#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);

void leftCounts(float movementSpeed, int counts);

void rightCounts(float percent, int counts);

bool foosHitSwitch();

void toButton2();

//get to foosball station

void toFoosball();

//set up foosball arm

void setFoosballArm();

//move with foosball

void moveFoosball();

//remove arm

void removeFoosballArm();

//toLever

void toLever();

void toButton();

//turn for a certain time and adjust for heading

void right(float time, float heading);

void left(float time, float heading);

//go to coin

void toCoin(float time);

//line following up to coin

void lineFollower(float color,float time);

//drop coin into clot

void dropCoin();

void slowLeft( float time);

void check\_x\_plus(float x\_coordinate);

void check\_y\_plus(float y\_coordinate);

//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::Servo0);

//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::Servo4);//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\_1);

DigitalInputPin topArmBump(FEHIO::P3\_0);

//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 line following

#define LINE\_ON\_RIGHT 0

#define ON\_LINE 1

#define LINE\_ON\_LEFT 2

int main(void)

{

//initialize all servo values and RPS

LCD.Clear();

armServo.SetMin(800);

armServo.SetMax(2200);

rotServo.SetMin(700);

rotServo.SetMax(2350);

// microServo.SetMin(1470);

// microServo.SetMax(2270);

microServo.SetMin(564);

microServo.SetMax(1270);

armServo.SetDegree(0);

microServo.SetDegree(85);

rotServo.SetDegree(94);// this is the starting angle - may need to be adjusted

left\_encoder.ResetCounts();

right\_encoder.ResetCounts();

int ninetyDegreeCounts=67;

/\* int x = 0;

while(x == 0){

LCD.Clear();

LCD.WriteLine(rightOpto.Value());

LCD.WriteLine(midOpto.Value());

LCD.WriteLine(leftOpto.Value());

Sleep(2.0);

}\*/

initializeRPS();

//start searching for light on touch

LCD.Clear();

LCD.WriteLine("Touch the screen as your final action");

float x\_position, y\_position;

/\* Wait until the user touches the screen \*/

while(!LCD.Touch(&x\_position,&y\_position)) {};

/\* Wait until the user releases the screen \*/

while(LCD.Touch(&x\_position,&y\_position)) {};

Sleep(0.2);

LCD.Clear();

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

start();

//movement to coin

float time = 0.7;

straight(time);

time = 0.8;

slowLeft(time);

//function that uses line following to line itself up with the front of the coin slot

toCoin(time);

//actually dropping coin

////uses microservo to drop coin into slot-may have arm servo rotate as well

dropCoin();

//move from coin to DDR

time = 0.1;

backRight(time);

//move back two inches and check heading

int counts=12;

float movementSpeed=40;

straightShaft(-movementSpeed, counts);

float heading=90;

checkHeading(heading);

movementSpeed=60;

straight2(-movementSpeed, -1);

//have a timer just in case both bumpswitches don’t get activated

float timer=TimeNow();

while((backLeftBump.Value()|| backRightBump.Value()) && (TimeNow()-timer<2.0));

stopMotors();

counts=38;

movementSpeed=40;

//move forward a little less than 6 inches

straightShaft(movementSpeed, counts);

//check y rps coordinate

// float yCoordinate=9.0;

// check\_y\_plus(yCoordinate);

//turn right 90 degrees

rightCounts(movementSpeed, ninetyDegreeCounts);

checkHeading(0);

//x coordinate to line up with DDR

float xCoordinate=18.6;

check\_x\_plus(xCoordinate);

//actual DDR run

int i=0;

int test=0;

time = 0.5;

Sleep(time);

lightTest();

backRight(0.2);

//movement from DDR to line up with foosball

toFoosball();

//set arm

setFoosballArm();

//doing foosball

moveFoosball();

removeFoosballArm();

//movement to lever

toLever();

//hitting the lever

leverSwitch();

toButton();

}

//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 and have a timer for 30 seconds in case the robot doesn’t detect the start light

bool check=true;

float timer=TimeNow();

while(check)

{

//check to see if light is turned on

if(cdsCell.Value()<thresholdValue || TimeNow()-timer>30.0)

