#pragma config(Hubs, S1, HTMotor, HTMotor, HTMotor, HTMotor)

#pragma config(Hubs, S2, HTServo, none, none, none)

#pragma config(Sensor, S3, SMUX, sensorI2CCustom9V)

#pragma config(Sensor, S4, Gyro, sensorI2CHiTechnicGyro)

#pragma config(Motor, mtr\_S1\_C1\_1, backRight, tmotorTetrix, openLoop)

#pragma config(Motor, mtr\_S1\_C1\_2, backLeft, tmotorTetrix, openLoop)

#pragma config(Motor, mtr\_S1\_C2\_1, emptyMotor, tmotorTetrix, openLoop)

#pragma config(Motor, mtr\_S1\_C2\_2, collector, tmotorTetrix, openLoop)

#pragma config(Motor, mtr\_S1\_C3\_1, frontRight, tmotorTetrix, openLoop)

#pragma config(Motor, mtr\_S1\_C3\_2, frontLeft, tmotorTetrix, openLoop)

#pragma config(Motor, mtr\_S1\_C4\_1, liftRight, tmotorTetrix, openLoop)

#pragma config(Motor, mtr\_S1\_C4\_2, liftLeft, tmotorTetrix, openLoop)

#pragma config(Servo, srvo\_S2\_C1\_1, rollingSide, tServoStandard)

#pragma config(Servo, srvo\_S2\_C1\_2, rollingBack1, tServoStandard)

#pragma config(Servo, srvo\_S2\_C1\_3, scoreRight, tServoStandard)

#pragma config(Servo, srvo\_S2\_C1\_4, scoreLeft, BadType)

#pragma config(Servo, srvo\_S2\_C1\_5, rollingBack2, tServoStandard)

#pragma config(Servo, srvo\_S2\_C1\_6, autonomousArm, tServoStandard)

//\*!!Code automatically generated by 'ROBOTC' configuration wizard !!\*//

#include "JoystickDriver.c"

#include "hitechnic-gyro.h"

#include "hitechnic-sensormux.h"

#include "lego-ultrasound.h"

#include "hitechnic-irseeker-v2.h"

const tMUXSensor sonarOne = msensor\_S3\_1;

const tMUXSensor sonarTwo = msensor\_S3\_2;

float initial; //initial gyro reading

float heading; //robots current yaw

float radheading; //heading in radians

int lastTime = 0; //last time header was updated, used to find average change in rotation

float FLset; //front-left set, refers to Front-Left and Back-Right

float FRset; //front-right set, refers to Front-Right and Back-Left motor

float joyAngle; // angle of the first joystick

float currHeading;

float delTime = 0;//calibration

float prevHeading = 0;

float curRate = 0;

int deadZone = 15;

float calibrate = 0;

float sonarValue;

float sonarValue2;

float goalWidth;

float lightVal;

void initializeRobot()

{

// accelStart(S4);//accel

initial = 0;

for(int i = 0; i < 100; i++){//Sensor

initial += SensorValue[S4];

wait10Msec(1);

}

initial = initial / 100;//Sensor

servo[scoreLeft]=35;

servo[scoreRight]=220;

servo[rollingBack1] = 0;

servo[rollingBack2] = 0;

servo[rollingSide] = 50; //rolling goal clamps are down

servo[autonomousArm] = 195;

return;

}

void motorTrigger(tMotor motorCall,int x,int y,int z)

{

motor[motorCall]= (y+x+z);

}

void stopMotors()

{

motor[backleft] = 0;

motor[frontright] = 0;

motor[backright] = 0;

motor[frontleft] = 0;

}

void move(int angle, int power, float time)

{

int x = power\*128/100\*cosDegrees(angle);

int y = power\*128/100\*sinDegrees(angle);

int z = 0;

