1. From the scan data how to determine where the obstacle is ?

Scan data gives detail of angle and distance at which the obstacle is located. According to SDK, the scanned data is being stored in a structure variable, so the data can be directly read from it.

2. We will use pwm  to drive the motors that move the vehicle, check how to use pwm to move / turn the vehicle.

The DC motor will rotate based on the potential difference between its two terminals. So one terminal connected to a pwm output and another to 0V is enough to make it rotate. Since we need both forward and backward and we would need 2 motors ( left/right ), there are two methods to do this:

(a) Give connection to each terminal from one PWM connection. This means we need 4 pwm connections.
(b) Use another device : L293D Motor Driver.

3. what should be pulse width to move the vehicle at its slowest speed and at its max speed.

Changing of speed is by changing the duty cycle of the pwm. We needed the max speed to be around 1m/s and min as 1cm/s. For max speed we can have duty cycle=100% and for min speed we can have duty cycle=1%.

4. algorithm for triangulation using wifi to determine best path to destination.

We use triangulation for IPS(indoor positioning system). For this we need distance from 3 sources. Wifi strength varies with distance as

$$RSSI = -10n \log\left(\frac{d}{d_0}\right) + A + X_\sigma$$

- *RSSI* = measured strength
- *n* = transmission loss factor
- *d* = distance from source
- $A, d_0$ = *A* is the *RSSI* value observed at distance $d_0$
- $X_\sigma$= It is gaussian constant

So if we have the initial positions of the wifi spots then we can easily find our position.

http://tdmts.net/2017/02/04/using-wifi-rssi-to-estimate-distance-to-an-access-point/

If there are 3 points and we know the distances of a target point from each of the sources, then we can find it's coordinates by finding the intersection of these three circles whose centres are the source coordinates and radius is the corresponding distance, so by that way we will be able 2 find position of vehicle at 2 different time Instances and basically calculate it's direction. (that $\Delta t$ is small).

5. write pseudo code for each driver, this will help to identify parts of code that can be put in a function, code that can be made generic, for example if a motor is replaced with another with higher or lower capacity, code should not be rewritten.

```
init() {
    Set destination and other route specific variables;
    Set initialization values for IPS ( Indoor positioning system);
    Check LiDAR health;
}

scandata() {
    Give scan command to LiDAR
    Process the scan data and obtain array of 360 values where
each has the distance of obstacle at the angle range of 1 degrees.
    Return Array
}

main() {
    init();
    Curr_destination = start;
    While ( curr_desination != Goal) {
        Data D = scandata();
```

```
            Direction G = check_direction_to_move(D);
            curr_destination = move_in_direction(G);
      }
      stop_scanning();
}


check_direction_to_move(Data D)
{
      Direction init=get_direction_towards_goal();// Uses IPS
      Direction curr = get_curr_direction(); // uses IPS
      Init = Find_best_direction( init,curr ,D);
      Return Init;
}

Find_best_direction(Direction init,Direction curr,Data D)
{
    {horizontal,vertical}  = optimal(init,curr); // returns optimal
direction
    if(check_obstacle(horizontal,D))
        Return horizontal;
    if(check_obstacle(vertical,D))
        Return vertical;
    if(check_obstacle(back(horizontal),D))
        Return back(horizontal);  // back(direction) gives opposite of
direction
```

```
     Return back(vertical);
}





optimal(Direction init,Direction curr)
{
   if(goal is in upright position wrt vehicle initial position)
       Return {Right,forward};
   if(goal is in up left position wrt vehicle initial position)
       Return {left,forward};
   if(goal is in downright position wrt vehicle initial position)
       Return {Right,backward};
   if(goal is in down left position wrt vehicle initial position)
       Return {left,backward};
}

check_obstacle(direction G, Data D)
{
     // from scandata() check whether we have obstacle in this
direction or not
   If (D.obstacle(G))
        Return false;
   Else
```

```
        Return true;
}
```