#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, emptyServo1, tServoStandard)

#pragma config(Servo, srvo\_S2\_C1\_2, emptyServo2, 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, rollingBack, tServoStandard)

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

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

#include "JoystickDriver.c"

#include "hitechnic-gyro.h"

#include "AdvancedSensors.c"

#include "hitechnic-sensormux.h"

#include "hitechnic-irseeker-v2.h"

//#include "accelerometer.h"

float initial, currHeading, globalZ, accelValue, lightValue; //initial gyro reading

float delTime = 0;//calibration

float prevHeading = 0; //USED

float curRate = 0;

int deadZone = 15;

float calibrate =0;

float slideBtn = 0;

float clampPress = 0;

const tMUXSensor Light = msensor\_S1\_4;

const tMUXSensor Accel = msensor\_S1\_1;

//const tMUXSensor Light2 = msensor\_S1\_1; //USE ONLY WHEN YOU HAVE 2 IR

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

{

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

}

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[rollingBack] = 200;

servo[rollingSide] = 200; //rolling goal clamps are open

return;

}

task FPSDrive()

{

while(true)

{

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

getJoystickSettings(joystick);

int y= joystick.joy1\_y1;

int x= joystick.joy1\_x1;

int z= joystick.joy1\_x2;

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

int trueZ = atan(joystick.joy1\_y1/joystick.joy1\_x1);

globalZ = trueZ;

if(y < 0 && x > 0) //fourth quadrant (ask lucas)

{

trueZ = (abs(trueZ) + 270);

globalZ = trueZ;

}

else if(y < 0 && x < 0)

{

trueZ = (trueZ + 180);

globalZ = trueZ;

}

else if(y > 0 && x < 0)

{

trueZ = (abs(trueZ) + 90);

globalZ = trueZ;

}

nxtDisplayCenteredBigTextLine(5,"%d",globalZ);

x = trueX;

y = trueY;

if ( (abs(joystick.joy1\_x1) < deadZone) && (abs(joystick.joy1\_y1) < deadZone) && (abs(joystick.joy1\_x2) < deadZone)) //drive only if joystick out of dead zones

{

x = 0;

y = 0;

z = 0;

}

motorTrigger(frontRight,x\*.75,-y\*.75,z);

motorTrigger(backRight,-x\*.75,-y\*.75,z);

motorTrigger(frontLeft,x\*.75,y\*.75,z);

motorTrigger(backLeft,-x\*.75,y\*.75,z);

//PRiansh dont go above here

/\*

getAccel(nAccelSensorPort, nXAxis, nYAxis, nZAxis);

nxtInvertLine(kXCenter, kYCenter, kXCenter + (int) (fXAxis \* kWidth), kYCenter);

nxtInvertLine(kXCenter, kYCenter, kXCenter + (int) (fYAxis \* kWidthY), kYCenter + (int) (fYAxis \* kWidthY));

nxtInvertLine(kXCenter, kYCenter, kXCenter, kYCenter + (int) (fZAxis \* kWidth));

fXAxis = ((float) nXAxis) / kGravityConst;

fYAxis = ((float) nYAxis) / kGravityConst;

fZAxis = ((float) nZAxis) / kGravityConst;

// Fudge to minimize very small values

if (abs(fXAxis) <= 0.03)

fXAxis = 0;

if (abs(fYAxis) <= 0.03)

fYAxis = 0;

if (abs(fZAxis) <= 0.03)

fZAxis = 0;

nxtInvertLine(kXCenter, kYCenter, kXCenter + (int) (fXAxis \* kWidth), kYCenter);

nxtInvertLine(kXCenter, kYCenter, kXCenter + (int) (fYAxis \* kWidthY), kYCenter + (int) (fYAxis \* kWidthY));

nxtInvertLine(kXCenter, kYCenter, kXCenter, kYCenter + (int) (fZAxis \* kWidth));

nxtDisplayStringAt(58, 63, " Raw");

nxtDisplayStringAt(58, 55, "X:%5d", nXAxis);

nxtDisplayStringAt(58, 47, "Y:%5d", nYAxis);

nxtDisplayStringAt(58, 39, "Z:%5d", nZAxis);

nxtDisplayStringAt(58, 31, " Normlz");

nxtDisplayStringAt(58, 23, "X:%5.2f", fXAxis);

nxtDisplayStringAt(58, 15, "Y:%5.2f", fYAxis);

nxtDisplayStringAt(58, 7, "Z:%5.2f", fZAxis);

wait1Msec(100);

\*/

//priansh dont go below here

//too bad Lucas, I like living on the edge

accelValue = SensorValue(Accel);

accelValue /= 200;

lightValue = SensorValue(Light);

}

}

task getHeading() {

HTGYROstartCal(Gyro);

PlaySound(soundBeepBeep);

while (true) {

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;

}

}

task liftFunction()

{

int triggerValue;

while(true)

{

if (abs(joystick.joy2\_y2) < deadZone) //drive only if joystick out of dead zones

{

triggerValue = 0;

}

triggerValue = joystick.joy2\_y2;

motor[liftRight] = -triggerValue;

motor[liftLeft] = triggerValue;

}

}

task driverSweep()

{

while(joy1Btn(5))

{

motor[collector] = 80;

}

while(joy1Btn(7))

{

motor[collector] = -80;

}

}

task motorFunctions()

{

int sweeperTrigger;

while(true)//

{

if(abs(joystick.joy2\_y1) < deadZone)

{

sweeperTrigger = 0;

}

sweeperTrigger = joystick.joy2\_y1;

motor[collector] = sweeperTrigger;

}

}

task main()

{

initializeRobot();

StartTask(getHeading);

StartTask(FPSDrive);

waitForStart();

while(true)

{

getJoystickSettings(joystick);

StartTask(motorFunctions);

StartTask(driverSweep);

StartTask(liftFunction);

if (joy2Btn(3))

{

servo[scoreLeft] = 200;

servo[scoreRight] = 255-200;

}

if (joy2Btn(4))//score position

{

slideBtn ++;

if(slideBtn % 2 == 1)

{

servo[scoreLeft] = 255;

servo[scoreRight] = 0;

}

else

{

servo[scoreLeft] = 35;

servo[scoreRight] = 255-35;

}

wait1Msec(500);

}

if (joy2Btn(2) == 1)//resting position

{

servo[scoreLeft] = 35;

servo[scoreRight] = 255-35;

}

if (joy1Btn(4)== 1)//changed 5 to 3

{

clampPress++;

if(clampPress % 2 == 1)

{

servo[rollingSide] = 250;

servo[rollingBack] = 250;//clamp onto rolling goal on side of robot

}

else

{

servo[rollingSide] = 200;

servo[rollingBack] = 200;

}

// servo[rollingSide] = 250;

wait1Msec(500);

}

if (joy1Btn(7) == 1)

{

servo[rollingSide] = 200; //release rolling goal

}

if(joy1Btn(3) && joy1Btn(1))

{

PlaySound(soundFastUpwardTones);

//initializeRobot(); //Bad idea

calibrate = 360 - currHeading;

}

} //DO NOT MOVE ANYTHING BELOW THIS

/\*

DO NOT MOVE ANYTHING BELOW THIS

\*/

while (true)

{

wait1Msec(2000); //REQUIRED, DO NOT MOVE

}

}