PURDUE UNIVERSITY

CANdroid: Freeing ISOBUS Data and Enabling Machine Big Data Analytics

Yang Wang[1], Andrew Balmos[1], James Krogmeier[1], Dennis Buckmaster[2]
1: Electrical and Computer Engineering, Purdue University
2: Agricultural and Biological Engineering, Purdue University

Introduction

The past decade has seen a tremendous increase in the number of agricultural machines whose internal data communications follow the ISO 11783 (ISOBUS) standard. In many cases, the ISOBUS data never leaves the vehicle because the exporting process is commonly a painstakingly manual task with little reward. However, advancements in big data analytics have increased the value of data and created a demand for an easy and efficient way of acquiring ISOBUS messages. Recent attempts to collect ISOBUS messages for observing in-field fuel usage [1],[2] were made but the cloud integration was lacking. CANdroid, a recent redevelopment of ISOBlue [3], is a standard Android tablet enhanced by CAN peripherals that directly connect to the equipment's ISOBUS. As a result, developers can leverage preexisting cellular and WiFi connectivity and exploit app development knowledge that software engineers already have.

This poster discusses the design of CANdroid including an example Android app that utilizes the Open Ag Data Alliance [4] (OADA) API server. Additionally, we will present a reference design for processing the uploaded data using a modern, highly distributed stream processing scheme known as the Kappa architecture [5]. In particular, the log-centric database Apache Kafka [6] is used to coordinate a variety of single-purposed microservices that accomplish various processing tasks such as conversion of ISOBUS messages to raw data, creation of visualizations, such as yield maps, monitoring of equipment health, or pushing real-time alerts and data back to online consumers like CANdroid. This design paradigm allows for highly distributed, horizontally scalable processing chains that are simple to extend with little to no side effect to existing services.

Development

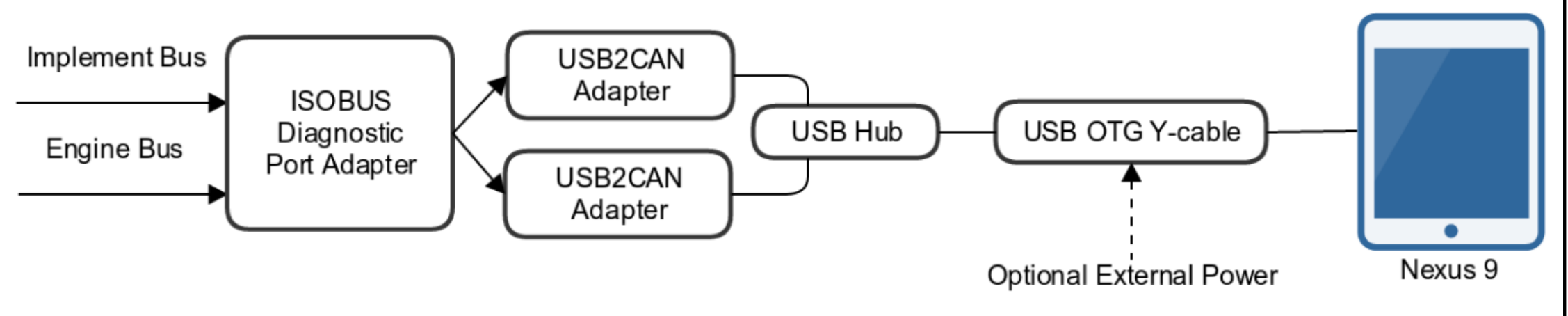


Figure 4: Overall system connection diagram for CANdroid

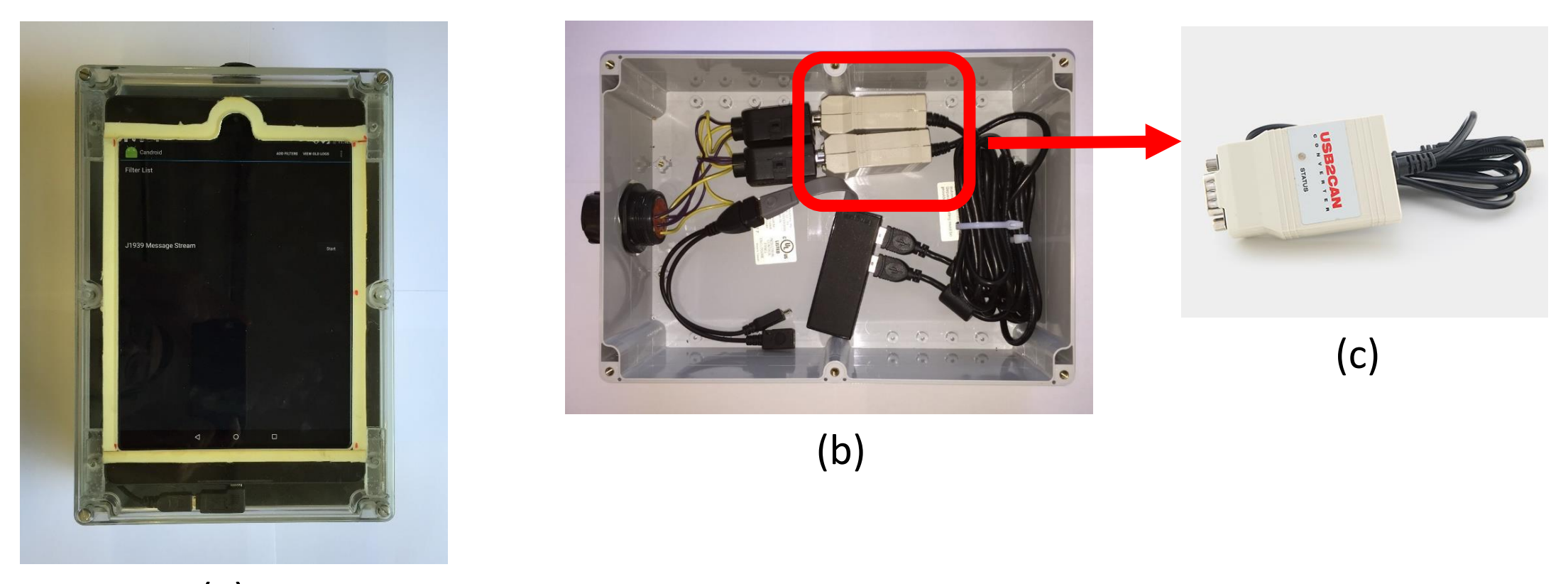


Figure 5 (a) to (c) : They represent top, side view of the CANdroid enclosure and the 8devices USB2CAN adapter respectively.

For hardware, we choose Nexus 9 as the main hardware platform for ISOBUS data collection and uploading. The tablet is built into a rigid enclosure for durability.

