WRO® WORLD ROBOT OLYMPIAD™

EARTH ALLIES

RFL Academy

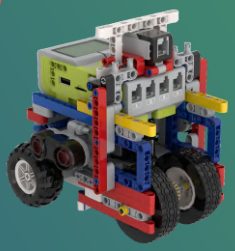
TEAM RGB

ENGINEERING PORTFOLIO

Lightweight Robot (0.9 kg) made up of LEGO parts

Custom Gear & Rack (Steering Mechanism)

EV3 Mindstorm Programming



#1843

Compact Robot (16.5 x 16 x 24) cm

Custom Blocks for PID and Gyro Control

PIXY 2.1 Camera for Obstacle Detection

Team Roles

We are **Team RGB**, a duo of engineering students from **RFL Academy**, united by our passion for **Robotics**. We have clearly divided our roles to maximize our strengths:

BHAVYA SANGHRAJKA

LEAD PROGRAMMER & ELECTRONICS

Bhavya is responsible for the robot's **programming and electronics systems.** He specializes in developing advanced control algorithms, including **PID control,** and designing the logic that powers the robot's movements and **decision-making.** Bhavya ensures the seamless integration of sensors and motors, programming the robot to execute **precise maneuvers** for both the open-round and obstacle-round.

SAHIL GAJERA

MENTOR

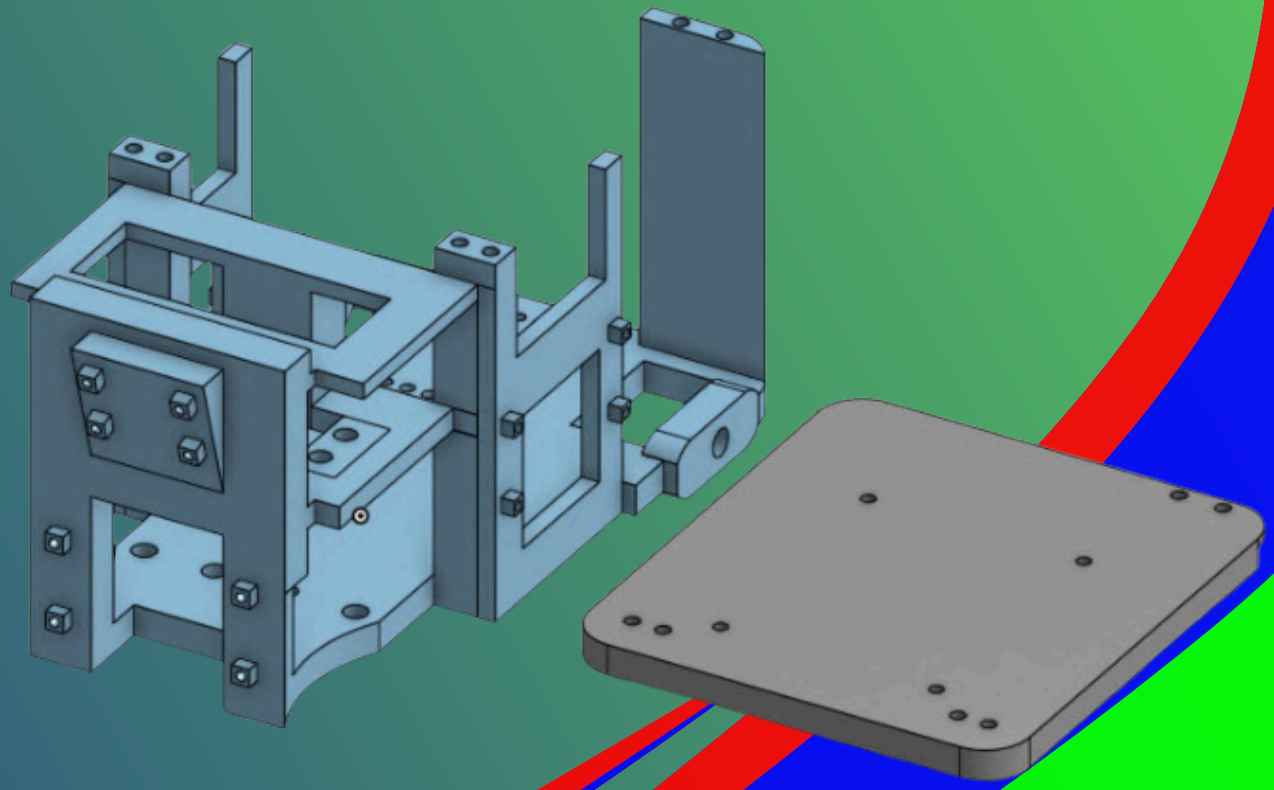
RAJ KOTHARI

LEAD DESIGNER & CONSTRUCTOR

Raj is responsible for the **structural and mechanical design** of the robot. With a focus on precision and functionality, he uses **CAD software** to create a robust and lightweight design that adheres to the competition's strict dimension and weight restrictions. Raj specializes in optimizing the integration of LEGO components with custom-designed parts, ensuring the robot performs reliably in both **maneuverability and stability** under competition conditions.

Construction Timeline

First Iteration



In the Initial phase of our design, we prioritized selecting **lightweight yet durable** components to ensure the robot maintained an optimal balance between strength and weight.

STRENGTHS
- Lightweight and Durable Focus
- Customization Options

WEAKNESS
- Structural Instability
- Bulky Design

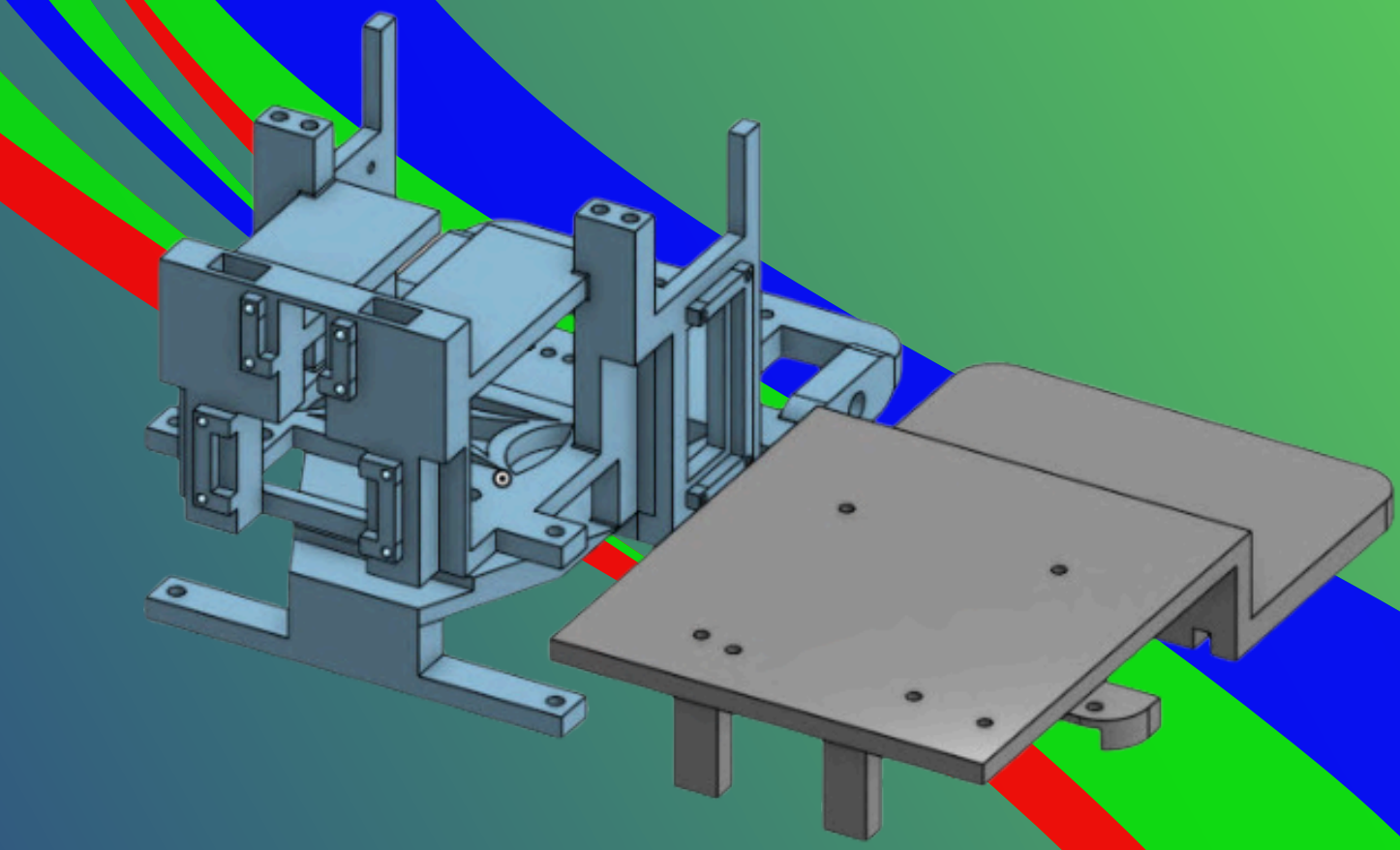
S W O T

OPPORTUNITIES
- Material Optimization
- Enhanced Modular Design

THREATS
- Performance Limitations
- Time and Resource Constraints

Construction Timeline

Second Iteration



For the Second iteration, we transitioned to a **fully 3D-printed design**, including the **steering mechanism**.

