



AN INFLUXDATA CASE STUDY

Development and Applications of Distributed IoT Sensors for Intermittent Connectivity Environments

Kevin Claytor

Research Physicist, United States Army
Research Laboratory



OCTOBER 2019 (REVISION 1)

Company in brief

The U.S. Army Combat Capabilities Development Command (DEVCOM) [Army Research Laboratory](#) (ARL) is the Army's corporate research laboratory strategically placed under the Army Futures Command. ARL is the Army's sole fundamental research laboratory focused on cutting-edge scientific discovery, technological innovation, and transition of knowledge products that offer incredible potential to improve the Army's chances of surviving and winning any future conflicts. ARL was activated in 1992, with a genealogy dating back to the early 19th century.

Case overview

In the operational environments where U.S. Soldiers operate, network connectivity is not ensured due to jamming, intermittent 4G signals, or paperwork. To address these issues, the United States Army Research Laboratory (ARL) runs [InfluxDB](#) in both the cloud and on IoT devices. When connectivity is available, the most recent data is replicated first to the cloud and historical data is replicated as bandwidth allows. This allows them to design products that can leverage the cloud, but aren't tied to it. As a result, they have been able to develop **electric power monitors for installations and microgrids**, **combine sensors into arrays**, and **strap sensors to vehicles for large area surveys**. All these solutions are built using an IoT stack fueled by InfluxDB and designed to run in environments of intermittent network connectivity. These solutions, covered below, were originally presented by ARL Research Physicist Kevin Claytor at [InfluxDays San Francisco 2019](#).

"We focus on early concept ideas to prototypes, and taking fundamental new science and new technology, transitioning that through demonstrations to partners who will then make it applicable to the warfighter. Underneath everything that we do is this idea of a transition to the warfighter, and to keep them in the front of our mind."

Kevin Claytor, Research Physicist, United States Army
Research Laboratory

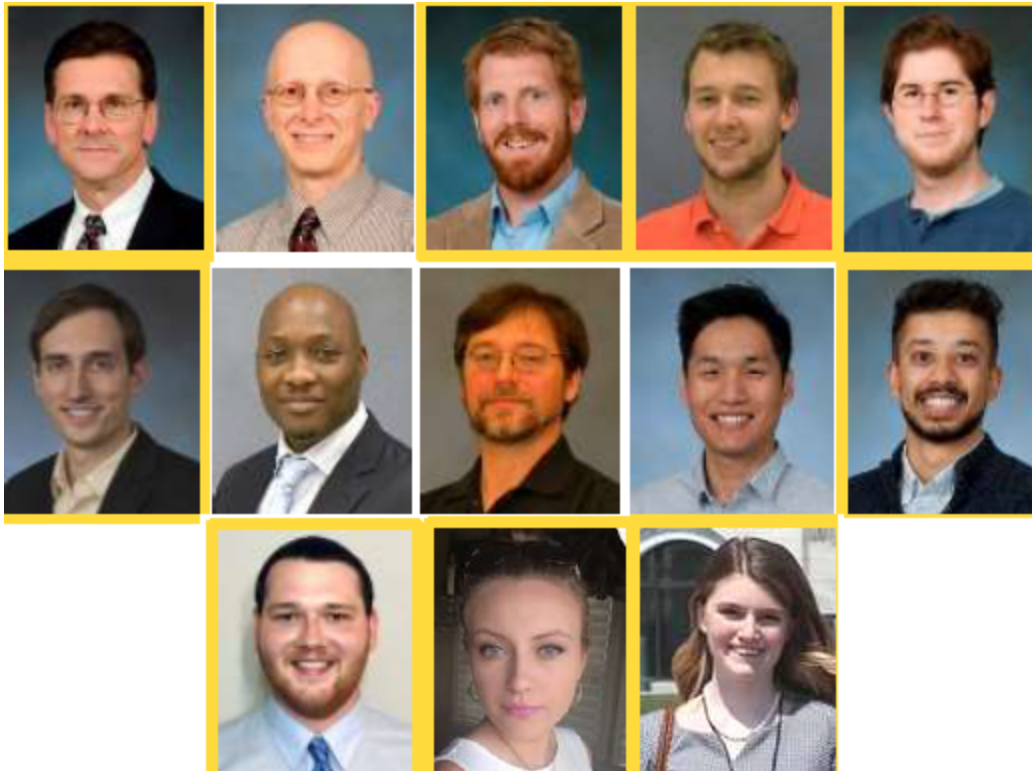
The army capabilities research challenge

Research enhances soldier decision-making. The DEVCOM Army Research Laboratory's mission is to discover, innovate, and transition science and technology to ensure dominant strategic land power.



Exterior view of the US Army Research Laboratory. Credit: Shaohan Hu. [Source](#)

As the nation's premier laboratory for land forces, the Laboratory conducts foundational research in support of U.S. Army Modernization and is focused on disruptive science and technology for the long term, performing research to answer the hardest S&T questions for future Army capabilities. This research is based on seven foundational research competencies: Ballistics Sciences, Computational Sciences, Human Sciences, Materials and Manufacturing Sciences, Network and Information Sciences, Propulsion Sciences, and Protection Sciences.



Members of the ARL team (Research Physicist Kevin Claytor, who presented this use case, is shown in the middle row to the left)

At the crux of the solutions developed by ARL is the ability to perform in the unpredictable and intermittent nature of network connectivity on the battlefield.

The Internet of Battlefield Things challenge

The Army IoT environment is not a typical IoT environment but an Internet of Battlefield Things (IoBT) environment, where the network is contested, congested and dynamic. Devices drop in and out, and you don't necessarily know if your nodes are your own or an opponent's. Actionable data is needed, and the ARL uses low-frequency electric- and magnetic-field sensing to enable data availability and rely on non-contact power monitoring.

ARL current applications areas include:

- **Mobile power meter** (has safety and cost advantages over traditional power meters and drives efficiency and resiliency, conserving fuel and maximizing its usage)
- **Environmental data** (monitoring temperature, humidity, and light to promote efficiency)
- **Sensor array** (a group of sensors, usually deployed in a certain geometry pattern, used for collecting and processing electromagnetic or acoustic signals)

- **Wide area surveys** (wide area electric & magnetic field surveys to enable disaster recovery rescue efforts)

Army Research Laboratory solutions deploying InfluxDB

“We like that the InfluxDB schema is NoSQL storage, so it's easy to get going. It's easy to just throw in a new measurement and start taking new data. You can also experiment with your schema.”

To ensure the latest points are replicated and have valid actionable data, ARL needed a scalable, purpose-built [time series database](#) with a high ingest rate. They chose InfluxDB stack for both the edge and the cloud to ensure data availability and to have a data replication scheme.

ARL Research Physicist Kevin Claytor had prior experience working with InfluxDB and found that it enables rapid prototyping and makes for great demos that you can show — and people can understand what you're doing — and that it also enables collaboration between groups across different locations.

Presented below is their approach to using InfluxDB, followed by solutions they built deploying InfluxDB in their application areas.

Common sensor processing in multiple form factors

Shown below is ARL's sensor range: Mobile Unattended Ground Sensor (MUGS), Mobile Power Meter, and Rack Mount Chassis. The framed box shows common features among these sensors.

Mobile Unattended
Ground Sensor (MUGS)



- Ruggedized for field experiments
- BNC inputs (voltage / current probes)
- LEMO inputs (TEDS compliant smart sensors)
- Integrated battery

Mobile Power Meter
(MPM)



- Designed for microgrid power cables
- Integrated noncontact field sensors
- Can be installed safely by untrained operator
- Integrated battery

Rack Mount Chassis
(Rack)



- Designed for substation installation
- Supports up to 128 inputs

- Common features
- 8-16 ksps raw data
 - FPGA / Linux hardware
 - Common processing software
 - IMU / Orientation
 - GPS / WiFi / Ethernet

As customer demands vary, ARL offers many hardware and software options and mixes-and-matches sensors, processing, and analysis to suit unique customer needs.

