Path-finding Robotic Car Design

Mingrui Shi, *104517541,* Tianrui Zhang, *204050487,* Shengjie Bi, *704515431,* and Zimu Li, *504516709*

Abstract—**Nowadays the robot is a great approach to achieving automation. To achieve better performance under certain situation, for example in a system where goods need to be fetched from different spots whose locations are unknown to us explicitly, specialized systems need to be developed. For our project, the overall goal is to create a prototype for a system where a robotic car can be called on demand by the caller so that certain messages can be conveyed safely to the car when it arrives at the caller. We also provide a user interface to check the progress of the trip of the car.**

Index Terms—**IoT, embedded system, EE202C.**

I. Introduction

As the need for automation and IoT increase, more and more efforts are spent in the field of embedded systems. However, general models can seldom provide expected performance without adaption to certain usage. Thus, it is important that engineers make ad-hoc systems for distinct situations, which is what we did for our project.

For this paper, team responsibility will be introduced in Section II, after which are technical approach and system design in Section III and IV respectively. Section V describes the detailed implementation of every component in depth, followed by testing relevant work stated in Section VI, VII and VIII.

II. Team Responsibility

Mingrui Shi: Designed and implemented robotic car motion control and feedback control system; implemented low-level query with RSSI code; helped calibrating RSSI and integrating all codes together at robotic car side; assisted with assembling the robotic car.

Tianrui Zhang: Designed and implemented both the sending-end and receiving-end of infrared transmission protocols; implemented the localization algorithm that uses the Bluetooth RSSI; help merging all codes together at message delivering side; assisted with assembling the robotic car.

Shengjie Bi: Assembled the robotic car; validated all hardware components in the system; assisted in all testing works; helped integrating all the components; helped to calibrate RSSI and implemented the code of intersection calculation of circles using RSSI.

Zimu Li: Designed and implemented path deciding algorithm; assisted with calibrating car motion; designed and implemented clients, data server and web service; assisted with integrating codes at robotic car side; assisted with assembling the robotic car.

All authors are with School of Electrical Engineering, UCLA.

III. Technical Approach

To achieve our goal, there are several main parts to implement. First of all, we will need to know where the target destinations are. Secondly, the car will need to know the path it will follow to trace all the destinations. Thirdly, on reaching each target, the car should be able to receive information from certain device at the destination. Moreover, it is better for us to design an application to display everything in the system.

With such considerations, we decided to adopt Bluetooth technology and use RSSI to determine the locations of destinations, and set up a server so that devices at destinations can send their coordinate information to it. The server, on demand of the car, can send the destination positions to the car so that the car can apply some algorithm to calculate an appropriate path. The car will also need some mechanism so as to move in the exact path it has decided. For information retrieval system on reaching destinations, we decided to use IR receivers to capture signals from the IR transmitters installed at each target position. In addition, we will design a web application for showing position information and information received by the car on a website.

During the development of this technology, we indeed encountered many challenges, some solved and others bypassed. One of the major problems was that bluetooth RSSI is unstable, resulting in an inaccurate calculation result. On the IR LED side, we adopted median filter to filter out the noise components. In this way, with some calibration, we can achieve an acceptable accuracy in the position information. It is worth mentioning that this is achieved at the cost of longer time period, making it impractical to apply on the feedback system for driving the car accurately. Another major problem is the accurate control of the car. In our development process, we find that even applied with the same voltage level, the motor cannot provide stable speed due to the change in battery voltage. As stated above, RSSI cannot be used to implement this feedback mechanism, we adopted rotary sensor for PID mechanism. Also, in the IR information exchange part, we needed a longer working range, for which we used MOSFET for amplifying the signal, and a protocol for transmission, in which we applied CRC to ensure information integrity.

IV. System Design

As mentioned before, the overall goal of this project is to create a prototype for such a system that a robotic car can be called on demand by the calling side and a short message can be securely delivered to the coming robotic car upon its arrival. Therefore, as shown in figure 1, the whole system can be divided into three parts in terms of functionalities: the message delivering client(infrared beacon), the robotic car, and the backend servers.

