VIRGINIA TECH™

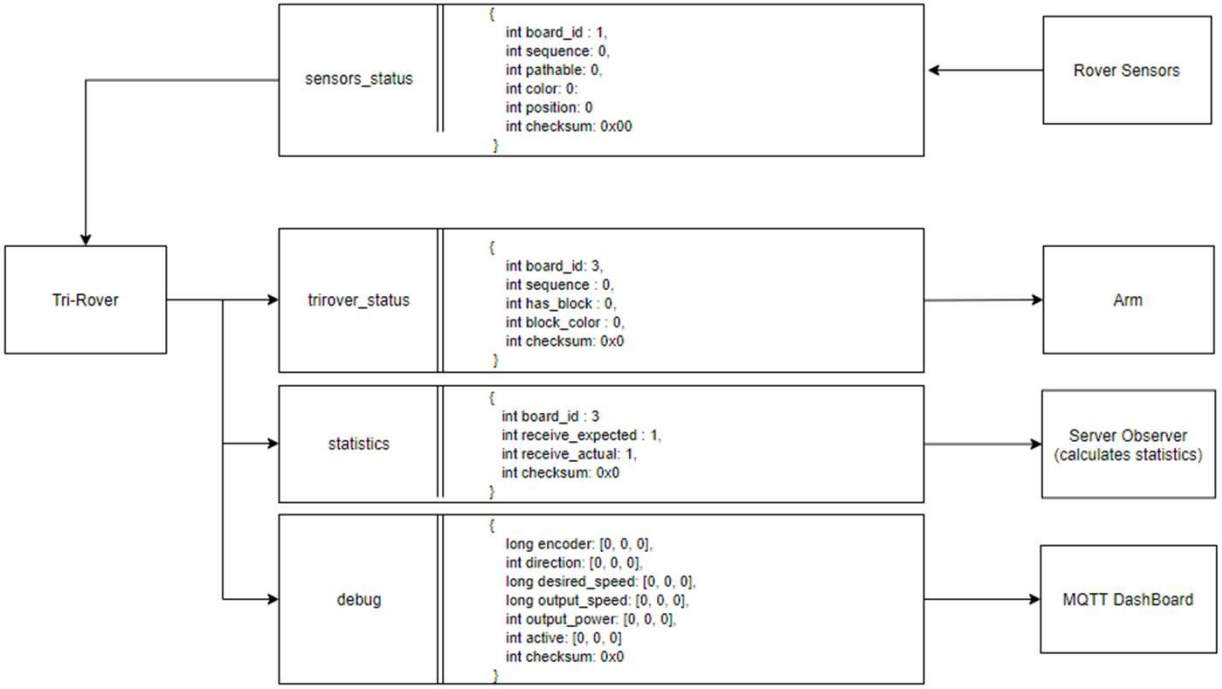
Team 15 Tri-Rover Demo

HUNG NGUYEN

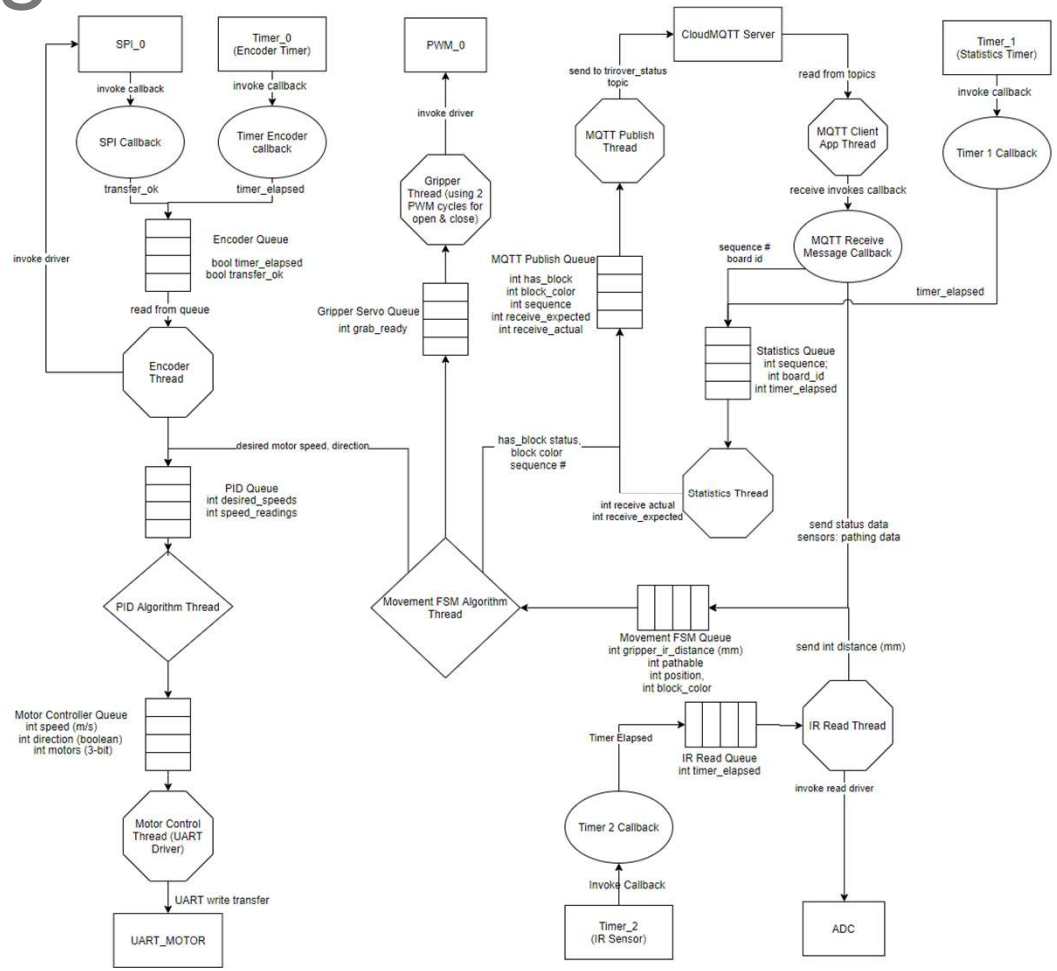
00

DIAGRAMS

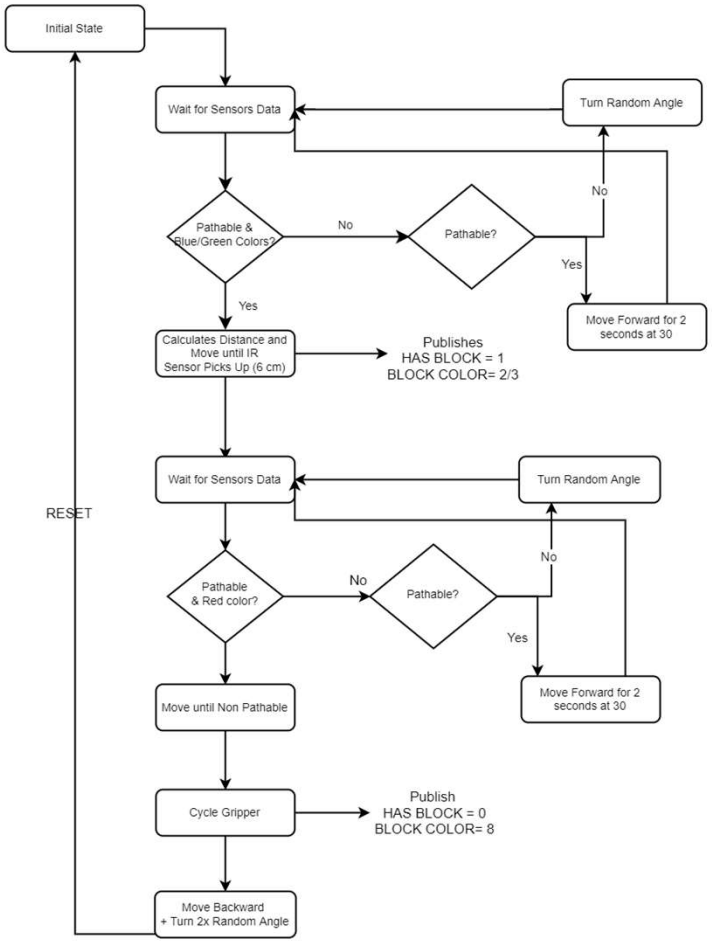
MQTT Messages Communication



Task Diagram



Movement Algorithm



01

TEST CASES

Test Case 0

1. Test when rover scans and is pathable, so it moves forward repeated (3 times)

2. Visual Verification: rover moves forward

3. MQTT format being sent to rover (red = 1, green = 2, blue = 3)
   { board_id: 1, sequence: 1, pathable: 0, color: 0, position: 1, checksum: 0x00 } MQTT expected from rover:

4. Expected sample output
   { board_id: 3, sequence: #, has_block: 0, block_color: 0, checksum: 0x00 }

Test Case 1

1. Test when rover scans and is not pathable (e.g. obstacle) to go forward, so it scans and and then is able to see path, and move forward

2. Visual Verification: rover should rotate and then move forward

3. MQTT format being sent to rover (red = 1, green = 2, blue = 3)
   { board_id: 1, sequence: 1, pathable: 0, color: 0, position: 1, checksum: 0x00 } MQTT expected from rover:

4. Expected sample output
   { board_id: 3, sequence: #, has_block: 0, block_color: 0, checksum: 0x00 }

Test Case 2

1. Test when rover scans and is not pathable, so it rotates, and then sees a block, so it moves forward to grab the block, and grips the block and stops. Should publish to MQTT with has block and block color

