AgriGuardian by kadd User Guide

May 30, 2020

Kaito Yoshida, Austin Wald, Davy Chuon, Daniel Weatrowski

Preface

Readme

This user guide is intended to outline the setup, usage, and avenues for future development related to the ATV Crash Detection and Ride Tracking system developed for UC Davis Biological and Agricultural Engineering department's Machine Systems Lab (UCD BAE MSL). This guide also aims to provide troubleshooting steps for issues discovered in testing. Due to the circumstances surrounding COVID-19, this project has received enough field testing to constitute a feature rich first prototype, but has not been tested in its specific use case (i.e. being mounted to a test vehicle). As a result, future development and fine tuning must be conducted for it to be as accurate of a utility as possible.

Audience

Since this project is intimately tied to a prototype device that has received testing exclusively outside of its intended use case, it is being explicitly created for the researchers at UCD BAE MSL. However, information pertaining to the usage of the iOS application can serve as a future guide for users outside of this intended audience.

Glossary of Terms

1. **XCode:** Xcode is an in, and tvOS.
2. tegrated development environment for macOS containing a suite of software development tools developed by Apple for developing software for macOS, iOS, iPadOS, watchOS**Swift:** Swift is a general-purpose, multi-paradigm, compiled programming language developed by Apple Inc. for iOS, iPadOS, macOS, watchOS, tvOS, Linux, and z/OS.
3. **Python:** Python is an interpreted, high-level, general-purpose programming language. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.
4. **Raspberry Pi:** Raspberry Pi is a small scale, single-board computer that enables portable computing practical to all types of applications world wide.
5. **IMU:** An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and magnetometers.
6. **BLE**: Bluetooth Low Energy
7. **Bleno**: Node.js module for implementing BLE peripherals

8. **RockBlock:** The RockBLOCK 9603 is an Iridium modem that enables the sending and receiving of short messages from anywhere on Earth via the global Iridium satellite network.
9. **GPS:** The Global Positioning System (GPS) is a satellite-based navigation system made up of at least 24 satellites. With a GPS receiver, GPS enables precise location tracking from anywhere in the world.
10. **Firestore:** Cloud Firestore is a flexible, scalable, realtime database for mobile, web, and server development from Firebase and Google Cloud Platform.
11. **Satellite (Communication):** A communications satellite is an artificial satellite that relays and amplifies radio telecommunications signals via a transponder; it creates a communication channel between a source transmitter and a receiver at different locations on Earth.
12. **UI/UX:** The UI (User Interface) of an application is the design of the application that users interact with such that the application can receive the user's input. The UX (User Experience) is the overall experience the user faces using the application.
13. **ATV**: An ATV (All-Terrain Vehicle) is typically a four-wheeled vehicle utilized by farmers to aid in everyday farm duties. ATV's have grown increasingly popular on the farm front due to their tractor-like abilities and inexpensive costs.
14. **First Responder:** Someone designated or trained to respond to an emergency, typically an ambulance, firefighters, etc.
15. **Riding Parameters:** The riding parameters are the different types of data that will be collected per ATV ride. These include: velocity, position, roll and pitch, vibration, terrain, slope.

Project Overview

Background

Agriculture is the backbone of the modern world, feeding both local and global communities and serving as an economic boon for developing countries in an ever-advancing, globalizing market. As with any economic staple, safe and reliable logistics are as important as the products themselves. Across the world and directly within the California Central Valley, small single-rider all-terrain vehicles (ATVs) are used to facilitate basic farm functions such as ploughing, harvesting, and applying pesticides in a cost effective manner. Their usage enables farms of all sizes to scale their operations quickly with low upfront cost. While ATVs perform many functions larger and more expensive tractors are readily capable of, they have severe drawbacks in terms of rider safety. Their short wheelbases, narrow track widths, and high centers of gravity have led to 26,800 ATV-related accidents in 2016 alone. Further compounding the issue is that the general use cases for ATVs on and off farms. The vehicles are nearly exclusively used in remote, rural locales with limited cellular service and hazardous terrain. This issue points candidly towards a need for a technical solution to improve first responder response times and increase rider safety by offering an extra layer of low-cost, easily integrated protection for an often overlooked aspect of the world's food system.

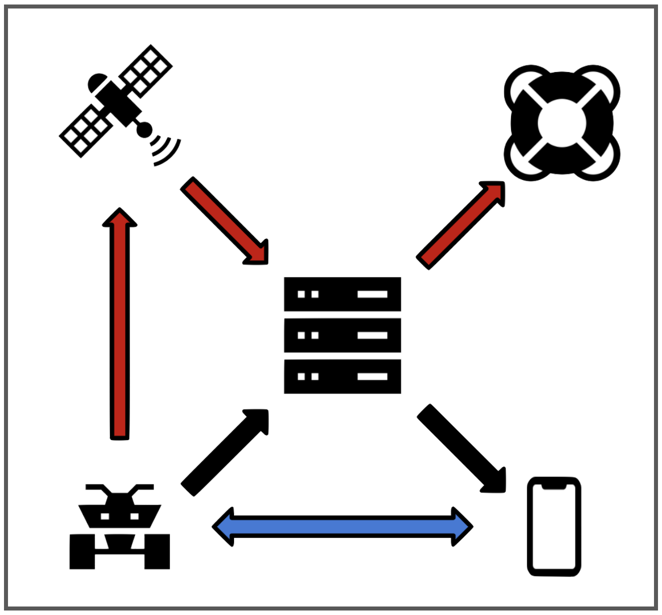
Description

UCD BAE MSL is seeking to implement an embedded system design mounted directly to ATVs that will notify first responders with location information in the event of a detected crash. Additionally, they seek to create a user-facing front end through a phone application that will allow users to track ATVs and view rider parameters such as velocity and trip history on demand.