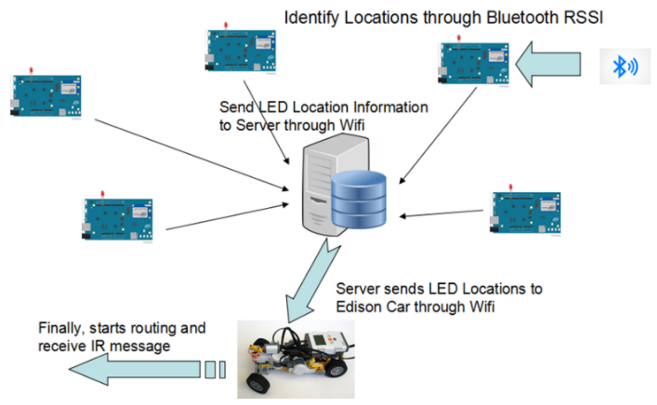
Identify Locations through Bluetooth RSSI
Send LED Location Information to Server through Wifi
Server sends LED Locations to Edison Car through Wifi
Finally, starts routing and receive IR message

Fig. 1: General Design Diagram

At the message delivering client side, first of all, it will determine its location through a series queries of RSSI values to all nearest Bluetooth beacons. This can be done because there exists a direct relationship between the distance to a Bluetooth beacon and its RSSI readings, and, then, we know the client sits on the circle drawn with the beacon as the center and the distance as the radius. The exact locations can be found through the intersection point of three such circles. The detailed algorithm can be found in section 6. Next, once the client side has its location, it then starts to call the robotic car. It will do so by acting as a TCP client and sending its own name tag and location information to the backend server through Wifi. The server will schedule a run of the car upon receiving enough requests. The last step for the client is then repeatedly broadcasting its message through infrared LED at its location and waiting for the car to come and receive it. We select infrared for transmission because it is invisible for human eyes and only transmit within a certain physical range. Please refer to section 6 for detailed IR protocols.

Next, at the server side, as mentioned, it will repeatedly listening for client requests at a certain TCP port and schedule a run of a robotic car as soon as receiving enough requests. In order to achieve this, there is also a TCP client running in the car. When the car is idle, the TCP client will repeatedly querying the server for information to start a run. The server will then redirect all necessary informations to the car. For the purpose of user-friendly design, the server also has a front-end web display running on Ruby on rails that shows all locations of IR beacons and the motion of the car in a map in real time. So, besides querying for start-up information, the TCP client in the car also constantly report the location of the car back to the server. In this project, for purpose of demonstration, the actual IR message received by the car will also be display on the front-end webpage.

Lastly, at the robotic car side, it also consists of several parts. The most important part is the path-finding algorithm. Due to the fact that there may be a large number of infrared beacons it needs to traverse in a single run, an optimized routing algorithm that gives the shortest traveling distance is then a must. The algorithm we use is called 2-opt, which has the main idea that it is to take a route that crosses over itself and reorder it so that it does not. It repeat this process to find the final route. Please refer to section 6 for details. In order to accurately control the car, we also has a feedback control system that relies on a rotary sensor on the back wheel to record the exact number of turns. The embedded Intel Edison will then run a PID(ProportionalIntegralDerivative) control logic to minimize the error so that the car can move to the exact location we want. The car also has an IR receiving client that works under the same IR protocols mentioned before.

V. Implementation Description

A. Bluetooth Positioning System

The reason why we include Bluetooth positioning system in our project is that, otherwise, user has to input the exact location information by itselfs. This requires measuring the exact locations for every infrared beacon by hands. Therefore, creating a system that can allow infrared beacon to automatically determine its location information is desired. We pick Bluetooth in our project because our testbed is relatively small and Bluetooth is well supported at software level.

We have mentioned briefly how to do localization using Bluetooth in section 4, which is through series of RSSI queries to all nearest Bluetooth beacons and find the intersection point of three circles. However, in real life, the RSSI values are not very stable and reliable. Hence, we introduce some variants to the system, and the detailed procedures are therefore as follows:

a. Measuring and processing the raw RSSI values
Because the RSSI varies all the time, we decide to query ten times and perform a "mid-band" filtering to all data such that only values within a certain range near the median will remain. We then average out the remaining values and use it in later calculations.

b. Calculating the distance to each Bluetooth beacon
Based on the following formula, we can easily calculate the distance given measured RSSI value, once we know $RSSI(d_0)$ and n, where $RSSI(d_0)$ is the RSSI value at reference distance d0 and n is the transmission loss index and marks the degree of decay of Bluetooth signal in this environment.
Therefore, in order to obtained the two values, we need to measure a series of pair values of distance and RSSI by hands beforehand. Also, it is worth mentioning that, the Bluetooth chip for each Bluetooth beacon may be different, so we need to calibrate the two values for each beacon.

