GMove: Gyroscopic Controlled Robotic Movement

Raymond Yap
Valley Christian High School
100 Skyway Drive
San Jose, CA, USA
mr.raymint@gmail.com

Abstract

Within the context of high precision robotics fields such as surgery, machinery requires the utmost accuracy. This paper introduces the method of proportional gyroscopic movement, applied through iterative testing with Lego Mindstorms EV3 technology. Consequences of unstable robotics on an industrial level can be as severe as death of a patient. To counteract the influence of external motion upon the linear movement of a robot, the gyroscopic algorithm ("gmove") being tested was applied to an educational Lego EV3 rover robot to perform along a laser straight reference line. Across all trials, gyroscopically oriented movement demonstrated superiority in both precision and consistency, bringing to light a novel method of precision movement in the robotic industry.

Keywords
Gyro control, Gyro sensors, EV3, Lego Mindstorms, Robotics, FIRST Lego League, FLL

1. Introduction
The first three years of experiences in FIRST Lego League ("FLL") robotic competition, particularly the failures and challenges during the robot design stage and competition, the author was inspired to pursue in-depth study in robot design and programming. It became obvious that higher precision movement of the robot is one key element of scoring higher and winning the FLL robot game. Throughout the fourth and fifth year, the author did further research, experiments, made significant improvements to the robot making it capable of moving with higher precision and eventually reached champion level of the FLL seasons in the local Silicon Valley district of California and the international arena at Razorback Invitational by the University of Arkansas, USA.

1.1 Objectives
This research project is aimed at discovering any possibilities of higher precision robot movement using non-industrial, educational Lego Mindstorms EV3 ("EV3"). The FLL robot game competition only allows original Lego bricks such as RCX, NXT, EV3, SPIKE Prime, original Lego parts, original Lego sensors, and original Lego programming environments.

In the FLL competition arena, the educational EV3 robot is considered sufficiently accurate if it can hit the target within plus minus 10 millimeters deviation. At this condition, the robot can still perform the next tasks such as reaching or capturing objects that are usually much larger than 10 millimeters in size. This project is to attempt robot movement accuracy within plus minus 10 millimeters.

1.2 Scope
The primary focus of this project is on designing a rover type of robot commonly used in FLL robotic competition. A rover robot is typically driven by a pair of motors capable of moving in a straight or curved line, forward or backward

direction, using the motor movement command available in the software library provided by Lego Education programming tool. Third party programming languages and third party Lego parts will not be discussed in this project.

1.3 Experiments

The first experiment is to measure several available EV3 motors to learn and understand whether all motors are created equal. The goal is to pick a pair of motors with minimal difference in their mechanical characteristics so as to move in a straight line of path with high accuracy.

The second experiment is to discover how accurately an EV3 rover robot performs by measuring how far apart from the expected straight line reference the rover robot arrives at the destination point. This experiment is open ended. It started by using the built-in Lego motor movement functions such as "move (50, 50, 5.5)" to move forward 5.5 rotations at 50% speed for both motors. Based on the result of this basic motor movement experiment, further improvement experiments using Lego gyro sensors are carried out.

2. Methods

2.1 Motor Measurement

Lego Mindstorms EV3 motors have a built-in rotation sensor which can be programmed by resetting to zero at the start of the run and reading its value at the end. By applying a fixed amount of load to the motor, running it at a certain speed for a fixed amount of time, total degrees of rotation of the motor run can be measured.

In this experiment, a simple pulley construction was used. The axle of the pulley was directly driven by the EV3 motor pulling a string tied to a 907 gram (2 lbs) fixed load. The motor was run at four different speed levels (25, 50, 75, 100%) for 5 seconds. At the end of each run, the total degrees of rotations were recorded and used for plotting motor characteristics graphs. These graphs will make it easier to pick two motors with similar or identical characteristics.

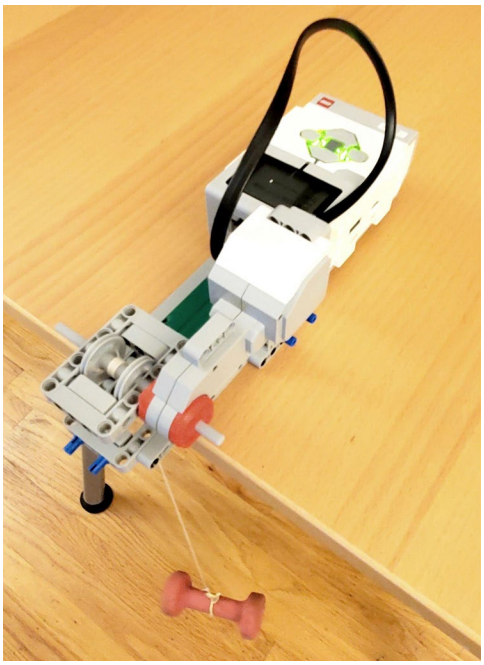


Figure 1. Lego EV3 motor measurement

For each available EV3 motor, do the following steps

1. Assemble the motor into pulley construction as shown in Figure 1.
2. Program the motor to run at speed 25% for 5 seconds and stop
3. Measure and record the degrees of rotation the motor achieved
4. Repeat step 2 to step 3 for total 10 times
5. Repeat step 2 to step 4 for speed 50%, 75%, 100%

6. Plot a graph of data measured in step 3

2.2 Robot Performance

In this experiment, an actual FLL robot game competition floor mat was used as shown in Figure 2. This is an effort to recreate a similar environment as the real world arena, in particular the wheels of the robot have a direct contact with the floor mat which has a certain amount of friction. Other factors such as ambience lighting and audio noises in the arena are insignificant, irrelevant, non contributing factors to the robot movement.

A common home DIY tool called Laser Straight is used to light up a straight line on the floor mat surface starting from the base area (the white quarter circle corner) aligned to one of grid lines. The laser line extends all the way to the other end of the mat where a millimeter graph paper is placed then secured by paper tapes after aligning one of the gridlines with the laser light path. This will be the reference line for the rover robot to arrive at. The base area is the starting position of the rover robot. The millimeter graph paper is the destination area during this experiment. The Laser Straight tool is needed only once during the initial setup of this experiment.

a) Laser Straight b) FLL robot game floor mat

Figure 2. Floor mat setup

The goal of this experiment is to compare the rover robot performance when run forward with and without gyro control. Without the use of a gyro sensor, both motors are run at the same speed. It is expected that the use of a gyro sensor will improve the performance and consistency in running along the straight line of path and arriving at destination with higher precision. This involves some programming on the rover motors where the error value from the gyro sensor reading is added and subtracted into the speed of each motor respectively so as to keep the rover staying on track along the reference line.

