[ET4388] Adhoc Networking
Lab manual (2016-2017)

Embedded Software group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

September 26, 2016

1 Introduction

This document describes hardware and software guidelines required to get the robot car up and running for the Msc course (IN4388) Adhoc Networking. The document provides a description of all APIs that can be used to control the robot.

2 Hardware overview

The overview of hardware is shown in Figure 1.

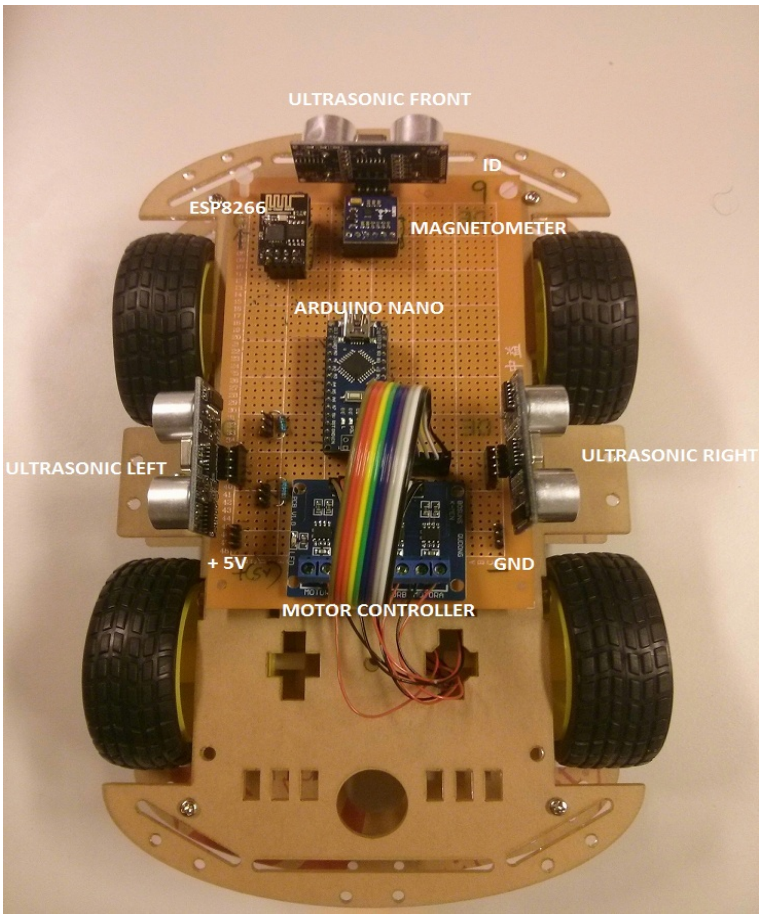

Figure 1: Robot hardware overview

3 Packet format

Figure 2 represents the packet format to be used by Arduino for communication.

