System Analysis 3 Progress Report

Submitted to

Dr. Janiszewska

Prepared by

Nick Stassen,

Matthew Geiger,

Ben Bazan

Engineering 1182

The Ohio State University

College Of Engineering

Newark, OH

March 08, 2022

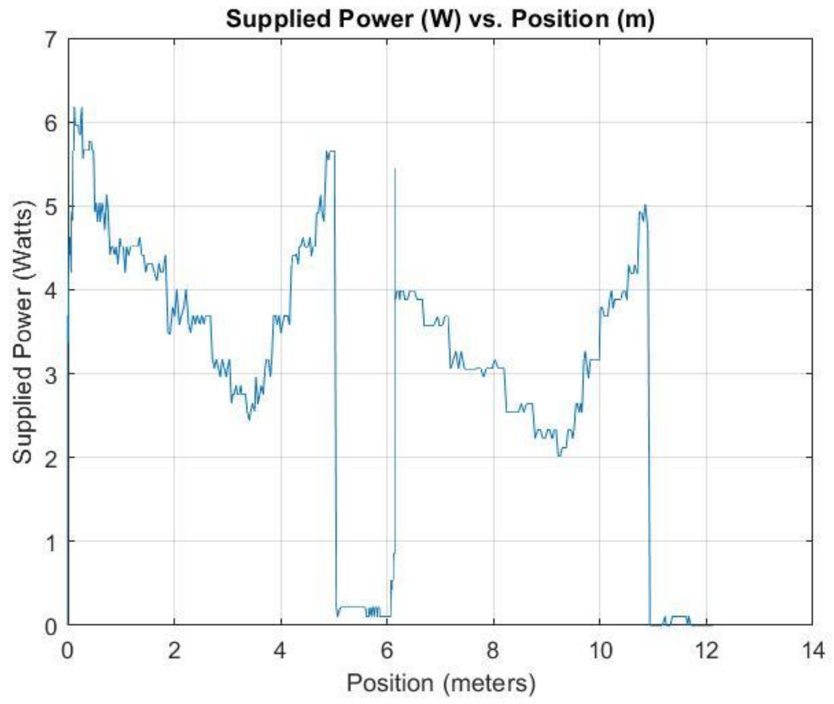
The results of this experiment demonstrate the capabilities of certain styles of coding. The celerate command in the AEV_Controller Arduino library lacked not only having a longer run-time but also indicated significant energy (see Table 1 in Appendix). However, the celerate command has its advantages: first, it allows for less stress on the motors, you can slowly increase the power supplied to allow the motors more time to accelerate. However, this command alone should not be used to control the speed of the AEV. Celerate, as seen in Table 1, used a total of 93 Joules in the flat track run. In comparison, the motorSpeed command, as seen in Table 2, only used 69 Joules. Additionally, the motorSpeed command code completed the track 9 seconds faster than the cellerate command code. The celerate command's purpose in future code will be restricted to large speed changes such as stopping and starting.

Promising energy usage can be seen with the motorSpeed function, meaning the motorSpeed command will be used in all cases besides the aforementioned cases. Another reason that cellerate is detrimental to the success of the AEV's success is that with large acceleration times the AEV can pass the position it was trying to go to. Since it has passed the goToAbsolutePosition value it will not proceed to the next statements and the motors will continue to run causing catastrophe. This means that the time values for the cellerate command must be small in order to not exceed a succeeding goToAboslutePosition statement. Figures 2 and 4 show the power vs time graphs for celerate and motorSpeed respectively. Celerate shows diagonal lines which represent the power increasing or decreasing during acceleration or deceleration, respectively. The motorSpeed graph shows instant steps in power representing instantaneous changes in motor power percentage.