Once the floor mat has been set up, do the following steps

1. Place the rover robot at the base area with its front and rear pointers aligned to the reference line. See Figure 3.
2. Run the robot forward at 25% motor speed, 5.5 motor rotations (see Figure 5), and stop the destination area (on top of the millimeter graph paper).

3. At the destination area, inspect the robot front pointer, measure its deviation (in millimeters) from the reference line, and record the measurement. See Figure 4.
4. Repeat step 3 for the rear pointer
5. Repeat step 1 to step 4 for total 10 times
6. Repeat step 1 to step 5 for 50%, 75%, 100% speed
7. Use gyro sensor to control the motor speed, repeat step 1 to step 6

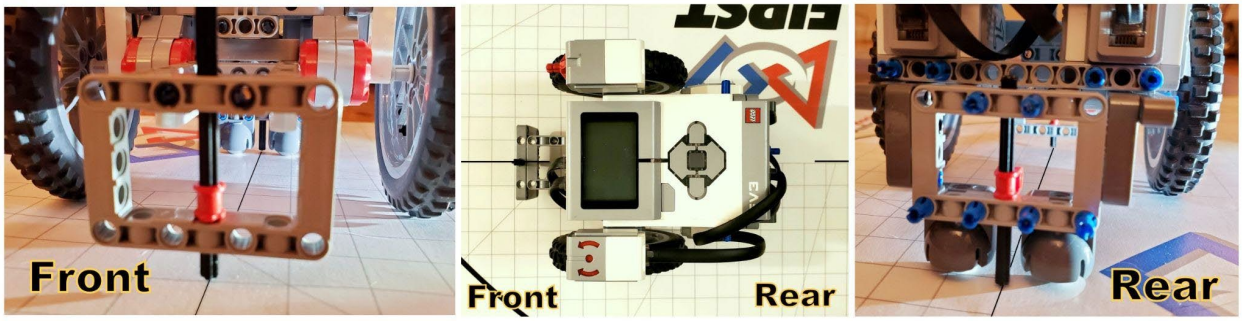


Figure 3. The Base area

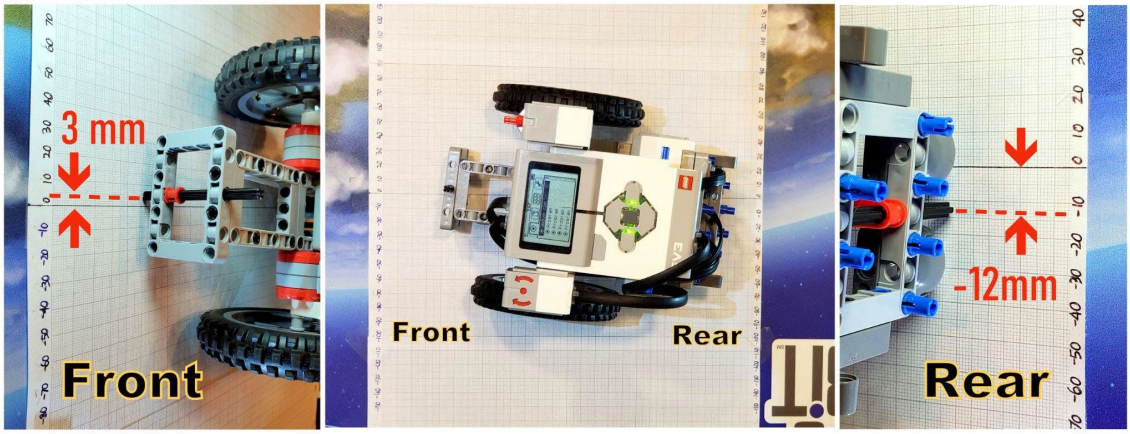


Figure 4. The Destination area

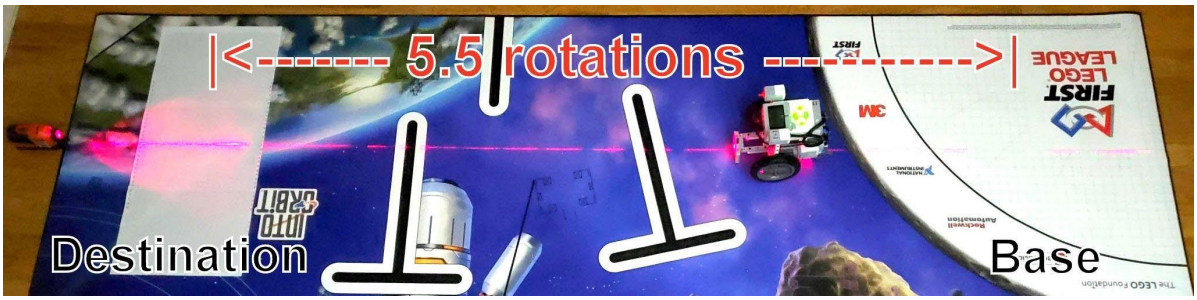


Figure 5. The distance from Base area to Destination area

3. Results and Discussion

3.1 Motor Measurement Results

The result of the motor measurement experiment described in Section 2.1 is shown in Table 1. Samples of 4 motors, each run at four different motor speeds, each for 10 times.

Table 1. Lego Mindstorms Motor Measurement

Motor	Speed	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
1	25	1265	1265	1265	1265	1264	1265	1264	1263	1264	1264

	50	2522	2520	2521	2519	2518	2521	2520	2520	2521	2522
	75	3752	3752	3752	3752	3751	3749	3753	3750	3750	3750
	100	4088	4103	4109	4091	4092	4085	4065	4062	4043	4040
2	25	1267	1266	1265	1266	1267	1267	1266	1266	1265	1266
	50	2522	2523	2521	2519	2522	2521	2520	2520	2520	2521
	75	3751	3753	3750	3752	3756	3750	3754	3753	3752	3741
	100	4060	4070	4059	4056	4047	4046	4042	3972	3979	3982
3	25	1266	1265	1265	1267	1267	1266	1265	1264	1265	1267
	50	2525	2523	2526	2520	2524	2527	2525	2525	2529	2527
	75	3756	3743	3743	3754	3753	3753	3756	3755	3756	3752
	100	4186	4170	4177	4146	4141	4157	4110	4119	4127	4110
4	25	1265	1265	1266	1266	1266	1265	1266	1266	1265	1266
	50	2519	2519	2518	2521	2519	2520	2518	2519	2519	2519
	75	3753	3751	3749	3757	3753	3757	3752	3756	3753	3752
	100	4177	4131	4106	4098	4091	3999	4001	3981	3996	3967

Graphical representation of Table 1 is shown in Figure 6.

All motors showed consistent characteristics with their linear graph when run at speeds up to 75%. At speeds above 75%, not only did they show inconsistency, their graphs were no longer linear. The intention of this experiment is to find nearly identical motor pairs for driving a rover robot. Motor 1 and Motor 2 appeared to be very similar in their characteristics as shown by their near identical graphs. Therefore, Motor 1 and Motor 2 were selected and used in the next experiment on robot performance described in Section 2.2.