STRENGTHS
- Fully 3D-Printed Design
- Precision in Gear Meshing

WEAKNESS
- Assembly Complexity
- Reprinting Delays

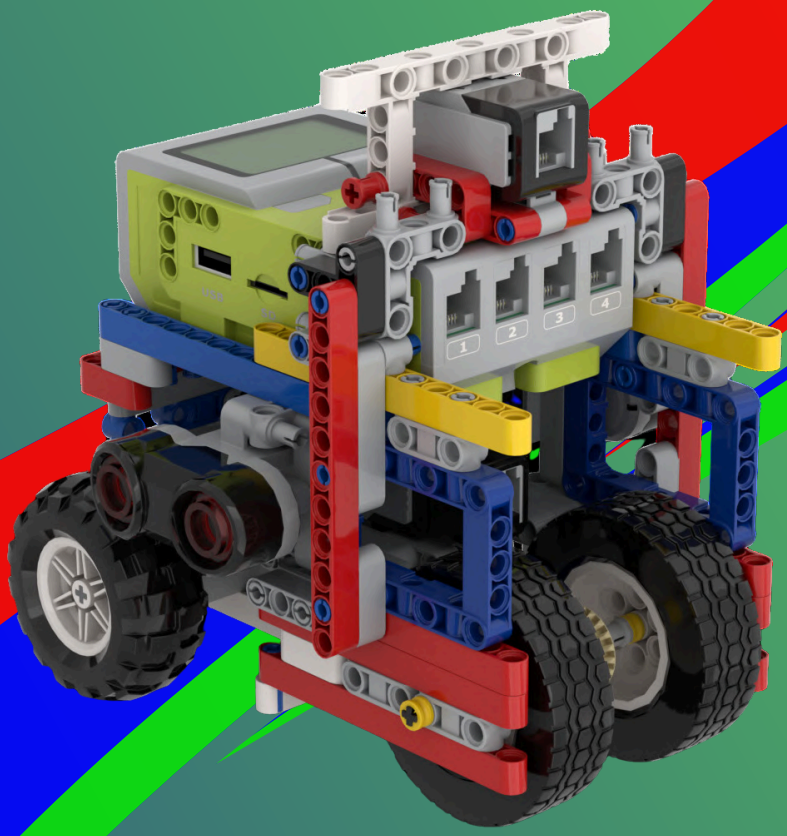
S W O T

OPPORTUNITIES
- Improved Assembly Process
- Iterative Refinement

THREATS
- Time Constraints
- Dimensional Inaccuracies

Construction Timeline

Final Iteration



In the Third iteration, we transitioned to a **fully LEGO-based chassis** including the **steering mechanism**.

STRENGTHS
- Customization & Reliability
- Precision Gear Integration

WEAKNESS
- Design Flexibility Limitations
- Potential Compatibility Issues

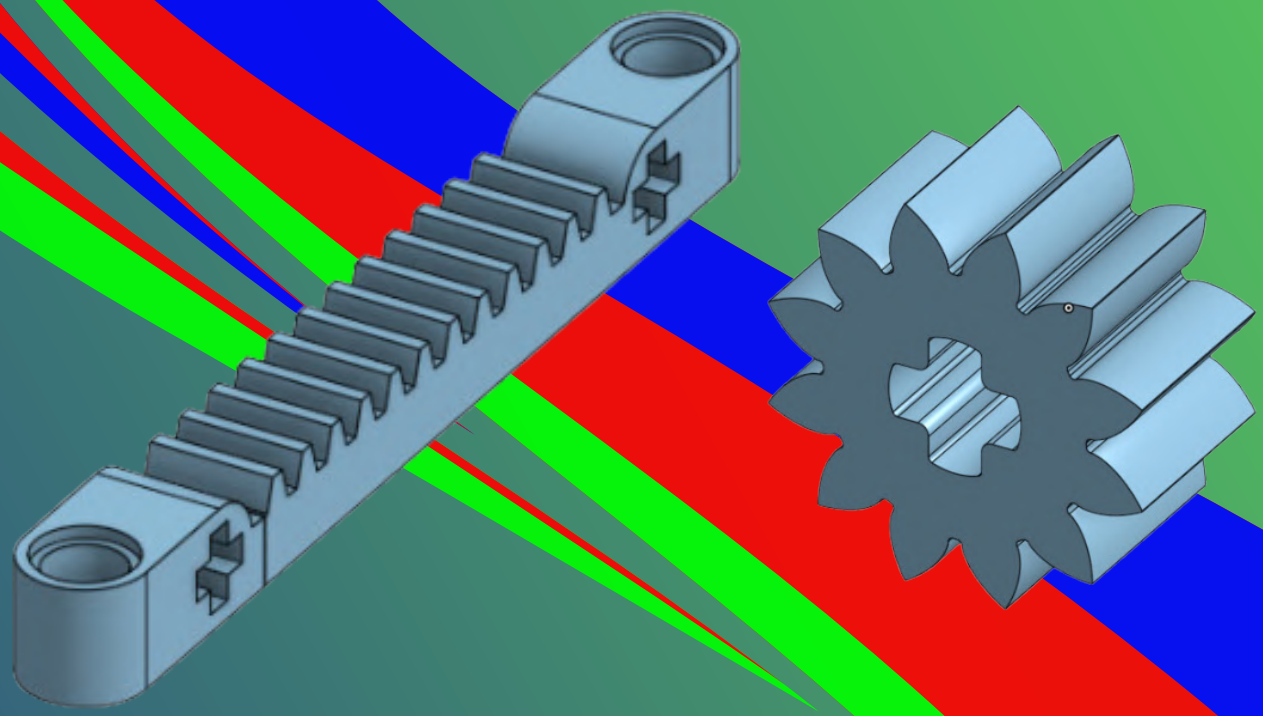
S W O T

OPPORTUNITIES
- Hybrid System Optimization
- Enhanced Design Modularity

THREATS
- Component Wear & Tear
- Structural Limitations

Custom 3D Printed Parts

Rack & Gear



We designed a **custom 7.2 cm rack** to replicate the **groove** pattern and dimensions of the standard **6.4 cm LEGO rack**.

STRENGTHS

- Improved Rack Design
- Enhanced Steering Precision

WEAKNESS

- Initial Misalignment Issues
- Complex Manufacturing

S W O T

OPPORTUNITIES

- Optimized Custom Components

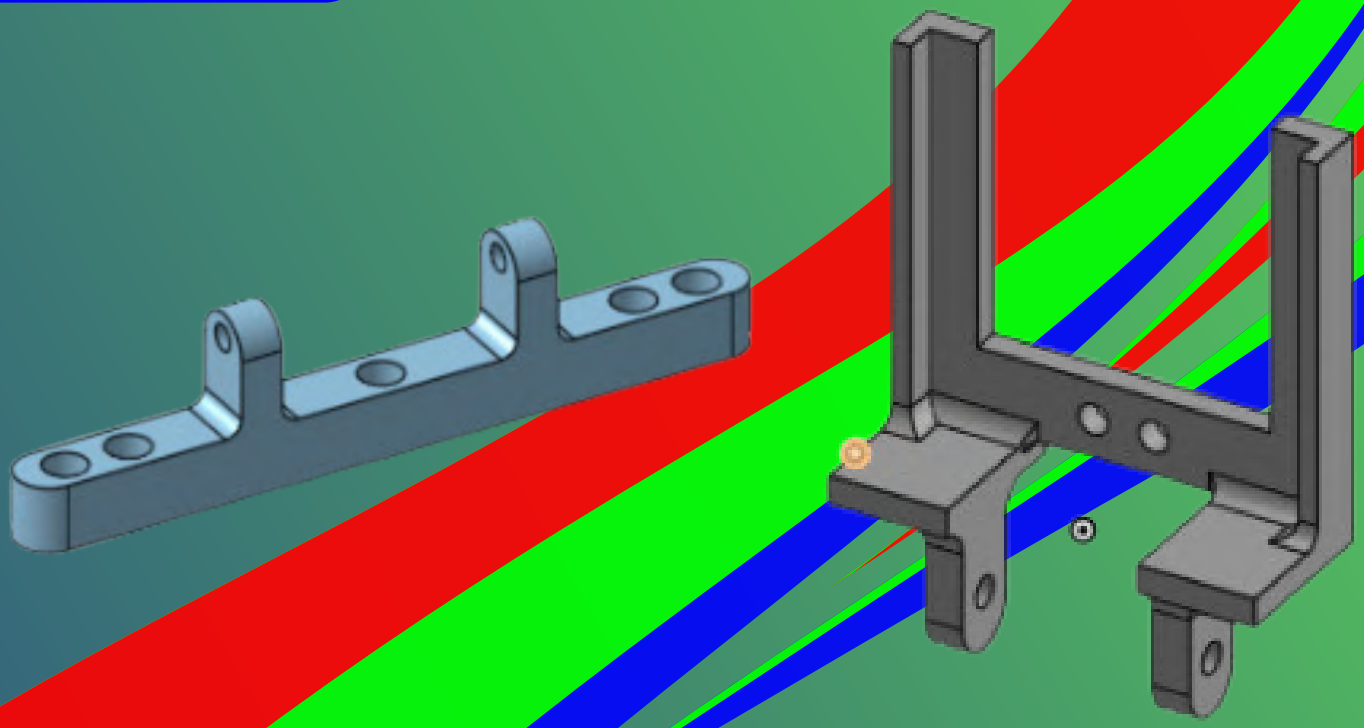
THREATS

- Component Wear & Tear
- Component Compatibility

#1843

Custom 3D Printed Parts

Pixy Mount



During the robot's journey, we encountered **instability** with the **Pixy camera**, which frequently shifted and struggled to accurately detect objects. To address this issue and secure the Pixy at a precise angle, we 3D-printed two custom components. First, we created a custom beam that perfectly aligned with the **holes of the H-frame**, enabling us to securely attach the Pixy to the connector. Additionally, we designed a **custom mount** to provide further support and stability for the Pixy camera.

