DMDII

a UI LABS Collaboration

FINAL PROJECT REPORT

Project Team Lead	**Georgia Tech Research Coproration**
Project Title	**Reconfigurable Retrofit Kit for Legacy Machine Sensing in Secured Data Environments**
Project Designation	**DMDII-16-02-03**
UI LABS Contract Number	**0320170002**
Project Participants	**ITAMCO, Mazak Corporation, Caterpillar Inc.**
DMDII Funding Value	**$274,673.00**
Project Team Cost Share	**$603,472.00**
Award Date	**February 21, 2017**
Completion Date	**July 31, 2018**

SPONSORSHIP DISCLAIMER STATEMENT: This project was completed under the Cooperative Agreement W31P4Q-14-2-0001, between U.S. Army - Army Contracting Command - Redstone and UI LABS on behalf of the Digital Manufacturing and Design Innovation Institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Army.

I. Introduction

High Level Problem

Legacy equipment platforms represent a critical element of the manufacturing ecosystem, especially for small to medium manufacturing organizations. While state-of-the-art equipment platforms offer capabilities for advanced connectivity and in situ machine sensing, strong advancements are needed to provide bridge or transitional capability for data sensing on legacy equipment platforms that are difficult to replace for many smaller organizations or severely cost-constrained industries. Due to the relatively lower cost of these legacy platforms, primary challenges are found in enabling such retrofit capabilities at low cost and at a high degree of implementation flexibility. Unfortunately, conventional hardware approaches to retrofit systems for such legacy platforms have missed the mark on addressing either or both of these challenges to enable widespread adoption of retrofit capabilities.

High Level Purpose

A new model for a retrofit system must be developed that enables ease of reconfiguration and customer-driven selection of sensing priorities. To address this challenge, this project assembled a strong multi-disciplinary team including Georgia Tech, Mazak, Cisco Systems, ITAMCO and Caterpillar. The team has expertise in the requisite technological areas that include manufacturing sensing and analytics, machine tool manufacture, and platforms for industrial network security and wireless communications. The team proposed development of a reconfigurable retrofit kit (RRK) for legacy equipment platforms that provides for low per machine implementation cost (less than $1000), flexibility in sensor package implementation and state-of-the-art network security.

Summary of Findings

This work formulated and demonstrated the ability to rapidly develop and deploy an open architecture process monitoring system using off-the-shelf and open technology (OTSOT). This final RRK is comprised of a centralized communications platform for integrating the shop floor networking infrastructure or cloud-based web services to a mixed set of legacy and MTConnect-compatible machine tools, this linked to individual sensor packages through wireless communications hardware. The central communications hub exceeds industry standard requirements for network security and system isolation. The wireless communications protocol and remote plug-and-play sensor packages provide for low-cost (less than $1000) and significant flexibility in data sensing implementation redesign as the shop floor evolves with plant/cell reorganization, changing product mix and different needs for in process sensing. Further, this reconfigurable sensor package network provides for true scalability of cost of ownership for enabling data sensing across the shop floor and enables customer-driven selection and implementation of sensing capabilities. While this platform is built upon open and commercially available tools, it will inform the design of similar other centralized and secure data sensing networks for manufacturing environments. To facilitate testing of the system and inform design of future systems, implementation was executed in testbeds at multiple partner sites and at DMDII to demonstrate the flexibility, interconnectivity and interoperability of the proposed retrofit approach.

Summary of Recommendations

The current project effort advanced development of a sensor retrofit solution that is scalable within small and large shop environments and is flexible in terms of the hardware/software implementation. The capabilities of the implemented solution will be attractive for technology adopters. In considering adoption of this technology, the low-cost IOT devices central to the RRK sensor packages may require integration with existing enterprise networking infrastructure. It is recommended that adopter facilities review compatibility of these devices with existing site-specific network security policies. Additionally, the technology space surrounding low-cost IOT devices is characterized by rapid pace of development, adopters should pay attention to developments as new capabilities may offer significant advantages over the current development.

II. Project Review

A. Project Scope and Objectives

The project technology effort included the team's development of a machine retrofit kit for enabling highly secure wired and wireless communications across a range of legacy and modern machine tool equipment. Low cost sensors were implemented on a flexible retrofit platform based on industrially hardened communications hardware. System functionality was tested in various industrial environments as part of the technology effort. The primary objective of the project was the development of a reconfigurable retrofit kit for legacy equipment platforms that provides for low per machine implementation cost (less than $1000), significant flexibility in sensor package implementation and state-of-the-art network security. Supporting objectives were to establish the communications infrastructure, implement a range of low cost sensors, and establish/demonstrate the retrofit kit's performance in industrial environments.

B. Technical Approach and Planned Benefits

The project work approach was divided into two efforts pertaining to: (1) establishing the wired/wireless system communications infrastructure and (2) implementing a suite of wireless sensor packages for legacy machine tools. The resultant RRK was implemented at 2 partner manufacturing facilities across a range of legacy and modern machine equipment platforms to demonstrate technological capabilities. These technical approach is described below:

Wired/Wireless System Communications Infrastructure

To achieve a reconfigurable, highly flexible sensing architecture for the RRK, we implemented a flexible wired/wireless network for forward/backwards-compatible communications integration across all legacy and modern machine platforms. While multiple configurations of the envisioned hardware package are possible, the lowest cost implementation is of primary interest to SMMs. In a low-cost configuration, the retrofit kit is built upon a central communications hub providing a secure link to enterprise IT infrastructure. The central communications hub is able to connect multiple MTConnect-compatible machines via hardline connections. To facilitate connections for legacy machines, wireless sensor packages were developed for data transfer to the communications hub. In this regard, implementation for legacy machines is limited solely to sensors and wireless

communications devices needed for those applications, providing a scalable solution for integrating legacy machines within an IIOT framework.