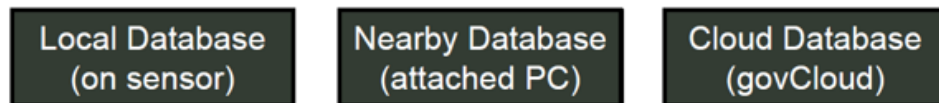
Hardware:



Processing:



Storage:



Network:



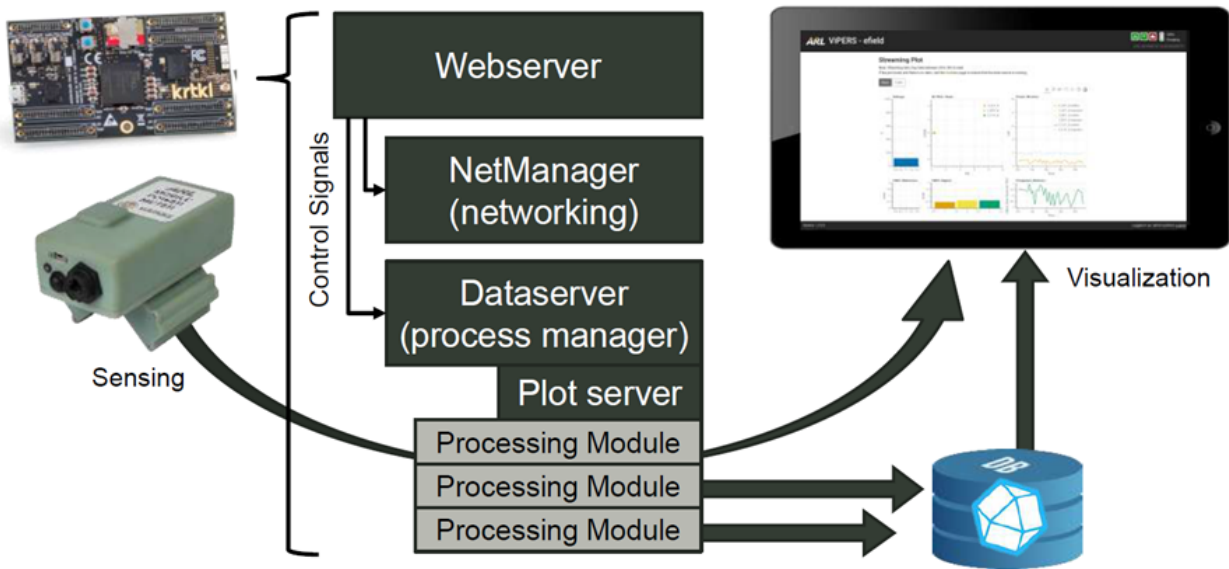
Analysis:



$5 \times 4 \times 3 \times 5 \times 4 > 1000$ combinations!

The ARL team also interfaces with legacy equipment. Their data processing and data storage can occur on the local database (sensor), on a nearby database (attached PC) or on a cloud database (govCloud). For the network, they prefer Wi-Fi for prototyping projects, but a lot of their customers prefer either Ethernet or ask not to establish a connection to their network at all. The team also uses a cellular 3G/4G network.

IoT device management – vipers (visualization and processing for embedded research systems)

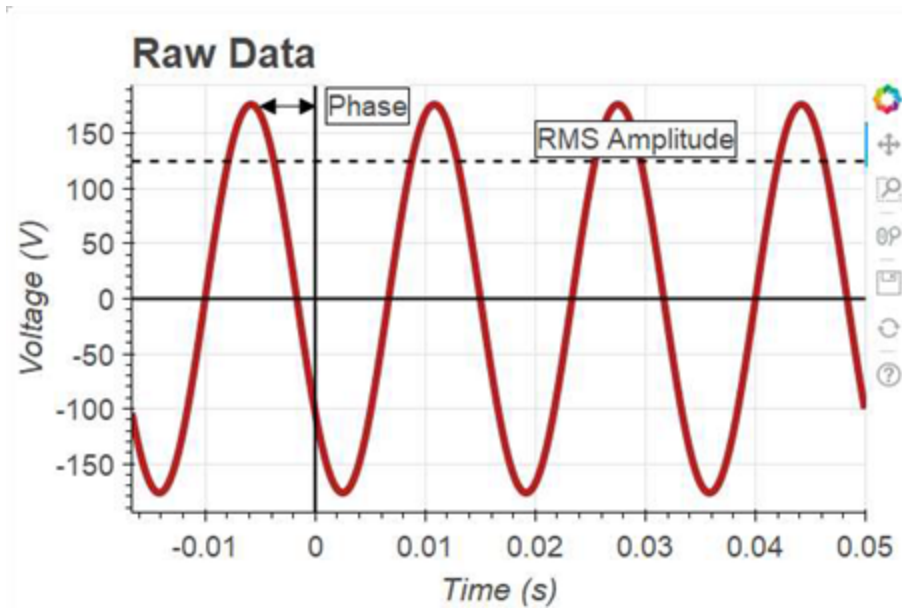


- On the top left of the above chart is the [snickerdoodle board produced by krtkl](#), which combines an ARM processor with a field-programmable gate array (FPGA). This board has two cores running at 600-800 megahertz (around half the power of a Raspberry Pi).
- On this board, they run a Webserver so that the customer can bring their own device and interface with it and configure it as they start viewing their data.
- They also run NetManager (so that they can configure networking to whatever configuration they want), and a process manager (which runs a Plot server).
- Their software sits underneath that, takes data from the sensor, runs it through a processing module, and then generates a real-time visualization.
- InfluxDB is run separately. Other processing modules take the shard data in InfluxDB, and that also supports some of the visualizations.

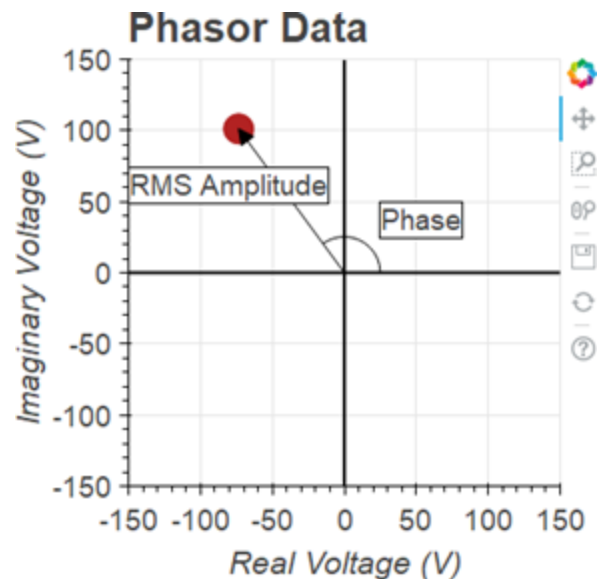
Sinusoidal power data is processed into “phasors”

The chart below shows the raw data streaming into the sensor, based on the following metrics:

- Sampled to get 8,000 – 16,000 samples per second
- (x8) 8-16 channels of data, resulting in 64,000 – 256,000 values per second



To avoid having to store all that data, they use the synchrophasor standard, which takes the A/C 60 hertz waveform, and samples it to a phasor.



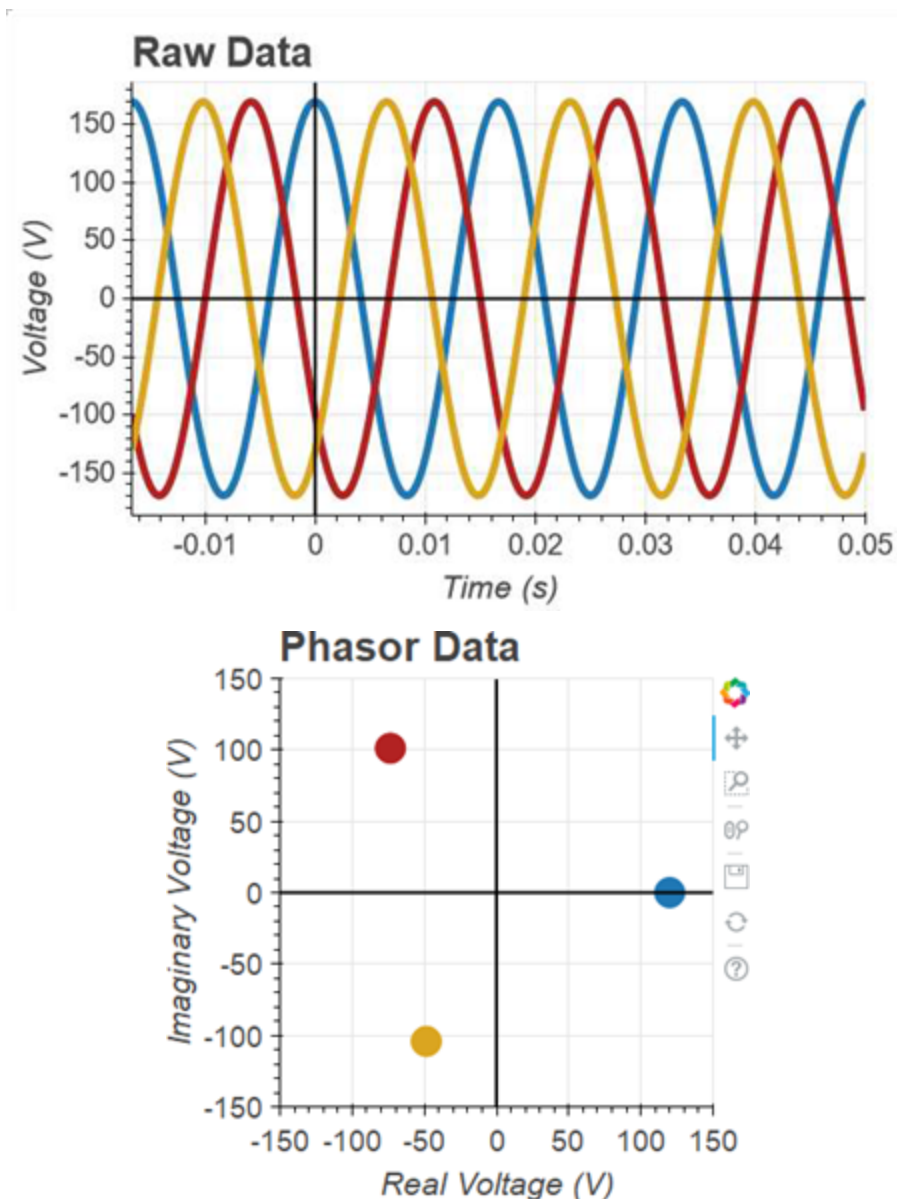
The IEEE “synchrophasor” Std. C37.118 for power systems converts one or more cycles of sine-wave data into a “phasor measurement”, reduces high data rate to a more manageable size, and loses high-frequency transients.