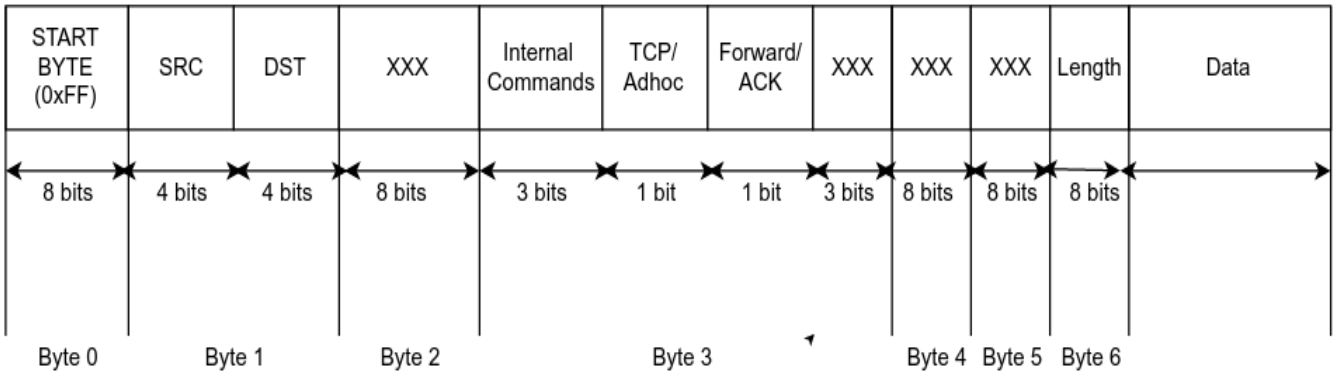

Figure 2: Packet format

Contents of each field is explained below

- **START BYTE** : Its the first byte. Represents the start sequence of the packet. Its is denoted by 0xFF.
- **SRC** : A 4 bit field of the second byte represents the unique identifier of the source Ad-hoc node.
- **DST** : A 4 bit field of the second byte represents the unique identifier of the destination Ad-hoc node.
- **Internal Commands** : This 3 bit field is used for communication between Arduino and ESP. It represents certain commands that are exchanged between Arduino and ESP during initialization. Internal command should be set to '0' when communicating with TCP or Ad-hoc node (other robot). It should be set only when the robot communicates with its ESP8266 module.
- **TCP/Adhoc** : 1 bit information used to identify whether the packet is intended for TCP server or Ad-hoc network.
  Note: When communicating with TCP, TCP bit should be made high and data should be packed as per the length. Rest of the bytes such as SRC, DST, Internal commands, Forward/ACK are don't cares.
- **Forward/ACK** : 1 bit information used to identify if the packet is ACK packet or a forwarding packet.
- **Length** : Represents the length of data in bytes present in the packet.
- **Data** : Represents data to be sent. This field can be up to 254 bytes in size.
- **XXX** : This byte should not be changed by the user. This is reserved for internal purpose.
- **Checksum** : Represents checksum to make sure that the complete packet is received. At present, this is not implemented in the server, ESP8266 or Arduino.

4 Network structure

The robot supports two modes of communication. In TCP mode, it can communicate with the TCP server to receive commands from it or send data to the server. In Adhoc mode, it can communicate with other robots without any intervention from the server. This can be done by setting **TCP/Adhoc** bit in the packet. Setting this bit sends the data to the TCP server whereas clearing this bit sends the data to the Adhoc network. Similarly, this bit can serve as an indication if the packet originated from the server or from the Adhoc network.

The robots can communicate with each other based on a network graph which is represented as a matrix in each Arduino. Consider an example of a simple network of four robots as show in figure 3.

In this topology, it can be observed that robot **1** can communicate only with robot **2**. Whereas robot **2** can communicate with robots **1**,**3** and **4**. This relation is represented in form a matrix as follows.

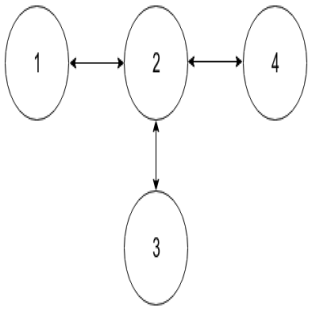

Figure 3: Sample Network graph for a set of three bots

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

5 Initialization

During the initialization routine, the Arduino sends initialization commands to the Wifi module. These commands are needed by the Wifi module to connect to the specified access point and the TCP server. The data sent to ESP8266 are: ID of the robot, connection matrix, IP address of teh server, and SSID and password of the AP. This can be seen as 4 blue blinks with a delay of one second on the Wifi module. After this, the Wifi module tries to connect to the specified access point which can be seen as pulsating blink on the Wifi module.

6 TCP Server

The TCP server provides a way to send commands to each robot individually by means of TCP sockets. The server is able to establish one-to-one communication with the robots in order to send commands or retrieve data from it. The server ip address should be configured in the Arduino code. This information is sent to the Wifi module during the initialization routine.

In order to execute the Server type the following command.

```
./tcpserver <Number-of-bots>
```

The server should be turned ON and should be connected to the AP before any of the clients (robots) are turned ON. The server waits for the robots to get connected. It displays the ID of the connected bots as soon as it gets connected.

```
*************** Server control program (Adhoc networking course)  ************
* Binding done
* Waiting for bots to connect
  - Accepted connection
  - Bot with ID : <2> Connected
```

In order to send command to a specific connected robot type the ID of the bot you wish to communicate with and select the appropriate command from the menu.

```
Enter Bot ID to send the packet
Enter the command(1-12) to the bot-2 :
  1. Move forward
  2. Move forward for time in seconds
  3. Move reverse
  4. Move reverse for time in seconds
  5. Move left time
  6. Move right time
  7. Stop the bot
```

```
 8. Get obstacle distance left
 9. Get obstacle distance right
10. Get obstacle distance front
11. Get RSSI value
12. Execute commands from file (cmd_file.txt)
```

NOTE : Guide to turning Android phone into wifi hotspot can be found here. Likewise guide to turning laptop into a hotspot can be found here.