2. Visual Verification: rover should grab block at the end

3. MQTT format being sent to rover (red = 1, green = 2, blue = 3)
   { board_id: 1, sequence: 1, pathable: 0, color: 0, position: 1, checksum: 0x00 } MQTT expected from rover:

4. Expected output format
   { board_id: 3, sequence: #, has_block: 1, block_color: 0, checksum: 0x00 }

Test Case 3

1. Test when rover scans and is pathable, moves forward 3 times, then is unpathable, so it rotates and move forward again 3 times

2. Visual Verification: rover should rotate and then move forward repeatedly

3. MQTT format being sent to rover (red = 1, green = 2, blue = 3)
   { board_id: 1, sequence: 1, pathable: 0, color: 0, position: 1, checksum: 0x00 } MQTT expected from rover:

4. Expected output format

   { board_id: 3, sequence: 1, has_block: 0, block_color: 0, checksum: 0x00 }

Test Case 4

1. Test when rover scans and rotates, sees block, grabs it, and then move forward and rotate scan to find station, then move forward to station and drop off the block and retreat backward and rotate around.

2. Visual Verification: block should be dropped off and rover facing opposite of block

3. MQTT format being sent to rover (red = 1, green = 2, blue = 3)
   { board_id: 1, sequence: 1, pathable: 0, color: 0, position: 1, checksum: 0x00 } MQTT expected from rover:

4. Expected output format (has block and block color should output)
   { board_id: 3, sequence: 1, has_block: 0, block_color: 0, checksum: 0x00 }

Test Case 5

1. Test when rover scans and rotates, sees block, grabs it, and then move forward and rotate scan to find station, and doesn't find station for 2 scan rotate and forward move cycles.

2. Visual Verification: block should still be in gripper

3. MQTT format being sent to rover (red = 1, green = 2, blue = 3)
   { board_id: 1, sequence: 1, pathable: 0, color: 0, position: 1, checksum: 0x00 } MQTT expected from rover:

4. Expected output format (has block and block color should output)
   { board_id: 3, sequence: 1, has_block: 0, block_color: 0, checksum: 0x00 }

Test Case 6

1. Test when rover moves and grab block, drops it off, and is able to repeat the cycle again (grips two times and dropped off two times)

2. Visual Verification: block should be dropped off at different location each time

3. MQTT format being sent to rover (red = 1, green = 2, blue = 3)
   { board_id: 1, sequence: 1, pathable: 0, color: 0, position: 1, checksum: 0x00 } MQTT expected from rover:

4. Expected output format (has block and block color should output)
   { board_id: 3, sequence: 1, has_block: 0, block_color: 0, checksum: 0x00 }

