David Darevsky, Joseph Liu, and Leland Yu

## Objective

According to the National Highway Traffic Safety Administration, 857 cyclists were killed in the United States in 2018 (NHTSA). Since most cycling fatalities occur as a result of car-on-cyclist collisions between the hours of 6-9pm, we set out to build a "smart" bike dashboard that would increase the safety of riding by (a) giving riders feedback on their speed and surrounding environment (temperature/humidity) as well as (b) implementing an automated turn/brake signal that would increase the rider's visibility and alert drivers to his or her movements (NHTSA). In keeping with the spirit of safety, our system was designed from the ground-up to require as little attention from the rider as possible: voice commands are used to actuate the turn/brake lights and to change the information presented on the dashboard displays; moreover, we provide redundancy in our system whereby the automatic decoding of a bike's kinematics are used to actuate the turn/brake lights in those cases where a rider forgets to explicitly signal turning/braking using their voice. Our work sets a robust framework for future "smart" bike development. For example, to make our current system more seamless for the rider, we envisage the use of augmented reality glasses (such as a Google Glass) to convey information to the rider without having the rider take their eyes off the road. Additional advances can involve the use of brain computer interfaces and machine learning to decode a rider's intention to turn/brake before the actions are physically initiated. And further refinement can be gained by developing an industry-wide standard for physically integrating our embedded system hardware into bicycles—thus making these safety features standard across the market.

## Overview of System Design

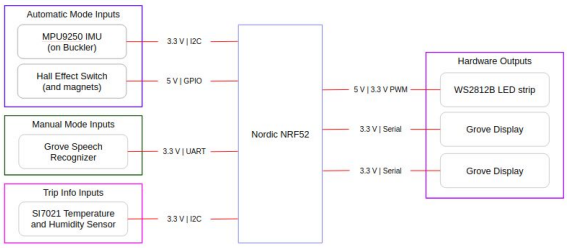


**Figure 1**: Holistic architectural diagram for the system. See https://tinyurl.com/soeqk6x for a larger version.

### *Hardware*

Our hardware architecture for the bike dashboard relies on eight hardware components: a Nordic nRF52832 ARM Cortex M4 SoC development board, a Grove Speech Recognizer, a WS2812B LED strip, two Grove 4-Digit Displays, a Hall effect sensor, an MPU9250 inertial measurement unit (IMU), and an SI7021 temperature and humidity sensor (see Figure 1 for architecture drawing). Note that the latter two sensors were included on the Berkeley Buckler—which also served as a central hub for all of the project's inputs/outputs. Our hardware components were mounted onto the rear pannier rack of a road bike and a breadboard was used as an intermediary hub between the peripherals and the Buckler (see Figures 2 for images of hardware components mounted onto bike).

**Figure 2**: Clockwise from left--picture of assembled bike with LEDs and Buckler on the rear; Grove 4-Digit Displays showing current speed (left) and Grove Speech Recognizer (right); Hall effect sensor next to magnets mounted on spokes of rear wheel.

### *Peripheral Drivers*

Integration of peripherals with our project required development of multiple drivers. For the Grove Speech Recognizer, we used Nordic's UART library to poll the voice sensor with each iteration through the main *while()* loop; importantly, this was implemented without blocking to prevent a delay in executing other code within our program. The WS2812B LED strip uses a self-clocking procedure where 1s and 0s are differentiated by the length the signal is held high; therefore, the library was implemented using a PWM signal to imitate this protocol. The driver for the Grove 4-Digit Displays was implemented using a custom-written bit-bang routine and allowed for the display of both numeric and letter characters. One display was used for numerical data while another was used for display of textual data (e.g. display of "15 MPH" was split across two displays with "15" shown on one and "MPH" shown on the other). The display library was

built to support both floating point, integers, and strings, even allowing for scrolling displays. The Hall effect sensor, which in essence is a magnet-activated switch, can be called through standard GPIO callbacks. It is triggered by nine magnets that were attached on the spokes of the rear wheel. For the MPU9250 IMU, we extensively modified the MPU9250 library developed by the *lab11* team at UC Berkeley. Changes were made to include affine sensor model calibration routines for the accelerometer, gyroscope, and magnetometer; new functions to read the three IMU data streams over $I^2C$; and significantly revised configuration register values (including revised scales for the gyroscope and accelerometer, enabling of a 41Hz lowpass filter, increased magnetometer sampling rate among other additions). The SI7021 temperature and humidity sensor was polled over $I^2C$ with each iteration of the main *while()* loop.

