TEAM SCARLET

EE 98 – Senior Design Final Report

Delay Tolerant Network for Autonomous Robotic Vehicles

Victor Ansart, Gerard Denoyer, Tolga Zeybek

Sponsor: Hwa Chang, Tufts Wireless Laboratory

Abstract

Autonomous robotic vehicles are getting increasingly popular. Uses for these vehicles include operations in remote areas where access to a structured network may be unavailable, making reliable communication an issue. Limited battery power of these vehicles is another issue that impairs their ability to perform autonomous duties for extended time periods. We propose a new delay-tolerant-network (DTN) topology that allows the vehicles seamlessly communicate with each other and additional nodes, and a novel autonomous charging system utilizing our network to solve the problem of limited battery power for extended autonomous duties. We designed a cost-effective, heterogeneous, ad-hoc DTN to carry and relay information between robotic vehicles, charging stations and additional nodes, where nodes can collect and store data from each other to relay to additional nodes. Furthermore, we developed an automated charging process, where autonomous robotic vehicles collect, share, and store information about the charging stations they have encountered using our DTN network. When the vehicles need charging, they use the data stored in their memory to autonomously navigate to the charging stations and connect with them using image detection to charge their batteries, eliminating any need for outside intervention. To demonstrate the autonomous charging process, we prototyped two autonomous robotic vehicles controlled by smartphones. These smartphones run a custom Android application we developed to perform wireless communication over DTN using the ZigBee protocol. The vehicles navigate autonomously using GPS and on board ultrasonic distance sensors, and precisely connect to charging stations by using the smartphone camera to detect specific patterns. Additionally, we built a prototype charging station that also acts as a gateway between our DTN and the Internet. We had partial success with the entire project where the individual parts worked, but the integrated did not properly work as a whole.

Introduction

The past decade has brought incredible innovation to the fields of power storage, alternative energy sources, communication networks, artificial intelligence, and device interactivity. In terms of these technologies, the automobile of 2014 is fairly similar to the automobile of 2004. However, notable exceptions such as the Tesla Model S and Google's driverless concept car have inspired consumers to expect more out of their automobiles. In this project, we imagined what challenges the automobile of tomorrow would face. We assumed that in the near future, vehicles will be (1) electric-powered, (2) autonomous, and (3) able to communicate as part of the Internet of Things.

With this scenario in mind, our goal was to create a system in which autonomous electric vehicles can navigate themselves to charging stations as needed and charge themselves without human interaction. For this to be successful, we wanted the system to be able to function outside the range of typical communication systems like WiFi and cellular networks. We set out to design a system in which a GPS-enabled autonomous bot could navigate itself to a charging station whenever it needed to charge its batteries. The bot, charging station, and other relay nodes would be integrated into a communication system so the bot and charging station can send status and location updates to each other even when they are too far removed to communicate directly.

This project builds on the previous work of Tolga Zeybek, who had already designed an autonomous vehicle as part of research during his sophomore and junior years.

Theory (or Problem Background)

Previous Research

A study conducted in 2011 found the accuracy of the GPS on the smartphones to be between 5.0 and 8.5 m on average, which is an error margin 2 to 3 times larger than the standalone GPS units [8]. Although this margin was still acceptable for autonomous navigation, it led us to add a standalone GPS receiver for our first ARV. Using two ultrasonic distance sensors on the autonomous robots is found to be reliable for avoidance [9]. We based our obstacle avoidance on this approach and increased the number of sensors for a better resolution. We also placed one of the sensors facing to the ground to monitor surface irregularities such as potholes. It is shown that sharing the charging stations to serve multiple ARVs extends the range and the duration of their operations [10]. In this approach, ARVs recognize the charging stations by detecting short-range infrared signals emitted by the stations, but they do not know where the stations are when they are outside the infrared range. Another approach is to autonomously move the charging station alongside the ARVs, which requires complex algorithms and a method to supply power to the mobile charging station [1]. Wireless sensor networks are used effectively to gather data outdoors, with a vast range of applications from airport perimeter detection [11] to underwater data collection [12]. However, majority of the uses rely on multiple reliable connections between static nodes for data transfer. Our application requires communicating with mobile nodes, which decreases the reliability of wireless sensor networks [12]. A DTN is found to be a reliable topology where most connections are unreliable and intermittent [13]. The most important issue about the DTNs is ensuring reliable and on-time transmission of the data due to the intermittent connection between the nodes. Several methods are proposed to increase the data transmission throughput that could be applied to our DTN topology [13]. Our DTN has mobile nodes, which increases the rate of spread of data but decreases the long-distance network reliability by not having continuous network paths between all nodes. Since a vehicle would benefit mostly from nearby roadside hazards and charging stations rather than the far ones, our application does not require the transmission to geographically far nodes from the origin of data over the DTN.

Project Design

The project is designed as the integration of three major parts; the DTN, the GPS based autonomous navigation, and the autonomous charging using image processing.

The DTN element requires making the physical hardware of the nodes as black boxes, which will be able to communicate wirelessly over the DTN and communicate with the structures they are installed in over TTL Serial. It also requires to design the software protocol for the DTN, where the devices will be able to acknowledge each other when passing by and synchronize to share data, and carry on further communication based on their needs.

The autonomous navigation requires designing the Autonomous Robotic Vehicles (ARV) to be able to seamlessly work in rough outdoor surfaces. Even regularly paved terrains such as asphalt could pose risks for little robots. Moreover, the ARV's should be equipped with sensors for obstacle detection, nodes for DTN communication, and Bluetooth for communicating with the mounted smartphone. This increases the number of devices to be integrated, and brings complexity to the project. On the software level a custom Android app is developed to perform the GPS based navigation and takes advantage of the GPS and compass capabilities of the smartphone, as well as external sensor input for obstacle detection. A reliable GPS signal may be difficult to obtain at all times and phone location estimate can suddenly jump from the fine precision GPS location to coarse network location, which is the greatest problem about the GPS based autonomous navigation and something we have not properly took into account.

