Movement API (for Rob to integrate with his API)

Variable Definitions:



Setup file:

Set pin direction modes

pinMode(LM\_enable, OUTPUT);

pinMode(LM\_dir, OUTPUT);

pinMode(LM\_enc, INPUT);

pinMode(RM\_enable, OUTPUT);

pinMode(RM\_dir, OUTPUT);

pinMode(RM\_enc, INPUT);

// set interrupt pull up resistors for consistency

digitalWrite(LM\_enc, HIGH);

digitalWrite(RM\_enc, HIGH);

// initialize motors (initialize LM and RM PWM at 0 when you define variables)

analogWrite(LM\_enable, LM\_PWM);

digitalWrite(LM\_dir, Forward);

analogWrite(RM\_enable, RM\_PWM);

digitalWrite(RM\_dir, !Forward);

//initialize interrupts

attachInterrupt(0,LMcounting, FALLING);

attachInterrupt(1,RMcounting, FALLING);

Functions:

void **calcSpeed**() {

LMspeedMeas = (LM\_count-oldLMcount)\*(1000/(3\*53))\*(204/loopTime); //speed in mm/s

RMspeedMeas = (RM\_count-oldRMcount)\*(1000/(3\*53))\*(204/loopTime);

oldLMcount = LM\_count;

oldRMcount = RM\_count;

}

//This function calculates the current speed of the rover. It takes the number of encoder clicks (new encoder count – old encoder count), multiplies it by 204mm/(3\*53) clicks (dimensions of the wheel and gear ratio), and divides by the loop time (converting to seconds). Then it stores the current count value as the next old count value.

int **calcPWM**(int oldPWM, int targetSpeed, int actualSpeed){

float PIDdelta = 0;

int error = 0;

error = abs(targetSpeed)-abs(actualSpeed); // calculate error

cumError += error; // calculate cumulative error

PIDdelta = Kp\*error-Kd\*(error-lastError)+Ki\*cumError;

lastError = error; // set last error

return constrain(oldPWM+int(PIDdelta),0,255); // constrain the new value to range

}

//This function calculates the new desired input of the motor according to the error in speed. It finds the error, the cumulative error, and the change since the last error and multiplies each by the appropriate PID constant. That value is then constrained to the 0-255 PWM range.

void **LMcounting**() {

LM\_count++;

}

void **RMcounting**() {

RM\_count++;

}

//These are the interrupt functions (set up in the setup file). They are kept as small as possible to minimize processor use (this function runs 159 times every wheel revolution per wheel).

Main speed function:

if (millis()-lastMilli >= loopTime) {

lastMilli = millis();

calcSpeed();

LM\_PWM = calcPWM(LM\_PWM, LMspeedRef, LMspeedMeas);

RM\_PWM = calcPWM(RM\_PWM, RMspeedRef, RMspeedMeas);

analogWrite(LM\_enable, LM\_PWM);

analogWrite(RM\_enable, RM\_PWM);

}

//This function runs only after the sample time has passed (looptime). It calculates the speed with calcSpeed(), calculates the new instruction with calcPWM(), and writes it to the motor enable ports.

Spiral function:

if (millis()-lastMilli >= loopTime) {

lastMilli = millis();

calcSpeed();

LM\_PWM = calcPWM(LM\_PWM, LMspeedRef, LMspeedMeas);

RM\_PWM = calcPWM(RM\_PWM, RMspeedRef, RMspeedMeas);

analogWrite(LM\_enable, LM\_PWM);

analogWrite(RM\_enable, RM\_PWM);

RMspeedRef = (127\*atan(Xspiral/SpiralTightness));

Xspiral++;

}

//The spiral function is different with the last two lines. The first increments the right motor’s speed (initially set at 0 while the LM is at 200) according to the arctan function. The arctan function has an asymptote at pi/2, so we convert to a 200 asymptote using 400/pi (127). Then the function is expanded by dividing the Xvalue by the SpiralTightness variable. Then increment the X value.