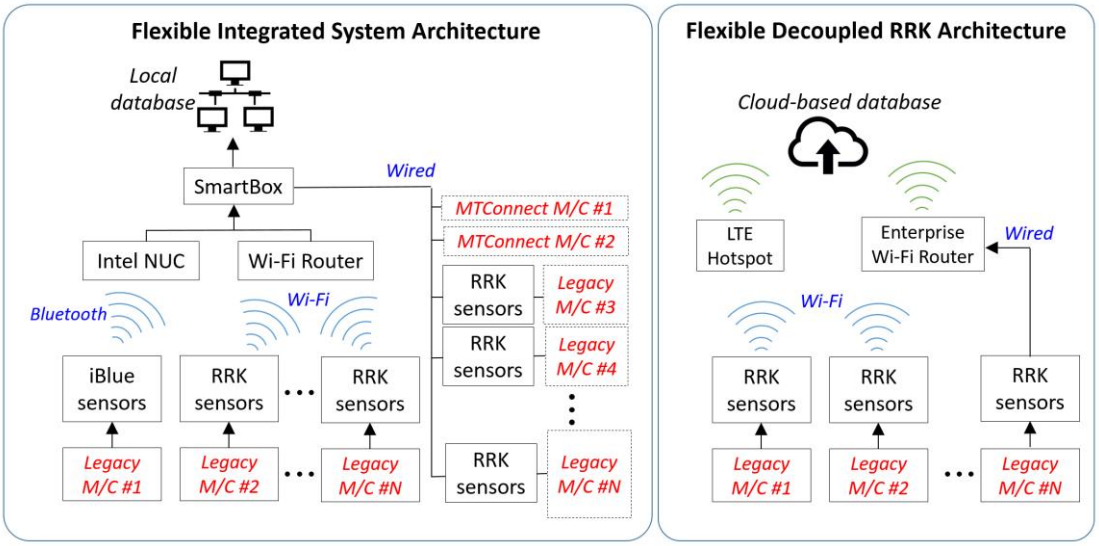


Figure 1. RRK Communications Architecture.

The implemented RRK was demonstrated in several architectures, as depicted in Figure 1. First, a flexible integrated system architecture was implemented wherein MTConnect was a base protocol for data transmission and three different sensor configurations were utilized. In this high security configuration, the Mazak SmartBox served as a central communications hub. The SmartBox provides connectivity, security, and expandability for all types and ages of manufacturing equipment. The SmartBox can be paired with a variety of third party products and provides a scalable solution or can be connected to MTConnect compatible hardware. For implementations consisting of completely MTConnect-compatible machine tools, up to 20 MTConnect interfaces may be attached. In this configuration, a legacy machine was also retrofitted with sensors coupled to the SmartBox via an ITAMCO iBlue industrial Bluetooth adapter. The iBlue adapter is capable of communicating directly to the SmartBox via a small form-factor Intel NUC device. The iBlue adapter is a standalone Bluetooth transmitter with available sensor inputs via a thermocouple probe port, a hardness probe port, and a USB Human Interface Device (HID)-enabled port that is compatible with a wide range of non-proprietary sensors and secondary computing hardware. Finally, the RRK sensor kit was also coupled to the SmartBox via wireless communication to a coupled Wi-Fi router, as in Figure 2 and Figure 3.

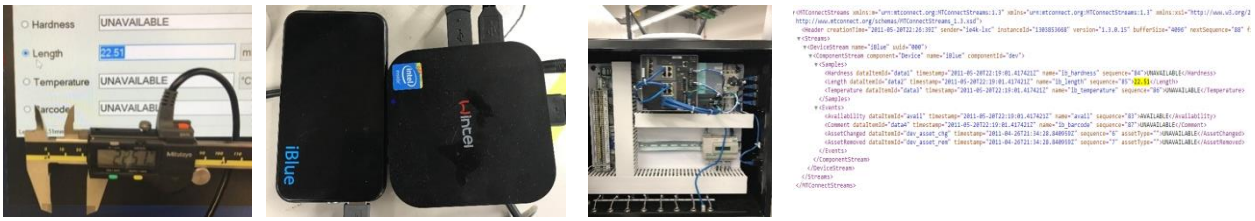


Figure 2. ITAMCO iBlue integration.

Second, a decoupled architecture was also implemented as in Figure 1 which eliminated the need for a central communications hub. In this case, a representational state transfer (REST) communications method was utilized and this provided data transmission to a cloud-based database. The RRK sensor packs communicated directly with the cloud database over wireless and wired protocols. Portable LTE hotspots (Verizon, AT&T) were used to provide secure LTE-based communication of data to a cloud-based storage services (Amazon Web Services) when internal enterprise networking was not possible. This is also depicted in Figure 3. While this limited the need for a centralized communications architecture, this provides for inherently lower security. Figure 4 shows the overall architecture for a REST-based communications system. In this case, the RRK sensor packs communicate directly with a web-hosted SQL database via HTTP GET/POST operations to the gateway. Visualization of data is possible by query of the database or the HTTP operations with the gateway.

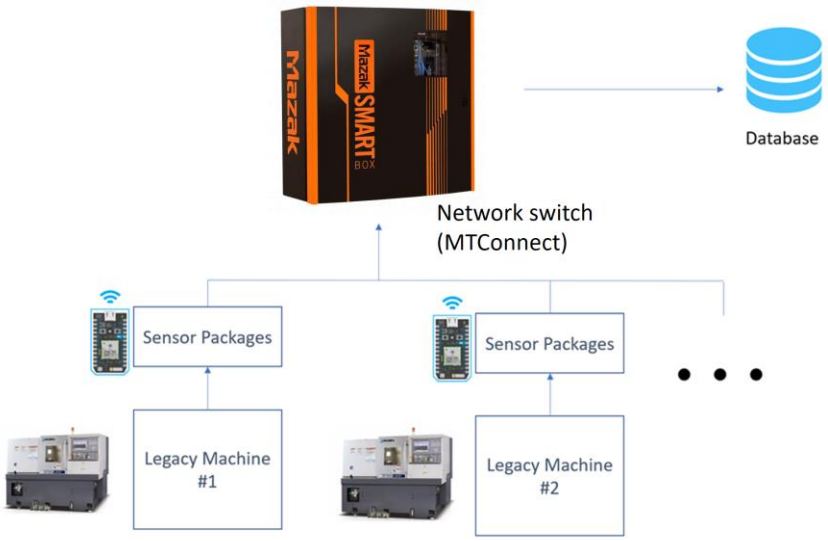


Figure 3. Local-based data service for hosting RRK sensor information using MTConnect.