Approach

This system comprises two components: an embedded platform that is dedicated to performing on-vehicle data collection, crash detection, and communication with emergency services, and a software iOS application for displaying collected data that is relative to its target audience. These two components are linked together by a cloud database that is run through Google's Cloud Firestore service.

System Architecture Overview



Red: Path of emergency messages (geofence, rollover)

1. Originate from device trigger
2. Sent via RockBLOCK to Iridium network
3. Relayed to Rock7 servers
4. Sent to Noonlight or vehicle owner

Blue: Bluetooth connection

1. Manual data pull

Black: Ride histories and statistics

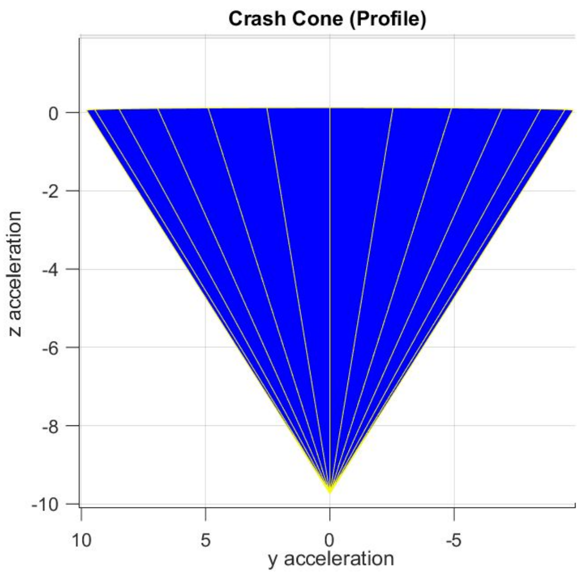
1. Files originate from device sensors stored in /data/rides/unsent
2. Sent to database once connection is established (checks every 10 seconds)

3. Read by app

Technical Specification

Embedded System - Raspberry Pi 4

Our hardware component records data from two sensors: an IMU (Adafruit LSM9DS1 and a GPS (Adafruit Ultimate GPS Breakout v3) at user specified rates approximately twenty seconds after the device is powered on, logging the output with the corresponding ride number in the filename in the data directory. All data is logged in a comma separated value (csv) format making it easily parsed by all common spreadsheet programs as well as our system when sending to the cloud database. Once the device is turned off the current ride is considered complete, and is queued for transmission to the database once the next time the device is powered on. Stead-state rollover detection is handled by a critical value cone and a weighted counter. The cone is fully parameterizable and is based on the possible acceleration of the device when the primary force acting upon it is gravity (in other words if the device is perfectly inverted, the z-axis receives the full 9.8 m/s^2). The cone is described by the equation $x^2+y^2 = [(m/(M-m))(z-m)k]^2$ (m = min z accel; M= max z accel; k = sensitivity coeff). Below, $m = -9.8$; $M = 0$; $k = 1$. This is the ideal case.



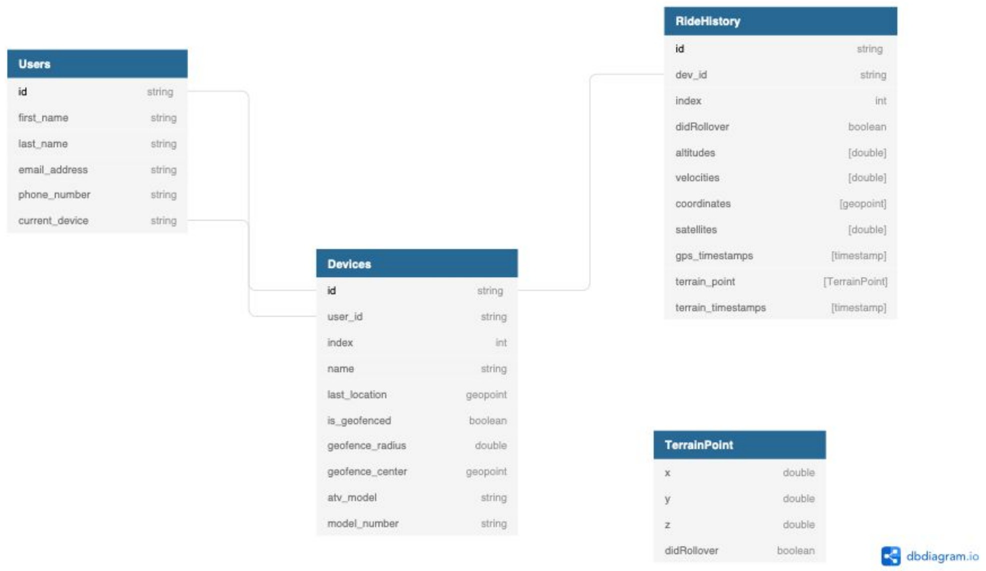
To deal with sensor error (±0.882m/s^2), the default values of $m = -11$; $M = -1$; $k = 2.5$ are recommended as a good starting point for fine tuning. If the IMU's accelerometer data falls within this cone it will increment the weighted counter, otherwise the counter is decremented. Once this reaches a user specified threshold, an alert is sent via the RockBlock satellite modem.

Application - iOS

Our application pulls data from the database and presents it to the user. The application allows users to view their previous ride history and query ride statistics such as GPS location, speed, acceleration, etc. Users will also be notified if any of their riders/vehicles are placed into an emergency situation. The application has been developed with Swift and Cocoa frameworks.

