

AR:IN

Autonomous Robotics: Intersection Navigation

GROUP 11
Proof of Concept

Derek Arts
Jonathan Klawunn
Kate Keskikyla
Kerianne Rikley
Nabil Hamza
Trent Stevenson

The largest technical concerns that our group has are the cars ability to communicate, its image recognition, its ability to detect obstacles, lines and intersections, and technical failure handling.

The plan to work with communication issue is to have each message start off with a 3 bit identifier first bit showing if it has an error within the sender's system followed by 2 bits divided as shown in figure 1. Any additional information needed in the message immediately following the identifier. After the message details, the next bits will be the size of the queue. The size of the queue will account for all the cars stopped at a stop sign with no other cars physically in front of them. Finally the message ends with the status of the intersection. A zero indicates that there is no car currently going through the intersection, informing the next car in the queue to start proceeding through the intersection. A one will indicate that there is a car currently in the intersection. Using these messages, the car will be able to collect the necessary information from surrounding vehicles to permit the safe navigation of an intersection. A proposed step by step interaction between two cars that meet in an intersection using this messaging system can be found in figure 2. In the event where the communication system in a car fails, it will rely on observing and sensing its surroundings to decide when it is safe to enter the intersection.

Figure 1: Message Break Down

- Acknowledge: a one way signal sent after receiving any other type of signal

Identifier: X00	Message: TBD	Queue Size: 2 bits	Intersection Status: 1 bit
-----------------	--------------	--------------------	----------------------------

- Stop: a signal sent out to all connected vehicles at arrival

Identifier: X01	Message: TBD	Queue Size: 2 bits	Intersection Status: 1 bit
-----------------	--------------	--------------------	----------------------------

- Go: a signal sent out when you are first in the queue after receiving a through signal

Identifier: X10	Message: TBD	Queue Size: 2 bits	Intersection Status: 1 bit
-----------------	--------------	--------------------	----------------------------

- Through: a signal sent out after clearing the intersection

Identifier: X11	Message: TBD	Queue Size: 2 bits	Intersection Status: 1 bit
-----------------	--------------	--------------------	----------------------------

Messages that must be paired with any of the four message types outline above

- Alert: a signal sent out if there are any technical failures or abnormalities in the system

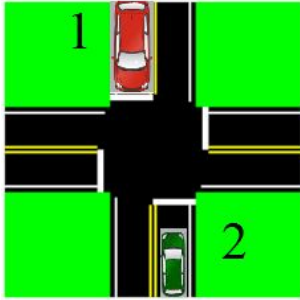
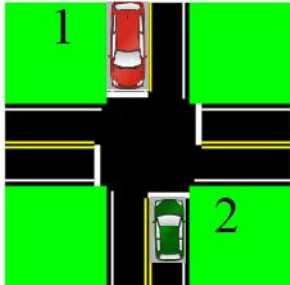
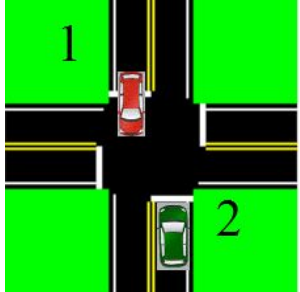
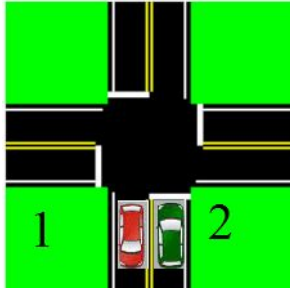
Identifier: 1XX	Message: TBD	Queue Size: 2 bits	Intersection Status: 1 bit
-----------------	--------------	--------------------	----------------------------

- Normal: a signal sent out when system is functioning as expected

Identifier: 0XX	Message: TBD	Queue Size: 2 bits	Intersection Status: 1 bit
-----------------	--------------	--------------------	----------------------------