Low-Cost Wireless Sensor Packages for Legacy Machines

Stand-alone wireless and wired plug-and-play sensor packages were developed for retrofit of legacy machines. The wireless sensor packages were built upon a Particle Photon device, which provides for low-cost connectivity for sensors to the cloud architecture in Figure 1. In some cases, the bandwidth required for sensing is greater than that capable within the architecture and for these the Particle Photon is used as an intermediary local processing unit for analysis before publishing the result to the cloud database. The project targeted development of sensor packages that can be integrated into a system network to specifically measure spindle/drive health (vibration), part/coolant/component temperature, coolant pH, tool wear and machine utilization. To enable a wired-based connection, the Particle Photon was paired to a Raspberry Pi platform across a USB interface. Figure 5 and Figure 6 show an RRK sensor package kit as assembled and deployed on an Okuma Genos lathe. The kit is contained within an IP-rated water-resistant housing and is affixed to the machine using a magnetic base. The kit is affixed outside of the spindle motor housing with accelerometers and temperature sensors led into the motor housing and attached to

the outer motor housing casing. Figure 7 shows instantaneous acceleration data from the accelerometer. Figure 8 shows an RRK sensor package kit deployed at the Caterpillar facility.

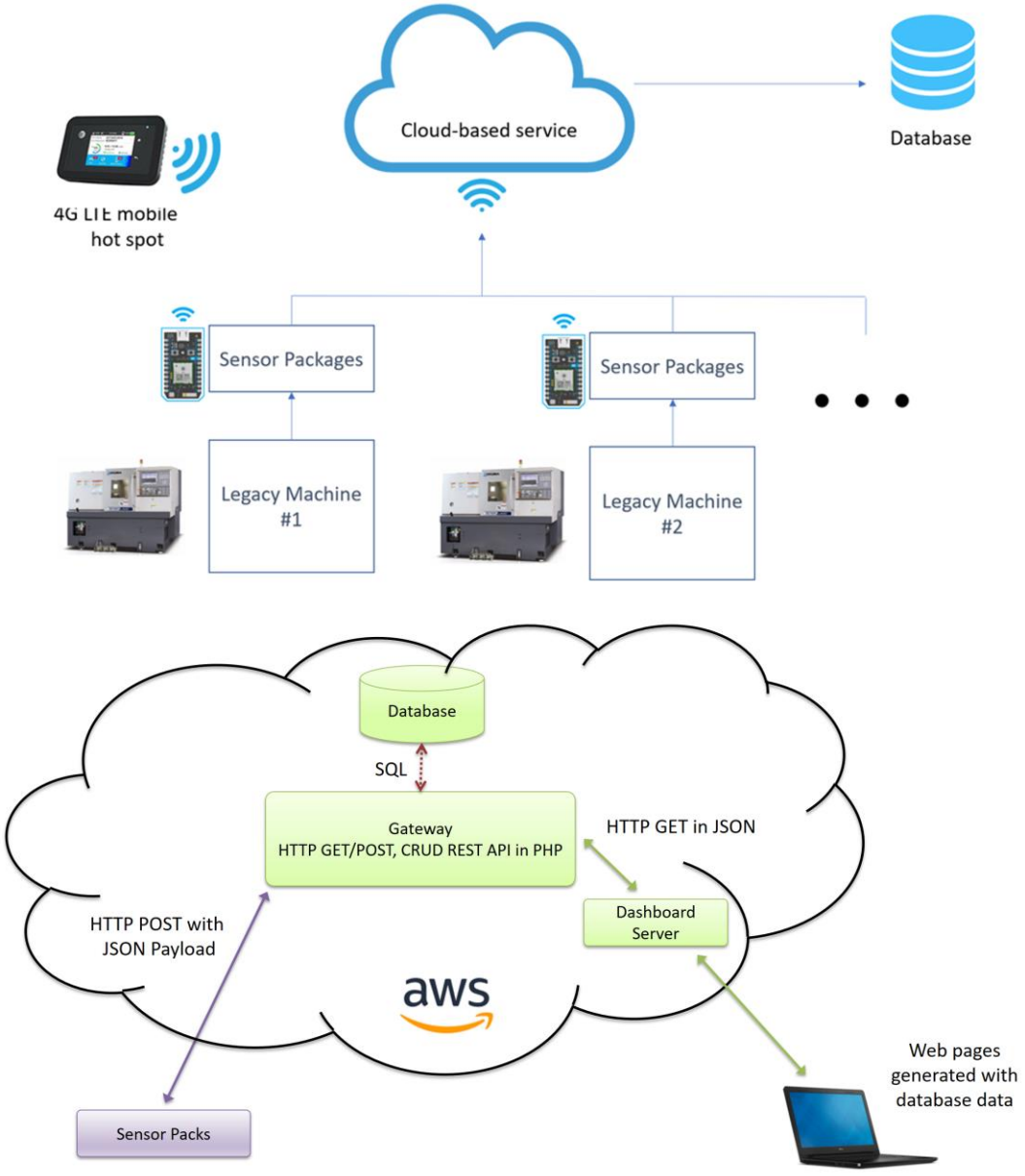


Figure 4. Cloud-based data service for hosting RRK sensor information using REST.

Wired/Wireless System Integration

Together with a functional wired/wireless communications system infrastructure based on SmartBox and iBlue, the assembled sensor packages were implemented at two facilities. The RRK implementation was conducted at manufacturing facilities for Caterpillar and GT. Data was streamed to an active cloud-based data visualization service and database. These tests enabled monitoring of the sensor pack reliability and connectivity evaluations. From these tests, best practices guides were developed to specify instructions for starting database configurations for

multiple production scenarios in terms of equipment mix (legacy/modern), sensing requirements, and cost constraints.

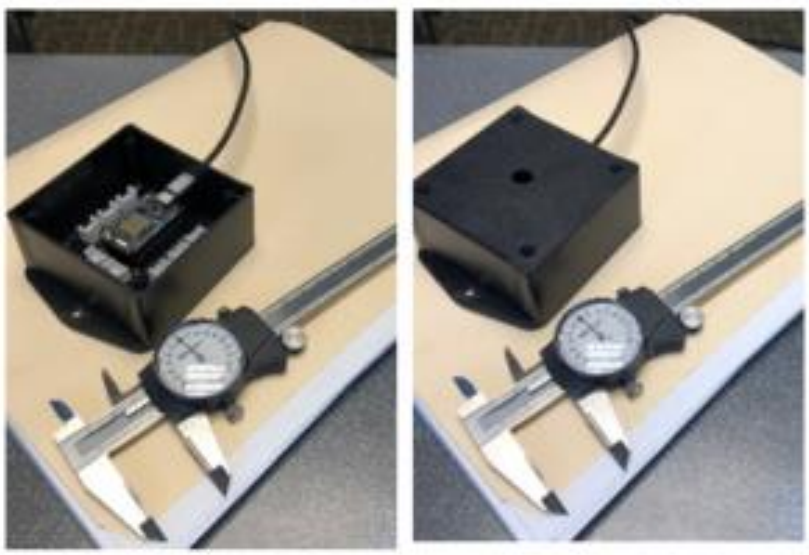


Figure 5. Photon-based reconfigurable retrofit kit sensor package.

