**Team: Bubble Narwhals**

**Robot Design**

We named our EV3 robot as Tim. Tim has a base for navigation and several mission dependent attachments that can be connected to the base with ease. Tim’s base is square shaped and its base dimensions are 6.6’(length) x 6.6’(width) x 5.6’(height).

1. **Mechanical Design:**

**A.1. Robot Base**

The base has two medium sized front wheels each powered by a large motor. The back of the base has one frictionless ball joint slider. One medium motor is attached in between the wheels and faces front with Lego joints around it that are used for connecting the attachments easily. The brick is positioned on top of the large and medium motors.

The base is surrounded in all sides by straight bumpers. These bumpers have several purposes. It aligns and provides additional stability to both front wheels and back slider. It also helps materials in the field to not get entangled under the wheels or the base by pushing them away from Tim’s path.

The base has two color sensors one each in front of each large motor and they face down. To avoid ambient light reflection on the light sensors and improve accuracy in sensing the colors, these sensors are covered around them by using various Lego beams. One gyro sensor is fixed at top of the brick. This gyro sensor tracks turning angle of the base in vertical axis when needed.

In summary, the base is designed to be:

* Symmetric in geometry and weight distribution to a large extent. This helps the wheels to carry approximately equal weights and reduces drifting.
* Short and keep the center of gravity close to the ground and in between wheels. This helps in stability of the base.
* Flexible enough to add mission specific attachments.
* Protected by bumpers on all sides to clear the path, increase stability of wheels, cover the light sensors and to properly align to the field walls during the missions.

**A.2. Attachments**

Attachments are designed to carry out missions. Core idea here is to make them both functional as well as quick to connect to the base. This is achieved by having attachments with either no pin (pinless attachments) or with one or two pins to attach to the base. Tim currently has a total of seven attachments. Many of them do one mission each and couple of them is designed to execute two or three missions. Three of the attachments use medium motors and gears to transmit the necessary motions needed for the missions. Others are used to holding, pushing and grabbing the mission objects. The attachment that carries large water has a claw powered by medium motor and rubber bands are used to center the claw and push the large water into the designated areas in the fountain and the flower.

1. **Software Design:**

**B.1. Programming Conventions and Change Management**

The names of the MyBlocks start with “MB\_” and the mission programs start with “P\_” for easy identification. Projects and programs were created and tested within an active “development” directory and sufficiently working programs were backed up into backup directories and a new version was created for further development. Once in a while, two different projects developed by different individuals were merged together and a new version of the development program was created. Once the versions were tested enough, the final working version was copied to “Final” folder along with corresponding version number. All programs are commented appropriately with more details. Following figure shows the folder structure described above.

**B.2. MyBlocks**

Repeatable and often used functionalities were turned into MyBlocks with relevant parameters as inputs. Some of the frequently used MyBlocks are listed below.

* Moving Straight
  + *MB\_MoveInches*: Rotates both wheels so as to move Tim straight by specified inches with specified power.
* Turning
  + *MB\_GyroTurn\_RW*: Turn using right wheel only. Uses gyro sensor to turn Tim using right wheel motor alone to the specified degrees, with specified power and direction. This block corrects overshoot errors that are common with gyro sensors and achieves the precise degree of turn to large extent.
  + *MB\_GyroTurn\_LW*: Similar to above, but uses only left wheel to turn.
  + *MB\_GyroTurn\_BW*: Uses above blocks simultaneously to turn both wheels to specified degree with specified power and direction.
* Line Alignment
  + *MB\_Align\_White1*: Used to position Tim perpendicular to white line. Uses both color sensors to detect white space and stop Tim when both color sensors senses white. If during alignment process only one of them senses white, the corresponding wheel stops and the other wheel keeps turning until the second sensor detects the white line and stops. Thus when both sensors are in the white line then Tim stops thus achieving approximate perpendicularity with the white line.
  + *MB\_Align\_Black1*: Same as above, but aligns approximately perpendicular to black line.
  + *MB\_Align\_White1B & MB\_Align\_Black1B* : These are follow up programs that helps to fine tune perpendicularity by making minor adjustments after stopping at white or black line using above programs. The perpendicularity is achieved by moving the appropriate wheel slightly so that the border of white and black line is reached.
* Line Following
  + *MB\_LineFollow*: Used to follow the border between black and white lines. Uses one color sensor (right side sensor) to detect the border. If the sensor goes to black it turns left wheel more than the right wheel hence turns clockwise until white line is found. The process is now reversed with right wheel moving faster than the left wheel. When Tim is between the border between black and white line both wheels move with equal speed and Tim follows the border line. The intelligent part of this action is to use simple math to control the speed of the left and right motors based on how far away they are with respect to the black or white lines. This way it aligns and follows the border quickly.

For example:

Formula for power of right wheel: (a – 50)\*0.4 + 20

Formula for power of left wheel: (50 – a)\*0.4 + 20

Here, ‘a’ represents light intensity from color sensor (black is low <15 and white is high > 95). 50 represent the approximate intensity that reads the border between black and white lines. 0.4 is the proportion that is used to specify if how aggressively we would like to search for the border. 20 is the constant that makes both wheels keep going straight while searching and aligning with border.

Apart from the above listed MyBlocks, there are additional simple MyBlocks used to organize the mission programs better.

**B.3. Mission Programs**

Each mission uses its own program consisting of different combinations of MyBlocks and other logics to complete a particular mission. The next section describes the possible missions that Tim can perform, each one of them being controlled by a corresponding program.

1. **Missions:**

Tim could do following missions. Other executable missions that are not being executed in the field due to competition time constraints are also listed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Mission** | **Points** | **Attachment** | **Program** |
| 1 | Filter | 30 | Filter Push | P\_Filter |
| 2 | Flow | 25 | Up / Down Movement | P\_Flow |
| 3 | Tripod | 15 / 20 | Hold and Push | P\_Tripod |
| 4 | Pump Addition | 20 | Hold and Push + Rotate | P\_PumpAdd\_Rain |
| 5 | Rain Cloud | 20 | Hold and Push + Rotate | P\_PumpAdd\_Rain |
| 6 | Fountain | 20 | Grab, Drop and Push | P\_Fountain |
| 7 | Pipe Removal | 20 | Catch and Lock | P\_Pipe\_Removal |
| 8 | Water Treatment\* | 20 | Up / Down Movement |  |
| 9 | Flower\* | 30 | Grab, Drop and Push |  |
| 10 | Water Well\* | 15 / 25 | Hold and Push |  |
| 11 | Sludge\* | 30 | Hold and Push |  |
| 12 | Water Collection (big water)\* | 10 | Grab, Drop and Push |  |
| \* The robot is capable of doing these missions with current attachments, but probably cannot complete them within the allotted time. | | | | |