The GPS is not accurate enough to dock into a charging station. Therefore, the autonomous charging part requires an image processing algorithm using the integrated camera without exploiting the limited processing power of the phone, and designing of patterns on the charging stations that are easy to recognize by the smartphone under different weather and lighting conditions. Moreover, an easy to dock and safe charging interface for both the ARV and the charging station is needed that will control the charging voltage according to requests from the DTN.

Method of Solution

Methods, Materials, & Equipment

DTN Communication

Communication over the DTN depends on wireless serial communication utilizing the ZigBee protocol. We define four types of nodes with common encryption standards and a common baud rate for seamless data exchange. All four nodes use the XBee Series 1 transceiver by Digi for the wireless transmission of serial data over the Zigbee protocol. The data is secured by built in 128-bit AES encryption on the firmware of the transceiver and the transceiver is configured to operate on a specific channel and a four-digit Personal Area Network ID (PANID) to broadcast only to transceivers with the same channel and PANID over the 2.4 GHz carrier frequency. The baud rate is set to 9600 bits/sec and range is up to 100 meters for the regular Series 1 and can be upgraded up to 1500 meters by using the Series 1 Pro with line of sight.

1) Vehicular Node: A vehicular node can be installed on the ARVs and other types of vehicles. The DTN transceiver is wired to the Arduino Mega, which acts as the internal gateway of the ARV by parsing and relaying the data between the smartphone and the vehicular node. Communication is done by exchanging structured data sentences as strings. An ARV periodically broadcasts a short sentence referred as a flag that contains the ARV identification information. The ARV sends an extended data sentence upon receipt of a flag from another ARV. This sentence includes current GPS coordinates of the ARV, three station flags indicating coordinates, availabilities and technical capabilities of the closest three charging stations , and additional information gathered from the onboard sensors or general purpose nodes, i.e. the coordinates of nearest roadside hazards. If a charging station flag is received, the ARV records the information to its internal memory. It may choose to communicate further with the charging station if it decides to be charged at that station.

2) Charging Station Node : The DTN transceiver of a charging station node is wired to the controller of the station, which is typically an Arduino Uno platform. The station periodically broadcasts a flag over the DTN that contains the GPS coordinates, type, availability, and technical information of the station. Further communication is made with the ARVs that decide to be charged at the station. A sample station flag sentence structure is "#+4240814971121448&1!084!000!000". In this charging station flag "#" is the charging station identifier symbol, "+42408149-71121448" are the GPS coordinates of the charging station corresponding to the location at +42.408149, -71.121448, "&1" is the station availability (1 is available, 0 is unavailable), and "!084!000!000" indicates the available charge voltages at the station, which is just 8.4V (084) at this station.

3) General-Purpose Node : The DTN transceiver of a general-purpose node is wired to the controller of the charging station, which is typically an Arduino Uno platform. It periodically broadcasts a flag that contains the GPS coordinates of the node and the information to be sent. It only acts as a transmitter beacon and does not listen for incoming data. Its intended uses include on-site marking of the roadside hazards to warn any ARV in vicinity. Since these places are likely to be in remote areas, these nodes are designed with reduced functionality to decrease the power consumption, and they are ruggedized against harsher environmental conditions. A sample sentence structure is "*+42408149-71121448&L05CURVE-AHEAD". In this flag "*" is the general purpose node identifier symbol, "+42408149-71121448" are the GPS coordinates of the charging station corresponding to the location at +42.408149, -71.121448, "&L0" indicates the speed limit as 5 mph, and "@CURVE-AHEAD" is the custom message to be broadcasted. The general purpose node is a future addition to the project.

4) Gateway Node : Gateway nodes are designed to connect the DTN to the Internet and other networks using the HNP. They can be standalone or added on to other nodes. A computer, or any device that has a network connection capability can act as a standalone gateway node by addition of a DTN via a USB or an RS-232 serial port, which is the way our DTN connects to the HNP. A charging station node can also act as a gateway node by addition of hardware such as Arduino Ethernet or Arduino Wi-Fi shields to enable Internet connection. The gateway node is a future addition to the project.

Autonomous Robotic Vehicles

We designed two prototype ARVs to test and demonstrate the capabilities of our concept. Both ARVs use Android smartphones as the master control device, and share some peripheral devices. The second prototype is an upgrade to the first one in terms of more processing power, better GPS connection, stronger chassis, more traction power, and availability of the image detection and the charging hardware.

1) Hardware: The first ARV is built on a four-wheel drive aluminum chassis, featuring an Android 2.3.5 SAMSUNG SGH I-897 smartphone as the master processing and control platform. The smartphone is paired to the ATmega2560 microcontroller on the Arduino Mega 2560 platform via a JYMCU Bluetooth-to-Serial adapter. The Arduino governs all the peripheral devices on the ARV via hardwired connections, with the exception of an external Bluetooth GPS receiver that is directly connected to the smartphone for enhanced GPS sensitivity. The peripheral devices connected to the Arduino are five HC-SR04 ultrasonic distance sensors placed in front, back, sides, and top of the ARV, four DC traction motors via an H-Bridge based AF motor motor controller, and a vehicular DTN node. Two 7.4V, 2.2Ah, LiPo batteries supply power to the ARV. Dimensions of the first ARV are 23 by 20 by 15 cm, which is shown in the Fig. 1.