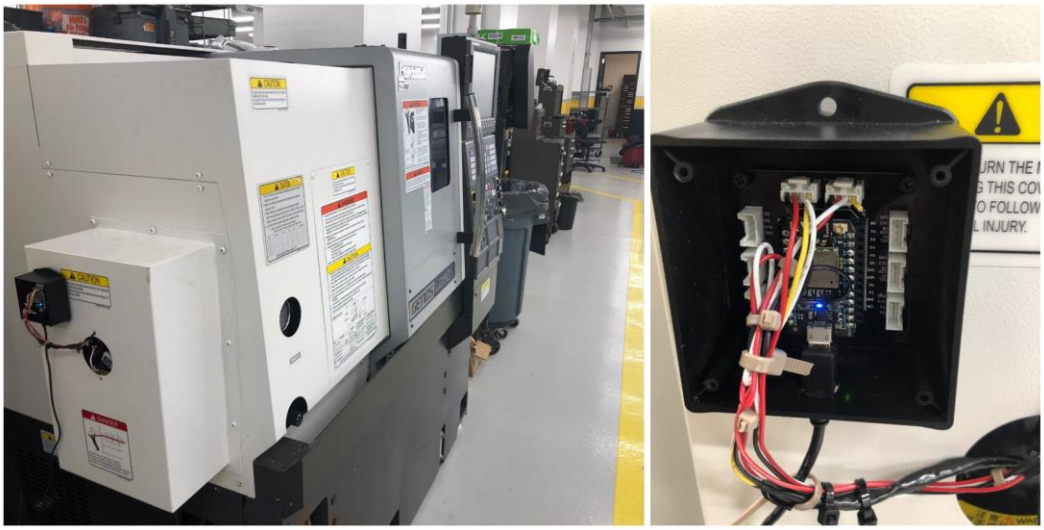


Figure 6. Photon-based reconfigurable retrofit kit sensor package at GT.

Figure 7. Data from reconfigurable retrofit kit sensor packages at GT.

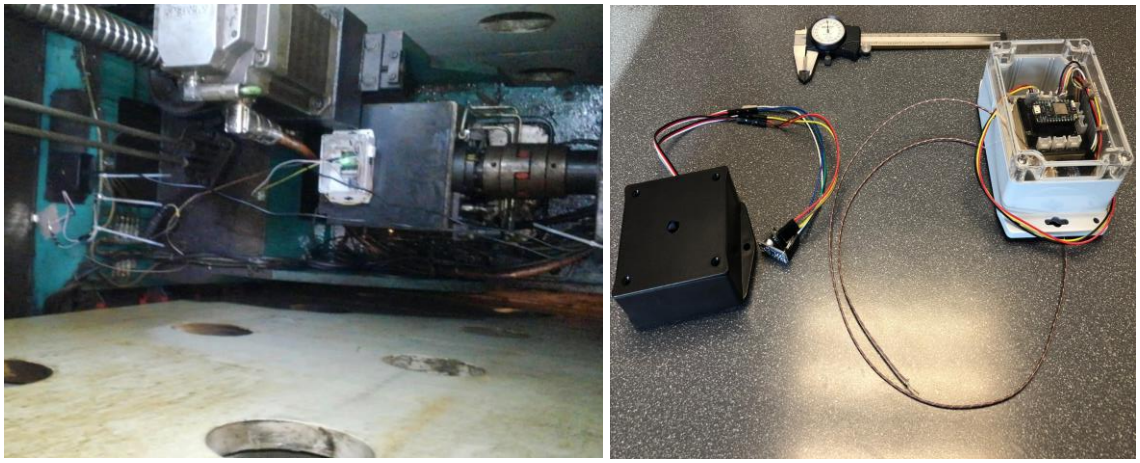


Figure 8. Photon-based reconfigurable retrofit kit sensor packages at Caterpillar.

C. Metrics Analysis and ROI Assessment

The primary metrics for the RRK kit pertain to sensor validation as well as data requirements for data storage. Figure 9 shows test data from a controlled frequency measurement test of spindle operations on a machine tool at Georgia Tech using the RRK sensor package kit as installed from the previous reporting period. The test involves running the spindle motor of the machine at a range of spindle frequencies. The data shows that the RRK sensor package accelerometer was able to accurately measure the preset spindle motor frequency, which demonstrates its capability for tracking usage of the spindle. Figure 10 shows test data from controlled fluid level monitoring sensors as well as coolant temperature sensors. In this case, the fluid level monitoring sensor is an ultrasonic sensor and from the test data, the measured and actual changes to fluid level were identical. Coolant temperature was monitored using a thermocouple and validated in a boiling test. From the data, the boundary conditions do correspond to the expected behavior. Figure 11 shows results from a current monitoring test as well as a pressure sensor test. In this regard, the data also validate the sensor behavior.

