**SOLUTION:**

The solution procedure that we provide can be investigated as several subsystems, with various options due to redundancy.

1. **Movement**

Movement is the first milestone of the project. Although it seems to be easy to implement, its sustainability is hard to achieve. Nevertheless, our solution about the drivetrain of the robot includes no other alternative than the below-explained one.

In X-Cali, RaspberryPi3 and L293D is used. Motors are two DC Motors. DC motors are chosen, because it provides two options for speed control such as

* PWM
* Voltage Control

In order to drive motors, power supply is used during early stages of the project. After full mobilization of the robot is achieved, two separate Li-Po batteries are going to take place of the power supply. The main reason for using two batteries is to isolate the power flow of the motors and the RaspberryPi3, since current drawn by the motors may result in a large voltage drop on the motor-side load, which decreases the voltage drop on RaspberryPi3 and causes restart, malfunctioning problems.

Another alternative to using two separate batteries is using a single battery instead of two, with appropriate regulator circuits and filter inductances.

Motors are not connected directly to the RaspberryPi3. Sinking current for motors from RaspberryPi3 is dangerous for the microprocessor. Thus, by buffering the motors from the control units, we provided safety for the overall circuitry.

X-Cali is chosen as rear-wheel drive. The main reason of this can be understood by thinking forklifts. Rear-wheel driving provides more instability however, this is something we desire. It provides more maneuverability. When maze dimensions are considered, small turn radii have to be achieved. Additionally, rear-wheel driving, we use tank-like wheel movements. This phenomenon is illustrated in the Figure YYYYYYYYYYYYYYYYY.

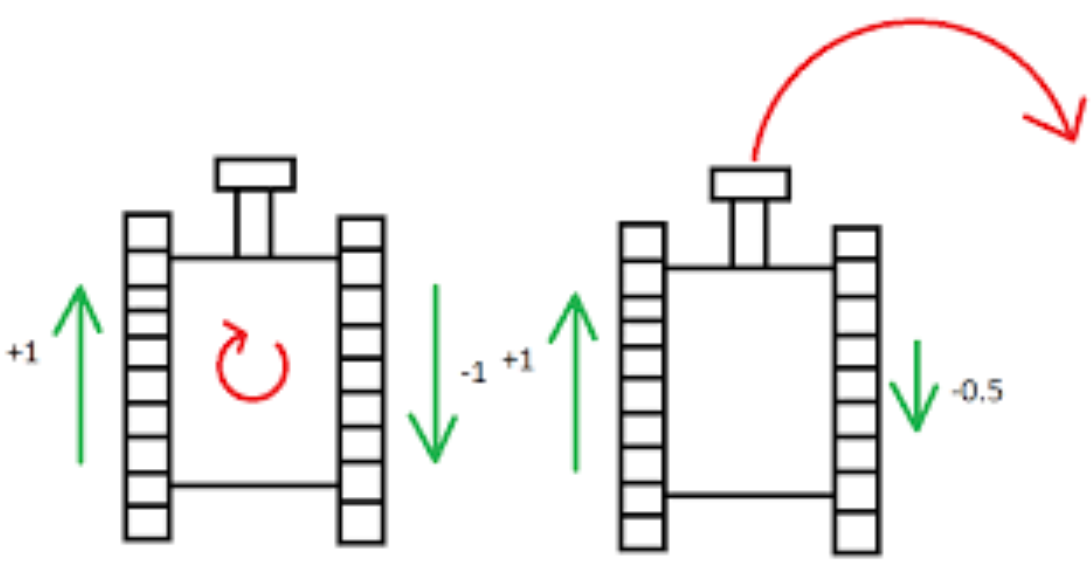


Figure YYYYYYYYYYYYY

A mad-wheel at the front is just used for keeping the balance of the robot.

1. **Sensors**

As sensors, after debating group members, we came up with three options due to redundancy considerations:

1. Ultrasonic Proximity Sensor
2. IR Proximity Sensor
3. Camera

The first two are active sensors. That is ultrasonic proximity sensor sends sound waves and receives the reflected waves. By calculating the difference between emitting time and receiving time, Python program that is embedded in RaspberryPi3 calculates the distance. IR Proximity works with the same principle but rather than sound waves, it uses light.

They have both advantages and drawbacks. Sonar sensors measure the distant obstacle with high accuracy, i.e. further than 500 mm. Whereas, IR sensor can measure the close objects, i.e. closer than 100 mm.

