RGP Presentation (SEM2)

Jasdeep & Ronak

R

1. Clear technical presentation of your hardware design and software design.

- Used 30x30 grid space
- Bayesian and A* Pseudocode
- Pseudocode for groupUp(), and rotateTo() methods
- Simulation/VT (Virtual Testing)
- Made robot more compact than first Semester
- Summary of sensors

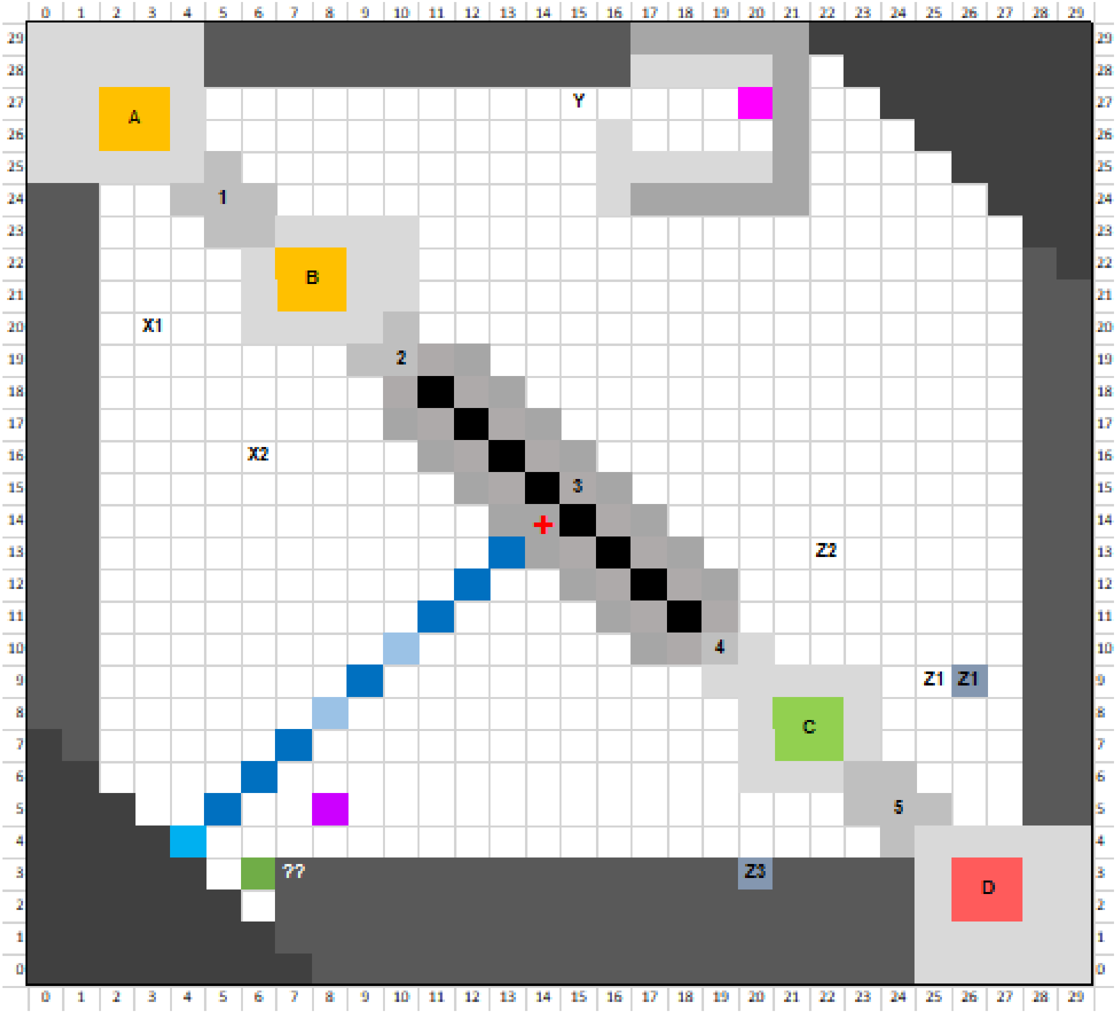
R

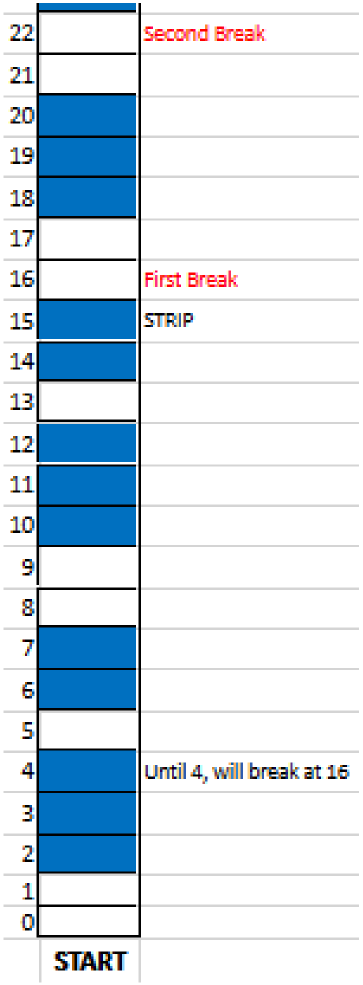
2. Evaluation results of both hardware and software designs

- A 30x30 grid space proved to be sufficient for this project.
- Old was 16x16 – not precise enough.
- Comprises of 40mm grids.
- Still managed to manoeuvre through obstacles.
- Processing the grid was not computationally complex, hence it did not slow the robot down (e.g. like a 100x100 grid would).
- Easy enough for us to manage for simulations/VT (virtual testing).