02

SUPPORT FINDINGS

Encoder Reading Verification Steps

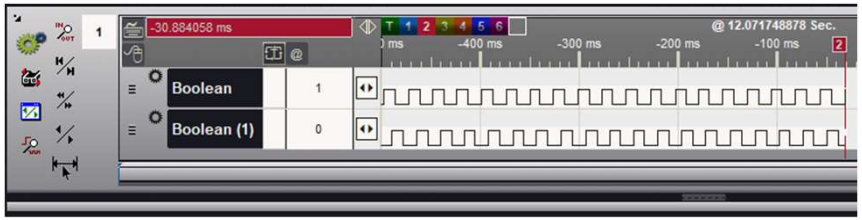
STEP 1	STEP 2	STEP 3	STEP 4
Waveform of encoders (U and G pins on the LS7366R) using Logic Analyzer	Average time deltas of each full pulse duration to get second per tick	Divide second per tick value to get ticks per second	Verify with TI Board's reading output on MQTT

Encoder Reading Verification Data

1. Digiview waveform for Motor 2 encoder for approximately 1 revolution (~12 seconds) at applied Motor Power of 10

2. Calculated average of time deltas from Digiview's exported data

3. MQTT Debug topic data received are very close to recorded logic analyzer data



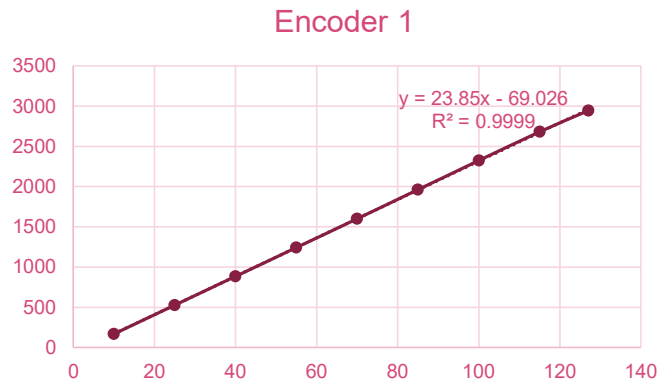
	A	B	C	D	E	F
1	Time (ns)	Boolean	Deltas (ns)		Average Nanosecond / Tick	
2	7,049,830	0			24,530,345.30	
3	20,999,410	1				
4	31,693,190	0	24,643,360		Average Second / Tick	
5	44,908,790	1	23,909,380		0.02453035	
6	54,982,860	0	23,289,670			
7	68,676,180	1	23,767,390		Average Ticks / Second	
8	79,594,420	0	24,611,560		40.77	
9	93,581,550	1	24,905,370			
10	105,098,380	0	25,503,960		Average x4 Ticks / Second	
11	118,396,700	1	24,815,150		163.06	
12	128,882,350	0	23,783,970			

```
> python .\debug.py
Start Running!
rc: 0
Subscribed: 1 QoS: (0,)
19:30:19 Encoders: 167, 164, 165 Desired: 150, 150, 150
19:30:20 Encoders: 167, 163, 165 Desired: 150, 150, 150
19:30:21 Encoders: 168, 163, 164 Desired: 150, 150, 150
19:30:23 Encoders: 167, 164, 164 Desired: 150, 150, 150
19:30:23 Encoders: 167, 163, 165 Desired: 150, 150, 150
19:30:24 Encoders: 166, 163, 165 Desired: 150, 150, 150
```

Converting Reading to Motor Output

1. Recorded Encoder Reading output for various motor power outputs (10, 30, 50, …, 127) (read period is 1 second)
2. Created Trendline equation for relating encoder reading to motor control power
3. Used y-intercept and slope in code (adjusted timescale of period)