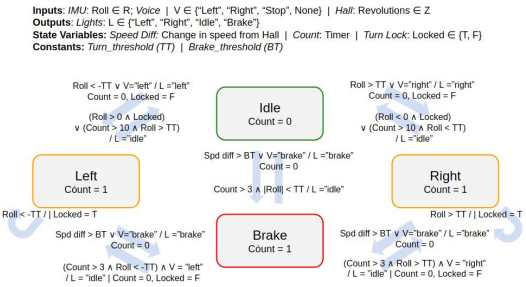


**Figure 3**: Software FSM. See https://tinyurl.com/s9ks2n8 for a larger version.

*Software*

The software component of our project revolves around the creation and tuning of an event-driven finite state machine (FSM; See **Figure 3**). The FSM takes as inputs the bike's calculated roll (determined using a sensor fusion algorithm; details below), the user's voice commands, and the bike wheel's revolutions over the past second. As outputs, the FSM actuates the four modes of the LED strip at the rear of the bike.

From the *IDLE* state the FSM may transition to one of *RIGHT*, *LEFT*, or *BRAKE* states depending on either voice command or bike kinematics input. If the voice command signals a turn, or if the bike's roll is greater than the *Turn_threshold* (*TT*; set to +/-14° roll in the right and left directions, respectively), then the FSM transitions to a *RIGHT* or *LEFT* turning state. For those cases where we transition to a turn state using voice commands, we implement a turn signal locking mechanism similar to that used on regular automobiles whereby the turn signal is "locked" into the respective turn state based on the bike's roll surpassing the *TT*; in other words, we transition back to *IDLE* state when a rider accidentally leaves a turn signal on for over 10 sec if a physical turn of the bike has not yet been initiated.

The *stop* signal comes from the detection of braking using either the bike's kinematics or from the rider's voice commands. To enter the *BRAKE* state using voice command, the rider simply needs to say "Stop." To enter the *BRAKE* state using kinematics we keep track of the bike's velocity using the Hall effect sensor and then compare the current speed with a weighted average of the bike's speed over the past second; if this difference is less than the *Brake_threshold* (*BT*; -0.6mph difference), then the braking state is entered on the next transition. In all cases, we remain in the *BRAKE* state for three seconds before allowing transitions to any other state; thus ensuring sufficient time for car drivers to see the cyclist braking.

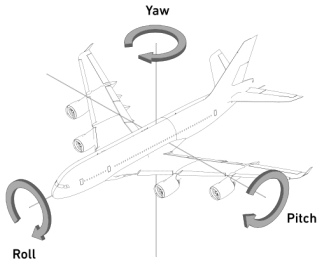


**Figure 4**: Pitch, yaw, and roll of an object. Source: Wikipedia.org

*Real-time Decoding of Bicycle Orientation*

While we easily implemented automatic brake detection by tracking the bike's velocity using a Hall effect sensor, unsupervised turn detection by tracking a bike's **roll** was more challenging (see **Figure 4** for description of an object's principal axes); while we could threshold roll on a single accelerometer axis, we decided against this approach for two reasons: (1) using acceleration alone leaves roll measurements prone to the influence of a rider accelerating or decelerating on the bike and (2) changes in acceleration on the remaining two axes of the accelerometer will impact acceleration on our axis of interest. In order to isolate the bike's roll independent of acceleration and motion on the other principal axes, we integrated Sebastian O.H. Madgwick's Attitude and Heading Reference System (AHRS) algorithm (Madgwick et al.). The Madgwick algorithm contains 89 lines of code and uses as input the accelerometer, gyroscope, and magnetometer

readings. The output produced is a quaternion describing the bike's orientation relative to a reference frame (in our case given by both the direction of gravity and the Earth's magnetic field). In brief, a quaternion is a four-dimensional complex number that describes the rotation of a coordinate frame around a vector relative to a predefined reference frame (see **Figure 5** for an example rotation). The Madgwick algorithm uses the IMU's accelerometer and magnetometer