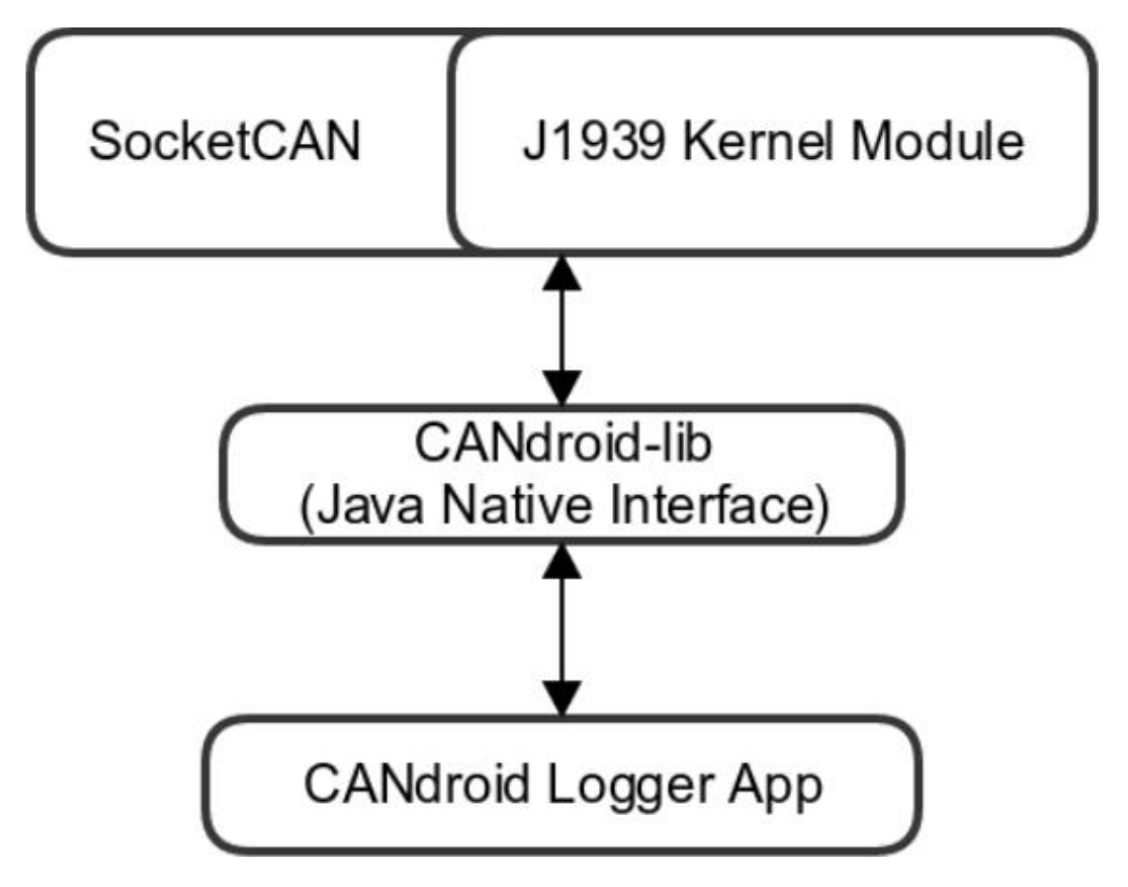
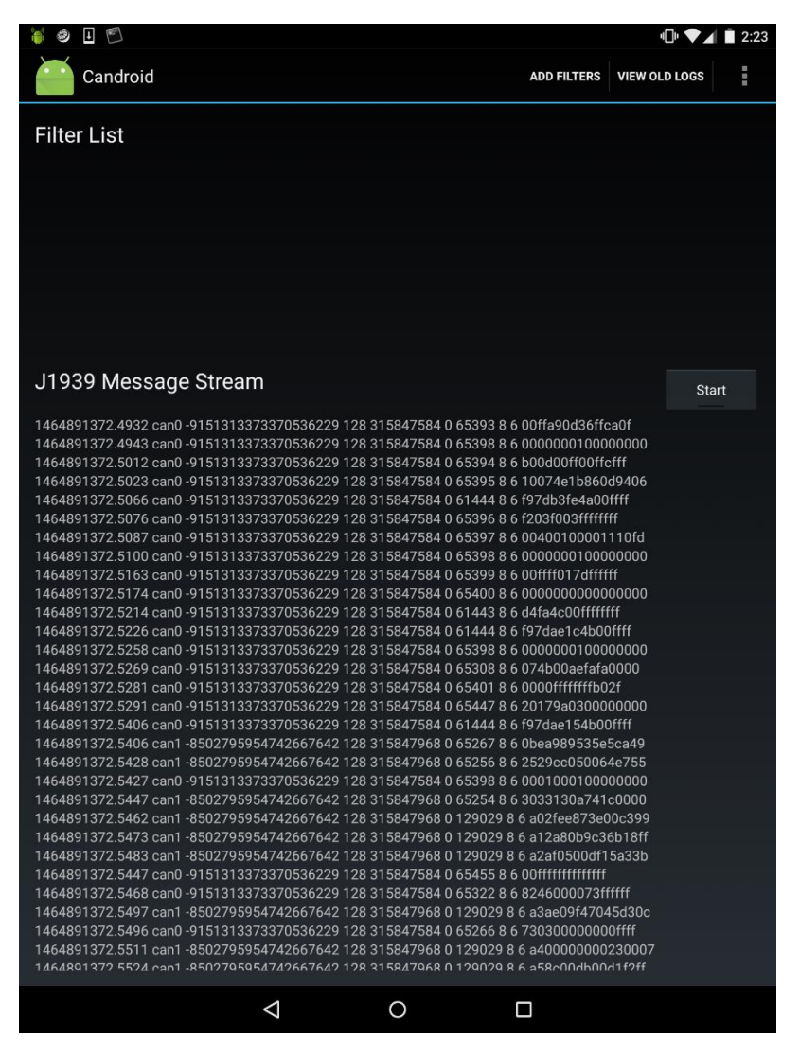