Database - Firestore

Google's Firestore will house all data generated by the Raspberry Pi on the ATVs. For the application and the Python script (for researcher), the database will be read only. Only the Python scripts running on the Pi will be able to write and save data to the database.



Features

1. Automated ride data collection and data transfer to the application.
2. Crash/Rollover detection.
3. Ability to summon first responders in the case of an emergency.
4. Detailed overview and presentation of ride parameter
5. Addition of multiple devices over bluetooth
6. Profile based application allows users to switch between their devices to view different data and ride parameters.
7. Manual data pull when no wifi is available on the device.
8. Automatic push notifications in the case of emergency or geofence boundary breaking.

Installation and Setup

Prerequisites

- Raspberry Pi 4 with Raspbian OS
  - Adafruit LSM9DS1 IMU
  - Adafruit Ultimate GPS Breakout v3
  - Rock7 RockBLOCK 9603
  - Mini HDMI Cable, Monitor, Mouse, and Keyboard
- iPhone with iOS version with iOS 10.1 and above
- Power source capable of supplying 12W

Setting Up the Raspberry Pi

1. Plug a mini HDMI cable into the Pi to connect it to a monitor and connect a mouse and keyboard
2. Power on the Pi by plugging in a USB-C cable connected to a power supply capable of producing 12W
3. Clone this repository to `/home/pi/`
    a. It is important to clone this repository to this directory since file paths are absolute due to issues running some threads before Raspbian has finished booting up.
4. Show hidden folders in the directory `/home/pi/` by opening the directory in the file explorer and right clicking on the window, this should give you the option to "show hidden folders"
5. Install Python 3 with `sudo apt-get update` and then `sudo apt-get install python3`
6. Install all the Python packages in `pip` using pip3
    a. `sudo apt-get install python3-pip`
    b. `pip3 install PackageNameFromPipFileHere`
7. Copy contents of `/home/pi/kadd-pi/setup/autostart` into `/home/pi/.config/lxsession/LXDE-pi`
    a. This will cause the terminal to launch at boot
8. Copy `/home/pi/kadd-pi/setup/.bashrc` and replace the existing `.bashrc` file in `/home/pi/` with it
    a. **NOTE:** Make a backup of the original `.bashrc` before overwriting it!
9. Restart the RaspberryPi while still connected to the monitor to see if any errors pop up after the main script boots.
10. If you see accelerometer data streaming across the terminal the device has been successfully configured.
    a. If you want to verify that the GPS is working, connect an external antenna and wait a few minutes; the status light should stop blinking and data should start streaming in the terminal when connected.
        i. Without an external antenna this connection may take longer to connect or may not connect at all if powered on inside of a room.
11. Now you can start configuring the device by editing the values in `/home/pi/kadd-pi/data/about.xml`
    a. `<mode>0</mode>`
        i. Activates Research Mode with 1, otherwise runs device in Farm Mode
            1. Research Mode activates imuSampRate and deactivates rollover emergency alerts
    b. `<gpsSampRate>15</gpsSampRate>`
        i. Sample rate for GPS in number of seconds between samples
        ii. Range: $1 \leq$ gpsSampRate $\leq$ sys.maxint
    c. `<imuSampRate>1</imuSampRate>`
        i. **Research Mode Only:** Sample rate for IMU in number of seconds between samples
        ii. Range: $0 \leq$ imuSampRate $\leq$ sys.maxint
    d. `<crashTimerThreshold>30</crashTimerThreshold>`
        i. **Farm Mode Only:** Number of consecutive rollover detections before an alert is sent

e. `<coneMinAccel>-11</coneMinAccel>`
   i. The z-axis coord of the critical value cone's point
   ii. Represents acceleration value in m/s^2
f. `<coneMaxAccel>-1</coneMaxAccel>`
   i. The z-axis coord of the critical value cone's base
   ii. Represents acceleration value in m/s^2
g. `<coneSensitivity>2.5</coneSensitivity>`
   i. The sensitivity coefficient, expands the radii of the cone

Bluetooth Setup

Initializing BLE on Raspberry Pi

The Bleno Library is a Node.js module for implementing Bluetooth capabilities on the Raspberry Pi. The Raspberry Pi will need NPM and Node.js installed (v8 recommended for full BLE usability) along with the following prereq applications. Use the command:

```
sudo apt-get install bluetooth bluez libbluetooth-dev
libudev-dev
```

From there, the Bleno library and all helper libraries can be installed via npm:

```
npm install bleno xml2js fast-xml-parser xmlbuilder
```

Simply run `npm install` if the node_modules from the project already exists on the pi.

Finally, install the Forever library via npm and run:

```
forever start service.js
```

Account Setup on Application

To begin using the app, a user must sign in via Google or Apple. From there, a user can add a device under Settings -> Manage Devices -> Add. The device must be powered on and in range of the phone. Once the device is added, all data will be populated as the device is used.

Starting the Device

To start the crash detection device you first have to initialize the device following both setup instructions. Once the power supply is confirmed to be able to supply 12W of power, connect the crash detection device to the power supply, it will begin sampling data within ~20 seconds.

Research Mode

Research mode allows for variable IMU sampling as determined by the device configuration file. This mode stores a complete log of IMU samples from across the entirety of a ride and stores the data in `kadd-pi/data/rides/imuComplete` This means that it has the potential to record an excess of data for transmission to the cloud database. GPS data will still be written out to `kadd-pi/data/rides/current`. Manual pull of data with an external memory device or hooking the Pi up to a monitor is required to view and export the data collected. To start in research mode set `mode` to 1 in `kadd-pi/data/about.xml`. It is **NOT** recommended for normal use as **this will disable emergency alerts**.