The camera is passive sensor, namely it does not change the environment. The first two proximity sensors are used to proceed in maze and to interpret the maze walls. The main sensor that determines the next movement is the camera. The camera is the crucial part of this solution. The algorithm is described under Decision and Image Processing title.

1. **Decision and Control**

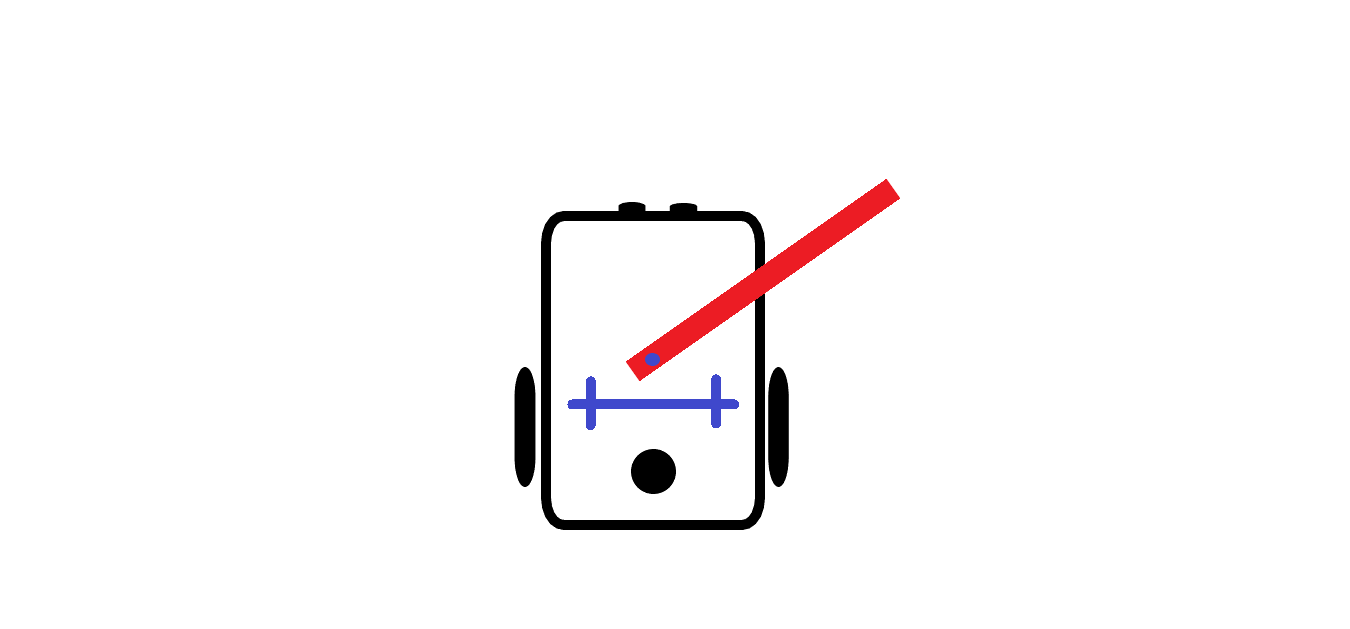
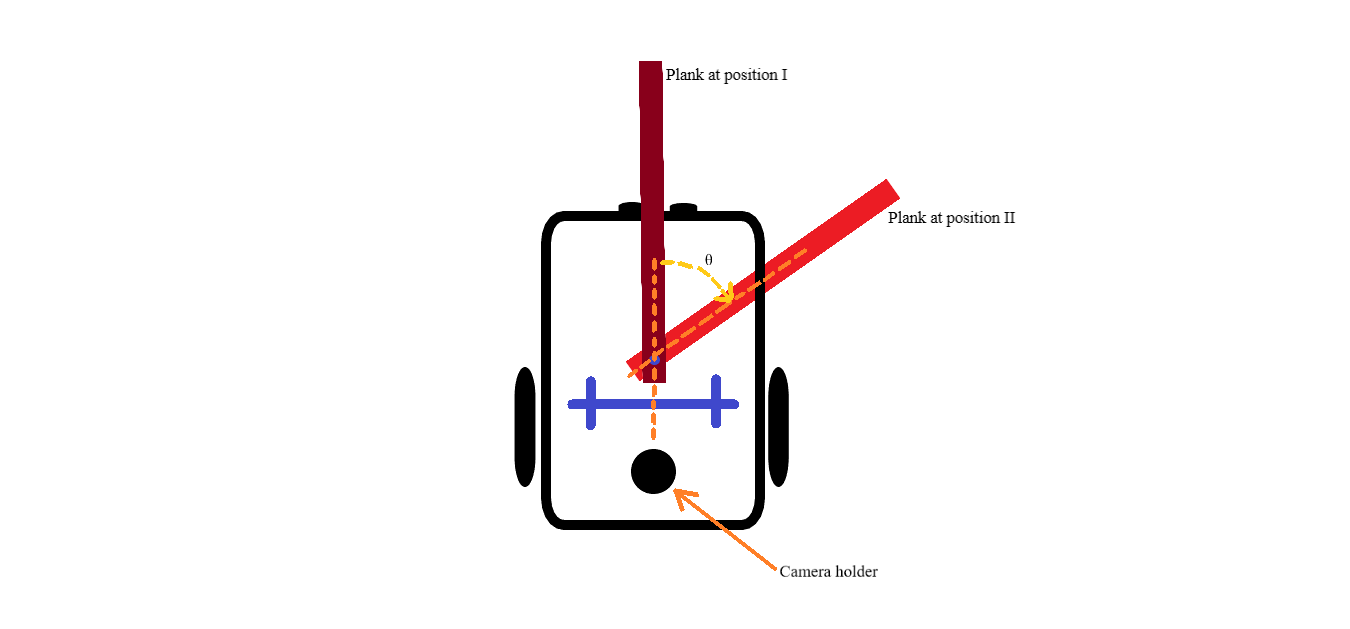
As stated in problem definition previously, the major issue in this project is to propose a method to overcome the lack of direct communication between the two robots, especially when one of them (mostly the master) is in situation of state change, such as entering a U or an L corner. Both active and passive methods should be developed to provide the indirect communication between the robots.

