Aylish Wrench
Annie Phan

Directional Communication Via LED Lights

Abstract:

A robot's motives are often difficult for humans to interpret. Unlike humans, robots do not convey their intentions through body language. As a result, humans cannot always understand a robot's intended path of travel. Turns, in particular, are hard to predict. In order to fix this, we explore the idea of implementing turn signals on a robot. We created a ROS node that monitors the current global path of the robot. When the node detected an upcoming change in direction, the node published a message to a particular topic. A sketch uploaded to an Arduino board controlling the LED light strip attached to the robot subscribed to this topic. It then altered the lights based on the messages it received. This method produced turn signals that were effective the majority of the time but issues with the robot's localization and serial port prevented them from being fully effective.

1. Introduction:

The Building Wide Intelligence (BWI) segway bots are currently limited in their ability to communicate with humans. The only method for communication is the laptop controlling it. This laptop can display a variety of information including a map showing the current path the robot intends to take to its destination. However the range at which one can view this information is extremely limited. The laptop screen can only be seen from the back of the robot. Therefore, anyone standing in front of the robot has no idea where the robot's goal is. Furthermore, even with perfect eyesight, there is limit to how far a human can stand behind a robot before they are unable to interpret the information on the screen.

One consequence of the robot's inability to communicate its intended actions is that it is difficult for humans to tell when and where a robot is going to change direction. This produces a variety of potential problems for human-robot interaction. For example, in order to get to its destination in the quickest and most effective way possible, a robot may plan to turn and cut through a particular hallway. However, a group of humans may be gathered in the hallway and blocking the robot's path. If they recognize that the robot wants to travel down the hallway, the humans may move out of its way. But without knowing that the robot is planning to turn into the hallway, the humans may assume that the robot will carry on traveling straight and bypass the hallway. Therefore, they will make no effort to move out of its way. The robot will then be unable to pass through the hallway and have to take an alternative route which may be much longer. Ignorance of when and in what direction a robot intends to turn could also threaten the safety of humans and the robots. Although the robots have obstacle avoidance protocols, a major error in the robot's software or sensors could lead to the robot not noticing a nearby human. A sudden turn could therefore lead to damage to the human and/or robot.

A potential solution to this problem lies in another human invention: the automobile. All modern cars have flashing lights that not only tell observers that the car is about to turn but the direction of the turn as well. The meaning of these lights, better known as "turn signals," are easily recognized by most humans. Because of the ease of understanding, we believed that this approach would prove effective if adapted for the segway bots. To implement the turn signals,

Aylish Wrench
Annie Phan

Arduino-controlled LED lights were first attached to a robot. The placement of these lights is shown in Figure 1. Software was then written to monitor for upcoming turns and flash a portion of the lights to signify when and where the robot was going to turn. For example, when the robot was going to take a left turn, all of the lights on the left half of the robot flashed while the half on the right remained unchanged. These LED lights are easy to see from almost all directions of the robot and from large distances away. Furthermore, because they model a concept humans are already familiar with, it does not take long for a human to recognize the meaning of the flashing lights. The lights ultimately provide a simple but effective way of communicating the robot's future actions to humans.

Figure 1: The placement of the LED light strip on the robot. The right turn signal is active in the figure.

2. Background and/or Related Work:

The importance of humans understanding a robot's goal and direction of travel was examined in a study by Lichtenthaler and Kirsch[1]. In their research, they compared "the legibility factors goal-predictability and trajectory-predictability" [1]. Goal-predictability was the ability of humans to understand the destination of the robot while trajectory-predictability was the ability of humans to understand the direction the robot was traveling. One conclusion of the study was that "trajectory-predictability is the more important factor in a human-robot interaction scenario when considering the factors safety, comfort, and reliability" [1]. In other words, humans responded to the robots better when they knew the direction of travel for the robot rather than the final goal. This means that turn signals could prove to be useful for improving human-robot interaction.

Using physical features to indicate direction of movement has been examined by a multitude of research groups. Many, however, like May, Dondrup and Hanheide [2], focused on more humanoid robots (robots with heads and other human-like body structures). Studies using these types of robots tend to center on using head orientation to represent direction. The segway bots, however, do not have moveable heads. Therefore, using gaze to represent directional intent is not possible. The study by May, Dondrup, and Hanheide [2] is notable however because the researchers compared the head orientation method to a flashing LED lights method similar to our own. They found that although both methods were successful in improving human's ability to

understand the robot's movements, the visual light method was "significantly better understood" than the head orientation [2]. A similar study by Shrestha, et al. [3] looked at using flashing lights to indicate the robot's intended direction versus using a blinking onscreen arrow to indicate the direction an approaching human should take to avoid the robot. The researchers discovered that the lights were "rated significantly better" to the arrow [3]. Both [3] and [4] show the effectiveness of turn signals as a means of communicating directional intent of a moving robot.

