//non robot includes

#include <iostream>

#include <memory>

#include <string>

#include <cmath>

//robot related includes

#include <IterativeRobot.h>

#include <LiveWindow/LiveWindow.h>

//#include <SmartDashboard/SendableChooser.h>

//#include <SmartDashboard/SmartDashboard.h>

#include "WPILib.h"

#include <PowerDistributionPanel.h>

#include "ctre/Phoenix.h"

#include "Drive/DifferentialDrive.h"

#include "DriverStation.h"

Compressor \*compressor;

BuiltInAccelerometer accelerometer;

class Robot: public frc::IterativeRobot {

public:

WPI\_TalonSRX l1 {32};

WPI\_TalonSRX l2 {38};

WPI\_TalonSRX l3 {39};

WPI\_TalonSRX r1 {33};

WPI\_TalonSRX r2 {34};

WPI\_TalonSRX r3 {35};

WPI\_TalonSRX arm1 {36};

WPI\_TalonSRX arm2 {37};

WPI\_TalonSRX w1 {41};

WPI\_TalonSRX w2 {30};

WPI\_TalonSRX w3 {31};

WPI\_TalonSRX m1 {40};

WPI\_TalonSRX m2 {42};

DoubleSolenoid \*left, \*right;

frc::Joystick joy {0};

frc::PowerDistributionPanel pdp {0};

std::shared\_ptr<NetworkTable> roboRealm;

const double kUpdatePeriod = 0.005; // 5milliseconds / 0.005 seconds.

static constexpr int kUltrasonicPort = 1;

static constexpr int kValueToInches = .125;

cs::UsbCamera camera;

bool seen = false;

bool turned = false;

int auton = 0;

int autonMax = 5;

Robot(){

compressor = new Compressor(0);

right = new DoubleSolenoid(2,3);

left = new DoubleSolenoid(0,1);

}

void RobotInit() /\*: accelerometer(Accelerometer::Range::kRange\_8G)\*/{

cs::UsbCamera camera = CameraServer::GetInstance()->StartAutomaticCapture();

compressor->SetClosedLoopControl(true);

}

void AutonomousInit() {

}

void DisabledPeriodic()

{

SmartDashboard::PutString("DB/String 7","");

if(joy.GetRawButton(6))

{

auton++;

Wait(.2);

if(auton == autonMax + 1)

{

auton = 0;

}

else if(auton == -1)

{

auton = 5;

}

}

else if(joy.GetRawButton(5))

{

auton--;

Wait(.2);

if(auton == autonMax + 1)

{

auton = 0;

}

else if(auton == -1)

{

auton = 5;

}

}

auto str = std::to\_string(auton);

SmartDashboard::PutString("DB/String 5","Auton mode: " + str);

if(auton == 0)

{

SmartDashboard::PutString("DB/String 6","GOTTA GO FAST(drive forward 7 seconds .75 power");

}

else if(auton == 1)

{

SmartDashboard::PutString("DB/String 6","OPEN THE BLAST DOORS");

}

else if(auton == 2)

{

SmartDashboard::PutString("DB/String 6","RIP SALLY");

}

else if(auton == 3)

{

SmartDashboard::PutString("DB/String 6","ROCKWALL");

}

else if(auton == 4)

{

SmartDashboard::PutString("DB/String 6","LOWBAR");

}

else if (auton == 5)

{

}

}

void TeleopPeriodic() {

if(joy.GetRawButton(5))

{

arm1.Set(.6);

arm2.Set(.6);

}

else if(joy.GetRawButton(6))

{

arm1.Set(-.6);

arm2.Set(-.6);

}

else

{

arm1.Set(0);

arm2.Set(0);

}

if(joy.GetRawButton(1))

{

w1.Set(.2);

w2.Set(.2);

w3.Set(.2);

}

else if(joy.GetRawButton(2))

{

w1.Set(-.2);

w2.Set(-.2);

w3.Set(-.2);