While phasor processing reduces sample rate, a high ingest rate is still required. They look at the **RMS magnitude** and at the **phase offset** between that peak and some timestamp (GPS time in their case). Those two measurements then map to the phasor where there is an “Amplitude”, which is then offset from the X-axis as a “Phase”.

Phasor data is therefore as follows:

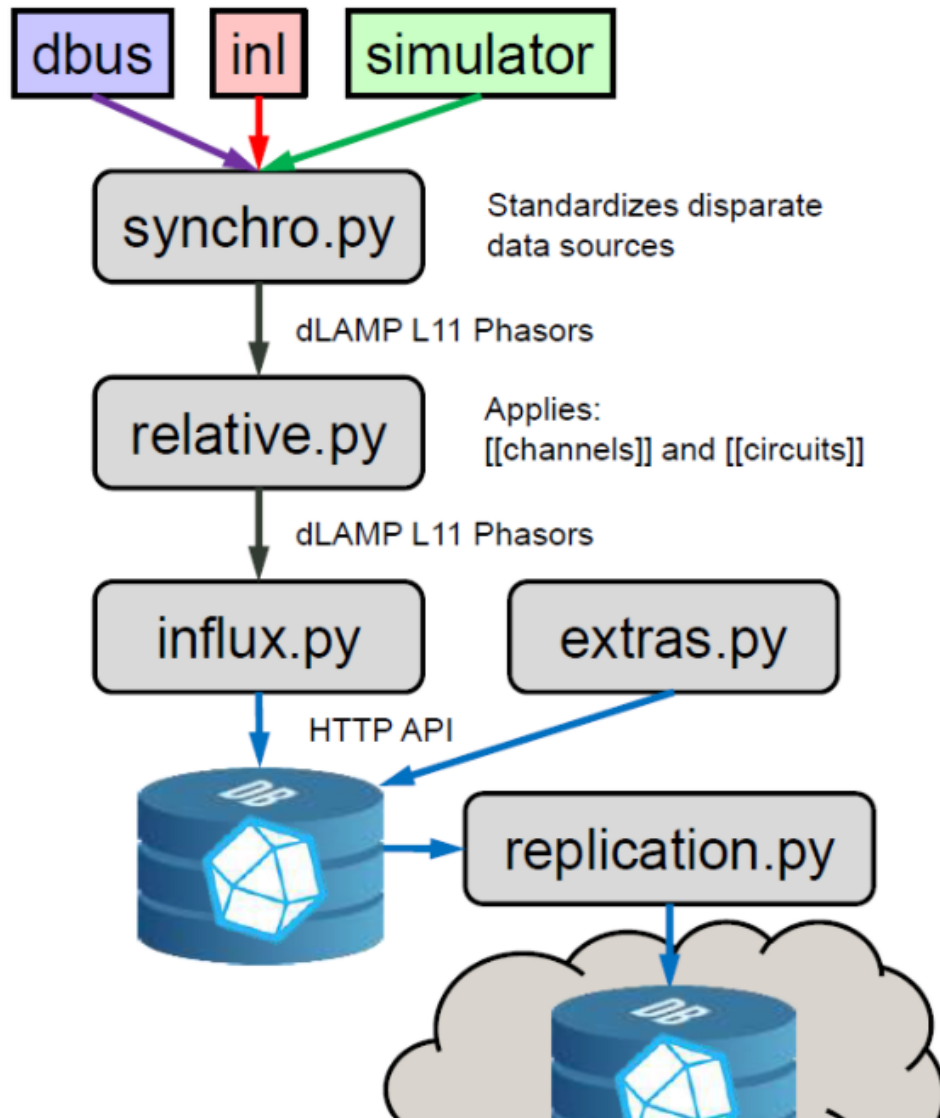
- Estimators produce data from 10-60 Hertz of 8-16 channels of data.
- The fundamental 60 hertz data is monitored, and so are two harmonics, the frequency deviation from 60 hertz, and total harmonic distortion.
- The resulting values are complex, so twice as much data has to be stored (around 1,000-10,000 samples per second into the database; i.e. 1,000 to 10,000 hertz of raw data), and it has to be stored at the edge. The data is time-stamped when it comes in, batched up, and inserted approximately every second.

Once the data is put into a power system, it maps as shown below. The graph shows three-phase power, which means voltages are offset by about 120 degrees. The same can be applied for the currents as well.



Python modules

Shown below is ARL's processing stack.



The data flow is as follows:

- They send real or simulated data to dLAMP packets, which standardize it and make it available to their historical architecture and processing system.
- After a little more processing, they use a Python module to bring that data via the HTTP API into InfluxDB.
- Then they run additional processes, such as for the IMU, GPS, and battery, that send more data into InfluxDB.
- Then, when there's a network available, they can replicate that data to the cloud.

Their processing stack has the following features:

- **Well defined API's** - Others can write software to their module APIs
- **Distribute as components** - Leave out unnecessary components
- **Smaller updates** - Update only the modules needed
- **Performance versions** - Rewrite single component for performance instead of entire stack
- **Can swap components based on license** - Exchange databases

Database schema considerations

InfluxDB has dynamic schemas that don't have to be defined up-front. This made it easy to get started. After experimenting with five different schema configurations, they found a schema that seemed very intuitive for power data.

InfluxDB does enforce type consistency. This ensures that numeric data stays numeric. Because the initial commit of an int (eg; 0) can prevent later floats from being committed (eg; 1.023), they settled on the compromise of explicitly casting values before writing them (eg; float(0)).

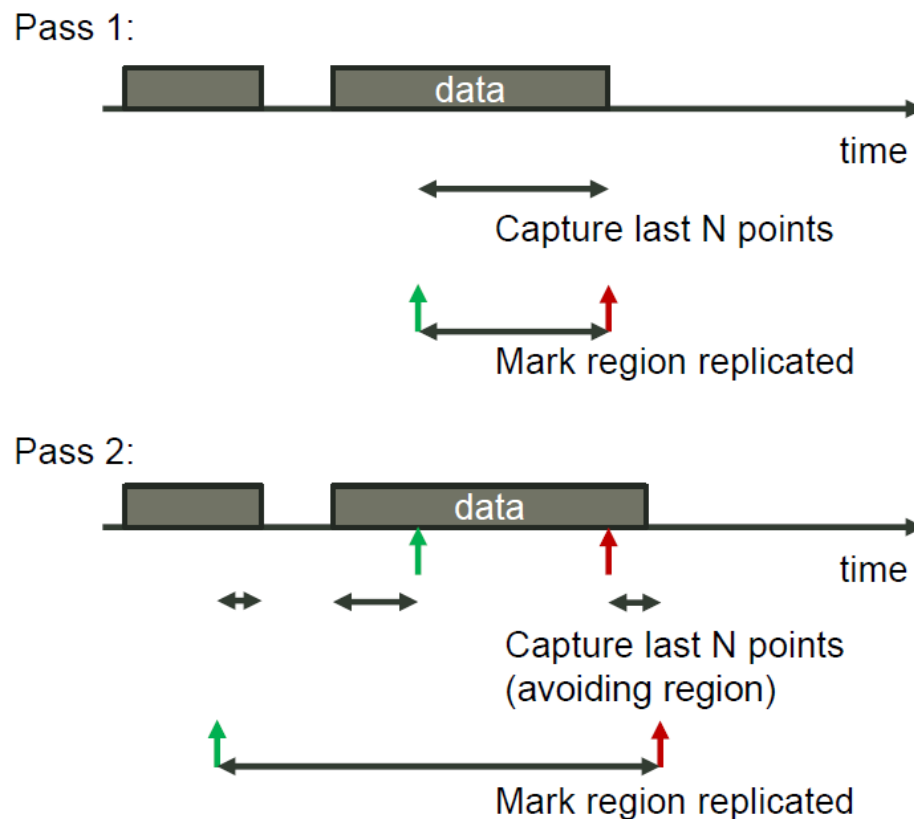
- They use a **local schema** on the Mobile Unattended Ground Sensor (MUGS).
- They use the **cloud schema** on influx.arlpower.net, and use databases for function / assets. The cloud schema allows for transition of assets in locations and allows access control by asset.
- **Database schemas** are identical: cloud queries can be used in MUGS, and MUGS queries can be used in the cloud. (To duplicate the replication schema identically on both the edge device and cloud, they had to develop a schema that works for either of them and import the processing software to the other location.)