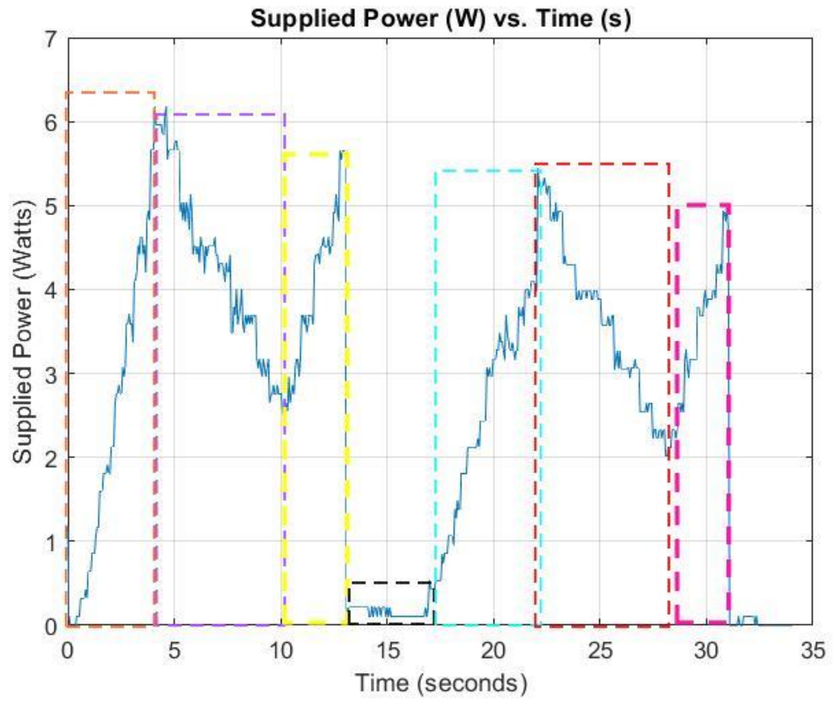
To program the half-track run, findings regarding motorSpeed were implemented. The bulk of the code was focused around using motorSpeed rather than celerate. This is because motorSpeed is more efficient than celerate as mentioned above. The celerate command was used for quick speed changes such as slowing down before the turn.

One of the issues that occurred during the flat track run cellerate command code was precision. Since the code used mostly cellerate commands rather than goToAbsoluteposition commands the lab group had to guess rather than know for certain where the AEV would be after a certain amount of time. This resulted in inaccurate positioning of the AEV during important times such as while the AEV is going around the curve. From this goToAbsolutePosition was found to be a necessity to complete the mission. From the half-track run the issue of the AEV being positioned 12 inches backward from the start of the 4 ft section arose. This issue if not fixed would cause the AEV to be 12 inches backward of the desired location. This would cause problems while going around the curve as well as stopping at necessary locations. The code was then changed to account for this issue. Another issue that arose during the half-track run was going down the slope from "the Grand Canyon", at first the AEV went down the track at a speed that was unsafe, this was due to the code inadequately accounting for the force of gravity. This was promptly changed by reducing the power supplied to the motors after the first trial.

Ben Bazan completed the issues and solutions section as well as individual portion for Ben Bazan (table 4). Nick Stassen completed discussion of results, information learned and then used for half-track run, and the individual portion for Nick Stassen. Matthew Geiger completed the MatLab code to convert the EEPROMM data into SI units and created all figures and tables (other than Nick stassen individual table and Ben Bazan individual table).



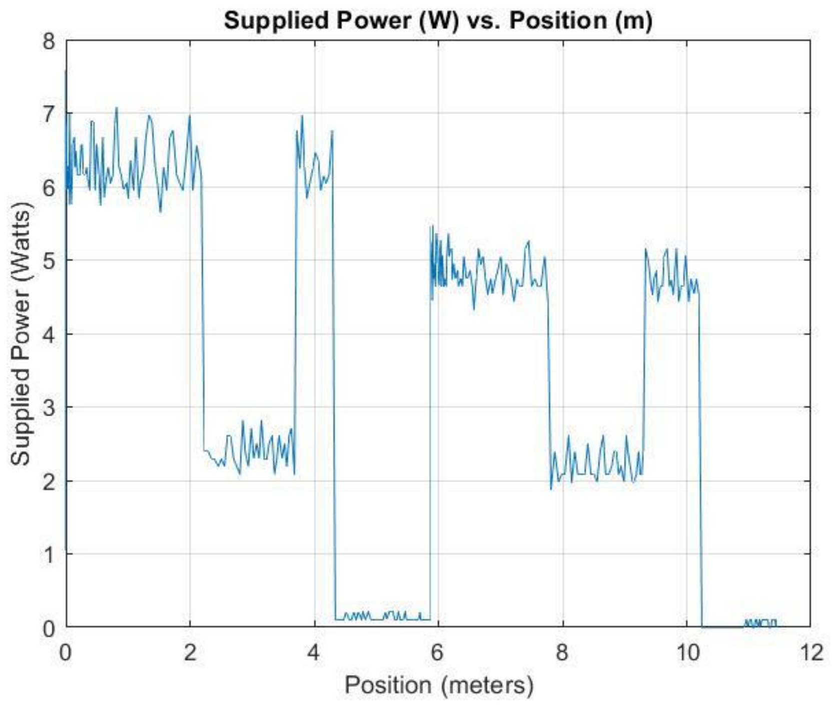
(Figure 1) describes the AEV flat track run, supplied power verses position, using cellerate commands.



