robot-config.h

#include <string>

vex::brain Brain;

using namespace std;

using namespace vex;

// main.cpp functions

void drive(int);

// General Functions

motor assignMotor(int);

controller assignControler();

int getAxisPosition(int);

void printError();

void printToScreen(const char\*);

void pauseTask(int); // time in milliseconds between 100 and 5000

// Controller Functions

// Button X used to stop controller daemon

// ProgramStop() uses OnOff bool switch

void ControllerDriveForward(int);

void ControllerDriveBackward(int);

void ControllerTurnLeft(int);

void ControllerTurnRight(int);

void ControllerMoveArmUp();

void ControllerMoveArmDown();

void ControllerClawOpen();

void ControllerClawClose();

void ProgramStop();

// Autonomous Functions List

// All two parameter functions take the form of Function(speed, time)

// All single parameter functions take the form of Function(time)

void AutoDriveForward(int, int);

void AutoDriveBackward(int, int);

void AutoDriveTurnLeft(int, int);

void AutoDriveTurnRight(int, int);

void AutoMoveArmUp(int);

void AutoMoveArmDown(int);

void AutoClawOpen(int);

void AutoClawClosed(int);

// Sensor Functions

// Declared and Defined Variables

motor DriveMotorLeft = assignMotor(1);

motor DriveMotorRight = assignMotor(2);

motor ArmMotor = assignMotor(3);

motor ClawMotor = assignMotor(4);

controller ConOne = assignControler();

bool OnOff = true;

/\*

\*\*\* RAW CODE BELOW \*\*\*

Do Not Change or Modify

\*/

// main.ccp Functions

void drive(int speed) {

if(speed < 15 || speed > 100) {

printError();

} else if(ConOne.ButtonUp.pressing()) {

ControllerDriveForward(speed);

} else if(ConOne.ButtonDown.pressing()) {

ControllerDriveBackward(speed);

} else if(ConOne.ButtonLeft.pressing()) {

ControllerTurnLeft(speed);

} else if(ConOne.ButtonRight.pressing()) {

ControllerTurnRight(speed);

} else {

DriveMotorLeft.stop();

DriveMotorRight.stop();

}

}

// End main.ccp Functions

// General Functions

void printError() {

Brain.Screen.print("Error: values input outside permitted limits!");

}

void pauseTask(int time) {

if(time < 100 || time > 5000) {

printError();

} else {

task::sleep(time);

}

}

int getAxisPosition(int motorPort) {

int posVal = 0;

if(motorPort == 3) {

posVal = ConOne.Axis3.value();

}else if(motorPort == 4) {

posVal = ConOne.Axis2.value();

}

return posVal;

}

motor assignMotor(int portNumber) {

return motor(portNumber - 1);

}

controller assignControler() {

return controller();

}

void printToScreen(const char\* input) {

Brain.Screen.clearScreen();

Brain.Screen.setCursor(1, 1);

Brain.Screen.print(input);

}

void ProgramStop() {

if(ConOne.ButtonX.pressing()) {

OnOff = false;

}

}

// End General Functions

// CONTROLER SECTION

void ControllerDriveForward(int speed) {

if(speed < 15 || speed > 100) {

printError();

} else if(ConOne.ButtonUp.pressing()) {

DriveMotorLeft.spin(directionType::fwd, speed, velocityUnits::dps);

DriveMotorRight.spin(directionType::rev, speed, velocityUnits::dps);

} else if(ConOne.ButtonDown.pressing()) {

} else if(ConOne.ButtonLeft.pressing()) {

} else if(ConOne.ButtonRight.pressing()) {

} else {

DriveMotorLeft.stop();

DriveMotorRight.stop();

}

}

void ControllerDriveBackward(int speed){

if(speed < 15 || speed > 100) {

printError();

} else if(ConOne.ButtonUp.pressing()) {

} else if(ConOne.ButtonDown.pressing()) {

DriveMotorLeft.spin(directionType::rev, speed, velocityUnits::dps);

DriveMotorRight.spin(directionType::fwd, speed, velocityUnits::dps);

} else if(ConOne.ButtonLeft.pressing()) {

} else if(ConOne.ButtonRight.pressing()) {

} else {

DriveMotorLeft.stop();

DriveMotorRight.stop();

}

}

void ControllerTurnLeft(int speed) {

if(speed < 15 || speed > 100) {

printError();

} else if(ConOne.ButtonUp.pressing()) {

} else if(ConOne.ButtonDown.pressing()) {

} else if(ConOne.ButtonLeft.pressing()) {

DriveMotorLeft.spin(directionType::rev, speed, velocityUnits::dps);

DriveMotorRight.spin(directionType::rev, speed, velocityUnits::dps);

} else if(ConOne.ButtonRight.pressing()) {

} else {

DriveMotorLeft.stop();

DriveMotorRight.stop();

