/////////////////////////////////Robot-config.h

vex::brain Brain;

vex::digital\_out Trans = vex::digital\_out(Brain.ThreeWirePort.B);

vex::gyro Gyro = vex::gyro(Brain.ThreeWirePort.H);

vex::controller Controller = vex::controller();

vex::digital\_in BallTop = vex::digital\_in(Brain.ThreeWirePort.G);

vex::motor Roll = vex::motor(vex::PORT19,vex::gearSetting::ratio18\_1,false);

vex::motor LF = vex::motor(vex::PORT2,vex::gearSetting::ratio18\_1,false);

vex::motor LM = vex::motor(vex::PORT4,vex::gearSetting::ratio18\_1,false);

vex::motor LB = vex::motor(vex::PORT5,vex::gearSetting::ratio18\_1,true);

vex::motor RF = vex::motor(vex::PORT6,vex::gearSetting::ratio18\_1,true);

vex::motor RM = vex::motor(vex::PORT7,vex::gearSetting::ratio18\_1,true);

vex::motor RB = vex::motor(vex::PORT8,vex::gearSetting::ratio18\_1,false);

vex::motor FlyL = vex::motor(vex::PORT9,vex::gearSetting::ratio18\_1,true);

vex::motor FlyR = vex::motor(vex::PORT10,vex::gearSetting::ratio18\_1,true);

vex::motor CapFlip = vex::motor(vex::PORT11,vex::gearSetting::ratio18\_1,false);

vex::motor Chain = vex::motor(vex::PORT21,vex::gearSetting::ratio18\_1,true);

vex::vision::signature REDFLAG (1, 7001, 10811, 8906, -869, -179, -524, 3, 0);

vex::vision::signature BLUEFLAG (2, -3013, -2549, -2781, 10803, 11651, 11227, 3, 0);

vex::vision::signature BALL (3, 227, 609, 418, -5229, -4229, -4729, 3, 0);

vex::vision::signature LINE (4, -1385, -931, -1158, 1065, 1765, 1416, 0.7, 0);

vex::vision::signature SIG\_5 (5, 0, 0, 0, 0, 0, 0, 3, 0);

vex::vision::signature SIG\_6 (6, 0, 0, 0, 0, 0, 0, 3, 0);

vex::vision::signature SIG\_7 (7, 0, 0, 0, 0, 0, 0, 3, 0);

vex::vision Vision (vex::PORT13, 50, REDFLAG, BLUEFLAG, BALL, LINE, SIG\_5, SIG\_6, SIG\_7);

/\*vex::vision::signature REDFLAG (1, 7511, 7999, 7754, -161, 299, 68, 4, 0);

vex::vision::signature BLUEFLAG (2, -3217, -2747, -2982, 11597, 12619, 12108, 4.099999904632568, 0);

vex::vision::signature BALL (3, 573, 1097, 836, -4403, -3911, -4156, 3, 0);

vex::vision::signature SIG\_4 (4, 0, 0, 0, 0, 0, 0, 3, 0);

vex::vision::signature SIG\_5 (5, 0, 0, 0, 0, 0, 0, 3, 0);

vex::vision::signature SIG\_6 (6, 0, 0, 0, 0, 0, 0, 3, 0);

vex::vision::signature SIG\_7 (7, 0, 0, 0, 0, 0, 0, 3, 0);

vex::vision::signature LINE (4, -1, 1, 0, -1, 1, 0, 0.6000000238418579, 0);

vex::vision::signature GREYTILE (5, 0, 0, 0, 0, 0, 0, 3, 0);

vex::vision Vision (vex::PORT13, 75, REDFLAG, BLUEFLAG, BALL, LINE, GREYTILE, SIG\_6, SIG\_7);\*/

#include "robot-config.h"

vex::competition Competition;

using namespace std;

char Alliance = 'R';

int AutoNumber = 1;//76 red 106 blue

float rpm = 0, rpmError = 0, TurnDiff = 0, TempHeight = 0, TempWidth = 0, TurnDir = 1, TempXDist = 200,GlobalFlagOffset=150;

int RPMGoal = 114, fly = 4, R = 4, Tran = 0, Shoot = 0, SnapToFlag = 0, AutoRunning = 0,FlipCount;//flywheel//Roller//Transmission

int T1 = 0, T3 = 0;

float avgSpeed = 0;

float avgError = 0;

string R1 = "R1", R2 = "R2", L1 = "L1", L2 = "L2", A = "A", B = "B", Y = "Y", X = "X", Up = "Up", Down = "Down", Right = "Right", Left = "Left";

float GlobalGyro=0;

int GyroTrack()