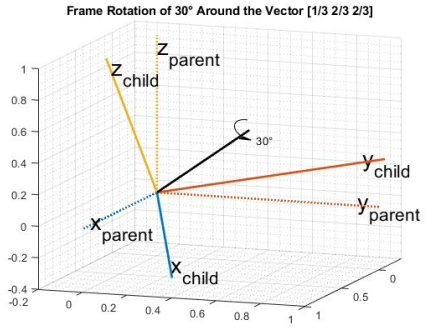


**Figure 5**: Example rotation of frame *child* relative to frame *parent* around a given vector. Source: https://www.mathworks.com/help/fusion/examples/rotations-orientation-and-quaternions.html

to infer the Earth's reference frame; the gyroscope is then integrated to yield the rotation of the IMU relative to Earth--with incorporation of additional corrections for gyroscope drift and magnetic field distortion (Madgwick et al.). The resulting quaternion can be multiplied by a rotation matrix to yield Euler angles which, when given the appropriate rotation matrix, yield the orientation with respect to the principal axes of pitch, yaw, and roll. Each iteration through the main *while()* loop allows the Madgwick algorithm to progressively refine of the quaternion; thus we smooth and average the most recent 300 roll measurements to infer when a rider is turning based on crossing of the *Turn_threshold*.

**Relationship to Course Contents**

Our bike dashboard incorporates many of the concepts taught within our class including affine sensor models; FSMs; interrupts, timers, and polling; peripheral communication protocols; and linear temporal logic.

*Affine Sensor Models for Sensor Calibration*

Since we utilized the Buckler's MPU9250 IMU as the input to Madgwick's AHRS algorithm, calibration of the IMU's accelerometer, gyroscope, and magnetometer was critical to achieving the best performance. Since the IMU's datasheet yielded the scale values for each measure, we instead focused on calculating the biases for each sensor, drawing inspiration from Kris Winer (https://github.com/kriswiner/MPU9250). We averaged 400 samples of accelerometer and gyroscope data while the Buckler was stationary on a flat surface. Gyroscope biases were then adjusted to minimize drift on all axes; accelerometer biases were adjusted to set the X and Y axes to zero while the Z-axis was set to +1g. The magnetometer was calibrated according to a routine again developed by Kris Winer (https://github.com/kriswiner/MPU6050/wiki/Simple-and-Effective-Magnetometer-Calibration). Briefly, the X-, Y-, and Z-axis magnetometer readings were zero-centered by first subtracting the average magnetic field strength measured for each axis. These were then normalized by a scale factor uniquely calculated for each axis such that the response to magnetic field flux was equivalent across the axes. Accelerometer and gyroscope calibration routines are run with system start; the magnetometer calibration was ran once and the values saved for future use.

*Event Driven Finite State Machine*

The core logic of our system is captured through an event driven FSM in the main *while()* loop. Complete details of the FSM are outlined above in the **Overview of System Design** section and visualized in **Figure 3**.

*Interrupts, Timers, and Polling*

Our system relies heavily on interrupts, timers, and polling. The Hall effect sensor was connected to a GPIO pin on the Buckler; triggering the sensor by magnets mounted onto the bike's wheel called an interrupt service routine (ISR) that increments a counter tracking the number of magnet passages. The IMU was also read on an interrupt that was triggered when new data was ready on the sensor. Our project included one timer to convert Hall effect sensor readings into velocity every 250 msec; we also used this timer to access RTC clock ticks as a means of tracking time (in milliseconds) within our main *while()* loop (necessary for proper gyroscope integration in the Madgwick algorithm). Lastly, the Grove Speech Recognizer was polled with each iteration of the main *while()* loop using a non-blocking read over UART. Great care was taken to minimize sharing of global variables between ISRs

and *main*; when sharing must occur, we momentarily disable IRQs to allow for copying of global variables to the local scope thus avoiding non-determinism. ISRs were kept as short as possible to maximize time spent running the Madgwick algorithm and the FSM in the main *while()* loop.