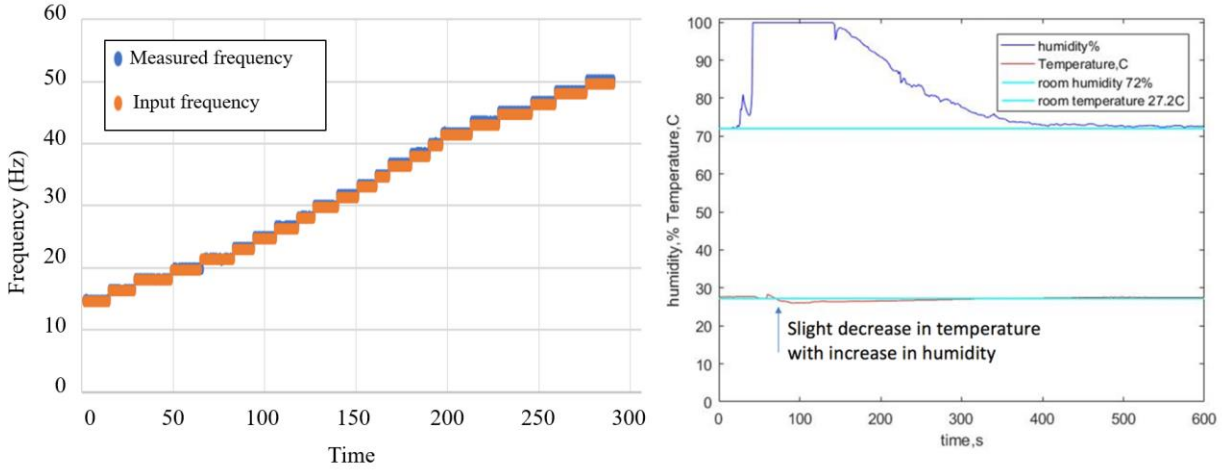


Figure 9. Sensor validation for spindle speed and humidity measurement.

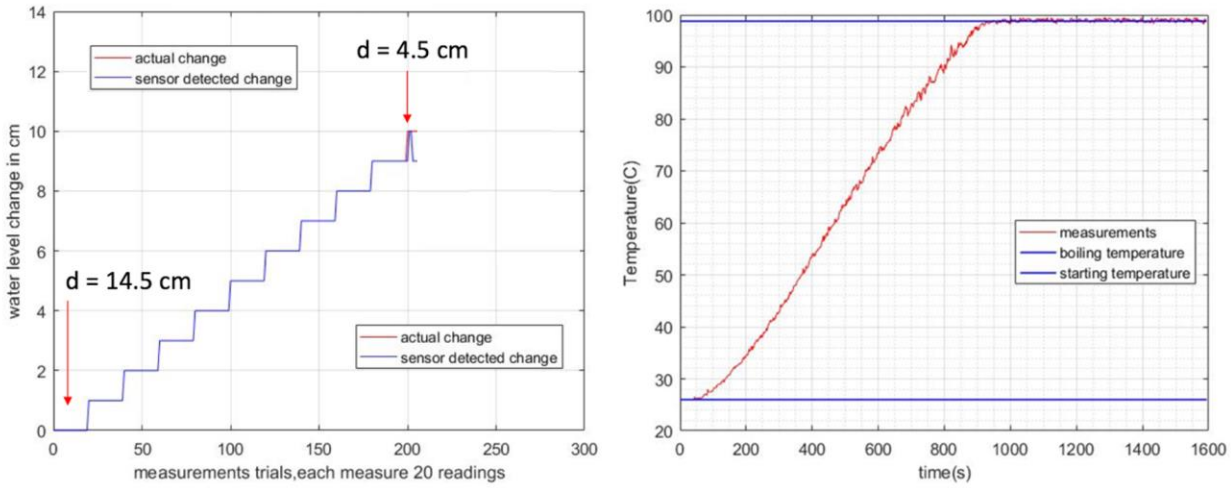


Figure 10. Sensor validation for fluid level and temperature monitoring.

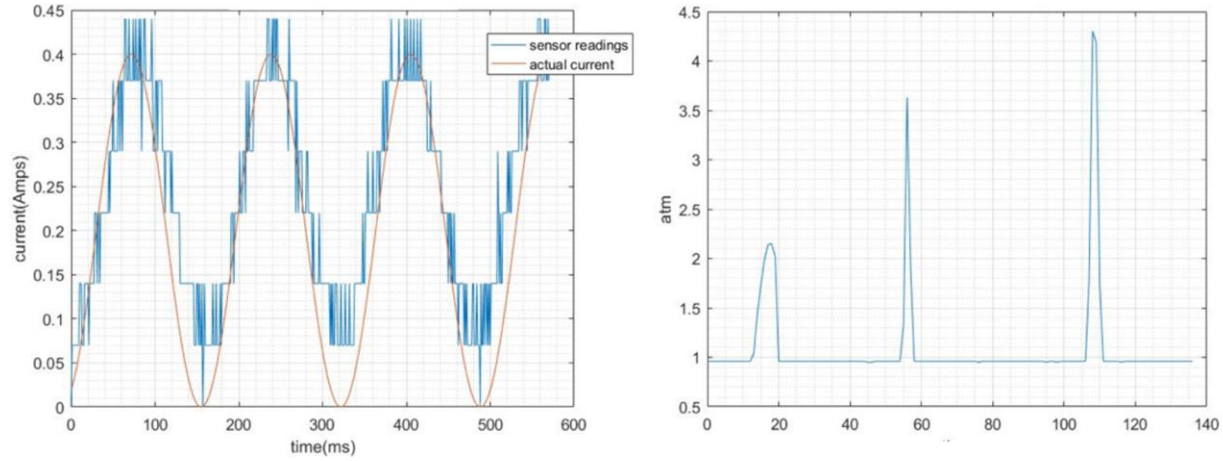


Figure 11. Sensor validation for current and pressure measurement.

Additionally, estimation of data storage requirements for the AWS database depending on the implementation of sensor retrofits at manufacturing facilities. Figure 12 shows the data requirements for bandwidth and data storage for a range of sensors both on a daily (left) and annual (right) data requirement perspective. From the figure, the data required is scales linearly with the number of sensors and it is clear that data requirements in the TB range may be required depending on the number of sensors implemented at a facility. This data will be useful for requirements planning for implementers of RRK sensor packages.

In terms of ROI, the cost basis for implementation is approximately $300/sensor package. In this regard, the base implementation cost is far below the $1000/machine implementation cost. Implementation of the iBlue and Mazak Smartbox would have to consider the actual retail prices of these hardware. For the Smartbox, an implementation across multiple machines would enable a $1000/machine implementation cost.