With nearly identical motor characteristics between Motor 1 and Motor 2, applying the same amount of speed, using the same size of wheels, it is expected that the robot would run in a straight line using the basic motor movement command in EV3 programming.

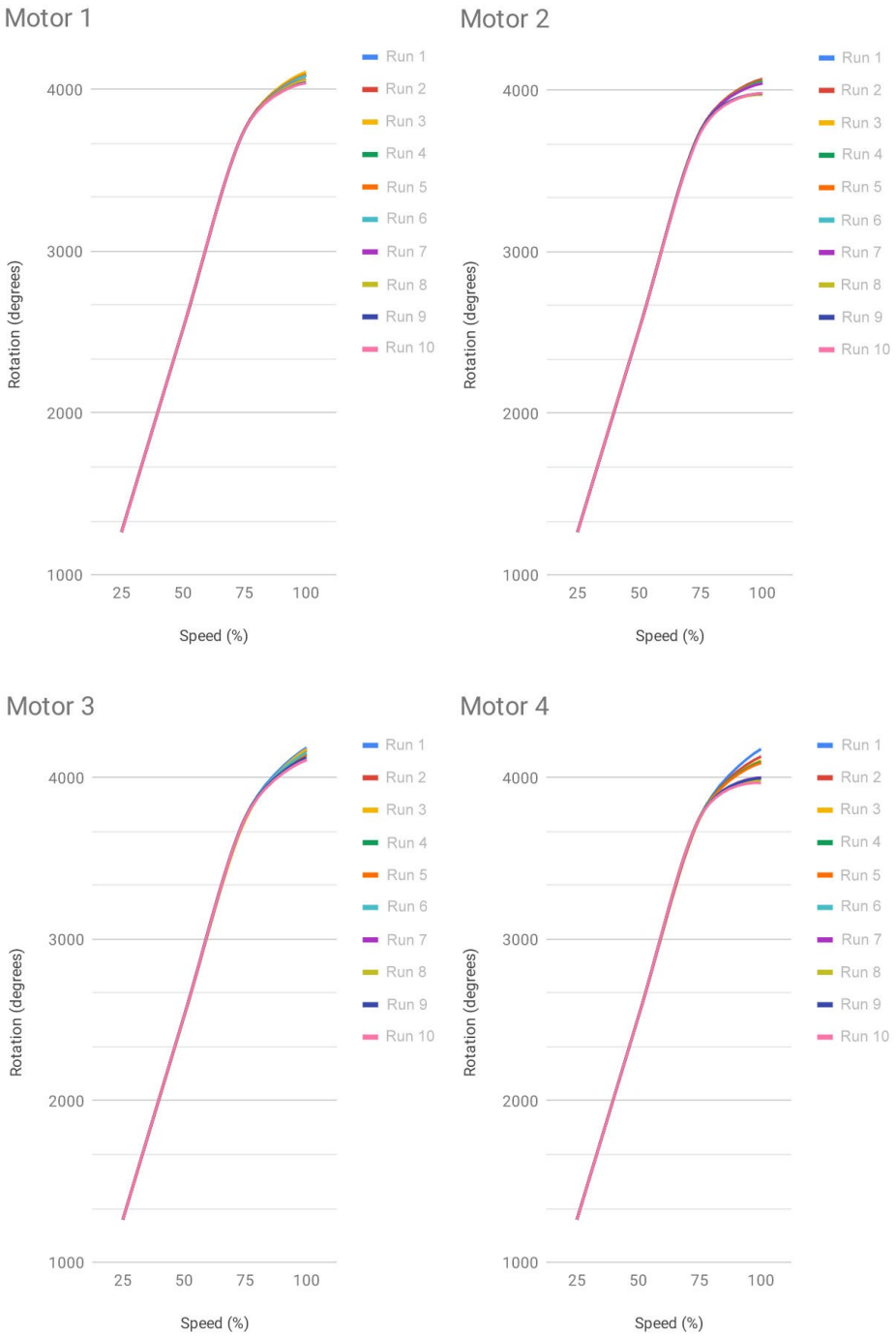
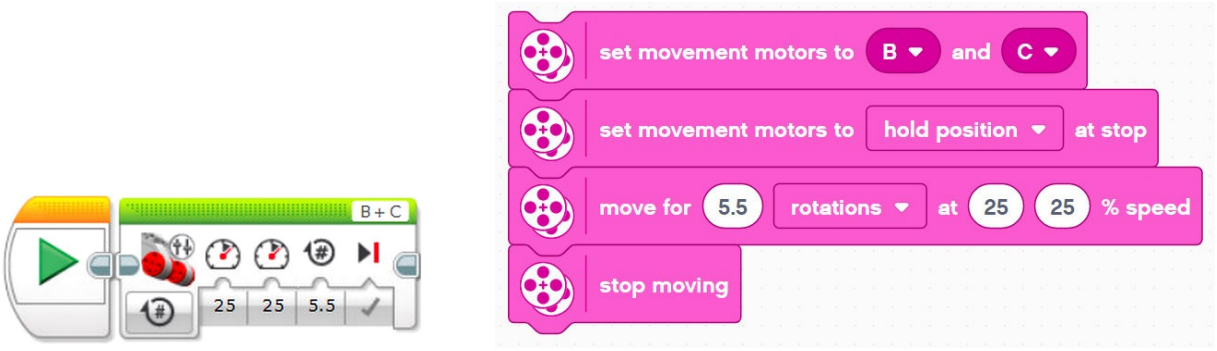


Figure 6. Lego Mindstorms motor measurement

3.2 Built-in Move Command

The following first robot performance experiment is to run the rover robot by using Lego built-in motor movement command move (Bspeed, Cspeed, rotations), where Bspeed and Cspeed are the speed (in %) of B and C motor pairs, which are the Lego EV3 motors connected to B and C ports. Figure 7 shows the built-in move command in both EV3 Lab and EV3 Education programming environments commonly used, approved for FLL competition.



a) EV3 Lab programming interface b) EV3 Education programming interface

Figure 7. Built-in Lego motor movement command

Despite the similarity between the chosen Motor 1 and Motor 2 for the rover robot, the results in Table 2 showed that the robot arrived at the destination area with significant distance deviating from the reference line. The visualization of Table 2 into scatter plots in Figure 8 showed the deviation is quite significant at relatively low speed of 25%. At the speed of 25%, only 2 out of 10 runs arrived within plus minus 10 millimeters from the reference line. It gets worse at higher speeds, none of them arrived near the reference line.