*Communication Protocols*

Our project incorporates multiple peripheral device communication protocols introduced in class and in lab including: $I^2C$ (MPU9250 IMU, SI7021 temperature and humidity sensor), UART (Grove Speech Recognizer), and PWM (WS2812B LED strip). As mentioned before, the LED strip does not use PWM explicitly; however, the protocol it does use can be modeled well with PWM duty cycles.

*Linear Temporal Logic*

A key safety property of our system was that the receipt of a *stop* signal (in the form of either a "Stop" voice command or automatic sensing of bike deceleration using Hall effect sensor data) would trigger a transition to the *BRAKE* state of our FSM from any other state irrespective of other inputs (such as whether the rider is turning). We formalized this specification with the LTL formula **G(*stop* ⇒ X *BRAKE*)**. Critically, we check for the presence of a *stop* signal in the first conditional statement within each state of our FSM; should the signal be present, we immediately transition to state *BRAKE* on the next iteration through the FSM in the main *while()* loop.

**Implemented System Evaluation & Results**

*Sensors*

We found the Hall effect sensor to be highly reliable; detecting nearly every crossing of the magnets passing it; therefore, brake detection was also highly reliable when using its data as input. The Grove Speech Recognizer proved much less reliable. While we had good performance when bench testing the Recognizer at home, in lab, or in the hallways of Cory, the sensor's performance fell precipitously in environments with high ambient noise; the Recognizer also occasionally interpreted the "Stop" command as "Up" (see **Table 1** for estimated performance of the Speech Recognizer in different environments). The MPU9250 IMU had acceptable performance for our application. The magnetometer and accelerometer were very robust to motion artifact; the gyroscope, on the other hand, had significant high frequency noise that was difficult to attenuate with the on-chip low pass filter (all low pass filters <41Hz decreased performance of the Madgwick algorithm). Our solution was to take a rolling average of the 300 most recent roll values from the Madgwick algorithm to control for gyroscope noise in our measurements (see **Figure 6** for comparison of unsmoothed and smoothed roll)

*Actuators*

The two Grove 4-Digit Displays proved highly reliable in all environments. The LED strip worked well during bench testing and on-the-bike testing in Cory Hall; however, once we brought the bike outdoors for demo day, the LED strip began exhibiting undefined behavior after several minutes in the cold. When we brought the bike back inside, the LED strip resumed functioning as expected. We suspect the cold outdoor temperature (around 9°C/48°F) decreased the nRF's clock oscillation frequency thus leading to inaccurate performance of the PWM protocol used to control the LED strip.

*Algorithms and Software*

Qualitative analysis of the Madgwick algortihm's output on the yaw and pitch principal axes demonstrated expected behavior save for the same high frequency noise also experienced on the roll axis. Printing the execution time of the main *while()* loop to RTT demonstrated an execution time that fluctuated between $0 < t \leq 1$ millisecond; thus the Madgwick algorithm iterates at least 1,000 times per second. We did not experience undefined system behavior during testing that may be attributed to interrupts or timers and the non-determinism associated with concurrency.

*Example Videos of System Behavior*:

Braking: https://tinyurl.com/rukylf4
Turn Detection: https://tinyurl.com/vwdmenf
Voice Control: https://tinyurl.com/qk5eupu

| Detection Performance (estimated % success rate) | | |
|---|---|---|
| | Turn | Brake |
| Kinematics | 80% | 90% |
| Voice (quiet environment) | 80% | 80% |