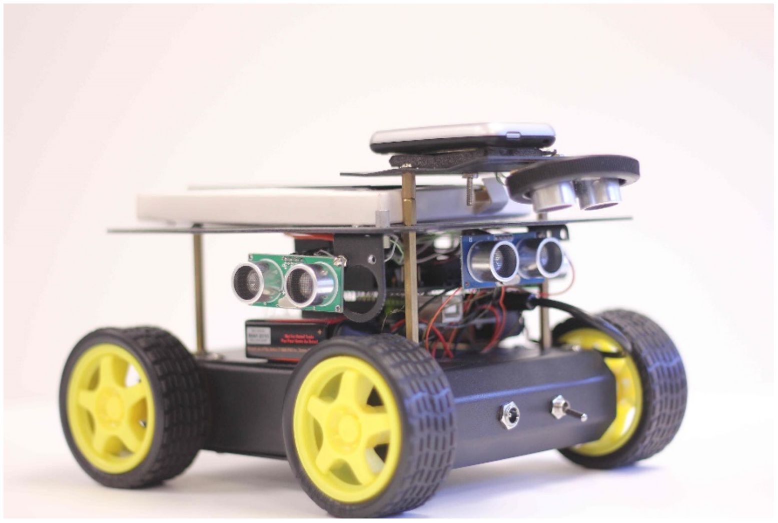


Figure 1 The first ARV

The second ARV is built on a six-wheel drive Wild Thumper ruggedized aluminum chassis with individual suspension, featuring an Android 4.3.3 Sony Xperia Z smartphone as the master processing and control platform. The smartphone is paired to the ATmega2560 microcontroller on the Arduino Mega 2560 platform via a JY-MCU Bluetooth to- Serial adapter. The Arduino governs all the peripheral devices on the ARV via hardwired connections. The peripheral devices connected to Arduino are six HC-SR04 ultrasonic distance sensors placed in front, back, sides, and at the top of the ARV, Atmega168 based Wild Thumper motor control and battery monitoring circuit, and a vehicular DTN node. This ARV does not feature an additional GPS module due to the adequacy of internal GPS of the smartphone. Additionally, it features a charging interface mounted to the front bumper, and a periscope to transform the camera angle of the horizontally mounted smartphone to the forward direction. A single 7.4V, 5Ah NiMH battery powers this ARV. Dimensions of the second ARV that is shown in Fig. 2 are 45 by 30 by 15 cm.

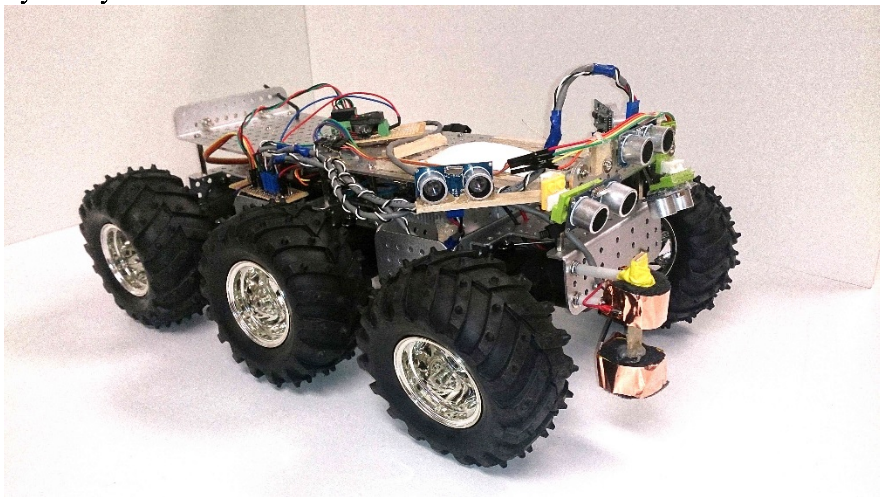
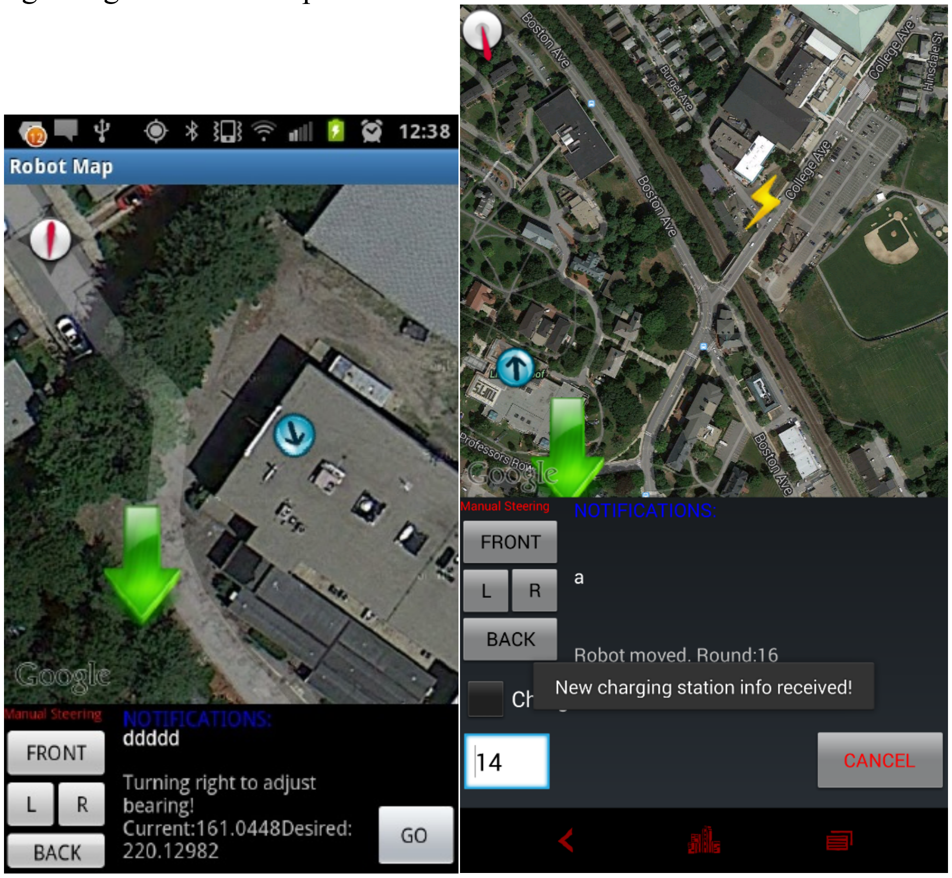


