AMR SS2023
TEAM 3

- **Software Requirement Specifications**

Software Components
- Ros 2
- Foxy
- Py-trees
- Robile_gazeebo

Tools:
- Visual studio code
- Github

Hardware Components

Device	Type	Label	Comments
Intel NUC	PC	MAS NUC 3	
1 x Hokuyo UST-10LX	2D Laser	BRSU-Hokuyo -UST-10LX-1 & -2	
TL-WR902AC AC750 Wireless Travel Router	Router	FREDDY ROUTER	
8-Port Gigabit Unmanaged Desktop Switch DGS-108	Switch	FREDDY SWITCH	

Wireless Logitech Joystick F710	Joystick	FREDDY PLATFORM JOYSTICK	
Mean Well H LG-480H-30A battery charger	Battery Charger	FREDDY BATTERY CHARGER	Charger is cust om-modified for Freddy by KEL O-Robotics. Note: if the robot is initially off when the charger is connected, the charging will not start; it will start only when the robot is on.
4 x KELO SmartWheel (S/N 200034, 214005, 214006, 214007)	Wheel Unit	MAS SMARTWHEEL 200034 & 214005 & 214006 & 214007	
2 x Accurat Traction T40 LFP 24V LiFePO4 Lithium 40 Ah Battery	Battery	FREDDY BATTERY 1 & 2	
IDEC YW-E01 E Stop	Emergency Stop Button for Platform	KELO ESTOP	This switch only disables SmartWheel units (makes them passive), but it does not cut the power to and com munication.

Platform Assembly

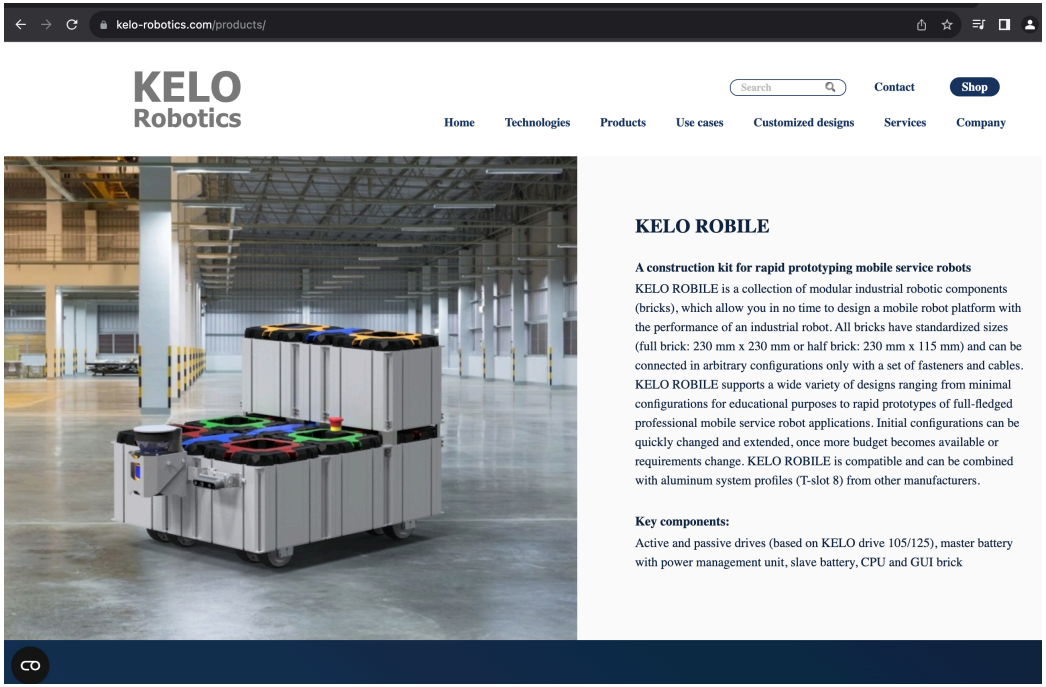
- Passive wheel brick
- Battery brick
- CPU brick
- Active wheel brick

How did we connect?

