Boston University
Electrical & Computer Engineering
EC463 Capstone Senior Design Project

First Prototype Testing Report

SmoothOperator

by

Team 30
SmoothOperator

Team Members

Celine Chen cchen@bu.edu
Eric Chen chene@bu.edu
Jacob Chin jacobc@bu.edu
Christian So cbso@bu.edu
Nicholas Nguyen nqnguyen@bu.edu

<u>**Equipment**</u>

Hardware:

- 4” Omni-wheels
- 2’x1’ Plywood Platform
- Aluminum C-Channels
- ESP-32
- VEX 393 Motors
- VEX Motor Controller 29
- 7.2V 3000 mAh NiMh Battery
- Power Bank (5v 2.4A)
- 2GB NVidia Jetson Nano
- Jumper wires
- Breadboard
- Ultrasonic Sensors
- Router
- USB Camera

Software:

- Node.js Server
- Nvidia Jetson Nano Client
- User Client
- ESP-IDF / Node.js Serial Communication code
- ESP-IDF ultrasonic safety code
- Python hand tracking with OpenCV
- SolidWorks

Set-Up

The SmoothOperator setup is divided into three main parts: wireless control of the robotic dolly, obstacle detection and tracking, and CAD motion analysis.

The wireless control system will integrate several hardware and software components. At the center is the Node.js server which will be hosted on a laptop; it will serve as a middle-man to communicate with the client interfaces and the Jetson Nano. The client side will involve manual inputs from the user for the robotic dolly to move around. The Jetson Nano will receive these commands and pass them to the ESP-32 via a serial connection. The ESP-32 will be responsible for communicating with the motors and ultrasonic sensors. Motors are programmed at a specific speed and perform turning maneuvers, while the ultrasonic will facilitate obstacle avoidance. This entire wireless communication system is isolated within our private subnet using a dedicated router, ensuring a reliable and secure connection.

The preliminary object tracking module is demonstrated with a USB camera connected to the Jetson Nano. Using a real-time video stream from the camera, the Jetson Nano will be able to detect and track the hand skeleton and movement of an individual, showcasing the ability to follow a user's gestures effectively. It will also be able to detect when the hand is made into a fist, similar to how a user would grab onto a handle of luggage.

The CAD motion analysis shows the mechanical design of the robotic dolly. Detailed simulations of the dolly's movement are performed to validate the design, providing insights into the structural and kinematic performance of the system under real-world conditions, which will guide our selection of motors and materials for fabrication.

Pre-testing Setup Procedure:

Updating movementClient.js and jetsonReceiver.js:

- Connect server laptop to NetGear network
    - SSID: NETGEAR52
    - Password: livelybanana831
- Go to 192.168.1.1 on the browser of the server laptop
- Go to “attached devices” tab and look for the IP address of the server laptop
- Copy that IP address to the appropriate line in movementClient.js and jetsonReceiver.js so they will attempt to connect to the server laptop

Network:

- Plug in router

Server-side:

- Connect to the NetGear network
- Run the node.js script on the server device.
    - CD to ../SmoothOperator/network/node-server (.. represents the SmoothOperator repo location)
    - Run node movementServer.js

Client-side (sends movement commands):

- Connect to the NetGear network
- Run the client.js script on the client device allowing the user to input commands.
    - CD to ../SmoothOperator/network/node-server (.. represents the SmoothOperator repo location)
    - Run node movementClient.js

Jetson Nano Client side:

- Wireless Communication
    - Connect to the NetGear network
    - Run the receiving.py script to listen to the server for commands.
        - CD to ~/Desktop/SmoothOperator/network/node-server
        - Run sudo chmod 666 /dev/ttyTHS1 (gives permissions to port)
        - Run node jetsonReceiver.js
- Object Detection
    - Plug-in webcam
    - Run image recognition Python script