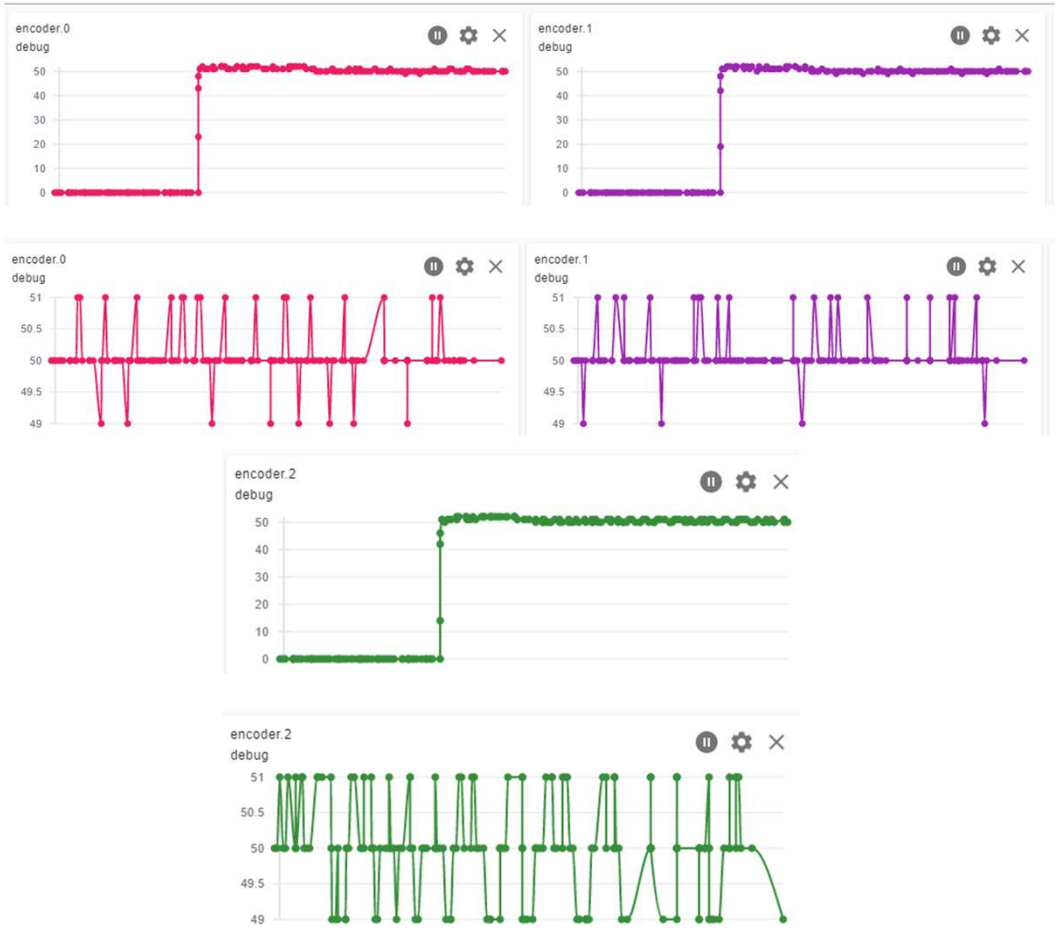
Motor Power	Encoder 1	Encoder 2	Encoder 3
10	168	163	164
25	526	519	515
40	883	874	867
55	1242	1232	1219
70	1602	1589	1571
85	1961	1950	1927
100	2324	2308	2279
115	2683	2670	2637
127	2944	2936	2899



PID Adjustment

1. Adjusted Kp and Ki parameters of PID equation while setting Kd=0
2. Adjusted Kp to undershoot slightly at the beginning
3. Adjusted Ki to stabilizes

(Software used: MQTT Explorer)



IR ADC Conversion

1. Calculated the ADC_convert() values at each inch value
2. Created Lookup Table for relating reading to distance in code
3. Created reading range to begin gripper claw grab

Distance (inch)	ADC_convert() reading	
	Min	Max
4.5	1282	1287
5	1151	1152
6	970	995
7	858	862
8	735	741
9	646	655
10	570	585
11	530	538
12	487	491

Reading when Block gripped (block can bend slight angle)			
1800	2016	2056	2165
Min			Max

Gripping Ready Reading			
Max		Min	Average
1898	1954	1995	1949

Gripper

1. Ran the PWM driver example code
2. Tested corresponding duty values (using fractional duty mode) that closes and open gripper
3. Adjusted duty tested incrementally for foam blocks grip