Figure 2 - The second ARV

2) Software: We developed two types of software for the joint operation of the smartphone and the Arduino Mega to control the ARV. The first software type is for the Arduino Mega, called an Arduino sketch. Written in C++, it takes advantage of the open-source Arduino IDE. The sketch periodically monitors the distance data from the on-board ultrasonic distance sensors, and parses the incoming TTL serial messages from the vehicular DTN node to send to the smartphone over the Bluetooth-to-Serial connection when demanded. Arduino Mega also controls the turning direction and speed of the traction motors (or communicates over the TTL Serial with the commercial motor controller for the second robot) as directed by the smartphone, and relays the messages to be sent by the smartphone to the vehicular DTN node. The second type of software is the custom Android application on the smartphone, which acts as the master control and processing software. We chose the open source Android over the iOS of Apple to have the opportunity of cross-platform development on Windows, Linux and Mac OS using Java, and to take advantage of Android's relatively simple Bluetooth communication protocol. Our Android application consists of a graphical user interface (GUI) layer and a background layer that manages the autonomous navigation, communication with the DTN, and communication with the ARV peripherals. For the second ARV, the background layer additionally monitors the state of charge, and governs the autonomous charging process by using image detection to precisely connect to the charging stations. The GUI layer, as shown in Fig. 3, consists of an embedded Google Maps interface centered at the current location of the ARV. This location is also overlaid as a light blue circle on the map that has a dark blue arrow showing the current heading of the ARV. Any received charging station location via the DTN is overlaid on the screen as a yellow lightning bolt at the map location of the station.



ddddd

Turning right to adjust bearing! Current:161.0448Desired: 220.12982

a

Robot moved. Round:16

New charging station info received!

Figure 3 GUI layers of the Android Apps for first (left side) and second (right side) ARVs

The user can choose the destination by tapping on the map, which creates a green arrow over the tapped area to mark the destination. Autonomous navigation starts when the user taps the GO button on the bottom right corner. The status of the ARV and the received messages over the DTN are displayed as text labels to the left of the GO button. On the bottom left corner, four direction buttons enable manual control of the ARV over the Bluetooth. The algorithm for the autonomous navigation runs in a background thread in the application and is activated when the user presses the GO button or when the ARV autonomously decides to go to a destination, i.e. for charging purposes. We summarize the structure of this algorithm below:

- The application calculates the bearing angle to the destination using the current GPS coordinates of the ARV, the GPS coordinates of the destination, and by subtracting the proper declination angle.

- The smartphone starts requesting the ultrasonic distance data from all sensors with a period of 0.5 seconds.
- The ARV turns around its own axis to align its heading with calculated bearing angle using internal compass sensor of the smartphone.
- The ARV goes forward while the received data from the ultrasonic sensors are within a safe distance limit and the ARV heading is consistent with the bearing angle that is updated every 10 seconds.
- If the sensor data shows a distance less than the safe limit, then the ARV stops and calculates the best maneuver to avoid the obstacle using the ultrasonic sensor data, and performs the maneuvers until the obstacle is cleared.
- If the heading is not within the 5°margin of the calculated bearing angle, then the ARV stops.
- The bearing angle gets recalculated for the current coordinate of the ARV and the navigation restarts.
- The navigation successfully ends if the current ARV coordinate is within the two meter radius of the destination.

If the ARV determines that avoiding an obstacle is not possible, it stops and notifies the user. A separate thread runs on the background to handle the DTN communication. The ultimate success of navigating with this algorithm depends on the accuracy of the GPS location calculated by the phone. If the phone is in urban areas with big obstacles, or under adverse environmental conditions the location will alternate between the fine GPS fix and the coarse network based location intermittently making the navigation inaccurate. The current implementation of the algorithm only notifies the user about the accuracy of the location to the nearest meter on a text label and does not distinguish location sources or act against inaccuracies in location, which sometimes causes poor navigation accuracy. More precautions such as stopping navigation until an accurate GPS fix is established, and making circular maneuvers in the absence of GPS signals as an attempt to get a better angle of the clear sky should be added to the algorithm implementation for better accuracy in navigation.

3) Integration: To ensure seamless operation of all parts of the ARV, our controller application on the smartphone needs to establish reliable three types of connections with the peripheral devices. The first type of connection is on software layer with the internal compass sensor, the GPS receiver and the camera of the smartphone via the Android API. The second type uses a serial connection over the Bluetooth to the Arduino Mega 2560, and additionally to the external GPS module for the first ARV. The third type is done by wiring the DTN vehicular node, the ultrasonic distance sensors and the motor controller circuit to the Arduino, and exploiting the second type of connection for between the Arduino and the smartphone. The system integration diagram for the first ARV is provided in Fig. 4, where red, blue and black arrows show the first, the second, and the third type connections respectively. The second ARV has an additional first type connection to the smartphone camera in addition to the connections shown in Fig. 4.

