#include <kipr/botball.h>

#include <GyroLib\_4.0.h>

int lm = 3;

int close = 800;

int open = 0;

int up = 450;

int down = 900;

int rm = 0;

int claw = 2;

int arm = 3;

double rm\_value = 1500;

double lm\_value = 1500;

int line\_sensorleft = 4;

int line\_sensorright = 5;

double degree\_count = 0;

//^set this to equal which way your bot is initially facing (0 = bnorthb, 90 = beastb, etc)

double sec\_per\_degreeright = 1.999/87.5;

double sec\_per\_degreeleft = 1.999/86.455;

//^measure seconds it takes to turn a precise amount of degrees

double sec\_per\_cm = 3/39.5;

//^measure how long it takes to move a precise amount of cm

double n = 0;

double ne;

double e;

double se;

double s;

double sw;

double w;

double nw;

double drive\_length = 58.42;

//^length from one section of the board to the one next to it(right/left) in cm (splitting opponents side of the board into 8 sections so itbs easy to spit a ton of already accurate code out, quick)

//^following this layout (view from your side of table to opponents side, looking perpendicular to pvc in middle):

//s8 s7 s6 s5 (rings located around s7/s6)

//s1 s2 s3 s4 (closest to pvc in middle)

double drive\_height = 58.42;

//^dist to other section (in front/behind) in cm

double drive\_diagonal1 = 84;

//^ex: dist from s1 to s7, s2 to s6

double drive\_diagonal2 = 130;

//^ex: dist from s1 to s6, s2 to s5

double drive\_diagonal3 = 180;

//^ex: dist from s1 to s5, s8 to s4

double diagonal\_degree1 = 110;

//^ex: angle to go s1 to s6, s2 to s5

double diagonal\_degree2 = 160;

//^ex: angle to go s1 to s5, s8 to s4

int current\_section = 1;

//^section attack bot lands in or section that a scoring bot starts in

double temp\_for\_sections;

//^used for diagonal equations in the section method

double accel=1;

double max\_accel = 0;

double min\_accel = 0;

double base\_accel = 1;

double invelocity = 0;

double velocity=0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Setup Methods

void stop()

{

mav(lm,0);

mav(rm,0);

msleep(50);

}

void hop\_fence()

{

double time = seconds();

while(seconds()<=time+3)

{

mav(lm,1500);

mav(rm,1465);

}

while(seconds()<=time+5)

{

mav(lm,1500);

mav(rm,1100);

}

while(seconds()<=time+8)

{

mav(lm,1500);

mav(rm,1465);

}

}

void right\_90()

{

double time = seconds();

while(seconds()<=time+1.925)

{

mav(lm,750);

mav(rm,-750);

}

mav(lm,0);

mav(rm,0);

}

void left\_90()

{

double time = seconds();

while(seconds()<=time+1.925)

{

mav(rm,750);

mav(lm,-750);

}

mav(rm,0);

mav(lm,0);

}

void right\_45()

{

double time = seconds();

while(seconds()<=time+0.95125)

{

mav(lm,750);

mav(rm,-750);

}

mav(lm,0);

mav(rm,0);

}

void left\_45()

{

double time = seconds();

while(seconds()<=time+0.95125)

{

mav(rm,750);

mav(lm,-750);

}

mav(rm,0);

mav(lm,0);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Motor Methods

void turn\_right()

{

mav(lm,750);

mav(rm,-750);

}

void turn\_left()

{

mav(lm,-750);

mav(rm,750);

}

void drive\_forward()

{

mav(lm,lm\_value);

mav(rm,rm\_value);

}

void drive\_backward()

{

mav(lm,(lm\_value\*-1));

mav(rm,(rm\_value\*-1));

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Servo Methods

void close\_claw()

{

set\_servo\_position(claw,close);

}

void open\_claw()

{

set\_servo\_position(claw,open);

}

void set\_arm()

{

set\_servo\_position(arm,1000);

}

void arm\_up()

{

set\_servo\_position(arm,up);

}

void arm\_down()

{

set\_servo\_position(arm,down);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Line Methods