c. Find the intersection areas of three calculated rings
As mentioned,given the distance to the beacon, we then know that the client sits on the circle drawn with the beacon as the center and the distance as the radius, and the exact locations can be determined at the intersection point of three such circles(shown in Figure 2).
However, in real life, these three circles may not even intersect at all due to many random errors from the environments. Therefore, we decide to introduce an error region to the distance, and the result will be a ring instead of a circle. The result is shown in Figure 3:

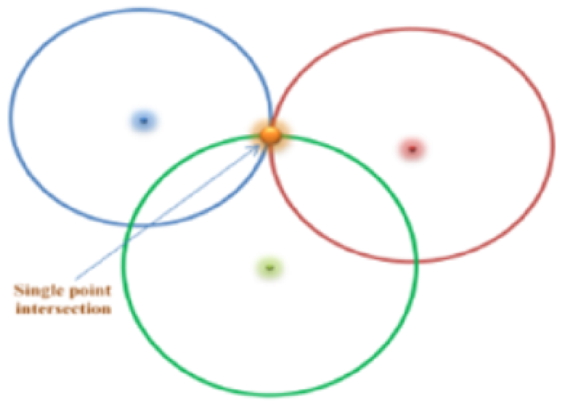

Fig. 2: Intersection of Three Circles

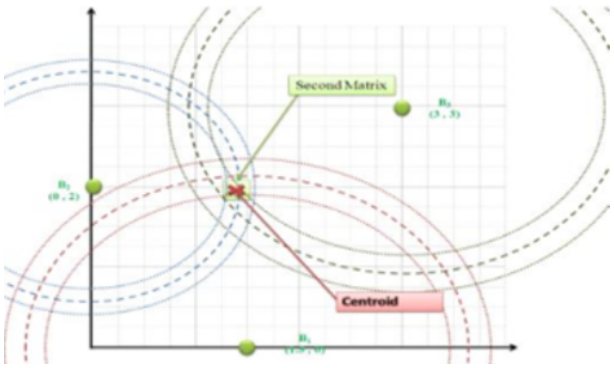

Fig. 3: Intersection of Three Rings

d. Finding the center point of the overlapping area

Intuitively speaking, the actual position should be close to the center point of overlapping area. Therefore, finding the center point of the closest-fitting quadrilateral should give us fairly accurate results. It also helps to reduce the workload of calculations because finding the center point of a general shape is relatively high.

$$RSSI(d) = RSSI(d_0) - 10 \times n \times log(\frac{d}{d_0}) \qquad (1)$$

B. Infrared Transmission System

From the design goal of the project, we notice that the message transmission model in our project is just a simple "Sender->Receiver" model. That is, there is no actual communications happening between the infrared sender and receiver. Also, from professor's descriptions, we know that the transmitted message need not to be long at the current stage. Given these requirements, both the hardware and software of the system should then be designed simple as well. Another benefit for keeping it simple is to easily assemble the whole system onto the robotic car. As a side note, although our design does not contain interactions between sender and receiver, it can be easily extend to support such features if needed in the future.

1) The hardware circuit: At receiving end, since infrared signals are quite common in real life, we decide to follow professor's suggestion and use an special infrared receiver that is purposely designed to only receive 38 kHz signals so that all natural infrared signals will be filter out. In short words, it will smooth out the signal in Figure 4 and turn it to signals in Figure 5.

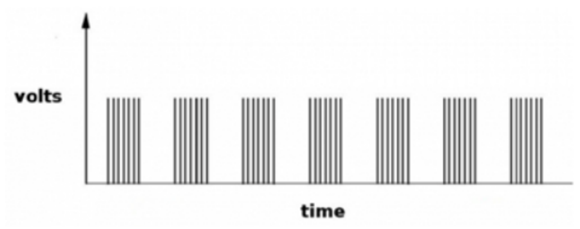

Fig. 4: Total IR Signal

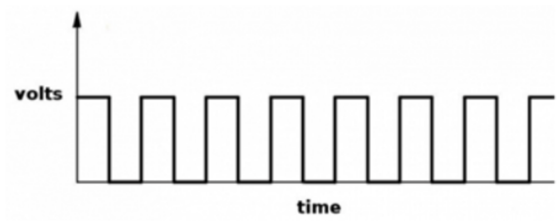

Fig. 5: Received IR Signal

At sending end, we do not have any special requirements to send the signal. Therefore, a simple infrared blinking LED will fit our needs. However, due to the fact that Intel Edison only has a maximum output of 5V, we will need extra high power circuit for IR LED to transmit enough range. The circuit is shown in Figure 6. The LED is connected to a 9V battery, a 300 resistor, and the source of a N-channel MOSFET. The gate is connected to a PWM pin of Edison board so that the whole high power circuit can be controlled by Edison to emit 38 kHz modulated signals.