| | | |
|---|---|---|
| Voice (noisy environment) | 30% | 30% |

**Table 1**: Estimated turn/braking detection as a function of environment and voice sensor or kinematics input.

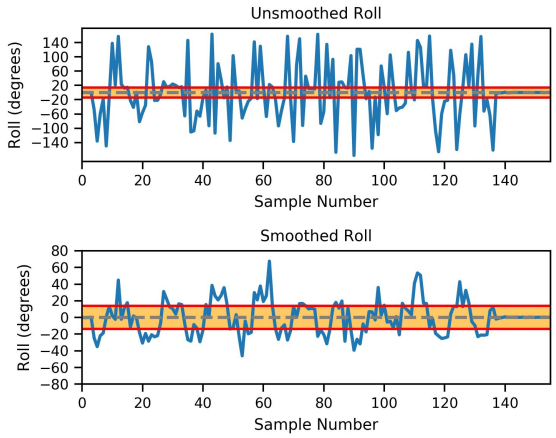


**Figure 6**: Plot of unsmoothed and smoothed roll data from the Madgwick algorithm when the Buckler is tilted +/-40° in bench testing. Red lines show the thresholds for turn detection (tuned for optimum performance on an actual bike ride).

### Analysis Of Why Things Work vs Not Work

Generally, the reliability of our project's various subsystems tended to covary with the reliability of peripherals. For instance, the Hall effect sensor provided highly reliable and accurate measurements of the bike's velocity/distance traveled. The Hall sensor detected nearly every magnet crossing and we had 9 equidistant magnets mounted onto the spokes of the bike's rear wheel giving us excellent spatiotemporal resolution.

In contrast to the Hall sensor, the Grove Speech Recognizer proved much less reliable—especially in noisy conditions. We had difficulty getting the sensor to respond both to its "Hicell" wake word and to voice commands while in the Cory courtyard during demo day. Unfortunately, due to the sensor's proprietary firmware, we had little recourse in tuning its performance. One possible solution to poor voice detection may involve implementing and training a small on-device neural network for speech recognition, as used by Apple for its "Hey Siri" wake word (https://machinelearning.apple.com/2017/10/01/hey-siri.html).

As previously mentioned, the MPU9250 IMU was adequate for accurate turn detection given sufficient smoothing of the resultant data from the Madgwick algorithm. Future generations of the Bike Dashboard can benefit from an IMU with a less noisy gyroscope allowing us to set lower thresholds for turn detection thus making the system more sensitive to turns. We could also attempt low pass filtering the output of the Madgwick algorithm using an embedded DSP library such as CMSIS-DSP. The WS2812B LED strip was also unreliable for our application. Despite working adequately in our test settings, we experienced undefined behavior in cold outdoor weather (most likely from the effects of temperature on the nRF's crystal oscillator affecting PWM frequency). Even if the nRF is to blame, the LEDs need to be robust to all sorts of weather a biker can face. Newer LED strips such as the DotStar from Adafruit (https://www.adafruit.com/product/2239) use SPI for communication thus making them less sensitive to timing. We could also weather-proof our setup to minimize temperature fluctuations. Lastly, though perhaps most importantly, we appreciate the high reliability of our FSM, peripheral drivers, and other software; we attribute this to a detail-oriented development process whereby all peripheral drivers were turned into libraries, debugging was done to mitigate library conflicts (e.g. by preventing two libraries from attempting to initialize the same TWI manager instance), and through careful attention to the risks posed by interrupts and timers on concurrent programs. In other words, the success of our project was dictated more by extrinsic factors (such as the quality and consistency of our sensors/actuators) rather than the code and system architecture that we implemented.

### Conclusion

For our final project, we successfully built a "smart" Bike Dashboard that improved the safety of cycling. Our system displays live trip information to the user through two displays and uses both a bike's orientation, and voice commands, to control an LED strip signalling turning/braking. We achieved robust performance for both voice and kinematics-based detection of turning/braking in testing environments with decreased performance of the voice commands and LED strip in outdoor environments due to noise and temperature, respectively. Future directions may involve acquiring a more accurate IMU, implementing an improved voice detection algorithm, weather-proofing the system, and implementing a newer generation LED strip

**References:**

"Bicycle Safety." NHTSA, 9 Sept. 2016, https://www.nhtsa.gov/road-safety/bicycle-safety.

Madgwick, Sebastian O. H., et al. "Estimation of IMU and MARG Orientation Using a Gradient Descent Algorithm." 2011 *IEEE International Conference on Rehabilitation Robotics*, 2011, pp. 1–7. *IEEE Xplore*, doi:10.1109/ICORR.2011.5975346.