Another notable study is one by Szafir, Mutlu, and Fong [4]. These researchers looked at using lights to signal the direction in which a flying robot was going to move in. Szafir, Mutlu, and Fong tested four different light behaviors, one of which was a design similar to that of a turn signal on a car. They found that this behavior "improved participants' speed and accuracy predicting robot intent" [4]. This study not only shows how effective turn signals are on human-robot interaction but shows that this method can be applied to robots with vastly different body structures and functions.

3. Technical Approach:

To implement the turn signal on the robot, we wrote an Arduino sketch to control the LEDs. The Arduino sketch would change the LED's colors and control the blink rate. We also wrote a ROS node to monitor the path the robot would take. The ROS node would send messages to the Arduino sketch when a turn signal needed to be activated.

3.1 Arduino Sketch:

Arduino sketches primarily consist of two functions, setup and loop. Setup executes only when the Arduino is powered up. Loop is continuously called while the Arduino is active. Our sketch was different than most Arduino sketches however. We used the rosserial_arduino package to turn our sketch into a ROS node. This allowed it to get information from other nodes by subscribing to a topic. In the setup function, we set up the digital pin, the default array of color objects, and the subscriber. The loop function did the actual work of blinking the LED lights. An array where each element represented the color of a single light was filled with a particular color. The sketch would then read a message from the topic it subscribed to and remove color objects from the array based on the message received. The topic published characters that determined which portion of lights that needed to turn off. If the sketch received the character '1', it would blink left. If it received a '2', it would blink right. Finally, its default state where nothing blinked was the character '0'. The function would then switch back and forth every half second between the default array with all colors active and the other array where a portion of the lights were turned off. This created a blinking action.

3.2 ROS Node

The next step was to write the ROS node that would examine the robot's current path for upcoming changes in direction and send messages to the Arduino if a turn signal needed to be activated. To do so, we first had to subscribe to the robot's pose and its global path. We chose to subscribe to the global path instead of other topics such as the local path because the global path

provided us with a vector of points that the robot planned to travel to and was easily accessible. Within the main part of the code, we set up a publisher to send messages to the Arduino and two subscribers: one for the robot's current pose and another for the global path. The node then iterated through the first half of points in the global path. If there was a major change in direction, the node would publish the appropriate message to the topic. It also recorded the resulting orientation after the turn. When a turn signal was active, the node would check if the robot's current orientation was close enough to the expected orientation. If so, it would send another message to return the lights to the default setting.

3.3 Combining Code:

After writing the code, we then had to figure out the communication between the ROS node and the Arduino sketch. To do this, we ran a simple command on the command line to open up the serial port which the Arduino could use. After making some adjustments to our sketch, we were able to open the serial port so that the sketch could properly receive messages to change the lights. One issue with using the serial port however occurred when we ran the segbot launch code. Other nodes using the serial port interfered with the messages being sent to the Arudino. The only solution we found was to kill those nodes before opening the serial port.

4. Experiments and/or Evaluation and/or Example Demonstration:

	Did it Work as Expected?		
Trial Number	Straight	Right Turn Signal	Left Turn Signal
1	Yes	Somewhat (signal did not turn off after turn)	Yes
2	Yes	Yes	Yes
3	Yes	No	Somewhat (right side blinked briefly)
4	No	Yes	No
5	Yes	Somewhat (left side blinked briefly)	Somewhat (signal did not turn off after turn)

Figure 2: Table showing the results of five trial runs. A "Yes" means the signals worked as expected with no bugs. A "Somewhat" means the signals worked as expected but with one or two minor bugs. A "No" means the signal did not work at all as expected.

To run our code, we had four terminals running simultaneously. The first terminal ran the segbot launch files. The second ran the turn_monitor node that examined the robot's current path for turns. The final terminal ran the code to open the serial port.

One problem we encountered when testing the robot was figuring out the best distances of paths to test on. For short distances that included turns, the robot would immediately flash the turn signal because it was expecting to turn shortly. We found that longer paths were better for testing as the robot would not signal until after it had moved forward a short distance. We also discovered that when a path included two turns in the same direction with a short distance between them, the turn signal would not terminate between them. Therefore, for testing, we use paths where the robot moved forward and then turned once and then moved forward again.

