ARL-TN-0910 • Sep 2018

US Army Research Laboratory

Analysis Techniques for Displaying Robot Intent with LED Patterns

by Yenet Deribe Tafesse, Maggie Wigness, and Jeff Twigg

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TN-0910 ● SEP 2018

US Army Research Laboratory

Analysis Techniques for Displaying Robot Intent with LED Patterns

by Yenet Deribe Tafesse
Science and Engineering Apprenticeship Program (SEAP), American Society for Engineering Education, Washington, DC

Maggie Wigness and Jeff Twigg
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2018	Technical Note	June 2018–August 2018

4. TITLE AND SUBTITLE
Analysis Techniques for Displaying Robot Intent with LED Patterns

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Yenet Deribe Tafesse, Maggie Wigness, and Jeff Twigg

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-CII-A
Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-0910

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
contact author's email: <maggie.b.wigness.civ@mail.mil>.

14. ABSTRACT
Autonomous field robots are often rugged and have limited capabilities to communicate with human teammates when working side-by-side in the field. For this project, we demonstrated how simple, lightweight hardware on the robot can be used to communicate onboard information and robot intent to a human teammate. Specifically, LED signal patterns served as a communication vehicle to visually display a mobile robot's traversal intent. Software was written to interface with the onboard LEDs, and the documentation of this software is included in this technical note.

15. SUBJECT TERMS
visualizing robot intent, LED signaling devices, human-robot communication, SPEAR, small ground robotic platform, ROS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maggie Wigness
Unclassified	Unclassified	Unclassified	UU	18	19b. TELEPHONE NUMBER (Include area code) 301-394-3927

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

There has been a rapid increase in the development and use of robotic agents for civilian and military applications, such as rescue missions, exploration, and mapping of unknown environments. Depending on the application, the robotic platforms either operate autonomously or need to be tele-operated. Many applications require heterogeneous robots, including ground and aerial vehicles, as well as humans to work together to achieve a certain objective. For this type of teaming to work effectively, improving the interaction and communication between the human and robots is crucial. The human has to be able to understand what the robots are sensing and the decisions they are making based on data from different sensors. This will help the human agent take more informed action as needed. The robots acquire lots of data using their various sensors; however, this information is not readily available for the human in real time.

During a 6-week summer project, we focused on demonstrating a system that displays visual LED signals with BlinkSticks[1] to convey the movement of a robot in both tele-operation and autonomous modes. The specific objectives of the project include 1) programming and integrating the BlinkSticks with the Robot Operating System (ROS), 2) acquiring and interpreting the data generated by the robot regarding its motion, and 3) displaying the data in coded colored lights on the BlinkSticks in real time. Our deliverables for the project include 1) a live robot demonstration and video showing the developed functionality in the tele-operation and autonomous modes, 2) a set of Python functions used to control the BlinkSticks, and 3) documentation of the code and research. In this technical note, we provide details about the specific tasks for the two modes of operation used to test the developed BlinkStick functionality.

2. Hardware Description and Modes of Operation

2.1 Hardware Description

The overall system used consists of a small ground robotic platform (SPEAR) developed at the US Army Research Laboratory (ARL).[2] Various sensors can be found onboard the robot, including a laser scanner, an inertial measurement unit, and odometry, which are used for navigation and other applications. Figure 1 shows the robotic platform used over the course of this project. Note, however, that the applicability of the BlinkStick communication is largely agnostic to the robotic platform itself.