STRENGTHS

- Improved Camera Stability
- Precise Alignment

WEAKNESS

- Potential Adjustability Limitations
- Complex Assembly

S W O T

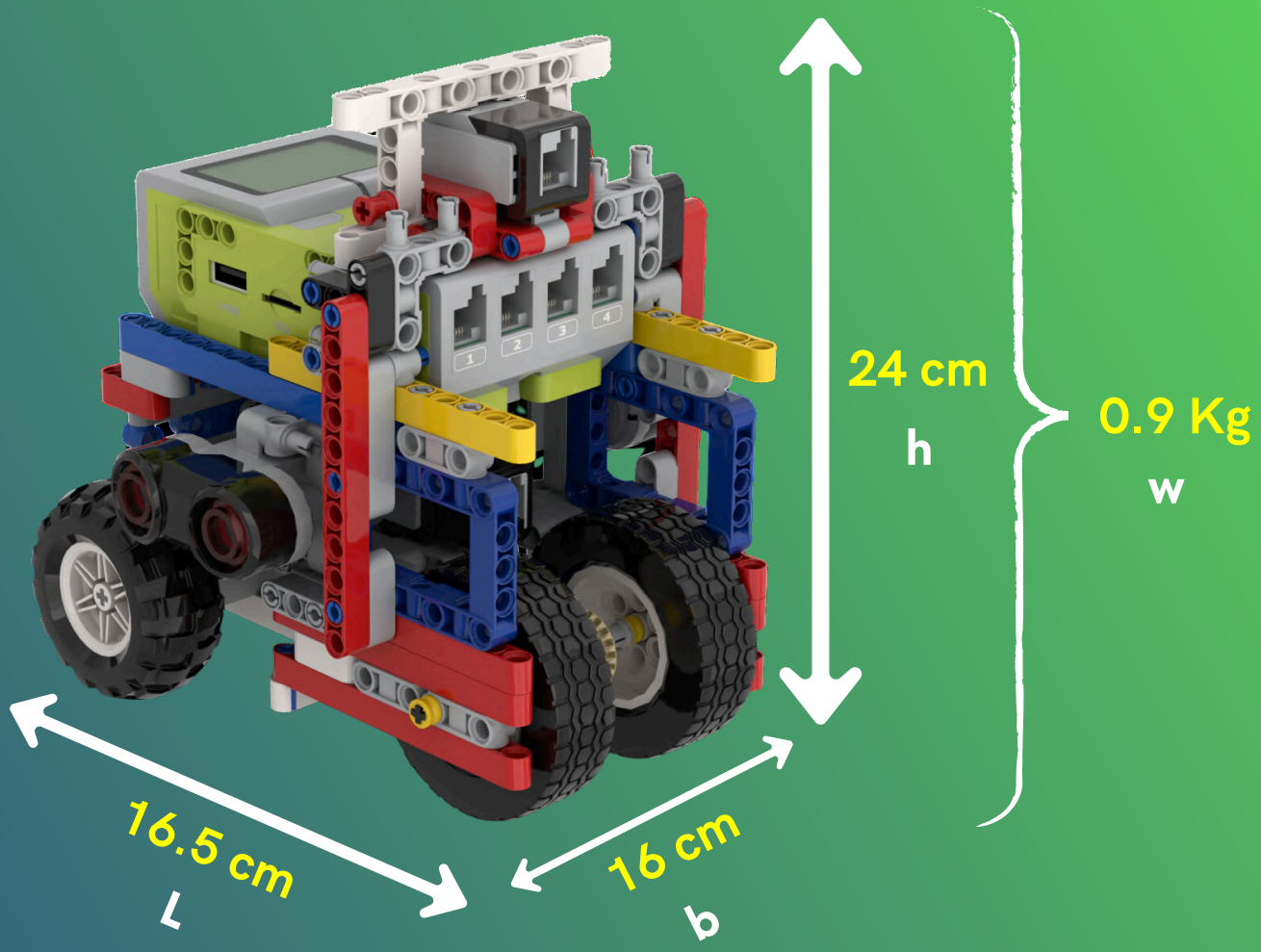
OPPORTUNITIES

- Modular Design Improvements
- Enhanced Object Detection

THREATS

- Component Durability

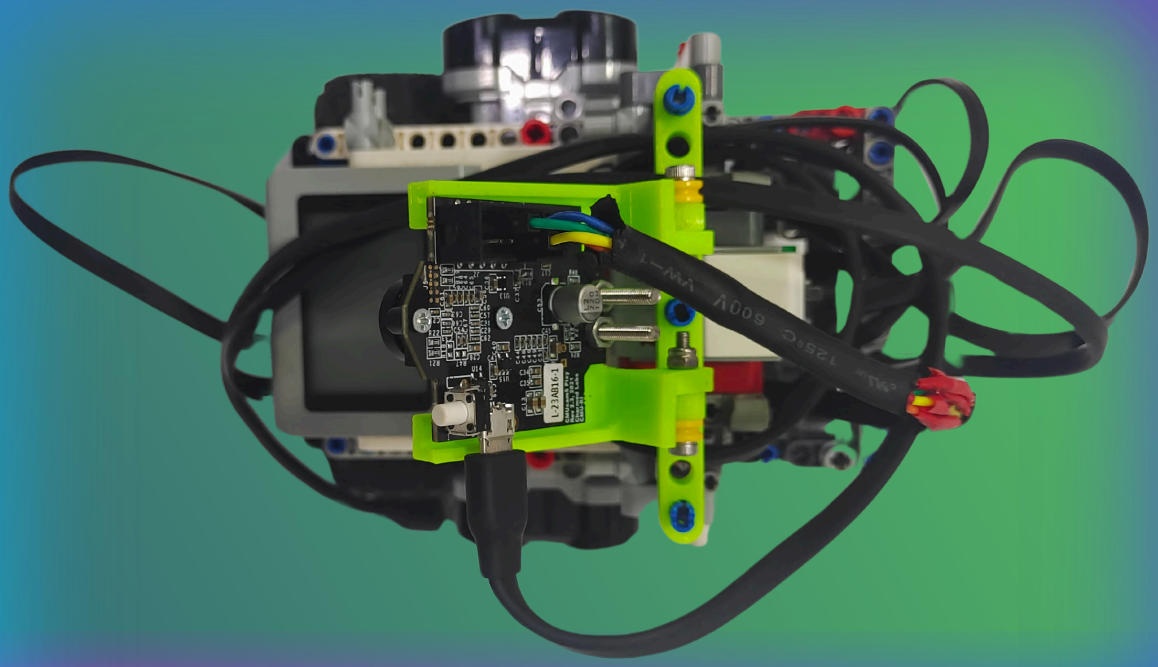
Our Robot

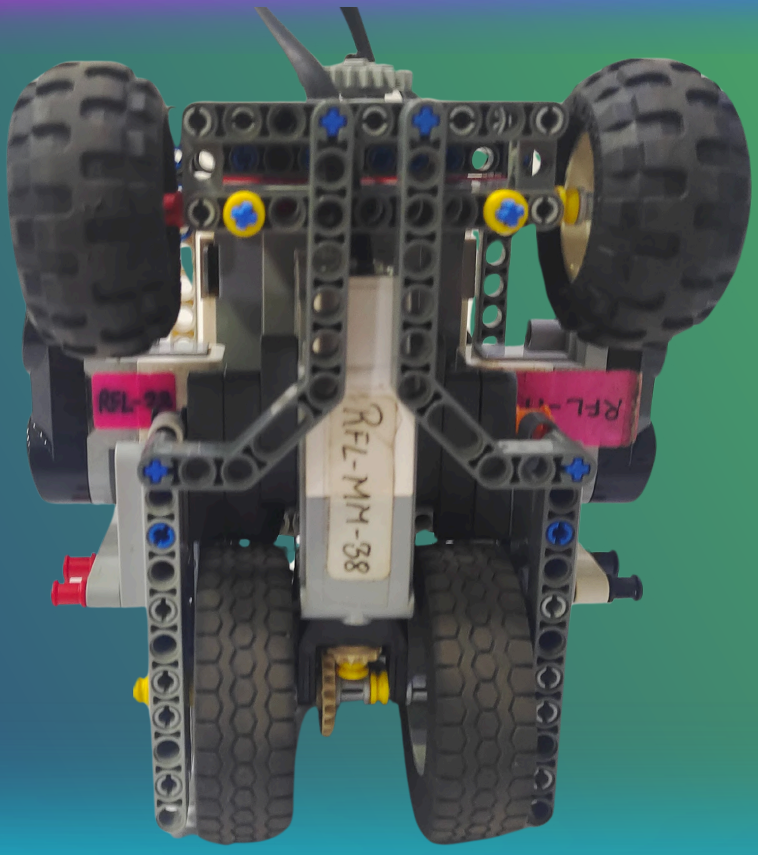


Turning Radius - 75 degree
Centre Distance between both wheels - 11 cm

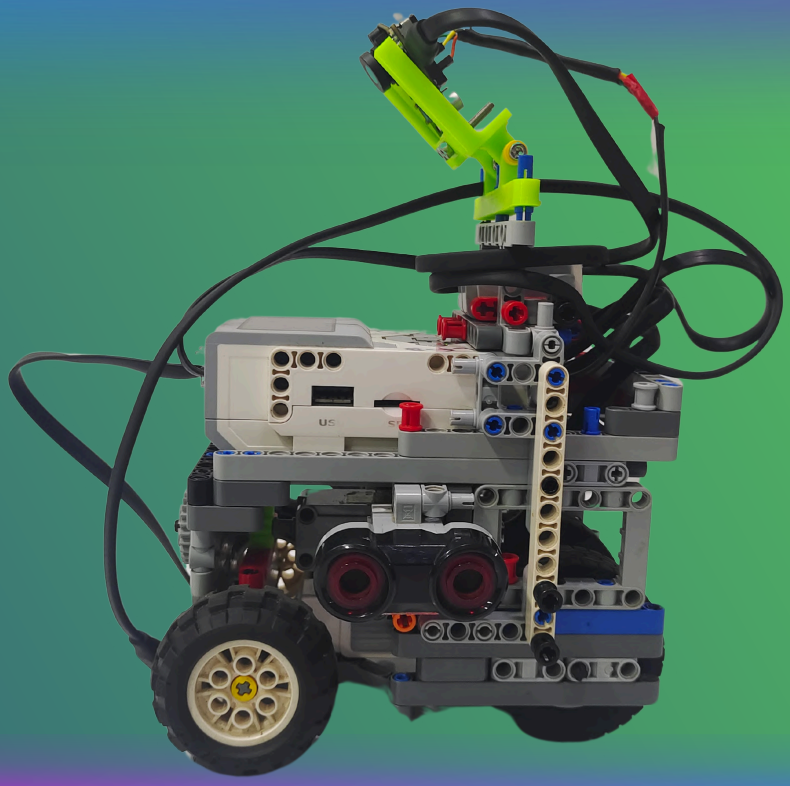
- **Motor Setup:** Two medium motors power the robot—one for rear wheels and one for steering.
- **Wheel Configuration:** Two Rear (62.4 x 20s) and steering wheels (62.3 x 21s) are mounted on single axles.
- **Steering System:** Custom 3D-printed gear and rack for precise steering movements.
- **Sensor Configuration:** Two LEGO ultrasonic sensors are mounted on the sides, and a gyro sensor is on top.
- **Camera Integration:** Pixy 2.1 camera is included for enhanced obstacle detection.
- **Control Unit:** LEGO Mindstorms EV3 brick manages all sensors and motors.