}

else

{

w1.Set(0);

w2.Set(0);

w3.Set(0);

}

if(joy.GetRawButton(0))

{

m1.Set(.2);

m2.Set(.2);

}

else if(joy.GetRawButton(3))

{

m1.Set(-.2);

m2.Set(-.2);

}

else

{

m1.Set(0);

m2.Set(0);

}

if(joy.GetRawAxis(2) > .1)

{

left->Set(DoubleSolenoid::kForward);

}

if(joy.GetRawAxis(4) > .1)

{

right->Set(DoubleSolenoid::kForward);

}

else

{

left->Set(DoubleSolenoid::kReverse);

right->Set(DoubleSolenoid::kReverse);

}

if(std::abs(joy.GetRawAxis(5)) > .2 && std::abs(joy.GetRawAxis(1)) < .2)

{

TankDrive(0, joy.GetRawAxis(5));//drive the right side only

//Reads the input voltage of the PDP

double pdpVin = pdp.GetVoltage();

printf("input voltage: %f \n", pdpVin);

//Reads the input amperage of the talons

double t1 = pdp.GetCurrent(12);

double t2 = pdp.GetCurrent(13);

double t3 = pdp.GetCurrent(14);

double t4 = pdp.GetCurrent(15);

//printf("t1: %f; t2: %f; t3: %f, t4: %f \n", t1, t2, t3, t4);

printf("Right side drive ");

}

else if(std::abs(joy.GetRawAxis(5)) < .2 && std::abs(joy.GetRawAxis(1)) > .2)

{

TankDrive(joy.GetRawAxis(1),0);//drive the left side only

//Reads the input voltage of the PDP

double pdpVin = pdp.GetVoltage();

printf("input voltage: %f \n", pdpVin);

//Reads the input amperage of the talons

double t1 = pdp.GetCurrent(12);

double t2 = pdp.GetCurrent(13);

double t3 = pdp.GetCurrent(14);

double t4 = pdp.GetCurrent(15);

//printf("t1: %f; t2: %f; t3: %f, t4: %f \n", t1, t2, t3, t4);

printf("Left side drive");

}

else if(std::abs(joy.GetRawAxis(5)) > .2 && std::abs(joy.GetRawAxis(1)) > .2)

{

TankDrive(joy.GetRawAxis(1),joy.GetRawAxis(5)); //drive both sides

//Reads the input voltage of the PDP

double pdpVin = pdp.GetVoltage();

printf("input voltage: %f \n", pdpVin);

//Reads the input amperage of the talons

double t1 = pdp.GetCurrent(12);

double t2 = pdp.GetCurrent(13);

double t3 = pdp.GetCurrent(14);

double t4 = pdp.GetCurrent(15);

//printf("t1: %f; t2: %f; t3: %f, t4: %f \n", t1, t2, t3, t4);

printf("Both sides");

}

else if(std::abs(joy.GetRawAxis(5)) > .2 && std::abs(joy.GetRawAxis(1)) < .2)

{

TankDrive(joy.GetRawAxis(1),joy.GetRawAxis(5)); //drive both sides

//Reads the input voltage of the PDP

double pdpVin = pdp.GetVoltage();

printf("input voltage: %f \n", pdpVin);

//Reads the input amperage of the talons

double t1 = pdp.GetCurrent(12);

double t2 = pdp.GetCurrent(13);

double t3 = pdp.GetCurrent(14);

double t4 = pdp.GetCurrent(15);

//printf("t1: %f; t2: %f; t3: %f, t4: %f \n", t1, t2, t3, t4);

printf("Both sides");

}

else

{

TankDrive(0,0);

}

}

void TankDrive(double left, double right)

{

l1.Set(-left);

l2.Set(-left);

l3.Set(-left);

r1.Set(right);

r2.Set(right);

r3.Set(right);

}

};

START\_ROBOT\_CLASS(Robot)