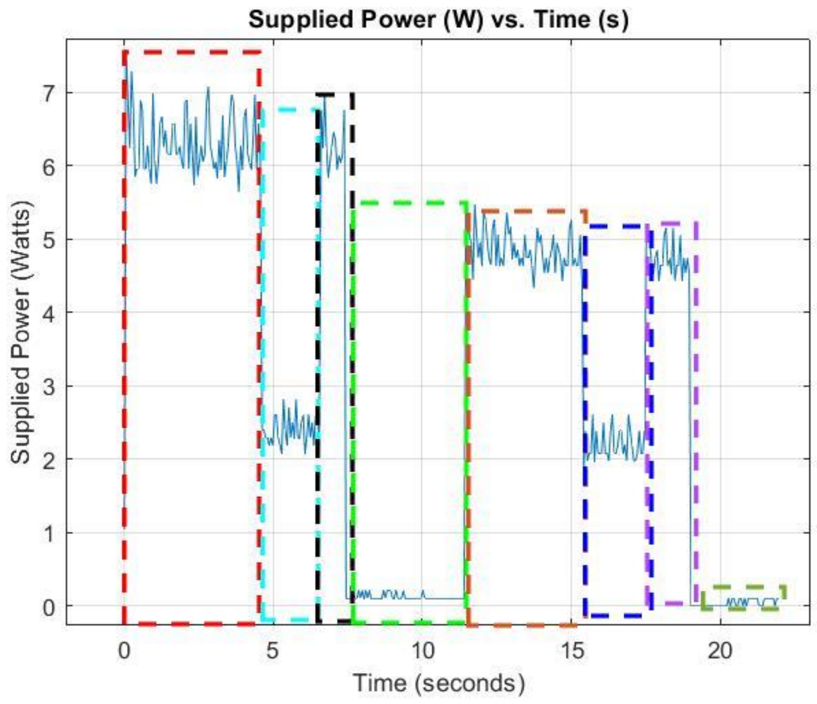
(Figure 2) describes the AEV flat track run, supplied power versus time, using celerate commands. Along with a phase breakdown.

Phase	Arduino Code	Time (seconds)	Total Energy (Joules)
1	celerate(4,0,22,4)	4.562	12.752
2	celerate(4,22,12,6)	5.641	23.303
3	celerate(2,12,22,3)	2.7	11.0299
4	brake(4)	3.72	1.41
5	celerate(4,0,22,6)	5.46	12.754
6	celerate(4,22,12,6)	6.06	22.3278
7	celerate(4,12,22,3)	2.82	9.853
Total		30.963	93.4297

(Table 1) describes the flat track run code using celerate. Total energy used is shown in the bottom row along with time.



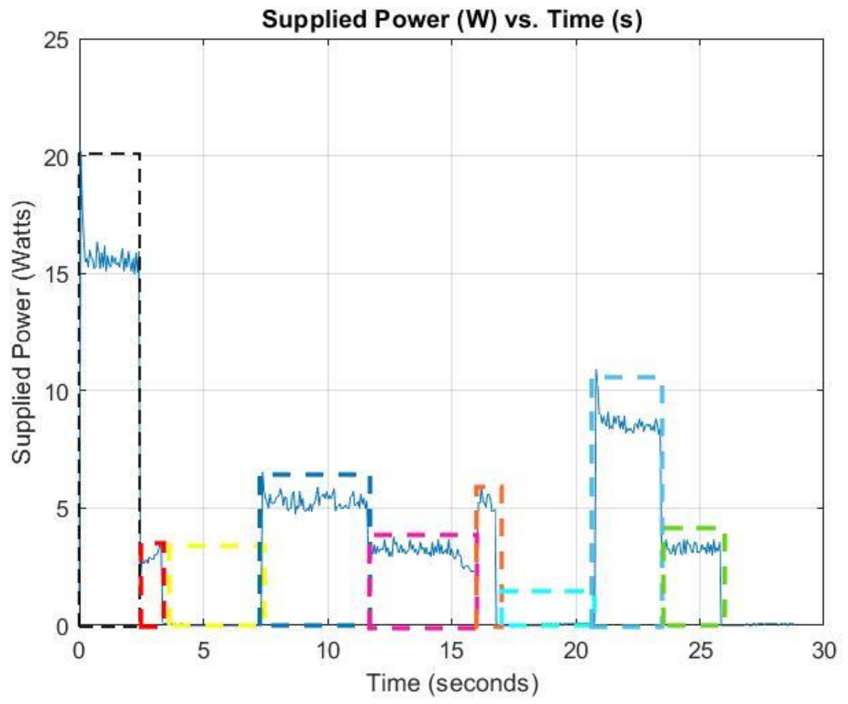
(Figure 3) describes the AEV flat track run, with supplied power versus position, using motorSpeed commands.