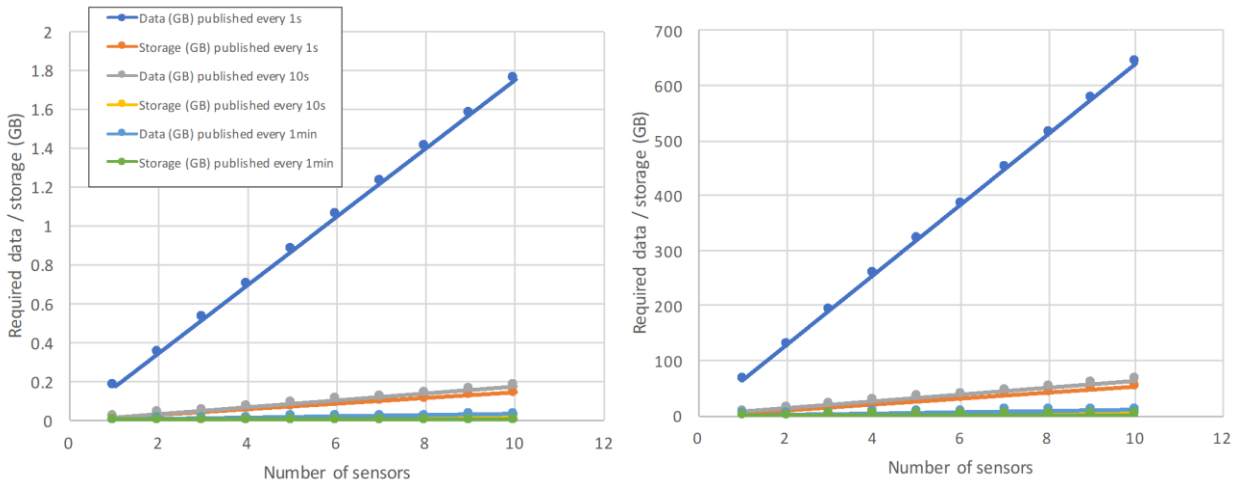


Figure 12. Data requirements for reconfigurable retrofit kit sensor packages.

D. Technology Outcomes

System Overview

The final system is a flexible approach to machine sensing that covers both legacy and modern machine tool implementations. In terms of data format and network security options, the system can be configured to adapt to the technology adopter's needs. The primary hardware elements of this sensing system are as follows: (1) RRK Sensor Kits, (2) Mazak Smartbox, and (3) ITAMCO iBlue. These hardware elements can be configured to support both a flexible integrated architecture and flexible decoupled architecture, which each have advantages and disadvantages in terms of cost, data format, and network security.

System Requirements

Installation of the system in a production environment requires: (1) access to 110/120V or 220/240V power, (2) wired/wireless/mobile connectivity depending on the selected architecture, (3) authentication of the Smartbox and RRK devices on the enterprise network (if needed), and (4) configuration of a local or cloud-based database for storing streaming data. The specification of the Mazak Smartbox and ITAMCO iBlue are provided as appendices to this report. Specifications for the RRK devices are provided below in Table 1. The RRK devices are built on the Particle Photon, which is a low-cost IOT development board. The Particle Photon specification is also appended to this report. As a summary, the Photon has a 120MHz ARM Cortex M3 micro-controller with a Broadcom Wi-Fi chip. Depending on the application of interest, the Photon can be paired with a variety of sensors to enable data collection from manufacturing equipment. Table 1 summarizes sensors that were used for collective various data modalities from manufacturing equipment in this project. It should be noted that these sensors represent a starting point for sensor selection. Many low cost sensor technologies are available of measurement of the various phenomena of interest. Selection of specific sensors for a particular application should follow the typical practice of balancing performance measures associated with sensitivity, range and noise level. The final measured signal should have sufficient voltage range to capture the phenomena of

interest without signal clipping and with sufficient sensitivity to ensure that the signal-to-noise ratio is acceptable from a signal use perspective. These performance measures are readily evaluated based on hardware specifications for commercially available sensors. The total cost of a sensor package deploying each of these sensors is approximately $300 and the price would be reduced based on need for various sensors.

Table 1. Sensor package specification.

Sensed data	**Sensor type**	**Sensor**
Spindle shock	Accelerometer (analog)	ADXL377 (3-axis, 1.3 kHz)
Spindle vibration	Accelerometer (analog)	ADXL001 (1-axis, 32 kHz)
Acoustic instability	Microphone (analog)	Electret MAX9814
Surface temperature	Thermocouple (digital)	K-type TC + amplifier
Air temperature/humidity	Capacitive, thermistor (digital)	DHT22/AM2302
Power	Current (analog)	ACS712 (20A)
Coolant pH	pH (analog)	SEN0161
Coolant hardness	PPM (analog)	DFR0300
Coolant level	Ultrasonic (digital)	HC-SR04
Pressure gauges	Pressure transducer (analog)	LX_QMCQCJPSS100805

System Architecture

These two types of system architectures were summarized graphically in Figure 1 and the salient difference summarized in Table 2. The flexible integrated architecture is built upon the MTConnect standard and assumes that the technology adopter has an ecosystem of other software applications for consumption and analysis of IOT data that is specific to MTConnect. In this case, MTConnect data is sent from adapters on MTConnect-compatible machine tools, the ITAMCO iBlue and the RRK Sensor Kits. The Mazak Smartbox can be used to integrate the data through a single controller with advanced network port security capabilities. This integrated architecture provides for high security, MTConnect-compatibility, and is a relatively higher cost due to the integration of the Mazak Smartbox. The flexible decoupled architecture is built upon REST-enabled communication to a SQL database hosted on Amazon Web Services. This implementation is designed with a general practitioner in mind whose current architecture is not established and/or does not require native compatibility with MTConnect. The benefits of this implementation are relatively simpler setup and information storage to the cloud database and ability to use cloud computing capabilities across other information service providers. The decoupled architecture provides for flexibility in information communication protocols and lower cost as each sensor package independently communicates directly with the cloud database. However, as this architecture does not have a common data controller, it relies on ability to set network security privileges using existing enterprise network hardware. In scenarios where this is not possible,

integration of this architecture with mobile-based hotspots (e.g., 5G, LTE), provides for connectivity of the sensor packs to the cloud database.

The architecture as described above is configurable for two approaches, the first based on MTConnect to an enterprise server database and the second based on REST communications to a cloud-based database. The MTConnect XML-based standard is well-documented and is not summarized here. The REST communication standard implemented in this project is depicted in Figure 13. In this case, the data is passed in packets between client and server. The architecture of the database is depicted in Figure 14, which shows relationships between various data tables and IDs.

