# Embedded-system Lab

## Final Project Report

## Topic: Mimicking Self-driving Cars

B07502027 聶家任
B07502028 吳宗翰
B08901162 林郁敏

### 1. Introduction

AGVs (automated guided vehicles), intended for transporting heavy materials or goods, has become a trend in factories or warehouses. Interested in cars, our team decided to make a self-driving car mimicking the behavior of an AGV.

### 2. Map

We used old calendar paper, aluminum sticks, and cardboard boxes to construct a map where the car drove. Our map can be represented by an un-directed graph stored in the server code. See Figure 1 for the whole view.

Figure 1. Our map

2.1. **Nodes.** There are 6 nodes in the map, each covered with tents made of paper and supported by aluminum sticks. The purpose of the tents is to produce a height difference, which can be detected by the two HCSR-04 sensors facing upward.

2.2. **Roads.** There are 6 roads in the map, each connecting two nodes and having walls on either side. The walls are made of calendar papers. Cardboard boxes are necessary to support the walls from behind to prevent walls from collapsing and bending.

2.3. **Destination.** At the destination node (farthest node in Figure 1), there is a smartphone waiting for the car to arrive. The cellphone writes a specific message into NFCEEPROM, and BL475E reads it out and sends it to the server. After the server receives this specific message, it confirms that the car has arrived at the destination node.

2.4. **Route.** The route, embedded in the server code, is specified by a sequence of nodes. When the car arrives at a node, the server receives a notification from the car and orders the car how to respond (turn left, right or go straight) according to the arranged route. More specifically, the server decides the throttle and angle values and sends them to Raspberry Pi, which is responsible for controlling the servo and motor of the car.

## 3. Car

There are various components on the car, shown in Figure 2. Their functionalities are briefly mentioned in Figure 2. Their interaction between one another are described in detail in the next section.

