Travel Buddy: ITMT 492 Embedded Systems Project

Douglas Wilhelm
Illinois Institute of Technology

Abstract

Travel Buddy is a service-providing, travel data platform designed for “anyone on two wheels”, with a particular focus on motorcycle use. The system consists of an Arduino to Android interface which relays sensor data from either device to the user / rider through an Android application. The Android application will communicate with the Arduino microcomputer on the back end, and will provide users with easy-to-read data statistics in the form of a graphical user interface on the front end of the application. This smart computer setup will be universal, having a variety of applications and sensor possibilities as outlined below in this documentation.

Travel Buddy: ITMT 492 Embedded Systems Project

Introduction

The problem with two-wheeled travel is a particular lack of information regarding both the trip and the vehicle. Modern passenger vehicles are sold with a plethora of gauges and information-relaying devices which work in conjunction with the vehicle's onboard computer, also known as the Electronic Control Module (ECM) (1 - howstuffworks). This data provided to the end user both assists the driver during vehicle operation, and also provides more control over the vehicle and overall trip. In contrast, bicycles, scooters, mopeds, motorcycles, and all other two-wheeled vehicular machines are sold with either bare minimum gauges / data relay devices, or they are sold with devices which are low quality, limited in functionality, or expensive to replace. Travel Buddy aims to fill this opportunity space by providing riders of two-wheel machines a simple smartphone application to deliver a steady data feed while they are riding, control over the electronic components on their vehicle, and other various information relating to their travel.

Product

Travel Buddy consists of two primary elements; hardware devices and software control. For my setup, I chose to use an Arduino Mint IOIO for the hardware side. This microcomputer has a very small form factor, comes with built-in bluetooth functionality, and includes a rechargeable battery to power itself. The IOIO (pronounced "yo-yo") runs code delivered from a smartphone, specifically an Android in my setup, so there is no need to program two devices (2 - Github). Since the IOIO hardware device is an Arduino, it can utilize other Arduino sensors and

accessories on the market, so it makes a great sensor-reading device for its quality and levels of support available. In conjunction with the IOIO, the Android smartphone software control device communicates with the IOIO's firmware directly using libraries provided online. For the user interface, I completed a wireframe app and a sample app. Both applications prove that my vision of a simple, tile-based interface for the user / rider is feasible and easy to use. On the home screen of the app are different "tiles", wherein each tile represents a different sensor. Each tile can expand to take up the screen when tapped, in case the user / rider wants more information from one of the tiles. Upon tapping again, a user can minimize the tile, viewing the home screen once more. Interestingly, the Android can act as both a software and hardware device since its own sensors can be used in the Travel Buddy application. The possibilities of the Travel Buddy platform are endless with what can be detected by Arduino sensors, controlled by an internet-enabled Android smartphone. Such possibilities could only be expanded upon by having a "tile store" with in-app purchasing, where users could purchase new tile modules for different sensors.

For my in-class presentation, I implemented a demo Android program showing how the tile interface would work with an alarm system on my motorcycle (one example of a sensor device working with the IOIO / Android platform). The program has a tile dedicated to the alarm setup (Figure 1), with two settings made via a button: Armed and Disarmed. When disarmed, the tile does nothing. However, when the tile is set to armed, the Android looks for a significant change to the device's accelerometer in any direction, which would detect movement during a supposed vehicle theft. When this significant change to the accelerometer is detected, a HIGH

signal is through one of the IOIO's pins to a relay module, which is wired to the motorcycle's electrical harness (Figure 2). The relay powers on and connects power through the bike's 12v battery to the bike's horn, which would honk at a specified interval. Basically, when the alarm system is armed and the bike is being stolen, the horn will honk, scaring away potential thieves.

Observations

During the implementation of my motorcycle alarm, I made several observations. Firstly, due to the nature of the IOIO running code from the Android device, the Android will always need connection with the IOIO, meaning the phone must be present on the bike while it is parked somewhere. I recognize this is not ideal, so future implementation of the alarm specifically will need perhaps a secondary computer to run code on the IOIO while the phone's connection is not present. This is an advantage as well, though, because if the bike was stolen with the phone present, GPS data can be sent over a cellular network to the owner directly, allowing tracking of the stolen vehicle without having to pay extra for a dedicated GPS / cellular device combo (3 - Gizmag).

Secondly, the IOIO has a limited battery, so letting it run from its own power source while riding is not ideal. A finished product will allow the Travel Buddy IOIO computer to run from power generated from the motorbike, using the onboard battery only as a backup. Another side to this observation is that in order for Travel Buddy to operate seamlessly with a motorbike, modification to the stock electrical harness will be necessary, which might void a factory warranty. Though, this may not be relevant since Travel Buddy is intended for older bikes, which

are typically out of warranty.