R

Grid Space + Bayesian Strip





J -> R

1. Clear technical presentation of your hardware design and software design.

- Used 30x30 grid space
- Bayesian and A* Pseudocode
- Pseudocode for groupUp(), and rotateTo() methods
- Simulation/VT (Virtual Testing)
- Made robot more compact than first Semester
- Summary of sensors

J

Bayesian Pseudocode

While threshold probability has not been reached
    Retrieve value from color sensor
    If color value is blue
        Update all blue positions with sensor true probability
        Update all white positions with sensor fail probability
    Else if color value is white
        Update all white positions with sensor true probability
        Update all blue positions with sensor fail probability
    Normalise all probabilities so that the sum of all probabilities is equal to 1 (mult = 1 / sum)
    If highest probability > threshold probability
        Exit loop, Bayesian terminated
    Else
        Update all positions with move true probability and move false probability
        Move robot one position forward

J

2. Evaluation results of both hardware and software designs

- Sense, normalize, move and repeat or end
- Base implementation was similar to the slides.
- Quantified localization strip for the robot to use. 37 points.
- Created an ArrayList for all BLUE Bayesian squares.
- Iterate through all 37 squares and check color.
- If color matches a BLUE position update the probability with sensorTrue constant. Else update with sensorFalse constant
- If position is not in the BLUE ArrayList it is WHITE.
- Every 5 steps rotateTo(0) is called for alignment.

R

A* Pseudocode

Set start position

Set end position

While target point has not been reached

    Find neighbours of current position which are not in the open list, the closed list, or the obstacle list

    Add the found neighbours to the open list and assign the current position as their parent

    If target position is in the open list

        A* Terminated

    Else

        Find position with lowest heuristic cost (Manhattan) in the open list [t.x – c.x]

        Remove current position from open list

        Add current position to closed list

        Current position is now position with lowest heuristic cost in the open list

While current position != start position

    Add current position to path

    current position = parent of current position

Add start position to path

Reverse path

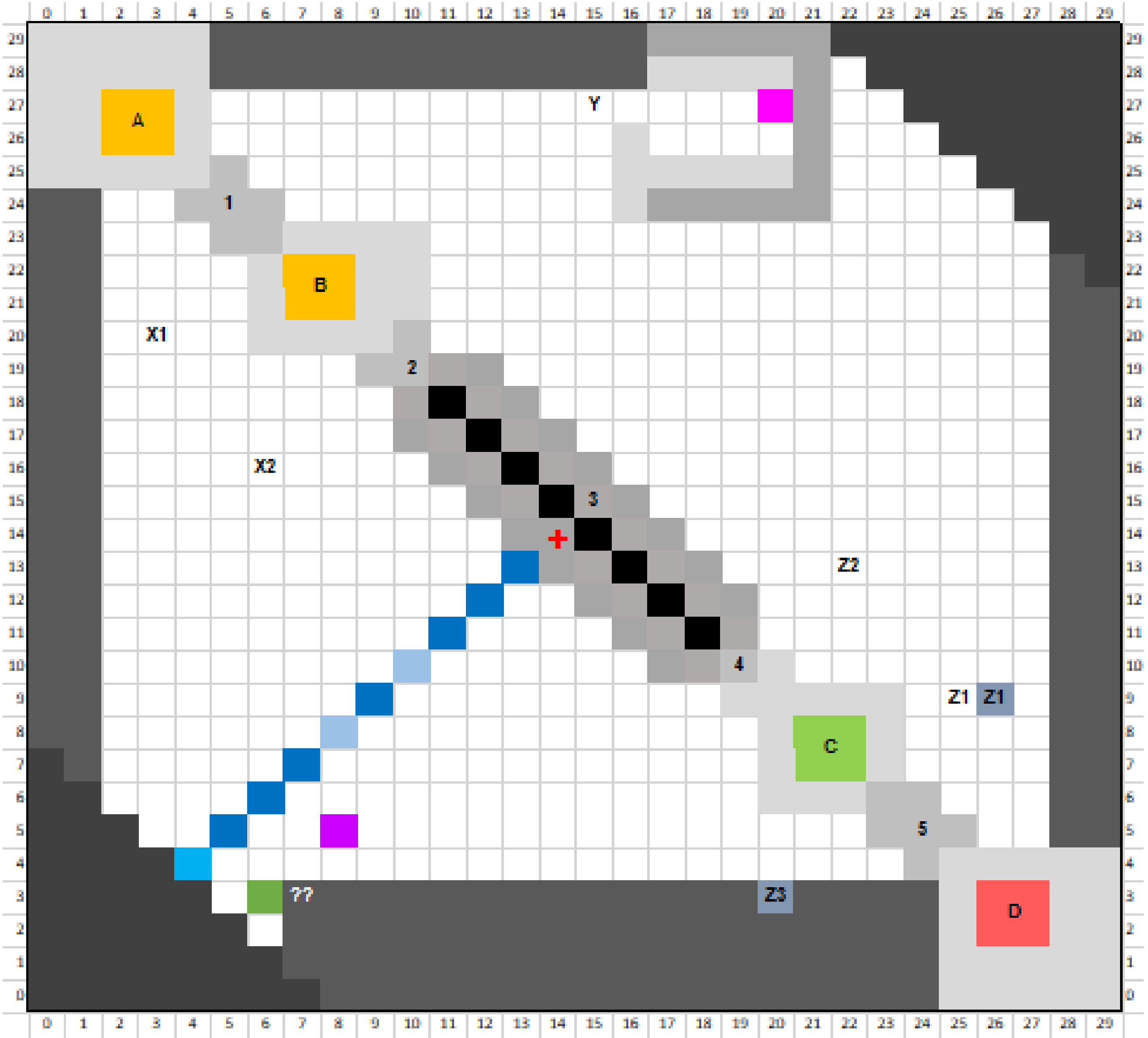
R

2. Evaluation results of both hardware and software designs

- A* implementation was also similar to the slides.
- Each grid has a Pair object (int x, int y).
- 30x30 implemented with a nested ArrayList, not double coords Array (Java double Array and generics).
- Path is created at the end by backtracking to the current node's parent (HashMap, current->parent), up until the start node – add the start node and reverse.
- Manhattan heuristic was used (easiest).