They run [Continuous Queries \(CQ\)](#) to downsample for long-duration data, and they backup data to RAID, a data storage virtualization technology that combines multiple physical disk drive components into one or more logical units.

Replication - getting the latest points while avoiding previously replicated regions

Shown below is their replication schema (in Pass 1 & 2, and in Pass 3). It's a separate module that queries the database for particular measurements. It captures the latest N points, and then marks that region in a new measurement, as a measurement that's over a specific time period. When it does its

next pass, to avoid redoing that, it grabs the latest data again, goes further back in time, and then marks this new region as replicated. Now there are two regions marked as replicated.



This process can be summarized as follows:

replication.py

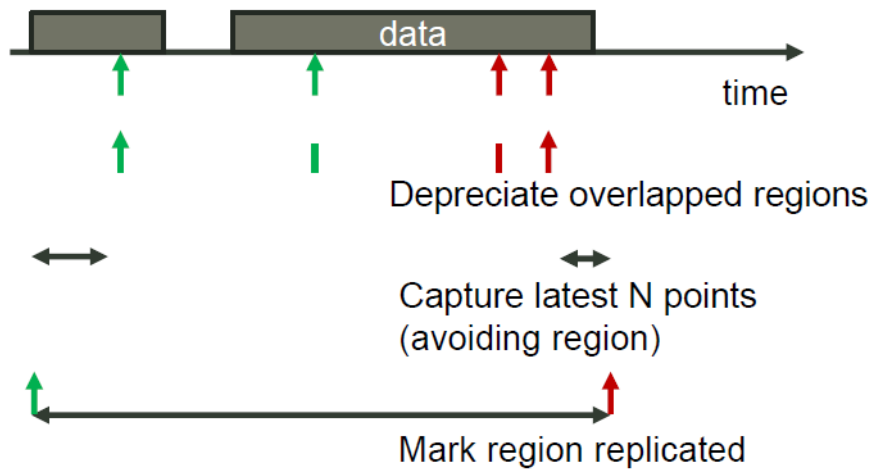
- Clones local data to the cloud
- Logs if data replication is successful
- Retries failed replications

Algorithm

- Query for previously replicated regions
- Query for last N points avoiding these regions
- Attempt to replicate
- If successful, mark a new replication region

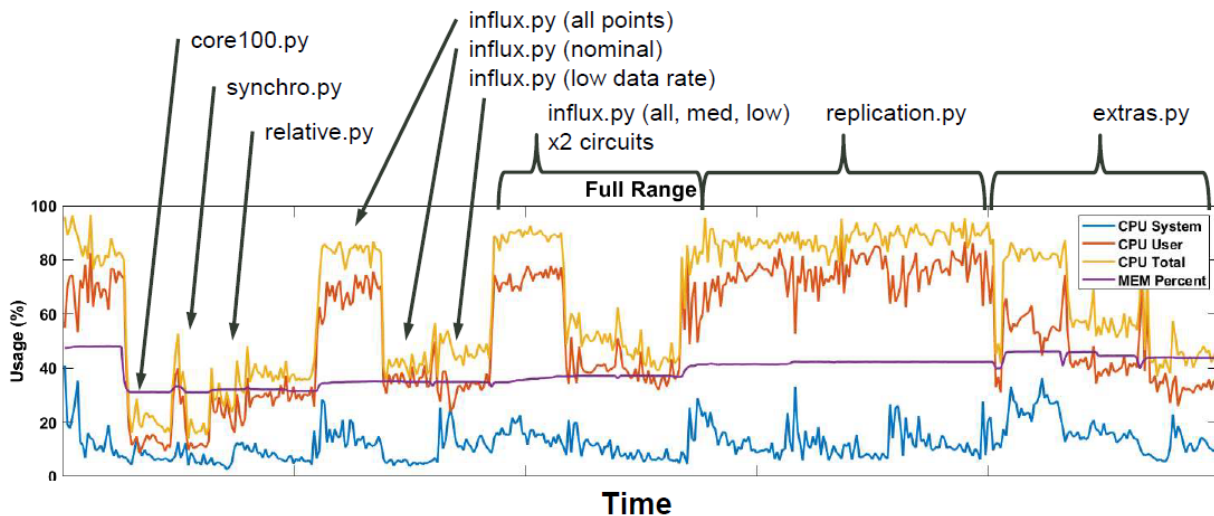
When they first query the replicated regions, they modify one of their tags to depreciate one of them, as shown below in Pass 3. Then they capture, again, the latest N points. This way, they can rapidly go through and access all the historical data, while ensuring that their most recent data is always being pushed. Pass 3 is where they merge overlapping replication regions (set field value = 0).

Pass 3:



Performance – easy to quantify with Telegraf

Since their system is small and every CPU cycle is valuable, they use Telegraf as a performance tester. The below chart shows components of their processing stack booting up, and the Influx inserter running at different settings.



To get the full phasor rate data, they needed to reduce CPU usage. Total CPU consumption includes total system usage, InfluxDB, Telegraf, etc.

They did performance testing on MUGS v2 (1st generation), with many modules written with adjustable inputs (eg; data rate limiting), and were able to highlight key bottlenecks.

Performance – quantifying allows improvements

By quantifying performance, they were able to improve it. The graph below shows CPU consumption for two different approaches:

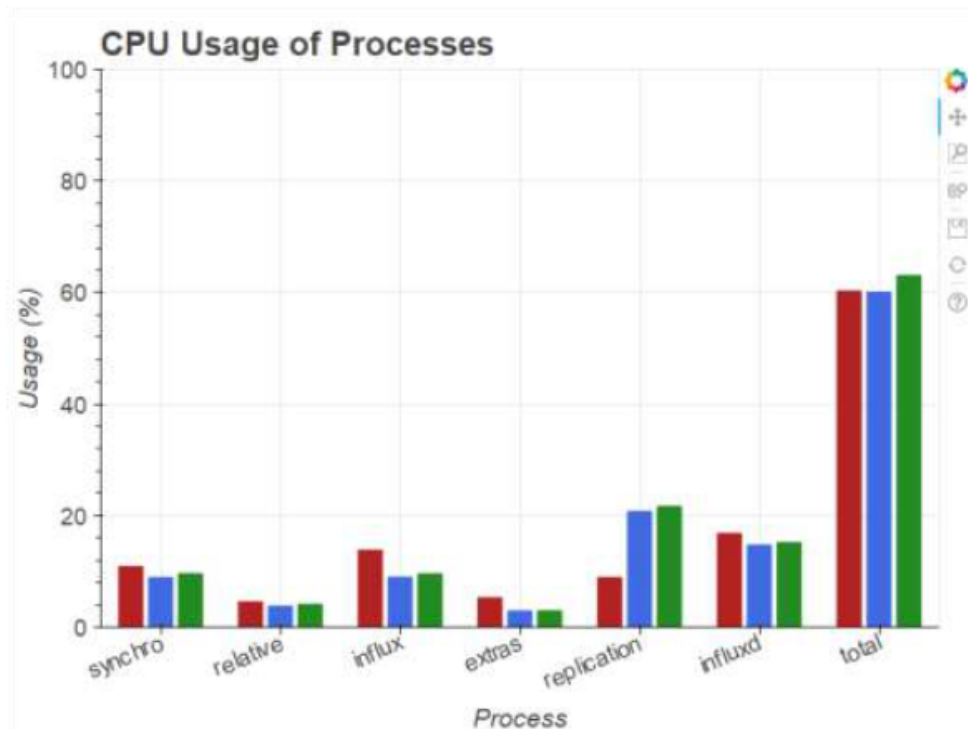
- The red is the initial approach where they were just collecting data for all the series, then sending that to InfluxDB all at once and asking the database to parse the data and put it into the individual series.
- The green is where the series are tracked individually, and then bulk writes are done by series into the database. That approach dramatically improved performance in the CPU.



With insight gained into bulk CPU usage, they could discover whether their processes, or InfluxDB, is using all the CPU. This is shown in the CPU Usage of Processes graph below.

- Red represents the old schema way of doing things. It involved an algorithm that was opaque and difficult to maintain.
- Blue is the new schema way. The new algorithm uses more CPU but accesses the data faster and more effectively.

For each insert by series, they were able to save approximately 30% CPU usage (this was possibly related to lower overhead in the Python client). The difference between the blue and the green bars below is that compression is enabled, which is helpful since they are at the edge.