It has already been concluded in the Standards Committee Meetings that the robot should stop and wait for 10 seconds if it encounters an obstacle and is going to start moving towards left or right (entering a U or an L turn). This delay time provides the other robot (mostly the slave) the information that it is in the act of state transition, indirectly. However, this passive method is obviously not enough to provide all necessary information between the robots. As a result, we have developed two different solution methods due to redundancy.

**Solution I:** Image Processing for Collaboration

As a solution, we came up with use of image processing methods. When the possible movements are considered, the only detection can be done by plank and the other robot’s position. From Standard Committee, color of plank is chosen as red. Top of the walls are white while the rest of the walls are white. Thus, the walls and the plank can be observed easily.

In this solution we are planning to sense the other robot’s movements by measuring the angle of the plank using image processing methods. This is method is shown in Figure ZZZZZZZZZZZZZZZZZZZ.



(1)

(2)

Figure ZZZZZZZZZZZZZZZZZZ

The algorithm is as follows.

1. The frame is slowed down in order not to busy the processor. Probably it will be lowered to 10 frames per second.
2. Using the blue double crosses seen in Figure ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ calibrate the zero angle.
3. Extract the region that we are interested: the plank itself and the joint point.
4. Using Otsu Thresholding Method, binary-mask the image.
5. Find the plank’s direction using tracing the boundaries.
6. Find the line that corresponds to planks orientation.
7. Find the point of intersection of plank’s direction line and the blue calibration line.
8. Using the dot product, find the angle of intersection.
9. If the angle is larger than a previously determined angle value, then this means the other robot is in turn.
10. Initial angle is known and it 90⁰ to the blue calibration line. Subtracting the current angle, θ is determined.

As can be seen above, this method leans on a simple idea. This method can be improved by adding the integrating the wall detection algorithms so that robot can observe the other one directly.

**Solution II:** Processing the Information Retrieved from the Plank for Collaboration

Even if a solution with image processing seems accurate, it should always be kept in mind that after the implementation, a solution may not always work as accurately as planned or may not work at all. For that reason, a back-up solution is developed in order to sense and process the movements of the other robot. The only thing that links the two robots is the plank, hence data collected from it may be manipulated and used as in the previous solution.

In this solution, it is planned to mount an encoder with Hall effect sensor and a 3-axis gyroscope to the shaft of holding point of the plank.