{

check=false;

}

}

}

void back( float time)

{

float rightMotorPercent=-70.0;

float leftMotorPercent=70.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

if(time>0)

{

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());

LCD.WriteLine(leftOpto.Value());

LCD.WriteLine(midOpto.Value());

LCD.WriteLine(rightOpto.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 = 1.4;

slowStraight(time);

time = 1.5;

slowRight(time);

leftMotor.SetPercent(-5.0);

Sleep(4.5);

stopMotors();

test = 2;

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

{

LCD.Clear(RED);

time = 0.1;

slowStraight(time);

time = 2.0;

slowRight(time);

leftMotor.SetPercent(-5.0);

Sleep(4.5);

stopMotors();

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 slowLeft( float time)

{

float rightMotorPercent=30.0;

float leftMotorPercent=-5.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

void backRight(float time){

float rightMotorPercent= 0.0;

float leftMotorPercent= 60.0;

rightMotor.SetPercent(rightMotorPercent);

leftMotor.SetPercent(leftMotorPercent);

Sleep(time);

stopMotors();

}

void backLeft( float time){

float rightMotorPercent=-60.0;

float leftMotorPercent=0.0;

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 < 4.0){

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)

{

rightMotor.SetPercent(percent);

leftMotor.SetPercent(-1.1\*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 straightShaft2(float percent, int counts)

{

rightMotor.SetPercent(percent);

leftMotor.SetPercent(-1.05\*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 leftCounts(float percent, int counts)

{

right\_encoder.ResetCounts();

left\_encoder.ResetCounts();

rightMotor.SetPercent(percent);

while(right\_encoder.Counts()<counts);

stopMotors();

}

void rightCounts(float percent, int counts)

{

right\_encoder.ResetCounts();

left\_encoder.ResetCounts();

leftMotor.SetPercent(-percent);

while(left\_encoder.Counts()<counts);

stopMotors();

}

void straight2(float percent, float time){

rightMotor.SetPercent(percent);

leftMotor.SetPercent(-1.05\*percent);

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

checkHeading(0);

straight2(movementSpeed, -1);

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

stopMotors();

//move back and then turn left

time=0.5;

//go back 0.7 inches

counts=4;

straightShaft(-movementSpeed, counts);

left(time, -1);

//hard coded to go up the ramp and check for being stuck using time

float bumpTimer=TimeNow();

rightMotor.SetPercent(movementSpeed);

leftMotor.SetPercent(-1.07\*movementSpeed);

//make sure right bumpswitch doesnt activate before the ramp

time = TimeNow();

while(TimeNow()-time<2.5);

stopMotors();

checkHeading(90.00);

//speed up

movementSpeed=60;

straight2(movementSpeed, -1);

//check for bumper

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

stopMotors();

Sleep(0.2);

//slow down

movementSpeed=50;

//drops back three and a half inches before turning left and going back into the wall

counts=21;

straightShaft(-movementSpeed, counts);

//turn left 90 degrees

counts=67; //counts for 90 degree turn

leftCounts(movementSpeed, counts);

straight2(-movementSpeed, -1);

//square up robot with right wall

while((backLeftBump.Value() && backRightBump.Value()));

stopMotors();

time = TimeNow();

if (!backLeftBump.Value() && backRightBump.Value()) {

rightMotor.SetPercent(-30);

leftMotor.SetPercent(8);

while(backLeftBump.Value() || backRightBump.Value());

} else if (backLeftBump.Value() && !backRightBump.Value()) {

leftMotor.SetPercent(30);

rightMotor.SetPercent(-8);

while((backLeftBump.Value() || backRightBump.Value())&&TimeNow()-time<1.0);

}

stopMotors();

Sleep(0.2);

//go forward and turn right a little bit, go, straight, then left a little bit to be closer to foosball.

counts=6;

movementSpeed=25.0;

straightShaft(movementSpeed, counts);

counts=25;

rightCounts(movementSpeed, counts);

//hard coded straight shaft to check for stuck robot

time=TimeNow();

counts=16;

rightMotor.SetPercent(movementSpeed);

leftMotor.SetPercent(-1.05\*movementSpeed);