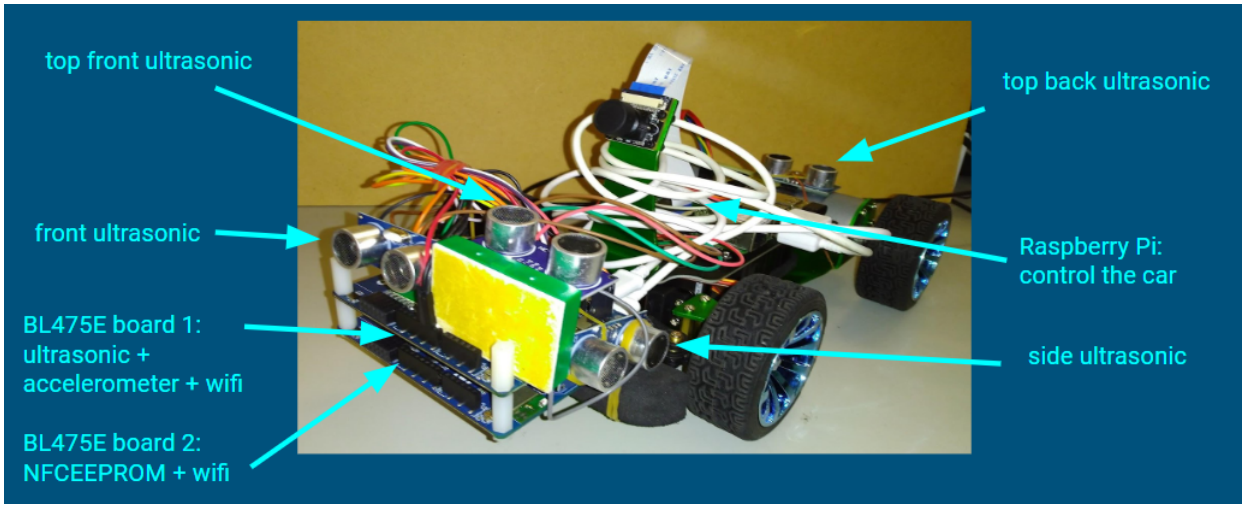


Figure 2. Our car

## 4. Approaches

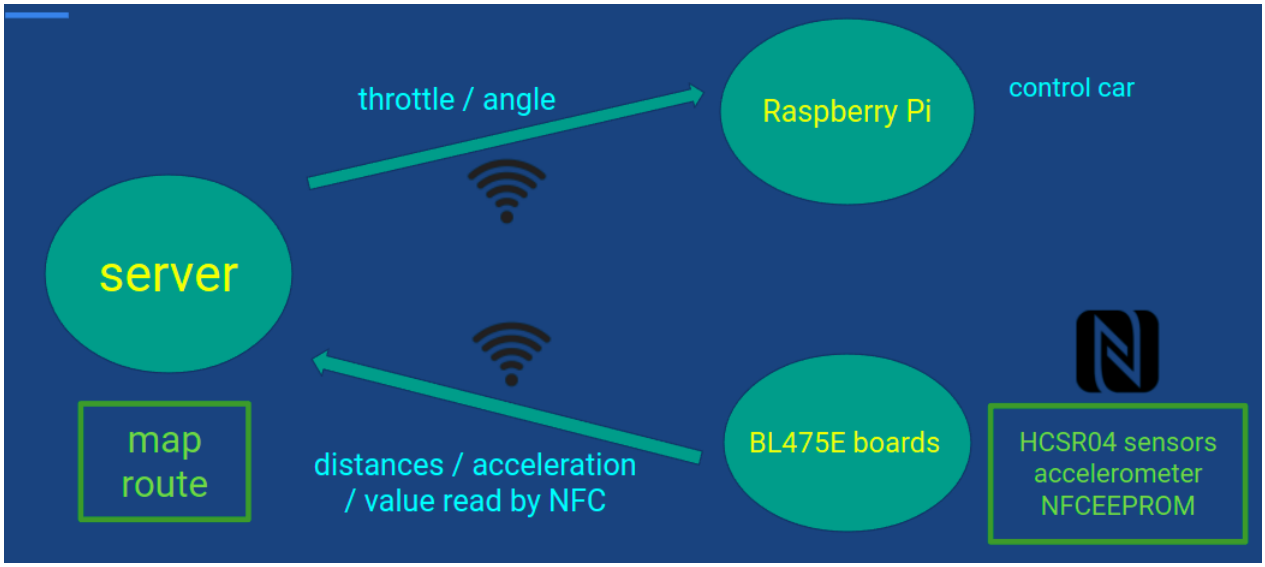


Figure 3. Data flow

4.1. **HCSR-04 ultrasonic modules.** There are 5 sensors installed to the first board. 2 top sensors measure the distance to the ceiling, if the distance is short, the car is located at one of the nodes. 2 side sensors measure the distances to the walls to avoid collision. 1 front sensor measures the distance to front obstacles to adjust the throttle value.

4.2. **Accelerometer.** The on-board accelerometer measures the acceleration value of the car. If the value is too high/low, the server will tune down/up the throttle value.

4.3. **NFC.** The second board periodically reads the value stored in on-board NFCEEPROM, if the specified value is written, the board will notify the server to indicate that the car has arrived at the destination.

4.4. **Wi-Fi.** The boards use Wi-Fi to send all sensor values to the server.

4.5. **Server.** Based on distances and acceleration values measured by ultrasonic sensors, we define 5 distance variables and 1 acceleration variable: the distance in front is defined as $dis$, the distance in left/right is defined as $dis\_left/dis\_right$, the distance in top front is defined as $dis\_up$, the distance in top back is defined as $dis\_back$, and the change of acceleration is calculated by the root of sum of squares of the changes of acceleration values on three axes and defined as $\Delta acc$. Then we propose a *steering* algorithm to compute the throttle and angle values. Note that angle ranges from -1 (leftmost) to 1 (rightmost) and throttle range from 0 (zero speed) to 1 (max speed).