Figure 6: Software interactions for CANdroid. CANdroid-lib bridges the communication for the kernel module and the app.



```
1464891372.4932 can0 -9151313373370536229 128 315847584 0 65393 8 6 00ffa90d36ffca0f
1464891372.4943 can0 -9151313373370536229 128 315847584 0 65398 8 6 0000000100000000
1464891372.5012 can0 -9151313373370536229 128 315847584 0 65394 8 6 b00d00ff00ffcfff
1464891372.5023 can0 -9151313373370536229 128 315847584 0 65395 8 6 10074e1b860d9406
1464891372.5066 can0 -9151313373370536229 128 315847584 0 61444 8 6 f97db3fe4a00ffff
1464891372.5076 can0 -9151313373370536229 128 315847584 0 65396 8 6 f203f003ffffffff
1464891372.5087 can0 -9151313373370536229 128 315847584 0 65397 8 6 00400100001110fd
1464891372.5100 can0 -9151313373370536229 128 315847584 0 65398 8 6 0000000100000000
1464891372.5163 can0 -9151313373370536229 128 315847584 0 65399 8 6 00ffff017dffffff
1464891372.5174 can0 -9151313373370536229 128 315847584 0 65400 8 6 0000000000000000
1464891372.5214 can0 -9151313373370536229 128 315847584 0 61443 8 6 d4fa4c00ffffffff
1464891372.5226 can0 -9151313373370536229 128 315847584 0 61444 8 6 f97dae1c4b00ffff
1464891372.5258 can0 -9151313373370536229 128 315847584 0 65398 8 6 0000000100000000
1464891372.5269 can0 -9151313373370536229 128 315847584 0 65308 8 6 074b00aefafa0000
1464891372.5281 can0 -9151313373370536229 128 315847584 0 65401 8 6 0000ffffffffb02f
1464891372.5291 can0 -9151313373370536229 128 315847584 0 65447 8 6 20179a0300000000
1464891372.5406 can0 -9151313373370536229 128 315847584 0 61444 8 6 f97dae154b00ffff
1464891372.5406 can1 -8502795954742667642 128 315847968 0 65267 8 6 0bea989535e5ca49
1464891372.5428 can1 -8502795954742667642 128 315847968 0 65256 8 6 2529cc050064e755
1464891372.5427 can0 -9151313373370536229 128 315847584 0 65398 8 6 0001000100000000
1464891372.5447 can1 -8502795954742667642 128 315847968 0 65254 8 6 3033130a741c0000
1464891372.5462 can1 -8502795954742667642 128 315847968 0 129029 8 6 a02fee873e00c399
1464891372.5473 can1 -8502795954742667642 128 315847968 0 129029 8 6 a12a80b9c36b18ff
1464891372.5483 can1 -8502795954742667642 128 315847968 0 129029 8 6 a2af0500df15a33b
1464891372.5447 can0 -9151313373370536229 128 315847584 0 65455 8 6 00ffffffffffffff
1464891372.5468 can0 -9151313373370536229 128 315847584 0 65322 8 6 8246000073ffffff
1464891372.5497 can1 -8502795954742667642 128 315847968 0 129029 8 6 a3ae09f47045d30c
1464891372.5496 can0 -9151313373370536229 128 315847584 0 65266 8 6 730300000000ffff
1464891372.5511 can1 -8502795954742667642 128 315847968 0 129029 8 6 a400000000230007
1464891372.5524 can1 -8502795954742667642 128 315847968 0 129029 8 6 a58c00db00d1f2ff
```

Figure 7: Screenshot of the CANdroid logger app collecting data.

For software, there are three main parts:

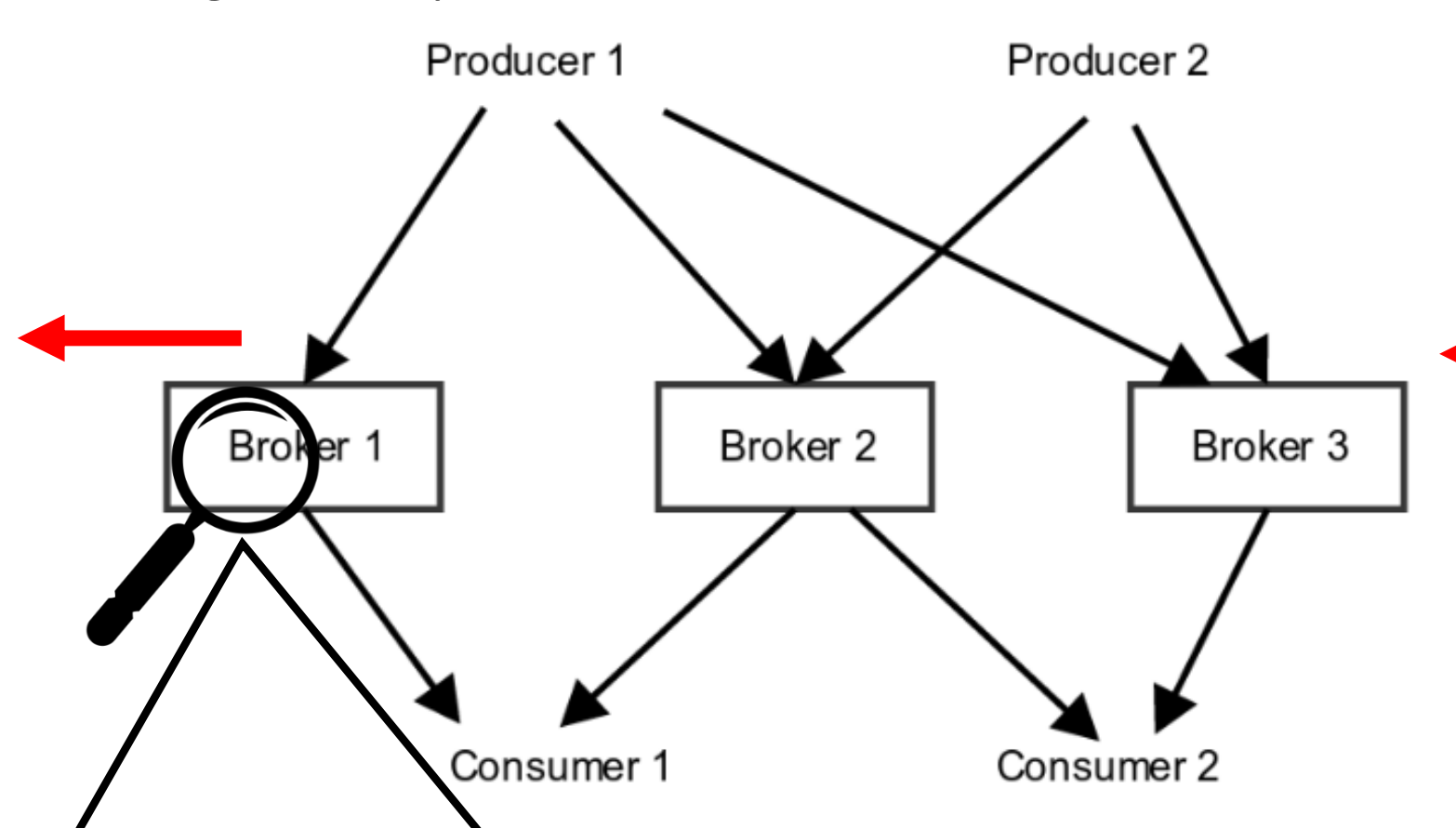
1. We have built a custom Linux image that enables the support for J1939 kernel module and USB2CAN adapter driver.
2. We have developed an open-source library called CANdroid-lib. It bridges the interactions between the kernel module and the Android app.
3. We have also developed CANdroid Logger app to collect data and upload them to the Cloud.