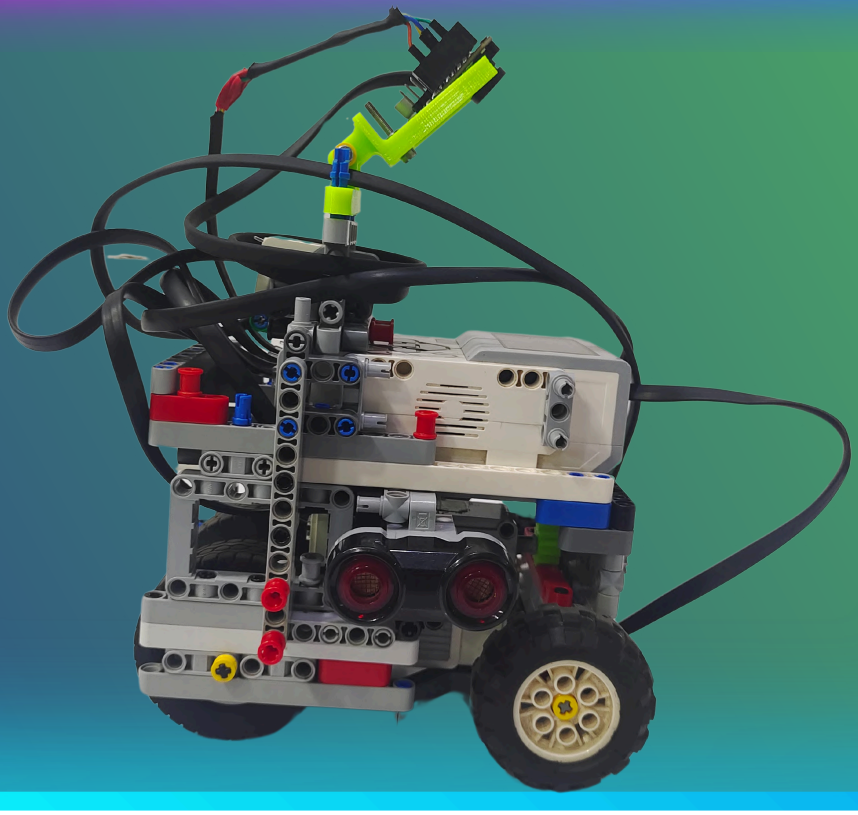
TOP & BOTTOM VIEW



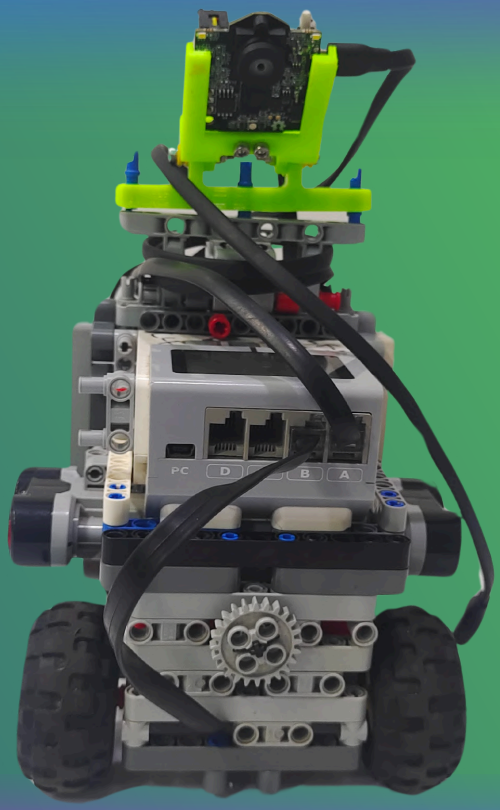


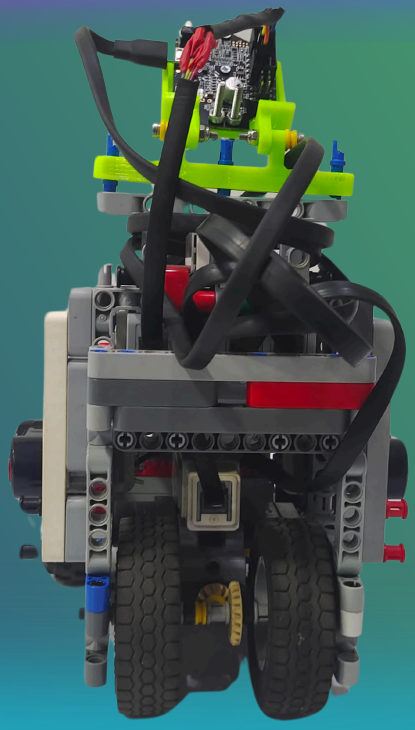
LEFT & RIGHT VIEW



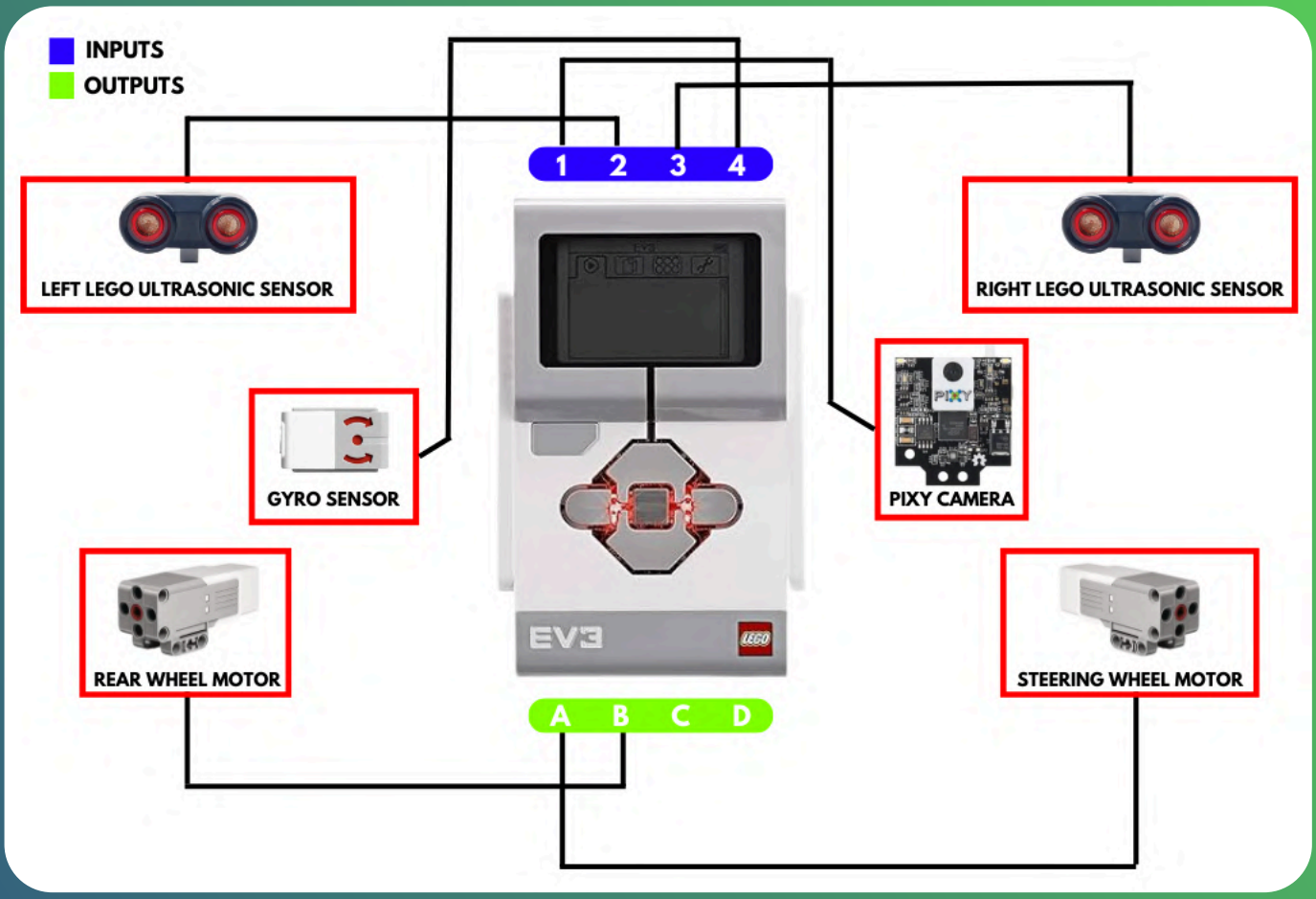


FRONT & BACK VIEW





Electricals



This diagram illustrates the wiring and components configuration of our robot

Central Unit - EV3 Brick

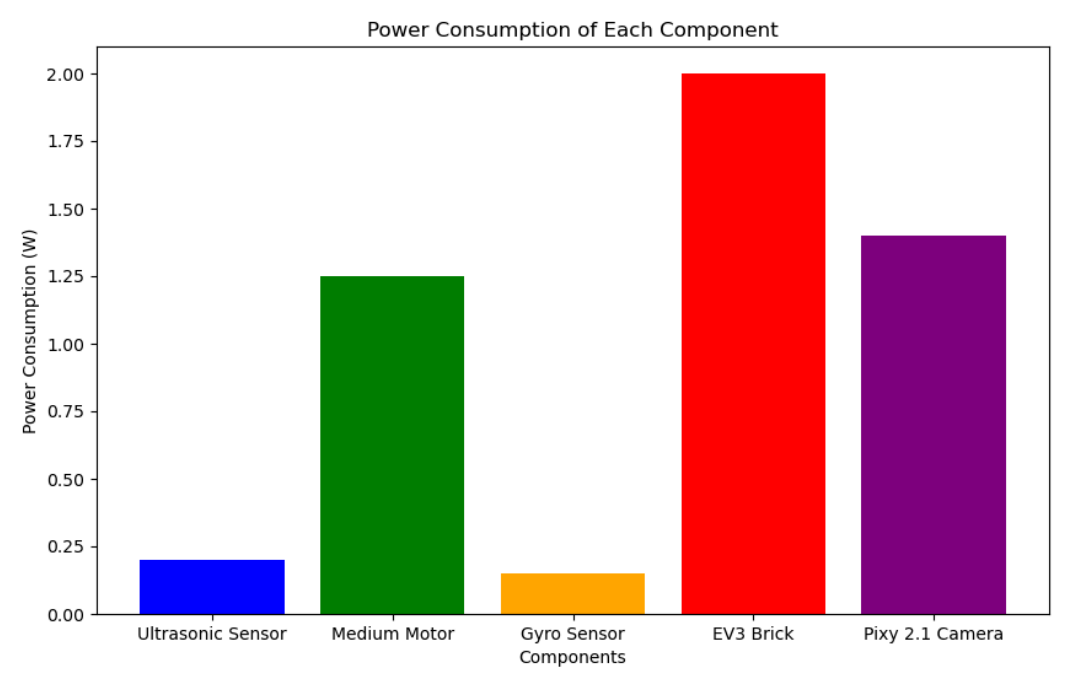
Input Ports (1-4)
- Port 1: Left LEGO Ultrasonic Sensor
- Port 2: Right LEGO Ultrasonic Sensor
- Port 3: Gyro Sensor
- Port 4: Pixy Camera

