//NOTES

//Done by Mitchell Castaldini

#include "WPILib.h"

#include "Commands/Command.h"

#include "Commands/ExampleCommand.h"

#include "CommandBase.h"

#include "NetworkTables/NetworkTable.h"

class Robot: public IterativeRobot

{

public:

Victor \*V1, \*V2, \*V3, \*V4, \*V5, \*V6, \*V7, \*V8;

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

NetworkTable \*table;

DoubleSolenoid \*MPS1, \*MPS2, \*MPS3, \*MPS4;

Compressor \*compressor;

CANTalon \*T1, \*T2;

Timer \*timer;

Robot()

{

//drive train motors

//left back wheel

V1 = new Victor(0); //red

V2 = new Victor(1); //green

//left front wheel

V3 = new Victor(2); //blue

V4 = new Victor(3); //yellow

//right rear wheel

V5 = new Victor(4); //black

V6 = new Victor(5); //white

//right front wheel

V7 = new Victor(6); //blue-red

V8 = new Victor(7); //black-red

//driver and operator controls

leftJoy = new Joystick(2);

rightJoy = new Joystick(1);

controller = new Joystick(0);

//Talons

T1 = new CANTalon(0);

T2 = new CANTalon(1);

//compressor

compressor = new Compressor(0);

//pneumatics

MPS1 = new DoubleSolenoid(0, 1);

MPS2 = new DoubleSolenoid(2, 3);

MPS3 = new DoubleSolenoid(4, 5);

MPS4 = new DoubleSolenoid(6, 7);

//network tables

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

//timers

timer = new Timer();

}

void RobotInit() override

{

CameraServer::GetInstance()->SetQuality(50);

//the camera name (ex "cam0") can be found through the roborio web interface

CameraServer::GetInstance()->StartAutomaticCapture("cam0");

SmartDashboard::init();

compressor->SetClosedLoopControl(true);

}

void OperatorControl()

{

while (IsOperatorControl() && IsEnabled())

{

/\*\* robot code here! \*\*/

Wait(0.005); // wait for a motor update time

}

}

void AutonomousInit()

{

T1->Set(-1);

T2->Set(-1);

Wait(3);

T1->Set(0);

T2->Set(0);

V1->Set(1);

V2->Set(1);

V3->Set(1);

Wait(4);

V1->Set(0);

V2->Set(0);

V3->Set(0);

/\*

MPS1->Set(DoubleSolenoid::kForward);

MPS2->Set(DoubleSolenoid::kForward);

MPS3->Set(DoubleSolenoid::kForward);

MPS4->Set(DoubleSolenoid::kForward);

Wait(3);

MPS1->Set(DoubleSolenoid::kReverse);

MPS2->Set(DoubleSolenoid::kReverse);

MPS3->Set(DoubleSolenoid::kOff);

MPS4->Set(DoubleSolenoid::kOff);

\*/

}

void TeleopInit() { }

void TeleopPeriodic()

{

//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));

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

{

MPS4->Set(DoubleSolenoid::kForward);

}

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

{

MPS4->Set(DoubleSolenoid::kReverse);

}

else

{

MPS4->Set(DoubleSolenoid::kOff);

}

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

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

{

MPS3->Set(DoubleSolenoid::kForward);

}

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

{

MPS3->Set(DoubleSolenoid::kReverse);

}

else

{

MPS3->Set(DoubleSolenoid::kOff);

}

if (controller->GetRawButton(1)) //A Button on controller

{

MPS2->Set(DoubleSolenoid::kForward);

}

else

{

MPS2->Set(DoubleSolenoid::kReverse);

}

if(controller->GetRawButton(4))

{

MPS1->Set(DoubleSolenoid::kForward);

}

else

{

MPS1->Set(DoubleSolenoid::kReverse);

}

if(controller->GetRawButton(5))

{

V1->Set(1);

V2->Set(1);

V3->Set(1);

Wait(2);

T1->Set(1);

T2->Set(1);

V4->Set(1);

V5->Set(1);

Wait(.5);

V1->Set(0);

V2->Set(0);

V3->Set(0);

Wait(2.5);

T1->Set(0);

T2->Set(0);

V4->Set(-1);

V5->Set(-1);

Wait(2);

V4->Set(0);

V5->Set(0);

}

}

void DisabledPeriodic()

{

}

void TestInit()

{

V1->Set(1);

Wait(3.0);

V1->Set(0);

V2->Set(1);

Wait(3.0);

V2->Set(0);

V3->Set(1);

Wait(3);

V3->Set(0);

V4->Set(1);

Wait(3);

V4->Set(0);

V5->Set(1);

Wait(3);

V5->Set(0);

V6->Set(1);

Wait(3);

V6->Set(0);

V7->Set(1);

Wait(3);

V7->Set(0);

V8->Set(1);

Wait(3);

V8->Set(0);

T1->Set(1);

Wait(3);

T1->Set(0);

T2->Set(1);

Wait(3);

T2->Set(0);

}

void TankDrive(double left, double right)

{

V4->Set(left);

V3->Set(left);

V1->Set(left);

V2->Set(left);

V7->Set(-right);

V8->Set(-right);

V5->Set(-right);

V6->Set(-right);

T1->Set(left);

T2->Set(-right);

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

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

}

};

START\_ROBOT\_CLASS(Robot);