(Figure 4) describes the AEV flat track run, with supplied power versus time, using motorSpeed commands. Along with a phase break down.

Phase	Arduino Code	Time (seconds)	Total Energy (Joules)
1	motorSpeed(4,22)	4.7	26.917
2	motorSpeed(4,12)	1.3	3.299
3	motorSpeed(4,22)	1	4.6024
4	brake(4)	5	6.101
5	motorSpeed(4,20)	3.5	16.695
6	motorSpeed(4,13)	1.5	3.459
7	motorSpeed(4,20)	4	8.275
Total		21	69.3484

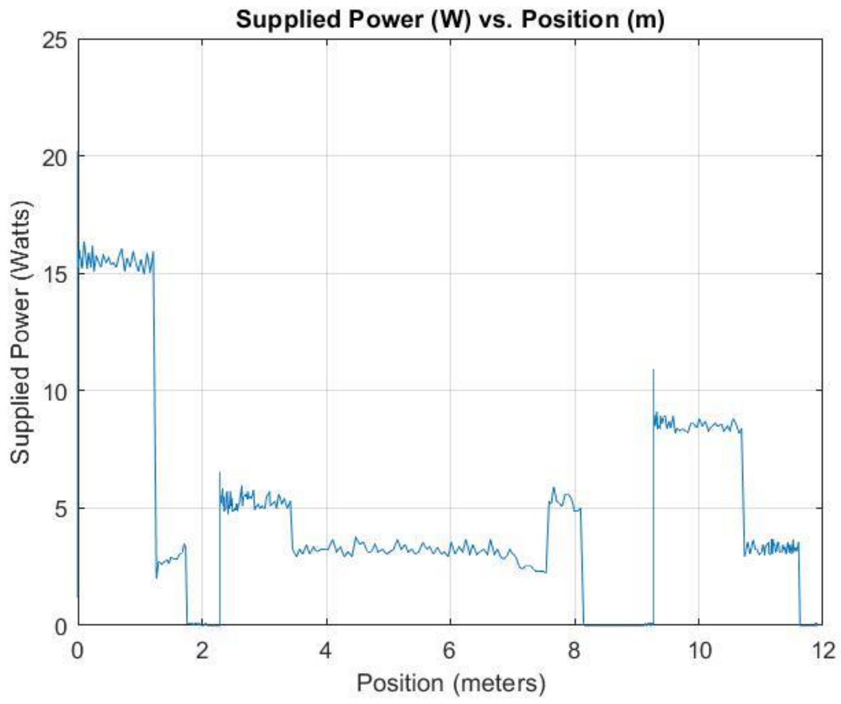
(Table 2) describes the flat track run arduino code phase breakdown. Total energy used is displayd in the bottom row along with time.