**Algorithm 1:** *computeThrottleAndAngle*

**Input:** $dis, dis\_left, dis\_right, dis\_up, dis\_back, \Delta acc, previous\_angle, previous\_dis\_up, previous\_dis\_back$

**Output:** $throttle, angle$

**Let** $max\_throttle = 0.75$, $roof\_height = 35$, $turning\_max\_throttle = 0.85$, $angle\_to\_radius\_ratio = 0.25$, $rotation\_radius = 7$. // parameters choosing

**if** $\Delta acc \; != 0$ **then** // if the car is not still

  $throttle = min(15/\Delta acc, max\_throttle)$ // adjust $throttle$ by $\Delta acc$

**else** // does not move, needing larger push when $previous\_angle$ is large

  $throttle = max\_throttle + 0.08 * abs(previous\_angle)$ // an extra push

**end**

// angle component caused by $dis\_right$

$right\_angle = dis\_right == 0 \; ? \; -\infty : -10/dis\_right$

// angle component caused by $dis\_left$

$left\_angle = dis\_left == 0 \; ? \; \infty : 10/dis\_left$

$angle = left\_angle + right\_angle$ // sum of angle components

$angle = min(angle, 1)$ // bound angle by 1

$angle = max(angle, -1)$ // bound angle by -1

**if** $dis\_up < roof\_height$ ***or*** $dis\_back < roof\_height$ **then** // in a node

  **if** $dis > 15$ ***and*** $0 < \Delta acc < 5$ **then** // $dis$ is large and $\Delta acc$ is small

    $throttle = min(turning\_max\_throttle, 1/\Delta\_acc + 0.5)$

  **else if** $dis > 15$ ***and*** $\Delta\_acc == 0$ **then** // avoid division by zero

    $throttle = turning\_max\_throttle$

  **end**

  // direction decision

  **if** $previous\_dis\_up > roof\_height$ ***and*** $previous\_dis\_back > roof\_height$ **then**

    // first get into a node

    Determine $action =$ "$ADVANCE$" **or** "$LEFT$" **or** "$RIGHT$".

    $throttle = 0$ // decelerate first

  **else** // already determine the direction

    **if** $action ==$ *"ADVANCE"* **then**

      **pass**

    **else if** $action ==$ *"LEFT"* ***and*** $dis\_left > rotation\_radius$ **then**

      // current rotational radius is large enough

      $angle = max(-angle\_to\_radius\_ratio * (dis\_left - rotation\_radius), -1)$

    **else if** $action ==$ *"RIGHT"* ***and*** $dis\_right > rotation\_radius$ **then**

      // current rotational radius is large enough

      $angle = min(angle\_to\_radius\_ratio * (dis\_right - rotation\_radius), 1)$

    **end**

  **end**

**end**

**Algorithm 2:** *steering*

**Input:** $dis, dis\_left, dis\_right, dis\_up, dis\_back, \Delta acc, previous\_angle, previous\_dis\_up, previous\_dis\_back, angle\_record, throttle\_record$

**Output:** $throttle, angle$

$original\_calculated\_throttle$, $original\_calculated\_angle$ = $computeThrottleAndAngle(dis, dis\_left, dis\_right, dis\_up, dis\_back, \Delta acc, previous\_angle, previous\_dis\_up, previous\_dis\_back)$

Update $throttle\_record$ and $angle\_record$ by $original\_calculated\_throttle$, $original\_calculated\_angle$.

$throttle$ = average of $throttle\_record$ `// calculate` $throttle$ `with past values`

$angle$ = average of $angle\_record$ `// calculate` $angle$ `with past values`

## 5. Knowledge Applied

5.1. **Multi-thread.** We use 2 threads in the $2^{nd}$ board: socket thread and NFC thread. NFC thread continuously reads the value stored in NFCEEPROM. When the specified value is read, the socket thread sends the value to the server.