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Board Positioning Methods

void midline\_line()

{

}

void midline\_no\_line()

{

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Turning stuff

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void move\_amount\_forward(double amount)

{

double timer = seconds();

while(seconds()<=amount\*sec\_per\_cm+timer)

{

drive\_forward();

}

stop();

}

void move\_amount\_backward(double amount)

{

double timer = seconds();

while(seconds()<=amount\*sec\_per\_cm+timer)

{

drive\_backward();

}

stop();

}

//S-Curve Profile:

//Phase 1: Rest to linear increasing acceleration

//Phase 2: Max acceleration has been met

//Phase 3: Max velocity almost met, linear decrease acceleration until 0

//Phase 4: Max velocity met, acceleration 0

//Phase 5: Linear decrease acceleration t

//Phase 6: Max deceleration has been met

//Phase 7: Acceleration increase to 0

//Pt=P0+V0T+1/2A0T^2+1/6JT^3

//Vt=V0+A0T+1/2JT^2

//At=A0+JT

//where:

//P0, V0, and A0, are the starting position, velocity, and acceleration

//Pt, Vt, and At, are the position, velocity, and acceleration at time T

//J is the profile jerk (time rare of change of acceleration)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void turn\_amount\_right(double amount)

{

double amount\_over = degree\_count+amount;

degree\_count = degree\_count+amount;

if(degree\_count>360)

{

degree\_count = amount\_over-360;

}

double timer = seconds();

while(seconds()<=amount\*sec\_per\_degreeright+timer)

{

turn\_right();

}

stop();

}

//^makes the bot turn a certain amount of degrees to the right, updates which way the bot is looking

void turn\_amount\_left(double amount)

{

double amount\_under = degree\_count-amount;

degree\_count = degree\_count-amount;

if(degree\_count<0)

{

degree\_count = 360-(amount\_under\*-1);

}

double timer = seconds();

while(seconds()<=amount\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

stop();

}

//^makes the bot turn a certain number of degrees to the left, updates which way the bot is looking

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void face\_north()

{

double timer = seconds();

if(degree\_count<=180)

{

while(seconds()<=(degree\_count\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

else

{

while(seconds()<=((360-degree\_count)\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

degree\_count = 0;

}

//if the bot is facing 1-180 degrees, it turns left until it reaches 0, if bot is facing over 180 degrees, turns right until 0 is met. Updates bot direction to 0

void face\_northeast()

{

if(degree\_count<=45)

{

ne = 45-degree\_count;

}

double timer = seconds();

if(degree\_count<225)

{

if(degree\_count>45)

{

ne = degree\_count-45;

while(seconds()<=(ne\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

}

if(degree\_count<=45)

{

while(seconds()<=(ne\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

else if(degree\_count>=225)

{

ne = (135-(degree\_count-225))+45;

while(seconds()<=(ne\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

degree\_count = 45;

}

//checks if the bot is facing between 0 and 45, if so, gives ne the difference between 45 and that number. Checks if bot is facing between 45 and 225 (45+180 or halfway around the circle from 45. This is to make sure that the bot turns the correct way (going left from here is quicker than right)), if true, sets ne to equal the bots angle-45, then turns left until 45 is met, updates bots angle to 45. Checks if botbs angle is 45 or less, if so, uses ne which was set earlier to make the bot turn to the right until 45 is met. Updates botbs angle to 45. If none of these conditions have been met, it means that the bot is angled somewhere from 225-360 degrees. The equation checks how many degrees it takes to get from the botbs current position to 360/0 and then adds 45 so that the bot turns until 45 is met. Updates the degree count to 45. All of the other direction methods are the same idea, just (sometimes) altered slightly due to differences in values.

void face\_east()

{

if(degree\_count<=90)

{

e = 90-degree\_count;

}

double timer = seconds();

if(degree\_count<270)

{

if(degree\_count>90)

{

e = degree\_count-90;

while(seconds()<=(e\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

}

if(degree\_count<=90)

{

while(seconds()<=(e\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

else if(degree\_count>=270)