Table 2. Robot performance using built-in Lego motor movement command

move (**Bspeed**, **Cspeed**, rotations)		**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**
25, **25**, 5.5	Front	18	8	20	57	10	31	49	23	14	19
	Rear	15	5	21	50	8	27	42	19	9	17
50, **50**, 5.5	Front	101	190	46	45	43	81	24	203	164	31
	Rear	89	168	38	44	35	70	18	182	146	27
75, **75**, 5.5	Front	99	30	75	164	15	128	106	157	86	36
	Rear	87	27	69	145	7	113	94	141	71	33
100, **100**, 5.5	Front	66	52	33	155	-29	89	61	85	78	59
	Rear	65	51	31	133	-34	79	53	69	67	56

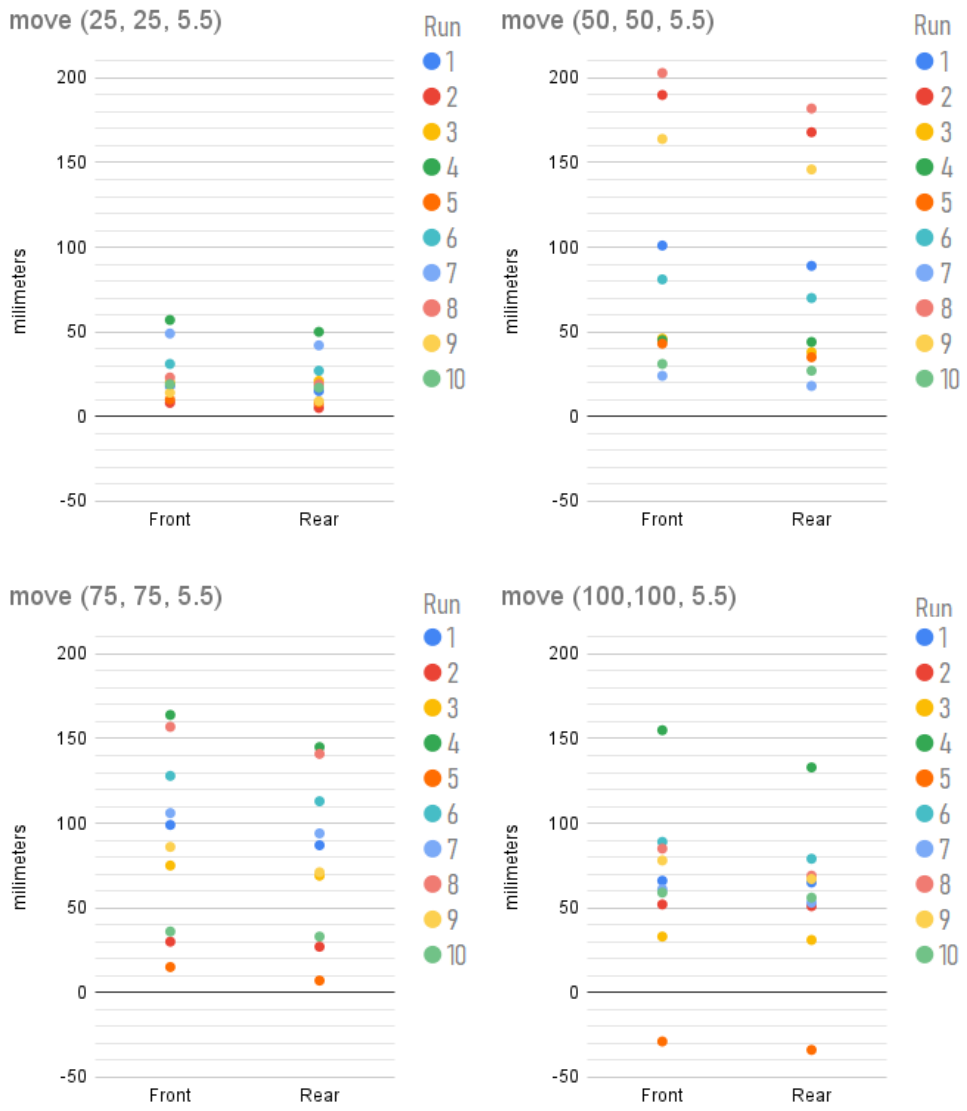


Figure 8. Robot performance using built-in Lego motor movement command

3.3 Simple GMove Command

The second robot performance experiment is to utilize a Lego gyro sensor. The idea is to aim the rover robot to the destination area, set the initial direction as a reference. As it starts moving forward, it continuously reads the direction from the gyro sensor and immediately makes corrections when it faces away from the reference direction. The gyro sensor produces negative value in degrees of angle when facing to the left and positive value to the right relative to the reference direction. The commonly known correction algorithm is by subtracting the Motor 1 speed (driving the left wheel, namely motor B in EV3) with the gyro sensor value and simultaneously adding to the Motor 2 speed (driving the wheel on the right side, motor C in EV3), as shown in Figure 9.

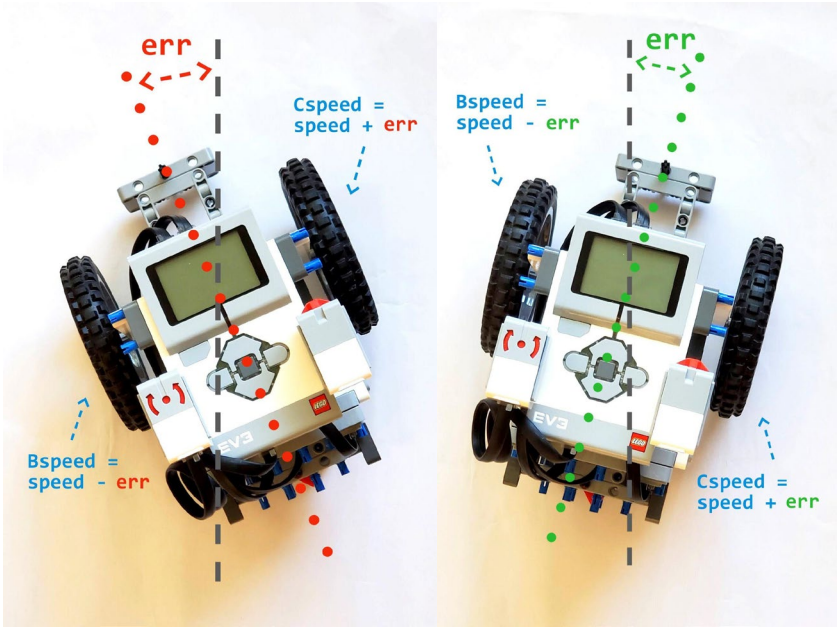


Figure 9. The basic algorithm of gyroscopic controlled robotic movement

During this experiment, it was discovered that simply subtracting and adding the gyro sensor value to the motor speed directly only worked reasonably well for lower speed at 25%.