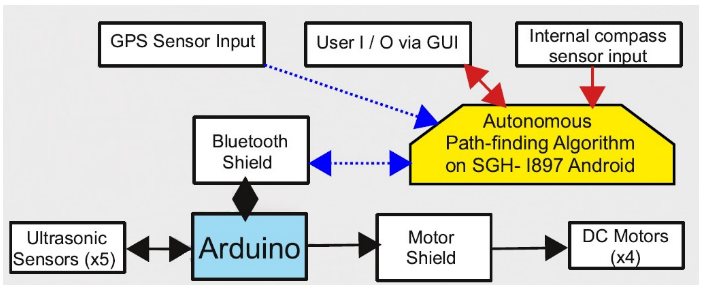


Figure 4 - Integration diagram of the first ARV

Charging

The ARV charging is the showcase application that utilizes all features of the ARV and the DTN that we have designed. A successful charging process requires all the subsystems on the charging stations and the ARVs to work harmoniously via the DTN connection. The prototype charging station is designed as a cylindrical prism with a diameter of 100 cm and a height of 50 cm. On the outer surface, two copper strips run through the entire circumference horizontally. The hot side of the charging voltage is carried on the upper strip and the ground side is carried on the lower strip as illustrated in Fig. 5. The upper circumference of the charging station is marked with four equally spaced black and white square-in-square patterns, which is a unique pattern for our pattern detection algorithm. Enclosed in the charging station is an Arduino Uno platform with an ATmega328 microcontroller, which is connected to a charging station DTN node and a DC relay that controls the charging voltage on the hot side copper strip. Optionally, an Arduino Ethernet Shield or a Wi-Fi shield may be connected to provide the capabilities of a gateway node. The charging station also utilizes a layer of red LED strips that turn on when voltage is present on the copper strips to warn people nearby about the voltage presence.

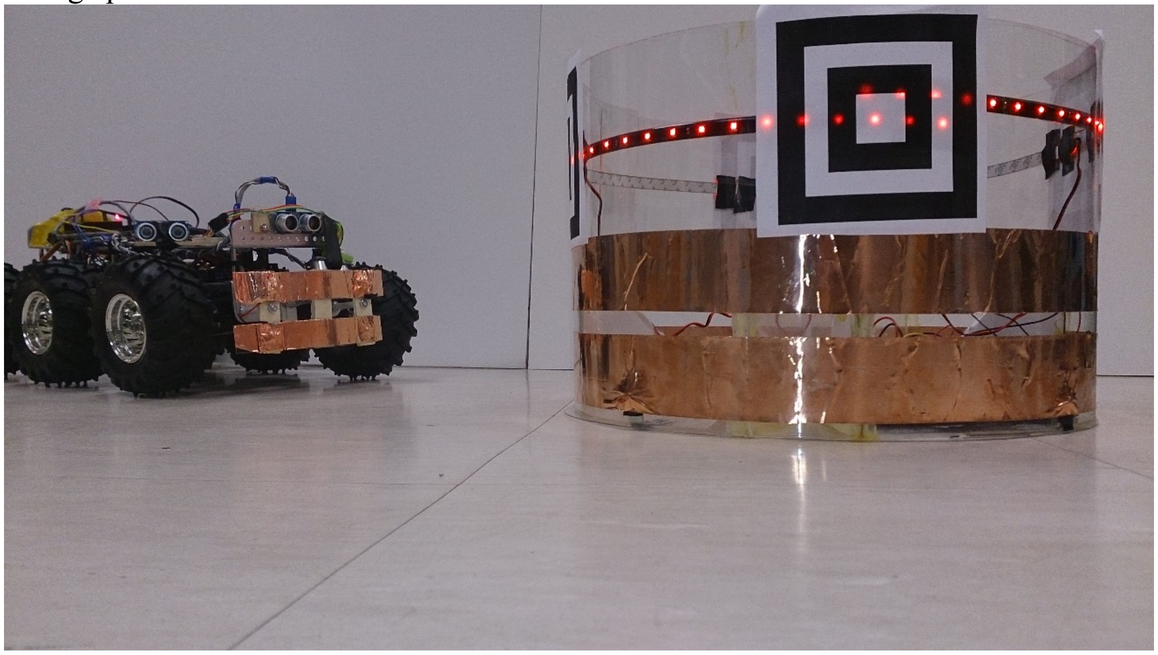


Figure 5 - Charging station with voltage present for the approaching ARV

1) Decision for Charging and Navigation to the Charging Station: Both ARVs can read, store and distribute the data obtained from the charging station nodes of the DTN. For the prototype application, we built a super-capacitor discharge circuit to simulate the charge and discharge process of the actual vehicle battery, but in less time. The 1 microfarad super capacitor is determined to be fully charged at 4.5V, and immediately starts discharging as soon as charging is finished over a resistive element with indicator LEDs until the voltage drops to 2.0V, which is decided to be limit of discharge. This simulation circuit will be referred as the battery in the following paragraph that explains decision for charging.

The ARV monitors its battery voltage by connecting the hot side of the battery to the internal Analog to Digital Converter (ADC) input of the Arduino Mega labeled as A IN2 via the voltage regulating resistors R1 and R2 as shown in Fig. 6. If this value falls below a certain threshold, the ARV pauses its current task and heads to the closest, available, and technically compatible charging station stored in the memory. The threshold is determined experimentally and hardcoded into the software as the battery voltage that allows the ARV to run for an additional 5-6 minutes (this is for an ideal case, in current simulation case the threshold of 2.0V is determined arbitrarily), which is an ample time to reach a charging station in our test scenario. Charging station information in the memory may come directly from stations passed by, or may be rebroadcasted by other ARVs. Rebroadcasting significantly increases the probability of the data distribution since nodes share their information as they encounter other nodes along their paths.