{

e = (180-(degree\_count-270));

while(seconds()<=(e\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

degree\_count = 90;

}

void face\_southeast()

{

if(degree\_count<=135)

{

se = 135-degree\_count;

}

double timer = seconds();

if(degree\_count<315)

{

if(degree\_count>135)

{

se = degree\_count-135;

while(seconds()<=(se\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

}

if(degree\_count<=135)

{

while(seconds()<=(se\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

else if(degree\_count>=315)

{

se = (180-(degree\_count-315));

while(seconds()<=(se\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

degree\_count = 135;

}

void face\_south()

{

if(degree\_count<=180)

{

s = 180-degree\_count;

}

double timer = seconds();

if(degree\_count>0)

{

if(degree\_count<180)

{

s = -1\*(degree\_count-180);

while(seconds()<=(s\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

}

if(degree\_count>=180)

{

s = degree\_count-180;

while(seconds()<=(s\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

else if(degree\_count==0)

{

while(seconds()<=(180\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

degree\_count=180;

}

void face\_southwest()

{

if(degree\_count<=45)

{

sw = 180-(45-degree\_count);

}

double timer = seconds();

if(degree\_count>45)

{

if(degree\_count<=225)

{

sw = (-1)\*(degree\_count-225);

while(seconds()<=(sw\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

}

if(degree\_count<=45)

{

while(seconds()<=(sw\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

else if(degree\_count>225)

{

sw = -1\*(180-(degree\_count-45));

while(seconds()<=(sw\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

degree\_count = 225;

}

void face\_west()

{

if(degree\_count<=90)

{

w = 180-(90-degree\_count);

}

double timer = seconds();

if(degree\_count>90)

{

if(degree\_count<=270)

{

w = (-1)\*(degree\_count-270);

while(seconds()<=(w\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

}

if(degree\_count<=90)

{

while(seconds()<=(w\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

else if(degree\_count>270)

{

w = -1\*(180-(degree\_count-90));

while(seconds()<=(w\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

degree\_count = 270;

}

void face\_northwest()

{

if(degree\_count<=135)

{

nw = 180-(135-degree\_count);

}

double timer = seconds();

if(degree\_count>135)

{

if(degree\_count<=315)

{

nw = (-1)\*(degree\_count-315);

while(seconds()<=(nw\*sec\_per\_degreeright)+timer)

{

turn\_right();

}

stop();

}

}

if(degree\_count<=135)

{

while(seconds()<=(nw\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

else if(degree\_count>315)

{

nw = -1\*(180-(degree\_count-135));

while(seconds()<=(nw\*sec\_per\_degreeleft)+timer)

{

turn\_left();

}

stop();

}

degree\_count = 315;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void calibrate\_south\_wall()

{

face\_north();

degree\_count = 0;

double timer = seconds();

while(seconds()<=timer+3)

{

drive\_backward();

}

move\_amount\_forward(21.36);

stop();

}

//^backs bot up against the south pvc so that it becomes straight and faces north, sets degree count to equal 0

void calibrate\_north\_wall()

{

face\_south();

degree\_count = 180;

double timer = seconds();

while(seconds()<=timer+3)

{

drive\_backward();

}

move\_amount\_forward(21.36);

stop();

}

void calibrate\_east\_wall()

{

face\_west();

degree\_count = 270;

double timer = seconds();

while(seconds()<=timer+3)

{

drive\_backward();

}

move\_amount\_forward(21.36);

stop();

}

void calibrate\_west\_wall()

{

face\_east();

degree\_count = 90;

double timer = seconds();

while(seconds()<=timer+3)

{

drive\_backward();

}

move\_amount\_forward(21.36);

stop();

}

void quick\_set()

{

arm\_up();

open\_claw();

double timer = seconds();

drive\_with\_gyro(1500,10000);

stop();

msleep(250);

timer=seconds();

drive\_with\_gyro(-1500,2200);

stop();

msleep(250);

move\_amount\_forward(5);

face\_northwest();

move\_amount\_forward(10);

face\_north();

move\_amount\_forward(40);

timer=seconds();

while(seconds()<=timer+5)

