GROUP 26

Individual Project Report

BY TUHIN SHAIKH

Introduction

In the following Report, I'll delve into the intricate mechanics behind Lego Mindstorms, exploring the captivating world of programming robots and unravelling the essence of its core concepts. I will provide a detailed breakdown of its functionality and a diagram to show each step of the robot and what would happen if it entered one of the loops. I will also discuss how our team worked and reflect on our collaborative journey and what was the most challenging bit of working as a 3 personal group.

Robot Description

The initiation of our robot is as elegant as it is efficient; a single clap sets it into motion, ready to gracefully trace its designated path. As it glides along the line, its vigilant sensors meticulously scan the environment, detecting obstacles with unwavering accuracy. With a detection range of 20cm, our robot anticipates potential hindrances and halts promptly, exemplifying safety and reliability at every juncture.

Apart from its sophisticated obstacle detection skills, our robot has a range of behaviours designed to improve its operational robustness. When an issue arises, such as sudden obstacles or low battery power, it automatically engages an emergency stop, putting safety first. Furthermore, our robot quickly avoids obstructions thanks to its clever trundle backup system, guaranteeing continuous progress along its intended path.

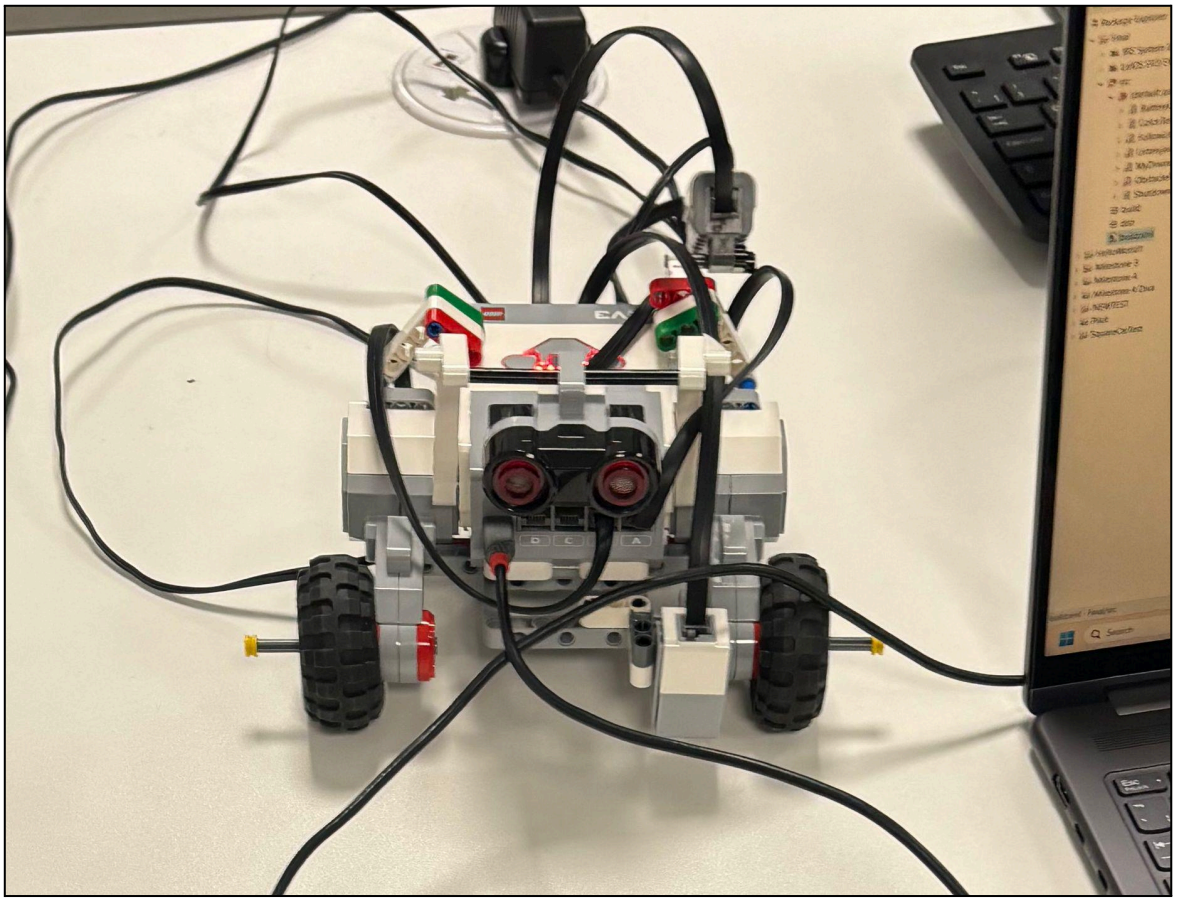

FIGURE 1: Image of Robot

Key Concepts

With the help of the LeJOS EV3 platform, we have created a robotic marvel that follows predetermined routes while also demonstrating professionalism and creativity with each accurate move and alert response. It was written in Java and can be uploaded to the robot and executed from there. Lego Mindstorms are bricks that run Java virtual machines on their basic hardware. Every brick has an input button, a speaker, and a button for displaying messages or making basic noises. In addition, they may accommodate up to 4 motors and 4 sensors which we as a group took advantage of and created our robot.