Table 2. System architecture configurations and attributes/comparisons.

Sensed data	**Flexible integrated**	**Flexible decoupled**
Recommended use cases	• Facilities with some degree of existing connectivity to machine tools through machine connectivity standards (MTConnect, OPCUA) • Enterprises with software relying on MTConnect or OPCUA data sources.	• Facilities with no existing connectivity or implemented software solutions for machine data consumption. • Integration of low-bandwidth machine state and high-bandwidth process condition data
Communication standard	MTConnect	Representational State Transfer
Advantages	• Local- or cloud-hosted • Native compatibility with MTConnect software solutions • Hybrid architecture inclusive of MTConnect-compatible machines	• Local- or cloud-hosted • Native scalability with number of devices • Potential for mesh-based WiFi implementations for wide-area coverage
Disadvantages	• Potential scalability issues with number of devices in terms of network configuration as well as database structure • Potential bandwidth limits for sensor measurements due to protocol overhead	• Requires some interfaces for linkage with existing MTConnect software solutions • Retrofit of all machines, including those that may be MTConnect compatible already
Cost drivers	• Network server equipment hosting MTConnect agents • Need for wired implementations and network repeaters	• Cost of IOT devices
Security considerations	• Configurable port security with local network configuration on shop floor • Encrypted data transmission • MAC authentication protocols	• Security only configurable at plant access points • Encrypted data transmission • MAC authentication protocols

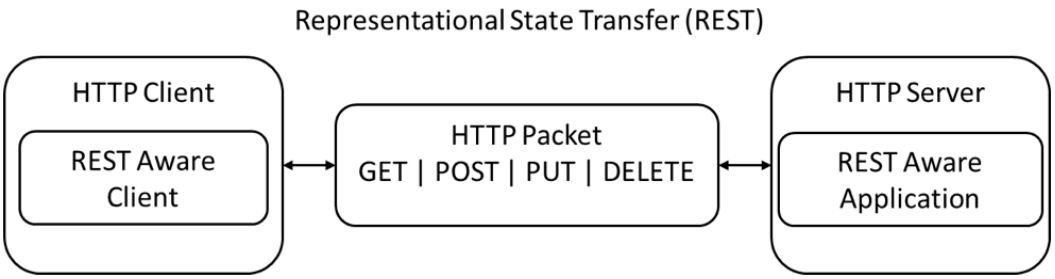


Figure 13. Data transfer protocols for cloud-based publishing.

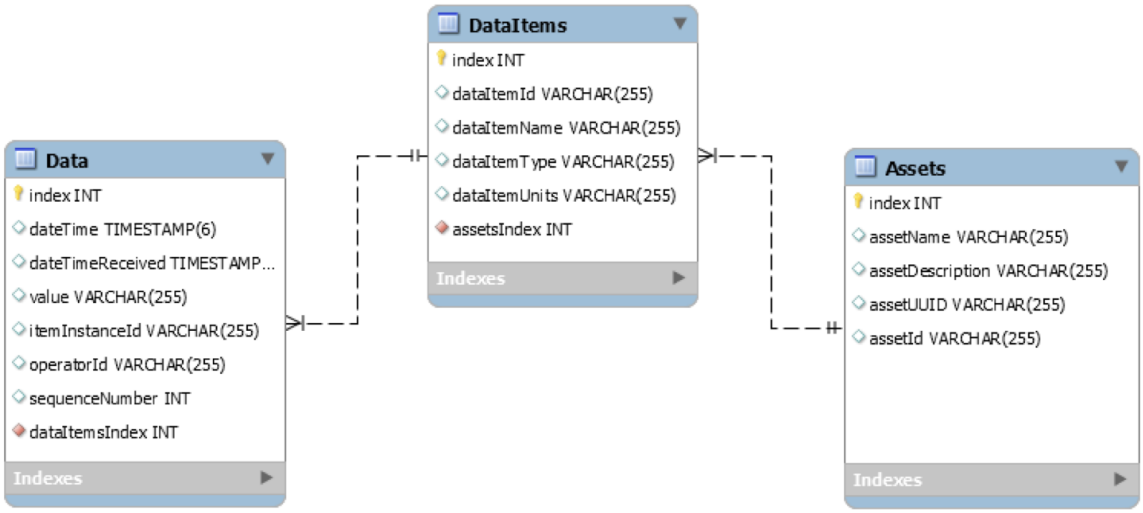


Figure 14. Database architecture for cloud-based publishing.

Features & Attributes

The machine monitoring solution includes a variety of features depending on the specific architecture implemented by the user. For the flexible integrated system architecture, the Mazak Smartbox provides for configurable port security. The Smartbox is an industrially hardened communications platform with a ruggedized Cisco IE4000 series industrial Ethernet switch. The SmartBox provides state-of-the-art Layer-3 compatible network security features necessary for securing the machine tool platforms from network intrusion and securely connecting machine tools to enterprise IT infrastructure, including 802.1x port security, dynamic port-based authentication, encrypted administrative traffic, IEEE 802.1AE MACsec encryption, FIPS compliant and centralized authentication. While the low-cost, single SmartBox implementation provides for network security for all attached machines and/or sensors, adopters may find enhanced security and further utility in deployment of multiple SmartBox units for compartmentalization of individual subsystems and machine groupings. Through the use of MTConnect, all information (state or sensor data) flows upwards to the enterprise, or cloud, in a read-only fashion. In this regard, the MTConnect standard enables manufacturing equipment to provide data in structured

XML rather than proprietary formats. MTConnect uses standards (HTTP / XML) to provide a simple and readily available transport, as well as a rich data model. For the flexible decoupled system architecture, the independent nature of the RRK sensor packages provides for low cost of implementation and ability to specify specific sets of sensors relevant to an implementer's application. In this architecture, the REST-based communications with the SQL database facilitates data storage in a format that can easily be addressed for historical or dynamic analysis.

Modes of Operation

The primary mode of operation for the sensor retrofit kit is as a monitoring solution for legacy and MTConnect-compatible machine tools. The architecture of the monitoring solution as described above is flexible to enable monitoring for a variety of stakeholders as described in the ensuing.

Software Development Documentation/Design Document

The primary focus of this project was development of the hardware kit. The codes associated with the RRK sensor packages and the configuration of the cloud database is provided in the appendices to this report.