{

drive\_backward();

}

stop();

msleep(250);

move\_amount\_forward(21.36);

msleep(250);

face\_east();

timer=seconds();

while(seconds()<=timer+2)

{

drive\_backward();

}

msleep(250);

move\_amount\_forward(21.36);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void move\_to\_section(int section)

{

double timer = seconds();

if(current\_section==1)

{

current\_section = section;

if(section<=4)

{

if(section>1)

{

face\_east();

move\_amount\_forward(drive\_length\*(section-1));

}

}

if(section==8)

{

face\_north();

move\_amount\_forward(drive\_height);

}

else if(section==7)

{

face\_northeast();

move\_amount\_forward(drive\_diagonal1);

}

else if(section==6)

{

timer = seconds();

if(degree\_count>=180+diagonal\_degree1)

{

while(seconds()<=((360-degree\_count)+diagonal\_degree1)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

else if(degree\_count<180+diagonal\_degree1)

{

if(degree\_count<=diagonal\_degree1)

{

while(seconds()<=(diagonal\_degree1-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

else

{

while(seconds()<=(degree\_count-diagonal\_degree1)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

}

move\_amount\_forward(drive\_diagonal2);

degree\_count = diagonal\_degree1;

}

else if(section==5)

{

timer = seconds();

if(degree\_count>=180+diagonal\_degree2)

{

while(seconds()<=((360-degree\_count)+diagonal\_degree2)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

else if(degree\_count<180+diagonal\_degree2)

{

if(degree\_count<=diagonal\_degree2)

{

while(seconds()<=(diagonal\_degree2-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

else

{

while(seconds()<=(degree\_count-diagonal\_degree2)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

}

move\_amount\_forward(drive\_diagonal3);

degree\_count = diagonal\_degree2;

}

}

else if(current\_section==2)

{

current\_section = section;

if(section<=4)

{

if(section>2)

{

face\_east();

move\_amount\_forward(drive\_length\*(section-2));

}

}

if(section==1)

{

face\_west();

move\_amount\_forward(drive\_length);

}

else if(section==8)

{

face\_northwest();

move\_amount\_forward(drive\_diagonal1);

}

else if(section==7)

{

face\_north();

move\_amount\_forward(drive\_height);

}

else if(section==6)

{

face\_northeast;

move\_amount\_forward(drive\_diagonal1);

}

else if(section==5)

{

timer = seconds();

if(degree\_count>=180+diagonal\_degree1)

{

while(seconds()<=((360-degree\_count)+diagonal\_degree1)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

else if(degree\_count<180+diagonal\_degree1)

{

if(degree\_count<=diagonal\_degree1)

{

while(seconds()<=(diagonal\_degree1-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

else

{

while(seconds()<=(degree\_count-diagonal\_degree1)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

}

move\_amount\_forward(drive\_diagonal2);

degree\_count = diagonal\_degree1;

}

}

else if(current\_section==3)

{

current\_section = section;

if(section<=2)

{

face\_west();

move\_amount\_forward(drive\_length\*((section-3)\*-1));

}

else if(section==4)

{

face\_east();

move\_amount\_forward(drive\_length);

}

else if(section==8)

{

temp\_for\_sections = 360-diagonal\_degree1;

timer = seconds();

if(degree\_count>=temp\_for\_sections)

{

while(seconds()<=(degree\_count-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<=temp\_for\_sections-180)

{

while(seconds()<=(degree\_count+diagonal\_degree1)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<temp\_for\_sections)

{

if(degree\_count>temp\_for\_sections-180)

{

while(seconds()<=(temp\_for\_sections-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

}

move\_amount\_forward(drive\_diagonal2);

degree\_count = temp\_for\_sections;

}

else if(section==7)

{

face\_northwest();

move\_amount\_forward(drive\_diagonal1);

}

else if(section==6)

{

face\_north();

move\_amount\_forward(drive\_height);

}

else if(section==5)

{

face\_northeast();

move\_amount\_forward(drive\_diagonal1);

}

}

else if(current\_section==4)