2) The transmission protocols: Similar to computer networking protocols, our protocols are designed to transmit messages through packets. The major benefit followed is then that the whole message can be splitted into small chunks. While some chunks may be corrupted during transmission, other chunks will be in good conditions and saved at the receiver end. Therefore, the receiver need not to record the whole message from the start again and can target only the

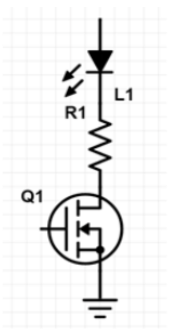

Fig. 6: High Power Circuit for IR LED

1111111111	00010101	10110111	10110100	11011	11	0000000000
Header	Addr	Data1	Data2	CRC	End	Interval

Fig. 7: Transmission Package Format

corrupted packets. It is worth reminding that the design goal specifies that the message will not be long, so we designed the packet in the format shown in Figure 7.

The finest granularity of the packet is a value of either one/high(LED on) or zero/low(LED off) in a fixed time interval of 8ms.

Header	header marks the beginning of a packet. It consists of ten consecutive ones.
Addr	address marks the index of current packet in all packets so that the receiving end can reconstruct whole message. With current setup, address field only has 8 bits. So, the maximum length of message transmitted is $2^8 \times 2 = 512$ bytes.
Data1	data1 is the first byte transmitted in current packet
Data2	data2 is the second byte transmitted in current packet
CRC	Cyclic redundancy check provides a mean to check the validity current packet. It treats the whole message you want to transmit(except for the crc field) as a high order polynomial, and do a polynomial long division using an agreeable short polynomials, namely, the "generation key", to get a polynomial reminder. An example is shown in Figure 8. In our project, a 5-bit crc field will be enough to check the validity of our packet of 34 bits.
End	end marks the end of packet
Interval	interval marks the interval between packets. It consists of ten consecutive zero's.

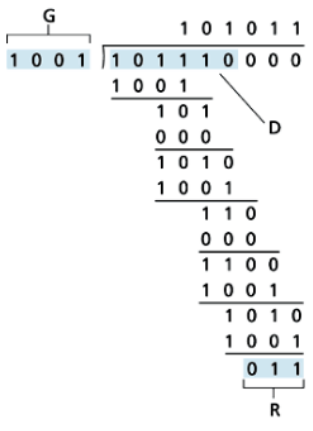

Fig. 8: CRC Calculation Example

The actual implementation is based on linux ISR(interrupt service routine). A timer is set at the beginning of program and will be triggered every 8ms. At each timing interval, the sending side will choose a value to transmit based on the packet constructed, and the receiving side will record the value received and try to reconstruct the packet.

C. Robotic Car

In our system, we use a lego car as our robotic car. Lego cars have scalability and are very easy to add on more components, but our system is not limited to lego cars and can be implemented on different kinds of robotic cars. From the hardware perspective, in addition to the basic robotic car, we add a rotary encoder on the rear wheel to record the distance that the robotic car has traveled. In our robotic car software system, we use multithreading to guarantee that the car can communicate with other components while it is moving. More specific, we have three threads for our robotic car; the main thread is responsible for path-finding algorithm and calculations of car motion and car position; the sensor thread is responsible for reading data from the rotary encoder; transmitting data between robotic car and cloud server uses another thread.

1) Path-Finding Algorithm: Before the car start moving, it will receive all the available LED locations in coordinate from the server and use the locations to calculate a relatively optimal path through path-finding algorithm.

The path-finding algorithm we used here is called two-opt, which is a simple local search algorithm first proposed by Croes in 1958 for solving the traveling salesman problem (TSP). The main idea behind it is to take a route that crosses over itself and reorder it. More specific, the car will first run greedy algorithm to get a path based on LED location coordinates. Then two-opt algorithm will be used to optimize the result of greedy algorithm to get the final path for the robotic car.