5.2. **Condition variable.** When the specified NFC value is read, the condition variable notifies the socket thread that the value is correct and should be sent back to the server.

5.3. **Wireless communication.** We use two types of wireless communication technology: NFC and wi-fi.

5.4. **Digital I/O.** The ultrasonic modules communicate with BL475E-IOT01A board by digital I/O. The board pull up TRIG pins to emit ultrasonic waves, and the ECHO pins are pull up upon receiving the reflection wave.

## 6. Problems and Solutions

6.1. **Large NFC reader.** We planned to use BL475E-IOT01A board to read the NFC tags located at all nodes originally, but it can only achieve the goal by a large and expensive NFC reader, which is not cost-effective. Hence, we added extra two HCSR-04 sensors and build some tents at all nodes. If the distance between the sensors and the ceiling is small, the car is now located at one of the nodes.

6.2. **Change Directions.** We only used 1 top sensor installed at the front of the car at first. When the car tried to change its direction at some nodes, the car front left the node earlier than the car tail, and then it lost its tendency of turning left/right while it was still not in the right position. To address the problem, we installed an extra top sensor at the car tail, which indicates that the car leaves a node only if the whole car leaves. Hence, the car is able to adjust itself to the correct position when it turns left/right.

6.3. **Unstable Sensor Values.** The sensor tends to malfunction probabilistically, which results in some unintended behavior of the car when using the throttle and angle values returned by *computeThrottleAndAngle* algorithm. For example, the car may dash against the wall with a high speed. To reduce the effects brought by erroneous values, we introduce the concept of PID control to our *steering* algorithm. The current angle and throttle values are functions of

previous angle and throttle values, which acts as a low-pass filter. By ignoring transient errors, the car can drive stably now.

## 7. Results

The result is in mp4 format. See the video in the demo link in the next setion.

In the video, The car's initial location was at the center of the map. During the driving process, it experienced left turns, right turns under several tents. Also, it autonomously corrected its path based on the distances to walls, acceleration values, and angles. After reaching the destination, the NFC component on the bottom BL475E touched the cellphone. As soon as NFC Tool (on the smartphone) wrote values into NFCEEPROM, the server (in the laptop) received some texts indicting that the car succeeded in arriving at the destination. The NFC Tool on the smartphone displayed a 'Write Complete' message as well. The whole driving process lasted for approximately 3 minutes and 50 seconds.

## 8. Links

### 8.1. **Demo video link.**

https://drive.google.com/file/d/1FODGB0fLX1GTEJusuuJcv6vQsmQN76PS/view?usp=sharing

### 8.2. **Demo slides link.**

https://docs.google.com/presentation/d/12gLSiqRFKGKp1VFn6QrPRFY1lkh13yt0l1DLiPiYWJk/edit?usp=sharing

### 8.3. **Github links.**

1. Ultrasonic, acceleration and Wi-Fi (for $1^{st}$ BL475E)
   https://github.com/yuarmy0921/eslab_final_socket
2. NFC and Wi-Fi (for $2^{nd}$ BL475E)
   https://github.com/yuarmy0921/eslab_final_nfc
3. Server, car controller (for Raspberry Pi), and final project report
   https://github.com/joshuanieh/embedded_system_final_project

### 8.4. **References.**


1. Car Assembly & Setup
   https://www.waveshare.com/wiki/PiRacer_AI_Kit
2. NFCEEPROM
   https://github.com/ARMmbed/mbed-os-example-nfc
3. NFC Simple Message Parser
   https://os.mbed.com/docs/mbed-os/v6.15/apis/simplemessageparser.html
4. Ultrasonic
   https://os.mbed.com/users/goeltanu/code/HCSR04/
5. NFC Tool
   https://play.google.com/store/apps/details?id=com.wakdev.wdnfc&hl=en&gl=US