Kappa Architecture

The three main components of an Apache Kafka database:

- A **producer** generates message streams and sends out to a **broker**.
- A **broker** receives the messages and allocates them on different **topics** with one or more **partitions**.
- A **consumer** requests messages from one or more brokers and consumes the messages.

Figure 2: Top level overview of the Kafka database.



- CANdroid, as the producer in the Kappa architecture, collects ISOBUS messages and pushes them onto the topic Raw ISOBUS Message within the broker.
- External **microservices**, which function as both consumers and producers, parse useful data out from the raw ISOBUS message stream and pushes the parsed data back to the Kafka database on new topics.

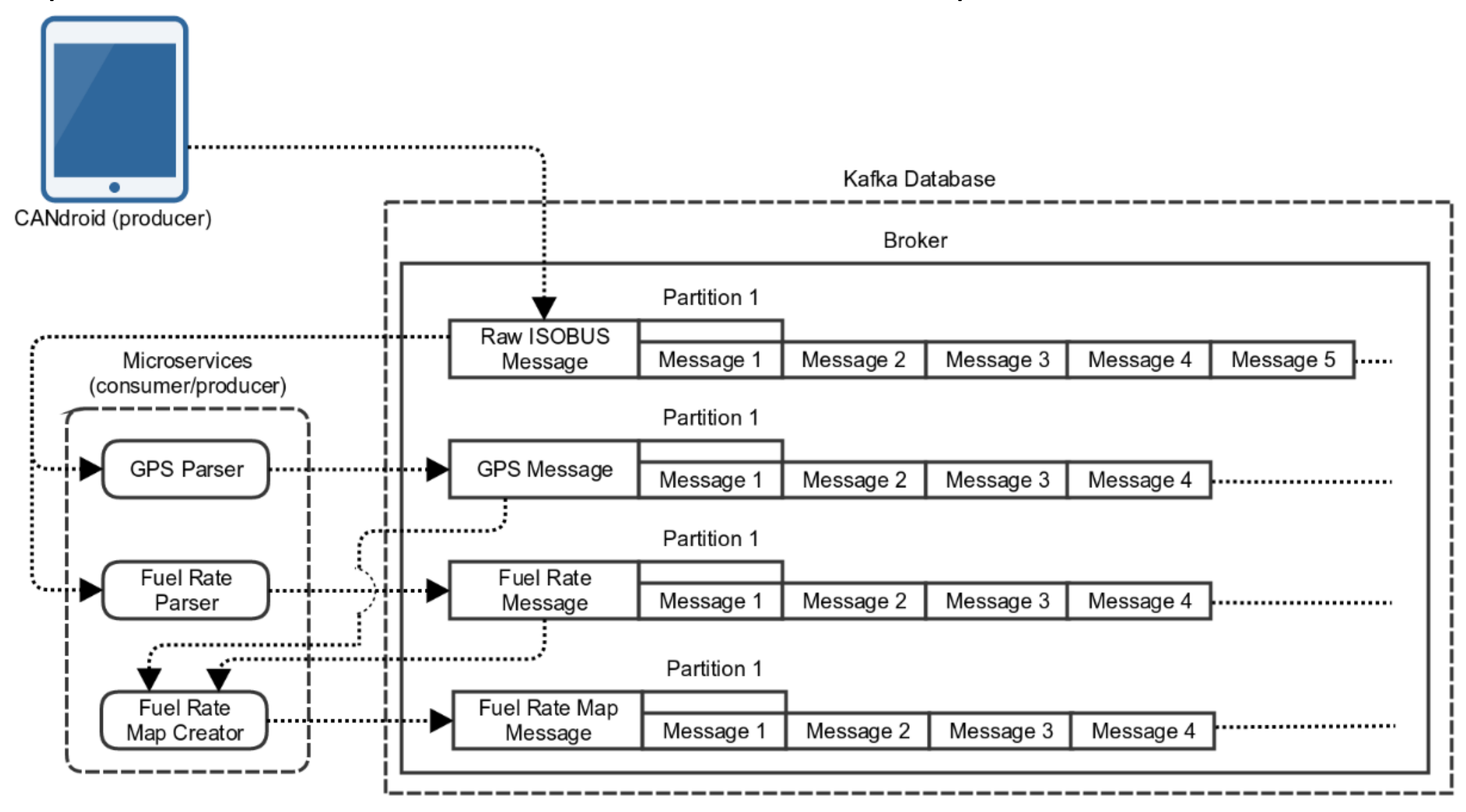
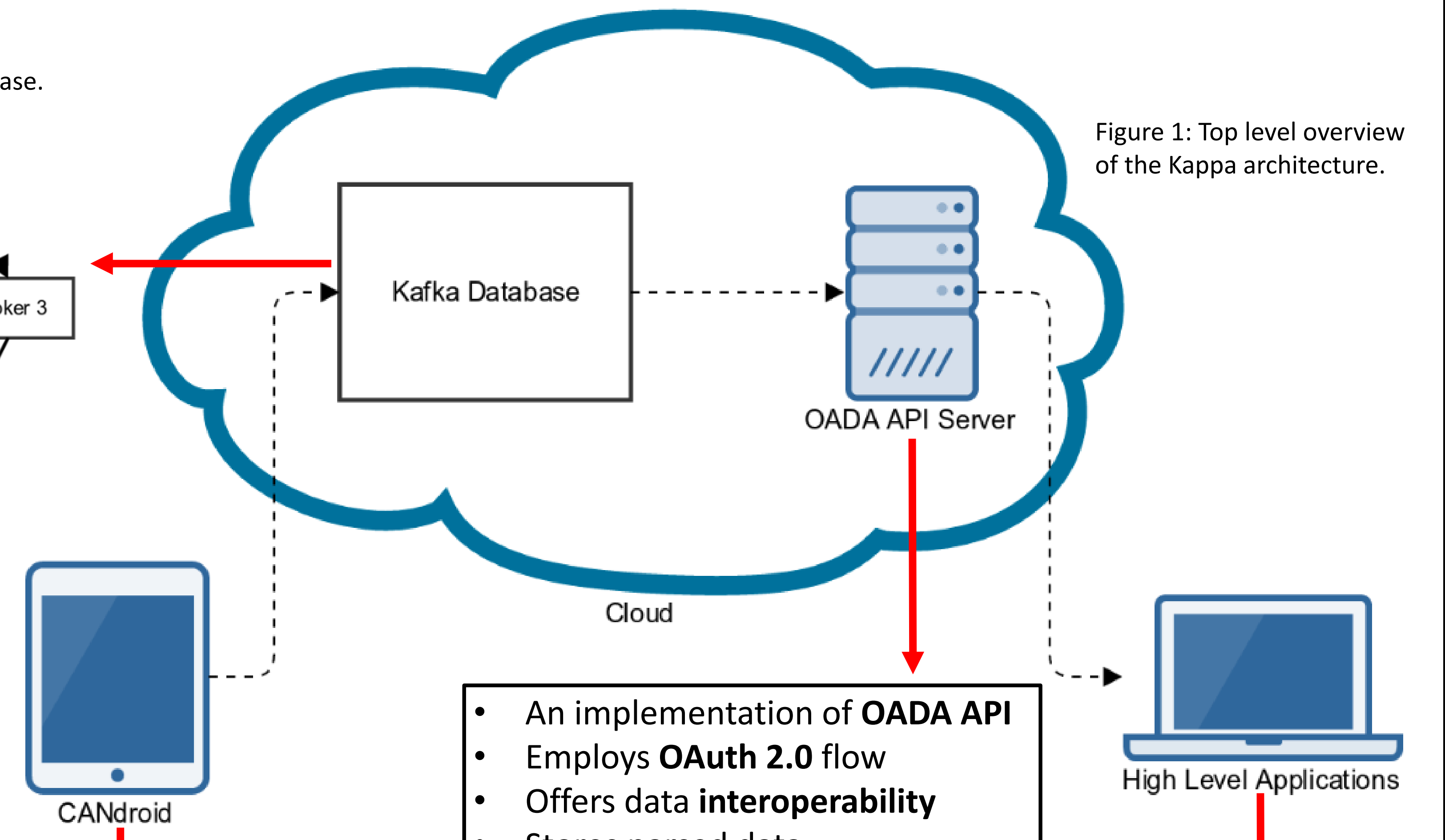