Figure 9 shows the result before and after using 2-opt algorithm.

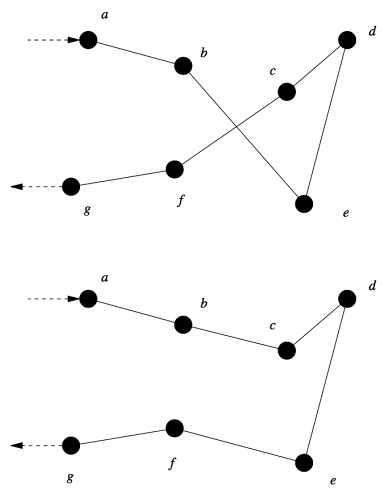

Fig. 9: 2-opt Optimization

2) Car Motion Control: Car motion control is mainly implemented in the main thread, and the control system is basically a feedback PID control system assisted with some mathematical calculations.

First of all, the car have a state value to save its offset between the positive x-axis, which is the direction of the robotic car. Then the car can decide to turn left or right based on its start and destination coordinate and also the offset. To get the car on the desired route, we need to calibrate the angle of the servo motor. Moreover, the rotary encoder keeps to track the distance of the car traveled in the sensor thread, and feedback control system will change the speed based on the distance left every 0.5 second. When the car reaches a destination, it will wait for IR emitter to transmit information to it. After receive the message, it will send the message to the server and then repeat the process to move to next destination. During the car is moving, it will also send its real-time location to the server through transmission thread. At last, the robotic car will go back to origin.

After implementing the feedback control system, we don't need to calibrate speed of the car to get the relationship between the speed in the software and the real distance, because control system can control the car in an accurate way, resulting in less error. For our prototype system, we just use a simple feedback control system, a better feedback control system with accelerometer and gyroscopes can perform more accurately.

3) Communication with LED and Cloud Server: The robotic car integrate LED client and web client to communicate with IR emitter and web server. By using different threads and interrupt routine, the car could receive data from IR and web server and send message to the web server. Details are included Cloud Server and Infrared Transmission System.

D. Client Design and Cloud Server

For completing transactions over the internet, we need to have clients installed on every LED and car so they can communicate with the server, which is, needless to say, necessary in this situation.

1) Client: In the design of client, we need to determine where the clients should be deployed. Definitely there need to be a client laid on every LED since it has to inform the server of its position. Meanwhile, the car also need to inform the server of its current position and the message it has received apart from its retrieving LED location information.

Every single transaction between a client and the server is conducted via TCP socket. The client sends a TCP connection request to the server and waits for its approval before they can communicate over the internet.

LED Client The client on LED is simple. What it is supposed to do is only reporting to the server on acquiring the location data it has calculated using RSSI information. Since information is traveling over the internet, it must be encoded before sent out to the destination. The encoding scheme we are adopting here is as follows:

$$\textit{LED index}\#x, y \tag{2}$$

Car Client Compared with the client on LED, there is much more to be done on the car side. First of all, it has to acquire destination information from the server. Secondly, it should report position of the car to the server periodically since we will display such information on the web application side. Also, when it arrives at a destination, it should send what it has received there to the server. Real-time location is calculated in the main thread of the car, which has already been discussed in the previous sections. This position information will be encoded, together with the message received by IR receiver if it has just arrived at a target, for transmitting over the socket. The encoding scheme will look like this:

$$c\#x, y\#message \tag{3}$$

Using the same format as that used by the LED client will save us effort on the decoder side while not sacrificing performance.

2) Data Collecting Server: We implemented actually two servers for distinct usage respectively. For avoiding confusion, this server introduced now is called a data collecting server to be separated from the web application server.

The server introduced here is for collecting location information from the LED clients and save them. When the car client connects to this server, it will send LED location information back to the car after receiving the car location information and, if any, message it just received from a target. Also, the other server, which is the web application server, will connect to this data collection server periodically to query the location information of all objects in this system as well as the received message.

When responding to the car client, data collection server will simply concatenate all the LED position information string together. To make it clear, the format is as below.

$$\{\textit{LED index } i\#x_i, y_i\backslash n\} \tag{4}$$

For the response to web application server, car position information and message are also needed. What we did is to add the car position string at the beginning, and the encoded message at the end, which is as follows.

$$c\#x_c, y_c\backslash n + \{\textit{LED index } i\#x_i, y_i\backslash n\} + m\#\textit{message} \tag{5}$$

3) Web Application: The web application part of this project is mainly displaying the real-time location information of LEDs and the car in the system. To achieve this aim, what we used is mainly Ruby on Rails, combined with HTML5 and jQuery.