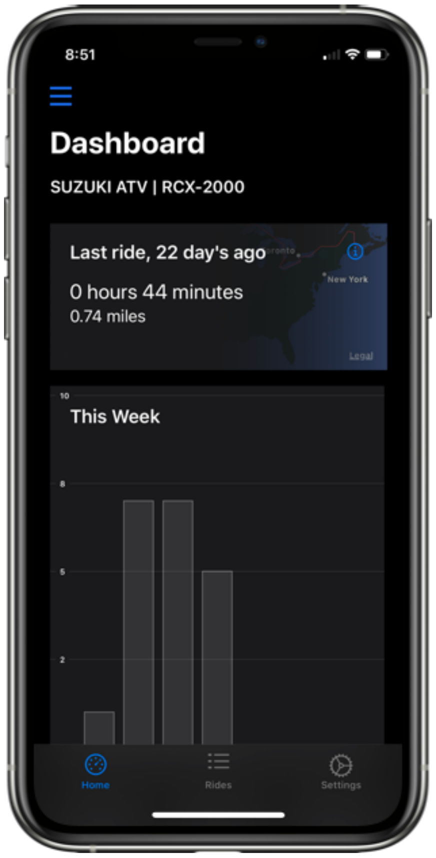
Farm Mode

Farm mode is the standard use case of the device. It will record the GPS according to the user defined rate, and record the IMU once a second. Unlike research mode which logs all IMU data, Farm mode simply keeps a log of the last minute of data. Upon a rollover, this data is written out to a file in `kadd-pi/data/rides/current`. GPS data will also be written to this file at the selected sample rate with extra samples being written whenever a rollover is detected.

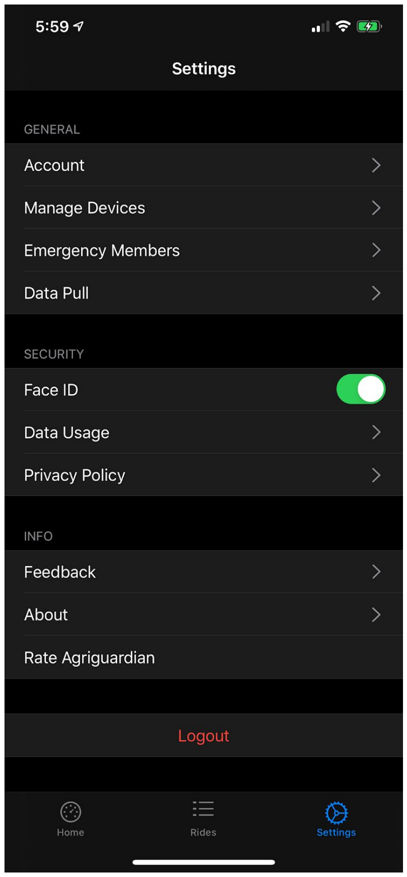
Functionality

Registering Devices

Within the application, first go to the Settings tab in the bottom right corner.



Then press Manage Devices under the General section.


5:59

Settings

GENERAL

Account

Manage Devices

Emergency Members

Data Pull

SECURITY

Face ID

Data Usage

Privacy Policy

INFO

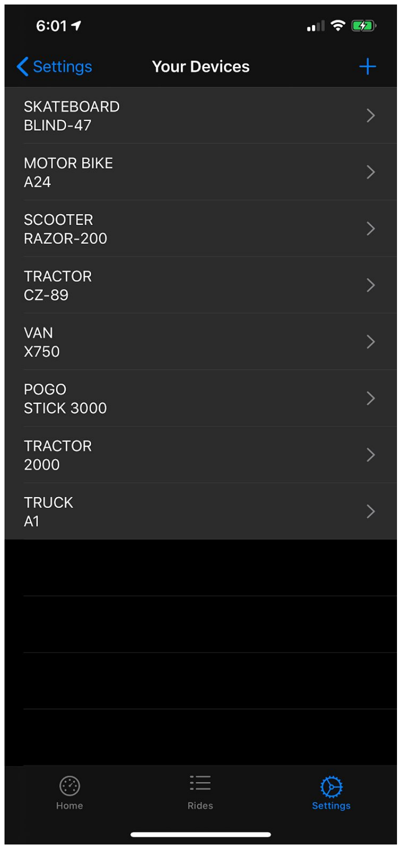
Feedback

About

Rate Agriguardian

Logout

Select the “Add” icon in the upper right hand corner of the view.