7 Flashing the Software to Arduino

The software for the robot is written in standard Arduino IDE which can be downloaded from Arduino download page.

- Select Arduino nano from the **Tools** >**Board** menu
- Select the correct serial port selected from the **Tools** >**Serial Port** menu
- Then simply press the upload button in the Arduino environment. The board will automatically reset and the sketch will be uploaded.

8 API listing

This section gives an overview of the APIs that can be used on the TCP server and the Arduino.

8.1 Server APIs

The APIs that can be used on the TCP server are summarized in table 1

Table 1: Server APIs

NAME	USAGE	DESCRIPTION
send_forward_time	void send_forward_time(int src, int dst, int time)	This API is used to move the bot with ID **dst** forward for **time** seconds. Time can be any positive number between (0-254). Specifying zero in time field moves the bot in forward direction indefinitely.
send_reverse_time	void send_reverse_time(int src, int dst, int time)	This API is used to move the bot with ID **dst** reverse for, **time** seconds. Time can be any positive number between (0-254). Specifying zero in time field moves the bot in reverse direction indefinitely.
send_rotate_left	void send_rotate_left(int src, int dst, int time)	This API is used to rotate the bot with ID **dst** left for **time** seconds.
send_rotate_right	void send_rotate_right(int src, int dst, int time)	This API is used to rotate the bot with ID **dst**right for **time** seconds.
stop_bot	void stop_bot(int src, int dst)	This API is used to stop the bot with ID **dst**.
get_obstacle_data	int get_obstacle_data(int src, int dst, int sensor_num)	This API is used to obtain the reading from obstacle sensor of the bot with ID **dst**. **sensor_num** is used to select the sensor on the bot. **sensor_num** can be ULTRASONIC_LEFT for left obstacle sensor. ULTRASONIC_RIGHT for right obstacle sensor and ULTRASONIC_FRONT for front obstacle sensor.
get_RSSI	long get_RSSI(int src, int dst)	This API is used to get received signal strength of the bot with ID **dst**.

8.2 Robot (Arduino) APIs

The APIs that can be used on the robot (Arduino) are summarized in table 2

Table 2: Arduino APIs

NAME	USAGE	DESCRIPTION
setup	void setup(()	This API is called once as soon as the Arduino is powered ON. All initializations takes place here.
loop	void loop()	This API is called repeatedly soon after setup.
CreatePacket	void sendPacket(char src, char dst,char internal, char isTCP, char isACK, char counterHigh, char counterLow, char dataLength, char *data)	This API can be called to send a packet over WiFi to TCP or ADHOC node. The API forms a packet containing **data** (and also other details such as source ID) and send to TCP/destination (**dst**) robot
OnReceive	void OnReceive(char src, char dst, char internal, char tcp, char fwd, char counterH, char counterL, char datalen, char command, char *data)	This API is a callback function which is called every time a packet is received by Arduino.
getDistanceFront	unsigned long getDistanceFront()	This API is used to get the distance of the nearest obstacle (in cm) from the front ultrasonic sensor.
getDistanceLeft	unsigned long getDistanceLeft()	This API is used to get the distance of the nearest obstacle (in cm) from the left ultrasonic sensor .
getDistanceRight	unsigned long getDistanceRight()	This API is used to get the distance of the nearest obstacle (in cm) from the right ultrasonic sensor.
moveForward	void moveForward()	This API is used to energize the motors such that the robot moves forward.
stopMotors	void stopMotors()	This API is used to de-energize the motors and stop locomotion of the robot.
moveForwardForTime	void moveForwardForTime (char *time)	This API is used to make the robot move forward for a specific amount of **time** seconds.
moveBackForTime	void moveBackForTime (char *time)	This API is used to make the robot move back for a specific amount of **time** seconds.
moveBack	void moveBack()	This API is used to energize the motors such that the robot moves back.
turnLeft	void turnLeft(char *time)	This API is used to rotate the robot left for **time** seconds.
turnRight	void turnRight(char *time)	This API is used to rotate the robot right for **time** seconds.
getRSSI	void getRSSI()	This API is used to get the RSSI of the robot with respect to the AP.
getID	char getID()	This API returns the ID of the robot.
setID	void setID(char ID)	This API sets the ID of the robot.
enableDemo	void enableDemo()	This is internal API used to enable demo mode in ESP8266. Enabling Demo will show the complete demonstration of ad-hoc network. For eg., if the connection matrix is as shown in Figure 3, the packet transmitted from Node 1 will reach Node 3 automatically (handled in ESP8266 code). In non demo mode, the packet from node 1 will reach node 2 only. Demo mode should be enabled in all the robots to make the logic work.