Figure 3: Detailed look into the Kafka broker.

Figure 1: Top level overview of the Kappa architecture.

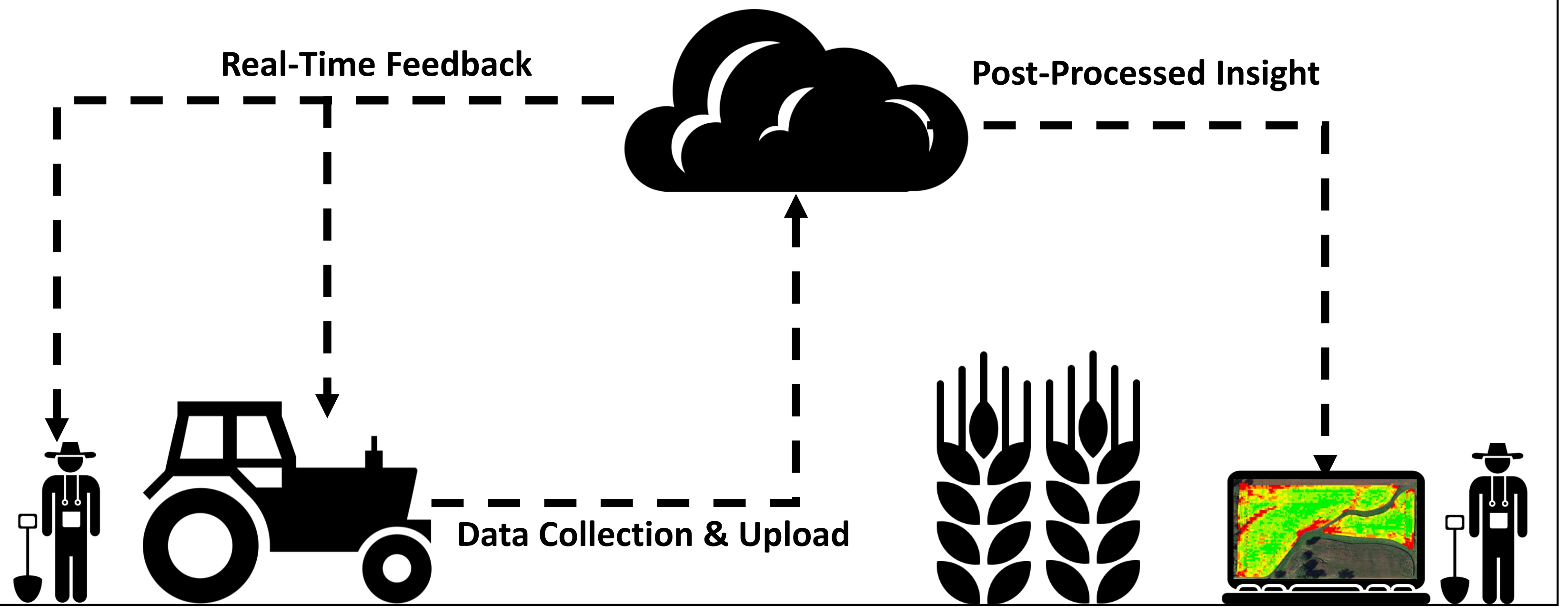


- An implementation of **OADA API**
- Employs **OAuth 2.0** flow
- Offers data **interoperability**
- Stores parsed data

- Could be replaced with other sensors, IoTs, etc.