R

Grid Space



J -> R

1. Clear technical presentation of your hardware design and software design.

- Used 30x30 grid space
- Bayesian and A* Pseudocode
- Pseudocode for groupUp(), and rotateTo() methods
- Simulation/VT (Virtual Testing)
- Made robot more compact than first Semester
- Summary of sensors

J

rotateTo() Pseudocode

```
If (target – gyroSample > 0)
        While (gyroSample < target)
                rotate anti-clockwise
else
        While (gyroSample > target)
                rotate clockwise
```

J

2. Evaluation results of both hardware and software designs

- Instead of rotating the robot with a fixed movement, we supply rotateTo() a target angle and the method slowly moves the robot in the direction and stops once the robot is aligned with the target.
- Method is called before every movement action, this means there is very little error as it terminates once the Gyro detects the target.
- Bayesian gyroAlign() -> rotateTo(0)
- Also this method compensates for any physical imperfections on the Arena, e.g. bumps and tape peeling off.

R

groupUp() Pseudocode

```
Loop through path;
        calculate difference in coordinates
        If direction is North-East
                If not same direction
                        change group
                Assign direction
                Increment direction
```

R

2. Evaluation results of both hardware and software designs

- Using this and the movement method speeds up the robot’s movements, as it is completing similar moves in a more grouped fashion.
- This also came in handy when debugging the program and VT.
- Used with move(), switch statement - first is direction, second is iterations.

R

1. Clear technical presentation of your hardware design and software design.

- Used 30x30 grid space
- Bayesian and A* Pseudocode
- Pseudocode for groupUp(), and rotateTo() methods
- Simulation/VT (Virtual Testing)
- Made robot more compact than first Semester
- Summary of sensors

R

VT Bayesian and A* results

Bayesian

```
13: 8.38198526183127E-5
14: 1.5540827587020163E-4
15: 0.005624899912709918
16: 0.9851253491098716
17: 1.3638809033761738E-9
18: 7.687076823031477E-9
19: 1.8465943441649505E-7
20: 1.1468680552942978E-6
21: 8.381994564509496E-5
22: 7.309257731822031E-5
23: 1.599813746201117E-4
24: 0.0027288792212357897
25: 1.3725284304587803E-9
26: 2.2639289160565595E-8
27: 1.58543980508125E-8
28: 5.58679493413833E-7
29: 9.746412622078724E-5
30: 2.3202725315222627E-9
31: 5.002725270865447E-9
32: 1.144702183632083E-7
33: 7.323755380209872E-7
34: 8.381989386015795E-5
35: 1.5540827650196819E-4
36: 0.005624899915586859
sum= 0.9999999999999999
-------------------

Actual Index: 16
Current Postision: 16
```

A*

```
Grouped Path:
(0,0)
(4,3)
(3,7)
(4,4)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)
(0,0)

MOVE :: South _ 3 times
MOVE :: South-East _ 7 times
MOVE :: South _ 4 times
```

R

2. Evaluation results of both hardware and software designs

- Virtual Testing (VT) helped us a lot throughout this project.
- It allowed us to test that our program was logically correct without physically testing and needing the robot or Arena.
- It saved us a tremendous amount of time while debugging logic errors without needing to actually test it on the Arena (rebooting the brick).
- Isn’t affected by wheel jerk and Gyro inaccuracy.
- Helped us plan certain paths in different scenarios much faster, e.g. obstacle placements.

J

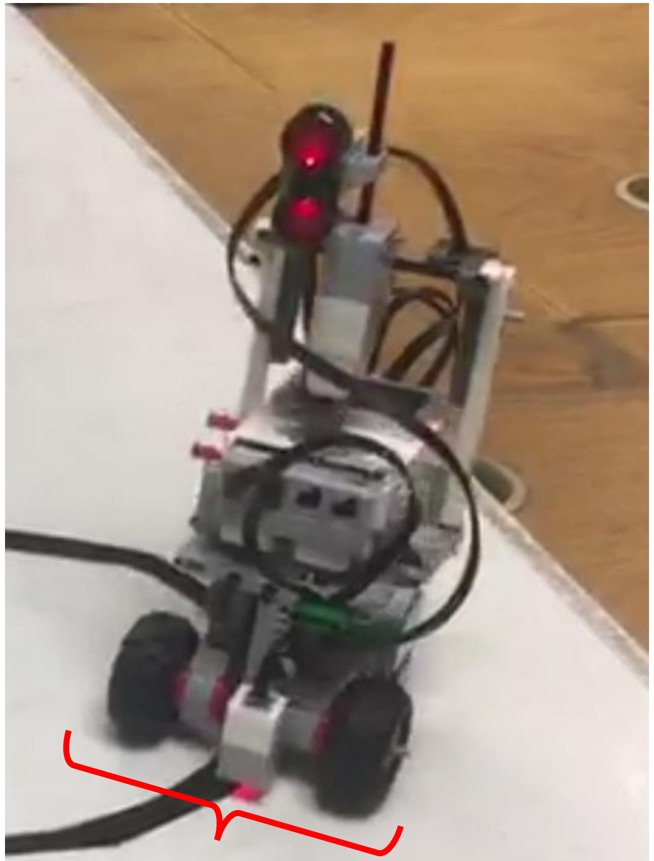
1. Clear technical presentation of your hardware design and software design.

