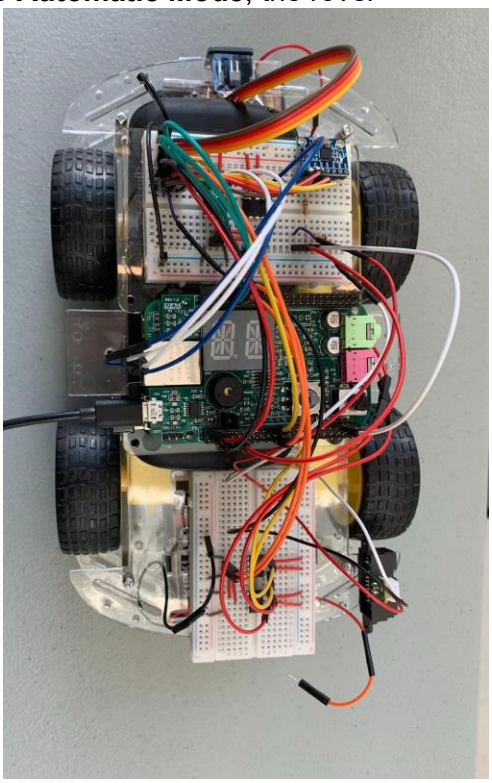
CMPT433 - PROJECT WRITE-UP

Team Smart-RC-Car:

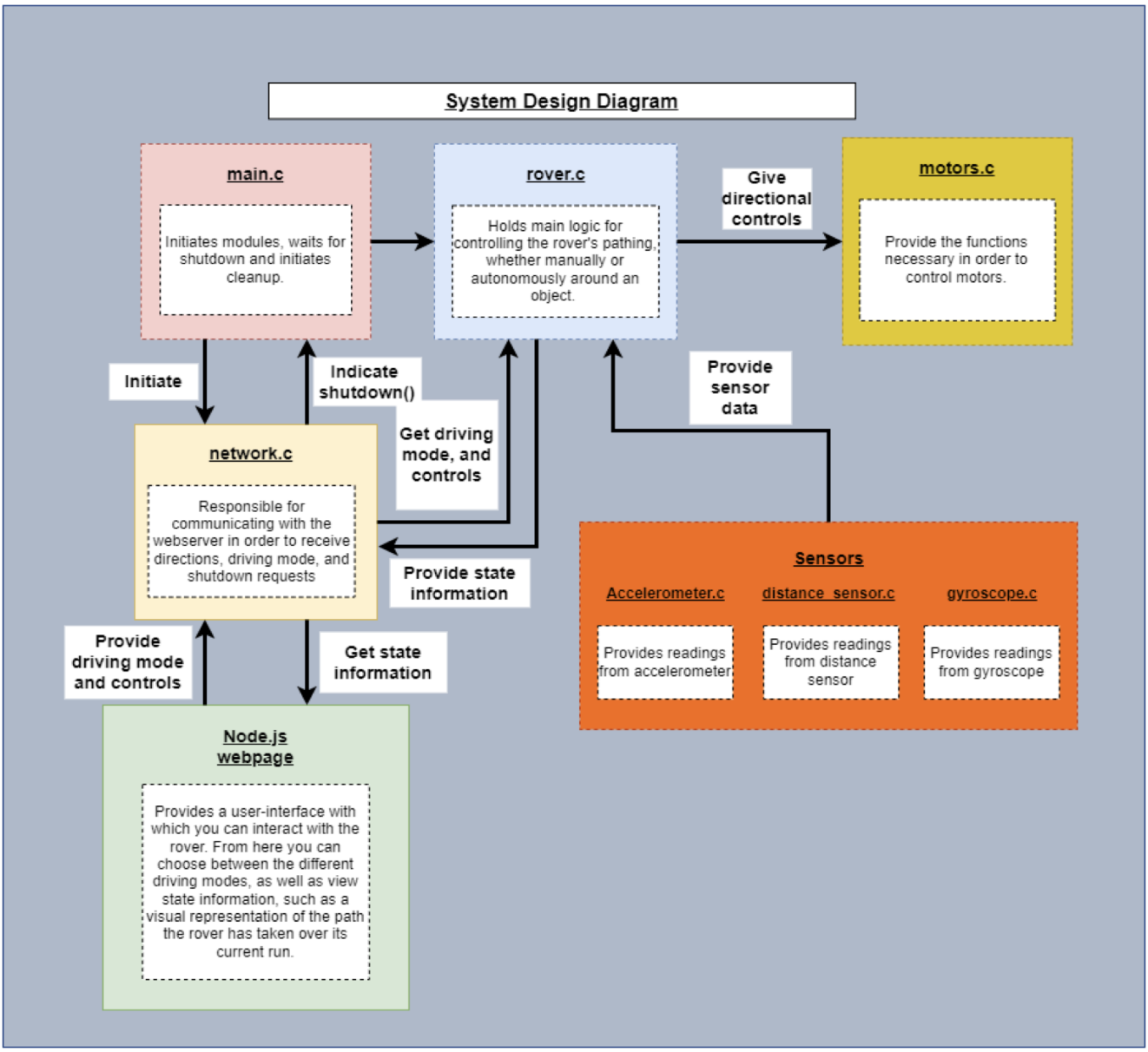
System Explanation

How the System works:

Our rover has two modes of operation: **Automatic** and **Manual**. When the rover is in **Manual Mode**, it can be controlled wirelessly over the network using a SNES controller (connected to another beaglebone). The controller allows it to go forwards/backwards, or pivot in-place in order to change directions. Using the select and start buttons, the user can alternate between the two modes, and initiate the automatic routine. In the **Automatic Mode**, the rover can drive itself by avoiding an object on its path. This is done using the **distance sensor** on the front, which stops the rover from collision using threshold, regardless of the mode the rover is in. The rover is able to keep track of its orientation using a **gyroscope**, which helps the rover rotate until it reaches the desired angle. We also use a second distance sensor, placed on the side of the rover. This lets the rover know when it has reached the boundaries of the object it was maneuvering around. We also have a **Web page** which provides another way to control the rover, as well as a map. The map is updated in real-time with the current orientation and displacement relative to the point it started at, as well as the path it took to get there. In order to operate the motors, we are using four L9110H motor driver ICs which allow them to run bi-directionally at up to 12V and handling up to 1.5A. We power the motors using a total of 8 AA batteries.



As the wheels lack the ability to tilt, motors from either side are synced in order to facilitate the rotation of the rover in the desired direction. The whole system, including the main app, the node.js webpage, as well as the joystick app, has been configured using systemd to launch at boot. Networking is also configured to auto-connect using a wifi-adapter at start-up to a pre-specified hotspot.



Things not working well

At the moment, the rover is hard-coded to travel a set path around an obstacle. This, of course, means that the automatic routine only works in ideal situations under controlled environments. . Furthermore, the rover lacks traction and power considering it is powered using weak, hobby motors. This means the rover physically stops with a slight delay upon software input causing slight displacement. We've adjusted for this gap between software and hardware through the use of sleep functions to simulate the extra movement inside the program. Also there is reduced maneuverability on non-smooth ground, eg. carpet due to the high amount of friction. In order to address this problem, we have implemented a routine which corrects the turn using micro-

adjustments. Finally, the web page interface can also be improved as the current web UI seems primitive.

Remaining/Future Challenges

- Make use of the OpenCV library along with the camera, for object detection
- Use some AI to improve the decision making of the car while in Auto mode ● Improve accuracy when more than one object is around.
- More subroutines for Rover (e.g. give it a predetermined destination and have it find a path there using AI)

Feature Table

H = Host, T= Target, O=Other

Description	H/T	Comp	Code	Notes
Wireless	T	5/2	connman/ C++	Tried setup wireless config through C++, unsuccessful;
Network/Network Controls	H/T	5	C++	(1) listen to UDP (2) periodically publishsystem state to subscribers
Motors	T	4	C++	Works, but one side is more powerful than the other causing it to drift off-center
Distance Sensors	T	4	C++	Detects object in path, avoiding crash
Joystick	T	5	C++	Controls the directions in manual mode, helps switch modes
Gyroscope	T	5	C++	Keeps track of orientation of the rover, helps in making turns.

Camera	T	2	C++	Stream video/Capture images to display on website; only tested on host; not enough USB port;
Led	T	5	C++	Flash Leds when the car enters the auto mode
Accelerometer	T	2	C++	Measure inertia; Very unreliable; drifts
Website	T	4	TypeScript / HTML / CSS	TCP between node server/clients, UDP from node to C; Transpile into JS; Bundles client JS when served; Used ExpressJS for http server
Run on start-up	T	5	systemd	Starts flawlessly

Extra Hardware/Software Used

- Connman: Command-line network manager ● Motors: Generic hobby motors
    - https://leeselectronic.com/en/product/47532-plastic-gear-motor-90-deg-out put-6v-dual-shaft.html
- Motor driver IC: Adafruit L9110H H-Bridge ● Distance sensors:
    1) Adafruit VL53L0X Time of Flight Distance Sensor
    2) Sharp GP2Y0A21YK0F IR Infrared Distance Sensor 10-80 cm
- Controller: SNES controller
- Wifi-adapter: Edimax EW-7822ULC
- Gyroscope: GY-521 MPU-6050 3-axis Gyroscope and accelerometer ● Node Modules:
    - ExpressJS: web application framework

    - P5.js: graphic context for displaying map
    - Browserify: bundle javascripts into single file to serve for client
- CMake: Maintained repository using CMakeLists.txt and setup cross compile