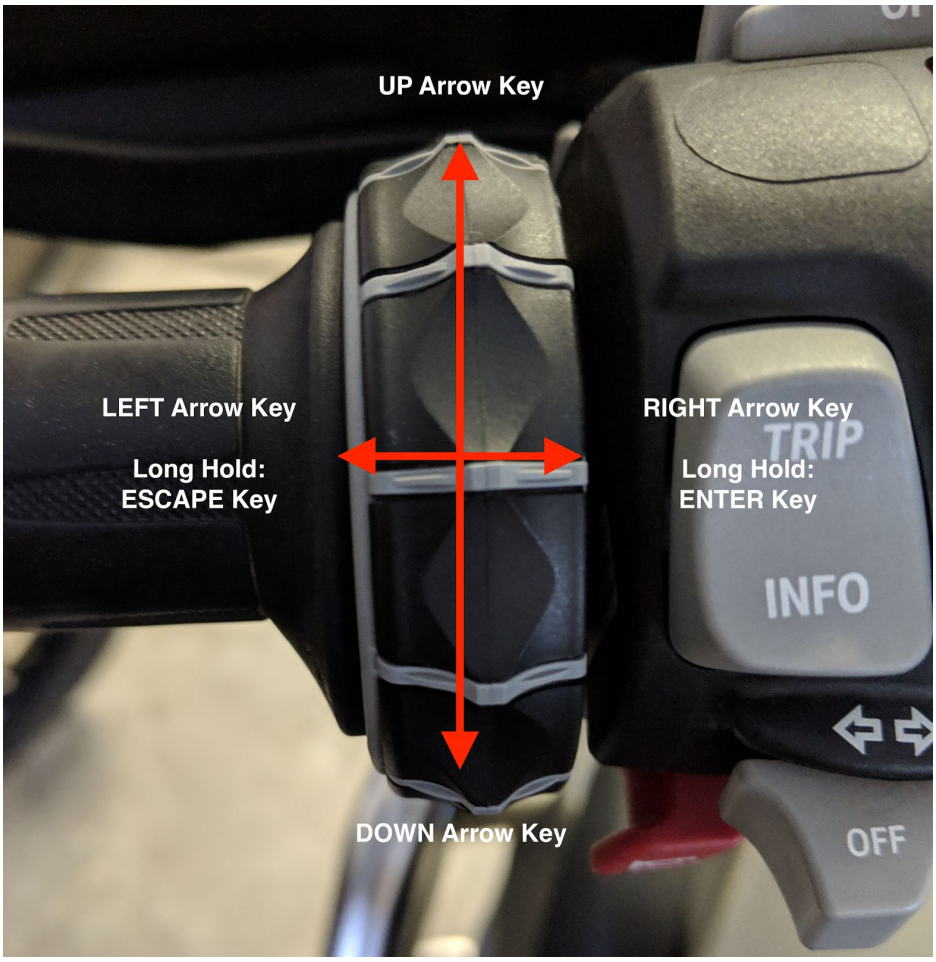
WunderLINQ 3rd Party Integration

The purpose of this document is to outline the steps to support the WunderLINQ hardware and App in your App. We will break those down separately and use a Navigation App as an example.

WunderLINQ Hardware Support

One of the great things about the WunderLINQ is that it's not proprietary and doesn't require any special SDKs or APIs. The WunderLINQ acts as a regular keyboard and by default sends the following key codes: UP, DOWN, LEFT, RIGHT, ENTER and ESCAPE. To use these keys you would use Android or iOS native keyboard handling. Some Android and iOS code snippets are provided at the end of the document. If you don't have a WunderLINQ compatible bike you can test with any Bluetooth keyboard that includes the UP, DOWN, LEFT, RIGHT, ENTER and ESCAPE keys. Below is an annotated picture to show the mapping to the physical controls.



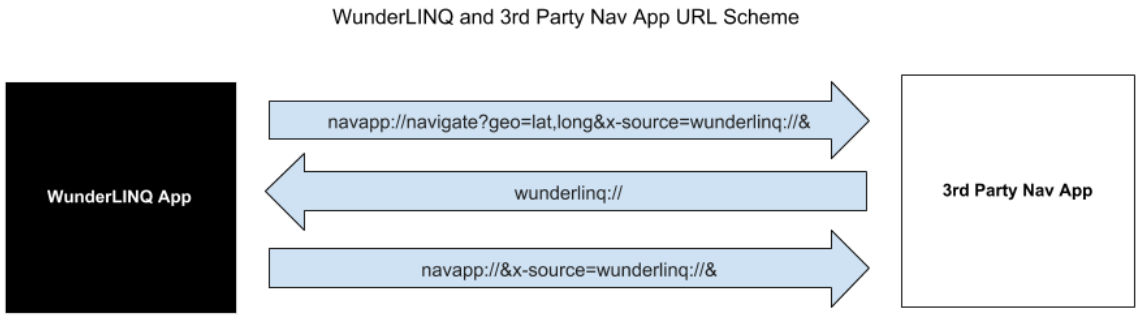
Navigation App Key Mapping Ideas:

UP/DOWN	Map zoom in/out
UP/DOWN/LEFT/RIGHT	Map scrolling
UP/DOWN/LEFT/RIGHT/ENTER	Menu navigation and selection
ESCAPE	Back to the WunderLINQ App

WunderLINQ App Support

For WunderLINQ App support we propose a solution that again is not specific to the WunderLINQ and could be reusable for other integrations. The basic idea is to provide a URL scheme that other Apps like the WunderLINQ App can use to open your App and provide a return URL. You could follow the standard URL scheme x-callback-url or it could be as simple as adding an argument to your existing URL scheme to accept a return URL. The return URL can then be used to provide a mechanism to go back to the App that launched your App. A button or the ESCAPE WunderLINQ hardware button in your App for example.