{float GyroAdd=0;

float GyroTempCheck=0;

while(1)

{

GyroAdd=Gyro.value(vex::rotationUnits::raw)-GyroTempCheck;

GyroTempCheck=Gyro.value(vex::rotationUnits::raw);

GlobalGyro+=GyroAdd;

vex::task::sleep(5);

}

return 0;

}

void run(vex::motor motorname, double speed)

{

if (speed != 0) { motorname.spin(vex::directionType::fwd, speed, vex::velocityUnits::pct); }

else { motorname.stop(vex::brakeType::brake); }

}

void brake(vex::motor motorname,char BRAKE)

{ if (BRAKE=='c'){ motorname.stop(vex::brakeType::coast); }

else if (BRAKE == 'h') { motorname.stop(vex::brakeType::hold); }

else{ motorname.stop(vex::brakeType::brake); }

}

void runRPM(vex::motor motorname, double speed)

{

if (speed != 0) { motorname.spin(vex::directionType::fwd, speed, vex::velocityUnits::rpm); }

else { motorname.stop(vex::brakeType::brake); }

}

void wait(int timeMs) { vex::task::sleep(timeMs); }

int btn(string but)

{

int val = 0;

if (but == L1) { val = Controller.ButtonL1.pressing(); }

else if (but == L2) { val = Controller.ButtonL2.pressing(); }

else if (but == R1) { val = Controller.ButtonR1.pressing(); }

else if (but == R2) { val = Controller.ButtonR2.pressing(); }

else if (but == A) { val = Controller.ButtonA.pressing(); }

else if (but == B) { val = Controller.ButtonB.pressing(); }

else if (but == X) { val = Controller.ButtonX.pressing(); }

else if (but == Y) { val = Controller.ButtonY.pressing(); }

else if (but == Up) { val = Controller.ButtonUp.pressing(); }

else if (but == Down) { val = Controller.ButtonDown.pressing(); }

else if (but == Left) { val = Controller.ButtonLeft.pressing(); }

else if (but == Right) { val = Controller.ButtonRight.pressing(); }

return val;

}

float ch(int channelnumber)

{

int val = 0;

if (channelnumber == 1) { val = Controller.Axis1.value(); }

else if (channelnumber == 2) { val = Controller.Axis2.value(); }

else if (channelnumber == 3) { val = Controller.Axis3.value(); }

else if (channelnumber == 4) { val = Controller.Axis4.value(); }

if (val > 0) { val = -127.0\*cos(pow((val / 86.0), 2.0)) / 2.0 + (127.0 / 2.0); }//-100cos((x/86)^2)/2+(110/2)

else if (val < 0) { val = 127.0\*cos(pow((val / 86.0), 2.0)) / 2.0 - (127.0 / 2.0); }

return val;

}

void AllianceSelect()

{ //screen 480x272

Brain.Screen.clearScreen(vex::color::black);//show red blue and confirm

int selection = 0;

while (selection < 4)

{

Brain.Screen.setPenWidth(1);

Brain.Screen.setPenColor(vex::color::black);

if (btn(Right)==1||(Brain.Screen.xPosition() > 300 && Brain.Screen.xPosition() < 420 && Brain.Screen.yPosition() > 20 && Brain.Screen.yPosition() < 60)) { selection = 1;Controller.Screen.print("Blue"); }//if Touch within blue box:: selection=1

else if (btn(Left)==1||(Brain.Screen.xPosition() > 60 && Brain.Screen.xPosition() < 180 && Brain.Screen.yPosition() > 20 && Brain.Screen.yPosition() < 60)) { selection = 2; Controller.Screen.clearLine(1);Controller.Screen.print("Red"); } //else if Touch within red box:: selection=2

else if (selection == 3 && Brain.Screen.pressing() == 0) { selection = 4; } //else if Touch within select box&& Selection >0 :: selection=3