G = gyro value
err = G
Bspeed = speed - err
Cspeed = speed + err

However, it did not work well at higher speeds. When the speed is higher, for instance at 75%, and gyro sensor value shows a relatively small value such as 2 degrees, subtracting and adding 2% to the 75% motor speeds did not immediately correct the rover direction. In other words, at higher speeds, the value of err needs to be proportionally larger to make a timely correction. A multiplier H is now introduced to see if it improves the situation.

After a few trial and error, the following conditioning seems to improve the performance,

speed $\leq$ 70: H = 1 + (speed / 150)
70 < speed $\leq$ 80: H = 1 + (speed / 100)
80 < speed $\leq$ 90: H = 1 + (speed / 50)
speed > 90: H = 1 + (speed / 25)

H can be considered as the “hardware” factor. The H formula may change accordingly as the robot design changes, such as the type and diameter of the wheels, possibly the physical position of the gyro sensor, the type of motor being used (EV3, NXT, SPIKE Prime). The value of H is then multiplied by the gyro value G and this is the optimal error value to be added to and subtracted from the given speed for Bspeed and Cspeed, the actual corrected speed of the two motors.

G = gyro value
err = G * H
Bspeed = speed - err
Cspeed = speed + err

Prior to trial and error, the initial experiment was basically done by having H = 1, thus err = G, leading to less optimal results. The improved H formula is implemented into custom gmove_simple command in Figure 10.

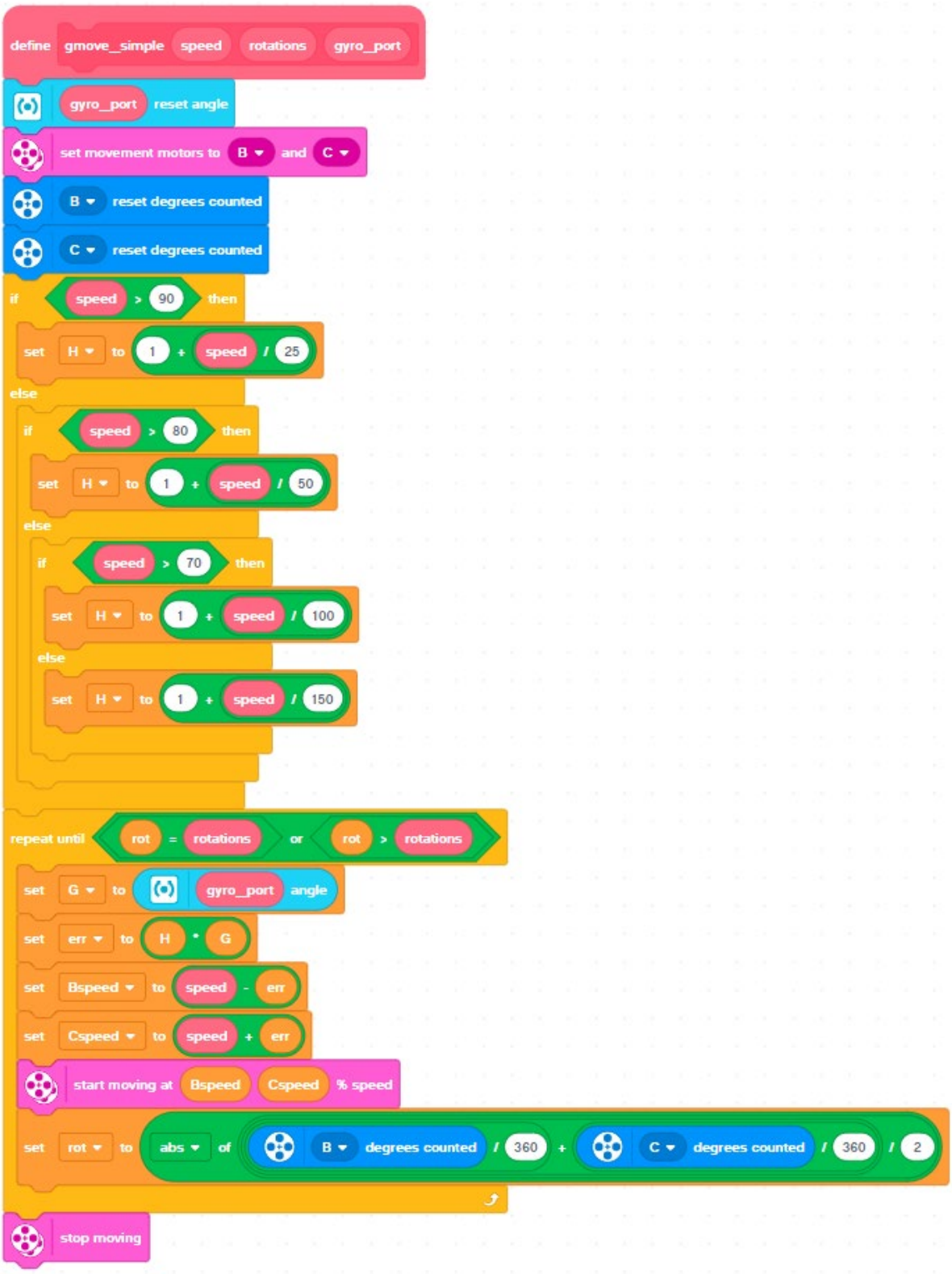


Figure 10. Custom motor movement program gmove_simple

The result of gmove_simple experiment is shown in Table 3. The use of gyro sensors significantly improved the results. The rover robot arrives at the destination area much closer to the reference line. Many of the test runs at various speeds up to 75% produced a result within plus minus 10 millimeters deviation from the reference line.

Table 3. Robot performance using gmove_simple command utilizing gyro sensor

gmove_simple (**speed**, rotations, gyro_port)		**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**
25, 5.5, 2	Front	3	0	14	6	2	1	11	-4	2	4
	Rear	4	-3	12	2	-3	0	7	-5	0	2
50, 5.5, 2	Front	-2	-9	7	-9	0	-5	-8	10	1	-4
	Rear	2	-8	7	-9	0	-5	-7	13	2	1
75, 5.5, 2	Front	6	1	-9	4	-9	-11	-3	-9	0	-14
	Rear	5	7	-9	-6	-15	-12	-2	-16	-3	-8
100, 5.5, 2	Front	-23	-14	-26	-21	-27	-40	-44	-43	-41	-40
	Rear	-17	-12	-29	-21	-19	-32	-34	-30	-30	-28

The visualization of Table 3 into Figure 11 shows a representation of the physical positions of the rover robot relative to the reference line. The dots represent various runs. Most of them are within the plus and minus 10 millimeters line from the reference.

At a speed of 100%, all the runs are better than without gyro control but did not meet the expected plus minus 10 millimeters deviation. Additional trials and errors in tuning the H formula did not improve the situation.