For general information on handling URLs in your app please review the Android or iOS SDK docs.



The above illustrates through URL schemes a scenario where a user selects a task in the WunderLINQ App to start Navigation to a location. Later the user touches a button or hits the WunderLINQ hardware ESCAPE button, to go back to the WunderLINQ App so they can check motorcycle performance data. Later in the WunderLINQ App the user selects a task to resume navigation in the Nav App

Navigation App URL Scheme Requirements

To provide full WunderLINQ Navigation App integration we need the following abilities via URLs

1) Open Navigation App without interrupting current routing or open in driving mode

Ie. navapp://&x-source=wunderlinq://&

2) Open Navigation App and view the waypoint

Ie. navapp://view?geo=lat,long&x-source=wunderlinq://&

3) Open Navigation App and navigate to waypoint

Ie. navapp://navigate?geo=lat,long&x-source=wunderlinq://&

Return URL Argument Usage

The return URL could be used to launch the calling app from a button and/or intercepted key press. For a WunderLINQ integration, we would like the ESCAPE keypress to bring you back to the WunderLINQ App.

Help

Please contact me at keith.conger@blackboxembedded.com if you have any questions or need any help. If you do integrate please let us know so we can add your App to the WunderLINQ App and Website.

Additional References

Android

Keyboard Handling
https://developer.android.com/training/keyboard-input/commands
URI Handling
https://developer.android.com/training/app-links/deep-linking

iOS

Keyboard Handling
https://developer.apple.com/documentation/uikit/uikeycommand
URI Handling
https://developer.apple.com/documentation/uikit/core_app/allowing_apps_and_websites_to_link_to_your_content/defining_a_custom_url_scheme_for_your_app

General

X-callback-url
http://x-callback-url.com/

Android Code Snippets

Intercept Key Presses

```
@Override
public boolean onKeyUp(int keyCode, KeyEvent event) {
	Log.d("Keycode", "Keycode: " + keyCode);
	switch (keyCode) {
		case KeyEvent.KEYCODE_DPAD_LEFT:
			//Do Something
			return true;
		case KeyEvent.KEYCODE_DPAD_RIGHT:
			//Do Something
			return true;
		case KeyEvent.KEYCODE_DPAD_UP:
			//Do Something
			return true;
		case KeyEvent.KEYCODE_DPAD_DOWN:
			//Do Something
			return true;
		case KeyEvent.KEYCODE_ESCAPE:
			// You'll probably want to parse the argument with the return URL
			String callingApp = "wunderlinq://datagrid";
			Intent intent = new
Intent(android.content.Intent.ACTION_VIEW);
			intent.setData(Uri.parse(callingApp));
			intent.setFlags(FLAG_ACTIVITY_LAUNCH_ADJACENT);
			startActivity(intent);
			return true;
		default:
			return super.onKeyUp(keyCode, event);
	}
}
```

Override Back Button

```
@Override
public void onBackPressed()
{
	// You'll probably want to parse the argument with the return URL
	String callingApp = "wunderlinq://";
	Intent intent = new	Intent(android.content.Intent.ACTION_VIEW);
	intent.setData(Uri.parse(callingApp));
```

```
        intent.setFlags(FLAG_ACTIVITY_LAUNCH_ADJACENT);
        startActivity(intent);
}
```

iOS Code Snippets

Intercept Key Presses

```
override var keyCommands: [UIKeyCommand]? {
        let commands = [
                UIKeyCommand(input: "\u{d}", modifierFlags:[], action:
#selector(enterPress), discoverabilityTitle: ""),
                UIKeyCommand(input: UIKeyInputEscape, modifierFlags:[], action:
#selector(escapePress), discoverabilityTitle: ""),
                UIKeyCommand(input: UIKeyInputLeftArrow, modifierFlags:[], action:
#selector(leftPress), discoverabilityTitle: ""),
                UIKeyCommand(input: UIKeyInputRightArrow, modifierFlags:[], action:
#selector(rightPress), discoverabilityTitle: ""),
                UIKeyCommand(input: UIKeyInputUpArrow, modifierFlags:[], action:
#selector(upPress), discoverabilityTitle: ""),
                UIKeyCommand(input: UIKeyInputDownArrow, modifierFlags:[], action:
#selector(downPress), discoverabilityTitle: "")
                ]
        return commands
}

@objc func escapePress() {
        if let returnURL = URL(string: "wunderlinq://") {
                if (UIApplication.shared.canOpenURL(returnURL)) {
                        if #available(iOS 10, *) {
                                UIApplication.shared.open(returnURL, options: [:],
completionHandler: nil)
                        } else {
                                UIApplication.shared.openURL(returnURL as URL)
                        }
                }
        }
}

@objc func enterPress() {
        //Do Something
}

@objc func leftPress() {
```

```
        //Do Something
}

@objc func rightPress() {
        //Do Something
}

@objc func upPress() {
        //Do Something
}

@objc func downPress() {
        //Do Something
}
```