9 Commands

The internal and external commands used in the project are as follows.

9.1 Internal Commands

The internal commands are represented by bit 6 to bit 8 (3 bits) in 5th byte of the packet. These commands are used by Arduino to communicate with ESP8266. The commands are always sent from Arduino to ESP8266. The commands used are:

- INT_ID : Represents that data contains ID
- INT_SSID_PWD : Represents that data contains SSID and Password
- INT_MATRIX : Represents that data contains its connection matrix
- INT_RSSI : Represents that Arduino is requesting RSSI value from ESP8266. The data length can be zero since this is request command
- INT_IP : Represents that data contains IP address of the server
- INT_DEMO : Enable demo mode in ESP8266

Note: When internal commands are sent, the first byte in data does not contain COMMAND unlike external commands.

9.2 External Commands

The internal commands are represented in first byte of data (make sure that Internal commands are set to zero). Followed by external command, arguments are sent in data section of the packet.

The external commands are sent by server to a robot in TCP mode, or from one robot to another in ad-hoc mode. The external commands used are:

- NOCOMMAND : Do nothing
- MOVEFORWARD : Command to move forward
- MOVEFORWARDTIME : Command to move forward for specific amount of time
- MOVEBACK : Command to move back
- MOVEBACKTIME : Command to move back for specific amount of time
- TURNLEFT : Command to turn left for specific amount of time
- TURNRIGHT : Command to turn right for specific amount of time
- STOP : Command to stop movement
- DISTANCELEFT : Command to get obstacle distance from left
- DISTANCERIGHT : Command to get obstacle distance from right
- DISTANCEFRONT : Command to get obstacle distance from front
- GETID : Command to get ID of the robot

Note: Students can add their own external commands.

10 Important Notes

- The power supply for the robot is 5V (6V if you use 4 AA batteries). If sufficient voltage is not provided, then unexpected behaviors such as restart of Arduino/ESP8266 on moving robots, are observed.
- To program Arduino, ESP8266 should be removed from the PCB as same TX RX pins are used to program Arduino. Also, note down the direction of ESP8266 (which is different on different PCBs) on PCB so that it can be inserted in right way again.
- It is better not to remove any of the modules (except ESP8266) from PCB unless you note down the pin - header mapping (direction of module on PCB). If the modules are inserted in opposite direction, VCC and GND pins may be reversed, damaging the module.
- Make sure that the ID stored on Arduino EEPROM is same as that printed on the robot PCB. If not, then ID of the robot can be set using setID API that can be called from setup().
- Students are allowed to change ANY part of the program (server or Arduino code) at their own risk.
- enableDemo() should be used only for understanding the ad-hoc network logic during project development and should not be used during final demonstration (exam).
- The robots are equipped with 3-axis MAG3110 magnetometer. Interested students can use this to make the robot to rotate at specific angles (for eg. rotate left at 50 degrees). The source code for initializing magnetometer and reading its value is not provided.
- The provided connection matrix in the source code should not be changed.
- Reading distance using ultrasonic sensor may take upto 0.5s when there are no obstacles as getDistanceXXX API is set to blocking mode. The pulseIn() function in getDistanceXXX() waits until the GPIO pin is high. The API (getDistanceXXX) can be set to non blocking mode by implementing GPIO and timer interrupt.
- The robot is under development stage. For any problems/bugs in code/questions related to the robot, you can contact Mr. Pranav Sailesh - P.S.Mani@student.tudelft.nl (for Arduino code/Robot hardware), and Mr. Renukaprasad - R.B.Manjappa@student.tudelft.nl (for server/ESP8266 code).