One consistent pattern observed when running at higher speeds is the rover robot sudden change in speed from zero at start to desired speed at almost instantaneously. This is causing vibration and wobble which is detected by the gyro and eventually is corrected along the way. However, the correction may be lagging. The higher the speed the faster it arrives at the destination with insufficient correction leading to larger deviation.

Higher speeds are also causing the rover robot to arrive with almost sudden stop at destination. The inertia pushes the robot further to the left, forward, or to the right depending where it is heading at arrival point. During this experiment, the rover robot is relatively lightweight. It did not carry any load nor additional attachments such as robot arms, extensions, or other objects. In a competition arena, the robot is loaded with more weight, and sudden starts and stops are likely to throw the robot further away when arriving at the destination point.

During the FLL robot game, the robot performs autonomously for 150 seconds to reach, move, collect objects at various locations trying to score as much as possible during the given duration. So, speed is a winning factor. This motivates participants to operate the robots at the highest speed possible.

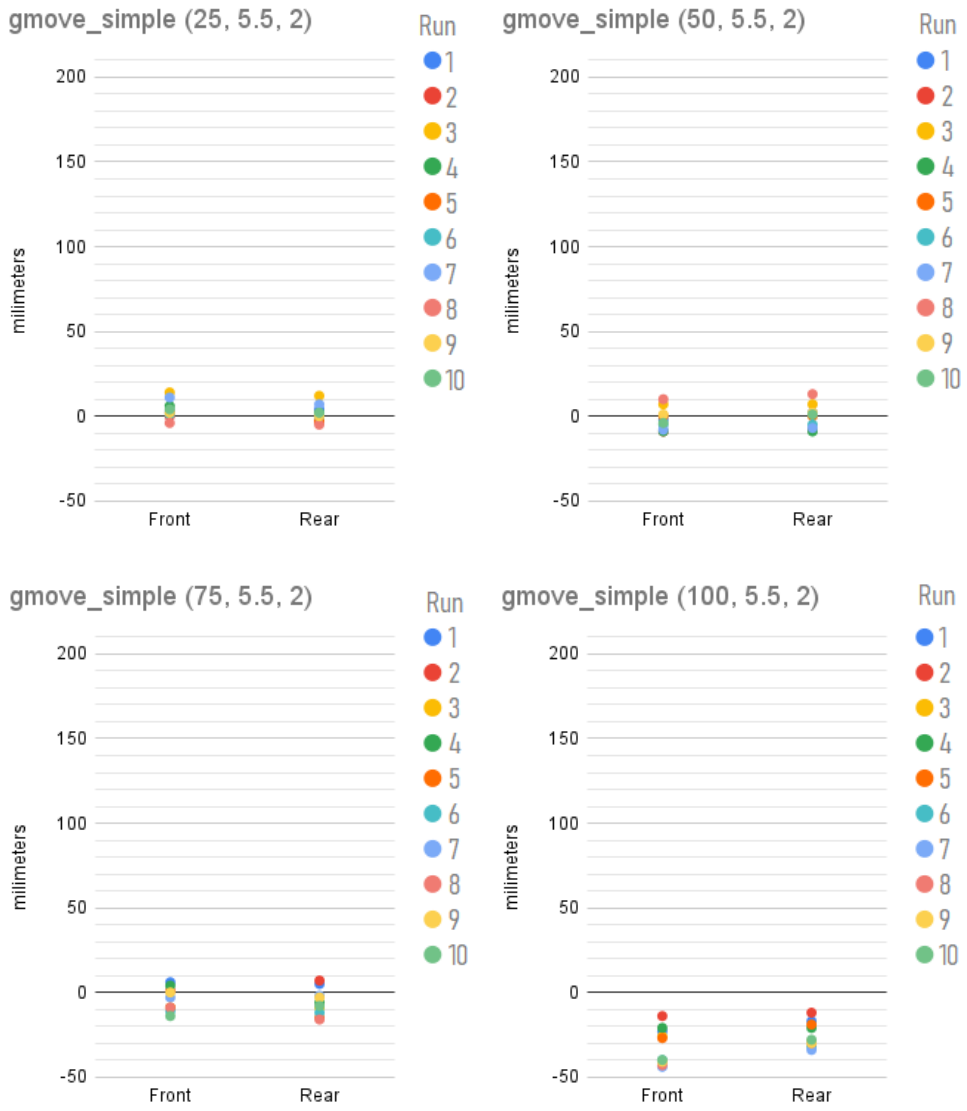


Figure 11. Robot performance using gmove_simple command utilizing gyro sensor

3.4 Improved GMove Command

The following robot performance experiment is inspired by the unsatisfactory results in running at higher speeds, in particular at 100% in the previous experiment. The following third experiment is an attempt to make further improvements in accuracy at higher speeds by adding acceleration and deceleration control at the start and the stop of the robot run.

The basic idea is instead of running the motor at sudden jump to a full speed at start, a gradual ramp up control is added. Similarly, instead of sudden stop from a full speed, a gradual ramp down control is added. For this, additional input parameters are added to the gmove algorithm,

fminspeed =	minimum speed (in fraction of the full speed)
fru =	ramp up from start point (in fraction of total distance between start and stop)
frd =	ramp down to stop point (in fraction of total distance between start and stop)

The fminspeed is a minimal sufficient amount to move the robot, must be greater than zero. Figure 12 illustrates the acceleration and deceleration control concept.

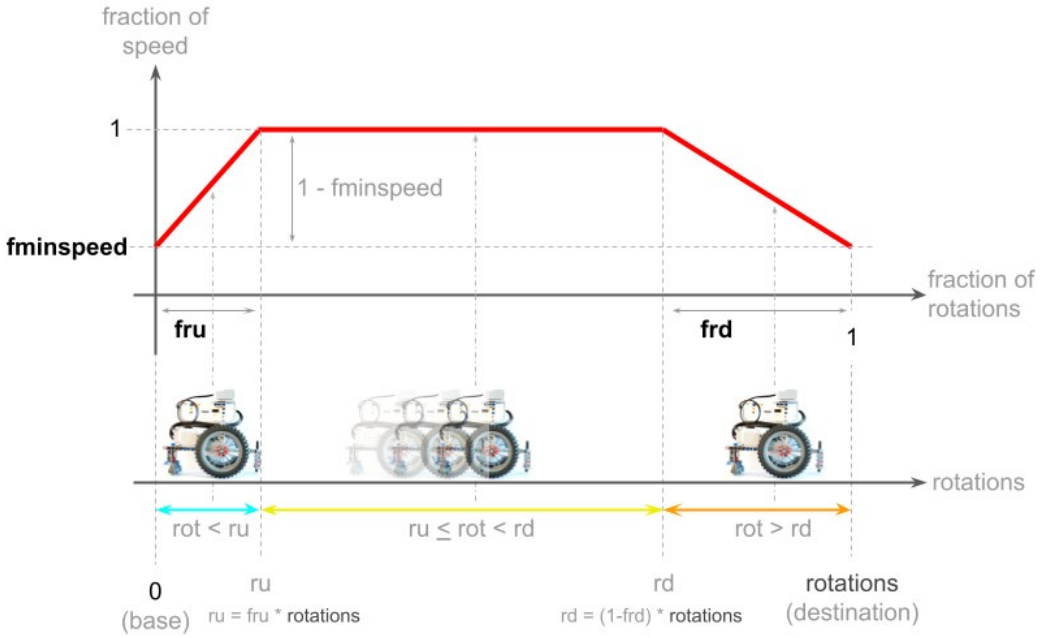


Figure 12. Speed acceleration and deceleration control

The gmove command will now have additional 3 inputs,
gmove (speed, rotations, gyro_port, **fminspeed**, **fru**, **frd**)