// int accelValue = SensorValue(Accel); //set accelerometer value to accelValue

int gyroValue = SensorValue(Gyro); //set gyroscope value to gyroValue

nxtDisplayCenteredBigTextLine(5,"%d",gyroValue); //display gyroValue to NXT

//nxtDisplayCenteredBigTextLine(5,"%d",accelValue); //display accelValue to NXT

int trueX = (cosDegrees(currHeading+calibrate)\*x)-(sinDegrees(currHeading+calibrate)\*y); //sets trueX to rotated x value

int trueY = (sinDegrees(currHeading+calibrate)\*x)+(cosDegrees(currHeading+calibrate)\*y); //sets trueY to rotated y value

x = trueX;

y = trueY;

motorTrigger(frontRight,x,-y,-z);

motorTrigger(backRight,-x,-y,-z);

motorTrigger(frontLeft,x,y,-z);

motorTrigger(backLeft,-x,y,-z);

wait1Msec(time);

motorTrigger(frontRight,0,0,0);

motorTrigger(backRight,0,0,0);

motorTrigger(frontLeft,0,0,0);

motorTrigger(backLeft,0,0,0);

wait1Msec(100);

}

void rotate(int relativeAngle, int direction, int power)

{

int finalHeading = currHeading + direction\*relativeAngle;

if (finalHeading > 360) finalHeading -= 360;

else if(finalHeading < 0) finalHeading += 360;

while(abs(finalHeading - currHeading) > 1)

{

int x = 0;

int y = 0;

int z = power\*direction\*128/100;

motorTrigger(frontRight,x,-y,-z);

motorTrigger(backRight,-x,-y,-z);

motorTrigger(frontLeft,x,y,-z);

motorTrigger(backLeft,-x,y,-z);

}

motorTrigger(frontRight,0,0,0);

motorTrigger(backRight,0,0,0);

motorTrigger(frontLeft,0,0,0);

motorTrigger(backLeft,0,0,0);

wait1Msec(100);

}

void releaseSide(bool release)

{

if (release)servo[rollingSide] = 200; //release

else servo[rollingSide] = 50; //clamp

wait1Msec(100);

}

void releaseBack(bool release)

{

if (release)//release

{

servo[rollingBack1] = 150;

servo[rollingBack2] = 150;

}

else //clamp

{

servo[rollingBack1] = 0;

servo[rollingBack2] = 0;

}

wait1Msec(100);

}

void scoreAutonomous()

{

servo[autonomousArm] = 100;

wait1Msec(1000);

servo[autonomousArm] = 195;

wait1Msec(100);

}

void waitSeconds(float x){

wait1Msec(x\*1000);

}

void moveDirection(char a){

if(a = 'r'){

motor[frontleft] = -50;

motor[frontright] = -50;

motor[backleft] = 50;

motor[backright] = 50;

}

if(a = 'l'){

motor[frontleft] = 50;

motor[frontright] = 50;

motor[backleft] = -50;

motor[backright] = -50;

}

if(a = 'f'){

motor[frontleft] = 50;

motor[frontright] = -50;

motor[backleft] = 50;

motor[backright] = -50;

}

if(a = 'b'){

motor[frontleft] = -50;

motor[frontright] = 50;

motor[backleft] = -50;

motor[backright] = 50;

}

}

void moveToGoal(){

moveDirection('l');

while(sonarValue > 10){

wait1Msec(10);

}

moveDirection('f');

/\*

while(sonarValue2 < DISTANCE\_IN\_CM\_FROM\_BACKBOARD\_TO\_SENSOR2\_WHEN\_PAST\_RAMP){

wait1Msec(10);

}

\*/

moveDirection('l');

while(sonarValue > 10){

wait1Msec(10);

}

//turn 90 degrees clockwise

moveDirection('r');

while(sonarValue > goalWidth){

wait1Msec(10);

}

stopMotors();

//clamp\_side

//turn 90 degrees clockwise

moveDirection('b');

while(sonarValue2 > goalWidth){

wait1Msec(10);

}

stopMotors();

//clamp\_back

//come back

}

task getSensorValues()

{

sonarValue = USreadDist(sonarOne);

sonarValue2 = USreadDist(sonarTwo);

while(true)

{

sonarValue = USreadDist(sonarOne);

sonarValue2 = USreadDist(sonarTwo);

wait1Msec(5);

}

}

task getHeading() {

lightVal = SensorValue(Gyro);

//sonarValue2 = USreadDist(sonarTwo);

HTGYROstartCal(Gyro);

PlaySound(soundBeepBeep);

//sonarValue = USreadDist(sonarOne);

//sonarValue2 = USreadDist(sonarTwo);

while (true) {

// sonarValue = USreadDist(sonarOne);

