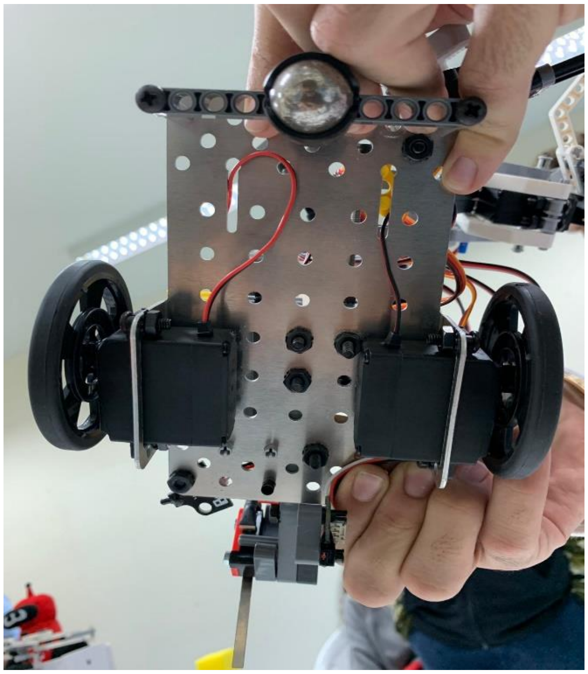
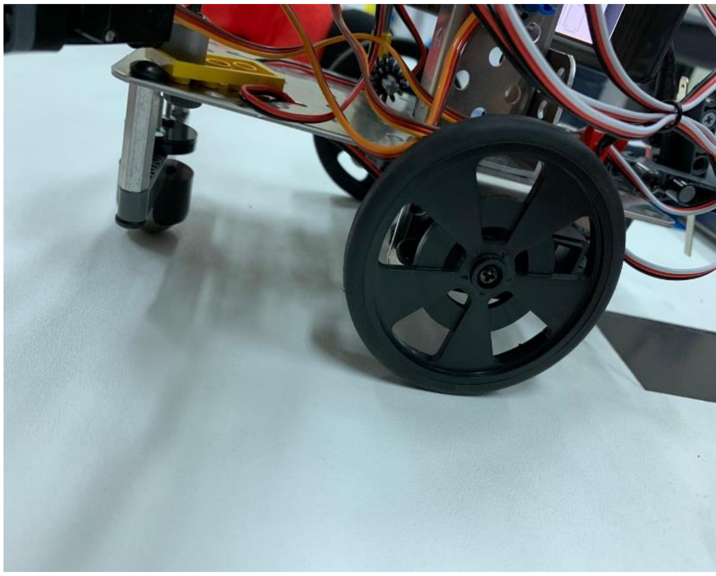
0299 Lebanese School of Qatar

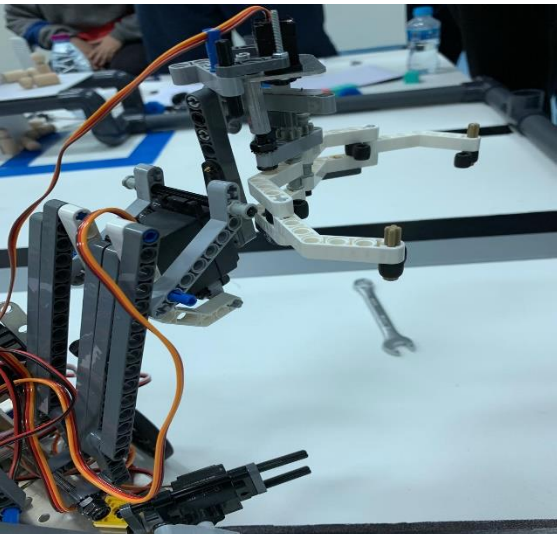
Period 2 Mechanical Design

Drivetrain:



As it is apparent in the photo to the left, we used the basic drivetrain structure taught at the workshop as the heart of the movement, saving ourselves the unnecessary troubles that come along with using a more complex one. More importantly, the efficient drivetrain keeps the weight distribution on both of its sides equal, thus eliminating even the slightest possibility of a deviation from the planned track. One hidden feature our drivetrain possesses is how it makes coding the movement faster and more convenient. The main function is no other than mav(port number, value) which allows our robot to move in any wanted direction. A neat new trick our team thought up is creating a shortcut for a shortcut by reducing "create_drive_forward" and "set_servo_position" to "cdd" and "ssp" respectively, simply improving the life quality of coders. We were presented with two options for our wheels. The first choice was the small LEGO wheels (56 x 26), but with these wheels the robot's turns were inaccurate, and their rotation drained too much of the battery's power due to the fact that they are heavy and uselessly require many rotations to cross a small distance. Their chunkiness gave little room for precision when it came to moving the drivetrain around the perimeter of the table. This made our other choice, the large rubber wheels, a much more tempting one. This is due to their slim shape which allows for wider turning angles and their large circumference that decreases the number of turns required to cross the very same distance. The latter, in turn, decreases the power the drivetrain needs to move to its destination and makes it more likely that it beats the other teams' drivetrains to grabbing the common gas pipes.

