**RTES Project Report**

*Professor Matthew Campisi*

*Thursday 6:00-8:30 pm*

**Group 6 team 3**

**December 20, 2019**

The project is divided into two parts: avoidance of the obstacle and the determination of the position of beacon.

**Basic modules**

Our project used a four-motor vehicle, with a PWM expansion board, a motor driver,  two servos, two ultrasonic sensors, three infrared sensors and an ESP8266 board.

Our project contains 5 libraries. The main control codes is Src.ino

**Motion library**

The control of motors by using PCA9685 PWM expansion board and motor driver. Code are in move.h/move.cpp.

**WiFi library**

The connection and detection of RSSI part is in wifi.h/wifi.cpp. It also contains a UDP broadcast function for remote debugging.

**Ultrasonic library**

The codes of ultrasonic sensor used to detect the obstacle are in ultra.h/ultra.cpp.

**Servo control library**

The servos’ turning module is in servo.h/servo.cpp. Ultrasonic sensors are attached to servos, because we want to detect the reflection of ultrasonic in different direction,

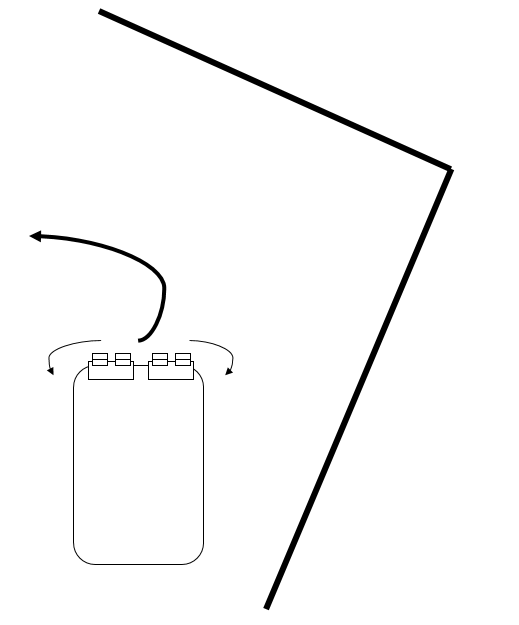
**Algorithm library**

The algorithm we used to determine beacon and related algorithm are in algo.h/algo.cpp.

**Avoidance of the obstacle**

We use the polling method to get the signal from the ultrasonic sensor and infrared sensor.

**Situation 1:**

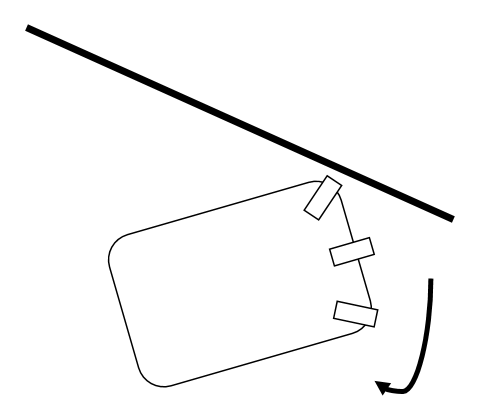


**Fig1.1 Using Ultrasonic Sensor**

When the ultrasonic sensors detect there is an obstacle in a threshold distance. Servos are controlled to rotate the sensors, and the distance of each detection are separately recorded. Then a way with a smaller turning angle are selected.

Using this method, the vehicle can avoid the majority of the obstacles.