- Used 30x30 grid space
- Bayesian and A* Pseudocode
- Pseudocode for groupUp(), and rotateTo() methods
- Simulation/VT (Virtual Testing)
- Made robot more compact than first Semester
- Summary of sensors

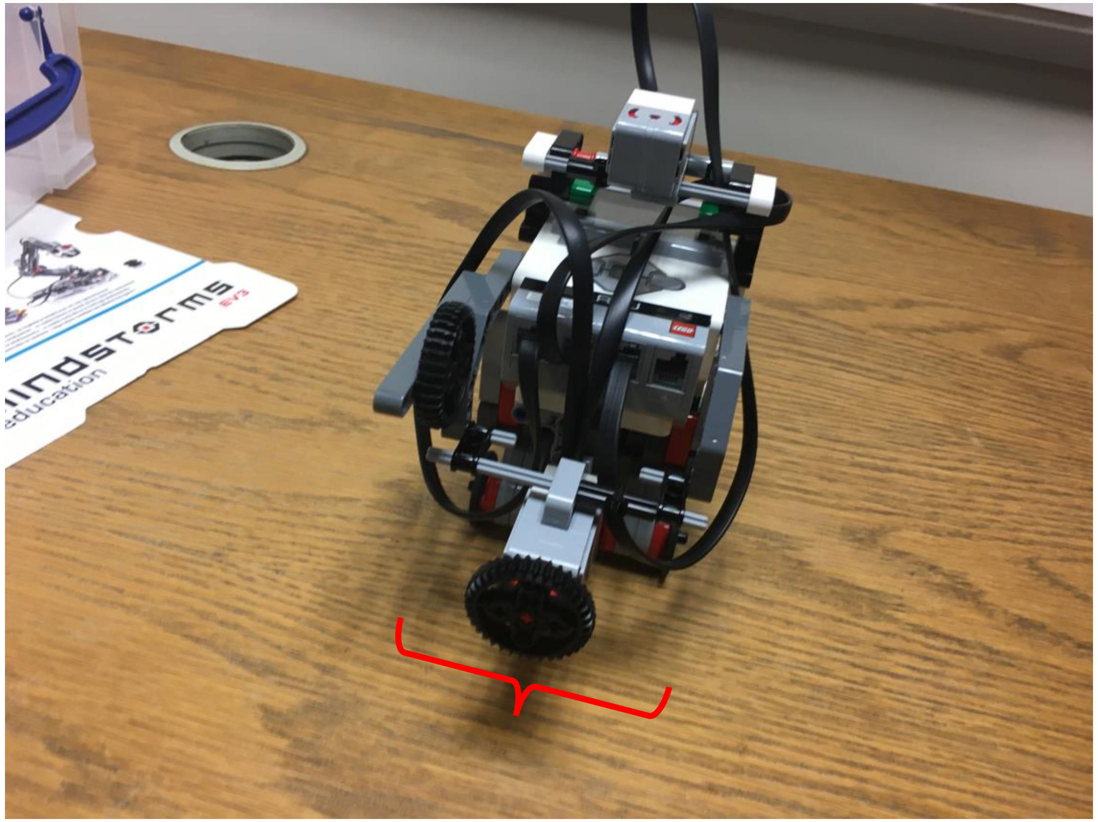
J

Robot image comparison

SEM1



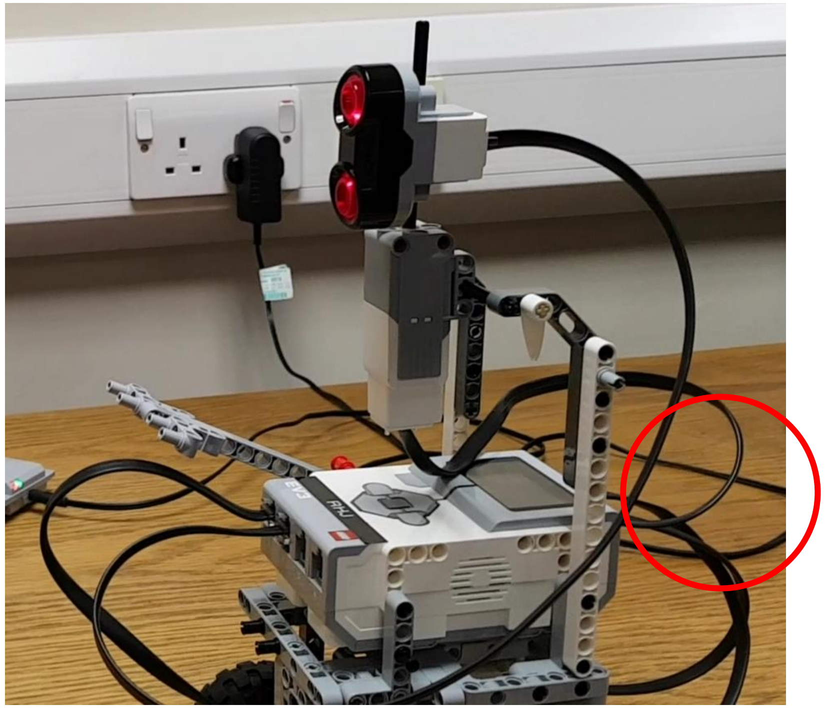
SEM2



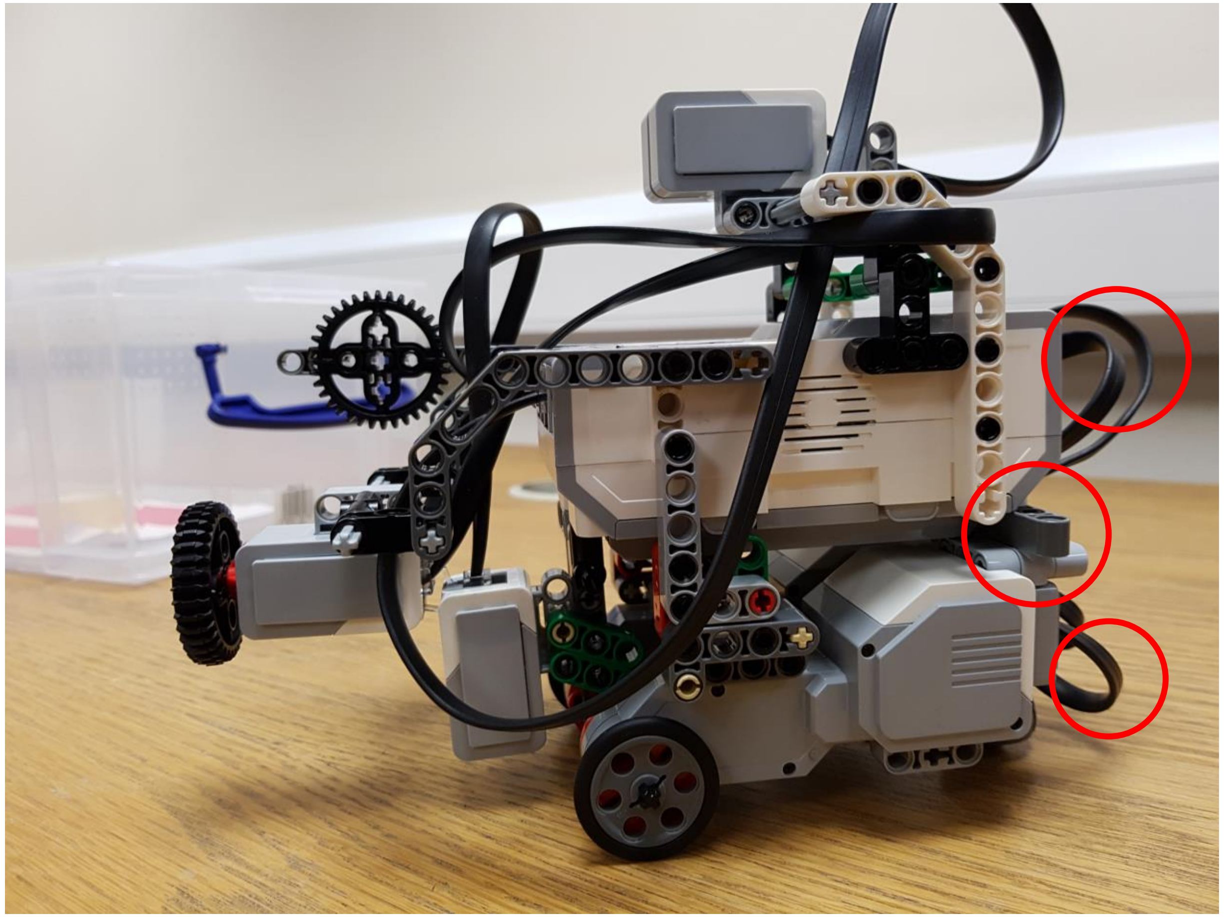
J

Robot image comparison

SEM1



SEM2 (week 10)



J

2. Evaluation results of both hardware and software designs

- A more compact robot allows for better manoeuvrability through obstacles and the Arena.
- This means we can afford to be a little less accurate with movement values, e.g. parking bay.
- Easy design for disassembly (e.g. swap batteries).
- Reinforced gyro sensor position on robot
- Previous main design iteration had a higher touch sensor, and untucked wires.
- However we couldn’t tuck in the back motor wires with just one piece, added another piece to hold it in place

R

1. Clear technical presentation of your hardware design and software design.

- Used 30x30 grid space
- Bayesian and A* Pseudocode
- Pseudocode for groupUp(), and rotateTo() methods
- Simulation/VT (Virtual Testing)
- Made robot more compact than first Semester
- Summary of sensors

R

Sensor Summary

- Color: Used for Bayesian, ColorID mode used to distinguish blue and white. Used in A* for parking bay to recognise obstacle color.
- Gyro: Used in Bayesian to align to strip if needed. Used in A* through rotateTo() method before every move to guarantee angle. Can be inaccurate and unreliable.
- Touch: Used in A* to exit out of Parking Bay and detect color, also when terminating program.

Positions (image)