Output Ports (A-D)
- Port A: Rear Wheel Motor
- Port B: Steering Wheel Motor

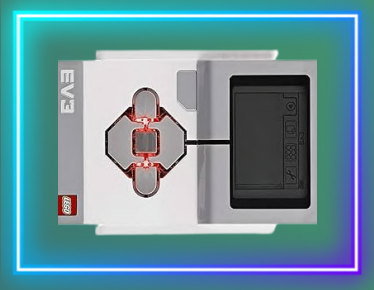
Power Management

This is our 2050 mAh DC Rechargeable Battery which provides power to the EV3 Brick and other components. Here's an estimated power consumption breakdown for each component:

1. **LEGO Ultrasonic Sensor:**
- Current Consumption: Approximately **20 mA**
- Power Consumption: Approximately **0.2W**
2. **LEGO Medium Motor:**
- Current Consumption: Between **100-150 mA**
- Power Consumption: Approximately **1.25W**
3. **LEGO Gyro Sensor:**
- Current Consumption: Approximately **10-15 mA**
- Power Consumption: Approximately **0.15W**
4. LEGO EV3 Brick:
- Current Consumption: Between **150-250 mA**
- Power Consumption: Approximately **2W**
5. **Pixy 2.1 Camera:**
- Current Consumption: Approximately **140 mA**
- Power Consumption: Approximately **1.4W**



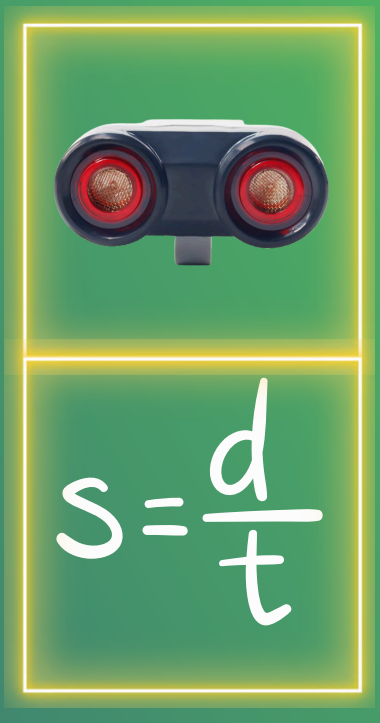
Components



The EV3 Brick is the central control unit of our robot.

This 2050 mAh battery provides power to the EV3 Brick and other components.

The medium servo motor is used for precise movements and rotations. It rotates at 250 rotations per minute (RPM)

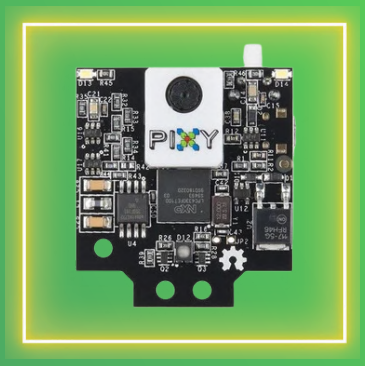


The sensors are mounted on **both sides** of the robot to accurately measure the **proximity** to adjacent walls. The sensor then measures the **round-trip time** for the reflected waves to return. By calculating the **time interval** between emission and reception, the sensor determines the **distance** between the robot and the wall using the **speed of sound** in air as a constant.

These cables help in building connections between motors, and sensors to the EV3 Brick.

Components

The PIXY 2.1 Camera is an **advanced vision** camera that can detect and track objects based on **color**. It provides real-time image **processing** capabilities and is used for object detection in obstacle challenge.



These rear wheels are **62.4 mm** in diameter and **20 mm** in width.

Center distance between steering wheels & rear wheels is 11 cm

These wheels are used as front wheels in our robot. These wheels are **56 mm** in diameter and **26 mm** in width.

The Turning Radius of steering wheels is 75 degree

The sensor is mounted on the **top** of the robot, it enables precise control over the robot's **orientation** adjustments and stability by detecting minute changes in angular displacement. It plays an essential role in trajectory tracking by calculating and storing the robot's **initial position**. Over the course of **three** laps, the gyro ensures that the robot consistently **returns** to its **starting** position.

Programming Timeline

In our first two iterations, we used a 3D-printed chassis to explore customization and new design possibilities.

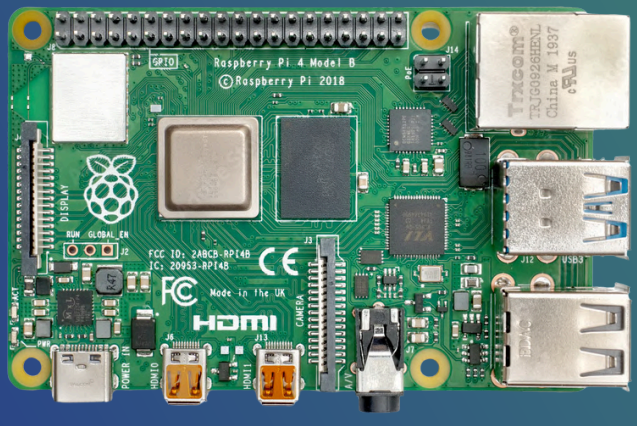
For processing and control, we chose the Raspberry Pi, despite it being our first time working with its schematics and programming interface.

To simplify our workflow, we researched methods to control the **Raspberry Pi** remotely, eliminating the need for peripherals like a monitor or keyboard.

Initially, we used **RealVNC Viewer** for remote access, but the virtual environment's lag hindered performance &efficiency.

We later improved by using **OBS Studio** with an **HDMI converter** to display the Raspberry Pi interface on a primary monitor for better control.

Being new to the Raspberry Pi, we faced challenges gathering resources, compiling code, and configuring features, which required extensive effort.



Raspberry Pi 4

Real VNC

OBS Studio

Programming Timeline

After over **30 days of working** with the Raspberry Pi, we identified ways to improve performance by incorporating an **Arduino board** for task distribution.

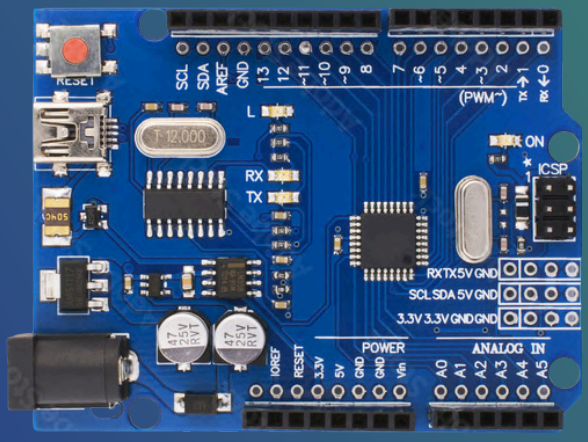
This integration allowed us to optimize the robot's functionality, achieving our desired performance goals.

However, we encountered space limitations in fitting components, wiring, and the power distribution board within the robot's size constraints.

After evaluating the construction & programming challenges, we decided to switch to the **LEGO EV3 system.**

Our familiarity with EV3 enabled a smoother, more efficient transition, improving our workflow.

The **EV3 Mindstorms platform,** combined with the **PixyCam**, allowed for better efficiency and flexibility in customizing the robot's design.

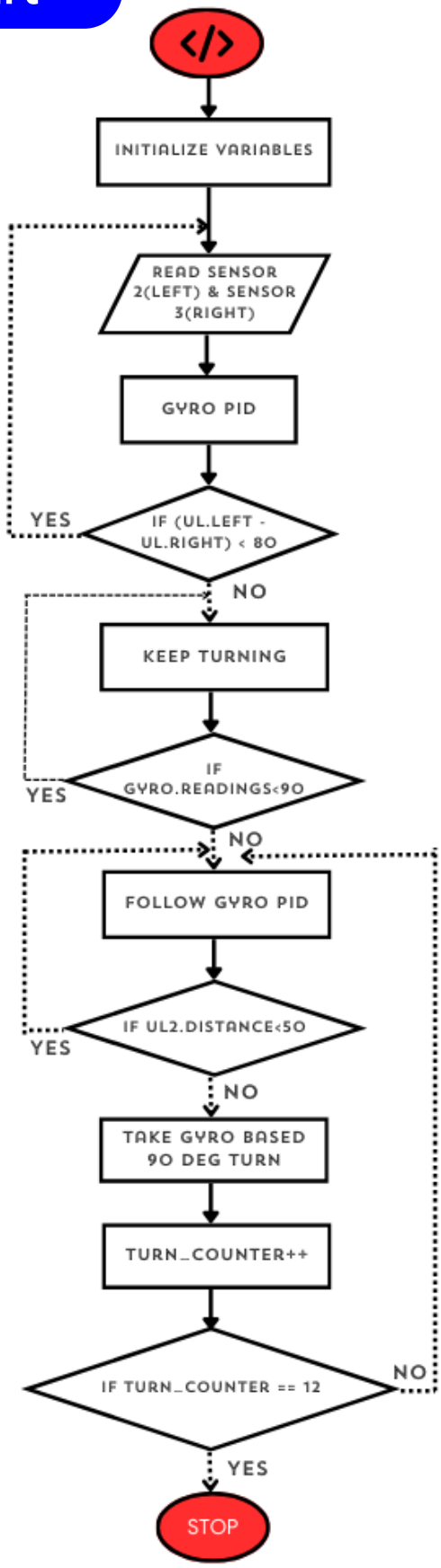


Arduino UNO

LEGO Minstorm

Open Round Challenge

Flowchart



This loop is to check what side the wall the missing so that the robot knows what turn to take

The robot keeps on performing the GYRO PID until the left wall is no longer sensed

Each time the robot takes a turn, the counter increments by 1, When there are 12 turns that are completed, i.e. 3 laps are finished the robot comes out of the loop and stops

Open Round Challenge

The robot begins its journey by determining which direction to turn using **Ultrasonic Sensors** on both sides to detect the absence of walls. As long as the average sensor reading is **below 80**, indicating walls on both sides, the robot moves **straight**, guided by the **Gyro Sensor**.

When the reading **exceeds 80**, indicating a missing wall, the robot checks if the **UltrasonicLeft** reading is **above 100** to decide the direction. A **left turn** is made if true; otherwise, a **right turn** is initiated. The Gyro Sensor ensures precise turning.

After the turn begins, the robot continues under **Gyro PID** control until the corresponding sensor (e.g., UltrasonicLeft after a left turn) detects a wall again, completing the turn. A **counter** tracks each turn, and when it reaches **12**, marking **three** laps, the robot stops.