{

current\_section = section;

if(section>=1)

{

if(section<4)

{

face\_west();

if(section==1)

{

move\_amount\_forward(drive\_length\*3);

}

else if(section==2)

{

move\_amount\_forward(drive\_length\*2);

}

else if(section==3)

{

move\_amount\_forward(drive\_length);

}

}

}

if(section==8)

{

temp\_for\_sections = 360-diagonal\_degree2;

timer = seconds();

if(degree\_count>=temp\_for\_sections)

{

while(seconds()<=(degree\_count-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<=temp\_for\_sections-180)

{

while(seconds()<=(degree\_count+diagonal\_degree1)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<temp\_for\_sections)

{

if(degree\_count>temp\_for\_sections-180)

{

while(seconds()<=(temp\_for\_sections-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

}

move\_amount\_forward(drive\_diagonal3);

degree\_count = temp\_for\_sections;

}

else if(section==7)

{

temp\_for\_sections = 360-diagonal\_degree1;

timer = seconds();

if(degree\_count>=temp\_for\_sections)

{

while(seconds()<=(degree\_count-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<=temp\_for\_sections-180)

{

while(seconds()<=(degree\_count+diagonal\_degree1)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<temp\_for\_sections)

{

if(degree\_count>temp\_for\_sections-180)

{

while(seconds()<=(temp\_for\_sections-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

}

move\_amount\_forward(drive\_diagonal2);

degree\_count = temp\_for\_sections;

}

else if(section==6)

{

face\_northwest();

move\_amount\_forward(drive\_diagonal1);

}

else if(section==5)

{

face\_north();

move\_amount\_forward(drive\_height);

}

}

else if(current\_section==5)

{

current\_section = section;

if(section>=6)

{

face\_west();

move\_amount\_forward(drive\_length\*(section-5));

}

else if(section==4)

{

face\_south();

move\_amount\_forward(drive\_height);

}

else if(section==3)

{

face\_southwest();

move\_amount\_forward(drive\_diagonal1);

}

else if(section==2)

{

temp\_for\_sections = 180+diagonal\_degree1;

timer = seconds();

if(degree\_count>=temp\_for\_sections)

{

while(seconds()<=(degree\_count-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<=temp\_for\_sections-180)

{

while(seconds()<=(degree\_count+360-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<temp\_for\_sections)

{

if(degree\_count>temp\_for\_sections-180)

{

while(seconds()<=(temp\_for\_sections-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

}

move\_amount\_forward(drive\_diagonal2);

degree\_count = temp\_for\_sections;

}

else if(section==1)

{

temp\_for\_sections = 180+diagonal\_degree2;

timer = seconds();

if(degree\_count>=temp\_for\_sections)

{

while(seconds()<=(degree\_count-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<=temp\_for\_sections-180)

{

while(seconds()<=(degree\_count+360-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<temp\_for\_sections)

{

if(degree\_count>temp\_for\_sections-180)

{

while(seconds()<=(temp\_for\_sections-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

}

move\_amount\_forward(drive\_diagonal3);

degree\_count = temp\_for\_sections;

}

}

else if(current\_section==6)

{

current\_section = section;

if(section>=7)

{

face\_west();

move\_amount\_forward(drive\_length\*(section-6));

}

else if(section==3)

{

face\_south();

move\_amount\_forward(drive\_height);

}

else if(section==4)

{

face\_southeast();

move\_amount\_forward(drive\_diagonal1);

}

else if(section==2)

{

face\_southwest();

move\_amount\_forward(drive\_diagonal1);

}

else if(section==1)

{

temp\_for\_sections = 180+diagonal\_degree1;

timer = seconds();

if(degree\_count>=temp\_for\_sections)

{

while(seconds()<=(degree\_count-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<=temp\_for\_sections-180)

{

while(seconds()<=(degree\_count+360-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<temp\_for\_sections)

{

if(degree\_count>temp\_for\_sections-180)

{

while(seconds()<=(temp\_for\_sections-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

}

move\_amount\_forward(drive\_diagonal2);

degree\_count = temp\_for\_sections;