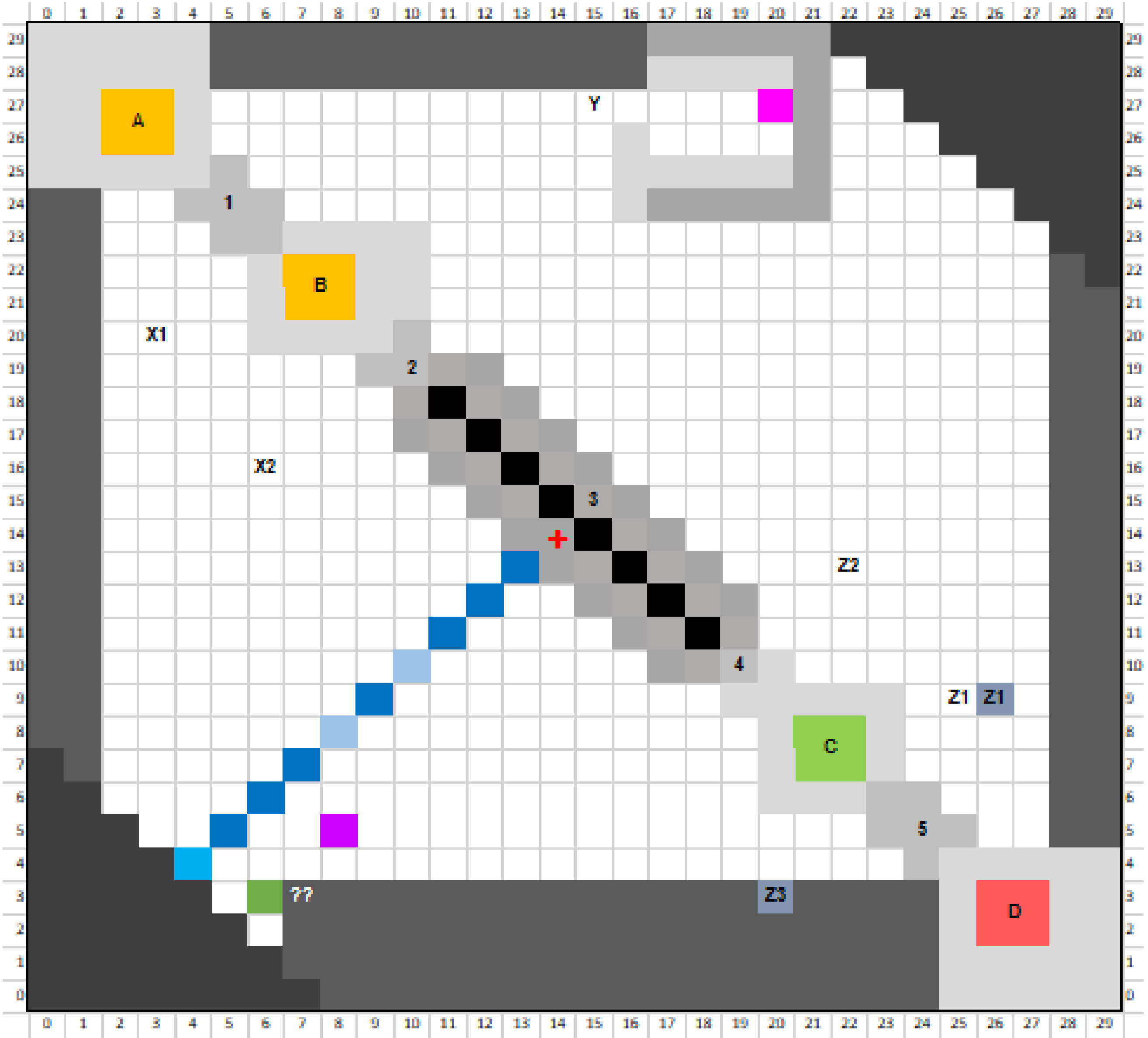
R

Method Summary

bayesian()
path_plan()
neighbours()
fillWalls()
fillObstacleA()
fillObstacleB()
fillObstacleC()
fillObstacleD()
fillWall1()
fillWall2()
fillWall4()
fillWall5()
groupUpPath()
move()/move2()
rotateTo()
bayesianMove()
moveStraight()
moveDiagonal()
park()
reverse()
end()
main()

R

Grid Space



J

2. Evaluation results of both hardware and software designs

Iteration Number

Expected/Theoretical values

Results for different rotation values

Diagonal	Target									
0	0	?	?	?	220	215	215	215	215	215
1	24.4	24.8	24.7	24.6	24.2	24.4	24.4	24.3	24.3	24.3
2	18.8	20	19.7	19.5	18.5	18.9	18.8	18.7	18.7	18.7
3	13.2	15	14.5	14.4	12.9	13.3	13.2	--	--	13.3
4	7.6	10.2	9.5	9.2	7.1	7.8	7.6	--	--	7.8
5	2	5.2	4.5	4.1	1.5	2.2	2	--	--	2.1

Bayesian	Target	65	65	68	68	68
0	30	0	0	0	0	0
1	28.3	28.3	28	28.2	28.1	28.3
2	26.6	16.7	26.6	26.5	26.5	26.6
3	24.9	25.1	25	24.9	24	25
4	23.2	23.4	23.5	23.2	23.2	23.2
5	21.5	21.9	21.8	21.5	21.6	21.5
6	19.8	20.3	20.2	19.8	19.9	19.8
7	18.1	18.3	18.7	18.1	18.2	18.2
8	16.4	--	17.1	16.5	16.5	16.5
9	14.7	15.5	15.6	14.9	14.9	14.9
10	13	13.9	13.9	13.1	13.2	131
11	11.3				--	11.4
12	9.6				9.7	9.7
13	7.9				8	8
14	6.2				6.2	6.3
15	4.5				4.6	4.6
16	2.8				2.9	2.9
17	1.1				1.2	1.2

R

2. Evaluation results of both hardware and software designs

- We tested components independently/individually straight after implementation, mainly using VT.
- We tried to perfect these models as much as possible before performing fully integrated testing in the last two labs (more on 'milestones').