6:01

Settings

Your Devices

SKATEBOARD
BLIND-47

MOTOR BIKE
A24

SCOOTER
RAZOR-200

TRACTOR
CZ-89

VAN
X750

POGO
STICK 3000

TRACTOR
2000

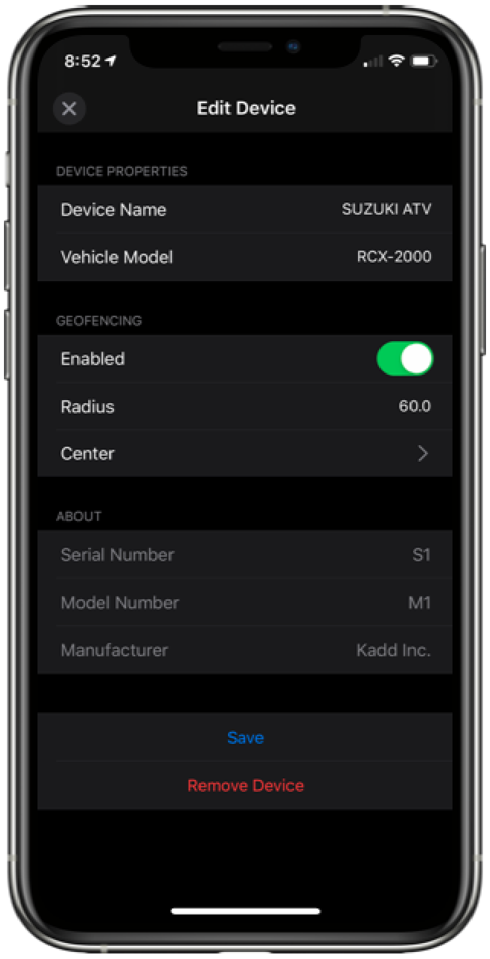
TRUCK
A1

If the device is powered on and in range, then the application will connect to the device and allow you to add the device under the user's profile. Follow the onscreen parameters to finish setting up the device.

Setting Geofences

Geo-fence allows device owners to limit the field of operation of registered devices. Geofencing settings can be set when first initializing a new device (See above). Secondly, geofencing parameters can be changed by editing a device under Settings -> Manage Devices.

When editing or adding a new device, location the Geofencing section. Here you can toggle to enable or disable geofencing and then set the center and radius for your geofence constraints.


8:52

Edit Device

DEVICE PROPERTIES

Device Name	SUZUKI ATV
Vehicle Model	RCX-2000

GEOFENCING

Enabled	✓
Radius	60.0
Center	>

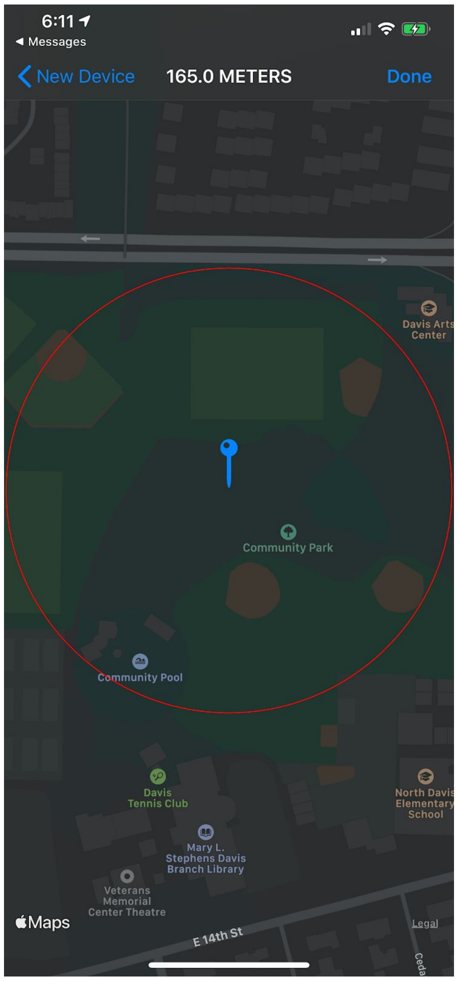
ABOUT

Serial Number	S1
Model Number	M1
Manufacturer	Kadd Inc.

Save

Remove Device

When selecting the appropriate center and radius for your geofencing, you will prompted onto a map view where you can drag the pin to set your center location for your geofence and zoom in and out to adjust the radius (in meters).

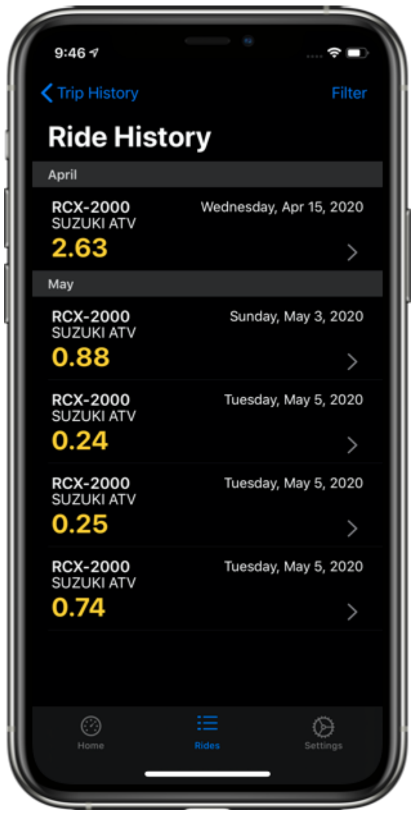


If a Geofence violation is detected on the crash detection device, device owners will be notified with a notification the the app. The notification will inform you of the location the device exited the Geo-fence boundary.

Viewing Ride History

If a registered WiFi connection is available to the crash detection device while power is supplied, all riding history will be uploaded to our server hosted on FireStore. All ride history can be viewed via the middle ride history tab in the bottom tab bar of the application.

All the rides for a device are organized in chronological order by month per year and by rides per month. After selecting a particular month within a year, you will then be shown a table of rides within the specified month.