Example: gmove (50, 5.5, 2, 0.2, 0.25, 0.3)
speed = 50
rotations = 5.5 wheel rotations
fminspeed = 0.2 = 20% of speed
fru = 0.25 = 25% of rotations
frd = 0.3 = 30% of rotations

The speed calculation is as follows
rot = current rotation count (read from built-in EV3 rotation sensor) = current position of the rover robot

ru = fru * rotations = 0.25 * 5.5
rd = (1 - frd) * rotations = (1 - 0.3) * 5.5

When ru ≤ rot < rd, the robot runs at full speed, set the speed multiplier F to 1,
F = 1

When rot < ru, the robot runs at a fraction of speed directly proportional to
the ratio of current position rot over ru,
F = (rot / ru) * (1 - fminspeed) + fminspeed

When rot ≥ rd, the robot runs at a fraction of speed directly proportional to
the ratio of current position rot relative to total rotations over rotations minus rd,
F = ((rotations - rot) / (rotation - rd)) * (1 - fminspeed) + fminspeed

Use F as a multiplier to calculate the actual err, Bspeed, Cspeed as shown in Figure 13.

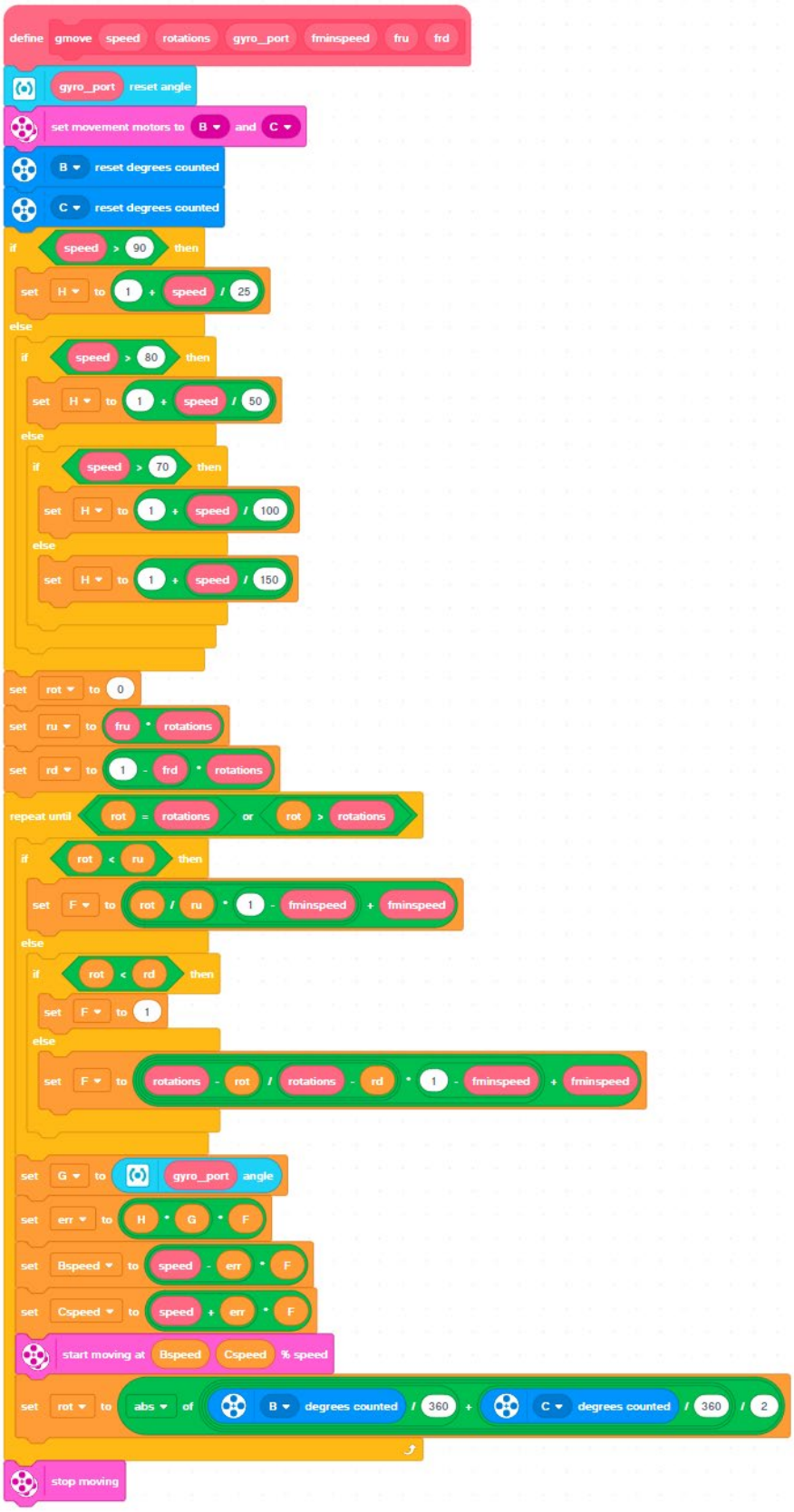


Figure 13. Improved gmove command with speed acceleration and deceleration control

Table 4. Robot performance using gmove command with acceleration and deceleration control

gmove (**speed**, rotations, gyro_port, **fminspeed**, fru, frd)		**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**
25, 5.5, 2, **0.4**, 0.25, 0.25	Front	0	14	14	2	9	-2	10	12	-4	15
	Rear	0	11	10	0	6	-2	8	9	-4	14
50, 5.5, 2, **0.3**, 0.25, 0.25	Front	10	9	6	1	4	6	11	2	13	6
	Rear	8	6	3	-2	2	2	7	0	9	4
75, 5.5, 2, **0.2**, 0.25, 0.25	Front	3	2	-1	-5	-7	11	-4	-9	-7	12
	Rear	0	-3	-4	-7	-8	8	-6	-12	-10	7
100, 5.5, 2, **0.1**, 0.25, 0.25	Front	-5	-18	-17	-31	-6	-16	-16	1	4	-7
	Rear	-8	-17	-18	-30	-9	-21	-17	-2	2	-10