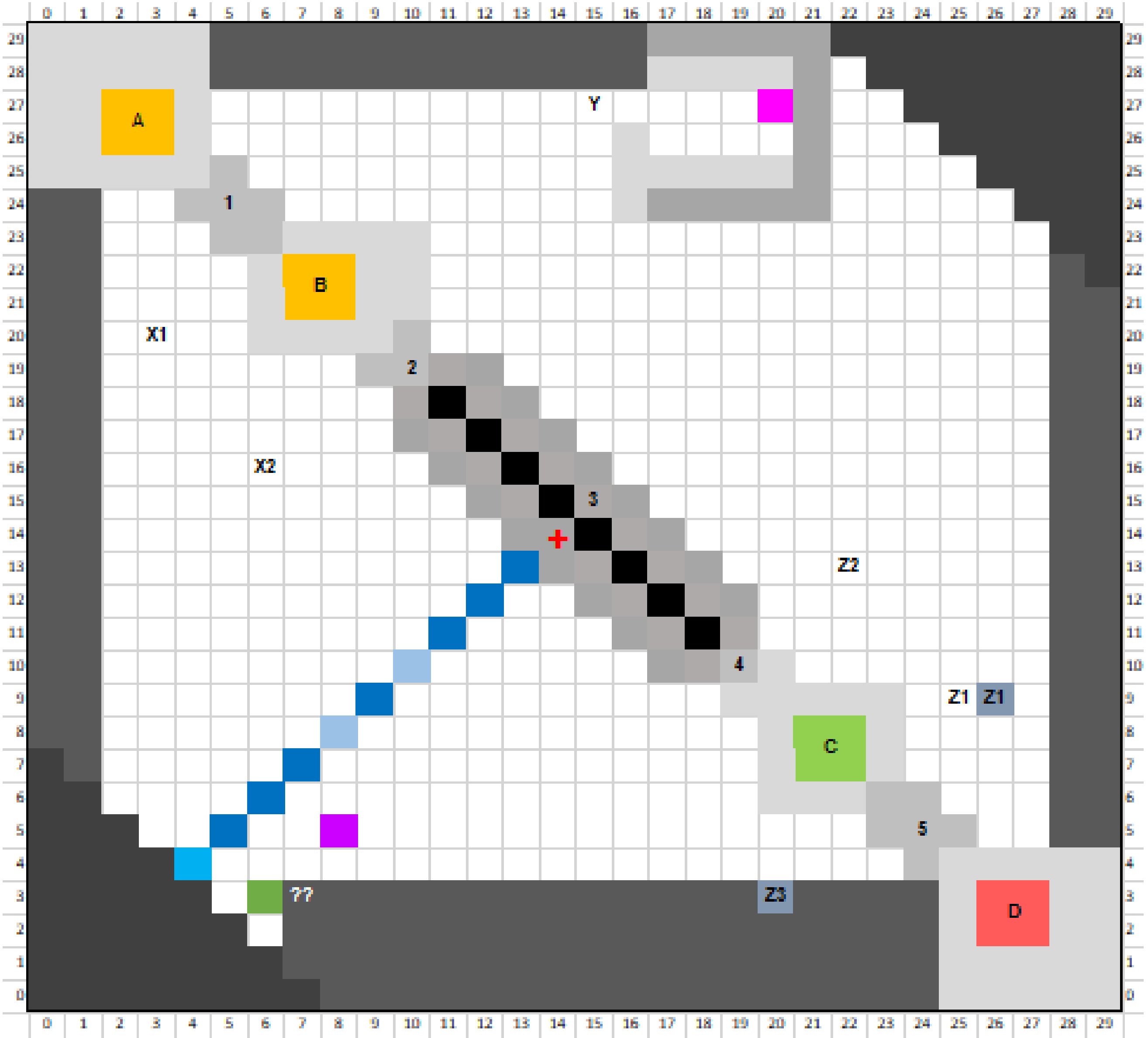
J -> R

3. Team management, including the project milestones and the contributions of members

1. Redesign Robot from SEM1
2. Complete Open Loop tests
3. Bayesian implementation + VT
4. Bayesian improvement;
    - 17mm value test
    - Align
5. A* implementation + VT
6. Bayesian testing
7. A* improvement;
    - groupUp()
    - 40mm and 56mm value tests
    - Designate waypoints
    - Parking bay methods
    - Extra padding
8. A* testing
9. Integrated testing

(In order of Labs)

Grid Space



J -> R

3. Team management, including the project milestones and the contributions of members

Jasdeep

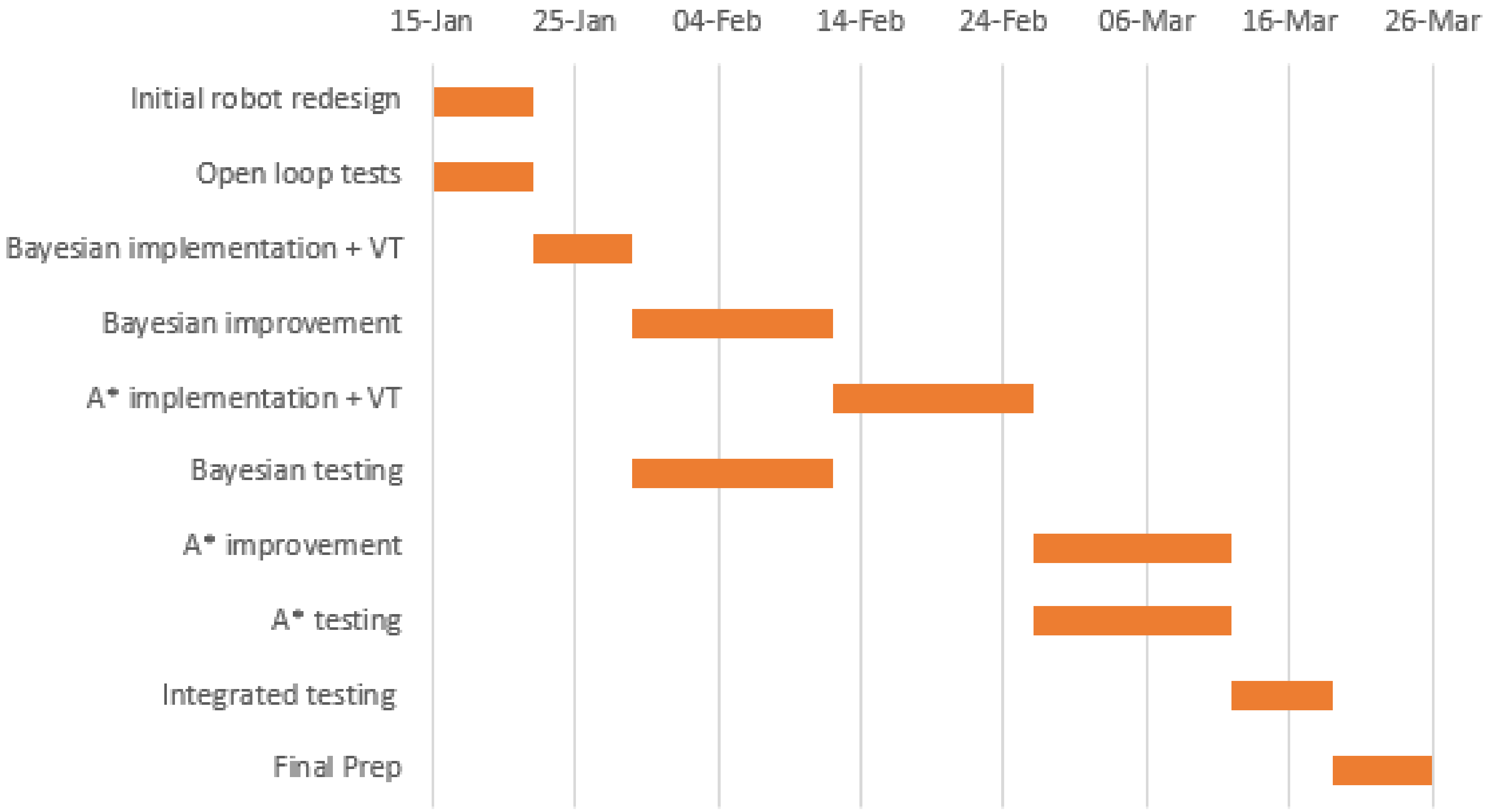
- Redesign robot from SEM1
- Complete open loop tests (both)
- Bayesian implementation + VT
- Distance tuning (both)
- GyroAlign() and rotateTo()
- Parking bay methods (both)
- A* implementation + VT (both)
- A* testing (both)
- Integrated testing (both)