**Situation 2:**

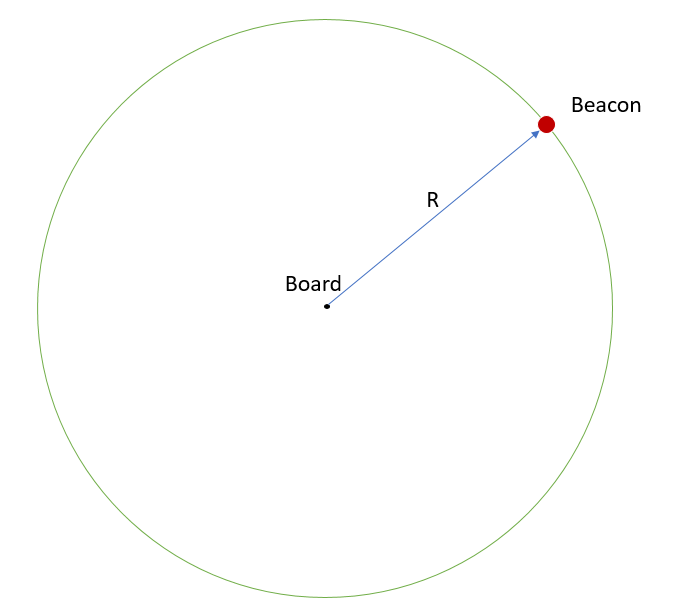


**Fig 1.2 Using infrared sensors**

In this situation, the infrared sensor can detect a broader angle and then can be used to detect in a shorter distance. when a sensor on one side detected obstacles and while the other side not, the board continued to turn to the non-obstacle side.

**Determination of the beacon’s position:**

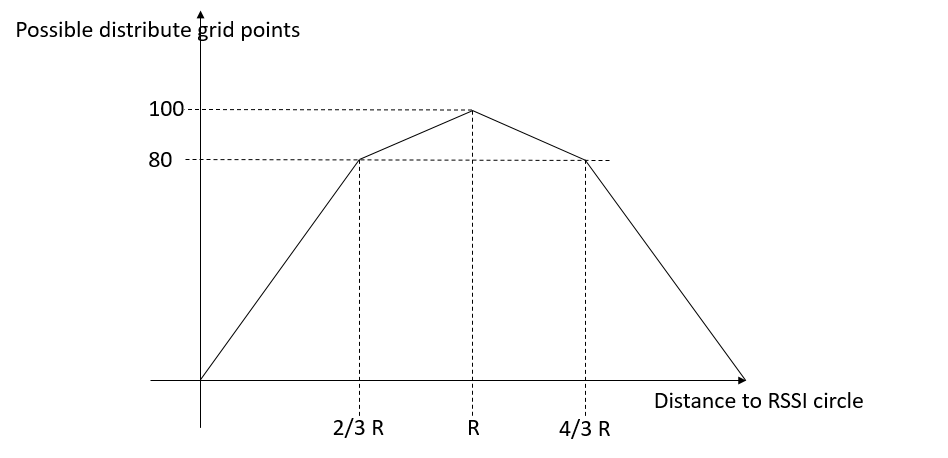
As we can see below, each time we can detect a RSSI from the board and we can using the function to calculate the distance from our board to the beacon:



**Fig 2.1: the possible circle of beacon to our board**

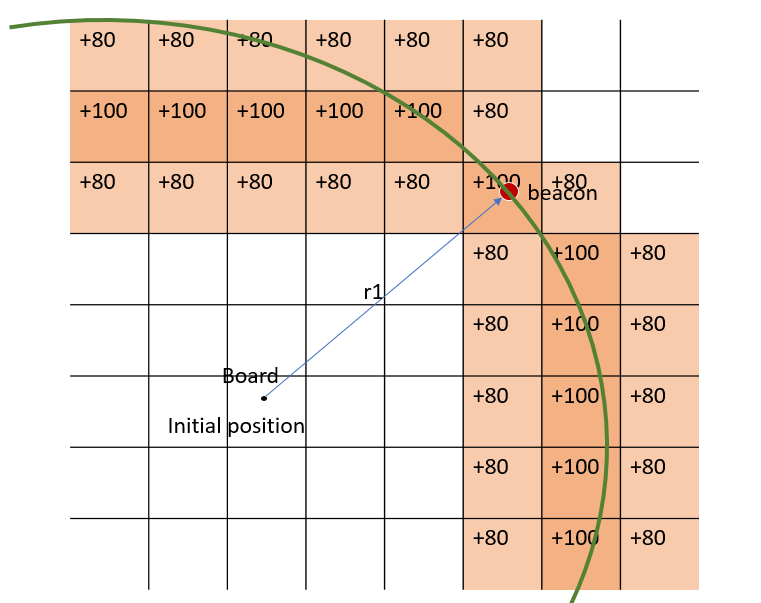
Now we set a grid to simulate the onsite.