Ride History

April

RCX-2000 SUZUKI ATV Wednesday, Apr 15, 2020
2.63

May

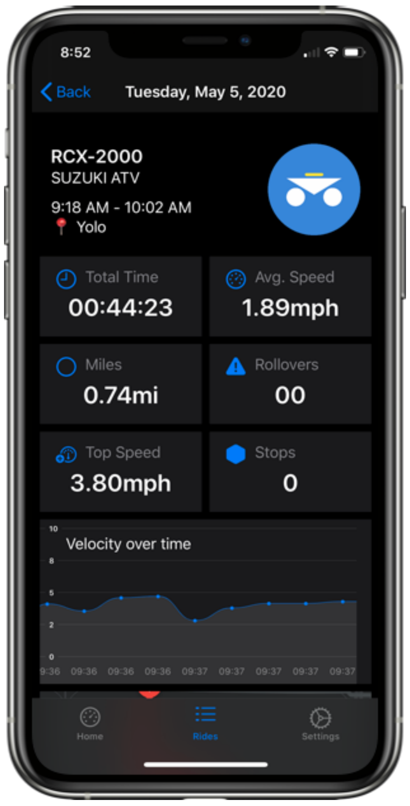
RCX-2000 SUZUKI ATV Sunday, May 3, 2020
0.88

RCX-2000 SUZUKI ATV Tuesday, May 5, 2020
0.24

RCX-2000 SUZUKI ATV Tuesday, May 5, 2020
0.25

RCX-2000 SUZUKI ATV Tuesday, May 5, 2020
0.74

Now with each ride divided by months, to view the ride history for one particular ride, you can select the ride you want to observe data for. This will show you all relevant riding parameters that were recorded during the ride.

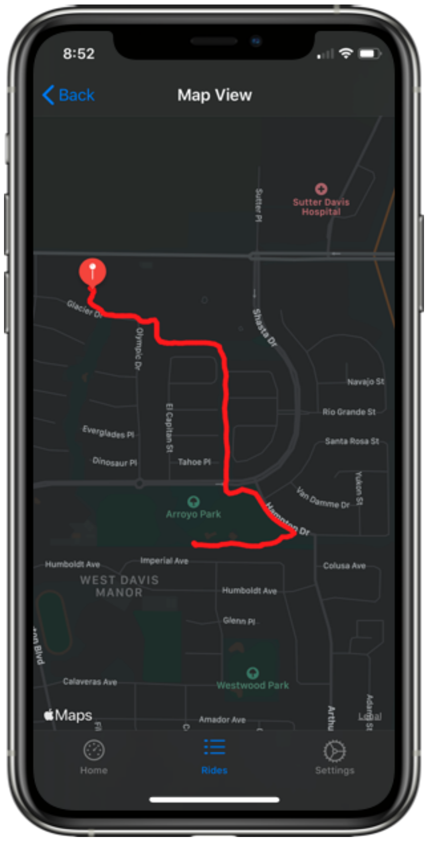


Tuesday, May 5, 2020

RCX-2000
SUZUKI ATV
9:18 AM - 10:02 AM
Yolo

Total Time 00:44:23
Avg. Speed 1.89mph
Miles 0.74mi
Rollovers 00
Top Speed 3.80mph
Stops 0

Scrolling to the very bottom of this view, you can see the route that the ATV had driven on a map for the chosen ride. If you click the map, then it will expand into a full screen map which you can interact with in order to better view the route of the ATV during its ride.



Automatic Data Upload

Once a WiFi connection is established when in farm mode, all logged data will be automatically uploaded to our server hosted on FireStore. To keep complete records of all ride history, routinely allow the crash detection device to upload data.

Manual Data Pull

Users can manually pull data from the crash detection device to the phone via bluetooth. The manually pulled data will be uploaded to our database using available connection on the phone. Make sure the device is on and in range. Go to Settings -> Data Pull and select start to begin the transfer. Depending on the data size, it could take a few minutes.

Contacting First Responders

In case of a roll over detection, the rockblock modem equipped on the crash detection device will notify our server. Without a registered number on the device, our server will redirect the crash coordinates sent by the device to a first responder using a 3rd party application called Noonlight.

In case a phone number is registered, instead of contacting first responders we will notify the device owner with a phone notification regarding the accident. Further explanation why we do not text the registered number in the Moving Forward section.

Troubleshooting

Unable to add device after connecting to it over Bluetooth

The app may fail to load the add device view after connecting to it. In this case, quit the application and reopen. If it fails again, it may be a general bluetooth error so the best case to solve this would be restarting the phone.

Failed Manual Data Pull

Similar bluetooth error to the above situation. Quit the application and if that fails, restart the phone and try again.

App not displaying rides properly

Ensure that all rides recorded by the Pi have been transmitted to the database. Check `/home/pi/kadd-pi/data/rides/unsent` to see if it is empty. If it is not empty, ensure that you are connected to WiFi and have network access. If it is, check `rides/current` and see that it is either recording the ride that you expect it to, or if it has excess rides that have not been moved to `rides/unsent`. If there are excess rides, feel free to move them to the unsent folder and they will be uploaded to the database. Finally, if none of these steps worked see if lastRide in `/data/rideHistory.json` corresponds to the highest indexed ride in `/rides/sent`.

Device no longer logs data or stops collecting at random points

The most likely cause of this issue is that the GPS has lost connection and is unable to regain it or is not able to connect to the satellite network in the first place. Try turning off the system, disconnecting the external antenna, powering the system on, and reconnecting the antenna. This should reset the GPS' antenna preference and force it to select the external antenna. If the connection is not physically damaged, you should see the FIX light stop blinking within 3-5 minutes.

If this does not fix your issue make sure that the GPS' internal antenna has a clear view of the sky and disconnects the external antenna. If the FIX light remains blinking for more than 10 minutes the GPS may not be wired correctly or has failed.