Throughout our project development, we utilized various online resources to enhance our understanding of LeJOS EV3 programming and robotics concepts. One invaluable resource was the LeJOS documentation, which provided comprehensive guidance on the platform's features and functionalities. Additionally, we frequently referred to online tutorials and forums to get inspired by what others have used their sensors for. We also found a youtube video of a lejos robot similar to one of our ideas but decided for the best to make something original.

Sensors

The robot utilizes a light sensor to detect changes in light intensity, which enables it to follow a line. The EV3ColorSensor is configured in red mode to measure the intensity of reflected light. By fetching samples from the sensor, the robot can determine whether it is over the line or off it, adjusting the motor movement accordingly to maintain its trajectory. For example, one of the sensors we used was a “light sensor” which uses the obtained light intensity values to control the robot's motor movements based on predefined thresholds. The colour we chose that will follow is black. Sensors are essential components in our robotic systems, enabling them to perceive and interact with their environment meaningfully.

Behaviour

Behaviours in LeJOS EV3 robotics serve as the building blocks of the robot's functionality, dictating its responses to different situations. These behaviours, such as following a line or avoiding obstacles, are tailored to specific tasks and triggered by certain stimuli. By organizing functionality into separate behaviours, programmers can create adaptable robotic systems capable of navigating diverse scenarios.

For instance, an "Obstacle Avoidance" behaviour utilizes sensors to detect obstacles, prompting the robot to autonomously navigate around them to continue its path. This involves algorithms for calculating alternative routes and ensuring safe navigation while avoiding collisions. Such behaviours showcase the intelligence and adaptability of LeJOS EV3 programming, enabling robots to respond dynamically to their surroundings.

Navigation/Pilots:

In our robotic creation, navigation and control are facilitated by the LeJOS EV3 platform's navigation framework, primarily through the use of MovePilot objects. These pilots serve as the backbone of the robot's locomotion, enabling precise control over its movements and actions. The MovePilot class provides an intuitive interface for controlling the robot's motors, allowing for smooth and accurate navigation along predefined routes or in response to

environmental stimuli. Through carefully crafted commands, such as travel distance, rotate angle, or arc radius, the robot can execute complex manoeuvres with ease and precision.

Technical Explanation

Classes

BatteryCheck:

The BatteryCheck class plays a crucial role in monitoring the robot's battery level and responding appropriately to low power situations. Similar to other behavior classes, it implements the Behavior interface and relies on a MovePilot instance for robot control. The constructor initializes the BatteryCheck object with the MovePilot reference. The takeControl() method evaluates whether the battery voltage is below a specified threshold (in this case, 7.0 volts) and if a clap is detected using the Listen.heardClap() method.

Upon meeting these conditions a warning message is displayed on the LCD screen indicating "BATTERY LOW!". Additionally, the robot emits a series of beeps to draw attention to the low battery condition. This behavior persists until suppressed, ensuring that the robot's operation ceases and prompts user attention to address the low battery situation.

Listen:
The Listen class is responsible for detecting sound events, specifically claps, using a sound sensor. It implements the Behavior interface and utilizes an NXTSoundSensor for sound detection and a MovePilot object for robot movement. When a clap is detected, as determined by the takeControl method, the robot performs a predefined action, such as travelling forward, in the action method. This behaviour is not suppressible, as indicated by the suppress method which is in the group file link.

FollowLine:
The FollowLine class handles the task of following a line using a colour sensor. It implements the Behavior interface and utilizes a MovePilot object for controlling movement and a colour sensor for detecting the line. The takeControl method checks if the colour sensor detects the line, and if so, returns true to indicate that this behaviour should take control. Moreover, the robot moves forward while continuously checking for the presence of the line. If the line is no longer detected, the robot stops. This behaviour can be suppressed if needed, as indicated by the suppress method.