We adopted Ruby on Rails for our project mainly because it provides us a convenient MVC framework for web development. However, what makes our application slightly different from the usual case is that we did not utilize the model provided by Rails, but rather we acquire data from the data collection server. From the viewpoint of clients, data collection server works as a server, while for our web application it works similar to a database.

For displaying location of every object timely, we decided to implement Ajax with jQuery, so that users will not need to

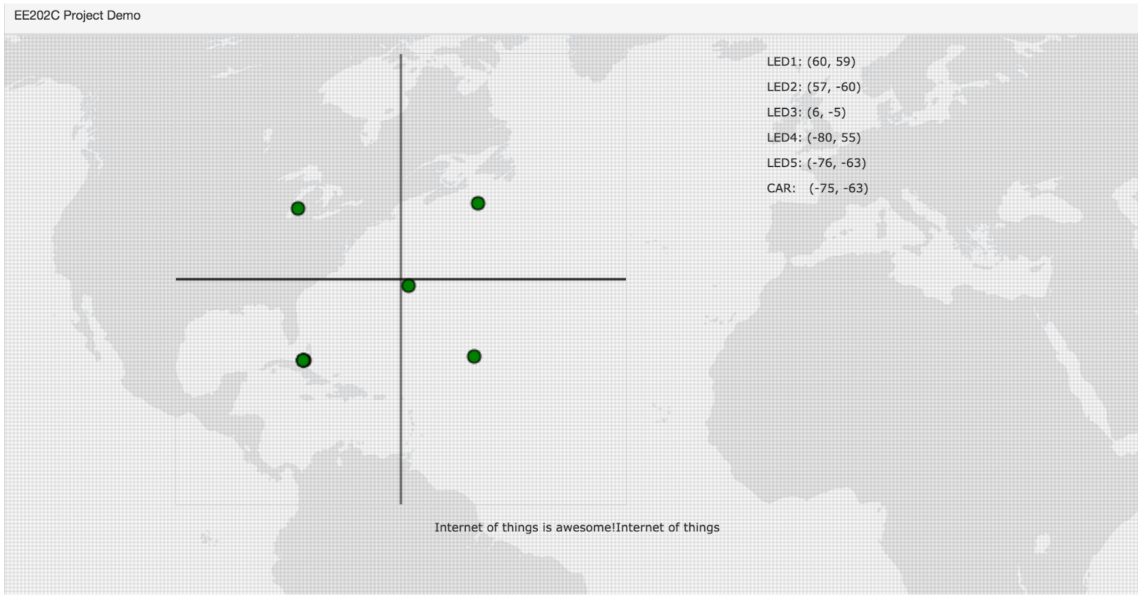

Fig. 10: Web Application Interface

refresh the web page to see the latest result. Also, to provide a graphical display, we adopted HTML5 canvas element to draw a coordinate system on the web page.

When updating location information, the application controller is sending out queries to data collection server every 0.5 second, which we believe is frequent enough for use. The controller also establishes connections via TCP socket, which is similar to that of client. The identifying message it sends to request information from the data collection server is formatted as below:

$$w\# \tag{6}$$

When the server receives this message, it will respond by sending the string introduced in the previous statement, so the application controller can parse the string for useful data.

VI. Test and Evaluation System

Since our project consists of several parts, it will be more appropriate to test them separately before testing the integration result. Thus our testing process includes the following parts:

1. Testing IR transmitter and receiver;
2. Testing whether the car can reach the targets following the scheduled order;
3. Testing whether the path-deciding code can generate an optimal path;
4. Testing whether the LEDs can decide their locations correctly via RSSI;
5. Testing whether standalone clients and servers can behave as expected;
6. Testing whether the web application can update location information timely;
7. Testing whether the integrated system can work correctly.

VII. Test Results

Our system has demonstrated that the overall system is successful in the demo. The robotic car can travel all the destinations in the desired order with relatively small error and frequently exchange information with IR and Web server. However, our system still have some limitations, such as the feedback control system is not a completely feedback system, which needs no calibration, RSSI localization is not very accurate. Each of our system components have a relatively small error range; RSSI error range is about 10cm, the car movement error range for each destination is about 5cm, and the IR transmission range is about 15-20cm. However, when all the system components integrate together and the car travels a numbers of destinations, the robotic car may not able to receive the message from IR because of out of range issue.