FIX light is not blinking but data is not being written

Ensure that the UART terminals are connected correctly (i.e. rx->tx and tx->rx)

Nonsensical IMU values

Restart the device, this will reset the IMU. A more permanent solution would be resoldering the board/connections to the Pi as this is likely caused by either excess noise or an issue with the Pi's SPI connection.

Crash notification was not sent despite an roll over happening

Rockblock is a subscription based service with a pay-by-packet plan. Contact BAE regarding the incident to verify the subscription is active and credits are available.

In addition, FireStore is only free upto a certain number of accesses. User notifications require an update on the database which consumes free credits. In case of unsuspected growth in user access, the server will not be responsive until admin upgrades the service.

Raspberry Pi will not connect to Wifi despite successful connections prior

This is a known issue of the Raspberry Pi where an ethernet connection overwrites a config file disabling Wifi connections. Restart the device to reset, and if this does not work plug in necessary accessories to manually reset the Wifi setting.

Contact Information

Kaito Yoshida
kaiyoshida@ucdavis.edu

Austin Wald
ajwald@ucdavis.edu

Davy Chuon
dchuon@ucdavis.edu

Daniel Weatrowski
dweatrowski@ucdavis.edu

Appendix: Design Document

Technology Survey

App Development

- Java
    - Pros:
        - Great libraries for expansive development tools
        - Widely used with tons of documentation available
        - Simple app acceptance process
        - Established and consistent style standards
    - Cons:
        - Complex readability
        - Android not as popular in the USA
        - Security / copyright issues
- Swift
    - Pros:
        - Larger population of iOS users in the USA
        - Simple, yet intuitive for powerful functionality
        - Type safety and type inferencing
        - Storyboard UI
        - Scalability
    - Cons:
        - Strict, selective app acceptance process
        - Apple device dependent
        - New / young language

Database

- MySQL
    - Pros:
        - SQL-based database (relational model)
        - Foreign keys

  - Cons:
    - Requires web-hosting server
    - Requires PHP knowledge to integrate
    - Cost

- MongoDB
  - Pros:
    - Security
    - Suitable for large-scale applications
    - Powerful storage capabilities, querying, and indexing
  - Cons:
    - NoSQL-based database (document store)
    - No foreign keys
    - Cost
- Firebase (Cloud Firestore)
  - Pros:
    - Easily configured via Cocoapods
    - Robust client libraries
    - Offline mode fully supported
    - Real-time database interaction / operations
    - Cheaper cost (initially free)
  - Cons:
    - NoSQL-based database (document store)
    - Suitable for small-scale applications
    - No foreign keys

Transmission Methods

- Bluetooth
  - Pros:
    - No fee
    - Many libraries for all devices
    - Reliable data transmission
  - Cons
    - Non-sufficient range
- Software Defined Radio
  - Pros:
    - Low or no fee
    - Sufficient range
  - Cons:
    - Poorly researched field
    - Unreliable encryption and decryption of data

        - Require base with alternate communication method to relay data
- Firstnet
    - Pros:
        - Satellite communication with library
        - Well known and reliable
    - Cons:
        - Must pay by month service fees
        - Requires expensive and inflexible module
- Rockblock
    - Pros:
        - Satellite communication with library
        - Allows pay by packet
    - Cons:
        - Have to pay to use service
        - Inflexible packet size(64 byte packet)

Computer/Microcontroller

- Arduino
    - Pros:
        - Lower power consumption
        - Beginner friendly
        - Sleep system integrated
    - Cons:
        - No OS
        - Not great at handling multiple tasks
        - Slower CPU
- Raspberry Pi
    - Pros:
        - Has OS
        - Good at multitasking
        - Can update its code easily
        - Expandable
        - Faster CPU
    - Cons:
        - Higher energy consumption
        - No sleep system to limit power energy consumption

Requirements

User Story	Description
#1	As an ATV rider, I will be able to automatically raise a panic alert for first responders during the event of emergency even if I am unable to do so.
#2	As a researcher, I will be able to collect and observe riding parameter data such as speed, rotation, vibration, location, trip history, surface slope, surface texture and pitch and roll.
#3	As a researcher, I will be able run a script that generates a spreadsheet with riding parameter data that can be analyzed for research.
#4	As a farm owner, I will be able to monitor, view riding parameter data and trip history for my ATV(s) via an iOS app.
#5	As a farm owner, I will be able to save and interact with my ATV(s)’ data (current and history) at any time via an iOS app.
#6	As a farm owner, I will be able to geofence my ATV(s) so that they may only be permitted to operate within a certain region.
#7	As a farm owner, I will be alerted whenever my ATV(s) has/have crashed.
#8	As a farm owner, I will be alerted when the ATV(s) are being stolen.
#9	As a farm owner, I will be able to remotely turn off the ATV(s) if stolen (or in any case will have the ability to do so).
#10	As a farm owner, I will be able to determine if an ATV is being used in/appropriately and can identify high risk operators.
#11	As a farm worker, I can rely on the device to notify the farm owner and first responder in case of emergency.

#12	As a farm owner, I can be alerted about dangerous riding behaviors to correct my employees and or myself.
#13	As a researcher, I can filter the database by users, regions, etc to better correlate behavior to causes.
#14	As a farm owner, I can modify my geo fencing settings and let other farms rent my ATV while monitoring them.
#15	As a farm owner, I do not have to worry about charging the crash detection device due to it being powered and charged from the ATV generator.