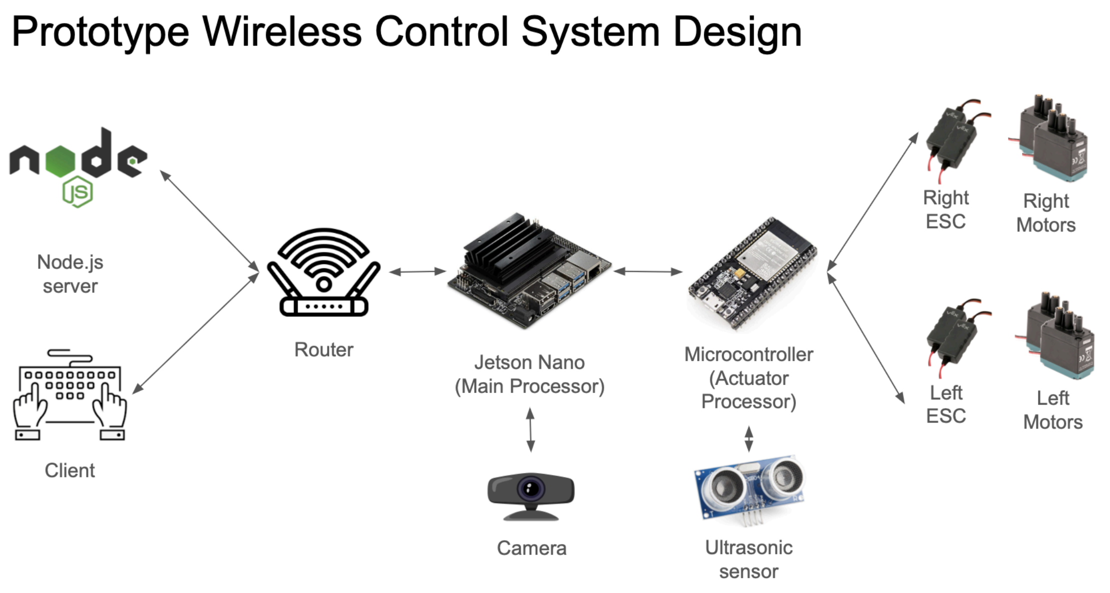
Prototype Wireless Control System Design
Node.js server
Client
Router
Jetson Nano (Main Processor)
Camera
Microcontroller (Actuator Processor)
Ultrasonic sensor
Right ESC
Right Motors
Left ESC
Left Motors

