Bike.ai - Adaptive Smart Bike

Berkeley | EECS
Electrical Engineering and Computer Sciences
EECS 149/249A - 2019

Bernard Chen, Arjun Mishra, Michael Duong

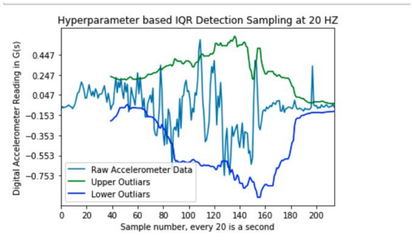
Objectives

- Prevent bike accidents due to sudden braking
- Alert bikers to moving objects in their blind spots
- Allow bikers to more easily signal turning
- Provide bikers access and customization of their bike

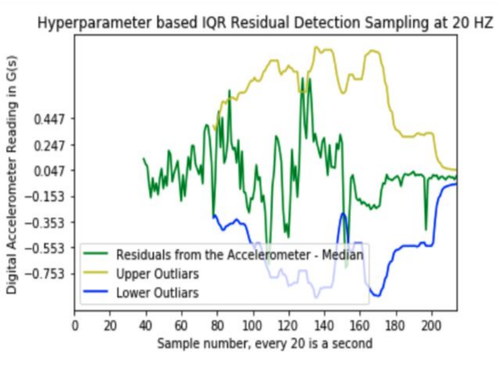
The diagram is our bike with the main sensors we are using for each subsystem. The BLE buttons are a non-intrusive way to signal turns, the accelerometer is used to detect braking without physical hardware, and the ultrasonic ranger is used to check “blind spots.”

Automatically Detecting Braking

We propose the following methods for online outlier detection. We propose IQR based outlier detection, IQR-residual based outlier detection, and variance based detection.



Plot of IQR based outlier detection, which utilizes 25% percentile -alpha(hyperparameter) *IQR and 75% percentile + alpha*IQR

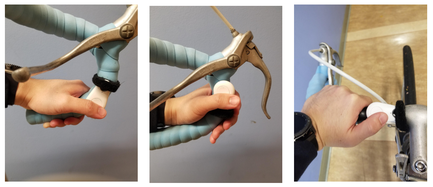


Very similar to the algorithm listed above but is instead based off of the residuals as opposed to the absolute values of the acceleration readings

** A few brake test-rides were conducted and analyzed within Jupyter notebook for the most reliable algorithm to detect braking. Test data came from 10 second to minute long rides both on flat surfaces and downhill slopes.

BLE-based Turn Signalling

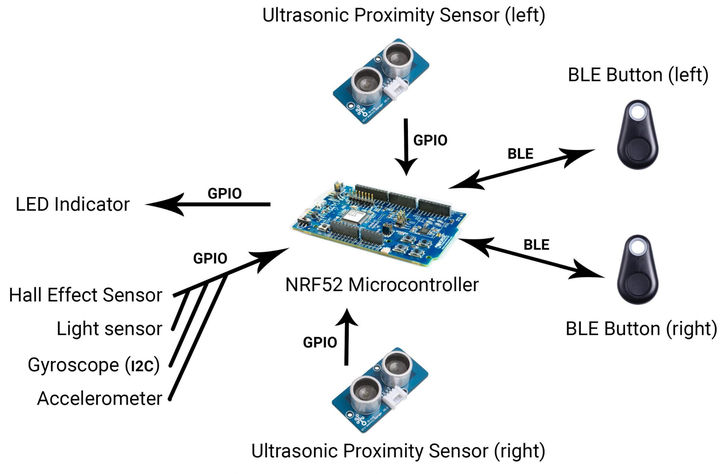
Uses nRF SDK to scan, connect, discover and communicate with wireless buttons. Additional features include auto-cancel after a successful turn and timeout.



All BLE code written inspired by examples from nRF SDK. Using scanning module and DB Discovery as well as calls to the SoftDevice API.

We use third party iTAG BLE buttons as our peripherals. This allows for smaller form factor but sacrifices customizability for peripheral firmware. A few options we tried:
Flic Buttons: Custom firmware on the button made it impossible to connect and interact.
nRF52 board as a button: Bulkier but customizable firmware from peripheral side.
iTAG button: Clean form factor, lacking notification CCCD, but able to interact with read/write

State Machines and Architecture Diagram



Each sub-component of our system is modeled by a state machine, and we combine them into one hierarchical state machine.

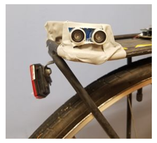
Braking: Input from accelerometer and output to LED. Switches states between LED off and LED on.

Turn signals: Input from BLE buttons and output to LED. Keeps variable for time. Switches between LED on and will transition to off when turn detected or timeout occurs.

Proximity: Input from ultrasonic sensor and output to LED. Keeps variable for time. The state machine goes through the process of switching to output, sending a signal, switching to input to wait for the echo, and calculating the distance.

Automatic Proximity Sensing

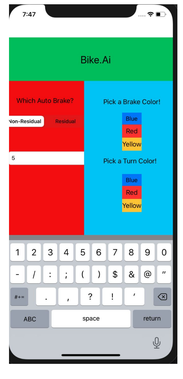
Using two ultrasonic rangers, we can detect how far away objects are in a rider’s blind spots. If the object is within a configurable threshold distance, a warning LED will turn on.

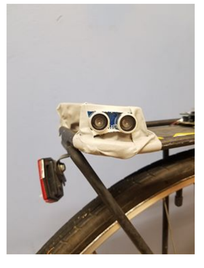


There is one ultrasonic ranger per side that corresponds to an LED. If an object is detected on the left side, the left LED on the handlebar will light up (as shown in the figure), and the same for the right side. Although the sensor can detect up to 5 meters, we limit our software to only 2 meters because detecting objects at a farther distance will cause our main loop thread to interrupt for too long, and 2 meters is a sufficient distance to say something is “close.”

iPhone BLE interface for customization

In order to provide the rider the ability to customize their interaction with the smart bike, we provide togglable settings through an iPhone app.





The iPhone app interface to choose some options for the smart bike. The app communicates with the Buckler through BLE and can control settings such as LED colors, brake detection algorithm to use, and threshold distance for the proximity sensor.