Beacon Tracking Using ESP32

Gautam Singh, G.V.V. Sharma
Indian Institute of Technology Hyderabad, India
Email: cs21btech11018@iith.ac.in, gadepall@ee.iith.ac.in

***Abstract*—This document is a report which demonstrates the use of machine learning in beacon tracking using an unmanned ground vehicle (UGV) and a WiFi-enabled microcontroller such as the ESP32.**

1 Assets

1) UGV chassis with DC motors
2) ESP32 microcontroller with Type-B USB cable
3) L293D Motor Driver IC
4) Breadboard and Jumper Wires
5) Android phone
6) (Optional) USB 2.0/3.0 Hub

2 Procedure

1) Make the connections as per the wiring diagram in Fig. 1.
2) Connect the ESP32 board to your Android Phone.
3) Generate the firmware by entering the following commands.

```
$ cd codes
$ pio run
```

4) Go to ArduinoDroid and select

```
Actions → Upload → Upload Precompiled
```

and choose the firmware file at

```
codes/.pio/build/firmware.hex
```

5) Now put the phone at a reasonable distance from the UGV with no obstacles in the way and then turn on the hotspot. The UGV should travel towards the phone and stop near it.

3 Working

To estimate (radial) distance to beacon, we use its signal strength. For WiFi, this is the **Received Signal Strength Indicator** (RSSI). The RSSI (in dBm) at radial distance of r metres is given by

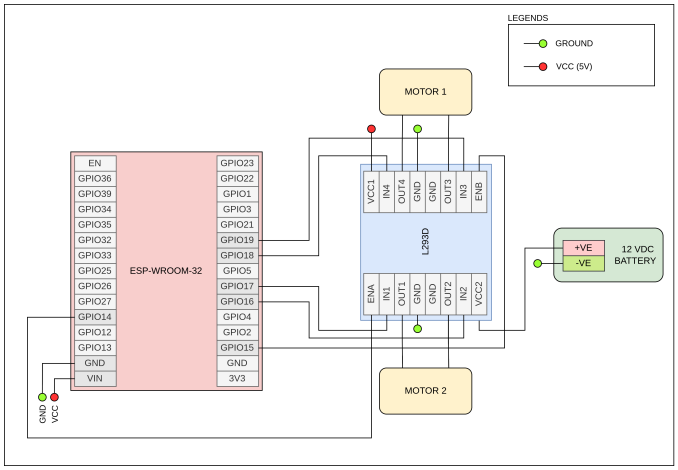

Fig. 1: Wiring Diagram for Beacon Tracking.

$$R(r) = R(1) - 10\log_{10}(r) \tag{1}$$

where $R(1)$ is the RSSI at a distance of 1 metre from the beacon. Clearly, $R(r)$ is a decreasing function as $\log_{10}(r)$ is an increasing function. Further, the second derivative of $R(r)$ is given by

$$\frac{d^2R(r)}{dr^2} = \frac{10}{\ln 10}\frac{1}{r^2} > 0. \tag{2}$$

Clearly, the RSSI is a convex function of the radial distance r. This implies that we can use gradient ascent to find the point where the RSSI is maximum, which would be the location of the beacon. The UGV uses a recursive algorithm to update its position using this principle until it is close enough to the beacon based on the RSSI measurements it takes at various points in its vicinity. The algorithm is described in Algorithm 1.

Algorithm 1 Beacon Tracking Algorithm

Input: RSSI threshold T, number of steps N

while GetRSSI() $< T$ **do**
    Take N steps in a straight line and measure the RSSI at each step.
    Suppose the maximum RSSI is measured at step i.
    Move to the position at step i
    if $i = N$ **then**
        Move one step forward.
    else if $i = 0$ **then**
        Move one step backward.
    else
        Turn left.

4 Observations

The UGV eventually converges close to the beacon (here, the hotspot). However, if there are a lot of nearby obstacles, the UGV may not converge close to the location of the beacon. It may either get physically blocked by the beacon or the signal interference may be too high.