Challenges Faced

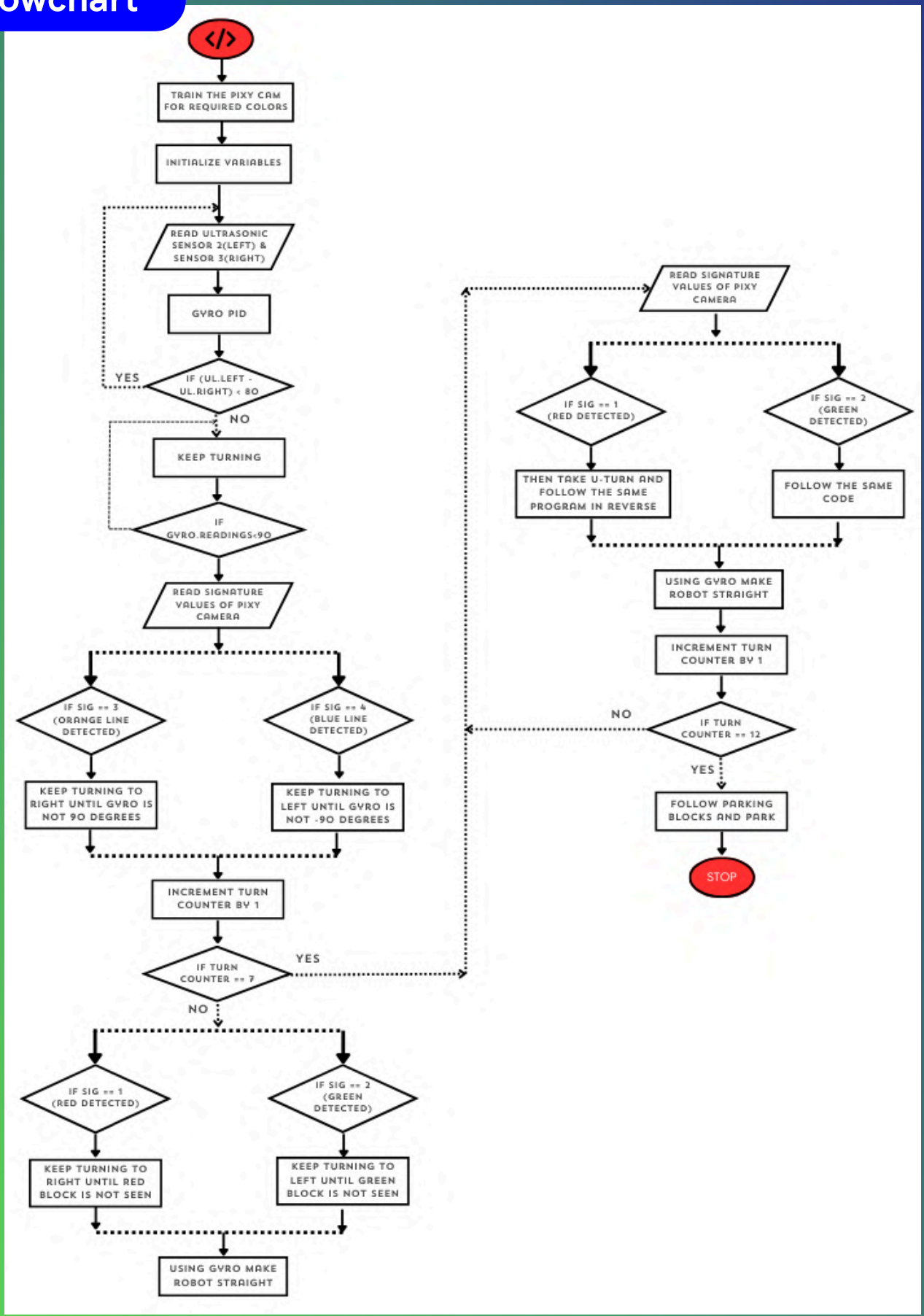
- **Inaccurate Turn Detection with Ultrasonic Sensors:** We encountered difficulties in detecting turns accurately due to the ultrasonic sensors occasionally providing false or inconsistent readings.
- **Maintaining a Straight Path Using Gyro PID:** Keeping the robot moving straight using Gyro PID control was challenging, especially when small disturbances caused the robot to drift.

Resolutions

- We improved sensor accuracy by **filtering** out erratic values using an **averaging** method or by setting specific **thresholds** to disregard outliers. This helped the robot make more consistent decisions about when to turn.
- We tuned the **PID values** carefully to minimize drift and oscillations, ensuring the robot maintained a **stable straight path** over the course of the laps.

Obstacle Round Challenge

Flowchart



Obstacle Round Challenge

The robot begins by using **Ultrasonic Sensors** to decide the turn direction. The **Pixy Camera** then checks for blocks **(Green or Red)**. If a block is detected, the robot moves closer until its area **exceeds 2000**. Red blocks trigger a **right turn**, and Green blocks trigger a **left turn.**

A separate loop checks for **Blue or Orange lines** to guide the robot's **Gyro-based 90-degree turns**. After **7 turns,** it stores detected elements, and after the **8th turn,** it decides on a **U-turn** based on the last block's color.

After completing **4 rounds**, the robot searches for the **Magenta parking** area and parks using motor encoder values.

Challenges Faced

- **Turning Without Collisions:** While making turns to avoid obstacles, we struggled to prevent the robot from colliding with objects or walls, as its turns were not always smooth or precise.
- **U-turn Logic and Block Detection:** Detecting blocks accurately for U-turn decisions posed difficulties. At times, the camera struggled to recognize blocks, leading to inconsistent U-turn execution.

Resolutions

- We refined the **robot's turn logic**, adjusting the speed and timing of the turns to give it enough clearance from obstacles. We also used the **Gyro Sensor** to ensure smoother and more controlled turns.
- We improved the **Pixy Camera's** block detection by **fine-tuning** the camera's **sensitivity** and ensuring the block's area was correctly calculated. This allowed the robot to make reliable **U-turn** decisions based on the **detected block color.**

Pixy Camera Detection

The Pixy camera utilizes color-based object detection and assigns specific identifiers based on the detected object's color and type. The classification scheme operates as follows:

Blocks:

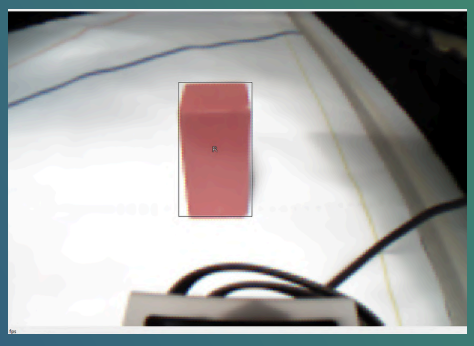
- **Green:** Represented as **'G'**, these objects are recognized as blocks with a green hue.
- **Red:** Represented as **'R'**, these objects are recognized as blocks with a red hue.

Lines:

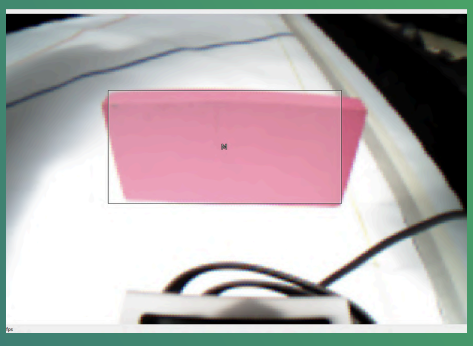
- **Orange:** Represented as **'O'**, these objects are identified as lines with an orange hue.
- **Blue:** Represented as **'B'**, these objects are identified as lines with a blue hue.

Parking Wall:

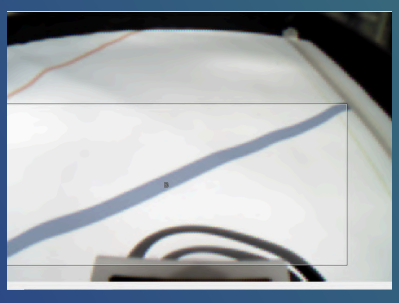
- **Magenta:** Represented as **'M'**, this object type indicates a parking wall with a magenta hue.