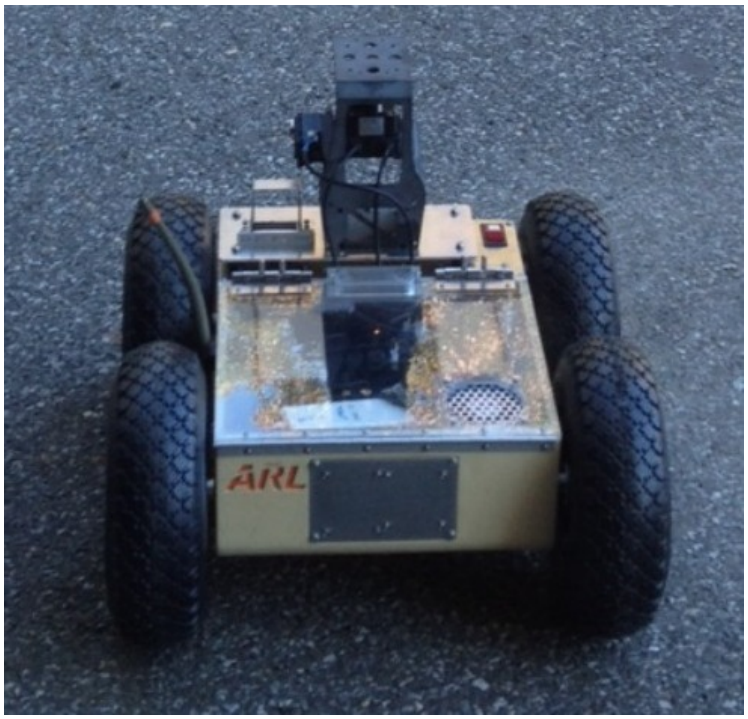

Fig. 1 Small ground robotic platform (SPEAR) robot that was used to interface with the BlinkSticks

To visually demonstrate the movement of a robot, we used BlinkStick Strips (seen in Fig. 2), which are programmable USB sticks that allow control over eight LED lights. In order to interface the BlinkSticks with the robot data, we had to first download the BlinkStick program to the robot from the BlinkStick website.[3]

The BlinkStick module can be easily installed in Linux using pip:

```
sudo pip install blinkstick
```

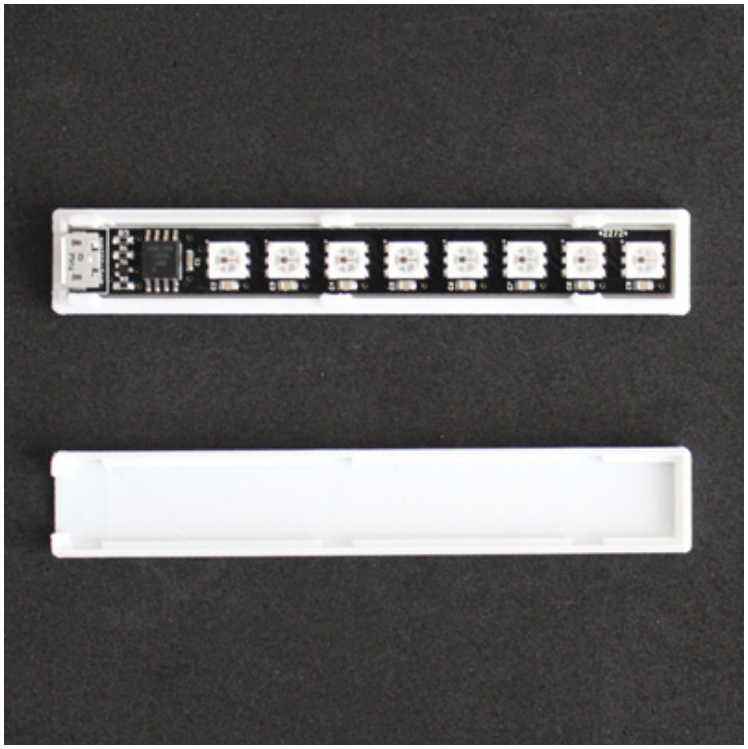

Fig. 2 BlinkStick Strip with eight LED lights used for visual interpretation of data from robot

By default, the BlinkStick device can only be accessed by root. To allow all users access to the device with root permissions, a udev rule can be added:

```
sudo blinkstick --add-udev-rule
```

We interfaced the BlinkStick with the SPEAR using ROS. The listener can subscribe to any topic, interpret the data, and use the existing BlinkStick application programming interface (API)[4] and our additional functionality to visually communicate via the LED lights. In the following subsections, we provide two examples of how the interfacing was performed during this project: when the SPEAR was being tele-operated and when it was running autonomously.