//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 && TimeNow()-time<3.0);

stopMotors();

//if timed out, go back an inch and a half

if(TimeNow()-time>2.9)

{

counts=9;

straightShaft(-movementSpeed, counts);

}

counts=25;

leftCounts(movementSpeed, counts);

//back up to wall again

straight2(-movementSpeed, -1);

//square up robot with right wall

while((backLeftBump.Value() || backRightBump.Value()));

stopMotors();

}

void setFoosballArm()

{

//variables for final angles

float finalArmAngle=163;

float finalMicroAngle=37;

int armAngle=90;

int microAngle=123;

int microAngle2=80;

int rotAngle=10;

microServo.SetDegree(microAngle);

Sleep(0.2);

armServo.SetDegree(armAngle);

Sleep(0.2);

rotServo.SetDegree(rotAngle);

Sleep(0.2);

//lower arm servo slowly

while(armAngle<finalArmAngle)

{

armAngle=armAngle+2;

armServo.SetDegree(armAngle);

if(armAngle==136)

{

microServo.SetDegree(microAngle2);

}

//lower slowly by sleeping inbetween

Sleep(10);

}

while(microAngle2>finalMicroAngle)

{

microAngle2=microAngle2-2;

microServo.SetDegree(microAngle2);

//lower slowly by sleeping inbetween

Sleep(10);

}

rotServo.SetDegree(18);

}

void moveFoosball()

{

// armServo.SetDegree(90);

// Sleep(100);

// rotServo.SetDegree(10);

// Sleep(100);

// armServo.SetDegree(160);

float movementSpeed = 20;

int counts = 50;

rightMotor.SetPercent(movementSpeed);

leftMotor.SetPercent(-1.15\*movementSpeed);

//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();

int armAngle=60;

int microAngle=120;

int rotAngle=10;

microServo.SetDegree(microAngle);

Sleep(0.2);

armServo.SetDegree(armAngle);

Sleep(0.2);

rotServo.SetDegree(rotAngle);

}

void removeFoosballArm()

{

//variables for final angles

float finalArmAngle=60;

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()

//{

//}

void toCoin(float time)

{

time = 0.05;

double Lv=leftOpto.Value();

double Mv=midOpto.Value();

double Rv=rightOpto.Value();

//got yellow value from earlier exploration, may need to be retaken

float yellow = 3.15;

int check = 0;

while (check ==0)

{

if(leftOpto.Value()>yellow&&midOpto.Value()>yellow&&rightOpto.Value()>yellow){

slowStraight(time);

}else{

LCD.Clear(YELLOW);

check++;

}

}

time = 1.0;

lineFollower(yellow,time);

Sleep(time);

stopMotors();

}

void lineFollower(float color, float time)

{

int state = ON\_LINE;

double Lv=leftOpto.Value();

double Mv=midOpto.Value();

double Rv=rightOpto.Value();

float t\_now;

t\_now=TimeNow();

while(frontBump.Value()&&TimeNow()-t\_now<4.0){

Lv=leftOpto.Value();

Mv=midOpto.Value();

Rv=rightOpto.Value();

if(Lv<color){

LCD.Clear(GREEN);

state = LINE\_ON\_LEFT;

}

if(Mv<color){

LCD.Clear(WHITE);

state = ON\_LINE;

}

if(Rv<color){

LCD.Clear(RED);

state = LINE\_ON\_RIGHT;

}

switch(state) {

// If the line is on my right...

case LINE\_ON\_RIGHT:

leftMotor.SetPercent(-20.0);

rightMotor.SetPercent(0.0);

break;

// If I am on the line

case ON\_LINE:

leftMotor.SetPercent(-20.0);

rightMotor.SetPercent(20.0);

break;

// If the line is on my left...

case LINE\_ON\_LEFT:

leftMotor.SetPercent(0.0);

rightMotor.SetPercent(20.0);

break;

}

}

LCD.Clear();

stopMotors();

straightShaft(-20,3);

checkHeading(90.00);

straight2(20,-1);

t\_now=TimeNow();

while(frontBump.Value()&&TimeNow()-t\_now<1.0){

}

stopMotors();