else if ((btn(A)&&selection>0)||(Brain.Screen.xPosition() > 180 && Brain.Screen.xPosition() < 300 && Brain.Screen.yPosition() > 100 && Brain.Screen.yPosition() < 140&&selection>0)) { selection = 3; } //else if Touch within select box&& Selection >0 :: selection=3

if (selection == 1) { Alliance = 'B'; Brain.Screen.drawRectangle(297, 17, 126, 46, vex::color::green); Brain.Screen.drawRectangle(57, 17, 126, 46, vex::color::black); } //if selection==1 draw green box around Blue and Black Box around red

else if (selection == 2) { Alliance = 'R'; Brain.Screen.drawRectangle(297, 17, 126, 46, vex::color::black); Brain.Screen.drawRectangle(57, 17, 126, 46, vex::color::green); }//else if selection==2 draw green box around red and Black Box around blue

else if (selection == 3) { Brain.Screen.drawRectangle(177, 97, 126, 46, vex::color::green); }//else if selection==3 draw green box around select

Brain.Screen.drawRectangle(60, 20, 120, 40, vex::color::red);

Brain.Screen.drawRectangle(300, 20, 120, 40, vex::color::blue);

Brain.Screen.drawRectangle(180, 100, 120, 40, vex::color::white);

Brain.Screen.setPenWidth(10);

Brain.Screen.setPenColor(vex::color::white);

Brain.Screen.printAt(105, 45, false, "RED");

Brain.Screen.printAt(340, 45, false, "BLUE");

Brain.Screen.setPenColor(vex::color::black);

Brain.Screen.printAt(210, 125, false, "SELECT");

vex::task::sleep(100);

}

}

int PrintScreen()

{

while (1)

{

Brain.Screen.clearScreen(vex::color::black);

Brain.Screen.setPenColor(vex::color::white);

Brain.Screen.printAt(1, 160, "RPM %1.2f", FlyL.velocity(vex::velocityUnits::rpm));

Brain.Screen.printAt(1, 180, "RPM Goal %d", RPMGoal);

Brain.Screen.printAt(1, 60, "%f", Gyro.value(vex::rotationUnits::raw));

Brain.Screen.printAt(80, 80, "%d", BallTop.value());

//106 best for blue

/\* Vision.takeSnapshot(BLUEFLAG,5);

Brain.Screen.setPenColor(vex::color::cyan);

for (int i=0;Vision.objects[i].exists==1;i++)

{if(Vision.objects[i].width>20&&Vision.objects[i].height>20){Brain.Screen.printAt(120,20+i\*20,"X:%d,Y:%d",Vision.objects[i].centerX,Vision.objects[i].centerY );}}

\*/

//76 best brightness for red

if (Alliance == 'B')

{//vex::motor Reset = vex::motor(vex::PORT12,vex::gearSetting::ratio18\_1,false);

//vex::vision Vision (vex::PORT12, 78, REDFLAG, BLUEFLAG, BALL, SIG\_4, SIG\_5, SIG\_6, SIG\_7);

Vision.setLedColor(255,0,0);

Vision.setLedBrightness(78);

Vision.takeSnapshot(1, 5);

Brain.Screen.setPenColor(vex::color::red);

}

else if (Alliance=='R')

{//vex::motor Reset = vex::motor(vex::PORT12,vex::gearSetting::ratio18\_1,false);

//vex::vision Vision (vex::PORT12, 106, REDFLAG, BLUEFLAG, BALL, SIG\_4, SIG\_5, SIG\_6, SIG\_7);

Vision.setLedColor(0,0,255);

Vision.setLedBrightness(79);

Vision.takeSnapshot(2, 5);

Brain.Screen.setPenColor(vex::color::cyan);

}

else if (Alliance=='Y'){//vex::motor Reset = vex::motor(vex::PORT12,vex::gearSetting::ratio18\_1,false);

//vex::vision Vision (vex::PORT12, 35, REDFLAG, BLUEFLAG, BALL, SIG\_4, SIG\_5, SIG\_6, SIG\_7);

Vision.setLedColor(255,255,102);

Vision.setLedBrightness(35);

Vision.takeSnapshot(3, 5);

Brain.Screen.setPenColor(vex::color::yellow);}

else if (Alliance=='L'){//vex::motor Reset = vex::motor(vex::PORT12,vex::gearSetting::ratio18\_1,false);

//vex::vision Vision (vex::PORT12, 35, REDFLAG, BLUEFLAG, BALL, SIG\_4, SIG\_5, SIG\_6, SIG\_7);