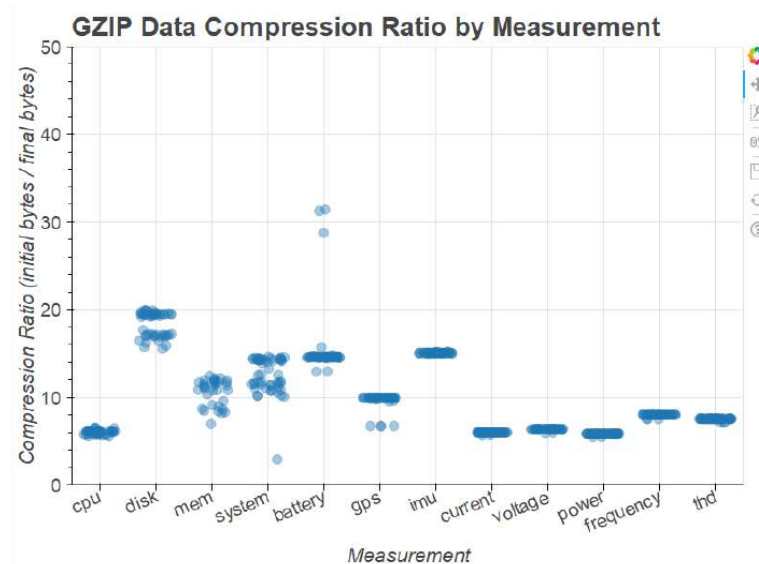


Open source contribution – gzip feature

The below script was ARL Research Physicist Kevin Claytor's first open source contribution to the Python InfluxDB library and shows GZIP compression on data through repeated tag values.

```
if self._gzip:
    # Allow us to receive gzip'd data
    # as well as write it out
    headers.update({
        'Accept-Encoding': 'gzip',
        'Content-Encoding': 'gzip',
    })
    if data is not None:
        # For Py 2.7 compatability use Gzipfile
        compressed = io.BytesIO()
        with gzip.GzipFile(
            compresslevel=9,
            fileobj=compressed,
            mode='w'
        ) as f:
            f.write(data)
        data = compressed.getvalue()
```

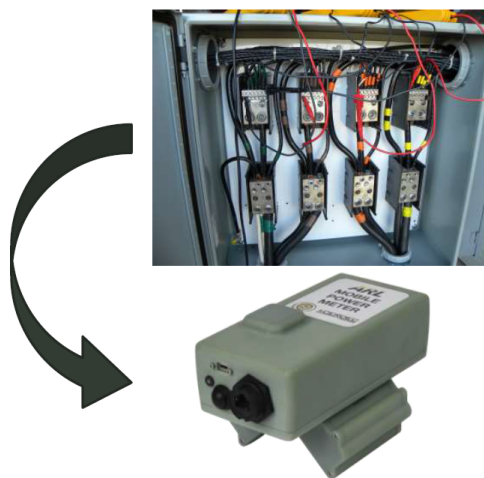
The below graph shows measurements going from CPU, disk, memory, system and battery and also shows phasor measurements.



Some measurements compress well, like disk, due to the slow uptick in data. Line protocol lends itself to compression very well, since there are repeated tag and field measurements. The compression values worked for ARL, especially since they were tracking whether they were going over their data plan limit for their 4G network.

Mobile power meter

Shown below is the ARL mobile power meter, which replaces the traditional power box, and how they use it to solve the power availability problem in an IoT environment.



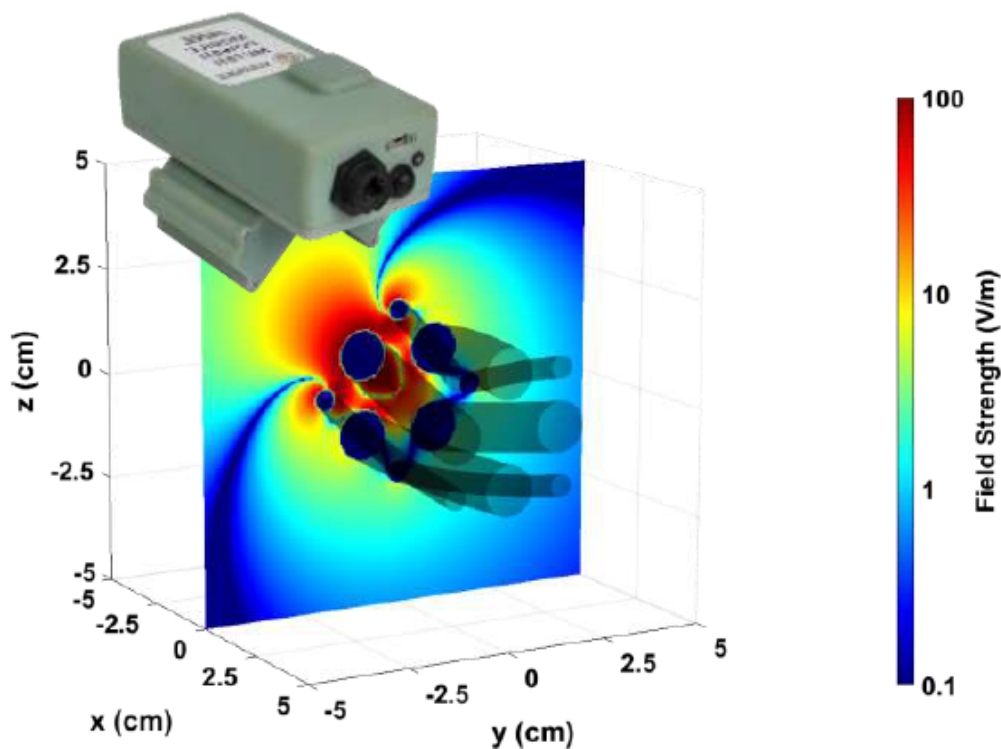
Problem: Power is a resource, and assured power needs measurement. An electrician with an hourly

cost would be needed on-site with a measurement device. The electrician, to do this safely, would have to interrupt power to the facility, which can be very costly if that facility is a command control or hospital. In that case, the electrician might be operating on a live system, putting his health and safety at risk.

The solution they found is to monitor the electromagnetic fields. Electric fields are created by differences in voltage: the higher the voltage, the stronger the resultant field. Magnetic fields are created when electric current flows: the greater the current, the stronger the magnetic field. Their monitoring solution measures both the voltage and the current, allowing them to calibrate as appropriate. This minimizes power interruption and drives down the main cost driver to zero. Since power monitoring enables usage optimization, the solution results in informed energy consumption, and saves energy, fuel, and lives (eliminating the need for the risky prospect of trucking to operating bases).

Mobile power meter – adding orientation data

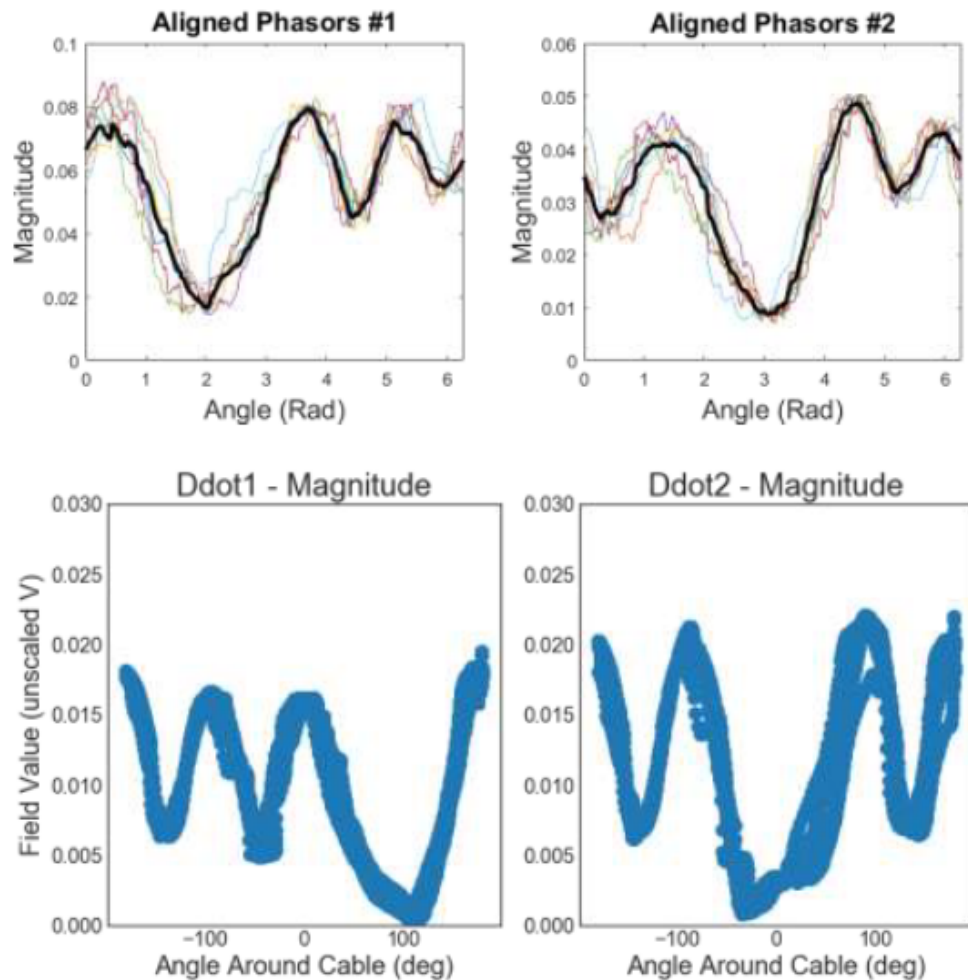
Calibration is dependent on the location around the cable. Below is a 3D computer simulation of a microgrid cable that the mobile power meter sits on top of. The electric field shows around one of the conductors.