2.2 Tele-operation Mode Testing

We used two BlinkSticks and the data from the *cmd_vel* topic to visually indicate the current movement of the robot. This topic publishes data regarding the direction and movement of the robot (e.g., turning left or right, backing up, or moving forward). Using the data, we created turn signals, break lights, and flashing back-up lights.

To display these signals, we used the functions *blink_up_stick* for right and left turns and *set_all_indices* for backing up and break lights, which can be found in the function documentation in Section 3. This code can be found in the listener callback function.

From the *cmd_vel* topic, we got the linear velocity and the angular velocity of the robot. For linear velocity, we looked at linear X, which told us everything we needed to know about the robot's forward or backward movement. From the robot's movements and echoing the topic, we figured out that X is negative when the robot is backing up and X is positive when the robot is moving forward. Based on that data, we were able to apply our code to indicate if the robot was going forward or backward.

From the angular velocity, we looked at angular Z. By driving the robot and echoing the topic, it was clear that Z is negative when the robot was turning right, and Z is positive when the robot was turning left. Based on that data, we were able to implement our code to indicated when the robot was turning right or left.

While testing the BlinkStick, there was a lagging problem. The rate at which the data was being obtained was a lot faster than the rate at which the data was being processed. To fix this problem, we added a timer to the listener so it would run the callback function when the timer calls it, not when it gets new data.

2.3 Autonomous Mode

We also used the BlinkSticks when the robot was operating autonomously. We obtained the data from the *global_plan* topic to read what the robot was planning to do in autonomous mode. In order to implement the BlinkStick code in this mode, we first had to understand the data that we were getting. From this topic we got two sets of data: the position and the orientation. The position is the X and Y values of the robot on a coordinate axis system. The orientation is the direction of the robot in quaternions. In order to understand when the robot was turning or where it was facing, we had to convert the quaternions to Euler angles—it is then easier to understand and detect the change in orientation. Using *tf.transformations.euler_from_quaternions*, we converted quaternions to Euler angles. From the three Euler angles, we could safely assume that the robot would not roll or pitch, so the only angle we needed to look at was yaw.

To understand if the robot was turning right or left, we had to look at the yaw between each pose in the plan, which would be too much data to process for each global plan. To avoid this problem, we looked at the the sum of the difference of yaw among 10 values of the global plan; this gave us a general idea of the robot's direction. Based on the number being negative or positive, we were able to show the corresponding sequence on the BlinkSticks.

3. Function Documentation

Listed below are functions that can be found in **blinkstick_signal_patterns.py** which are not included in the BlinkStick Python API. These functions were written over the course of the project to better understand the BlinkSticks. This documentation serves to guide future BlinkStick users.

In order to run the BlinkStick code in the listener, we had to import the blinkstick module from the blinkstick package. This can be done as follows:

```
from blinkstick import blinkstick
```

Most of the functions listed below ask for a color name to be passed as a parameter. In order for the functions to execute properly, the color has to be recognized by the cascading style sheets (CSS) language. These colors can be found in the following dictionary:

```
matplotlib.colors.cnames
```

blink_at_index(bs_obj,color_name,index)

This function blinks a specified color on any given BlinkStick on a selected index.

Parameter Description:

- bs_obj: reference to a BlinkStick object
- color_name: string color name
- index: integer index with range [0,7] of the LED to blink

Example Usage:

```
bsticks = blinkstick.find_all()
bs_obj = bstick[0]
blink_at_index(bs_obj, "red", 5)
```

blink_up_ stick(bs_obj,color_name,sec)

With a specified color name, BlinkStick, and seconds, this function blinks the color from index 0 to index 7 at the specified rate.