Figure 1. *System Diagram*

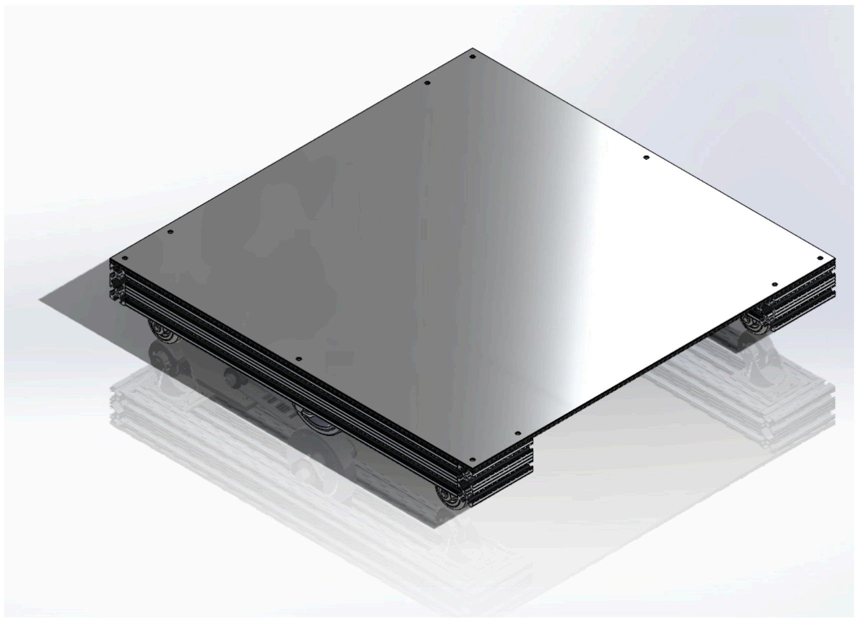

Figure 2. *Isometric view of the SmoothOperator smart dolly*

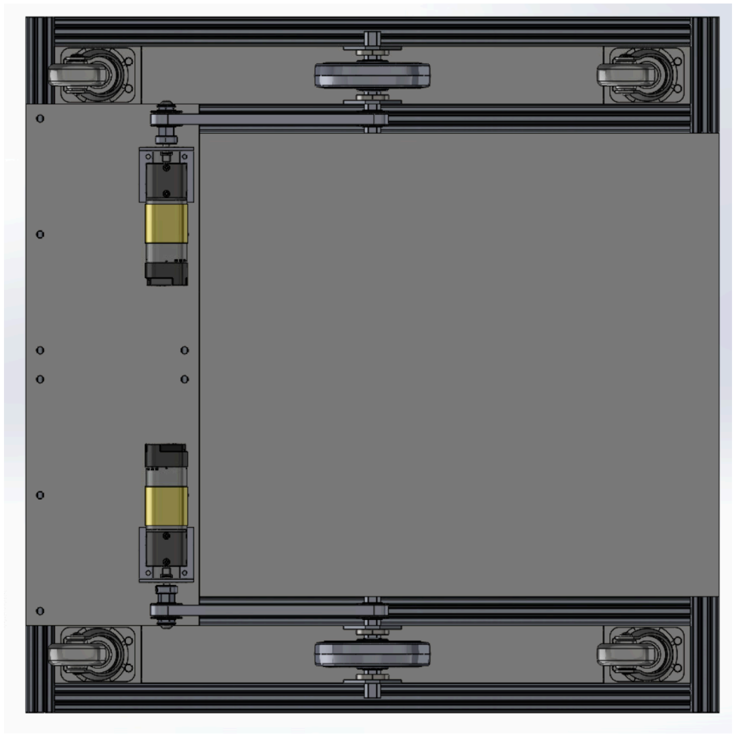

Figure 3. *Bottom view of the SmoothOperator smart dolly*

<u>Testing Procedure</u>

Mechanical Simulation Tests:

1. Apply sample weight to the platform in SolidWorks assembly.
2. Set weight of sample weight to 100 lbs.
3. Open motion study.
4. Activate SolidWorks Motion add-in to allow access to the motion analysis tab
5. Set rotary motor speed to triangular velocity-time graph with data points of 0 deg/s at 0 seconds, 360 deg/s at 1 second, and 0 deg/s at 2 seconds.
6. Calculate motion study.
7. Switch to the motion analysis tab and create a plot of the rotary motor to graph the motor torque over time.
8. Observe and record the resulting motor torque necessary for the robot to carry the sample weight at the specified velocity. This is the resulting motor torque necessary for a safety factor of 2.
9. Set weight of sample weight to 150 lbs.
10. Recalculate motion study.

11. Open plot of rotary motor, and observe and record the resulting necessary motor torque. This is the resulting motor torque for a safety factor of 3.

Hardware Test:

1. Place 50 lbs on the platform and demonstrate that the platform, axles, and wheels do not buckle or deflect.
2. Place two pieces of tape on the floor, about 2 meters apart. Place the robot dolly at the end of one tape. Make the robot traverse at full speed from one tape to the other tape and measure the time it takes. Divide 2 by the time to calculate the speed(m/s) of the dolly.
3. Check for external cables and wires to walls or computers. Should not have any wires hindering its movement.
4. Send left and right commands and monitor the center of mass. The center of mass should not move from its position while turning.

Software Test:

1. Turn on the power bank, attach the NiMH motor battery to the motors, and wait for the Jetson Nano to boot.
2. Connect to Jetson Nano and PC to our local network.
3. SSH connects to Jetson Nano via NoMachine to access Jetson Nano's file system
4. Start the Node server on the PC.
5. Start Node client on the Jetson Nano.
6. Start the Node client on the PC and press "A", "W", "S", "D", or "X" on the keyboard.
7. Show through NoMachine that the Jetson Nano is receiving these keyboard commands from the user.
8. These keyboard commands will be then fed to the ESP-32 via serial connection.
9. The ESP-32 will decode these keyboard commands and output the appropriate control signal for the four motors.
10. Demonstrate when the user sends a keyboard command, the motors will move wirelessly and the keys map to the appropriate actions (forward, backward, left turn, and right turn).
11. Put an object in front of the ultrasonic sensor and send a user command to the Jetson Nano to send to the ESP-32. Demonstrate that the ESP-32 overrides the commands and

does not move the motors when the object is within 30 cm of the robot to ensure safety.

12. For hand tracking, run the Python3 script for hand tracking and wait for the OpenCV window to show up.
13. Then move your hand in front of the camera and check if the green skeleton outline outlines your hand.
14. After, make a closed fist and see if it detects the gesture by monitoring the top left of the screen for “OK” to show up. If it detects the gesture as you do it, then it satisfies the metric.