My final observation during implementing the motorcycle alarm example was the extreme difficulty of getting the Arduino IOIO to talk to the Android over bluetooth, Many library files I attempted to use were intended for a newer firmware version of the IOIO. My IOIO had an older firmware, which does not work with newer libraries. In order to upgrade a Mint IOIO (different from sister IOIO, the OTG unit), one must have a secondary Mint IOIO to act as the programmer. Nothing else can be used to complete this action, including Android or a PC, and this was very frustrating. Additionally, the library files provided by IOIO manufacturers are intended to work with Eclipse, an integrated development environment which is not outdated by Google's Android Studio. Converting the provided library / project files to Android Studio was no easy task (4 - Google Groups).

Conclusion

Conclusively, Travel Buddy as a sensor-relay framework platform has many possibilities for expansion and great usefulness for two-wheeled riders alike. Once a solid connection has been established between the Arduino IOIO and the Android smartphone, the wiring and circuit integration is easy, allowing primary focus on development of the software user interface which will undeniably enhance user experience. Travel Buddy can be adapted for many different uses, it's just a matter of implementation.

Appendices

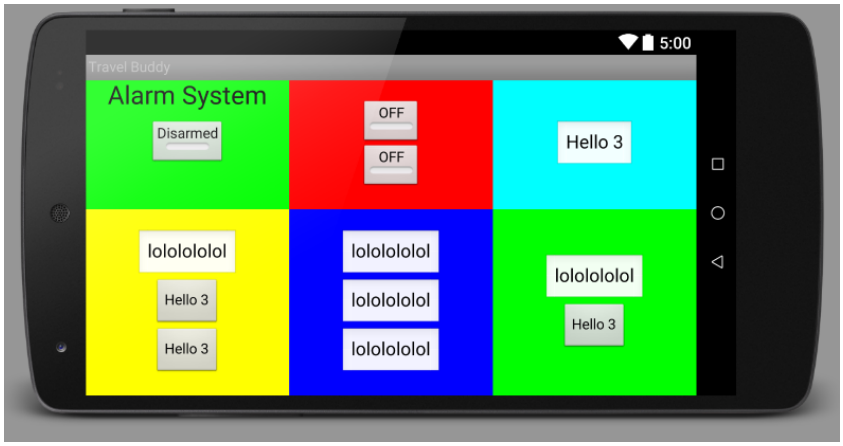

Figure 1 - Android Tile Interface

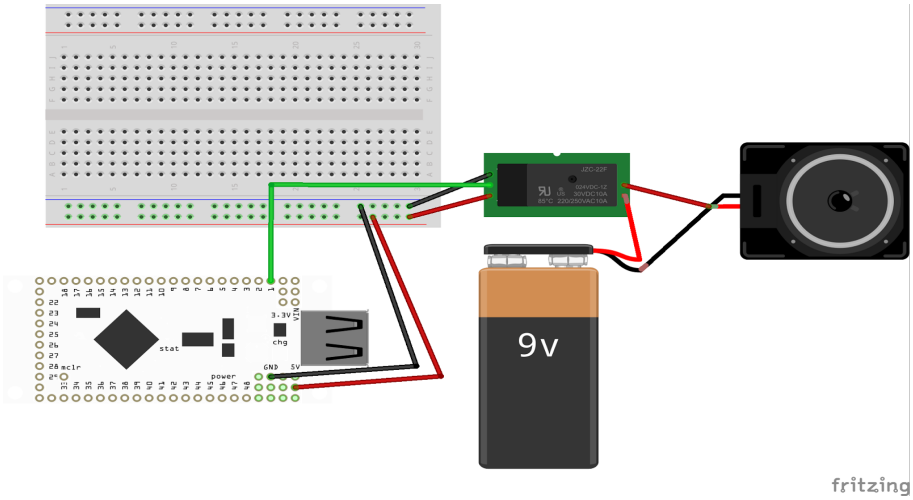

Figure 2 - Motorcycle Theft Alarm Circuit

References

How the Engine Control Module Works. howstuffworks. Retrieved from http://auto.howstuffworks.com/engine-control-module.htm, 5-2-2015

IOIOLib Basics. Github. Retrieved from https://github.com/ytai/ioio/wiki/IOIOLib-Basics, 5-2-2015

SpyBike GPS Tracker is like LoJack for Bikes. Gizmag. Retrieved from http://www.gizmag.com/spybike-gps-tracker/22999/, 5-2-2015

Using IOIO with Android Studio. Google Groups. Retrieved from https://groups.google.com/forum/#!topic/ioio-users/EasTREiJf0Y, 5-2-15