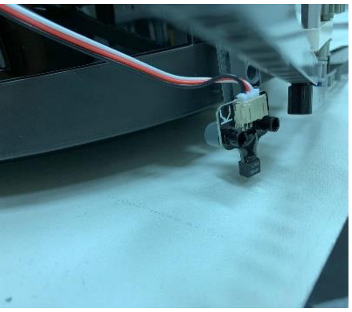


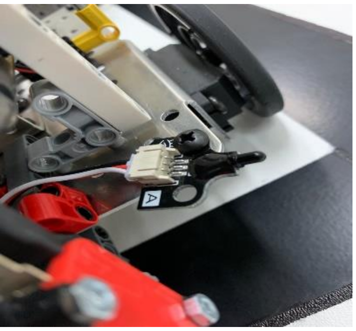


Effector:

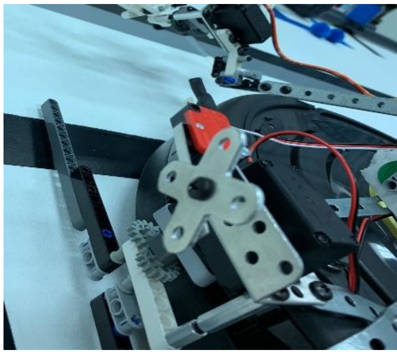
Contrary to our simple drivetrain, our claw (shown clearly in the photo on the left) is a sophisticated structure that not only can move vertically and horizontally, but also rotate 360 degrees to grab the vertical gas pipe and position it horizontally. It consists of LEGOs and one micro-servo. Our first design only had an elevation system and a claw. The goal was to hold the gas pipe up while it's touching its scoring area, but we realized that we shouldn't waste the robot's potential on one task only, but instead diversify its capabilities, thus maximizing our scored points. Once the first gas pipe is placed properly, the robot can go and grab the other one and do the same, granting us another 100 points on top of the original ones.

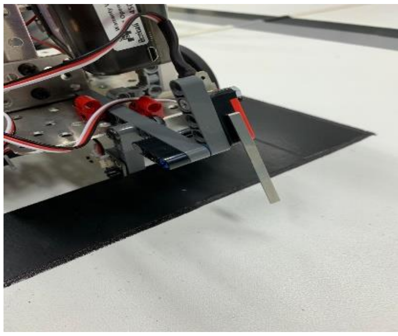
Sensor Mount Section:





The light sensors viewed in the picture to the bottom left were the first type to be mounted onto the base frame of each of our robots seeing as how the game is initiated by a bright light; once the robots detect the light, a countdown of 118 seconds starts so that the robots are shut down a couple of seconds before the given time limit of two minutes. This sensor is simply screwed onto the base plate since it basically only needs to detect the initiation light. Our other two sensors, the touch sensor (bottom right) and the reflectance sensor (top left), were both fixed onto the drivetrain with LEGO. It would have been easy to use UGLu to attach them both, but that would have made them more flimsy and prone to snapping off in case of any rough handling. This is especially true for the touch sensor, which routinely bumps into things just to do its job, and with which we can't afford to have even the slightest bit of fragility.





Data:

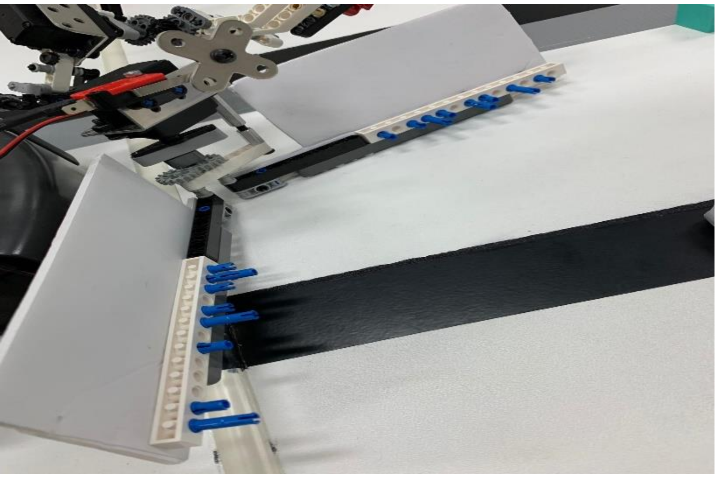
Before weight distribution			
Speed	Delay (ms)	Distance (m)	Deviation (Deg)
250	1000	1	**20**
250	900	1	**18**
200	800	0.97	**16**
200	1000	0.95	**19**

After weight distribution			
Speed	Delay (ms)	Distance (m)	Deviation (Deg)
250	1000	1	**3**
250	900	1	**2**
200	800	0.97	**2**
200	1000	0.95	**3**

The robot was run multiple times from its starting area in a straight line, and the final positions were used to calculate how much it deviated. A simple weight reallocation drastically improved the robot's accuracy as seen in the tables above.

Data Evaluation:

Our drivetrain's steering system accuracy is evaluated in the previous graph. Thankfully, we have managed to distribute weight evenly on either side of the drivetrain, and that is shown in the graph. After the modification, the drivetrain became generally quite consistent and accurate in the angles of its turns. According to our assessed data, simple changes in the weight distribution can drastically affect the robot. With the right code, the robot now easily maneuvers around the table.



Modified system:

Many performances later, we noticed that our robot needed some adjustments. The primary claw with one stationary beam and the arm with merely metal parts were too fragile and totally undependable. The outcome wasn't up to our belief as some of our poms fell off occasionally. Our program didn't need any change and was perfect but the mechanical design permeable to change. The advanced design accommodates a combination of metal and Lego parts, ticks and foam board, as well as a servo for both gears. For further support, ticks were used on the foam board pieces as teeth to secure all the poms and made sure that none of them escaped the claws. We will let the robot go through extra testing phases while carrying poms, just to make sure everything works as intended.