UIUC CS 437 INTERNET OF THINGS

REPORT FOR LAB 2 IoT Networking

Mariano Ovalle lo22@illinois.edu

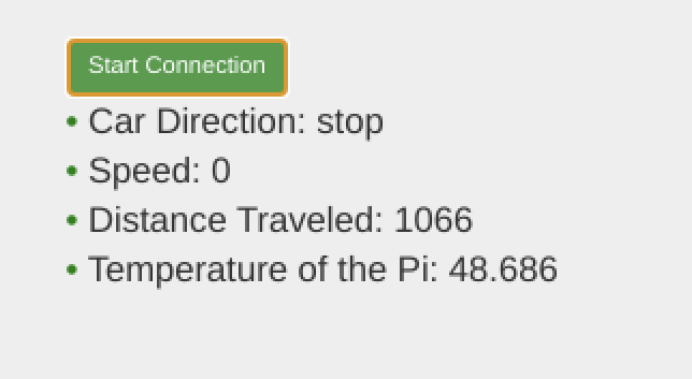
Introduction

We will be using Bluetooth and Wifi connectivity to explore the networking possibilities of IoT devices. In the first part, a Bluetooth connection sends instructions back and forth between a PC and the Raspberry Pi. We will also be sending commands to move the Picar. In the second part, we use a Wifi connection to create a Web Service, and a Front End will let the user see useful status information from the Picar and control its movements.

Functionality / Design

Part 1: Bluetooth networking: In this section, we establish a Bluetooth connection between the PC (client) and the Raspberry Pi (server). The client asks the user to input a message and sends it to the server. Then, the server receives the message and checks if is any of the following options: "**forward**", "**backward**", "**turn left**" or "**turn right**" in that case, it executes the requested Picar movement; otherwise, it just sends back the message to the client.

Part 2: Wifi / Web Service / Frontend: For this section, we extend the files index.js and index.html to create a front end that displays helpful information about the Picar and allows to control it with the keys "W" forward, "S" backward, "A" turn left, and "D" turn right.



The server side starts two threads; one continuously sends status information to the Frontend, and the other executes the Picar movements. The status is stored inside a dictionary:

```
server recv from:  ('192.168.1.30', 55642)
{'direction': 'stop', 'temp': '47.712', 'speed': '0', 'distance': '706'}
{"direction": "stop", "temp": "47.712", "speed": "0", "distance": "706"}
server recv from:  ('192.168.1.30', 55643)
{'direction': 'stop', 'temp': '47.712', 'speed': '0', 'distance': '706'}
{"direction": "stop", "temp": "47.712", "speed": "0", "distance": "706"}
server recv from:  ('192.168.1.30', 55644)
{'direction': 'stop', 'temp': '47.712', 'speed': '0', 'distance': '706'}
{"direction": "stop", "temp": "47.712", "speed": "0", "distance": "706"}
server recv from:  ('192.168.1.30', 55645)
{'direction': 'stop', 'temp': '47.225', 'speed': '0', 'distance': '706'}
{"direction": "stop", "temp": "47.225", "speed": "0", "distance": "706"}
```

Real Self Driving Car Protocols

In the case of a real self-driving car, I would use the following protocols:

Open/Lock car: RFID

There is no need for a large range or bandwidth in the case of opening/locking the car; usually, the driver is close to the vehicle when this happens, and the data used for this function is small. Also, the low power consumption is an advantage in this case.

Software upgrades: 5G

For software updates, we can use 5G. In this way, the SW can be upgraded everywhere a compatible cell provider is available, and the bandwidth is enough to handle large files. A Wifi option can be helpful too, but this would need the user to be near an AP.

Hands-free operation of cellphone: Bluetooth

The range is enough to connect a cell phone inside the car, and the bandwidth is large enough to carry audio for a phone conversation and even control some functions of the vehicle.

Camera feed: Wifi

This function needs a large bandwidth to continuously feed the camera video to the other systems in the vehicle. For example, a high refresh rate is required to avoid collisions.

Collision avoidance (localization of other vehicles): ZigBee

ZigBee offers an adequate range to communicate with cars in the vicinity and provides enough bandwidth for a high refreshing rate to track them at high speeds.