#include <FEHLCD.h>

#include <FEHIO.h>

#include <FEHUtility.h>

#include <FEHServo.h>

#include <FEHMotor.h>

#include <FEHRPS.h>

void initializeRPS();

void slowLeft( float time);

void straight(float time);

void checkHeading(int heading);

void start();

void leverSwitch();

void stopMotors();

void back(float time);

void left(float time);

void right(float time);

void read();

int lightTest();

void slowStraight(float time);

void slowRight(float time);

void backLeft(float time);

void backRight(float time);

//going straight

void straight2(float percent, float time);

void straightShaft( float percent, int counts);

//get to foosball station

void toFoosball();

//set up foosball arm

void setFoosballArm();

//move with foosball

void moveFoosball();

//remove arm

void removeFoosballArm();

//turn for a certain time and adjust for heading

void right(float time, float heading);

void left(float time, float heading);

//front motor in motor port 0

FEHMotor rightMotor(FEHMotor::Motor0, 7.2);

//back motor in motor port 1

FEHMotor leftMotor(FEHMotor::Motor1, 7.2);

//arm servo in servo port 0

FEHServo armServo(FEHServo::Servo7);

//cds cell in pin P1\_7

AnalogInputPin cdsCell(FEHIO::P1\_7);

//sets servo for arm horizontal rotation

FEHServo rotServo(FEHServo::Servo2);

//sets microservo for arm

FEHServo microServo(FEHServo::Servo5);//Need to find min and max for microServo

//set bump switches for front and back of bot and on arm

DigitalInputPin rightBump(FEHIO::P0\_0);

DigitalInputPin leftBump(FEHIO::P0\_6);

DigitalInputPin frontBump(FEHIO::P0\_3);

DigitalInputPin backLeftBump(FEHIO::P0\_7);

DigitalInputPin backRightBump(FEHIO::P0\_2);

//declare encoders

DigitalEncoder right\_encoder(FEHIO::P1\_2);

DigitalEncoder left\_encoder(FEHIO::P1\_0);

//Set inputs for line readers

AnalogInputPin rightOpto(FEHIO::P2\_0);

AnalogInputPin midOpto(FEHIO::P2\_1);

AnalogInputPin leftOpto(FEHIO::P2\_3);

//declare encoders

int main(void)

{

LCD.Clear(BLUE);

armServo.SetMin(800);

armServo.SetMax(2200);

rotServo.SetMin(700);

rotServo.SetMax(2350);

microServo.SetMin(1470);

microServo.SetMax(2270);

armServo.SetDegree(180);

microServo.SetDegree(120);

rotServo.SetDegree(85);

left\_encoder.ResetCounts();

right\_encoder.ResetCounts();

LCD.Clear(RED);

initializeRPS();

//check to see if light is on and start moving once detected

start();

float time = 0.3;

//move to the ddr first light with cds cell centered on the first light

straight(time);

time=0.6;

right(time);

checkHeading(0);

time=0.43;

straight(time);

//read light color and press correct button

int i=0;

int test=0;

//read(cdsCell);

time = 1.0;

Sleep(time);

lightTest();

//have light test to return int that can be used to determine the color

//if (test = 2){ // run if loop to bring bot back if test is blue

//time = 0.2;

//back(time);

//backRight(time);

//time =1.2;

//back(time);

//stopMotors();

//}

//if (test = 1){ //run if loop to bring bot back if test is red

//time = 0.2;

//back(time);

//backRight(time);

//stopMotors();

//}

backLeft(0.2);

checkHeading(0);

toFoosball();

//setFoosballArm();

//moveFoosball();

}

//function to stop motors

void stopMotors(){

rightMotor.Stop();

leftMotor.Stop();

}

//function to go straight for time

void straight( float time)

{

float rightMotorPercent=70.0;

float leftMotorPercent=-70.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

//function to detect start light

void start()

{

//declare maximum value for light being off

float thresholdValue=1.0;

//keep checking to see if start light is on

bool check=true;

while(check)

{

//check to see if light is turned on

if(cdsCell.Value()<thresholdValue)

{

check=false;

}

}

}

void leverSwitch()

{

float leverAngle = 180.00;

float postLever = 2.00;

armServo.SetDegree(leverAngle);

Sleep(postLever);

}

void back( float time)

{

float rightMotorPercent=-70.0;

float leftMotorPercent=67.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

void left( float time){

float rightMotorPercent=40.0;

float leftMotorPercent=0.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

void read(){

LCD.SetFontColor(WHITE);

while(true){

LCD.Clear(BLACK);

LCD.WriteLine(cdsCell.Value());

Sleep(50);

}

}

void right(float time){

float rightMotorPercent=0.0;

float leftMotorPercent=-60.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

int lightTest()

{

int test =0;

float time = 1.0;

if(cdsCell.Value()>0.63) //Testing for Blue value and will move straight before turning right into button

{

LCD.Clear(BLUE);

time = 1.35;

slowStraight(time);

time = 1.5;

slowRight(time);

stopMotors();

Sleep(5.0);

test = 2;

}else //Testing for red value and will turn right directly into button

{

LCD.Clear(RED);

time = 2.0;

slowRight(time);

time = 6.0;

//slowStraight(rightMotor,leftMotor,time);

stopMotors();

Sleep(5.0);

test = 1;

}

return test;