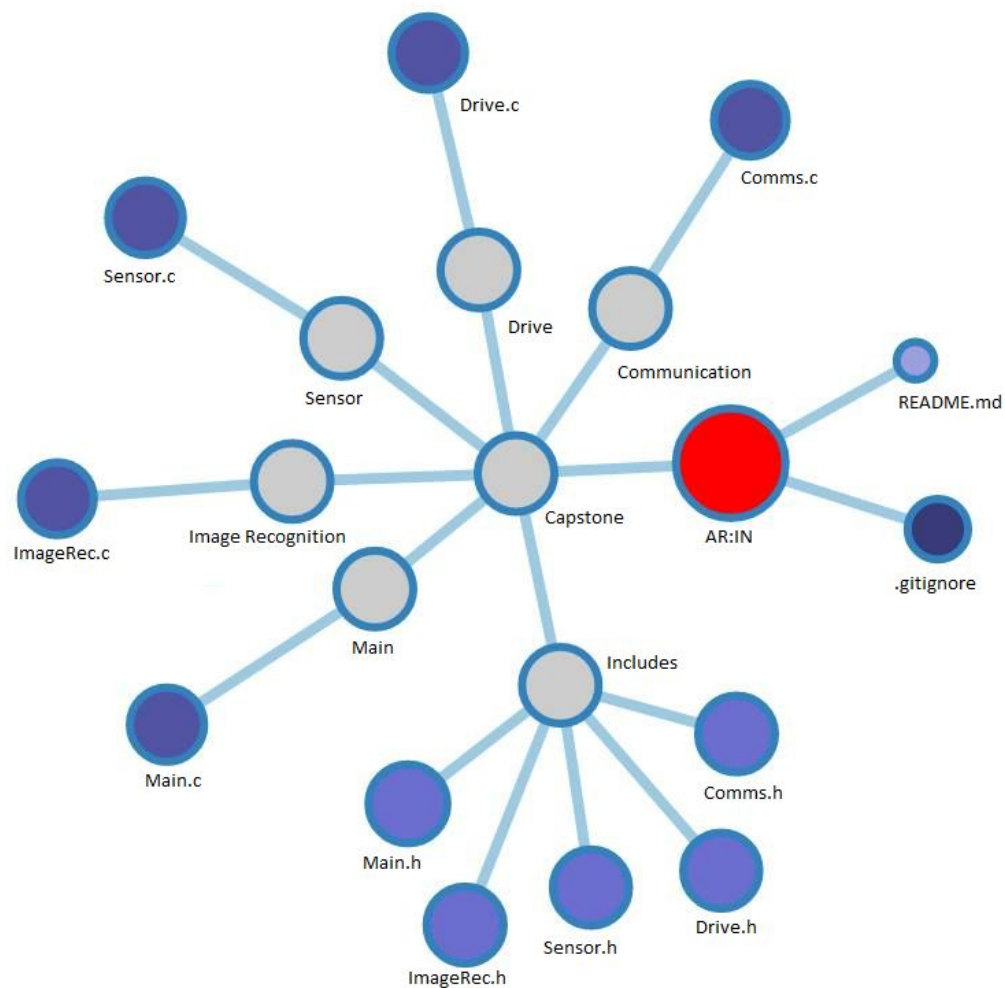
The strategy AR:IN will employ to handle image recognition will be using a series of filters on a grayscale image retrieved from the cameras. This processes starts with some low pass filters, gaussian or otherwise, to clear away noise, followed by some high pass filters to sharpen the edges. With the high contrasting image the prewitt edge enhancer can now be applied to intensify the edges. What remains is an image with defined edges which will use shape detection to identify lines, stop signs and less defined obstacles.