- Connecting adjacent bricks using brackets
- Connecting multiple bricks internally

• Connecting the wires - Battery connection
• Connecting the wires - EtherCat connection
• Fixing the plastic covers
• Turning on the robot

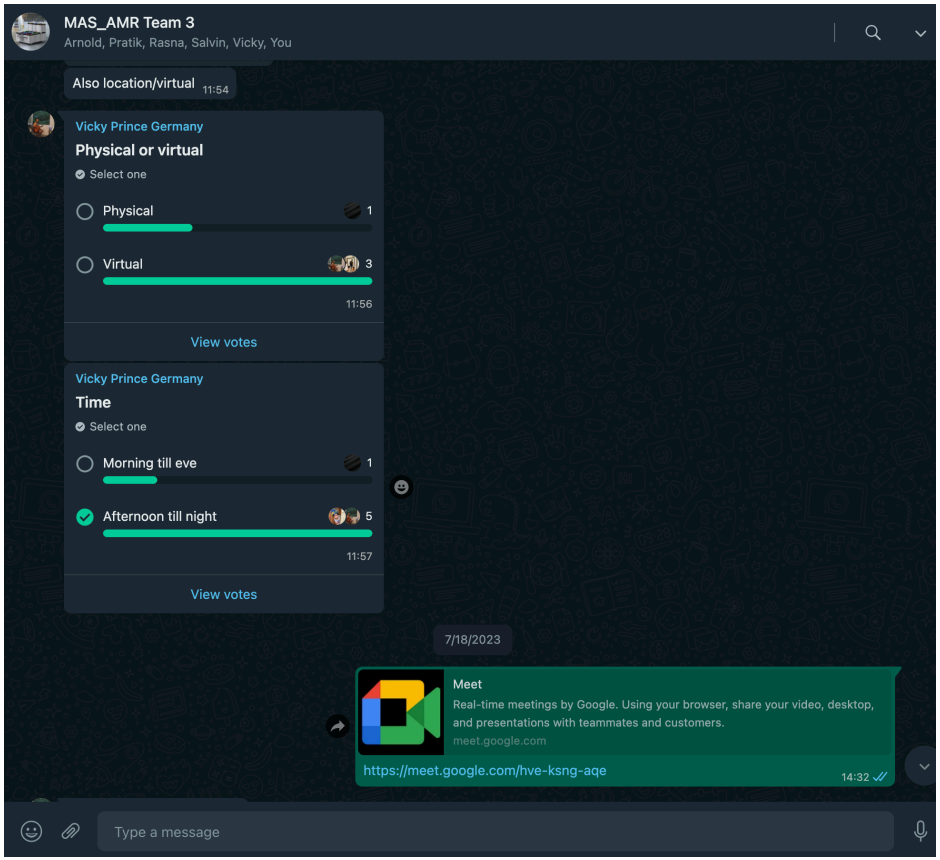
More info about the KELO ROBOTICS is available from https://www.kelo-robotics.com/



• System Overview

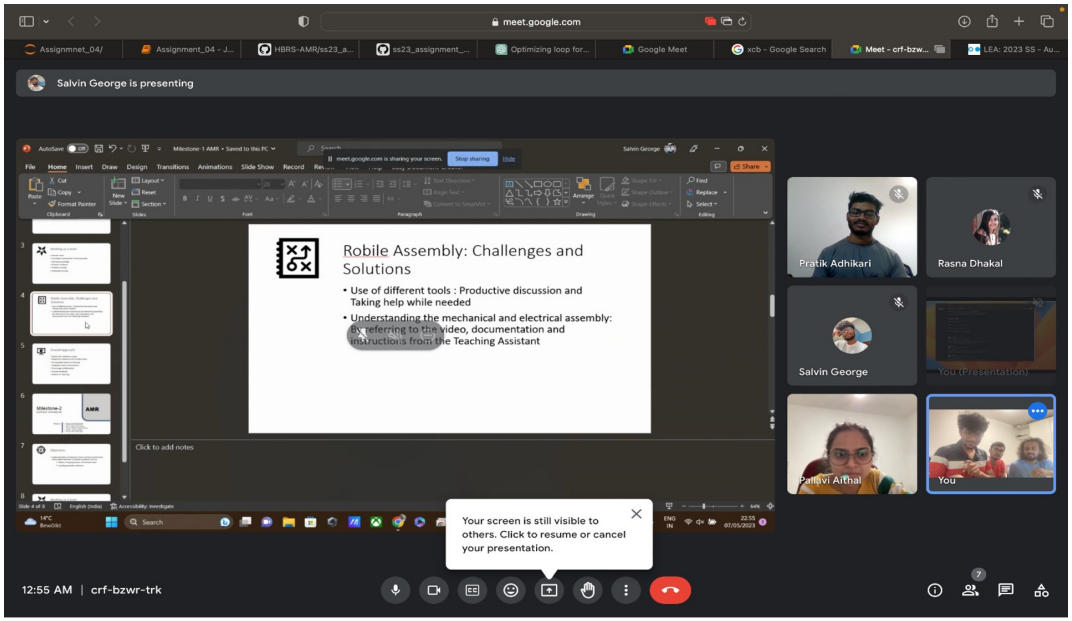
Among several approaches we are following the agile method.

