFunc\_GetTote();

AutoFunc\_DriveBackShort();

AutoFunc\_Strafe(direction);

AutoFunc\_GetTote();

AutoFunc\_DriveBackLong();

}

/\*

0. One Tote -> Auto\_OneTote();

1. Two Totes - Strafe Right -> Auto\_Strafe\_TwoTotes(true);

2. Two Totes - Strafe Left -> Auto\_Strafe\_TwoTotes(false);

3. Back Only -> AutoFunc\_DriveBackLong();

4. Three Totes Right -> Auto\_ThreeTotes(true);

5. Three Totes Left -> Auto\_ThreeTotes(false);

\*/

void Auto\_Strafe()

{

//open forks

forks->Set(Relay::kReverse);

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

while (!limitSwitch->Get())

{

ArcadeShift(0, 0.3, 0);

}

ArcadeShift(0, 0, 0);

//retract forks

forks->Set(Relay::kForward);

//drive back then stop

ArcadeShift(0, -0.3, 0);

Wait(2);

ArcadeShift(0, 0, 0);

//strafe either left or right (replace with vision tracking while loop)

if (direction = true)

{

ArcadeShift(0.8, 0, 0);

}

else

{

ArcadeShift(-0.8, 0, 0);

}

Wait(5);

ArcadeShift(0, 0, 0);

//move arm up

bertha->Set(Relay::kReverse);

Wait(0.5);

bertha->Set(Relay::kOff);

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

while (!limitSwitch->Get())

{

ArcadeShift(0, 0.3, 0);

}

ArcadeShift(0, 0, 0);

//lower arm to set top tote on bottom tote

bertha->Set(Relay::kForward);

Wait(0.5);

bertha->Set(Relay::kOff);

//open arms to drop top tote onto bottom tote

Wait(0.25);

forks->Set(Relay::kReverse);

Wait(0.25);

//lower arm to grab bottom tote

bertha->Set(Relay::kForward);

forks->Set(Relay::kForward);

Wait(0.25);

bertha->Set(Relay::kOff);

//lift bottom tote up

bertha->Set(Relay::kReverse);

Wait(0.5);

//reverse the robot into the autozone

ArcadeShift(0, -1.0, 0);

Wait(5);

ArcadeShift(0, 0, 0);

}

void Auto\_MoveBack()

{

ArcadeShift(0, -1.0, 0);

Wait(5);

ArcadeShift(0, 0, 0);

}

void Auto\_GetToteMoveBack()

{

//open forks

forks->Set(Relay::kReverse);

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

while (!limitSwitch->Get())

{

ArcadeShift(0, 0.3, 0);

}

ArcadeShift(0, 0, 0);

//retract forks

forks->Set(Relay::kForward);

//drive back then stop

ArcadeShift(0, -1.0, 0);

Wait(5);

ArcadeShift(0, 0, 0);

}

void Auto\_1Right2Right3Back()

{

}

void Auto\_1Left2Left3Left()

{

}

void ArcadeShift(double x, double y, double z)

{

double lf, lr, rf, rr; //lf = left front, lr = left rear, rf = right front, rr = right rear

lf = x + y + z;

lr = -x + y - z;

rf = -x + y - z;

rr = x + y - z;

double max = 0;

if (max < fabs(lf))

{

max = fabs(lf);

}

if (max < fabs(lr))

{

max = fabs(lr);

}

if (max < fabs(rf))

{

max = fabs(rf);

}

if (max < fabs(rr))

{

max = fabs(rr);

}

if (max > 1)

{

lf = lf/max;

lr = lr/max;

rf = rf/max;

rr = rr/max;

}

leftFront->Set(lf);

leftFrontCenter->Set(lf);

leftRear->Set(lr);

leftRearCenter->Set(lr);

rightFront->Set(rf);

rightFrontCenter->Set(rf);

rightRear->Set(rr);

rightRearCenter->Set(rr);

}

void Wait(double time)

{

timer->Reset();

timer->Start();

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

timer->Stop();

}

};

START\_ROBOT\_CLASS(RobotDemo);