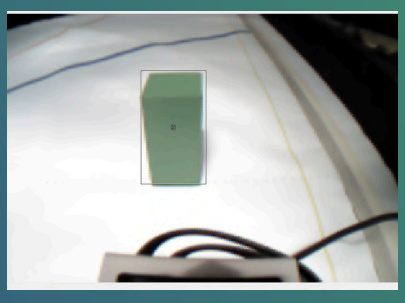
Red Block



Parking Wall



Blue Line

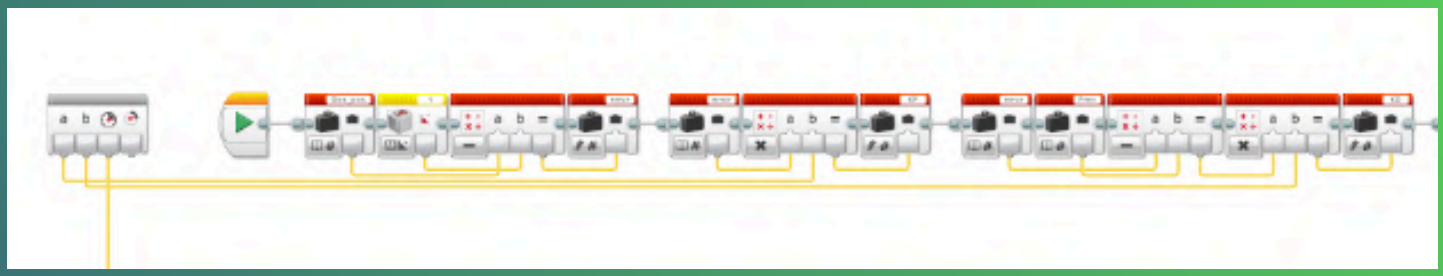


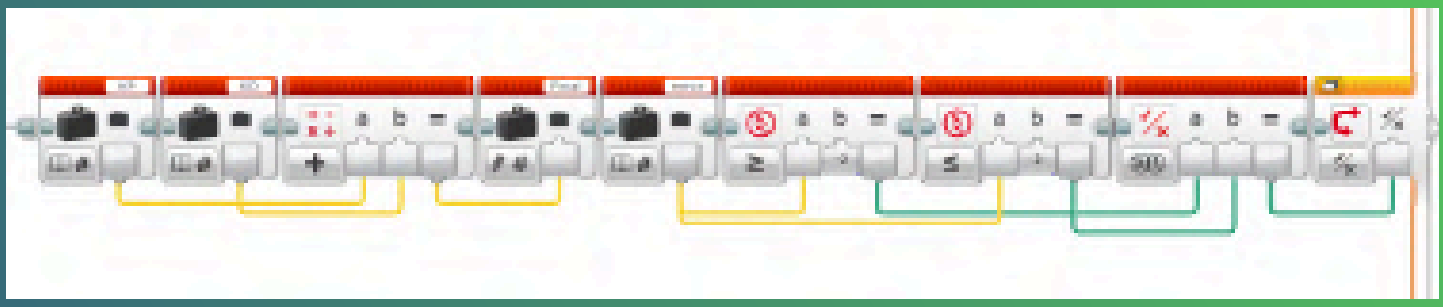
Green Block

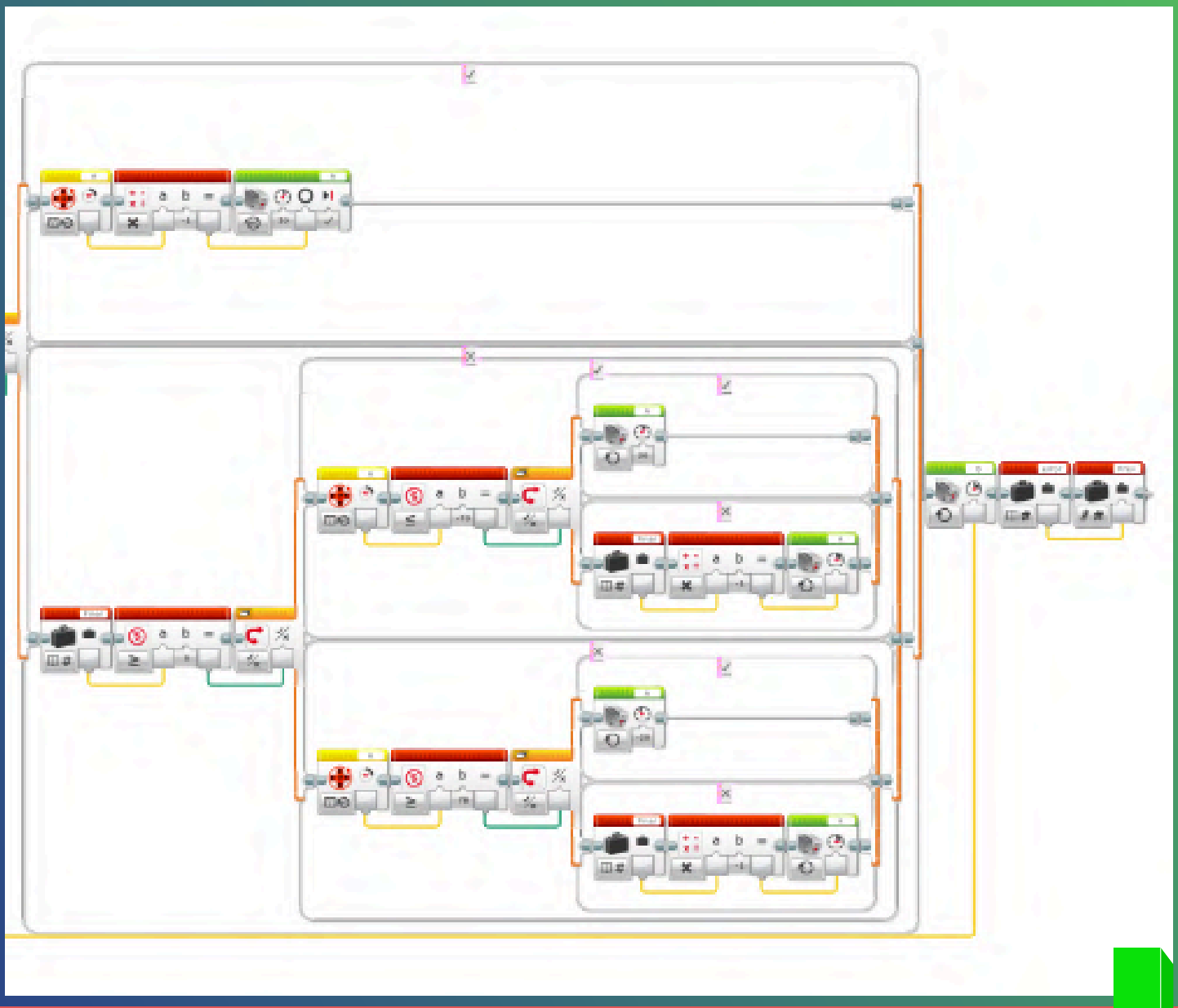
Orange Line

Custom Blocks

Gyro PID Block Flow

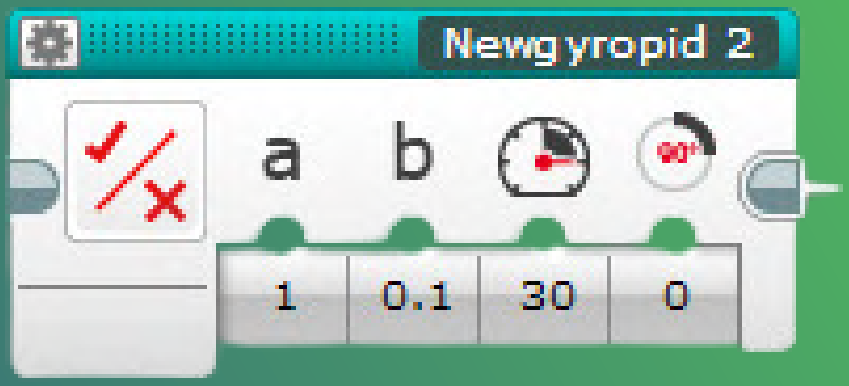






Custom Blocks

Gyro PID Block Algorithm

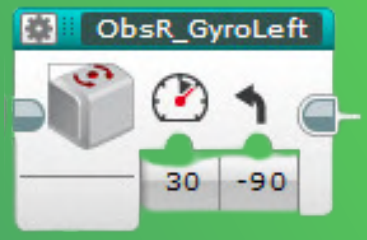


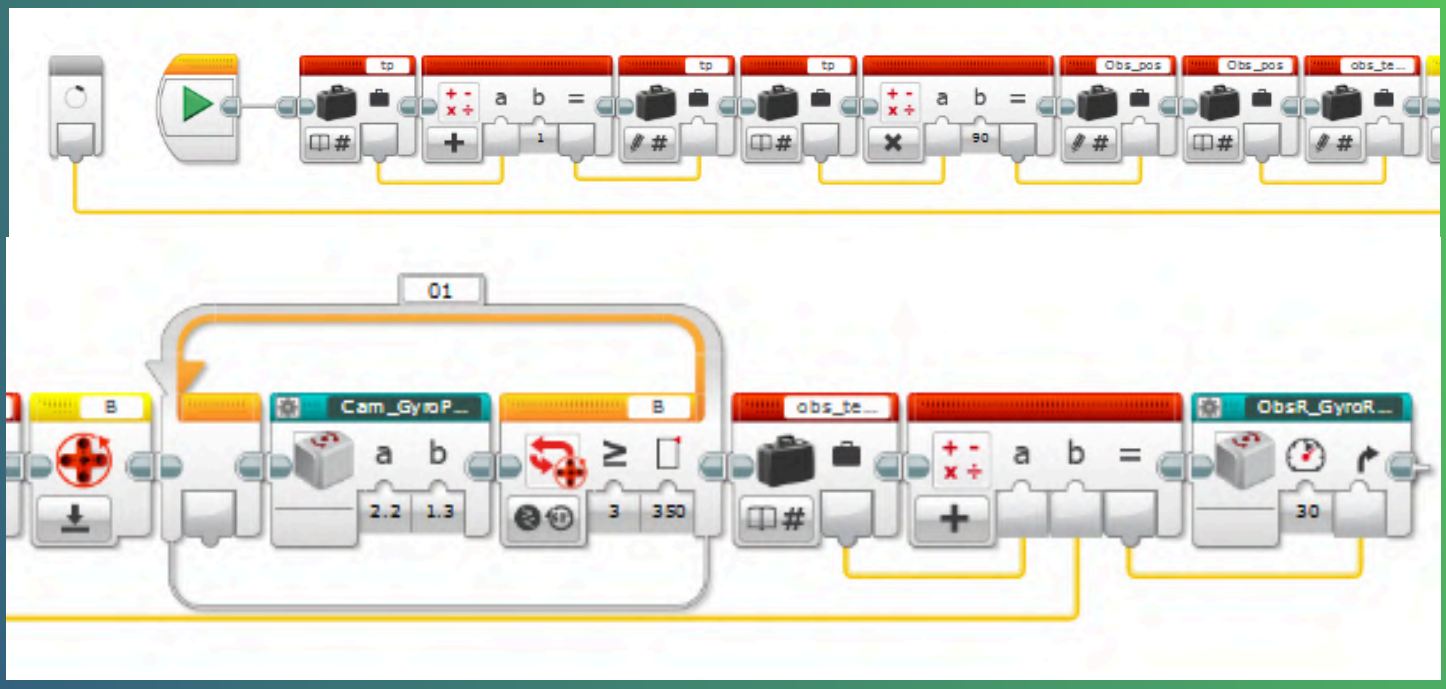
This block is responsible to keep the robot **driving straight** with respect to the **gyro sensor's value**. The calculation goes as follows:

- Gyro Sensor's current reading – Threshold value(set by the user) = Current Error
- Current Error * 0.8 = kp (The value 0.8 needs to be tuned with respect to each robot)
- (Current Error – Previous Error) * 0.5 = kd (The value 0.8 needs to be tuned with respect to each robot)
- Kp + kd = final_value (This value needs to be passed to the Steering motor's power)
- Assign the previous value as the current error for the next loop
- (previous_error = current_error)