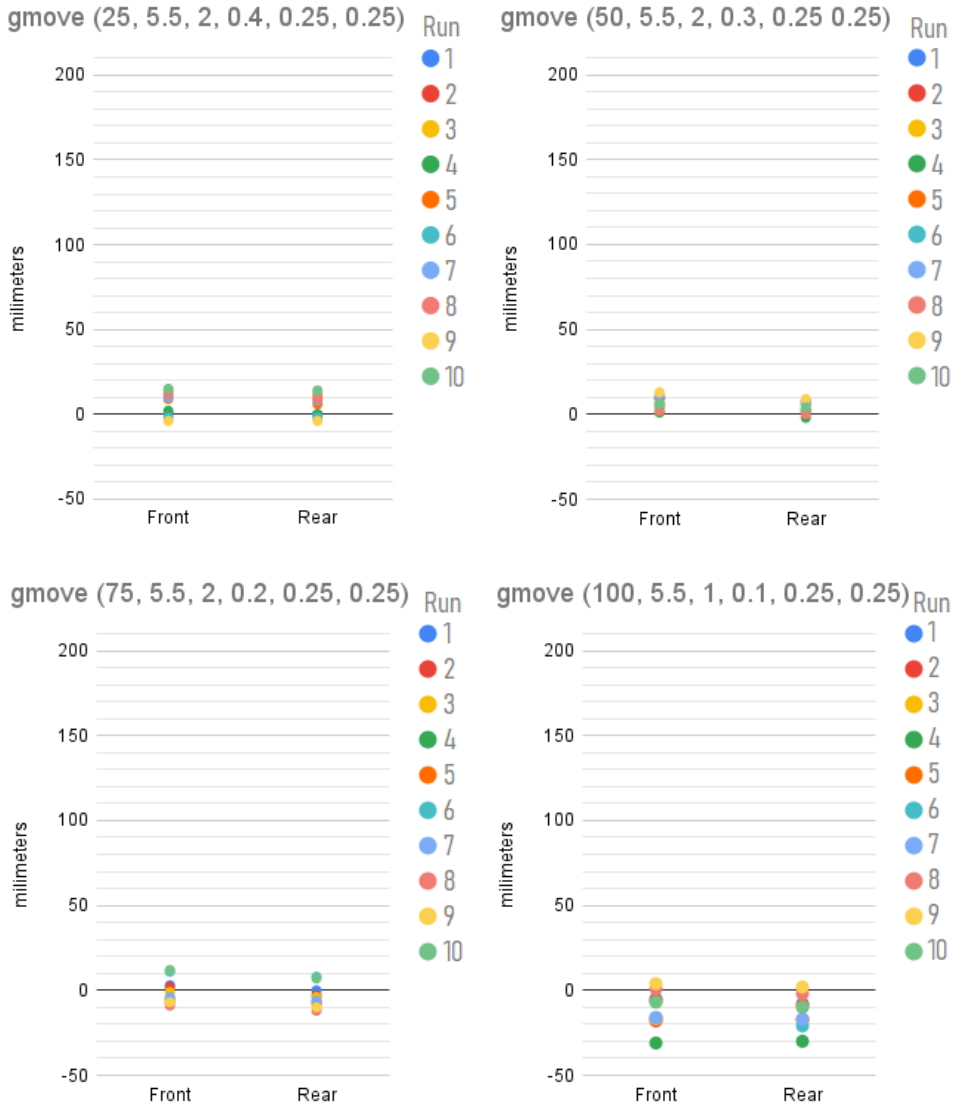


Figure 14. Robot performance using gmove command with acceleration and deceleration control

The third robot performance experiment results are shown in Table 4 and visualized in Figure 14. The acceleration and deceleration control added to gmove certainly improved the accuracy when running the rover robot at higher

speed. In particular, at speed 100%, more dots in Figure 14 are within the target plus minus 10 millimeters accuracy. A better result may still be possible by further fine tuning the fminspeed, fru, frd input parameters.

4. Conclusion

This project indeed discovered the possibilities of higher precision robot movement using non-industrial, educational Lego Mindstorms EV3. The use of Lego gyro sensors made it possible to control and keep the EV3 rover robot movement on target path. The implementation in gmove command and its unique approach in combining proportional gyro control with acceleration deceleration control have demonstrated significant improvements.

Within the limitation of FLL competition rule, the educational EV3 robot movement using gmove is sufficiently accurate where the robot is now capable to arrive at the destination point within plus minus 10 millimeters deviation. This not only meets the original goal of this research but also proven satisfactory when used in real world FLL robot game competition participated by the author, improved robot game scores, winning the robot performance award category.

Acknowledgements

The author would like to thank FLL team 23841 TechFusion for their enthusiasm to learn and their team spirit much needed by the author to make continuous improvements to the robot design and performance, for two consecutive seasons, leading to champion level at Razorback Invitational international arena.

References

Martin, F. G., Real Robots Don't Drive Straight: https://www.aaai.org/Papers/Symposia/Spring/2007/SS-07-09/SS07-09-020.pdf, September 2007

Maxim, Mathematical Model of Lego EV3 Motor: http://nxt-unroller.blogspot.com/2015/03/mathematical- model-of-lego-ev3-motor.html, March 2015

Hurbain, P., LEGO® 9V Technic Motors Compared Characteristics: https://www.philohome.com/motors/motorcomp.htm, undated

Hurbain, P., Wheels, Tyres and Traction: https://www.philohome.com/traction/traction.htm, undated

Biography

Raymond Yap is a Valley Christian High School student. He participated in FIRST Lego League competition for five consecutive years, successfully led his team for four consecutive years to win the Robot Design, Robot Performance Award, Champion Award from 2016 to 2019 at local and international arena. Raymond also participated heavily in the high school International Space Station Laboratory program, developing his engineering skills to assume roles as Mechanical Leader in his second year and Team Leader in his third year. He presented his experiment on the Effects of Copper Paint as an Antimicrobial in Microgravity at the American Society for Gravitational and Space Research (ASGSR) conference in Denver. He participated in Montessori Model United Nations (MMUN) and was elected to speak at the United Nations General Assembly in New York, USA in March 2017 on "Nuclear Disarmament and Prevention of An Arms Race In Outer Space" resolution. He is an advanced level 10 pianist and has received the Music Teachers' Association of California (MTAC) Branch Honors award twice in 2014 and 2015. He is a junior black belt in Aikido martial arts. Raymond is also an experienced MIT Scratch programmer, studying Python and Java, his research interests include biomedical engineering, digital music composition, and architecture.