Parameter Description:

- bs_obj: reference to a BlinkStick object
- color_name: string color name
- sec: time in milliseconds between blinks

Example Usage:

```
bsticks = blinkstick.find_all()
bs_obj = bstick[0]
blink_up_stick(bs_obj, "red", 500)
```

set_all_indices(bs_obj,color_name)

Given a color and a BlinkStick, this function sets the color on all the indices of the blink stick. This function can be used to turn off all the lights by passing “off” as the color_name parameter.

Parameter Description:

- bs_obj: reference to a BlinkStick object
- color_name: string color name

Example Usage:

```
bsticks = blinkstick.find_all()
bs_obj = bstick[0]
set_all_indices(bs_obj, "red")
```

blink_random(bs_obj)

This function blinks random colors on any given BlinkStick in a random order.

Parameter Description:

- bs_obj: reference to a BlinkStick object

Example Usage:

```
bsticks = blinkstick.find_all()
bs_obj = bstick[0]
blink_random(bs_obj)
```

blink_rainbow_any_order(bs_obj,indlist)

With a specified BlinkStick object and a list of numbers, this function blinks the rainbow colors on the BlinkStick in that order.

Parameter Description:

- bs_obj: reference to a BlinkStick object
- indlist: list of eight integers to index with range [0,7] of the LED to blink

Example Usage:

```
bsticks = blinkstick.find_all()
bs_obj = bstick[0]
blink_rainbow_any_order(bs_obj, [0,1,2,3,4,5,6,7])
```

blink_alternating_rainbow(bs_obj,bs_obj2)

Given a color and two BlinkStick devices, this function blinks that color from index 0 to index 7 while alternating on the two given BlinkSticks.

Parameter Description:

- bs_obj: reference to a BlinkStick object
- bs_obj2: reference to a second BlinkStick object

Example Usage:

```
bsticks = blinkstick.find_all()
bs_obj = bstick[0]
bs_obj2 = bstick[1]
blink_alternating_rainbow (bs_obj, bs_obj2)
```

blink_using_rgb(bs_obj,r,g,b,index)

With any given color specified by RGB (red, blue, green), this function blinks that color on any given index on a given BlinkStick.

Parameter Description:

- bs_obj: reference to a BlinkStick object
- r: set the Red value
- g: set the Green value
- b: set the Blue value
- index: integer index with range [0,7] of the LED to blink

Example Usage:

```
bsticks = blinkstick.find_all()
bs_obj = bstick[0]
blink_using_rgb (bs_obj, 255, 0, 0, 5)
```

4. Summary and Conclusion

Human–robot communication, particularly the movement of the robot, is still being researched. In order for humans to understand robots, we developed an easier way to indicate the direction of the robot's movement using visual interpretation. In this project, we programmed and implemented BlinkSticks to a robot. By doing this we were able to create easily understandable visual signals to insinuate the movement and, specifically, the direction of the robot. In the future, the BlinkSticks and this approach could be used to display other types of data in addition to information about the movement of the robot.

5. References

1. Blinkstick strip. [date unknown; accessed 2018 Aug 10]. https://www.blinkstick.com/products/blinkstick-strip.

2. Twigg JN, Gregory JM, Fink JR. Towards online characterization of autonomously navigating robots in unstructured environments. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS); p. 1198–1205.

3. Blinkstick module for python. [date unknown; accessed 2018 Aug 10]. https://github.com/arvydas/blinkstick-python.

4. Berwick R, Juskevicius A. Blinkstick-python api documentation. [date unknown; revised 2014 Oct 13; accessed 2018 Aug 10]. https://arvydas.github.io/blinkstick-python/.

List of Symbols, Abbreviations, and Acronyms

API - application programming interface

ARL - Army Research Laboratory

CSS - cascading style sheets

LED - light-emitting diode

ROS - Robotic Operating System

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIR ARL IMAL HRA RECORDS MGMT RDRL DCL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
10 (PDF)	US ARMY RESEARCH LAB RDRL CII A M WIGNESS J TWIGG J ROGERS J FINK D BARAN E STUMP C REARDON G WARNELL J GREGORY N FUNG
1 (PDF)	MAHARISHI SCHOOL Y DERIBE TAFESSE