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 Introduction

2 Related Work

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

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

where $R(1)$ is the RSSI at a distance of 1 metre from the beacon. The beacon tracking problem can be formulated as the following optimization problem.

$$\max_{r} R(r) \text{ s.t. } r > 0. \tag{2}$$

The derivative and second derivative of $R(r)$ is given by

$$R'(r) = -\frac{10}{\ln 10}\frac{1}{r}, \tag{3}$$

$$R''(r) = \frac{10}{\ln 10}\frac{1}{r^2} > 0. \tag{4}$$

Notice that for $r > 0$, $R'(r) < 0$, thus $R(r)$ is a decreasing function of r. This implies that the maximum RSSI is at $r = 0$, as expected. Since $R(r)$ is a convex function of r, we can use gradient ascent to recursively find the point where the RSSI is maximum, which would correspond to the location of the beacon. Using (3), the gradient ascent update equation is given by

$$r_{n+1} = r_n + \alpha R'(r_n) = r_n - \frac{10\alpha}{r_n \ln 10} \tag{5}$$

where r_i is the radial distance at the i-th step and α is the step size. Since $r_n > 0$, we can see that $r_{n+1} < r_n$ for all n. In other words, the radial distance to the beacon is decreasing with each step. This means that the UGV will converge towards the beacon. However, due to the explosion of the gradient in (3), the update steps become larger as r_n decreases, which could lead to overshooting the beacon. To prevent this, 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 the vicinity of its current position. The algorithm is described in Algorithm 1.

Algorithm 1 Beacon Tracking Algorithm

```
Input: RSSI threshold $T$, number of steps $N$
1: while GetRSSI() $< T$ do
2:     Take $N$ steps in a straight line and measure
       the RSSI at each step.
3:     Suppose the maximum RSSI is measured at
       step $i$.
4:     Move to the position at step $i$
5:     if $i = N$ then
6:         Move one step forward.
7:     else if $i = 0$ then
8:         Move one step backward.
9:     else
10:        Turn left.
```

4 Implementation

4.1 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

4.2 Procedure

1) Make the connections as per the wiring diagram in Figure 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.

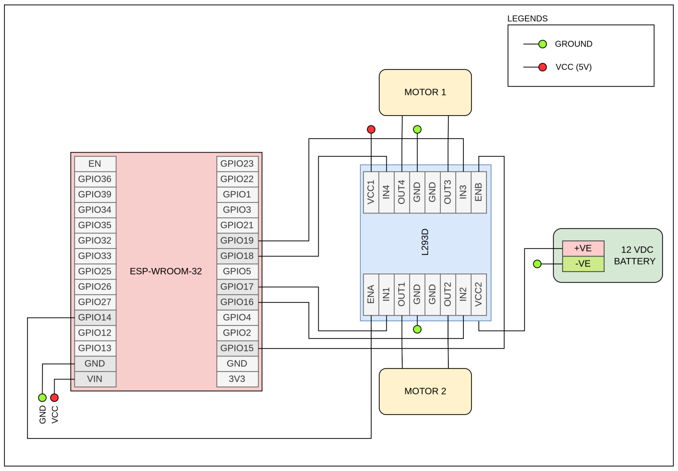

Fig. 1: Wiring Diagram for Beacon Tracking.

5 Results

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.