As you go around that conductor, that electric field changes, requiring recalibration. In this case, they're starting to insert inertial measurement unit (IMU) data, so that if it gets kicked or moved or a truck rolls over it, it can now alert them, or ideally alert an automated system, that it needs to be recalibrated.

Adding inertial measurement unit (IMU) data to the database has been advantageous:

- They can now identify if the sensor has moved and needs recalibration.
- Measurements of the electric field are a function of the angle around the cable.
- They previously had to estimate cable angle assuming "constant rotation rate" (resulting in the jagged lines below that don't line up on each other).



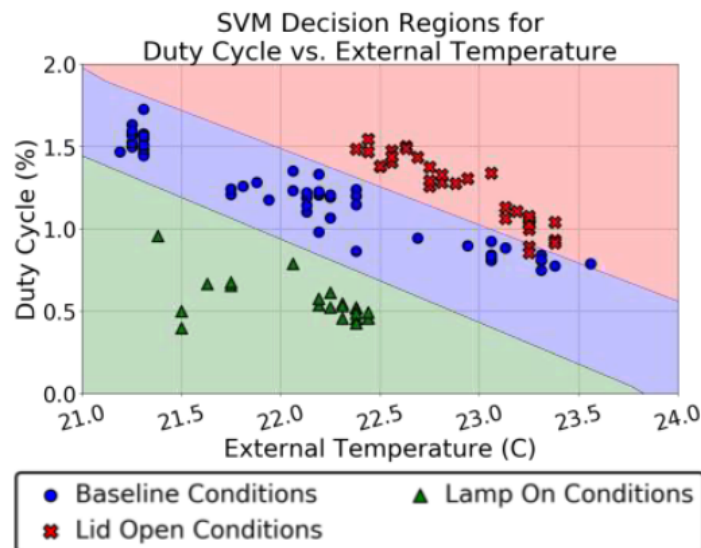
With the IMU data in the database, they now know exactly what the orientation of the device is, and can get much more reproducible measurements.

Environmental sensing to promote conservation and efficient buildings

Shown below is an environmental sensing project by a summer student at ARL. The project aimed at promoting power consumption efficiency by applying machine learning using the building's power monitoring data. Efficiency insights gained inform building renovations by identifying buildings that yield the highest ROI from renovations.

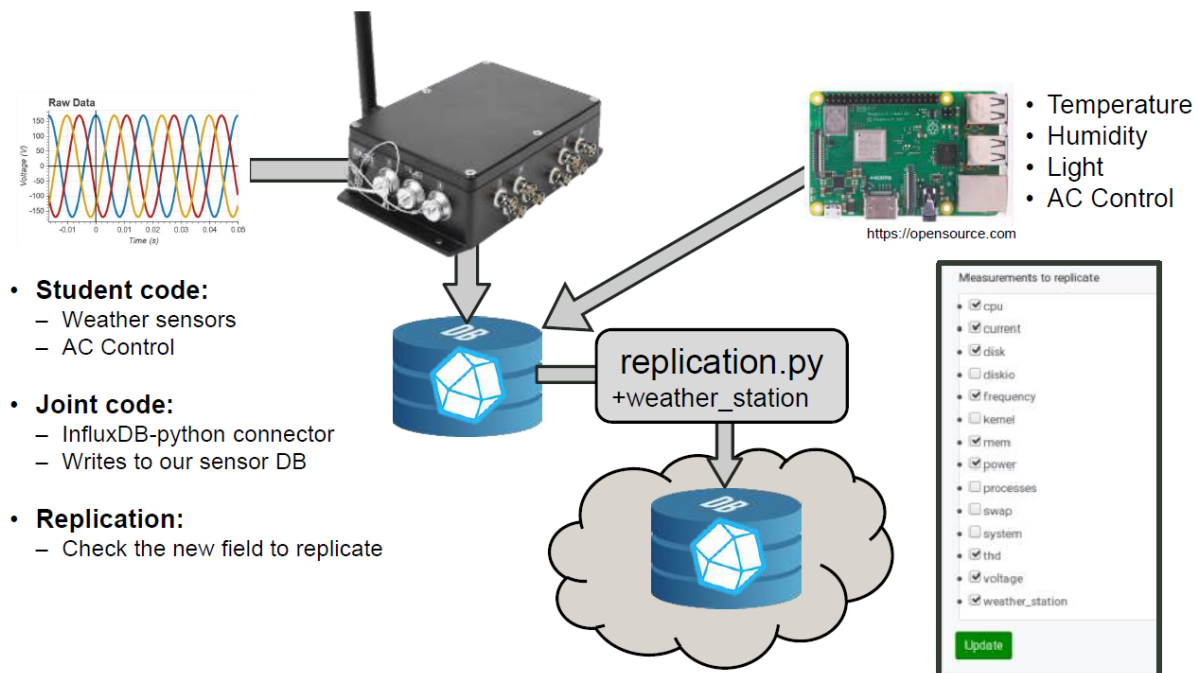


As a proof of concept, the student set up a light on a small box with a space heater inside it, and then ran a few different test conditions. The graph below maps the heater's duty cycle versus external temperature. A machine learning classifier was used to classify the conditions as Lid Open, Baseline, or Lamp On. The results were encouraging, and it was decided to scale up the project.



Supplemental measurements are easy to replicate

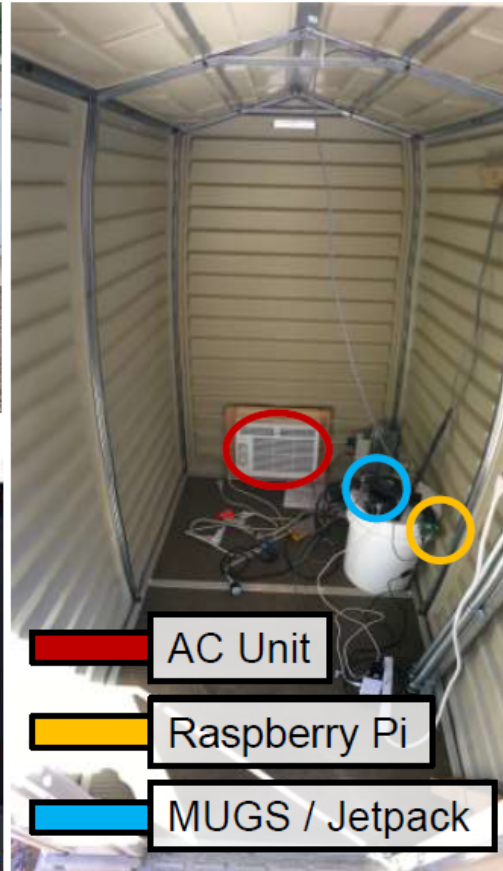
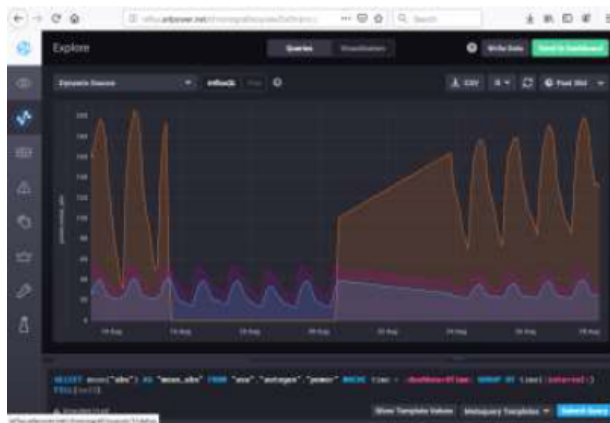
In the next instance, they took their sensory unit, which is monitoring power, and hooked up a Raspberry Pi to an A/C control. Now they had full control over the A/C unit, and can run it at whatever duty cycles or intervals they want. They also log temperature, humidity, and light — all set up on the same wireless network. They send data to their instance of InfluxDB. Then from their UI, they can decide to replicate the new measurement, and the replication script immediately starts replicating that measurement to the cloud.



Version 2 of this experiment is shown below. The A/C unit and Raspberry Pi are in a shed in Arizona. The Raspberry Pi sends data via Jetpack up to the cloud. The Raspberry Pi pushes data to the ARL sensor over the local network, and the ARL sensor replicates data into the cloud. The data can be seen in the Chronograf instance as shown below, which is great for collaboration, whereby the same data can be pulled up and the same events viewed no matter the viewers' location.



Credit: Drummond, Zachary

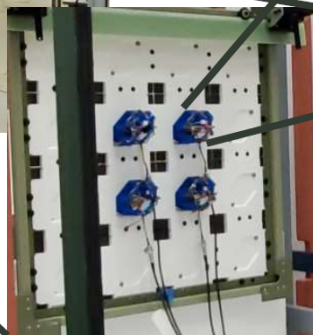


- AC Unit
- Raspberry Pi
- MUGS / Jetpack

Credit: Drummond, Zachary

Magnetic field sensor array – 3 axes sensors

Below is another exciting project: an imaging array. This consists of a magnetic field cage which creates a very uniform region of magnetic field. It's the large regions in which they can put entire sensors or entire platforms, like quadcopters. But then they wanted to image with these magnetic or electric fields and use a 4x4 array to do so, as shown below.