Figure 2: Standard Communication Between two Cars that Arrive One After the Other

 <p>(A)</p>	<p>First Car Stopped at Intersection while the Second is Approaching</p> <p>Car 1 stops at intersection and sends stop message to all connected cars.</p> <p>Car 2, approaching the intersection, receives the stop message from car 1 and returns an acknowledge message with no queue information.</p>
 <p>(B)</p>	<p>Both Cars Stopped at Intersection</p> <p>Once stopped, car 2 sends stop message to all connected cars. Car 1 receives the acknowledge and stop messages from car 2. Since no queue information was received in the acknowledge message, the stopping message was received after the acknowledge message, and did not see any cars in or stopped at the intersection upon arrival (determined with image recognition/sensors), car 1 determines it is first in the queue. Car 1 sends acknowledge message with new queue.</p>
 <p>(C)</p>	<p>Second Car is Stopped while the First Car Enters the Intersection</p> <p>Car 1 sends out go message, locking intersection and passing priority to next car in queue.</p> <p>Car 2 receives go messages and sends acknowledge message, updating the queue and busy status.</p>
 <p>(D)</p>	<p>First Car has Passed through the Intersection, and the Second Car Prepares to Enter</p> <p>Car 1 sends through message after lines switch from dotted to solid.</p> <p>Car 2 receives through message and sends acknowledge message to car 1, sends go message to car 3 if present.</p>
<p>Assumptions: Connection is established between the cars</p> <p>More Cars Arriving: When a new car arrives to the intersection it will have similar interactions with the vehicle at the front of the queue. Every car is responsible for keeping track of their position in the queue (assigned upon arrival)</p>	

To handle technical failures, redundancies will be built into the system. To tackle any software failures, the system design will be modular, as seen in Figure 3, so any technical failures can easily be isolated. The car system will also implement default settings for the program to run when a failure arises. For the hardware, many devices will be used for collecting all types of data that are required, such that when one component fails, the information it was collecting is not lost. The system will also incorporate an indicator on the car to let us (the passenger) know when a hardware failure has occurred. For example, the system can use an infrared sensor or sonar in addition to the cameras to gather obstacle data. This would allow the car to still see obstacles even if the cameras fail. Such a failure would trigger an ALERT message to be sent to all other cars nearby so they can better handle any abnormal behaviors caused by the technical failure, and the internal indicator to signal the passenger of the failure.

Figure 3: Basic Software Structure



In the event where the car is relying on a single component for collecting a piece of data, it will navigate to a safe location and await repairs. If an error occurs that results in the complete loss of a particular piece of data, such as multiple backups failing in quick succession, the car will use the last piece of data to try and reach a safe stopping location.

As can be seen in figure 4 and figure 5 below, modular code for both comms.h as well as comms.c have already been started. The modular form of the code, as was mentioned earlier, will allow for isolation of any issues that may arise, and will also allow for specific portions of the code to be tested for potential bugs.

Figure 4: comms.c Modular Code

```
1  /*
2  * Comms.c
3  *
4  * Created on: Nov 18, 2016
5  * Author: Paul
6  */
7
8  /*****
9  *                               Includes
10  *****/
11
12  #include "Comms.h"
13
14  /*****
15  *                               Global Variables
16  *****/
17
18  /*****
19  *                               Functions
20  *****/
21
22  void sendMessage(message m){
23
24  }
25
26  void getMessage(){
27
28  }
29
30  void processMessage(message m){
31
32  }
33
```

Figure 5: comms.h Modular Code

```

1  /*
2   * Comms.h
3   *
4   * Created on: Nov 18, 2016
5   * Author: Paul
6   */
7
8  #ifndef INCLUDES_COMMS_H_
9  #define INCLUDES_COMMS_H_
10
11  /*=====
12   * Includes
13   *=====*/
14
15  /*=====
16   * TypeDefs
17   *=====*/
18  typedef enum{
19
20      STOP = 0,
21      ACKNOWLEDGE = 1,
22      GOING = 2,
23      ARRIVED = 3
24
25  }identifier;
26
27  typedef struct{
28
29      identifier ident;
30      int queuePos;
31      int intersectionLock;
32      unsigned short fullmessage; // place holder for datatype of messages
33
34  }message;
35
36  /*=====
37   * Defines
38   *=====*/
39
40  /*=====
41   * Function Prototypes
42   *=====*/
43
44  void sendMessage(message m);
45  void processMessage(message m);
46  void getMessage();
47
48
49
50  #endif /* INCLUDES_COMMS_H_ */

```