Measurable Criteria

- ☐ Chassis sustains at least 50lb (at rest)
- ☐ Chassis is wireless through the local sub-network
- ☐ Chassis moves forward ~0.5 m/s
- ☐ Chassis turns left and right
- ☐ Jetson Nano receives commands from the keyboard
- ☐ ESP-32 receives commands from Jetson Nano via serial communication
- ☐ Motors reflect commands given keyboard inputs
- ☐ Chassis stops in front of obstacles $\leq$ 30cm
- ☐ Chassis CAD
- ☐ Chassis motion analysis shows the required numerics for motor and chassis material selection

Results

Mechanical Simulation Test Results:

For the mechanical simulation tests through SolidWorks’ motion analysis features we sought to measure the motor torque necessary for SmoothOperator to carry an applied load and transport it at a specified acceleration. We chose an acceleration of 360 deg/s^2 for the test and set a motor within the motion study to a triangular velocity-time graph that would represent this acceleration. When running the motion study with a 100 lbs weight applied to the top of the chassis, the resultant motor torque necessary to move at the desired acceleration was 9 lb-in (Figure. 4). Performing the motion study with a 150 lbs weight applied to the chassis resulted in a motor torque of 12 lb-in (Figure. 5).

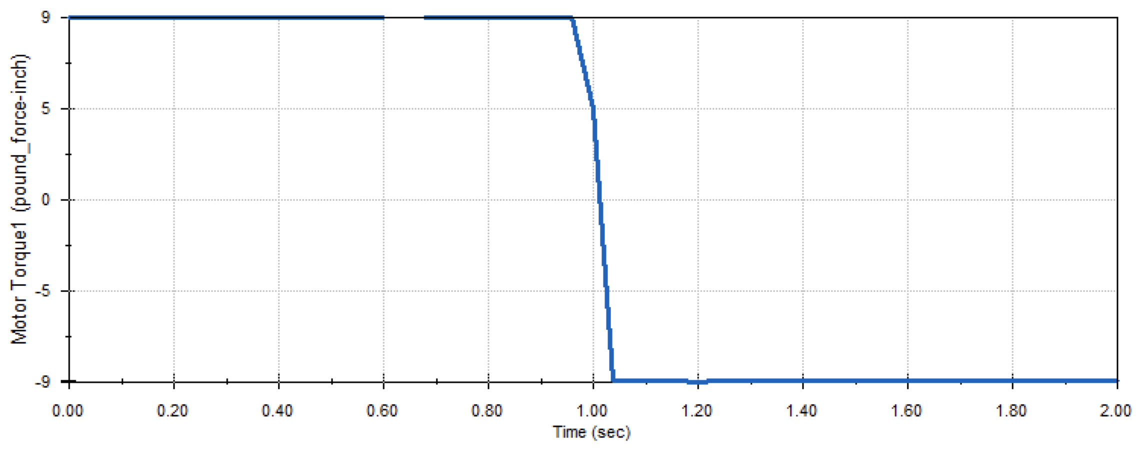

Figure 4. *Motor torque vs. time for a 100 lb applied load*

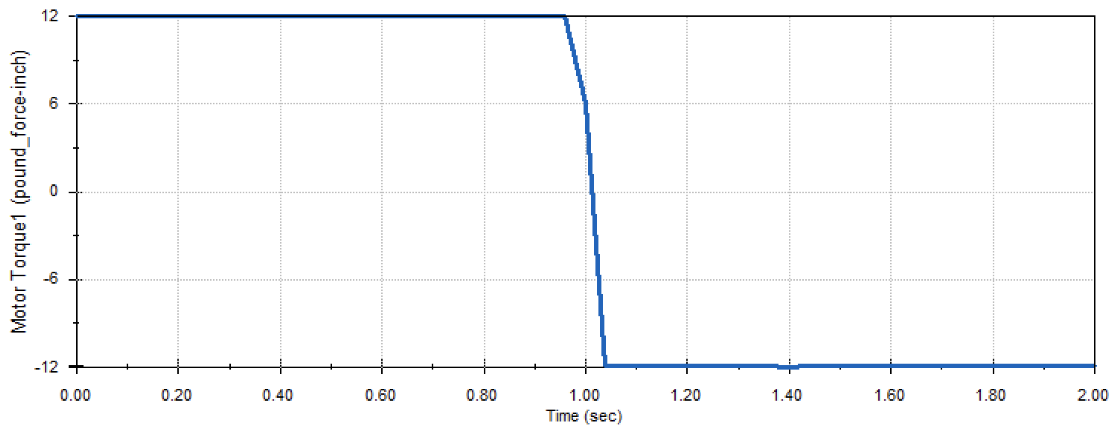

Figure 5. *Motor torque vs. time for a 150 lb applied load*

Hardware Test Results:

We conducted our hardware test in the Photonics Senior Design Lab. Eric (~130 lbs) stepped on the platform while holding someone else's hands for safety. This test can be seen in Figure 6. The board and axles of the wheels showed no visible deflection while at rest. Then, we sent commands from the keyboard which the ESP-32 decoded as turning either the left or right motors forward and backward and the other motor side in the opposite direction. The center of mass of the platform stayed in its absolute position while turning. With the same commands, we measured a 2m distance between two pieces of tape and timed how fast the robot traversed this distance and measured it. The robot completed the traversal in 3.85 seconds, achieving a speed of approximately 0.52 m/s. This performance exceeds the initially calculated projected speed of 0.5

m/s based on the current motor specifications. The test was successfully passed, demonstrating that the robot operates more efficiently than anticipated under the given conditions. The robot additionally was not connected to a wall outlet or tethered to any external computer that was not attached to the robot's local computing seen in Figure 7. Overall, the robot was able to satisfy all our set hardware metrics.

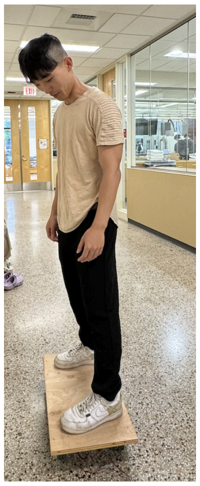

Figure 6. *Eric (~130 lbs) standing on our robot at rest showing no deflection or damage to the chassis.*

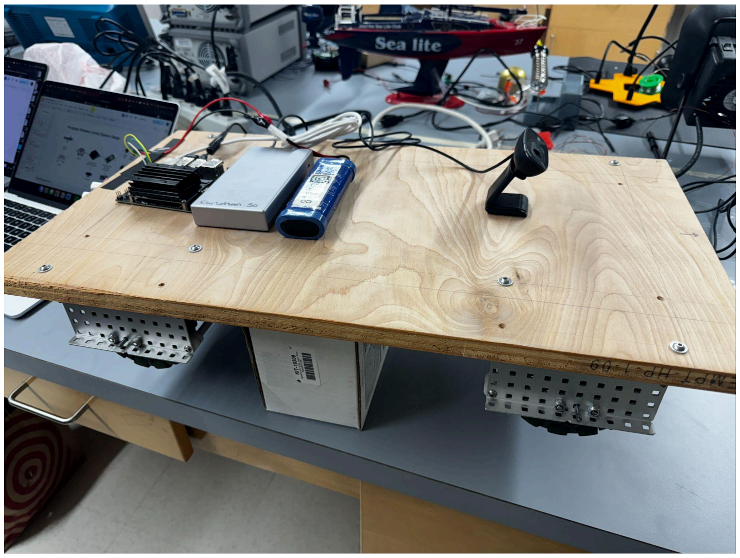

Figure 7. *The robot's wireless set-up with two batteries. One battery is for the processors and another is dedicated to the motors.*

Software Test Results:

First, we initialized the system by connecting a power source to the motor battery and the Jetson Nano. We then successfully linked both the user's laptop and the Jetson Nano to our local network to facilitate communication. For testing purposes, we set up a node server on the user's laptop and launched the node client on the Jetson Nano. We began transmitting keyboard commands such as "A", "W", "S", "D", and "X" and verified through NoMachine that the Jetson Nano was receiving these inputs. The commands from the client side are sent to the Jetson Nano, which relays the information to the ESP-32 via a serial connection. The ESP-32 decodes these commands and generates the appropriate PWM control signals for the motor controllers. Consequently, the motors respond by moving in the directions commanded: forward ("W"), backward ("S"), left ("A"), right ("D"), and stop ("X"). A safety mechanism utilizing an ultrasonic sensor was also incorporated. When an object was positioned within 30 cm of the sensor, it automatically overrode any user commands to prevent collisions. Lastly, we demonstrated hand tracking with the Jetson Nano, where the camera successfully detected fist gestures, as confirmed via NoMachine shown in Figure 8. Overall, our implementation satisfied all of our software metrics.

