AUTODESK Instructables

K3VN - a Furniture Moving Robot

By joshzbeckett in CircuitsRobots
Unpublished

Introduction: K3VN - a Furniture Moving Robot

K3VN is an autonomous robot developed by **Re-Space**, specialising in picking up furniture and moving it around a room. It can be controlled from an easy-to-use app on your phone!

K3VN's key features:

1. Manual Mapping of a Room
2. Automated Furniture Movement
3. Funiture Locating
4. Easy-to-use Phone App
5. Scissor Lift
6. Omni-Directional Drive
7. Centring Under Furniture

8. Custom PCB

Supplies

1. A1 RPLiDAR
2. Raspberry Pi 5 with Heat Sink and Fan
3. Micro SD Card 64GB
4. Picam
5. Pi Pico
6. PCB ordered using the Gerber file
7. 3x Omni-Directional Wheels (we used the 58mm options with the 6mm hubs)
8. 3x Encoded Motors
9. Stepper Motor - 23HS32-4004S
10. Stepper Driver - DRI0043
11. 2x Microswitches
12. 3x BTS7960 Motor Drivers
13. Threaded Rod (we use 140132-L-Tr10x2-1R-1000-C15)
14. 4x 8mm Guide Rods
15. 4x Linear Bearings (8mm ID 26mm OD)
16. Coupling (9mm to 8mm)
17. 3x Neopixel LED Strips

Step 1: Construct the Scissor Lift

To assemble the Scissor lift, you need to print out the following components:

1. Top Plate
2. 3x Outer Arms
3. 3x Inner Arms
4. Base
5. Small Slider
6. Slider
7. 22x Fasteners (or you can use 8mm bolts)

Once the parts are printed, couple the stepper motor to the threaded rod and attach the stepper motor to the base of the scissor lift.

Next, put two linear bearings into the slider and put it on the threaded rod. Then, put two guiding rods through the sliders' linear bearings and attach them to the base.

After that, attach the microswitches to the base.

Now that the base is fully assembled, it's time to attach the arms of the scissor lift. Attaching three of each is suggested, but any equal amount will work - the key thing to take note of is that the more you add, the higher it can go, but the more unstable it will be!

Once you have attached the arms you want, it's time to assemble the top plate and attach it. First attach the Picam to the top plate and screw it in. Next, attach the acrylic top plate to protect the Picam. After that, put two linear bearings in the small slider and two guide rods through the small slider and the top plate. Finally, fasten the arms to the top plate, and you have assembled the scissor lift!

Step 2: Construct the Mobile Base

To construct the mobile base, you need to laser cut the following plates:

1. Top plate
2. Middle plate
3. Base plate

And these 3d printed parts :

1. 3x Motor Mounts
2. 3x Middle Posts
3. 3x Lower Posts
4. 2x PCB Holder

First, attach the motors to the motor mounts and put them onto the base plate. Once you have mounted the motors, connect the wheels using the hubs linked in the supply section. After that screw on the lower posts to the base plate and screw the LiDAR PCB onto the base plate. Next, attach the middle plate above, mount the LiDAR into the desired spot, and attach the PCB connecting wires. Then attach the middle posts to the middle plate and close it off with the top plate. This will give you the basic assembly of the mobile base!

Step 3: Assemble PCB

To assemble the PCB you require the components from the BOM and solder them following the schematics provided.

Step 4: Import ROS Package and Low-level Logic

First, you need to install ROS 2 Jazzy Jalisco and clone the ROSSpace package into your src file in your workspace. To run the main system, you need to run the following command: ros2 launch respace launch.py. This will launch the main brain of the robot that communicates with the app!

For the low-level logic, all you need to do is upload main.ino to the Raspberry Pi Pico, and then you should be good to go!

Step 5: The App Setup

Prerequisites:

VSCode

https://code.visualstudio.com/download

Expo Go

https://expo.dev/go

Dependencies:

React Native

React Native uses the MIT License, found at https://github.com/facebook/react-native/blob/main/LICENSE

Expo

Expo uses the MIT License, found at https://github.com/expo/expo/blob/main/LICENSE

NodeJS

NodeJS uses the MIT License, found at https://github.com/nodejs/node

To setup the application side of the Re-Space project, you need to download the application via the GitHub project repository: https://github.falmouth.ac.uk/GA-Undergrad-Student-Work-24-25/ReSpace_3-24/tree/main/Re-Space-App

After downloading, open your selected code editor and select File -> Open Folder. Within the window, navigate to the project repository and select the "ReSpace_3-24" folder within the directory and click "Select Folder". This process selects the correct file path for the project. Once the code editor has loaded the project files, open a new terminal by selecting Terminal -> New Terminal.

In the terminal, paste the following command and press enter to build the application:

```
cd Re-Space-App; npm install; npx expo start -c
```

Once entered, the command will start building the application using Expo Go. First, it navigates to the application file path under "Re-Space-App". Next, it automatically installs the required packages using npm (NodeJS). Finally, it starts the application, clearing the cache in the process. During the building phase, the terminal will output many different logs and debug messages, displaying its progress, and eventually outputting a scannable QR code containing the URL for the expo application. On first startup, running this command will take longer as it downloads the required packages.

Before progressing on the application, ensure that your mobile device and computer are on the same WIFI network, otherwise the devices will not be able to communicate with one another. Once each device is on the same network, on your mobile device open the Expo Go application. Select the "Scan QR" button and scan the QR code located on your computer.

If successful, the application will begin building on your device. It will display the bundling progress on your application as well as in the terminal. Once complete, you will now have full access to the Re-Space application.

After scanning the QR code, if the application does not begin building and instead outputs an error, click on the terminal on your computer and enter the command "ctrl+c", pressing enter once done. Once the terminal stopped running, paste the following command in the terminal and press enter:

```
cd Re-Space-App; npm install; npx expo start --tunnel -c
```

After building, scan the new QR code with the Expo Go application.

Step 6: Construct the Dock

To construct the dock, you need to laser cut the following plates:

3x Side panels

2x Top panels

And these 3d printer parts:

4x 2-way slot holder

4x 1-way slot holder

Also, you will need:

Router

2x Raspberry Pi 4B

Once you have laser cut/ 3d printed all the materials, connect the side panels and the slot holders with bolts, leaving an opening for the robot. Connect the two top plates with 2 hinges, then connect the router and the Raspberry Pi to the front top plate.

The complete top plates, attached to the rest of the hub construction with hot glue.

Hub:

To setup the hub:

1. First install Raspberry Pi OS on a Raspberry Pi 4B or newer

2. Set its hostname to "respace-hub"

3. Transfer the files within the "server" directory in the respace github repository to the Pi

4. Run python swarm_logic.py which will in turn spin up hub_communication to allow communication between the app and robots

Cybersecurity: This system uses WebSocket Secure (WSS) to encrypt the communications handled by the hub. To enable WSS, the system uses its own DNS domain provided by DuckDNS.com. Using DuckDNS, the system was given the free DNS "respace-hub.duckdns.org. To get the required certificates, Let's Encrypt was used on Linux. Certbot was installed and the following command was entered:

sudo certbot certonly --standalone --staging -d respace-hub.duckdns.org

Once created, copy them across to the server system under Server -> keys.

Within DuckDNS, input the hub device's local IP address (192.128...) into the "current IP" input, then select "update ip". Once updated, enter the domain into the WebSocket Provider within the app and hub_communication in the format:

hub_communication:

wss://respace-hub.duckdns.org:8002

websocketProvider:

wss://respace-hub.duckdns.org:8002/app

com_handler:

wss://respace-hub.duckdns.org:8002/robot