Hall effect sensor is a sensor which creates a voltage difference between its output terminals as a result of change in magnetic field, due to Lorentz force. This working principle is clearly illustrated in Figure ABC below.

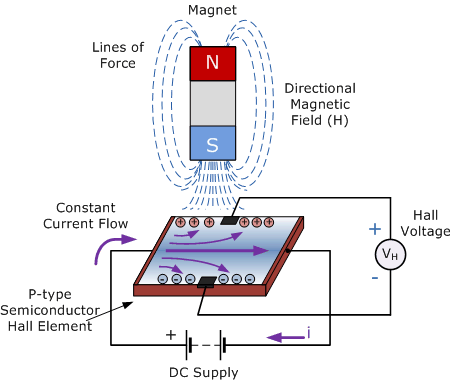


Figure ABC: Illustration of Hall Effect

When such a sensor is mounted on an encoder, continuously changing magnetic field while the rotor of the motor or shaft of the holding point rotates creates continuous impulses at the output voltage. This phenomenon is also illustrated to clarify, in Figure ASD below.

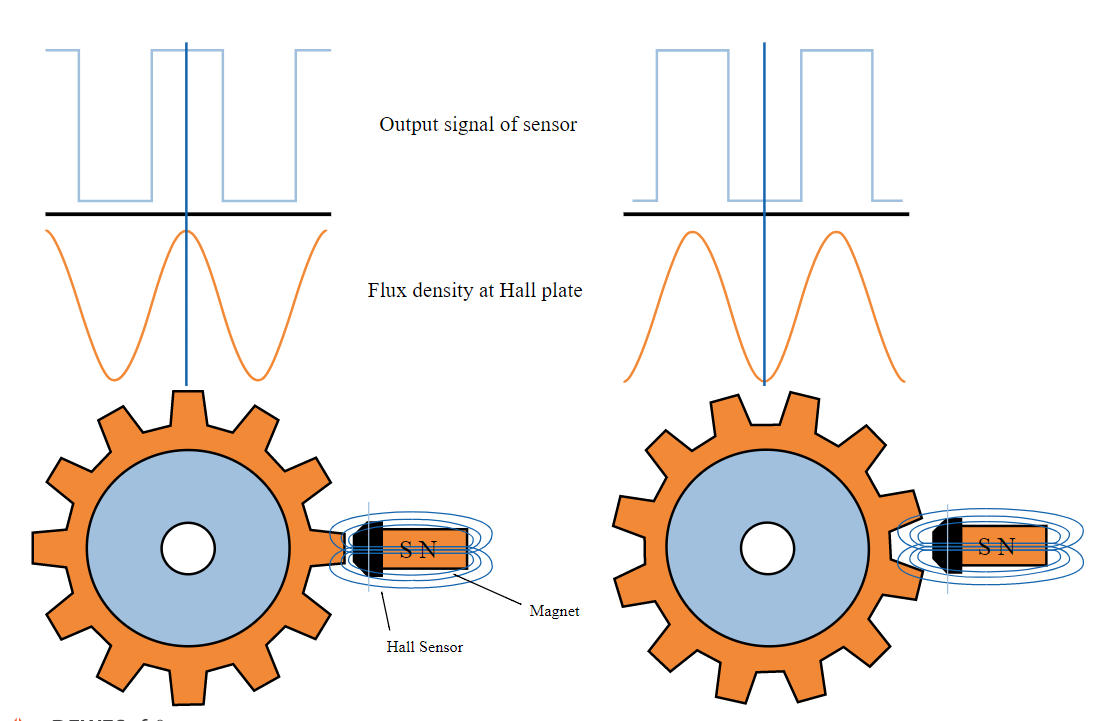


Figure ASD: Output Voltage of Encoder with Hall Effect Sensor

Apparently, the rotation angle of the shaft can easily be found by fractioning the number of upcounts as the output of the encoder with Hall effect sensor, to number of upcounts corresponding to a full rotation of the encoder.

where θ is the rotation angle of the plank in degrees.

The main problem with this solution is that even if the rotation angle of the plank can be calculated very accurately, unfortunately it does not allow the robot to sense if this angle is positive or negative. In other words, it does not provide the information whether the collaborating robot moves towards left or towards right. Using a 3-axis gyroscope, this missing information can also be obtained. The acceleration of the shaft of the holding point causes a digital output, which is positive if the acceleration is in positive x, y, z direction or negative, if the acceleration is in negative x, y, z direction.