// sonarValue2 = USreadDist(sonarTwo);

time1[T1] = 0;

curRate = HTGYROreadRot(Gyro);

if (abs(curRate) > 3) //sets deadzones for gyroscope

{

prevHeading = currHeading;

currHeading = prevHeading + curRate \* delTime; //changes current heading based on the rate of change and time elapsed

if (currHeading > 360) currHeading -= 360; //keeps current heading between 0 and 360

else if (currHeading < 0) currHeading += 360; // keeps curent heading between 0 and 360

}

wait1Msec(5);

delTime = ((float)time1[T1]) / 1000;

}

}

void moveRight(float s) {

motorTrigger(frontright, 0, 50, 0);

motorTrigger(backright, 0, -50, 0);

motorTrigger(frontleft, 0, 50, 0);

motorTrigger(backleft, 0, -50, 0);

wait1MSec(s\*1000);

}

void moveForward() {

PlaySound(soundBeepBeep);

while(sonarValue >= 100)

{

motorTrigger(frontright,0,-50,-0);

motorTrigger(backright,-0,-50,-0);

motorTrigger(frontleft,0,50,-0);

motorTrigger(backleft,-0,50,-0);

}

stopMotors();

}

task autoAlign()

{

while(true)

{

if(currHeading >= 3 && currHeading <180) // currHeading <= 355

{

motorTrigger(frontright,0,0,20);

motorTrigger(backright,0,0,20);

motorTrigger(frontleft,0,0,20);

motorTrigger(backleft,0,0,20);

//PlaySound(soundBlip);

//PlaySound(soundUpwardTones);

//wait1Msec(100);

}

else if(currHeading <=357 && currHeading >= 180)

{

//PlaySound(soundDownwardTones);

motorTrigger(frontright,0,0,-20);

motorTrigger(backright,0,0,-20);

motorTrigger(frontleft,0,0,-20);

motorTrigger(backleft,0,0,-20);

}

}

}

void singleAlign()

{

alignmentLoop: while(true)

{

if(currHeading >= 5 && currHeading <180) // currHeading <= 355

{

motorTrigger(frontright,0,0,20);

motorTrigger(backright,0,0,20);

motorTrigger(frontleft,0,0,20);

motorTrigger(backleft,0,0,20);

//PlaySound(soundBlip);

//PlaySound(soundUpwardTones);

//wait1Msec(100);

}

else if(currHeading <=355 && currHeading >= 180)

{

//PlaySound(soundDownwardTones);

motorTrigger(frontright,0,0,-20);

motorTrigger(backright,0,0,-20);

motorTrigger(frontleft,0,0,-20);

motorTrigger(backleft,0,0,-20);

}

else

{

alignmentLoop: break;

//PlaySound(soundBeepBeep);

}

}

}

task calibrateDrive()//needs work, doing some wack stuff

{

calibrate = 360 - currHeading;

PlaySound(soundUpwardTones);

wait1Msec(500);

/\*

initializeRobot();

HTGYROstartCal(Gyro);

delTime = 0;//calibration

prevHeading = 0;

curRate = 0;

wait1Msec(500);

\*/

}

void turnDegrees(int degreesToTurn)

{

PlaySound(soundFastUpwardTones);

int initialHeading = currHeading;//0

turnDegreesLoop: while(true)

{

int sampler = currHeading;

if((sampler - initialHeading) <= degreesToTurn)

{

motorTrigger(frontright,0,0,-20);

motorTrigger(backright,0,0,-20);

motorTrigger(frontleft,0,0,-20);

motorTrigger(backleft,0,0,-20);

wait1Msec(10);

}

else

{

stopMotors();

PlaySound(soundBeepBeep);

turnDegreesLoop: break;

}

}

}

task main()

{

initializeRobot();

StartTask(getSensorValues);

StartTask(getHeading);

//StartTask(autoAlign);

waitForStart();

move(90,70,100); //move function needs to be fixed

//singleAlign();

move(0,70,800);//(int angle, int power, float time) needs to be fixed

singleAlign();

wait1Msec(500);

PlaySound(soundFastUpwardTones);

moveForward();

StopTask(autoAlign);

PlaySound(soundDownwardTones);

turnDegrees(90);

singleAlign();

wait1Msec(5000000);

while(true)// post proccessing

{

wait1Msec(2000);

}

}