SubGoal #1:

* Reading black tape on gameboard and navigating.

int blackReadingone(black){

typedef enum { false, true } bool;

if (analog(0) == black){

bool firstblack = true;

while (firstblack){

movement(80,80);

if (analog(0) == black && analog(1) == black){

firstblack = false;

}

}

}

return (1);

}

SubGoal #2:

* Programing servo motors to grab and drop.

int pomsPickup(){

enable\_servos();

set\_servo\_position(0,0);

set\_servo\_position(1,900);

msleep(1000);

set\_servo\_position(1,700);

msleep(3000);

motor(3,40);

msleep(500);

motor(3,0);

set\_servo\_position(0,600);

msleep(1000);

set\_servo\_position(1,900);

msleep(3000);

motor(3,-50);

msleep(500);

return 0;

}

SubGoal #3:

* Reusable functions for basic movements.

Int turnRight (x,y){

motor(0,x);

msleep(t);

motor(1,y);

ao();

return 0;

}

Int turnLeft (x,y,t){

motor(1,x);

msleep(t);

motor(0,y);

ao();

return 0;

}

Int movement(x,y){

motor(0,x);

motor(0,y);

return 0;

}

SubGoal #4:

* Using Sensors to gudie pathways.

**[Line follower]**

int WalkTheLine(AnalogValueZero, AnalogValueOne){

if (analog(0) >= AnalogValueZero && analog(2) >= AnalogValueOne){

movement(60,60);

}

else if (analog(0) <= AnalogValueZero || analog(2) <= AnalogValueOne){

if (analog(0) <= AnalogValueZero){

turnRight();

}

if (analog(2) <= AnalogValueOne){

turnLeft();

}

}

return(0);

}

**[Ultrasonic]**

int main(){

while (analog(2) < 2900){

movement(60,60);

}

turnRight();

ao();

return 0;

}

============================================================================================

Roomba Below:

SubGoal #1:

* React to bump sensor.

int main()

{

create\_connect();

create\_full();

while (get\_create\_lbump() == 0 && get\_create\_rbump() == 0){

create\_drive\_direct(100, 100);

}

create\_stop();

printf("Distance travel: %d\n", get\_create\_distance());

int distance\_travel = get\_create\_distance();

set\_create\_distance(0);

printf("Distance travel: %d\n", distance\_travel);

while (abs(get\_create\_distance()) < distance\_travel){

create\_drive\_direct(-100, -100);

printf("%d\n", get\_create\_distance());

}

create\_stop();

create\_disconnect();

return 0;

}

SubGoal #2:

* Black tape.

int main()

{

create\_connect();

while (get\_create\_rfcliff\_amt() > 2000 || get\_create\_lfcliff\_amt() > 2000){

create\_drive\_direct(100, 100);

}

create\_stop();

set\_create\_total\_angle(0);

while (get\_create\_total\_angle() < 90){

create\_drive\_direct(-200, 200);

}

create\_stop();

create\_drive\_direct(100,100);

msleep(3000);

create\_stop();

create\_disconnect();

return 0;

}

SubGoal #3:

* Rotations.

int main()

{

create\_connect();

set\_create\_total\_angle(0);

while (get\_create\_total\_angle() != 90){

create\_drive\_direct(-50, 50);

}

printf("Angle turned: %d\n", get\_create\_total\_angle());

create\_stop();

create\_disconnect();

return 0;

}

SubGoal #4:

* Detect yellow card w/ red tape (fire).

int Camera() {

camera\_open\_black();

typedef enum { false, true } bool;

bool run = true;

while (run) {

msleep(500);

camera\_update();

int Color\_Yellow = get\_object\_count(0);

int Color\_Red = get\_object\_count(1);

int Yellow\_X = get\_object\_center\_x(0, 0);

int Yellow\_Y = get\_object\_center\_y(0, 0);

//printf("Yellow-X: %i \n ", YellowX);

//printf("Yellow-Y: %i \n ", YellowY);

int Red\_X = get\_object\_center\_x(1,0);

int Red\_Y = get\_object\_center\_y(1,0);

//printf("Red-X: %i \n ", RedX);

//printf("Red-Y: %i \n ", RedY);

if ((abs(Yellow\_X - Red\_X)) <= 10 && (abs(Yellow\_Y - Red\_Y) <= 10)){

printf("Fire!!!\n");

}

else if (Color\_Yellow == 1){

printf("One Yellow\n");

}

else if (Color\_Yellow > 1){

printf("Multiple Yellows\n");

}

else if (Color\_Red == 1){

printf("One Red\n");

}

else if (Color\_Red > 1){

printf("Multiple Reds\n");

}

else{

printf("Nothing\n");

}

}

ao();

return 0;

}