Vision.setLedColor(255,255,25);

Vision.setLedBrightness(140);

Vision.takeSnapshot(4, 5);

Brain.Screen.setPenColor(vex::color::white);}

else{

Vision.setLedColor(255,255,255);

Vision.setLedBrightness(35);

Vision.takeSnapshot(1);

Brain.Screen.setPenColor(vex::color::white);}

TempXDist = 200;

for (int i = 0; Vision.objects[i].exists == 1; i++)

{

if (abs(GlobalFlagOffset - Vision.objects[i].centerX) < fabs(TempXDist)) { TempXDist = GlobalFlagOffset - Vision.objects[i].centerX; }

else {}

}

TempWidth = 0;

TempHeight = 0;

for (int i = 0; Vision.objects[i].exists == 1; i++)

{

if (Vision.objects[i].width > 8 && Vision.objects[i].height > 8) { Brain.Screen.printAt(230, 20 + i \* 20, "X:%d,Y:%d W:%d H:%d D:%1.2f", Vision.objects[i].centerX, Vision.objects[i].centerY, Vision.objects[i].width, Vision.objects[i].height, 0.0157\*pow(Vision.objects[i].height, 2) - 1.0842\*Vision.objects[i].height + 19.93); }

if ((Vision.objects[i].width > 8 && Vision.objects[i].height > 8) && (Vision.objects[i].centerX > (GlobalFlagOffset-10 - TempXDist) && Vision.objects[i].centerX < (GlobalFlagOffset+10- TempXDist)) && Vision.objects[i].centerY > TempHeight) { TempHeight = Vision.objects[i].centerY; TempWidth = 150 - Vision.objects[i].centerX; }

else {}

}

TurnDiff = TempWidth;

if (TurnDiff > 0) { TurnDir = 1; }

else { TurnDir = -1; }

Brain.Screen.printAt(190, 200, "%d", Vision.objects[0].exists);

Brain.Screen.printAt(120, 200, "%1.2f", TurnDiff);

/\* Vision.takeSnapshot(BALL,5);

Brain.Screen.setPenColor(vex::color::yellow);\*/

for (int i=0;Vision.objects[i].exists==1;i++)

{if(Vision.objects[i].width>20&&Vision.objects[i].height>20){Brain.Screen.printAt(340,20+i\*20,"X:%d,Y:%d",Vision.objects[i].centerX,Vision.objects[i].centerY );}}

if (rpmError < 2.0) { Brain.Screen.setPenColor(vex::color::green); if (Shoot == 1) { Controller.rumble("."); } }

else { Brain.Screen.setPenColor(vex::color::white); }

Brain.Screen.printAt(1, 120, "RPM error %1.2f", rpmError);

wait(75);

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////

int RPMCalc()

{

while (1)

{

int TempRPM = RPMGoal;

if (fly == 2 || fly == 3)

{

TempRPM = RPMGoal;

FlyR.spin(vex::directionType::fwd, RPMGoal, vex::velocityUnits::rpm);

FlyL.spin(vex::directionType::fwd, RPMGoal, vex::velocityUnits::rpm);

while ((fly == 2 || fly == 3) && RPMGoal == TempRPM)

{

rpmError = fabs(RPMGoal - fabs(FlyL.velocity(vex::velocityUnits::rpm)));

wait(5);

}

}

else

{ FlyL.stop(vex::brakeType::coast); FlyR.stop(vex::brakeType::coast);

while ((fly == 4 || fly == 1) && RPMGoal == TempRPM){wait(100);}

//rpmError = fabs(RPMGoal - fabs(FlyL.velocity(vex::velocityUnits::rpm)));

}

wait(100);

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int ShootBall()

{

while (1)

{

if (Shoot == 1) //when button is pressed

{

fly = 2;

while (BallTop.value() == 1 && Shoot == 1) { run(Chain, 70); wait(10); }//while ball is not seen bring up

run(Chain, 0); //stop chain

while (rpmError > 2 && Shoot == 1) { wait(5); }//wait till rpm in range

while (BallTop.value() == 0 && Shoot == 1)//while ball still in bot

{

while (BallTop.value() == 0) { run(Chain, 70); wait(200); }//while ball is still in bot

run(Chain, 0);

wait(20);