Technology Employed

Databases

Firebase Cloud Firestore: A flexible, scalable NoSQL cloud database used to store and sync data which provides efficient real-time updates, expressive querying, and offline support.

iOS Application

Swift: The primary programming language for iOS app development that is fast, powerful, safe, and intuitively simple. The IDE being used is XCode 11.

Cocoapods: A dependency manager for Swift, built with Ruby, that allows packages to be installed and used in our Swift application that provides customizable features and functionality.

Hardware Application (Hardware + Python code)

Raspberry Pi4: A single board computer with wifi, bluetooth, and 4 usb ports. We installed a linux based OS, Raspbian, to provide an easy visual development environment. We are using the onboard GPIO pins to connect the breakout boards for the other sensors.

Python: The official preloaded programming language for Raspbian OS. Most sensors made for the Raspberry Pi provide a Python based library. It also has multi-thread capability which allows us to run multiple sensors with ease.

NodeJS/Bleno: Bleno library utilizes the Raspberry Pi's built in BLE modem and allows the Pi to advertise BLE signals to the iPhone application.

Google Cloud Firestore and Firebase Admin: Libraries that allow simple firestore/firebase integration and management via a Python script.

Adafruit-Blinka: Allows the Raspberry Pi to easily interface with Adafruit hardware components via sensor specific libraries. This library emulates CircuitPython libraries, allowing all manner of easy GPIO expansion when the user is within the Adafruit ecosystem.

RockBlock: Satellite data transmitter that utilizes iridium. It has a pay by packet plan that we will use in case of emergencies. It will allow us to communicate with first responders when cellular connection is not available.

IMU / sensors: We are using IMU sensors that communicate with the Raspberry pi with SPI to measure pitch, yaw, roll, acceleration, vibration of the ATV. These data allows us to analyze user driving behavior for research purposes.

GPS sensor: GPS sensor with breakout-board that powers the sensor with an independent battery. It communicates with the Raspberry Pi through UART. It utilizes multiple satellite connections to accurately measure the GPS coordinates. We will be storing and transmitting these data to make a trip history and allow real time tracking if cellular connection is available.

Tools

Slack: A cloud-based instant messaging platform that allows users to effectively communicate and collaborate as a team on a working production. The service has the feature for divisions of subchannels to organize different aspects of the task.

GitHub: A platform that is a host for software development as it provides version control and source code management for solo or collaborative work.

Cost Analysis

As a whole, BAE MSL wants this system to have a net cost below $500. The hardware is well below this threshold, but the satellite modem subscription drives the cost to just over this budget, but not by a prohibitive amount.

- Device Cost:
    - Raspberry Pi 4 - $40
    - IMU (Adafruit LSM9DS1) - $14.95
    - GPS (Adafruit Ultimate GPS Breakout v3) - $39.95
    - RockBlock 9603- $214.73

  - Rechargeable Battery - $50
  - Current Upfront Total - $359.63 (w/ battery) $309.63 (w/ ATV battery)
    - This is subject to change based on RockBlock Data usage and whether the device is being powered by the ATV’s onboard battery or a stand-alone portable one.
- Ongoing Fees:
  - Rockblock Line Rental - $14.72
  - Rockblock Credits - $0.003 per byte
    - Different packages can be purchased which offer better cost to data ratios. This is the worst case cost where users buy batches of 100 credits. Best case cost is $0.001 per byte, but is only economic for large deployments.
  - Estimated Annual Fees - ~$195/yr for first device, ~$176/yr for subsequent
- Grand Total: $554.63 (w/ battery) $504.63 (w/ ATV Battery)
  - The portable battery is currently a panacea for not being able to gain access to the ATVs due to COVID-19 related campus closures as we are currently unable to configure the device with an onboard battery.

Github Repository

iOS: https://github.com/daygodavy/KaddSeniorDesign
Pi: https://github.com/OfcPeriwinkle/kadd-pi

Social and Legal Aspect

1. Legal

a. User’s will be aware that this project is largely for research purposes. All riding data will be saved and analyzed by researchers at BAE labs. Most importantly, users will be sharing their location while riding the ATV to all members of the BAE and Kadd teams. All data will be secure and private to anyone/any companies outside of these two teams.

2. Social

a. Kadd and BAE CANNOT ensure the safety of any riders utilizing this project. We do our best to make sure everyone will get the care they need in the case of an emergency however we do not take responsibility otherwise.

Moving Forward

Over the last twenty weeks we have developed a basic framework for the UCD BAE MSL research team led by Dr. Khorsandi. However, as with all things worth doing, this project still has a long road to travel before becoming what we would consider complete and robust enough to become a commercial product. This is mainly a result of the COVID-19 pandemic in conjunction with the wide set of challenges associated with a project of this scope. Not only was use case testing with an ATV an impossibility, but other features like a solid enclosure for the device and soldered components were also victims of the circumstances. But as a proof of concept, this device and companion app have come an impressively long way from the description provided at the start of this year. We are happy with what we were able to accomplish in trying to conquer the vast scope of this challenging project in the time we were allotted. We sincerely hope that you will be too. To help guide future development here are a few of our suggestions, most of which have already been discussed:

- Construction of a proper enclosure through cut acrylic or other materials that will not denature easily in high vibration and potentially high heat environment
- Plugs for peripheral connection ports when not in use to prevent sediment build-up
- Soldering of components to ensure connections to sensors
- Switch to an Arduino based platform to help miniaturize the project and help bring costs down
- ATV battery connection for power
- Find a cost effective solution for sending SMS messages to users (currently push notifications are sent, but this assumes the owner has an iPhone)
- Construction of a web app (and/or Android app) to allow for easier, less restrictive access to data
- Testing, testing, testing