Because of the uncertainty of the RSSI and the approximately of using grid, we implemented a function to record and update the position:



**Fig 2.2 Possible distribute point function**

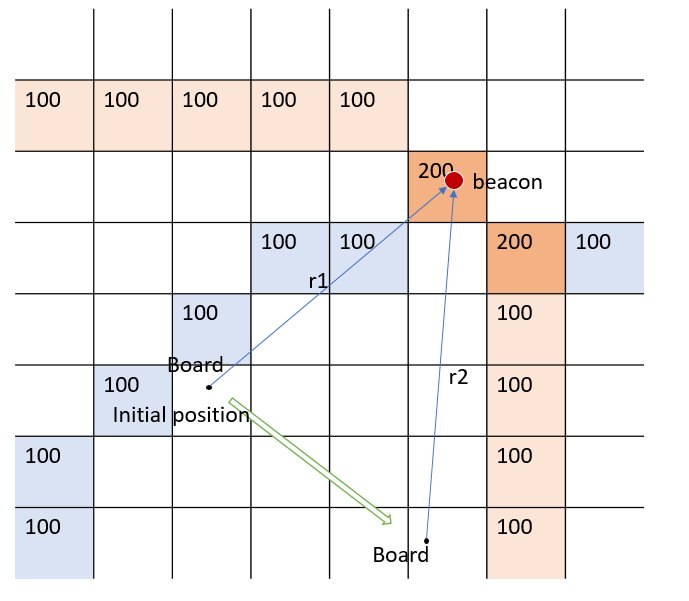
Using the formula (1) we can calculate the approximate radius of possible circle. Then we update the grid points.



**Fig 2.3 Using grid to store the possible circle**

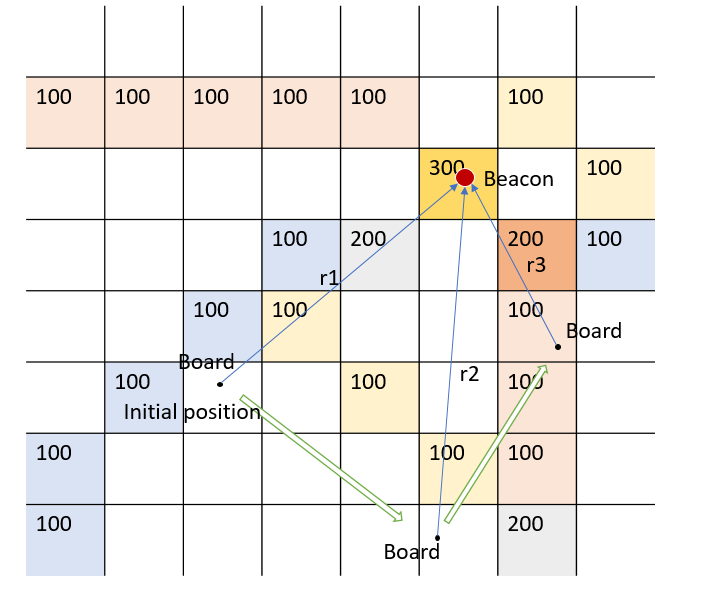
**Basic Algorithm:**

1. We start our vehicle randomly, record and update the gird.
2. Vehicle choose a direction to the highest points and move on a given distance. Calculate and update the gird.



**Fig 2.4 Update and find the highest position in grid**

1. Check the new RSSI, if it stronger than a threshold, stop, otherwise repeat the step 2.



**Fig 2.5 Approach to the beacon**