- **4D Sensor “pixel”**
 - 3-Axis H-field sensor
 - 1-Axis E-field sensor
 - 2x pixels per processing board

They are expanding it, and it consists of 4D sensor “pixels”. Each pixel has a 3-axis magnetic field sensor (so it’s measuring X, Y, and Z in the magnetic field), and has a single-axis electric field sensor.

They also have an analog for the electric field. Shown above, to the left, is the electric field sensor array sitting in their electric field cage. The cage is designed to have a large uniform volume of electric field in the sensor module sitting behind it.

Full 6x8 array with border elements

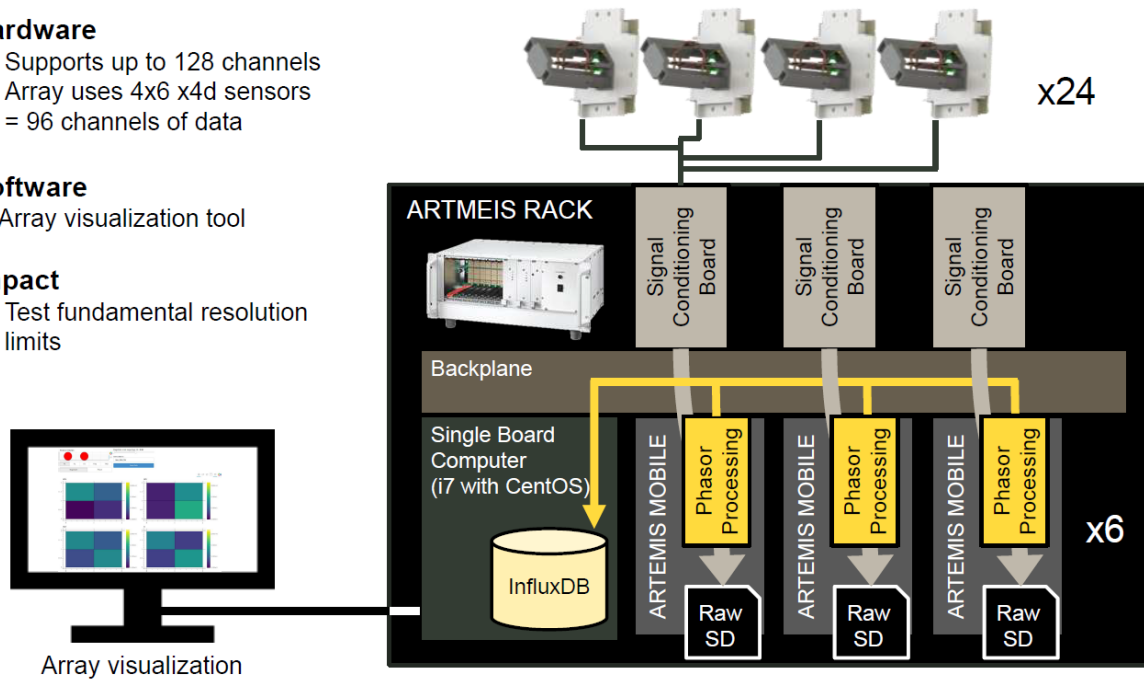
Below is a 6x8 array, and they have an edge array of pixels to absorb the electric flux, as it goes around the array.



Imaging array: a local area “cloud”

The data processing is shown below.

- **Hardware**
 - Supports up to 128 channels
 - Array uses 4x6 x4d sensors = 96 channels of data
- **Software**
 - Array visualization tool
- **Impact**
 - Test fundamental resolution limits

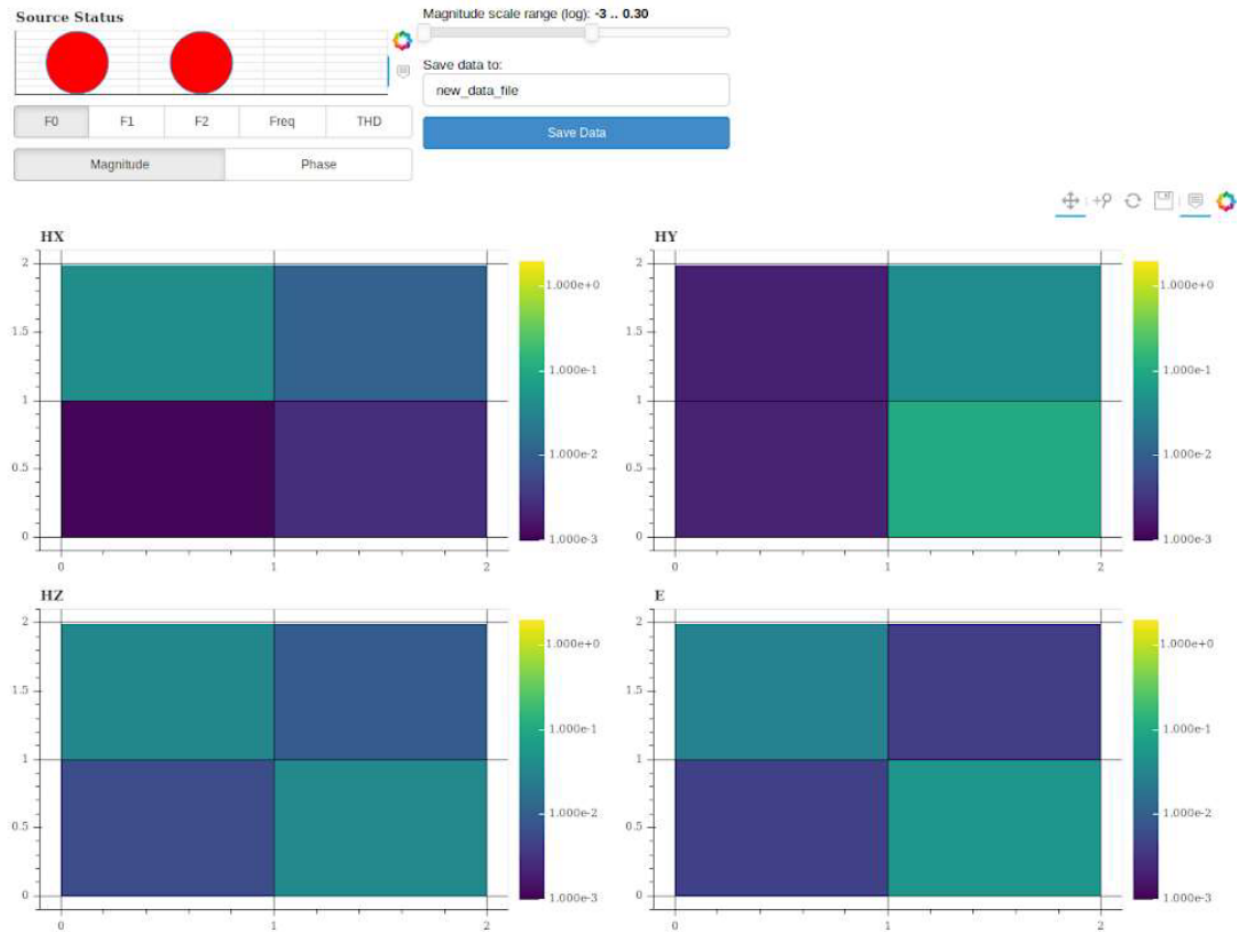


The data from the pixels goes through a conditioning board, and then the standard phasor processing is done. The data is sent into an Intel i7 Single Board Computer, which sends it to the database where the data can be queried for visualizations.