Iterative: Cycles of a few weeks, at most 5 months from April 27th 2023.
Team: 5-6 people
Requirements: Considering Milestones and actively implementing based on the different perspectives.
Method: Dislike dogmatic rules, open for the adaptability.



Extreme Programming (XP) Values: Simplicity, feedback, communication
XP Practices:

- Management
  - Stand-up meetings
- Team
  - Joint responsibility for the code
  - Continuous integration
- Programming
  - Pair programming
  - Refactoring



Scrum Process

- Daily scrum
  Discuss progress, synchronize day plan, discuss and document new obstacles.
- Retrospective
  Assess how well scrum was implemented, identify actions for improvement.

EARS Approach

In our requirements we have observed it to be an **event-driven pattern** and it involves **Behavior-driven development**.

Example:

As a system builder I want the joypad control software to work for different hardware configurations of the Robile So that the maintainence of the software is simplified.
Given a robot assembled in config and is initialized When signal is sent via the joypad Then the robot moves.

• Functional Requirements / User Stories

US 1.1 Assemble robile, joypad control

As a system builder I want the joypad control software to work for different hardware configurations of the Robile So that the maintenance of the software is simplified.

Use case 1.1.1

Given a robot assembled in config and is initialized When signal is sent via the joypad Then the robot moves
Examples: | configuration | command | motion | | 1 driving wheel | left stick up fully + right trigger | forward at most 1m/s | | 1 driving wheel | left stick left fully + right trigger | left at 1m/s | | 2 driving wheel | left stick left fully + right trigger | left at 1m/s |

Expected Behaviour:

Given: The robot is assembled in config and initialised
When: Signal is sent via the joypad

Then: Robot moves

Achieved Behaviour:

Given: The robot is assembled in config and initialised
When: Signal is sent via the joypad
Then: Robot moves

US 2.1 Collision avoidance

As a safety engineer I want the robot not to run into static or dynamic obstacles So that injuries to human and damages to the environment and the robot can be avoided

Use case 2.1.1 - static obstacles Given the robot is moving When is in the robot path Then the robot goes around the (or stops)
Examples: | location | obstacle | | in the kitchen | chair | | through door | vacumm cleaner |

Expected Behaviour:

Given: Static obstacles, the robot is moving
When: Is in the robot path
Then: The robot goes around (or stops)

Achieved Behaviour:

Given: Static obstacles, the robot is moving
When: Is in the robot path
Then: The robot goes around (or stops)

Use case 2.1.2 - dynamic obstacles Given the robot is moving When a moves through the robot path Then the robot does not collide with the obstacle Examples: | location | agent | | in straight corridor | person | | through door | robile |

Expected Behaviour:

Given: dynamic obstacles, the robot is moving
When: Is in the robot path
Then: The robot does not collide

Achieved Behaviour:

Given: dynamic obstacles, the robot is moving
When: Is in the robot path
Then: The robot does not collide

US 3.1

As a behaviour developer I want the robot to be able to follow the wall So that the robot can deal with getting lost when navigating in long corridors

Safe range (collision)	distance (detecting wall)
0.2	0.7

Use case 3.1.1 - Aligning to wall

(Positive slope: rotates right, negative slope: moves left, perpendicular, any)