}

void dropCoin()

{

int count=5;

int m = 85;

float leverAngle = 83.0;//need to test this value, Completely guessing here

float postLever = 2.00;

float k = 0.0;

while (k < leverAngle) {

armServo.SetDegree(k);

Sleep(3);

k += 1.1;

if (k>75&&count==5){

count = 0;

m=m-3;

microServo.SetDegree(m);

}else{

count++;

}

}

Sleep(postLever);

armServo.SetDegree(0);

}

void check\_x\_plus(float x\_coordinate) //using RPS while robot is in the +x direction

{

//check whether the robot is within an acceptable range

while(RPS.X() < x\_coordinate - 0.2 || RPS.X() > x\_coordinate + 0.2)

{

if(RPS.X() > x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

rightMotor.SetPercent(-15);

leftMotor.SetPercent(15);

Sleep(80);

rightMotor.Stop();

leftMotor.Stop();

Sleep(100);

}

else if(RPS.X() < x\_coordinate)

{

//pulse the motors for a short duration in the correct direction

rightMotor.SetPercent(15);

leftMotor.SetPercent(-15);

Sleep(80);

rightMotor.Stop();

leftMotor.Stop();

Sleep(100);

}

}

stopMotors();

}

void check\_y\_plus(float y\_coordinate) //using RPS while robot is in the +y direction

{

//check whether the robot is within an acceptable range

while(RPS.Y() < y\_coordinate - 0.2 || RPS.Y() > y\_coordinate + 0.2)

{

if(RPS.Y() > y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

rightMotor.SetPercent(-15);

leftMotor.SetPercent(15);

Sleep(80);

rightMotor.Stop();

leftMotor.Stop();

Sleep(100);

}

else if(RPS.Y() < y\_coordinate)

{

//pulse the motors for a short duration in the correct direction

rightMotor.SetPercent(15);

leftMotor.SetPercent(-15);

Sleep(80);

rightMotor.Stop();

leftMotor.Stop();

Sleep(100);

}

}

stopMotors();

}

void toLever()

{

float movementSpeed=60;

float time;

straight2(-movementSpeed, -1);

//turn left, go straight, turn right, and go to square up with left wall

while((backLeftBump.Value() && backRightBump.Value()));

stopMotors();

time = TimeNow();

if (!backLeftBump.Value() && backRightBump.Value()) {

rightMotor.SetPercent(-30);

leftMotor.SetPercent(8);

while(backLeftBump.Value() || backRightBump.Value());

} else if (backLeftBump.Value() && !backRightBump.Value()) {

leftMotor.SetPercent(30);

rightMotor.SetPercent(-8);

while((backLeftBump.Value() || backRightBump.Value())&&TimeNow()-time<0.7);

}

stopMotors();

int counts=32;

leftCounts(movementSpeed, counts);

counts=96;

straightShaft2(movementSpeed, counts);

counts=34;

rightCounts(movementSpeed, counts);

straight2(movementSpeed, -1);

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

stopMotors();

counts=67;

leftCounts(-movementSpeed, counts);

float yCheck=43;

checkHeading(90);

check\_y\_plus(yCheck);

counts=27;

leftCounts(movementSpeed, counts);

straightShaft2(movementSpeed, 8);

rightCounts(movementSpeed, counts);

counts=36;

straightShaft2(movementSpeed, counts);

}

void leverSwitch()

{

rotServo.SetDegree(90);

microServo.SetDegree(35);

float leverAngle = 155;//need to test this value, Completely guessing here

float postLever = 0.3;

float k = 90.0;

while (k < leverAngle) {

armServo.SetDegree(k);

Sleep(7);

k += 1;

}

Sleep(postLever);

}

void toButton()

{

int movementSpeed=50;

int counts=60;

rightMotor.SetPercent(1.14\*-movementSpeed);

leftMotor.SetPercent(movementSpeed);

float time=TimeNow();

//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 && TimeNow()-time<4.0);

stopMotors();

armServo.SetDegree(0);

rotServo.SetDegree(93);

checkHeading(90);

straight2(-movementSpeed, -1);

}