The following sequence of operations represent our system operations of path-finding robotic car:

1. Infrared emitters receive RSSI signal from 4 corner bluetooth sources and calculate their positions;
2. Infrared emitters propagate their locations to the server and the server gives this information to the robotic car;
3. The robotic car run the path-finding algorithm based on locations received and get the path;
4. The robotic car calculate its route to next destination and start movement controlled by feedback control system. Meanwhile, the car report its real time location to the server;
5. The robotic car reaches the destination and waits for information from IR;
6. Information from IR received, the car start moving again and send the information to the server at the same time. Repeat step 4-6 until reach the last destination;
7. The car finishes all the destinations and goes back to the origin starting point.

VIII. Analysis

Looking back at the whole project at this time, our team agree that the major challenge we faced in the project is to integrate all parts of the designed system together. On one hand, specifically, the whole system requires the knowledge of robotics, the usage of various embedded sensors, and a combination of three distinct wireless technologies: the WIFI, the Bluetooth, and Infrared Transmissions. On the other hand, these requirements also drove us to learn something we do not easily learn from other lectures. In order to smoothly integrate the system together, we studied simple mechanics, debugged the hardware circuit plenty of times, learned and implemented several feedback control algorithms and programming skills for the first time. The result is satisfying. In our testbed with five message-sending beacons, the system can finish one single run in less than 3 minutes.

Another success in our design is that we implemented the entire infrared transmission scheme from scratch. This enables the whole system to work with very simple and low costs infrared LEDs and receivers.

Our front-end web display is another thing that is worth mentioning in the project. Not only it provides means for a user-friendly way to monitor the running of entire system, but it allows the user to remotely control the whole system as well.

Certainly, there are also rooms for our project to be further improved. The biggest bottleneck right now is the throughput of infrared transmission when it comes to long messages. The reason for this is that we did not foresee that there is a limitation on the speed of PWM(maximum speed is 5ms per change) on Edison board with normal MRAA libraries in Linux environment. Our mentor Chris told us there is a quicker way using on-board MCU. However, due to the limitation of time, we haven't tried it yet.

Another thing that we can improve is the feedback control system on the front wheel of the robotic car. Currently, the car is running only with the back wheel feedback control, and the front wheel will always turn in a fixed degree. We tried to implemented the front wheel feedback control in such a way that we link the front wheel servo and the rotary sensor in a conveyor belt way, but the mechanics part did not go well. We are sure we can do a better job if we can find fitting mechanics parts. Then, the robotic car does not have to turn in circles anymore and can directly go anywhere it wants.

Last but not least, our prototype system is relatively small in size. It cannot be directly used in daily life environment yet. However, this can be changed if we switch to other robotics cars in bigger sizes with better mechanical components.

IX. Shared Systems

We can share the entire system design and code of our project.

The links below are the GitHub repositories of our project:

https://github.com/darkherald/intel_edison_lego_car.git

https://github.com/RayZhangUCLA/Intel_robotic_car_IR_communication_client

X. Conclusion

We designed and implemented an IoT system where the car can retrieve target locations from the server, visiting them consequently and receiving information from the target devices and report to the server where application for monitoring the whole procedure is built. Techniques including RSSI, PID, IR, sensor network, socket programming and MVC are adopted in our project. The outcome of our project is a working system with expected behaviors. Future expansion of our project can be eliminating unavailable targets from the path, moving targets or more efficient car motions. We have made our implementation easy for expansion and modification.

Acknowledgment

The authors would like to thank Professor William Kaiser for the great lectures throughout the whole quarter. Also, the authors hope to thank Chris Baek for his kind help in many aspects.

References

[1] A. Thaljaoui, T. Val, N. Nasri and D. Brulin, *BLE Localization using RSSI Measurements and iRingLA*, ICIT, 2015.

[2] *Cyclic Redundancy Check*, https://en.wikipedia.org/wiki/Cyclic_redundancy_check, 2015.

[3] *2-opt algorithm*, https://en.wikipedia.org/wiki/2-opt, 2015.

[4] Jimbo, *IR Control Kit Hookup Guide*, https://learn.sparkfun.com/tutorials/ir-control-kit-hookup-guide?_ga=1.181684519.1084991569.1442520821, 2013.