Aylish Wrench
Annie Phan

We ran multiple trials with different types of movement. The results of these trials are shown in Figure 2. We started with five trials where the robot did not make any turns and use these as our control. We wanted to insure that the turn signals would not go off when there were no changes in direction in the robot's path. This also helped us to test that the global path was reliable. We also used these trials to see how much change in orientation was due to the robot's normal movement rather than an actual turn. This helped us to determine a decent threshold for determining whether a turn was about to occur. The second and third set of trials tested the actual turn signals. In these trials, the robot would move straight for a bit and then move around a corner and into another hallway. We then recorded if the turn signal worked as we intended it to. Many of the trials were fairly successful.

As we tested, a number of interesting bugs occurred. Sometimes the robot would briefly flash the opposite turn signal. For example, when about to turn left, the right turn signal would activate for a second and then the robot would correct itself. One possible reason for this is that one of the points in the global path has an orientation which, when subtracted from the robot's current orientation, produces a result that makes the robot think a turn in the opposite direction is coming.

Another issue was the turn signal not terminating once the robot finished turning. This may be caused by the robot looking too far ahead in its path and finding another turn. Another possibility is that the turn monitor node is publishing messages faster than the Arduino sketch can process them. As we tested, we found that the turn signals worked the best when the code to run them was first executed. The longer the code ran, the more bugs occurred. It is likely that as the code is run, the topic is filled with more and more messages. Because the sketch can only handle one at a time and each message takes at least half a second to finish executing, the sketch may still be processing old messages while new ones are coming in. Rerunning the code proved to be effective in stopping bugs but was extremely tedious.

We also encountered a variety of problems not directly related to our code. Localization was another issue we encountered. If the robot was not localized accurately before setting the navigation goal, the robot would easily lose its true position on the map therefore losing the global path altogether which would break the code. These causes impacted the performance of the robot greatly and are hard to correct because they are directly affected by the global path itself, which cannot be altered by our powers currently.

One of the biggest problems we had with our project was having limited access to a working robot. Initially, there was only one robot with LED lights attached to it. This light strip was later damaged beyond repair, forcing us to attach another light strip to a different robot and use that instead. Because of this, some of our code had to be changed to reflect the new robot. The change of robots and the robots continually running out of charge slowed us down considerably and forced us to abandon some of our more ambitious plans such as having different colors when the robot traveled in different directions.

Conclusion and Future Work:

Aylish Wrench
Annie Phan

Although this project seemed simple in theory, the actual implementation of it was more difficult than expected. We encountered many bugs and mishaps throughout the process of writing and testing out the code. One of our biggest problems was time. Because of the limited opportunities we had to test on a fully working robot, we were unable to fully debug our code. We have many ideas about how to fix our programs but did not have time to implement and test the changes. With more time, we can hopefully work out the rest of the bugs and perfect more of the code.

Additionally, some of the bugs we need to fix could be taken care of if we explore other possible solutions. We would like to investigate a way to clear messages in a topic or reduce unnecessary messages being published by the turn monitor node. Upon the reparation of that problem, we would be able to properly test the robot continuously without having to restart a lot of the code. This would save time spent on the robot and reduce the use of the robot's battery. Another suggestion would be to find a way to implement the code more efficiently and easily into the robot. Our code currently requires a three terminals and a large amount of steps to run correctly. Preferably, we would like to execute our code by having one terminal run the segbot code and another terminal run a launch file that would run the turn monitor node, open the serial port, and kill the nodes interfering with the serial communication. Lastly, we want to see if there is a way to get a more accurate current path for the robot instead of using the global path. Perhaps if we used the local plan or other navigation topics, we would discover a more reliable and stable plan we could use.

Ultimately, we hope that in time, we will be able to return to this project and smooth out the mishaps that we identified during our trials. A lot of the problems could be solved by altering our code and actually experimenting with the suggestions. Perhaps some of the solutions proposed here could be the key to perfecting our turn signals. Turn signals could be an interesting and useful addition to the segbots. If we could get the turn signals to work correctly, it would be interesting to see how people would respond to them.

References:

[1] C. Lichtenthaler and A. Kirsch, "Goal-predictability vs. trajectory-predictability: which legibility factor counts," in *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction.* ACM, 2014, pp. 228-229.

[2] A. D. May, C. Dondrup and M. Hanheide, "Show me your moves! Conveying navigation intention of a mobile robot to humans," *Mobile Robots (ECMR), 2015 European Conference on*, Lincoln, 2015, pp. 1-6.

[3] M. C. Shrestha, A. Kobayashi, T. Onishi, E. Uno, H. Yanagawa, et al, "Intent Communication in Navigation through the Use of Light and Screen Indicators," in *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*. IEEE Press, 2016, pp. 523-524.

Aylish Wrench
Annie Phan

[4] D. Szafir, B. Mutlu, and T. Fong, “Communicating Directionality in Flying Robots,” in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction.* ACM, 2015, pp. 19-26.