}

void slowStraight( float time)

{

float rightMotorPercent=20.0;

float leftMotorPercent=-20.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

void slowRight( float time)

{

float rightMotorPercent=5.0;

float leftMotorPercent=-30.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

void backRight(float time){

float rightMotorPercent=-60.0;

float leftMotorPercent= 0.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

void backLeft( float time){

float rightMotorPercent=0.0;

float leftMotorPercent=60.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

//function to check heading

void checkHeading(int heading) //using RPS

{

float startTime = TimeNow();

if(heading!=0){

while((RPS.Heading()<(heading-3) || RPS.Heading()>(heading+3)) && TimeNow() - startTime < 3.0){

if(RPS.Heading() < heading){

leftMotor.SetPercent(20);

leftMotor.SetPercent(20);

Sleep(20);

stopMotors();

Sleep(200);

}

else if(RPS.Heading() > heading){

rightMotor.SetPercent(-20);

leftMotor.SetPercent(-20);

Sleep(20);

stopMotors();

Sleep(200);

}

}

}

else if(heading==0){

while((RPS.Heading()<357 || RPS.Heading()>3) && TimeNow() - startTime < 3.0)

{

if(RPS.Heading() < 360 && RPS.Heading()>180){

rightMotor.SetPercent(20);

leftMotor.SetPercent(20);

Sleep(20);

stopMotors();

Sleep(200);

}

else if(RPS.Heading() > heading && RPS.Heading() <180){

rightMotor.SetPercent(-20);

leftMotor.SetPercent(-20);

Sleep(20);

stopMotors();

Sleep(200);

}

}

}

stopMotors();

}

void initializeRPS(){

//Initialize RPS and declare any variables you might need

float touch\_x, touch\_y;

//Call this function to initialize the RPS to a course

RPS.InitializeTouchMenu();

LCD.Clear();

}

void right(float time, float heading){

int motorPower=40;

leftMotor.SetPercent(-motorPower);

rightMotor.SetPercent(-motorPower);

Sleep(time);

stopMotors();

if(heading>=0){

checkHeading(heading);

}

}

void left(float time, float heading){

int motorPower=40;

leftMotor.SetPercent(motorPower);

rightMotor.SetPercent(motorPower);

Sleep(time);

stopMotors();

if(heading>=0){

checkHeading(heading);

}

}

void straightShaft( float percent, int counts)

{

LCD.WriteLine("in straight function");

rightMotor.SetPercent(percent);

leftMotor.SetPercent(-percent);

//Reset encoder counts

right\_encoder.ResetCounts();

left\_encoder.ResetCounts();

//While the average of the left and right encoder is less than counts,

//keep running motors

while((left\_encoder.Counts() + right\_encoder.Counts()) / 2. < counts);

stopMotors();

}

void straight2(float percent, float time){

rightMotor.SetPercent(percent);

leftMotor.SetPercent(-percent\*1.1);

if(time>0){

Sleep(time);

stopMotors();

}

}

//function to move to foosball

void toFoosball()

{

float movementSpeed = 50;

float heading;

float time;

float ninetyDegreeTime=1.4;

int counts;

//go straight until left and right bumpers are pressed

straight2(movementSpeed, -1);

while(leftBump.Value() || rightBump.Value());

stopMotors();

//move back and then turn left

time=0.5;

//go back 1 inches

counts=5;

straightShaft(-movementSpeed, counts);

left(time, -1);

//go forward until right bumper is pressed, go back a little, and then turn left

straight2(movementSpeed, -1);

while(rightBump.Value());

stopMotors();

//go back 0.3 inches

counts=2;

straightShaft(-movementSpeed, counts);

//turn left to 90 degrees

heading=90;

time=0.3;

left(time, heading);

//go straight until any bumper is pressed

straight2(movementSpeed, -1);

while((leftBump.Value() && rightBump.Value()) && frontBump.Value());

stopMotors();

Sleep(2.0);

// move back and then turn left

// go back 1 inches

counts=60;

straightShaft(-movementSpeed, counts);

left(ninetyDegreeTime);

// go straight back until back left and right bumpers are pressed

straight2(-movementSpeed, 1.0);

stopMotors();

// go forward one inch

counts=6;

straightShaft(movementSpeed, counts);

}

void setFoosballArm()

{

//variables for final angles

float finalArmAngle=20;

float finalMicro=80;

int armAngle=180;

int microAngle=120;

int rotAngle=85;

armServo.SetDegree(armAngle);

microServo.SetDegree(microAngle);

rotServo.SetDegree(rotAngle);

//lower arm servo slowly

while(armAngle>finalArmAngle)

{

armAngle=armAngle-2;

armServo.SetDegree(armAngle);

//lower slowly by sleeping inbetween

Sleep(20);

}

//clamp on foosball using microservo

microServo.SetDegree(finalMicro);

}

void moveFoosball()

{

float movementSpeed = 10;

int counts = 72;

straightShaft(movementSpeed, counts);

}

void removeFoosballArm()

{

//variables for final angles

float finalArmAngle=90;

float finalMicroAngle= 120;

float finalRotAngle=0;

microServo.SetDegree(finalMicroAngle);

armServo.SetDegree(finalArmAngle);

rotServo.SetDegree(finalRotAngle);

}

//function to navigate to and press final button

//void foosballToFinal()

//{

//}