}

Shoot = 0;

}

else {}

wait(20);

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int TurnToFlag()

{

while (1)

{

if (SnapToFlag == 1)

{

if (SnapToFlag == 1 && fabs(TurnDiff) > 20)

{

for (int Count = 0; Count < 10; Count++)

{

int speed = TurnDir \* 50;

LF.spin(vex::directionType::fwd, -speed, vex::velocityUnits::rpm);

LM.spin(vex::directionType::fwd, -speed, vex::velocityUnits::rpm);

LB.spin(vex::directionType::fwd, -speed, vex::velocityUnits::rpm);

RF.spin(vex::directionType::fwd, speed, vex::velocityUnits::rpm);

RM.spin(vex::directionType::fwd, speed, vex::velocityUnits::rpm);

RB.spin(vex::directionType::fwd, speed, vex::velocityUnits::rpm);

wait(10);

}

}

do

{

int speed = TurnDir \* 10;

LF.spin(vex::directionType::fwd, -speed, vex::velocityUnits::rpm);

LM.spin(vex::directionType::fwd, -speed, vex::velocityUnits::rpm);

LB.spin(vex::directionType::fwd, -speed, vex::velocityUnits::rpm);

RF.spin(vex::directionType::fwd, speed, vex::velocityUnits::rpm);

RM.spin(vex::directionType::fwd, speed, vex::velocityUnits::rpm);

RB.spin(vex::directionType::fwd, speed, vex::velocityUnits::rpm);

wait(10);

} while (SnapToFlag == 1 && fabs(TurnDiff) > 20);

LF.stop(vex::brakeType::hold);

LM.stop(vex::brakeType::hold);

LB.stop(vex::brakeType::hold);

RF.stop(vex::brakeType::hold);

RM.stop(vex::brakeType::hold);

RB.stop(vex::brakeType::hold);

wait(500);

SnapToFlag = 0;

}

wait(40);

}

return 0;

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

float enc(vex::motor motorname) {

return motorname.rotation(vex::rotationUnits::deg);

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

int TIMER2()

{

while (1)

{

vex::task::sleep(1);

T3 += 1;

}

return 0;

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

void StopDrive()

{

brake(RF, 'b');

brake(RM, 'b');

brake(RB, 'b');

brake(LF, 'b');

brake(LM, 'b');

brake(LB, 'b');

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

void rightDrive(int power)

{run(RF, power);

run(RM, power);

run(RB, power);

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

void leftDrive(int power)

{run(LF, power);

run(LM, power);

run(LB, power);

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

int Driver()

{

return 0;

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

int MoveCounter = 0;

int PID\_MOTOR\_SCALE = 1;

int PID\_MOTOR\_MAX = 100;

int PID\_MOTOR\_MIN = (-100);

int PID\_INTEGRAL\_LIMIT = 50;

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

float pid\_Kp = 0.59;

float pid\_Ki = 0.09;//0.05;

float pid\_Kd = 0.0;//.05;

int pidRunning = 0; //TURN ON OR OFF PID

float RequestedAngle = 0, pidDriveR;

int angleSentinel = 1;

void pidTurn(float globalDegrees, float pid\_Kp, float pid\_Ki, float pid\_Kd, int timeout)

{

//float pid\_Kp = 0.35;

//float pid\_Ki = 0;//0.05;

//float pid\_Kd = 0;//1.40;

int direction = 0;

//StopDrive();

GlobalGyro=0;

Brain.resetTimer();

float pidError = 0;

float pidLastError = 0;

float pidIntegral = 0;

float pidDerivative = 0;

float pidDriveR = 0;

T3 = 0;

while (T3 < timeout)

{

//Calculate Error//(Convert Dintance in inches to encoder ticks)

pidError = fabs(globalDegrees \* 10) - fabs(GlobalGyro);

// integral - if Ki is not 0(can put threshold)

if (pid\_Ki != 0)

{

// If we are inside controlable window then integrate the error

if (fabs(pidError) < PID\_INTEGRAL\_LIMIT) pidIntegral = pidIntegral + pidError;

else pidIntegral = 0;

}

else pidIntegral = 0;

///////////////////////////////////////

//CALCULATE DERIVATIVE/////////////////

pidDerivative = pidError - pidLastError;

pidLastError = pidError;