}

}

void ControllerTurnRight(int speed) {

if(speed < 15 || speed > 100) {

printError();

} else if(ConOne.ButtonUp.pressing()) {

} else if(ConOne.ButtonDown.pressing()) {

} else if(ConOne.ButtonLeft.pressing()) {

} else if(ConOne.ButtonRight.pressing()) {

DriveMotorLeft.spin(directionType::fwd, speed, velocityUnits::dps);

DriveMotorRight.spin(directionType::fwd, speed, velocityUnits::dps);

} else {

DriveMotorLeft.stop();

DriveMotorRight.stop();

}

}

void ControllerMoveArmUp() {

if(getAxisPosition(3) > 0) {

ArmMotor.startRotateTo(30, rotationUnits::deg, 15, velocityUnits::dps);

} else if(getAxisPosition(3) < 0) {

} else {

ArmMotor.stop();

}

}

void ControllerMoveArmDown() {

if(getAxisPosition(3) > 0) {

} else if(getAxisPosition(3) < 0) {

ArmMotor.startRotateTo(-30, rotationUnits::deg, 15, velocityUnits::dps);

} else {

ArmMotor.stop();

}

}

void ControllerClawOpen() {

if(getAxisPosition(4) > 0) {

ClawMotor.startRotateTo(30, rotationUnits::deg, 15, velocityUnits::dps);

} else if(getAxisPosition(4) < 0) {

} else {

ClawMotor.stop();

}

}

void ControllerClawClose() {

if(getAxisPosition(4) > 0) {

} else if(getAxisPosition(4) < 0) {

ClawMotor.startRotateTo(-30, rotationUnits::deg, 15, velocityUnits::dps);

} else {

ClawMotor.stop();

}

}

// END CONTROLER SECTION

// AUTONOMOUS SECTION

// Speed 15-100 dps Time 500-5000

void AutoDriveForward(int speed, int time) {

if(speed < 15 || speed > 100) {

printError();

} else if(time < 500 || time > 5000) {

printError();

} else {

DriveMotorLeft.spin(directionType::fwd, speed, velocityUnits::dps);

DriveMotorRight.spin(directionType::rev, speed, velocityUnits::dps);

task::sleep(time);

DriveMotorLeft.stop();

DriveMotorRight.stop();

task::sleep(700);

}

}

void AutoDriveBackward(int speed, int time) {

if(speed < 15 || speed > 100) {

printError();

} else if(time < 500 || time > 5000) {

printError();

} else {

DriveMotorLeft.spin(directionType::rev, speed, velocityUnits::dps);

DriveMotorRight.spin(directionType::fwd, speed, velocityUnits::dps);

task::sleep(time);

DriveMotorLeft.stop();

DriveMotorRight.stop();

task::sleep(700);

}

}

void AutoDriveTurnLeft(int speed, int time) {

if(speed < 15 || speed > 100) {

printError();

} else if(time < 500 || time > 5000) {

printError();

} else {

DriveMotorLeft.spin(directionType::rev, speed, velocityUnits::dps);

DriveMotorRight.spin(directionType::rev, speed, velocityUnits::dps);

task::sleep(time);

DriveMotorLeft.stop();

DriveMotorRight.stop();

task::sleep(700);

}

}

void AutoDriveTurnRight(int speed, int time) {

if(speed < 15 || speed > 100) {

printError();

} else if(time < 500 || time > 5000) {

printError();

} else {

DriveMotorLeft.spin(directionType::fwd, speed, velocityUnits::dps);

DriveMotorRight.spin(directionType::fwd, speed, velocityUnits::dps);

task::sleep(time);

DriveMotorLeft.stop();

DriveMotorRight.stop();

task::sleep(700);

}

}

void AutoMoveArmUp(int time) {

if(time < 500 || time > 5000) {

printError();

} else {

ArmMotor.startRotateTo(30, rotationUnits::deg, 15, velocityUnits::dps);

task::sleep(time);

ArmMotor.stop();

task::sleep(700);

}

}

void AutoMoveArmDown(int time) {

if(time < 500 || time > 5000) {

printError();

} else {

ArmMotor.startRotateTo(-30, rotationUnits::deg, 15, velocityUnits::dps);

task::sleep(time);

ArmMotor.stop();

task::sleep(700);

}

}

void AutoClawOpen(int time) {

if(time < 500 || time > 5000) {

printError();

} else {

ClawMotor.startRotateTo(30, rotationUnits::deg, 15, velocityUnits::dps);

task::sleep(time);

ArmMotor.stop();

task::sleep(700);

}

}

void AutoClawClose(int time) {

if(time < 500 || time > 5000) {

printError();

} else {

ClawMotor.startRotateTo(-30, rotationUnits::deg, 15, velocityUnits::dps);

task::sleep(time);

ArmMotor.stop();

task::sleep(700);

}

}

// END AUTONOMOUS SECTION