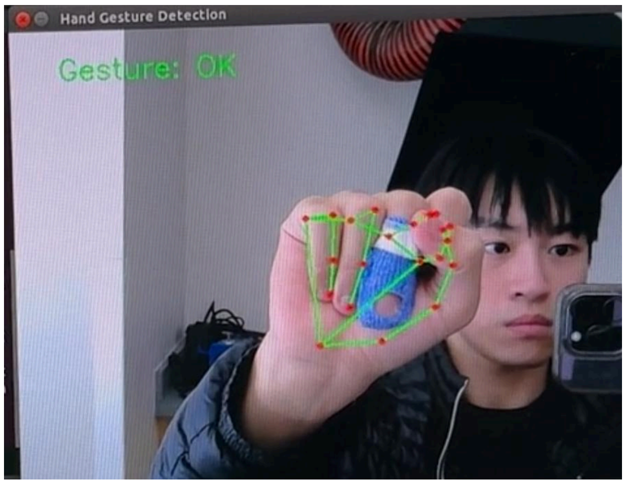

Figure 8. *Hand Tracking with Jetson Nano allows the camera to detect fist gestures.*

<u>Conclusions</u>

The first prototype testing of SmoothOperator demonstrated the successful integration and functionality of all core components of the system. The wireless communication setup, facilitated by a dedicated local subnet and controlled via the Node.js server, proved reliable, ensuring seamless interaction between the client sending commands, Jetson Nano, and ESP-32. The camera was able to detect simple fist gestures while our single ultrasonic allowed SmoothOperator to stop within 30 centimeters of any obstacles and override the Jetson Nano commands, demonstrating our safety system design. The preliminary CAD motion analysis validated the design, providing essential insights into motor and material selection while ensuring the chassis could sustain significant loads and move with appropriate speed and maneuverability. The next steps include moving on to fabricating our second prototype with a stronger, metal frame and the motors determined by our motion analysis. From the testing, it appears that the ultrasonic sensors are sufficient for our safety needs of stopping before the collision, so we will continue developing and incorporating the ultrasonic sensors in all directions. As our low-level control of the motors is sufficient, we can move on with high-level controls and planners for navigation. We will also go back to working on the LiDAR and looking into SLAM implementations. This will serve as the initial mapping processing step needed for both future navigation algorithms and potential updates for the safety layers. Overall, the testing confirmed that SmoothOperator meets key design specifications, including reliable wireless communication, effective obstacle avoidance, and precise motion control, setting a strong foundation for future enhancements and real-world deployment.

Score Sheet

Criteria	**Point**
Mechanical Design	2/**2**
Chassis CAD Design Complete	1/1
Chassis Motion Analysis for 100 lbs and 150 lbs	1/1
Hardware	4/**4**
Chassis moves ≥ 0.5m/s	1/1
Chassis turns right and left with zero degrees	1/1
Capacity of ≥ 50 pounds (at rest)	1/1
Chassis is wireless (not connected to a laptop or outlet)	1/1
Software	4/**4**
Stops in front of obstacle ≤ 30cm	1/1
Jetson Nano receives commands from server	1/1
Keyboard Teleoperation Control (‘WASDX’) of Chassis Movements	1/1
Jetson Nano with a USB camera tracks hand skeletons and detects closed fists.	1/1
Total Score:	10/**10**

Hardware Pinout

Pin	**Usage/Description**
ESP-32	
33	Right Motor
15	Left Motor
13	Ultrasonic Sensor Trigger

12	Ultrasonic Sensor Echo
17	UART TX Pin connected to Jetson Nano RX pin (Serial communication)
16	UART RX Pin connected to Jetson Nano TX pin (Serial communication)
Micro-USB	5v 2.4 A Power Supply
GND	Ground of Power Supply
NVIDIA Jetson Nano	
Micro-USB	5v 2.4 A Power Supply
USB 3.0	Web-camera
RX	Connected to ESP-32 TX pin (Serial communication)
TX	Connected to ESP-32 RX pin (Serial communication)
GND	Ground of Power Supply
Left Motors	
GND	Ground of 7.2V 3000 mAh NiMh Battery
Vin	Power of 7.2V 3000 mAh NiMh Battery
Signal	Connected to ESP-32 GPIO 15
Right Motors	
GND	Ground of 7.2V 3000 mAh NiMh Battery
Vin	Power of 7.2V 3000 mAh NiMh Battery
Signal	Connected to ESP-32 GPIO 33