//////////////////////////////////////

//CALCULATE DRIVE/////////////////////

pidDriveR = (pid\_Kp \* pidError) + (pid\_Ki \* pidIntegral) + (pid\_Kd \* pidDerivative);

///////////////////////////////////////

//LIMIT DRIVE//////////////////////////

if (pidDriveR > PID\_MOTOR\_MAX) pidDriveR = PID\_MOTOR\_MAX;

if (pidDriveR < PID\_MOTOR\_MIN) pidDriveR = PID\_MOTOR\_MIN;

///////////////////////////////////////

//SEND POWER TO MOTORS/////////////////

if (globalDegrees > 0) direction = 1;

if (globalDegrees < 0) direction = -1;

//if(gmoveandturn==false)

rightDrive(direction \* pidDriveR \* PID\_MOTOR\_SCALE);

leftDrive(-direction \* pidDriveR \* PID\_MOTOR\_SCALE);

if ((fabs(pidError) < 5) && (fabs(avgSpeed) < 10)) {

StopDrive();

break;

}

if (Brain.timer(vex::timeUnits::msec) < 40) { avgError += pidError; }

else { avgSpeed = avgError / 3; avgError = 0; Brain.resetTimer(); }

//REFRESH RATE 60Hz

wait(20);

}

StopDrive();

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

int anglePIDControl()

{

//float pidSensorCurrentValue;

float pidError;

float pidLastError;

float pidIntegral;

float pidDerivative;

// Init the variables

pidLastError = 0;

pidIntegral = 0;

while (true)

{

Brain.Screen.clearScreen();

Brain.Screen.printAt(1, 40, "%f", GlobalGyro);

// Is PID control active ?

if (pidRunning)

{

// calculate error

//pidError = ((1100+nMotorEncoder[Angle])/47.64) - RequestedAngle;

pidError = RequestedAngle - (-1)\*LF.rotation(vex::rotationUnits::deg);// - (RequestedAngle);

// integral - if Ki is not 0

if (pid\_Ki != 0)

{

// If we are inside controlable window then integrate the error

if (fabs(pidError) < PID\_INTEGRAL\_LIMIT)

pidIntegral = pidIntegral + pidError;

else

pidIntegral = 0;

}

else

pidIntegral = 0;

// calculate the derivative

pidDerivative = pidError - pidLastError;

pidLastError = pidError;

// calculate drive

pidDriveR = (pid\_Kp \* pidError) + (pid\_Ki \* pidIntegral) + (pid\_Kd \* pidDerivative);

// limit drive

if (pidDriveR > PID\_MOTOR\_MAX)

pidDriveR = PID\_MOTOR\_MAX;

if (pidDriveR < PID\_MOTOR\_MIN)

pidDriveR = PID\_MOTOR\_MIN;

// send to motor

if (fabs(pidError) < 15) angleSentinel = 0;

else angleSentinel = 1;

}

else

{

// clear all

pidError = 0;

pidLastError = 0;

pidIntegral = 0;

pidDerivative = 0;

}

// Run at 50Hz

wait(15);

}

return 0;

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

int Move(float speed, float dist){

LF.resetRotation();

RF.resetRotation();

float dir;

if(dist<0){dir=-1;}

else{dir=1;}

float tempdir=dir;

int counter=0;

while (fabs(enc(LF)) < fabs((dist\*360/(4.0\*3.14159))\*(5.0/3.0))){

float Roffset=1.0;

if(fabs(enc(LF))<fabs((enc(RF))+5)){Roffset=0.9;}

else if(fabs(enc(LF))>fabs(enc(RF))-5){Roffset=1.1;}

else{}

if (counter<10){dir=tempdir\*0.1;counter++;Roffset=1.0;}

else if (counter<25){dir=tempdir\*0.25;counter++;}

else if (counter<45){dir=tempdir\*0.4;counter++;}

else{dir=tempdir;}

run(RF, speed\*Roffset\*dir);

run(LF, speed\*dir);

run(RM, speed\*Roffset\*dir);

run(LM, speed\*dir);

run(RB, speed\*Roffset\*dir);

run(LB, speed\*dir);

wait(10);

}

leftDrive(0);

rightDrive(0);

return 1;

}

////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////

void pre\_auton( void ) {

Gyro.startCalibration();

vex::task::sleep(2000);