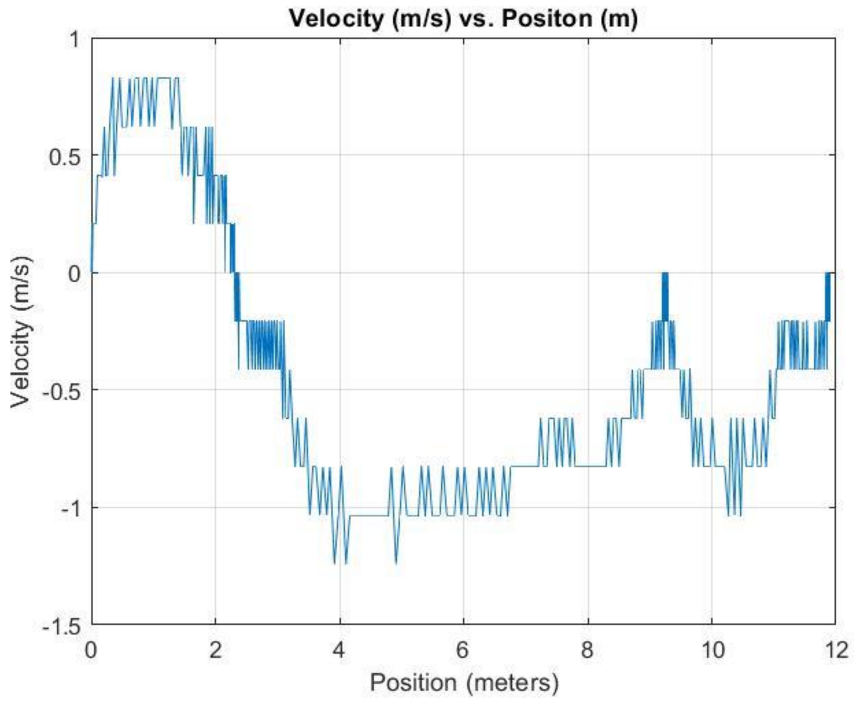
(Figure 5) describes the AEV half-track run supplied power versus time with phase breakdown.

Phase	Arduino Code	Time (seconds)	Energy per Phase (J)	Distance traveled (m)
1	motorSpeed(4,45)	2.401	37.86	0-1.26
2	motorSpeed(4,14)	0.901	3.0633	1.26-1.81
3	brake(4) goFor(4)	3.96	0.949	1.81-2.29
4	motorSpeed(4,22)	4.2	22.47	2.29-3.37
5	motorSpeed(4,16)	4.56	13.92	3.37-7.58
6	celerate(4,15,12,1) motor Speed(4,22)	0.72	3.64	7.58-8.65
7	brake(4) goFor(4)	4.02	0.492	8.65-9.28
8	motorSpeed(4,30)	2.64	23.115	9.28-10.65
9	motorSpeed(4,16)	9.06	30.894	10.65-11.91
		32.462	Total energy per kilogram 507.07	0-11.91

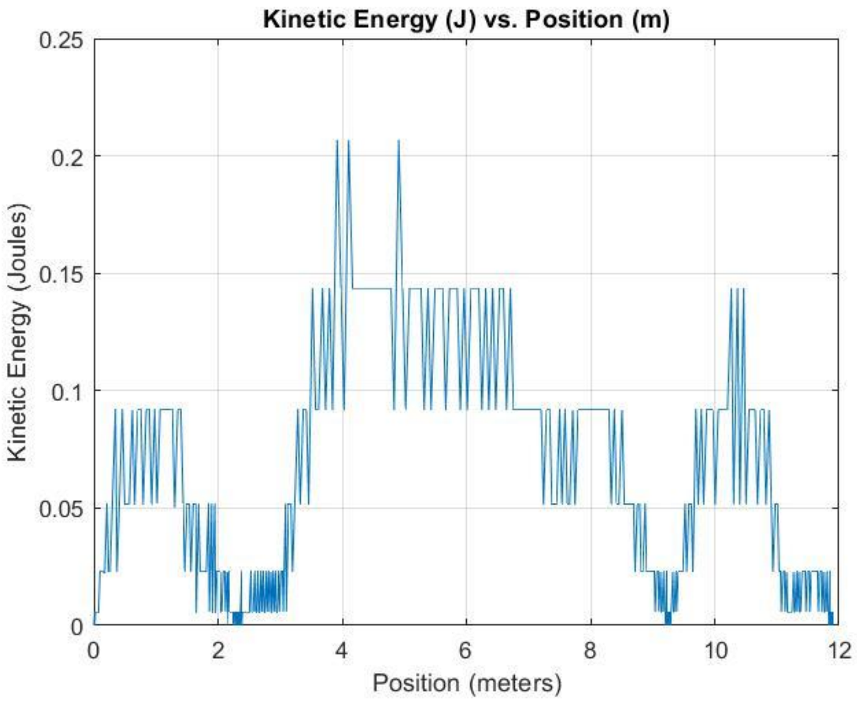
(Table 3) describes the half-track run Arduino code with phase breakdown. Total energy per kilogram used is displayed in the bottom row along with total time.