Streaming array visualization

Their visualizer (Array Designer + Streamer), which was running the above-mentioned 4x4 array, is shown below. The visualizer:

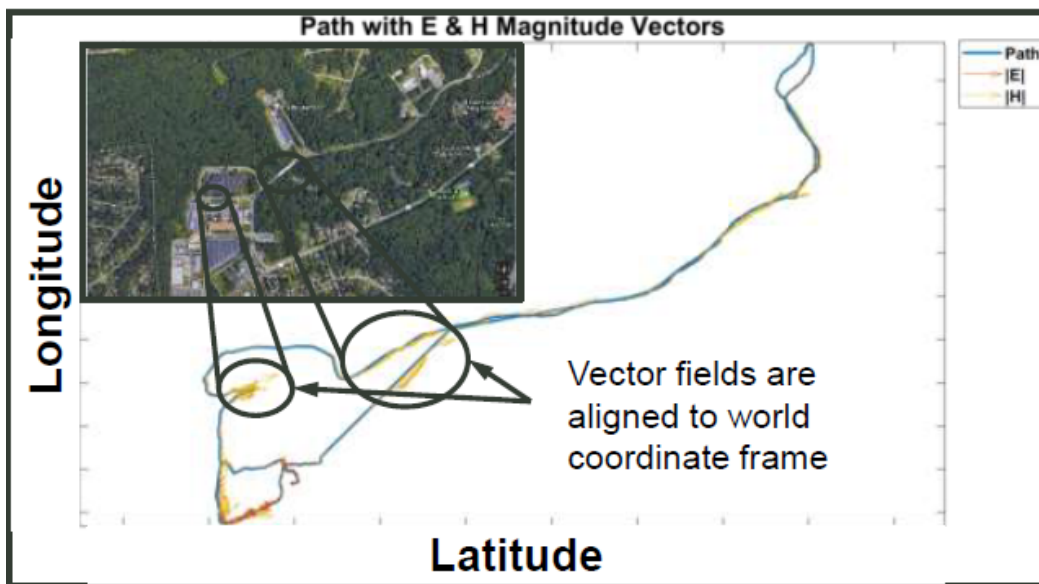
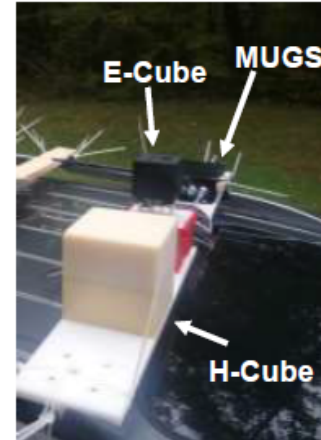
- Allows the design of NxM arbitrary rays. Arrays can be sparse (saves data efficiently).
- Does live visualizations of all the parameters, including both magnitude and phase.
 - Select between harmonics, magnitude or phase
 - Dynamically connects (handles disconnects, etc.)
 - Dynamically adjust scaling



While they're sending all the data to InfluxDB, it was easier to reshape the data packet than to manipulate query results. But they may need a more accurate solution for higher frequency, phase-sensitive measurements, and may need to have their data time-stamped in the database accurately, to interpolate between times so that all the phases line up.

Wide area electric & magnetic field surveys

Shown below is another interesting experiment. They took one of the 3-axis magnetic field sensors, added a 3-axis electric field sensor, strapped it to the top of a car with their sensor processing unit, and drove it around. This allowed them to measure power signals as they drove.



The hardware used in this experiment was:

- 3-Axis E-field sensor
- 3-Axis H-field sensor
- 6-Axis Inertial measurement unit (IMU)
- Integrated GPS

By having the IMU, they were able to orient the power signals to world coordinates. While the car might have been moving left and right, up and down, they can transform those coordinates so that they're all the same coordinates, and then they can georegister them. As they drove around the installation, their vector fields can be seen as they were driving along the world coordinates.

The software used in this experiment involved:

- Measuring 60-Hz signals
- Using IMU to align to world-coordinates
- Using GPS to place on map

They are planning on proposing this project for disaster recovery in Puerto Rico or the Bahamas where this setup can be mounted on a quadcopter, flown over an area, and used to survey power infrastructure damage and thereby determine where to concentrate relief efforts. They are also using this project for background measurements, which facilitate sensor placement for high-precision measurements. They have taken a survey of a field site, and are georegistering those coordinates so they can find the area with the lowest interference noise, and then set up their experiments there.

Results

"I like InfluxDB because it allows us to do a job more effectively, and this is reducing time to awesome. So we can come up with a concept for an experiment, start shoving data into it, and then query that back out, and do the data processing fairly easily. The visualization also helps for demos, and generally just helping to quantify things makes it so you can optimize them."

InfluxDB platform has helped ARL accelerate technology development, demonstrations and transitions:

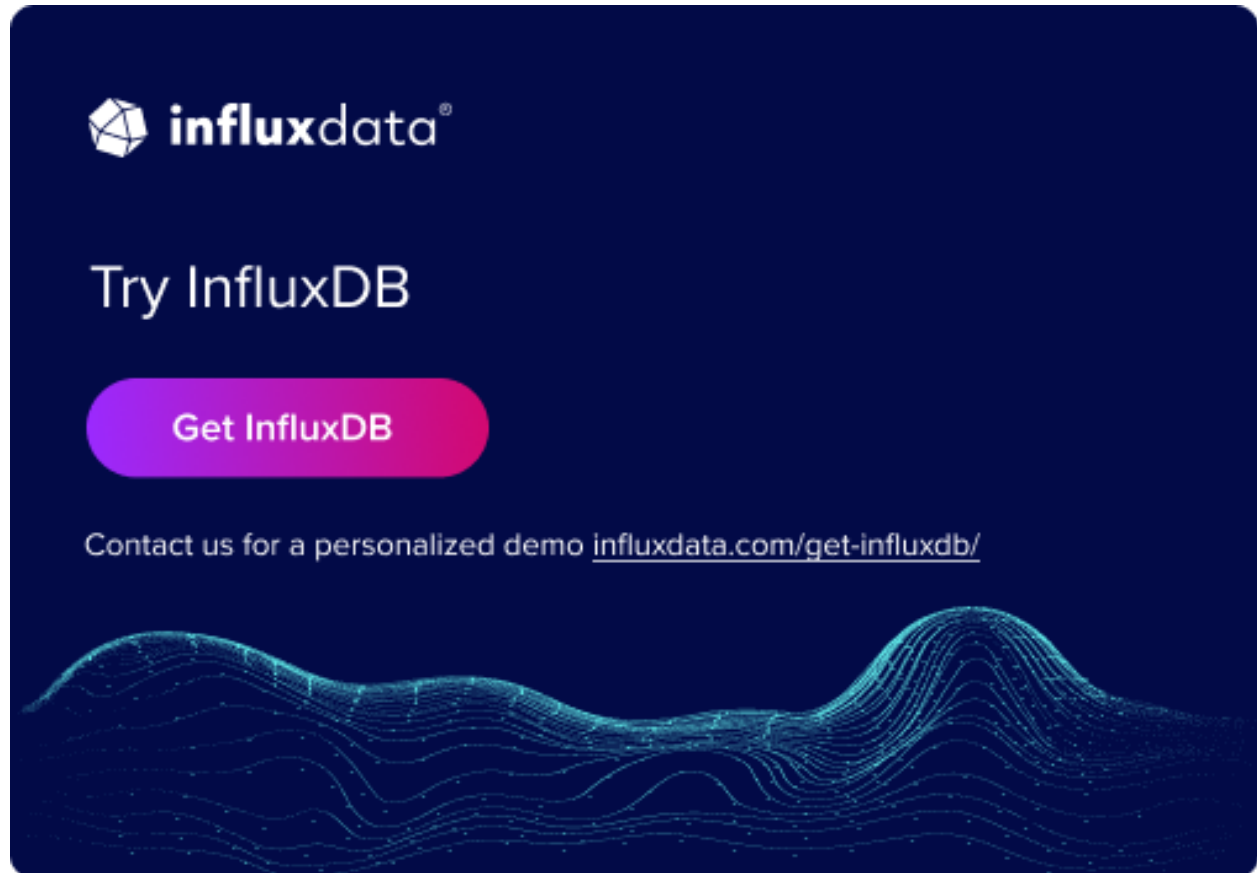
- **InfluxDB** makes it easy to insert new measurements
- **Chronograf** in the cloud is a convenient collaboration tool. With Chronograf as its native visualization engine, InfluxDB provided a short turnaround time for new demos.
- **Telegraf** helps identify resource bottlenecks and optimize code, and the InfluxDB platform overall helps to quantify things.
- **Kapacitor** alerts provide actionable information that inform transitions to the warfighter.


What's next for the Army Research Laboratory?

ARL Research Physicist Kevin Claytor is looking forward to using more Kapacitor alerts, especially for actualizing the transitions to the warfighter, since warfighters don't care about metrics but rather about "What should I do?" The alerts indicate: "This is what you need to do".

About InfluxData

InfluxData is the creator of InfluxDB, the leading time series platform. We empower developers and organizations, such as Cisco, IBM, Lego, Siemens, and Tesla, to build transformative IoT, analytics and monitoring applications. Our technology is purpose-built to handle the massive volumes of time-stamped data produced by sensors, applications and computer infrastructure. Easy to start and scale, InfluxDB gives developers time to focus on the features and functionalities that give their apps a competitive edge. InfluxData is headquartered in San Francisco, with a workforce distributed throughout the U.S. and across Europe. For more information, visit influxdata.com and follow us [@InfluxDB](https://twitter.com/InfluxDB).

A promotional banner for InfluxData with a dark blue background. At the top left is the InfluxData logo, which consists of a white geometric cube icon followed by the text 'influxdata' in white. Below the logo, the text 'Try InfluxDB' is written in a large, white, sans-serif font. Underneath this is a prominent pink button with rounded corners containing the white text 'Get InfluxDB'. Below the button, the text 'Contact us for a personalized demo' is followed by the URL 'influxdata.com/get-influxdb/' which is underlined. The bottom of the banner features a decorative graphic of glowing blue and white wavy lines, resembling a data visualization or a stylized horizon.

 **influxdata**[®]

Try InfluxDB

[Get InfluxDB](#)

Contact us for a personalized demo influxdata.com/get-influxdb/