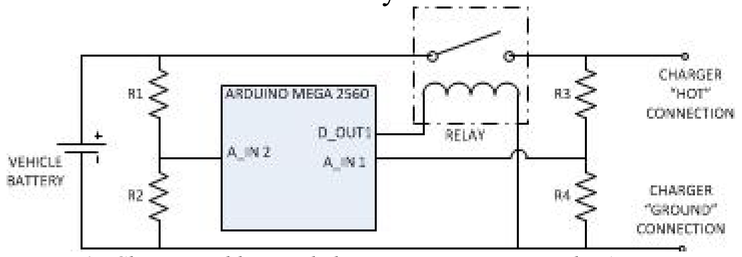


Figure 6 - Charge enabling and charge sensing circuit in the ARV

2) Charging process: The ARV uses our autonomous navigation algorithm to get in vicinity of the chosen charging station. (This capability is partially achieved and worked only for the first ARV that does not have charging functionality, since we did not have enough opportunity to fully test the second ARV before the motor control circuit malfunctioned.) When it is in the DTN range of the charging station node, it updates the charging station information, proceeds to be charged if the station is available, and notifies the station of its presence. It continues using the same algorithm until it reaches the two-meter range of the station. In the meantime, the station switches-on its DC relay to provide the charging current to the copper strips and broadcasts its unavailability to the rest of the ARVs.

3) Image Processing for Precise Charging Connection: We have developed a pattern recognition algorithm that looks for a square-in-square pattern in the smartphone cameras line-of sight using the OpenCV open source image processing library. This algorithm runs on a separate activity in the application with its own GUI, when the main activity requests image detection. Our pattern detection algorithm detects the location of the

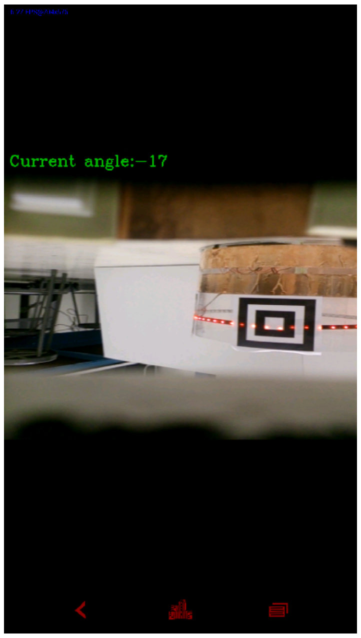


Figure 7 - Pattern Detection

pattern on screen and returns the adjustment angle to align the heading with the pattern. The adjustment angle is used to maneuver the ARV to the correct heading to align with the pattern that is placed on the charging station for roughly a second, then the algorithm recalculates the angle and directs the ARV maneuvers the ARV again until the pattern is within $\pm$ 5° of the center of the screen. At this time the robot is instructed to go forward for approximately 0.5 meters and stop to make angle readjustment via the pattern detection algorithm. The process continues until the robot physically connects to the copper strips on the charging station and the external voltage from the station is sensed by the robot. Fig. 7 shows this algorithm in action. In this case a pattern is successfully detected that is 17° degrees to the right of the center vertical line of the screen, and the ARV is directed to turn left in the background process. Note that the DTN communication already took place to enable the charging voltage on the charging station side, as indicated by the illumination of the red LED strips. The algorithm is capable of distinguishing a pattern oriented within $\pm$ 33° of the viewing angle.

4) Charging at the Station: When the alignment is established, the ARV drives forward until the charging contacts in front of the ARV successfully contact the copper strips on the charging station, an instance just before the contact is shown in Fig. 5. The ARV detects this by the presence of the charging voltage on the hot side charging contact, which is connected to the ADC on the Arduino Mega using the A IN1 input via the voltage regulating resistors R3 and R4. The Arduino then allows charging by writing a high value to the D OUT1 pin to switch on a relay on the charging path as shown in Fig. 6. When the battery voltage reaches the high limit the Arduino sends a disconnect request to the charging station to end charging. The charging station switches-off the charging voltage and informs the ARV over the DTN that it is safe to disconnect, while changing its status as available. Since image detection takes a considerable amount of processing power, only the second robot running on the more powerful Android 4.3.3 platform has the autonomous charging ability in our test case.

Results

Test Component	Success	Failure	Acceptance Criteria	Test Result
Accurate GPS navigation		X	Bot navigates itself within 4m of charging station 4 out of 5 times	Cloudy weather prevented bot from locating charging station. More reliable when sunny.
Functioning DTN system	X		Data transmission occurs within 5 seconds of node overlap	Transmission consistently below 5 seconds, typically 1–2
Image processing algorithm	X		Correct heading angle detected from within 4m 4 out of 5 times	Image recognized and angle determined 4 of 5 times
Consistent charging	X		Charging connection made 4 of 5 times that bot determines heading angle	Final charging interface successful in nearly every instance
ARV controller & bot construction		X	Bot can navigate to a point 10m away within 30 seconds	Bot was unable to consistently drive to given coordinates

Table 1: Success and failure of major components. Acceptance criteria and test results determine success.

As is seen in Figure 8, the bot was able to recognize the inscribed square pattern whenever it was in its field of vision and quickly determine how much of an angle separated it from the bot's centerline.