//AllianceSelect();

vex::task fourth(PrintScreen);

vex::task second(ShootBall);

vex::task first(RPMCalc);

vex::task fifth(TurnToFlag);

vex::task mid2(TIMER2);

//vex::task last(anglePIDControl);

// vex::task third (Driver);

vex::task one(GyroTrack);

//CapFlip.spin(vex::directionType::fwd,10,vex::velocityUnits::rpm);

}

void autonomous( void ) {

CapFlip.spin(vex::directionType::fwd,10,vex::velocityUnits::rpm);

Trans.set(0);

fly = 2;

run(Roll, 120);

RPMGoal = 120;

wait(200);

run(CapFlip,5);

StopDrive();

//Move(8, -1);

Move(100.0, 40.0);

wait(200);

Move(50, -20.0);

Shoot = 1;

Brain.resetTimer();

while (Shoot == 1){wait(10);if (Brain.timer(vex::timeUnits::msec) > 2000){break;}}

Shoot=0;

Move(100, -23);

wait(500);

pidTurn(-90, 0.1, 0.05, 0.05, 6000);//Amount initial, secondary

run(Roll, 120);

Alliance='B';

wait(400);

leftDrive(-50);

rightDrive(-50);

wait(1000);

StopDrive();

CapFlip.rotateFor(-100,vex::rotationUnits::deg,20,vex::velocityUnits::pct);

CapFlip.stop(vex::brakeType::hold);

Alliance='Y';

while(Move(100, 48)!=1){if(Vision.objects[0].exists==1){ CapFlip.spin(vex::directionType::fwd,-15,vex::velocityUnits::rpm); }vex::task::sleep(10);}//Drive forward but if ball is seen by sensor kill flipper

Alliance='L';

leftDrive(20);

rightDrive(20);

while(Vision.objects[0].exists!=1){vex::task::sleep(5);}

//Move(100, 4); //move 6 inches past the line

wait(400);

pidTurn(87, 0.1, 0.05, 0.05, 6000);//Amount initial, secondary

CapFlip.spin(vex::directionType::fwd,10,vex::velocityUnits::rpm);

wait(100);

wait(300);

Move(100, 55);

wait(400);

Move(100, -20);

Alliance='B';

GlobalFlagOffset=150;

wait(400);

SnapToFlag=1;

wait(40);

while(SnapToFlag==1){wait(100);}

Shoot = 1;

Brain.resetTimer();

while (Shoot == 1){ wait(10);if (Brain.timer(vex::timeUnits::msec) > 2000){break;}}

Shoot=0;

run(Roll, 0);

run(CapFlip,-10);

pidTurn(90, 0.100, 0.05, 0.05, 5000);//Amount initial, secondary

Move(60, 20);

run(CapFlip,40);

Move(30, 5);

wait(500);

run(CapFlip,0);

Move(60, -15);

fly=4;

R=4;

wait(1000);

Brain.resetTimer();

pidTurn(-90,0.100, 0.05, 0.05, 5000);//Amount initial, secondary

Move(100, -60);

fly=4;

R=4;

StopDrive();

while(1){wait(100);}

}

void usercontrol( void ) {

CapFlip.spin(vex::directionType::fwd,10,vex::velocityUnits::rpm);

CapFlip.resetRotation();

SnapToFlag=0;

R=4;

FlipCount=2;

Tran=2;

while (1)

{ //CapFlip.spin(vex::directionType::fwd,10,vex::velocityUnits::rpm);

if (btn(X) == 1) { SnapToFlag = 1; }

else if ((fabs(ch(3)) > 2 || fabs(ch(4)) > 2) && SnapToFlag == 1) { SnapToFlag = 0; }

else if (SnapToFlag == 0&&AutoRunning==0&& ((fabs(ch(3)) > 2 || fabs(ch(4)) > 2))&&(Tran==2||Tran==3))

{

run(LF, (ch(3) + ch(4)) ); //(Axis3+Axis4)/2

run(LM, (ch(3) + ch(4)) ); //(Axis3+Axis4)/2

run(LB, (ch(3) + ch(4)) ); //(Axis3+Axis4)/2

run(RF, (ch(3) - ch(4)) );//(Axis3-Axis4)/2

run(RM, (ch(3) - ch(4)));//(Axis3-Axis4)/2

run(RB, (ch(3) - ch(4)) );//(Axis3-Axis4)/2

}

else if (SnapToFlag == 0&&AutoRunning==0&& ((fabs(ch(3)) > 2 || fabs(ch(4)) > 2)))