Expected Behaviour:

Given: The robot is initialised
When: Wall is in view of the robot, when the safe range, distance threshold is reached
Then: Robot reports longest wall with slope
And: The robot moves to align its longer side with the wall

Achieved Behaviour:

Given: The robot is initialised
When: Wall is in view of the robot, when the safe range, distance threshold is reached
Then: Robot reports longest wall with slope
And: The robot moves to align its longer side with the wall

Use case 3.1.2 - Corner Alignment

3.1.2.1 Corner Alignment - Pointing towards corner

Expected Behaviour:

Given: The robot is initialised.
When: Corner is in view of the robot when the safe range, distance threshold is reached
Then: Robot reports longest wall with slope
And: The robot moves to align its longer side with the wall

Achieved Behaviour:

Given: The robot is initialised and is already parallel to one wall.
When: Corner is in view of the robot when the safe range, distance threshold is reached
Then: Robot reports longest wall with slope
And: The robot moves to align its longer side with the wall

Use case 3.1.2.2 - Corner Alignment - Already parallel to one wall

Expected Behaviour:

Given: The robot is initialised and is already parallel to one wall.
When: Corner is in view of the robot when the safe range, distance threshold is reached
Then: Robot reports perpendicular wall with slope
And: The robot moves to align to perpendicular wall

Achieved Behaviour:

Given: The robot is initialised and is already parallel to one wall.
When: Corner is in view of the robot when the safe range, distance threshold is reached
Then: Robot reports longest wall with slope
And: The robot moves to align its longer side with the wall

Use case 3.1.3 - Robot Already Aligned - Slope is 0, follows the wall

Expected Behaviour:

Given: Robot is initialised
When: Parallel wall is in view of the robot
Then: Robot reports slope zero
And: Robot will follow wall

Achieved Behaviour:

Given: Robot is initialised
When: Parallel wall is in view of the robot
Then: Robot reports slope zero
And: Robot will follow wall

Use case 3.1.4 - No Wall found (milestone - 4)

Expected Behaviour:

Given: The robot is initialised
When: Wall is not in view of the robot
Then: Robot reports out of bound value
And: The robot rotates in either of two directions in search of wall.

Achieved Behaviour:

Given: The robot is initialised
When: Wall is not in view of the robot
Then: Robot reports no lines from online detection
And: yet to implement

Use case 3.1.5 - Detects Glass Door (won’t consider as wall/obstacle)

Expected Behaviour:

Given: Robot is initialised
When: Glass door is in view of the robot
Then: Robot reports line from the online detection
And: The robot aligns to the glass door

Achieved Behaviour:

Given: Robot is initialised
When: Glass door is in view of the robot

Then: Robot reports no line from the online detection
And: The robot tries to move through considering no wall/avoidance

Use case 3.1.6 - Detects Pillar (considers as obstacle)

Expected Behaviour:

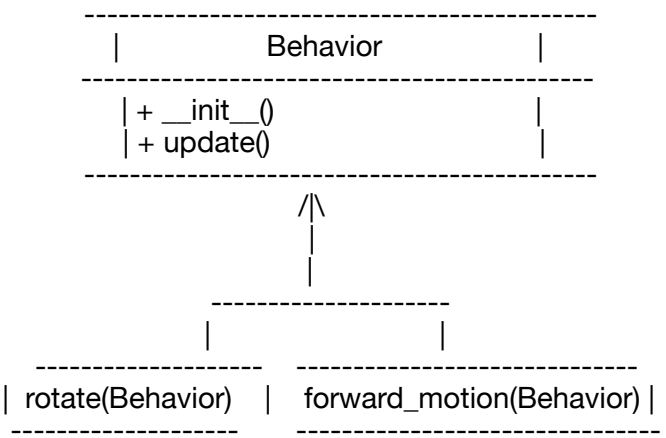
Given: Robot is initialised
When: Pillar is in view of the robot
Then: Robot reports no wall
And: The robot is not aligned to pillar