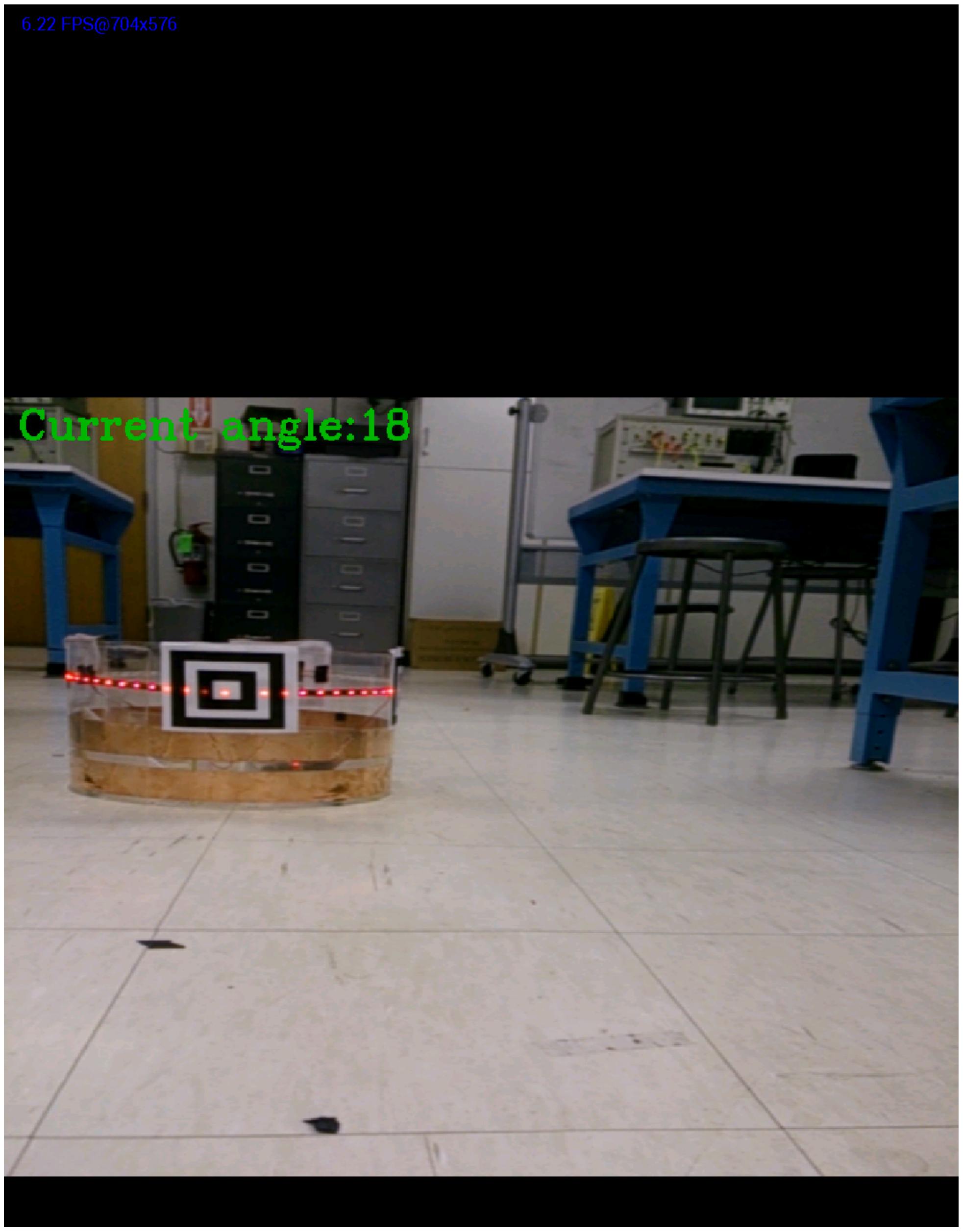
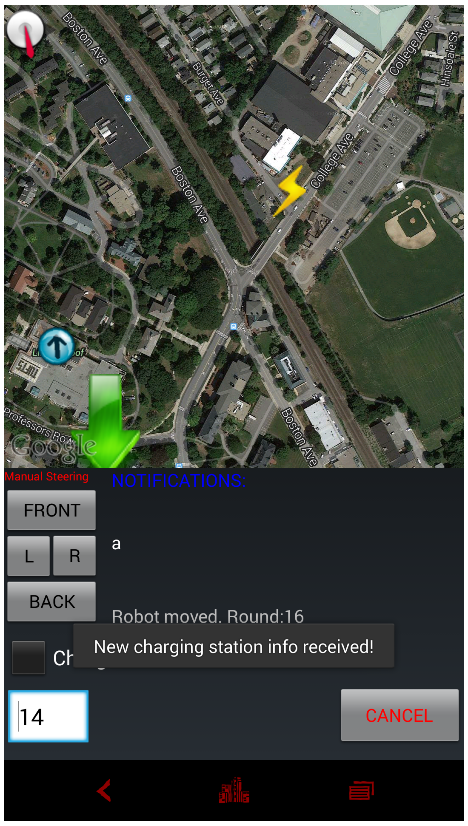


Figure 8: Screenshot of bot's vision through the periscope. Bot recognizes station is 18º from center.

Figure 9 shows the GPS system on a sunny day. The bot is accurately able to determine its location and heading. At this point, the location remained static and the bot experienced no errors. However, when GPS coverage was spotty, the bot would sometimes believe its location had jumped slightly on the map and navigation became spastic and suffered.



a

Robot moved. Round:16

New charging station info received!

14

Figure 9: Screenshot of GPS app. Bot was located in Tisch Library while charging station remained outside Halligan.

Analysis

The results in Table 1 show that we were successful constructing the majority of the main components. The delay tolerant network worked consistently and quickly, image processing was accurate and reliable, and the addition of wider strips and a spring connection made the charging interface very successful. However, the unreliability of the Android's built-in GPS caused frustrating problems during testing. Finally, a short circuit damaged the bot's main controller board, and the unavailability of a replacement part meant that testing had to be cut short before the GPS could be improved. As a result, we were left with a collection of functional but separate components, not a unified system.

The Delay Tolerant Network system that we created was able to exchange its data with another node within 5 seconds of connecting with each other. The nodes of the DTN could also distinguish between the various different nodes they were connecting to (vehicular, charging station, general purpose, and gateway nodes) and respond accordingly. We were successful in using the DTN to relay information across multiple nodes. A vehicular DTN node going by the charging station was able to pick up the charging station's location and then relay that information to another bot that was running low on battery. The bot that needed charging was then able to interpret the GPS coordinates of the charging station and calculate a route to the station.

