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

\* rc\_wheeled\_auto.c

\*

\* This is a program that uses the compass to turn the wheeled

\* platform and then go a certain distance.

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

#include <rc\_usefulincludes.h>

#include <roboticscape.h>

//struct to hold new data

rc\_imu\_data\_t data;

void process\_data();

double angle;

int distance;

int turn;

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

\* int main()

\*

\* This main function contains these critical components

\* - call to initialize\_cape

\* - set up the compass

\* - initiate the turn

\* - after it comes back - go a certain distance

\* - cleanup\_roboticscape() at the end

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

int main(int argc, char\*\* argv){

// always initialize cape library first

rc\_initialize();

printf("\nHello BeagleBone\n");

angle = atof(argv[1]);

if (angle > 0)

turn = 1;

else

turn = 0;

distance = atoi(argv[2]);

// done initializing so set state to RUNNING

rc\_set\_state(RUNNING);

// bring H-bridges of of standby

rc\_enable\_motors();

rc\_set\_led(GREEN,ON);

rc\_set\_led(RED,ON);

rc\_set\_motor\_free\_spin(1);

rc\_set\_motor\_free\_spin(2);

printf("Motors are now ready.\n");

// start with default config and modify based on options

rc\_imu\_config\_t conf = rc\_default\_imu\_config();

conf.dmp\_sample\_rate = 20;

conf.enable\_magnetometer = 1;

// now set up the imu for dmp interrupt operation

if(rc\_initialize\_imu\_dmp(&data, conf)){

printf("rc\_initialize\_imu\_failed\n");

return -1;

}

rc\_set\_imu\_interrupt\_func(&process\_data);

// set the unit turning

if (turn)

{

rc\_set\_motor(1, 0.2);

rc\_set\_motor(2, -0.2);

}

else

{

rc\_set\_motor(1, -0.2);

rc\_set\_motor(2, 0.2);

}

//now just wait, print\_data() will be called by the interrupt

while (rc\_get\_state()!=EXITING) {

usleep(10000);

}

int movement = 0;

// Now move forward

while (movement < distance)

{

rc\_set\_motor(1, 0.2);

rc\_set\_motor(2, 0.2);

usleep(1000000);

movement++;

}

rc\_set\_motor\_brake\_all();

// shut things down

rc\_power\_off\_imu();

rc\_cleanup();

return 0;

}

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

\* int process\_data()

\*

\* - Called each time the compass interrupts

\* - Compares angles to see if the platform has moved enough

\* - If it has, stop the platform

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

void process\_data() // imu interrupt function

{

printf("\r");

printf(" ");

printf("Angle = %6.1f\n",angle);

printf("Distance = %2d\n",distance);

printf(" %6.1f |", data.compass\_heading\_raw\*RAD\_TO\_DEG);

printf(" %6.1f |", data.compass\_heading\*RAD\_TO\_DEG);

if (turn)

{

if ((angle - data.compass\_heading\*RAD\_TO\_DEG) < 1.0)

{

rc\_set\_motor\_brake\_all();

rc\_set\_state(EXITING);

}

}

else

if ((-angle + data.compass\_heading\*RAD\_TO\_DEG) < 1.0)

{

rc\_set\_motor\_brake\_all();

rc\_set\_state(EXITING);

}

fflush(stdout);

return;

}