- Data visualizations
- Aggregate data processing

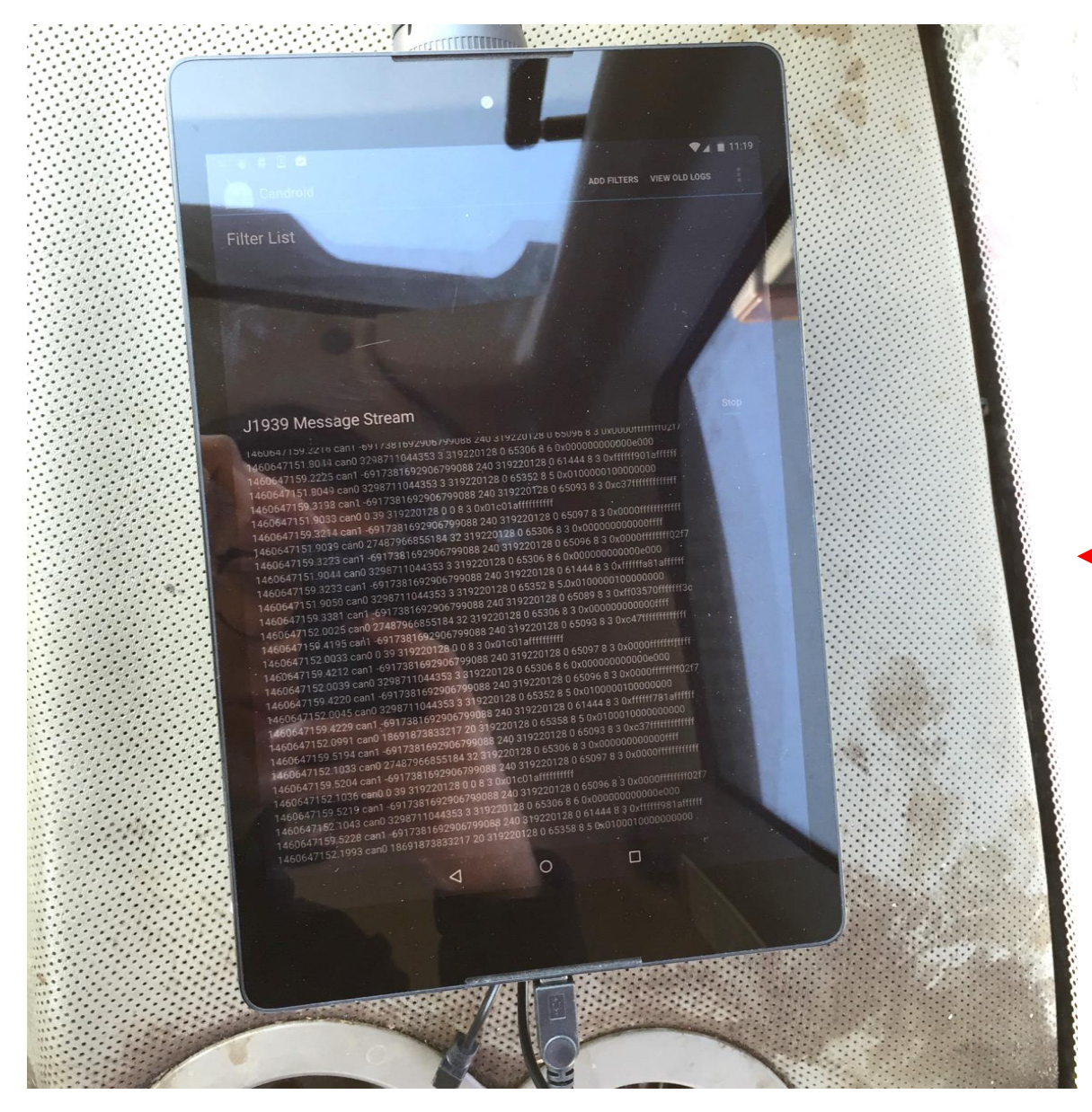
With the Kafka database's integration into the Kappa architecture, the architecture is scalable, efficient, and stream based with low-latency. Both real-time feedback and data post-processing are realizable with this Kappa architecture. This offers a continuous flow of ag machine data which is beneficial for farmers, researchers and manufacturers.



Experiment

We utilized CANdroid to collect ISOBUS data during a 2-hour long manure spreading session. CANdroid was connected to a tractor via its ISOBUS diagnostic port. There were 6 passes of manure spreading in total. In addition, during the manure spreading session, various aspects of the tractor (speed, RPM, location, etc.) were constantly changing. We bookkept some data such as vehicle speed and PTO on paper. Our intension was to compare the data recorded on paper with the parsed data from CANdroid to validate the usefulness of data collected from CANdroid.

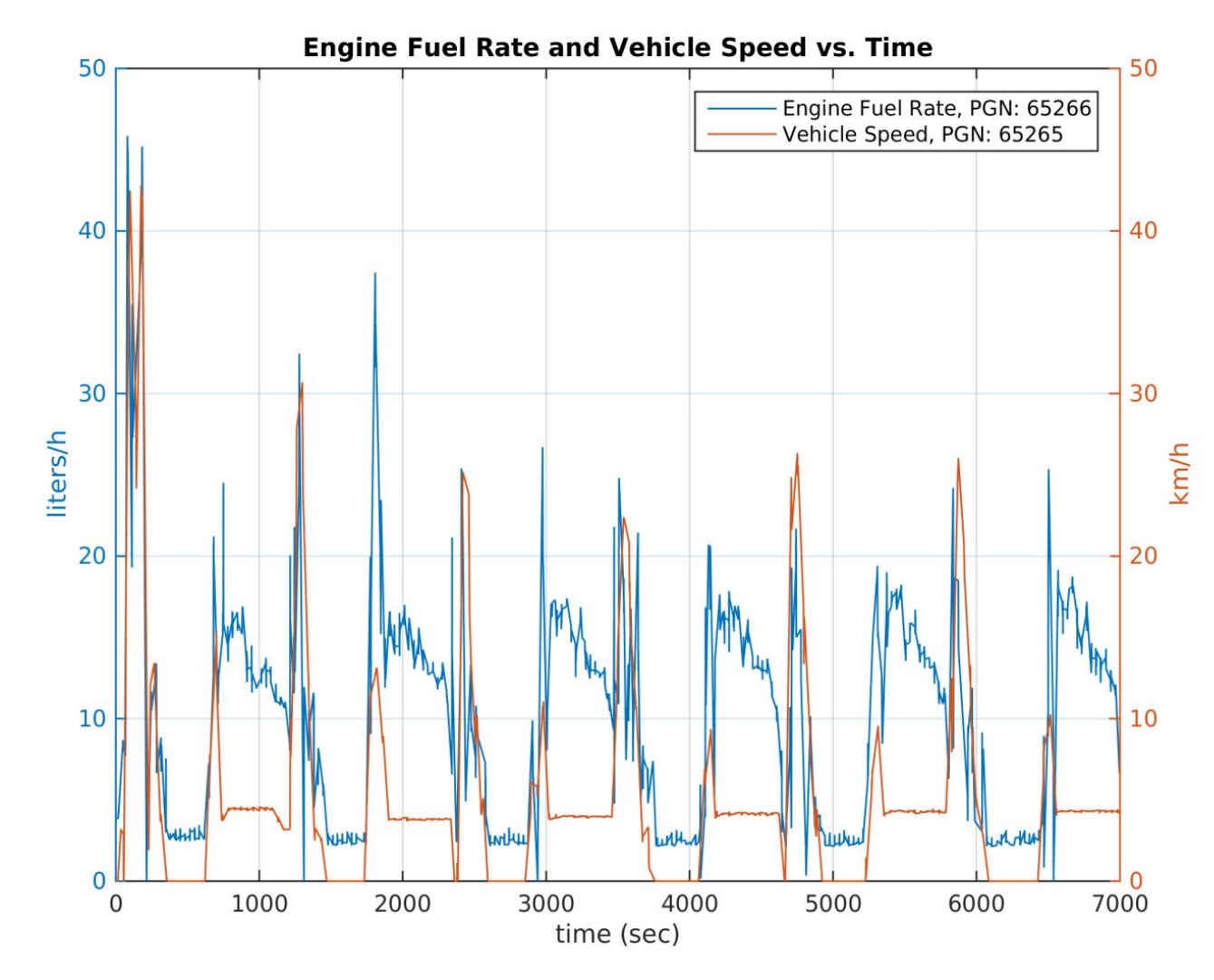
(a)