We were also successful in creating a robot that would find a charging station when its batteries ran low and re-charge itself when the bot was within 4 meters of the station. The bot was able to detect the charging station using an image-processing algorithm that searched for a specific pattern on the station. The image-processing algorithm was very reliable and would detect the pattern whenever it fully came into view of the camera as long as there was no motion blur.

Unfortunately, we had difficulty programming the robot to autonomously navigate to a GPS location. Under optimal conditions, the location algorithm functioned as desired, but it would not work if there was a weak GPS signal. The weak signal would cause the estimated location of the bot to jump around which confused the robot and caused it to act unpredictably.

The unreliability associated with the GPS navigation made it very hard to debug problems while testing as it was during a period of weak GPS signal. During one test when the robot went off course, it hit a bump, which caused a wire to disconnect and cause a short in the ARV motor controller. Following this incident, the motor controller behaved erratically and it was impossible to continue testing with the main robot. This prevented us from being able to test the entire system working together when the GPS signal would have been stronger. Therefore, we were not able to achieve a completely working prototype.

Conclusions

In conclusion, we had mixed success creating the system described in the introduction. Because the system did not function as described, we have failed at our overall goal. However, many of the individual components created for the system performed better than required and will create a very successful system when the remaining bugs are fixed.

The app design and interfaces taught us cross-platform design including Java and C++. Additionally, the challenges we faced in the final weeks reinforced the importance of front-end planning. If testing had taken place in the weeks prior, the controller isolation issue could have been identified soon enough to order replacements that would have allowed us to fix the underlying GPS problem.

Luckily, we are left with several important takeaways. The Tufts Wireless Lab now possesses delay tolerant network code that can be used in a variety of applications. The code was intentionally written to be modular and not proprietary to our project so that it is quickly adaptable to other uses. Future students are also left with two functioning bots that can be used to further refine this project and add functionality of their own. Just as our team benefitted from previous Senior Design groups advising us on the use of OpenCV image processing algorithms, we have learned much about its functionality and our code can be used by other groups in coming years.

Recommendations

The main improvements that could be made at this point center around the two components that failed during testing– the GPS and ARV controller.

The bot relied on the onboard GPS installed into its Android phone. This was a simple solution that interfaced easily with the bot’s app, however, the GPS signal and accuracy proved unsatisfactory. Previously, we tested separate GPS units that were much more reliable and accurate in all scenarios. Installing one of these units would be the first change we would make when continuing work on the system. This would solve navigation problems because the bot would better be able to (1) determine its own starting location, (2) plot a path to its destination, and (3) resolve its location while moving on its path. When the phone GPS occasionally jumped, the bot would behave spastically and suddenly change its course. More error detection could also be added into the code to slow the bots response so that rapid changes do not cause the bot to immediately change course.

The bot's structure and fasteners are predominantly metal, so it proved to be prone to short circuits. One hard-to-find short damaged the bot's main controller board. To avoid this in the future, we would attempt to isolate all major circuit components and connecters to avoid the possibility of accidental grounding. Insulating all the different circuit boards will improve reliability and give testers a much easier time debugging.

With these two areas improved, there would be significantly lower barriers between the system and its goals. Beyond these, a more effective collision detection system could keep the bot safer. The charging connectors rely on the bot colliding with the station, so they are vulnerable to unintentional crashes when the bot is moving at its faster cruising speed and colliding into harder objects. The ultrasonic sensors on board may be used to remedy this issue.

References

[1] A. Couture-Beil and R. Vaughan, "Adaptive mobile charging stations for multi-robot systems," in Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, Oct 2009, pp. 1363–1368.
[2] "Ieee standard for local and metropolitan area networks–part 15.4: Lowrate wireless personal area networks (lr-wpans)," IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006), pp. 1–314, Sept 2011.
[3] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks," Communications Magazine, IEEE, vol. 43, no. 9, pp. S23–S30, 2005.
[4] Z. Yang and H. Chang, "An innovative unified platform for heterogeneous network communications," 2014.
[5] R. Gao and H. Chang, "A scalable and flexible communication protocol in a heterogeneous network," 2014.
[6] "Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications - redline," IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999) - Redline, pp. 1–1238, June 2007.
[7] "Ieee standard for ethernet - section 1," IEEE Std 802.3-2012 (Revision to IEEE Std 802.3-2008), pp. 1–0, Dec 2012.
[8] P. A. Zandbergen and S. J. Barbeau, "Positional accuracy of assisted gps data from high-sensitivity gps-enabled mobile phones," The Journal of Navigation, vol. 64, pp. 381–399, 7 2011. [Online]. Available: http://journals.cambridge.org/article S0373463311000051
[9] C.-C. Wang, C.-L. Lin, K.-H. Hsia, and Y.-C. Hsieh, "Obstacle avoidance and wireless network surveillance of a weapon robot," in Computer

Communication Control and Automation (3CA), 2010 International Symposium on, vol. 2, May 2010, pp. 261–264.
[10] F. Michaud and E. Robichaud, “Sharing charging stations for long-term activity of autonomous robots,” in Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on, vol. 3, 2002, pp. 2746–2751 vol.3.
[11] A. Davis and H. Chang, “Airport protection using wireless sensor networks,” in Homeland Security (HST), 2012 IEEE Conference on Technologies for, Nov 2012, pp. 36–42.
[12] ——, “Underwater wireless sensor networks,” in Oceans, 2012, Oct 2012, pp. 1–5.
[13] N. Ng, H. Chang, Z. Zou, and S. Tang, “An adaptive threshold method to address routing issues in delay-tolerant networks,” in GLOBECOM Workshops (GC Wkshps), 2011 IEEE, Dec 2011, pp. 1122–1126.