Ronak

- Complete open loop tests (both)
- Bayesian testing
- A* implementation + VT (both)
- Distance tuning (both)
- Parking bay methods (both)
- A* groupUp() improvement
- Initialise grid – waypoints, obstacles & padding
- A* testing (both)
- Integrated testing (both)

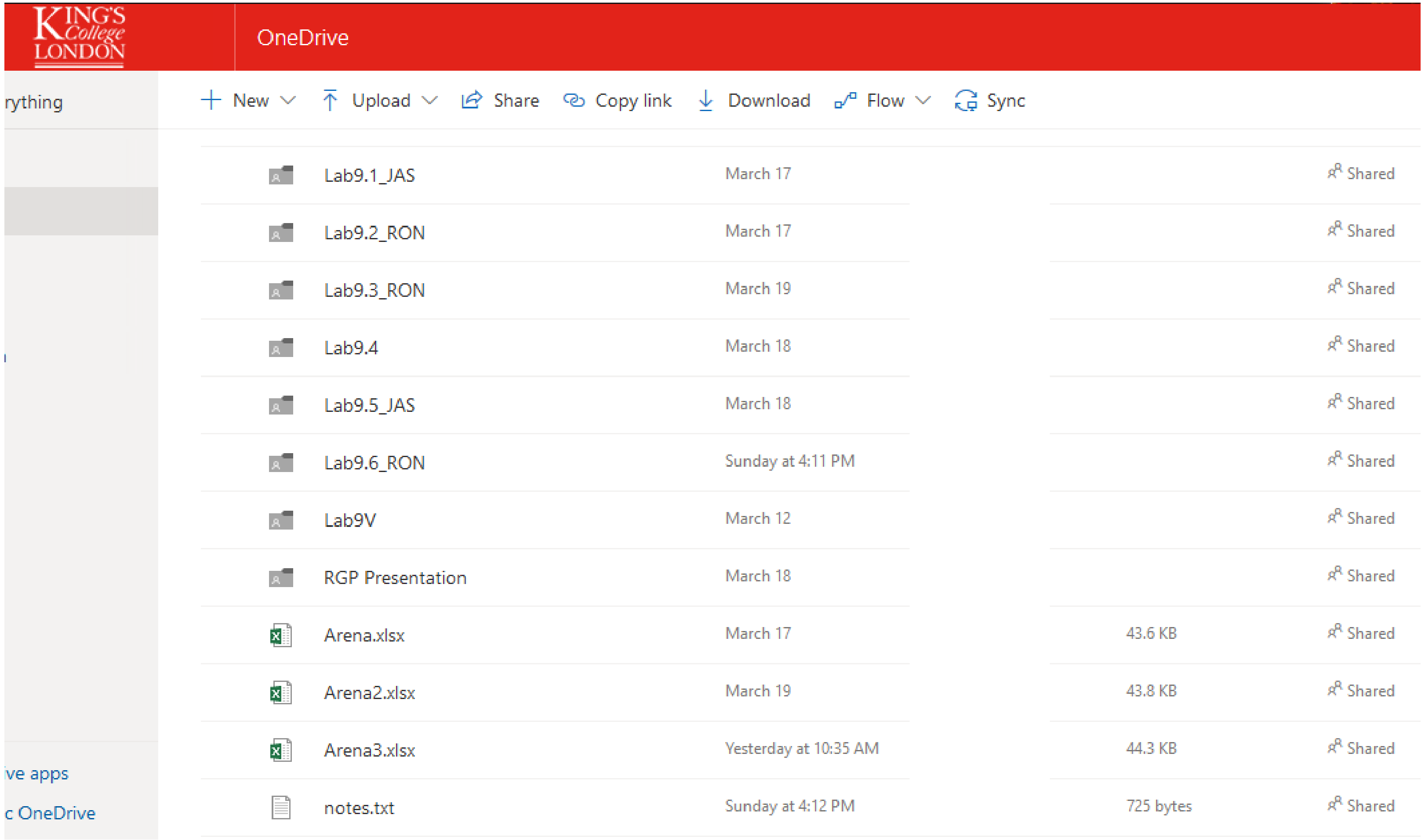
R

3. Team management, including the project milestones and the contributions of members



R

3. Team management, including the project milestones and the contributions of members



J

4. Conclusion and comments on places can be improved if more time was given

- Potential Field: More accurate, faster and fluid movement.
- Faster Movements: Tuned the rotation values for faster robot movement
- Movement tuning: More accurate values and introduce better offsets.
- More compact robot: touch sensor inside more, back wires tucked in – less risk of touching.
- Heuristics: Try Diagonal/Euclidian.
- Lego threading: Better understanding of Lego motor threading (random movements) issues, REMEDY prints.
- Larger grid and tuning intervals for it: Complexity/time wouldn’t increase drastically, but more precise – upgraded hardware to process extra complexity.