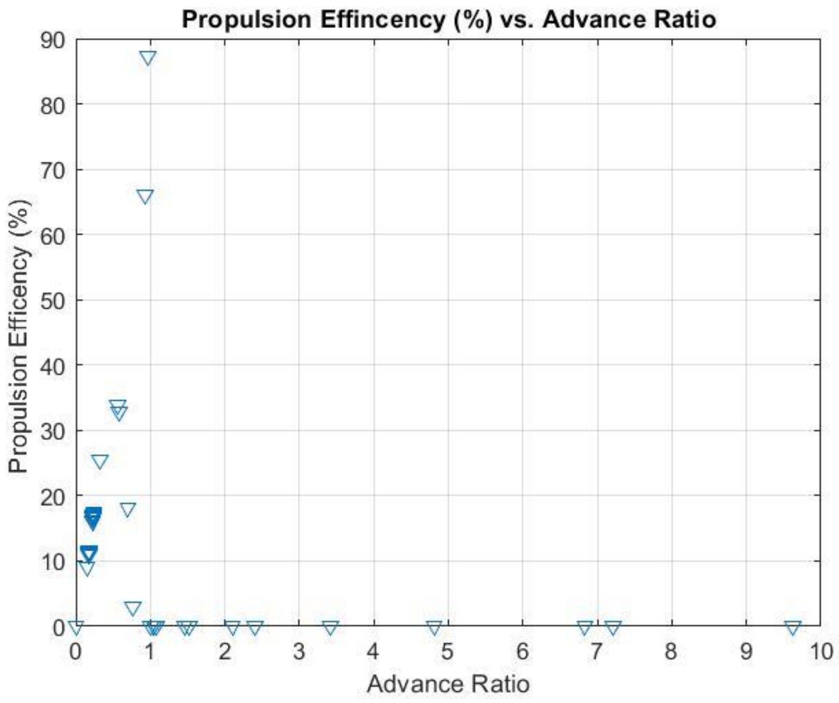
(Figure 6) describes the AEV half-track run with the supplied power versus position.



(Figure 7) displays the AEV half-track run with velocity versus position. Due to EEPROM being



(Figure 8) shows the AEV half-track run with Kinetic energy versus Position.



(Figure 9) presents the AEV half-track run with propulsion efficiency versus advance ratio.

			Ben Bazan sample calculations for flat track run at 3.002 s				
							p_j=5.95767
				V_r=2.46			p_{j+1}=6.049
	t_e=3002	marks=75	pos=-73	I_e=58	V_e=54		t_j=3.002
							t_{j+1}=3.062
	Time	Distance	Position	Current	Voltage	Supplied Power	Incremental Energy
Equation	$t=t_e/1000$	d=0.0124* marks	s=0.0124* pos	$I=(I_e/1024)*V_r*(1$ amp/0.185 volts)	$V=(15*V_e)/1024$	P=V*I	$E_j=(P_j+P_{j+i})/2*(t_{j+1}-t_j)$
Results	3.002	0.93	-0.9052	0.75316723	7.91015625	5.95767047	0.360205094

(Table 4) Ben Bazan sample calculations for flat track run at 3.002 seconds.

	Time	Distance	Relative position	Current	Voltage	Supplied Power	Incremental Energy
	t_e = 10023	marks = 460	relative marks = -458	I_e = 2	V_e = 553	(derived)	(derived)
Equation	t = t_e/1000	d = 0.0124 * marks	s = 0.0124 * relative marks	$I=(I_e/1024)*V_r*(1$ amp/0.185 volts)	$V=(15*V_e)/1024$	P = V * I	$E_j=(P_j+P_{j+i})/2*(t_{j+1}-t_j)$
Results	10.023	5.704	-5.6792	0.025971	8.10058	0.21038	0.009467

(Table 5) Nick Stassen sample calculations for flat track run at 10.023 seconds.

	Time	Distance	Relative position	Current	Voltage	Supplied Power	Incremental Energy
	t_e = 6543	marks = 297	relative marks = -458	I_e = 20	V_e = 548	(derived)	(derived)
Equation	t = t_e/1000	d = 0.0124 * marks	s = 0.0124 * relative marks	$I=(I_e/1024)*V_r*(1$ amp/0.185 volts)	$V=(15*V_e)/1024$	P = V * I	$E_j=(P_j+P_{j+i})/2*(t_{j+1}-t_j)$

Results	6.543	5.704	2.294	0.259713	8.027	2.0848	0.26517

(Table 6) Matthew Geiger sample calculations for flat track run at 6.58 seconds.