ObstacleAvoidance:
The obstacle avoidance class handles the avoidance of obstacles detected by an ultrasonic sensor. It implements the Behavior interface and utilizes a MovePilot object for robot movement and an EV3UltrasonicSensor for detecting obstacles. When an obstacle is detected within a certain range, specified by the takeControl method, the robot executes avoidance maneuvers in the action method. It rotates to find a clear path and then moves forward to avoid the obstacle. The suppress method allows for the behaviour to be interrupted if needed.

MyDriving:
The MyDriving class serves as the entry point for the program and orchestrates the initialization of sensors, motors, and behaviours. Upon execution, it displays a welcome message and initializes necessary components such as ultrasonic and colour sensors, as well as a moving pilot for controlling movement. It then creates instances of various behaviours including Listen, FollowLine, ObstacleAvoidance, BatteryCheck, and Shutdown, and adds them to an arbitrator. The arbitrator manages the execution of behaviours based on their priority levels, ensuring the smooth operation of the robot.

Shutdown:
The Shutdown class is responsible for gracefully terminating the program when the ESCAPE button is pressed. It implements the Behavior interface and utilizes a MovePilot object for controlling the robot's movement. Upon detecting the ESCAPE button press, the takeControl method returns true, indicating that this behaviour should take control. In the action method, the robot stops its movement and then exits the program. This behaviour ensures that the user can safely terminate the program.

Code Examples:

```
@Override
public boolean takeControl() {
    soundProvider.fetchSample(sample, 0);
    float soundLevel = sample[0];
    if (soundLevel > 0.5 && !heardClap) { // Adjust the threshold as needed
        heardClap = true;
        return true;
    }
    return false;
}

public static boolean heardClap() {
    return heardClap;
}

@Override
public void action() {
    pilot.stop();
    pilot.travel(100);
    pilot.stop();
}
```

FIGURE 2: Code exemplar

In this code, the robot won't move until it detects a clap.

```
public class MyDriving {
    public static void main(String[] args) {
        // Display welcome screen
        LCD.clear();
        LCD.drawString("Welcome!", 0, 0);
        LCD.drawString("Authors:", 0, 1);
        LCD.drawString("Zuhaib, Savir", 0, 2);
        LCD.drawString(" & Tuhin", 0, 3);
        LCD.drawString("Version: 1.0", 0, 4);
        LCD.drawString("Press a key", 0, 6);
        Button.waitForAnyPress();
        LCD.clear();

        // Initialize sensors and pilot
        EV3UltrasonicSensor us = new EV3UltrasonicSensor(SensorPort.S1);
        EV3ColorSensor cs = new EV3ColorSensor(SensorPort.S2);
        NXTSoundSensor ss = new NXTSoundSensor(SensorPort.S3);
        cs.setFloodlight(true);
        SampleProvider redLight = cs.getRedMode();
```

FIGURE 3: Code example

In this code, when the robot boots up it will be welcomed by a home screen. They will then be able to press a button to start the robot.

```
@Override
public boolean takeControl() {
    colorProvider.fetchSample(sample, 0);
    float color = sample[0];
    // Listen.heardClap() &&
    return !suppressed && color <= COLOR_THRESHOLD;
}

@Override
public void action() {
    while (!suppressed) {
        colorProvider.fetchSample(sample, 0);
        float color = sample[0];
        if (color <= COLOR_THRESHOLD) {
            ObstacleAvoidance.flag = 0;
            pilot.forward();
            ObstacleAvoidance.setTurned();
        } else {
            pilot.stop();
            return;
        }
        Delay.msDelay(100);
    }
}

public static void setSuppressed() {
    suppressed = false;
}
```

FIGURE 3: Code exemplar

In this code, the robot will follow the colour black. If it can't detect a black line the robot would not be working. Moreover, we also have an Obstacle Avoidance system as the obstacles in our grid would be different colours.

Control Flow:

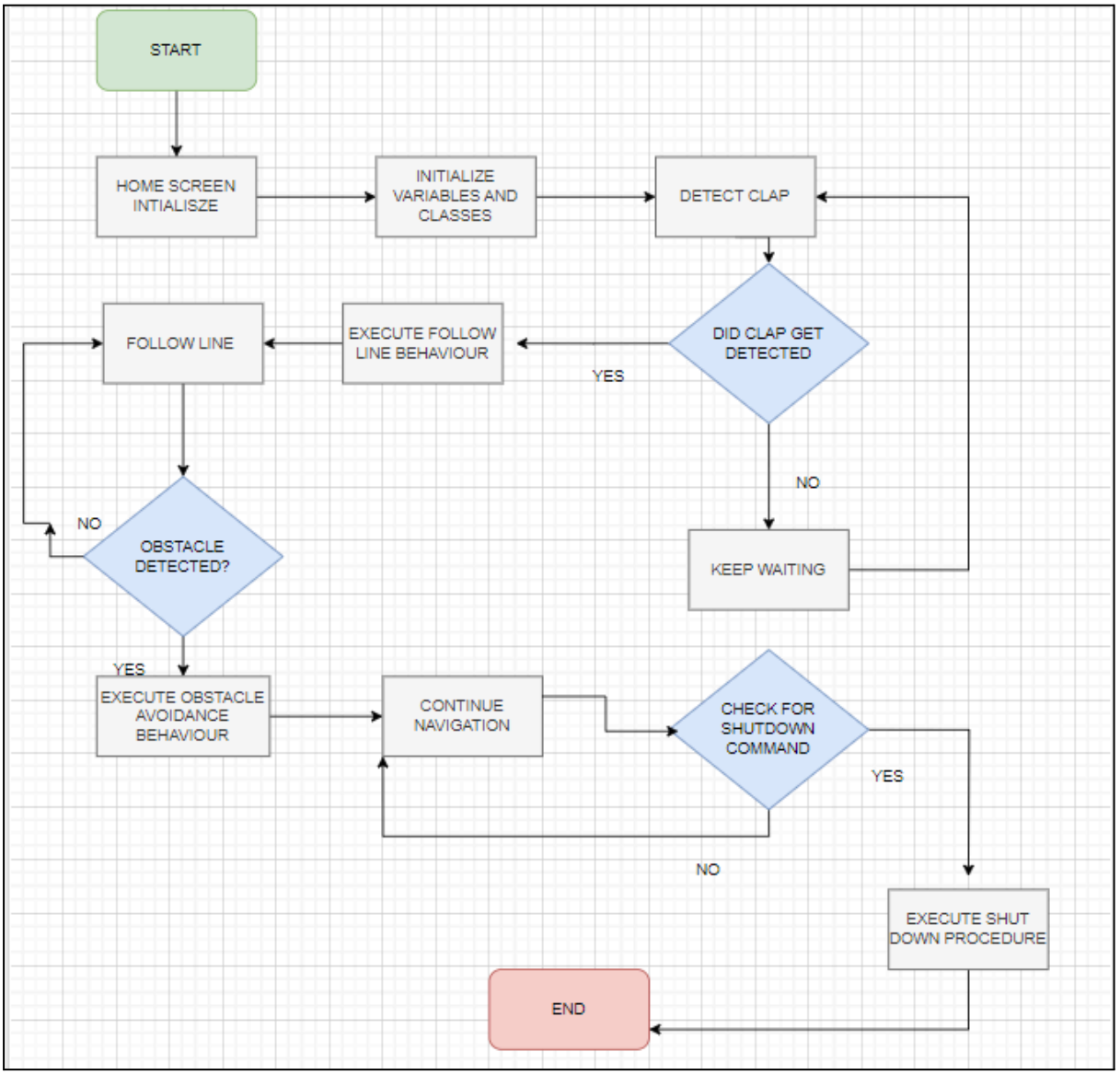

FIGURE 3: Diagram of what the robot does

The following flowchart shows that the robot displays a home screen message, waits for a clap signal to start, and then follows a predetermined line path. While navigating, it checks for obstacles and adjusts its course if necessary. If a shutdown command is received, the robot gracefully ends its operation. This loop ensures continuous responsiveness to user signals and environmental changes.

Reflection:

Throughout the process, our group demonstrated a thoughtful consideration of the entire project, from initial conceptualization to execution. We worked cohesively as a three-person team, effectively leveraging each member's strengths and expertise. Despite the inherent challenges of collaborating remotely, we managed to maintain clear communication and shared goals, resulting in a successful outcome.

One of the significant hurdles we encountered was refining the obstacle avoidance functionality, which required extensive troubleshooting and iteration. Reflecting on our experience, we recognize the value of incorporating a Bluetooth connection for enhanced versatility and control. Overall, we are proud of our accomplishments and the cohesive design of our robot, which surpassed our initial expectations. Moving forward, we look forward to the possibility of collaborating again and building upon this foundation of teamwork and innovation. It's worth noting that despite facing occasional challenges, we successfully met four out of five milestones, underscoring our ability to overcome obstacles and achieve our objectives.

Conclusion:

In conclusion, I believe this was a major success as I personally also passed ⅘ Milesheets. The robot design was also amazing and with the robot working I think it was a major achievement since before this module none of us had ever coded a robot. I would also like to mention despite losing a team member we still carried on and were resilient enough to build our obstacle detection robot.