{

run(LF, (ch(3) + ch(4)) / 1.25); //(Axis3+Axis4)/2

run(LM, (ch(3) + ch(4)) / 1.25); //(Axis3+Axis4)/2

run(LB, (ch(3) + ch(4)) / 1.25); //(Axis3+Axis4)/2

run(RF, (ch(3) - ch(4)) / 1.25);//(Axis3-Axis4)/2

run(RM, (ch(3) - ch(4)) / 1.25);//(Axis3-Axis4)/2

run(RB, (ch(3) - ch(4)) / 1.25);//(Axis3-Axis4)/2

}

else if (SnapToFlag == 0 && AutoRunning == 0)

{

StopDrive();

}

else {}

if (fly == 4 && btn(R1) == 0) { fly = 1; }//Flywheel

else if (fly == 1 && btn(R1) == 1) {fly = 2; }

else if (fly == 2 && btn(R1) == 0) {fly = 3; }

else if (fly == 3 && btn(R1) == 1) {fly = 4; }

else {}

if (R == 4 && btn(R2) == 0) { run(Roll, 0); R = 1; } //Roller

else if (R == 1 && btn(R2) == 1) { run(Roll, 120); R = 2; }

else if (R == 2 && btn(R2) == 0) { run(Roll, 120); R = 3; }

else if (R == 3 && btn(R2) == 1) { run(Roll, 0); R = 4; }

else {}

if (Tran == 4 && btn(B) == 0) { Trans.set(true); Tran = 1; } //Trans

else if (Tran == 1 && btn(B) == 1) { Trans.set(false); Tran = 2; }

else if (Tran == 2 && btn(B) == 0) { Trans.set(false); Tran = 3; }

else if (Tran == 3 && btn(B) == 1) { Trans.set(true); Tran = 4; }

else {}

if (btn(A) == 1) { Shoot = 1; } //Shoot ball

else if (btn(Y) == 1) { Shoot = 0; } //Stop Mid shot

else {}

if (btn(Y) == 1) { run(Chain, 0); }

else if (btn(L1) == 1 && Shoot == 0) { run(Chain, 80); } //Controls will move chain as long as Shoot = 0 (Shoot button not pressed)

else if (btn(L2) == 1 && Shoot == 0) { run(Chain, -80); }

else if (Shoot == 0) { run(Chain, 0); }

else {}

if (btn(Up) == 1 && RPMGoal < 200) { RPMGoal += 2; wait(200); }

if (btn(Down) == 1 && RPMGoal > 50) { RPMGoal -= 2; wait(200); }

else {}

/\*if (FlipCount == 4 && btn(Left) == 0) { CapFlip.rotateFor(-110,vex::rotationUnits::deg,50,vex::velocityUnits::pct);

CapFlip.stop(vex::brakeType::hold); FlipCount = 1; } //Trans

else if (FlipCount == 1 && btn(Left) == 1) { FlipCount = 2; }

else if (FlipCount== 2 && btn(Left) == 0) { CapFlip.rotateFor(10,vex::rotationUnits::deg,50,vex::velocityUnits::pct);

CapFlip.stop(vex::brakeType::hold); FlipCount = 3; }

else if (FlipCount == 3 && btn(Left) == 1) { FlipCount = 4; }

else {} \*/

if (btn(Right)==1){run(CapFlip,-40);FlipCount=0;R=4;}

//else if (FlipCount<5){run(CapFlip,10);FlipCount++;}

else if (FlipCount<80){run(CapFlip,90);FlipCount++;}

else {CapFlip.stop(vex::brakeType::hold);}

wait(20);

}

}

int main() {

CapFlip.spin(vex::directionType::fwd,10,vex::velocityUnits::rpm);

AutoRunning = 0;

//Run the pre-autonomous function.

pre\_auton();

//Set up callbacks for autonomous and driver control periods.

Competition.autonomous( autonomous );

Competition.drivercontrol( usercontrol );

//Prevent main from exiting with an infinite loop.

while(1) {

vex::task::sleep(100);//Sleep the task for a short amount of time to prevent wasted resources.

}

while (1) { vex::task::sleep(1000); }

}