}

}

else if(current\_section==7)

{

current\_section = section;

if(section<=6)

{

if(section>4)

{

face\_east();

move\_amount\_forward(drive\_length\*((section-7)\*-1));

}

}

if(section==8)

{

face\_west();

move\_amount\_forward(drive\_length);

}

else if(section==2)

{

face\_south();

move\_amount\_forward(drive\_height);

}

else if(section==1)

{

face\_southwest();

move\_amount\_forward(diagonal\_degree1);

}

else if(section==3)

{

face\_southeast();

move\_amount\_forward(diagonal\_degree1);

}

else if(section==4)

{

temp\_for\_sections = 180-diagonal\_degree1;

timer = seconds();

if(degree\_count>=temp\_for\_sections)

{

while(seconds()<=(degree\_count-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<=temp\_for\_sections-180)

{

while(seconds()<=(degree\_count+360-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<temp\_for\_sections)

{

if(degree\_count>temp\_for\_sections-180)

{

while(seconds()<=(temp\_for\_sections-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

}

move\_amount\_forward(drive\_diagonal2);

degree\_count = temp\_for\_sections;

}

}

else if(current\_section==8)

{

current\_section = section;

if(section<=7)

{

if(section>4)

{

face\_east();

move\_amount\_forward(drive\_length\*((section-8)\*-1));

}

}

if(section==1)

{

face\_south();

move\_amount\_forward(drive\_height);

}

else if(section==2)

{

face\_southeast();

move\_amount\_forward(drive\_diagonal1);

}

else if(section==3)

{

temp\_for\_sections = 180-diagonal\_degree1;

timer = seconds();

if(degree\_count>=temp\_for\_sections)

{

while(seconds()<=(degree\_count-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<=temp\_for\_sections-180)

{

while(seconds()<=(degree\_count+360-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<temp\_for\_sections)

{

if(degree\_count>temp\_for\_sections-180)

{

while(seconds()<=(temp\_for\_sections-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

}

move\_amount\_forward(drive\_diagonal2);

degree\_count = temp\_for\_sections;

}

else if(section==4)

{

temp\_for\_sections = 180-diagonal\_degree2;

timer = seconds();

if(degree\_count>=temp\_for\_sections)

{

while(seconds()<=(degree\_count-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<=temp\_for\_sections-180)

{

while(seconds()<=(degree\_count+360-temp\_for\_sections)\*sec\_per\_degreeleft+timer)

{

turn\_left();

}

}

else if(degree\_count<temp\_for\_sections)

{

if(degree\_count>temp\_for\_sections-180)

{

while(seconds()<=(temp\_for\_sections-degree\_count)\*sec\_per\_degreeright+timer)

{

turn\_right();

}

}

}

move\_amount\_forward(drive\_diagonal3);

degree\_count = temp\_for\_sections;

}

}

current\_section=section;

}

void horizontal\_rings()

{

move\_to\_section(5);

face\_north();

move\_amount\_forward(20);

move\_amount\_backward(3);

face\_west();

move\_amount\_forward(10);

turn\_amount\_right(20);

move\_amount\_forward(30);

while(analog(line\_sensorleft)<=2000)

{

drive\_forward();

}

face\_west();

arm\_down();

move\_amount\_backward(15);

close\_claw();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//###############################################################################################################

int main()

{

start\_theta\_tracker();

declare\_motors(3,0);

declare\_degrees(659);

//calibrate\_degrees();

// ^comment out once degrees has been declared

//Variables

int rm = 0;

int lm = 3;

int arm = 1;

//###############################################################################################################

enable\_servos();

double timer=seconds();

while(seconds()<timer+118)

{

turn\_with\_gyro(-90);

turn\_with\_gyro(90);

quick\_set();

move\_to\_section(4);

calibrate\_south\_wall();

calibrate\_east\_wall();

horizontal\_rings();

stop();

msleep(1000000);

}

//###############################################################################################################

//Run--------------------------------------------------------------------------------------------------------

/\* we didnt start the fire

#its been always buring since the worlds been turning\*/

}