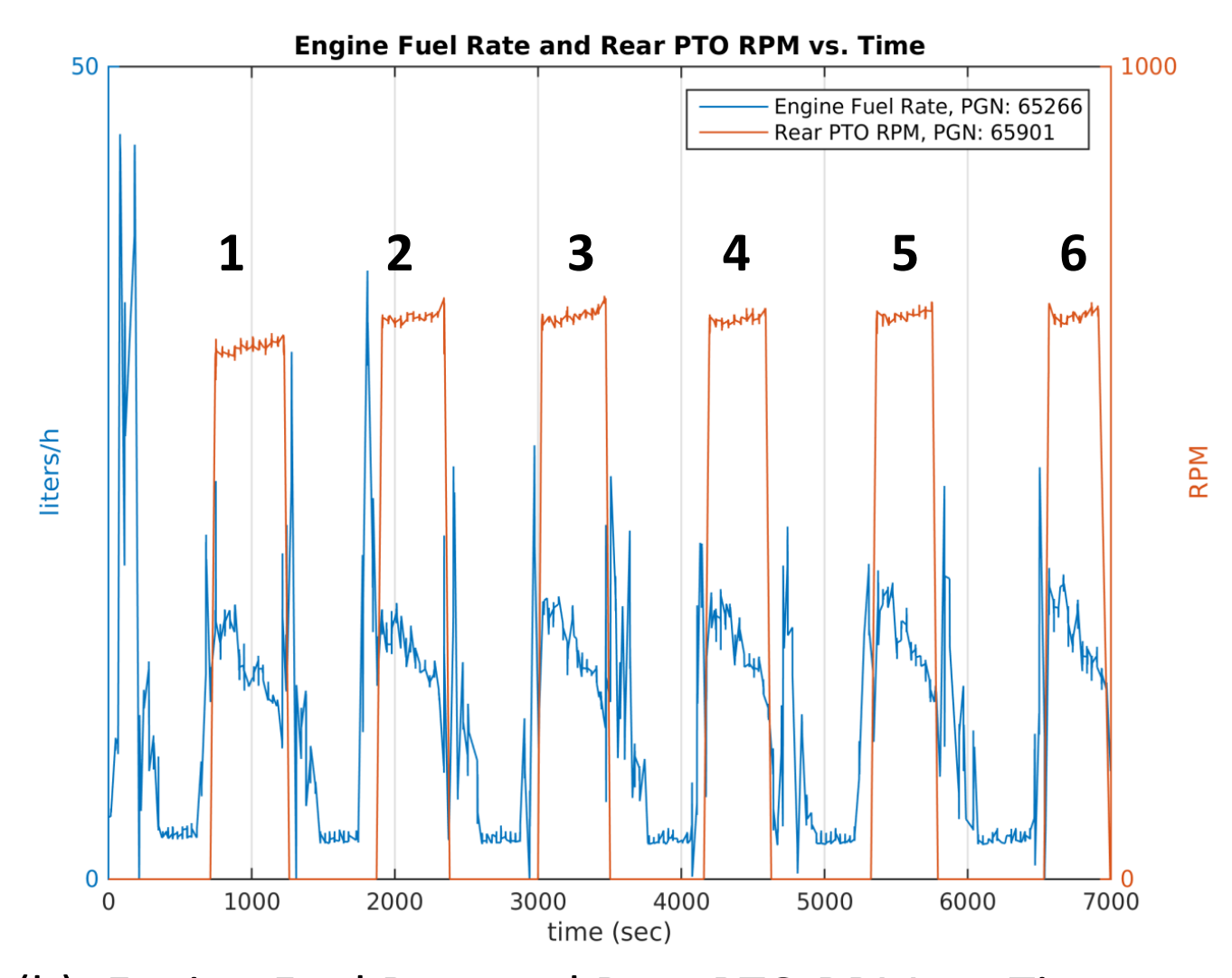
(b)

Figure 8: (a) Panoramic view of the cabin of the tractor to which CANdroid was connected. (b) Closer-up view of CANdroid collecting data during the manure spreading session.