Custom Blocks

Gyro Turn Left or Gyro Turn Right



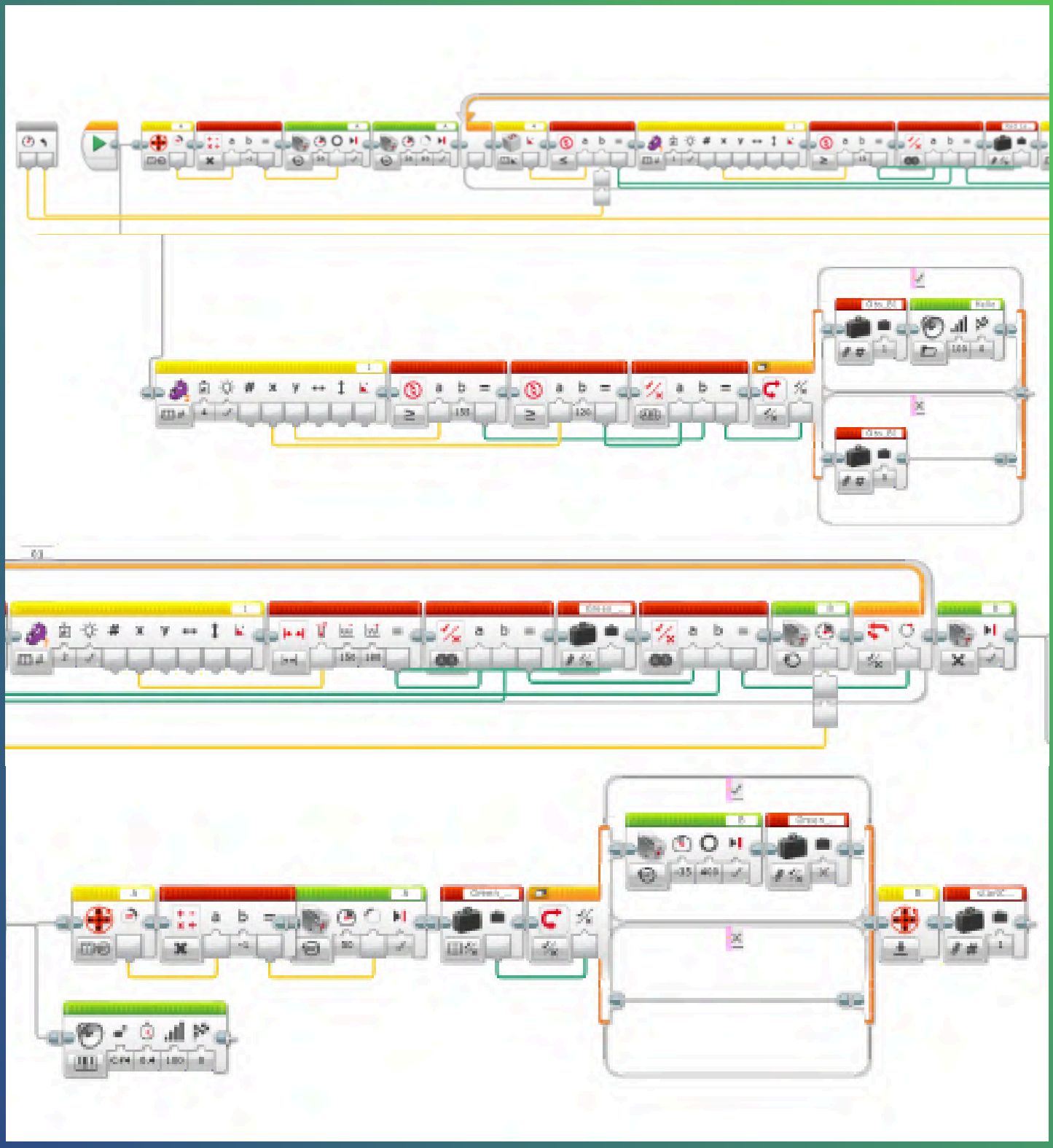


This block is used to ensure that the robot turns a **specific angle** that is the required **target position**. Here the current value of Gyro sensor is noted.

- Turn and Steering motor and keep the drive motor ON until the gyro sensor's value is not greater than the target value
- Run this in a loop
- Make the steering motor back to its original position
- (Motor Encoder value * -1) is given to the motor's target position to bring the steering motor back to its original position

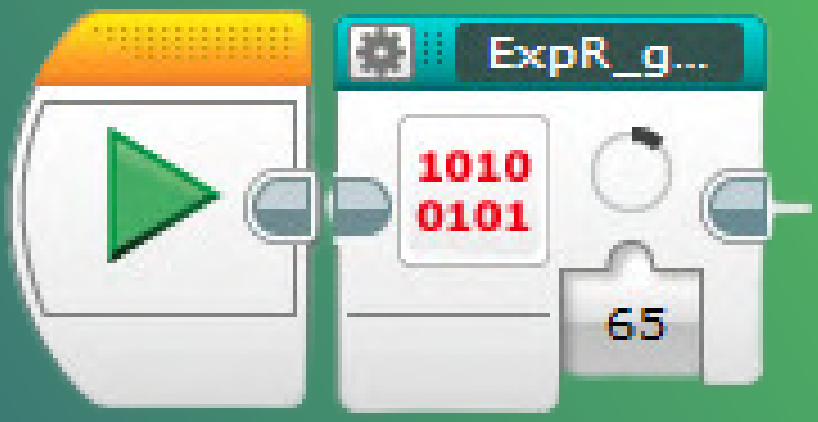
Custom Blocks

Obstacle Sense Flow



Custom Blocks

Obstacle Sense Algorithm

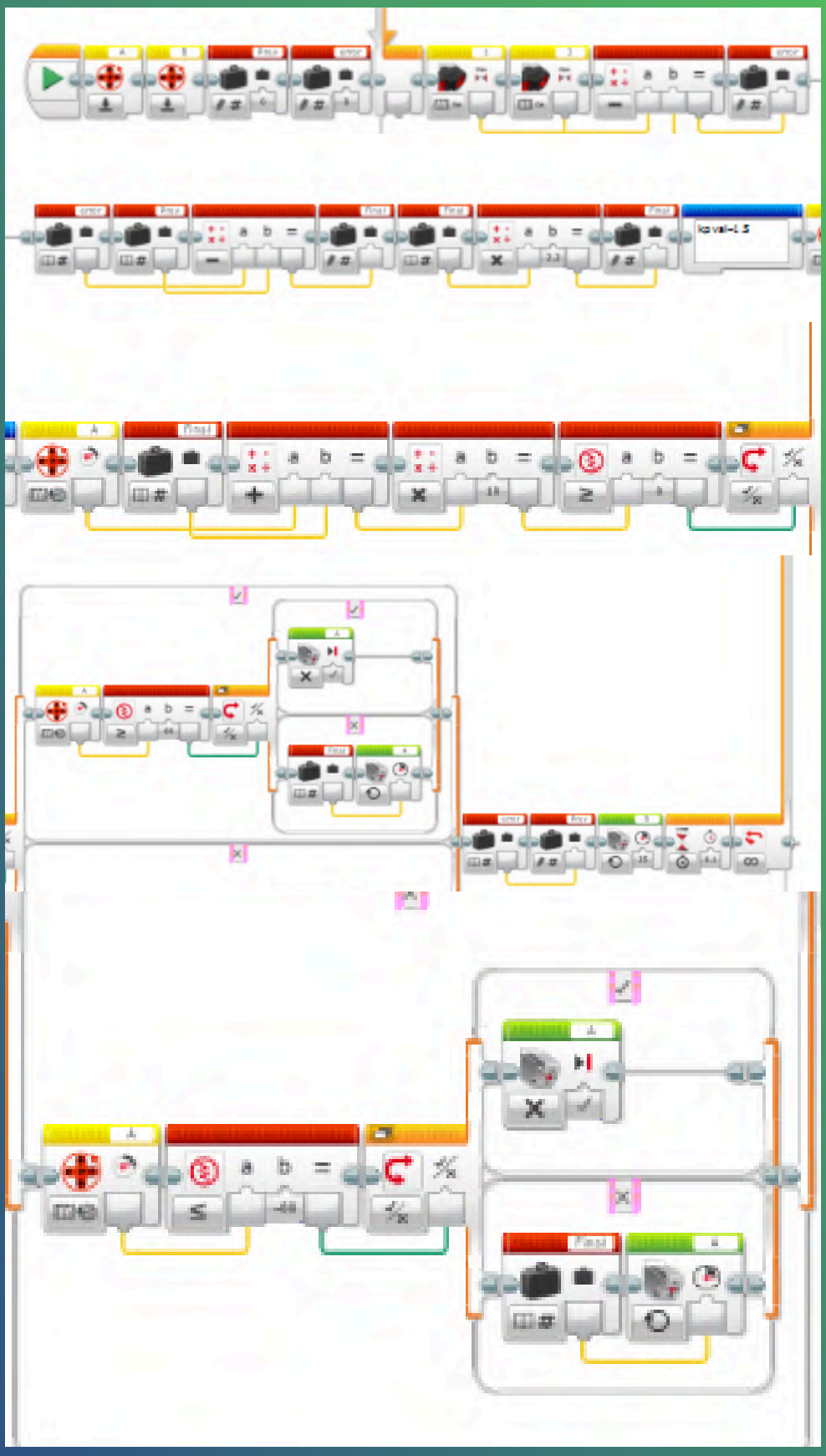


Here we calculate how much the robot has to turn and return to its value for passing an **obstacle (red or green)**

- Read the values of the detected object, Multiply the height and width of object to check how near it is to the robot
- When it is near to the robot, take a turn and drive the motor until The camera cannot see the block
- Return the Steering motor to its original positon and keep moving ahead

Custom Blocks

Ultrasonic Wall Sensor Follow PID



Custom Blocks

Ultrasonic Wall Sensor Follow PID

This block is responsible to keep the robot **driving straight** with respect to the **deviation of the robot** from both the side walls. We'll be using 2 sensors, 1 on left and the other on right for better accuracy. The calculation goes as follows:

- Left Ultrasonic Sensor's current reading – Right Ultrasonic Sensor's current reading = Current Error
- (Current Error – Previous Error) * 0.5 = magnified final value (The value 0.5 needs to be tuned with respect to each robot)
- This final value needs to be passed to the Steering motor's power
- Assign the previous error value as the current error value

Conclusion

The robot's **construction** and **programming** are optimized for high precision and efficiency, tailored to meet the requirements of both the **Open Round** and **Obstacle Round challenges.**

Construction

The robot's compact size **(16.5 cm x 16 cm x 24 cm)** and light weight **(0.9 kg)** ensure a balance of agility and stability. Powered by two medium LEGO motors, it controls rear-wheel propulsion and steering through a custom 3D-printed gear and rack system. Equipped with LEGO ultrasonic sensors, a gyro sensor, and a Pixy 2.1 camera, the robot effectively detects obstacles and navigates the course, including parking walls, with precision.

Open Round Challenge

In the Open Round, the program uses Ultrasonic Sensors and Gyro PID control to maintain alignment and execute precise turns. The robot detects missing walls **(sensor reading >80)** and turns left or right based on sensor data. With the Gyro sensor ensuring accuracy, the robot completes **12 turns over 3 laps in 65 seconds**, efficiently navigating using optimized sensor feedback.

Obstacle Round Challenge

In the Obstacle Round, the Pixy 2.1 camera detects colored blocks and lines, prompting the robot to turn based on the block color **(right for red, left for green)**. It continuously checks for blue or orange lines to adjust its direction, storing detected data in an array to guide decisions, including **U-turns after 8 turns**. The round concludes with the robot searching for a magenta parking area and using motor encoder values to park precisely. The entire Obstacle Round, **including parking, is completed in 140 seconds.**