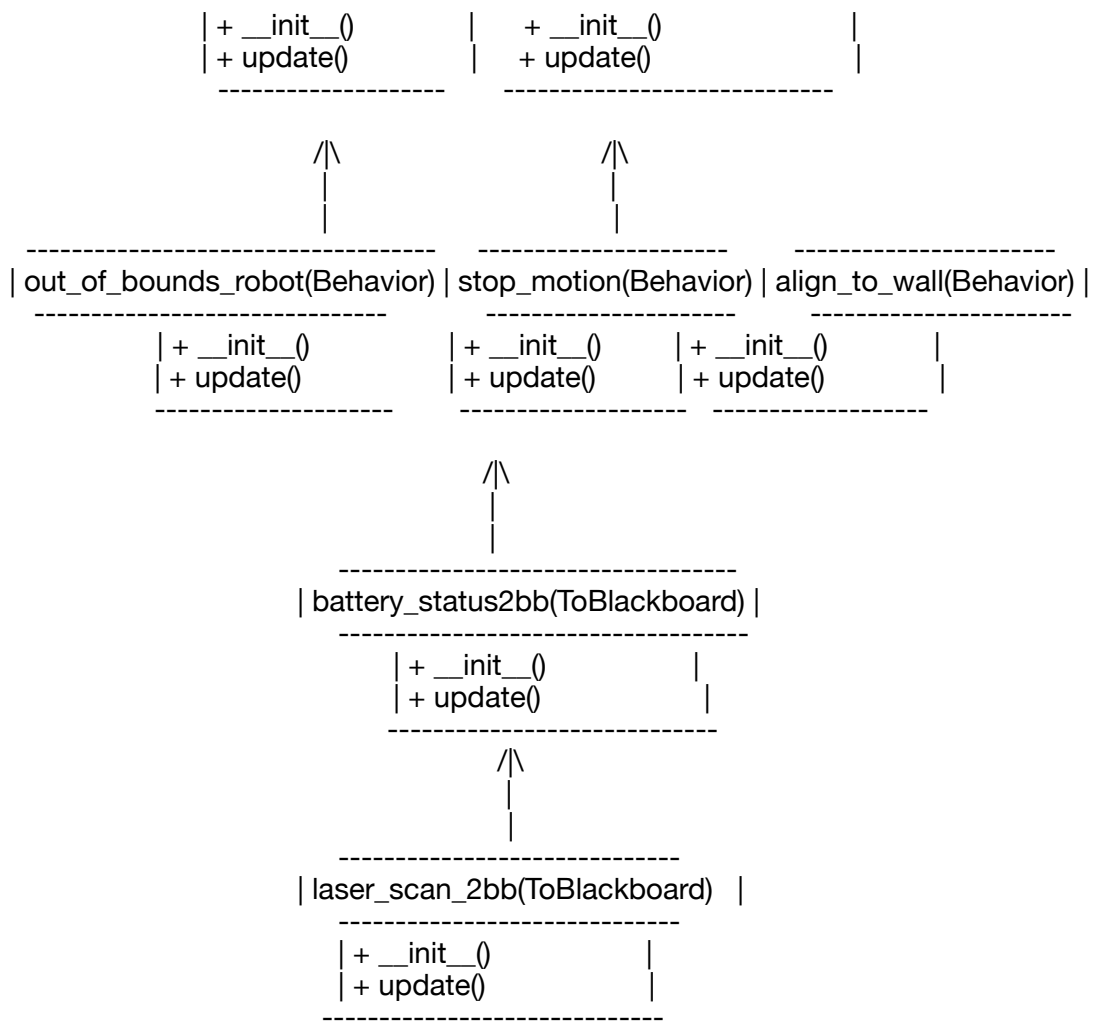
Achieved Behaviour:

Given: Robot is initialised
When: Pillar is in view of the robot
Then: Robot reports no wall
And: The robot is not aligned to pillar considers it as collision avoidance

- **System Behavior**

UML Diagram for the Behavior Tree




| + __init__()
| + update()
+ __init__()
+ update()
out_of_bounds_robot(Behavior)
stop_motion(Behavior)
align_to_wall(Behavior)
| + __init__()
| + update()
| + __init__()
| + update()
| + __init__()
| + update()
battery_status2bb(ToBlackboard)
| + __init__()
| + update()
laser_scan_2bb(ToBlackboard)
| + __init__()
| + update()