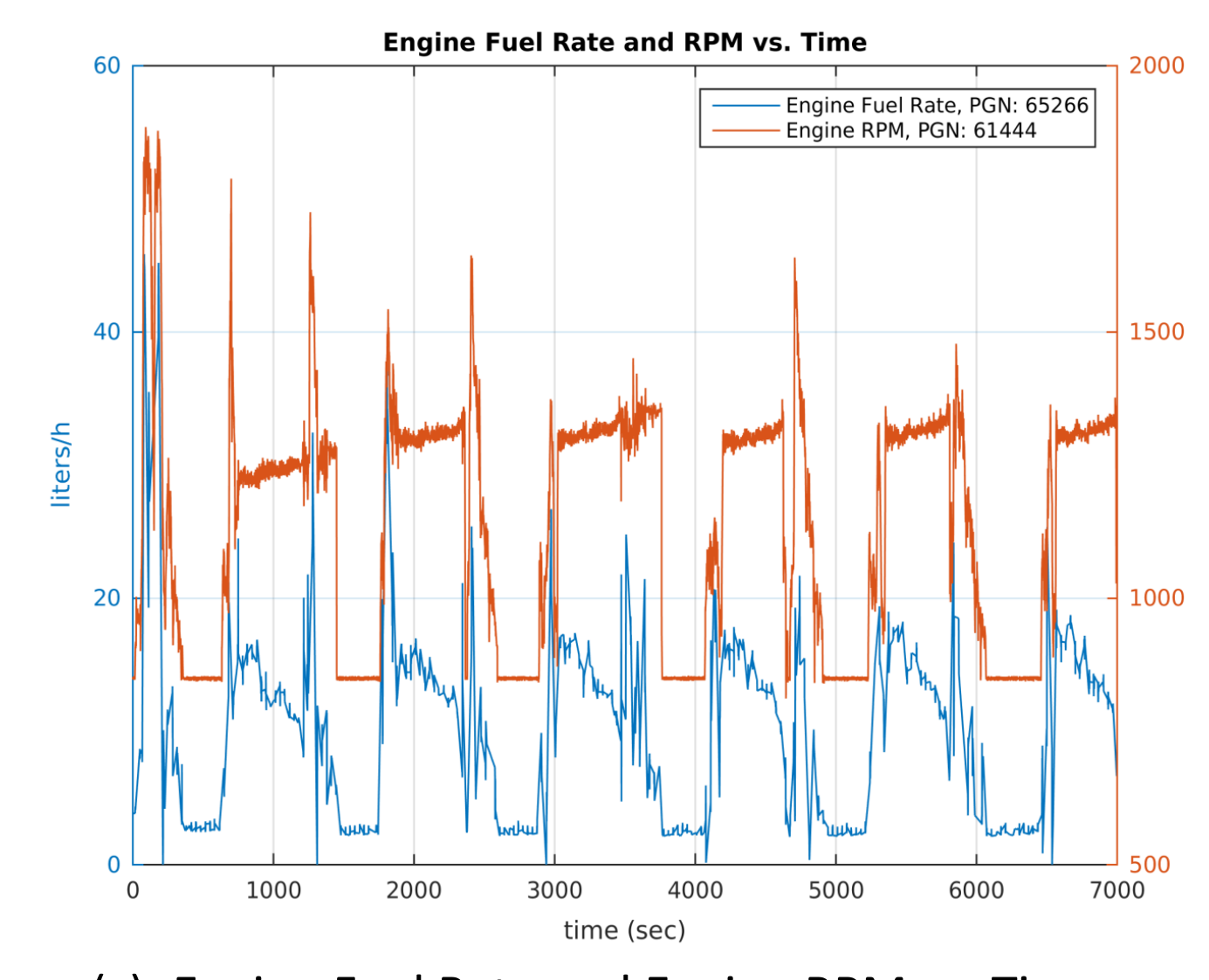
Results



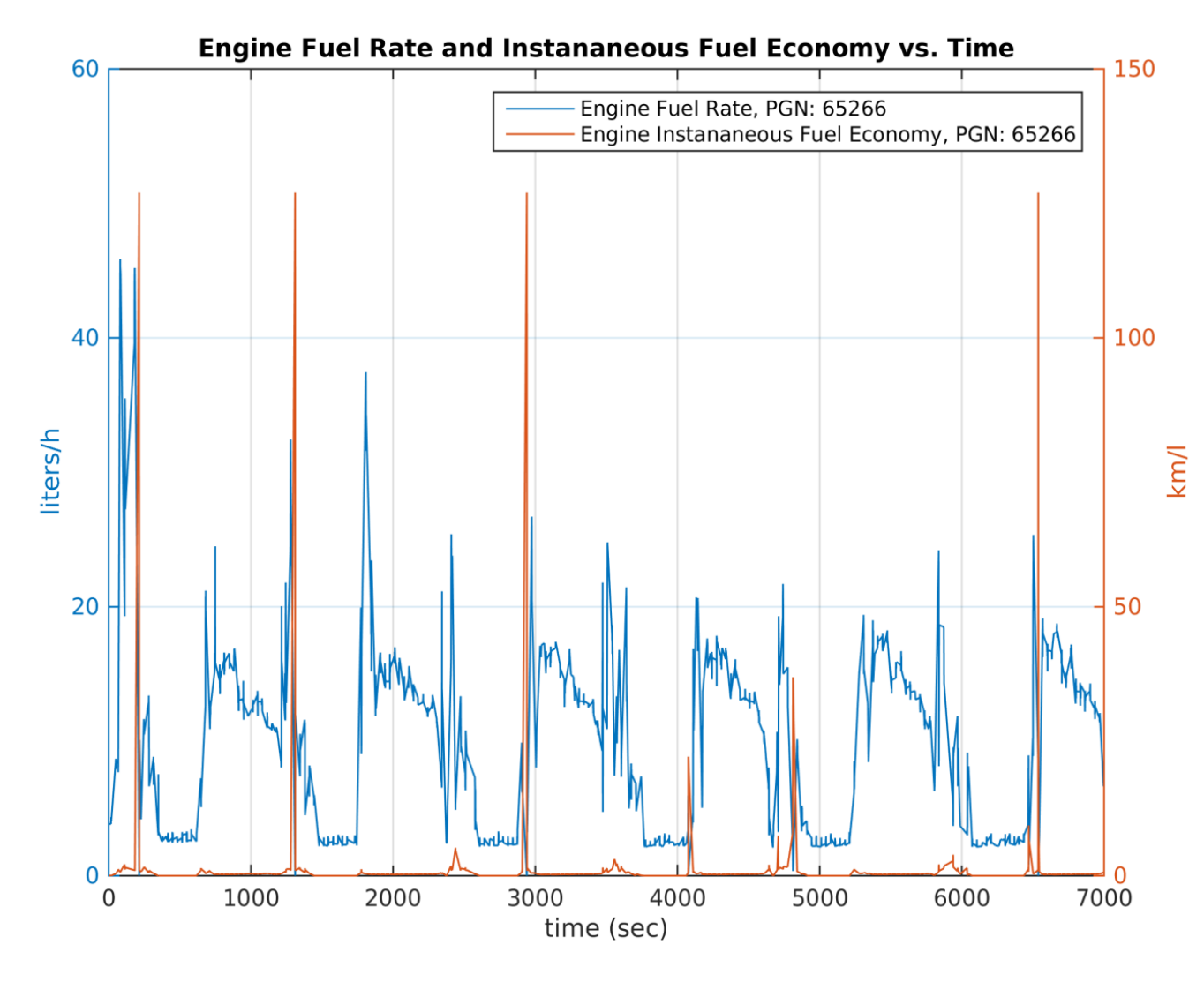
(a): Engine Fuel Rate and Vehicle Speed vs. Time



(b): Engine Fuel Rate and Rear PTO RPM vs. Time



(c): Engine Fuel Rate and Engine RPM vs. Time



(d): Engine Fuel Rate and Instataneous Fuel Economy vs. Time

Figure 9: Engine Fuel Rate and Various Other Data vs. Time

We utilized microservices to parse out CANdroid's collected data based on some known Parameter Group Numbers (PGNs) and plotted them. It is apparent that all these plots follow the same pattern that indicates there were 6 passes in total.

[1] Pitla, Santosh K., Joe D. Luck, Jared Werner, Nannan Lin, and Scott A. Shearer. "In-field Fuel Use and Load States of Agricultural Field Machinery." Computers and Electronics in Agriculture 121 (2016): 290-300. Web.
[2] Marx, Samuel E., Joe D. Luck, Roger M. Hoy, Santosh K. Pitla, Erin E. Blankenship, and Matthew J. Darr. "Validation of Machine CAN Bus J1939 Fuel Rate Accuracy Using Nebraska Tractor Test Laboratory Fuel Rate Data." Computers and Electronics in Agriculture 118 (2015): 179-85. Web.
[3] http://www.isoblue.org/
[4] http://openag.io/about-us/
[5] http://milinda.pathirage.org/kappa-architecture.com/
[6] http://kafka.apache.org/