Users & Use Cases

The RRK system and associated monitoring architecture is anticipated to have users that include production engineers and managers, skilled workforce, and information technology developers across large and small business entities. Implementers can be demarcated depending on degree of existing plant machine monitoring hardware/software infrastructure, investment cost sensitivity, network security sensitivity, technical workforce capacity for IOT and sensor deployment. The system architecture as described provides for flexibility according to the use cases wherein these demarcations are set.

In the case of a greenfield implementation with no existing hardware/software infrastructure for machine monitoring, flexibility in an MTConnect or REST-based communication standard provides ideal configuration options. For cost-conscious deployment in this case, the REST-based communication standard can be integrated into plant operations and per machine implementation costs can be kept minimal using the RRK sensor kits as well as the cloud-based data storage options where dashboards and user queries can be designed to consume/process/analyze sensor data. In cases where enterprise software is of interest to the implementer, the MTConnect solution can be deployed wherein the RRK sensor kits are configured to connect to the enterprise database. Challenges in this solution would be in security-based authentication policies of the IOT devices within the implementer's existing IOT network. In the case of a brownfield implementation wherein an MTConnect solution already has been adopted by the implementer, the RRK sensor kits can be configured to connect to MTConnect agents using wired protocols if necessary. In this case, the scalability of the implementation for a $1000/machine cost would be 1 machine. In the case where cost isn't a significant driver limiting deployment, but point-of-use security is critical to deployment, a machine group / hub implementation use case is considered where the Mazak Smartbox can provide state-of-the-art MTConnect secure connections. In this case, the scalability

of the solution for a $1000/machine cost would roughly be 10 machines. These use case specifications are detailed in Table 3.

The specific machine sensing information targeted in this development was indicators of machine status pertaining to spindle condition monitoring and machine crash monitoring. The specific business problems associated with these machine sensing use cases are to provide data for asset health, utilization and machine uptime. The first of these two cases was implemented in a low-bandwidth sensor pack wherein accelerometers and temperature sensors are used to monitor the condition of motor equipment. The second of these two cases was implemented in a high-bandwidth sensor pack wherein accelerometers and microphones are used to detect rapidly occurring events wherein maximum measurement bandwidth is needed.

Table 3. Use case specification.

Use case	Setup	Components
Single machine	Wireless	Particle Photon (wi-fi, $19) + sensor package ($283)
Machine group	Hub	SmartBox (+ wireless router)
Machine group	Node (Wireless)	iBlue (wireless), Particle Photon (wi-fi, $19)

E. Implementation

Deliverables and their TRL

The flexible integrated architecture as well as the decoupled integrated architecture follows a strategy to provide flexibility based on industrially hardened communications network hubs and wireless sensor packages. Each of the individual systems components are mature from an operational perspective (TRL6-7) and their integration as a system for the retrofit concept is also demonstrated in a relevant production environment (TRL6). The key technical advances that were made during this project include demonstration of prototype systems-level integration and true flexibility of data sensing for customer-specific applications using low-cost, industry-standard high security integration platforms. The open nature of the data transfer, particularly for the decoupled integrated architecture, enables remote data analysis as well as plug-and-play incorporation of new models and sensors into the overall diagnostic system architecture. Ability to analyze data remotely enables business models wherein third party suppliers might provide diagnostic/prognostic services to companies from larger enterprises to small/medium enterprises.

The specific deliverables of this project are as follows are provided as an appendix to this report:

- Deliverable 1. Communications system specifications report. This document provides a full description of the wired/wireless communications platform system specifications and network setup for the retrofit kit implementation.

- Deliverable 2. Legacy sensor package assessment report. This document provides design/implementation details and performance information of sensor packages selected for implementation in the wired and wireless communications platforms.
- Deliverables 3 and 4. Report on installation operation feedback and on RRK system test and performance. This document, shown in Figure 13, provides details pertaining to installation of the retrofit kit systems at the project partner sites, as well as any technological issues arising during the installation process. This document also summarizes the system test and quantitative performance for the retrofit kit implementations for the project partner sites.
- Deliverable 5. Transition plan and improvements report. This document, shown in Figure 14, identifies pathways for transitioning the retrofit kit system to DMDII membership and also identify areas for improvement for the final retrofit kit system.
- Deliverable 6. Final Project report. The present document serves as the final report for the retrofit kit project and summarizes all data pertaining to design, implementation and performance of the retrofit kit system.

The technology developed in this project are openly available in the commercial retail market and are evolving technologies used in commercial IOT-enabled products. In this regard, there is no background intellectual property required to implement these technologies in the use cases of interest to the DMDII membership. To realize the project outcomes, adopters must be able to integrate these low cost IOT devices within their existing plant infrastructure and internal IT security practices. Guidance on compatible system architectures relative to existing infrastructure is provided below.

F. Technology Transition Plan and Commercialization

Identify Future Plans

The project has resulted in development of a RRK sensor package that can be integrated into commercial industrial solutions. Industry partners involved in development of technology solution (ITAMCO) as well as implementation of the technology at their facilities (Caterpillar) have provided useful perspectives on the capabilities of the system. The project team has future plans to further develop additional edge computing applications using the base retrofit kits. To complement the data connectivity, the project team will develop cloud-based analytics-based methods for processing of retrofit kit sensor data. The project team will consult with existing firms and new startup firms interested in developing commercial offering for retrofit kits and web applications. Transition opportunities with companies interested in developing and commercializing sensor packs are welcome. Elements of our team have had discussions with others in this regard.

Market Assessment

According to the IDC Worldwide Semiannual Internet of Things Spending Guide (2017), IOT spending in the manufacturing area is predicted to be $183B by 2021. The core technology of the RRK sensor package is approximately $300/package in terms of cost of materials. Further, the components that the machine monitoring and RRK sensor packages are built upon are

commercially available items that are in high supply. Implementers can easily purchase base components in small lot sizes and at low cost. It is anticipated that the market for these technologies is significant considering that a sensor retrofit solution at this price level does not currently exist in the market. The unique element is how to integrate these to achieve ubiquitous sensing for manufacturing. Companies that exist in this market include technology developers with expensive and cost-intensive data collection devices. Low cost computing for manufacturing is an open market. In applying the technical advancements made in this project, future development and/or implementation of these technologies can be made by new startup firms or existing firms interested in offering/adoption of these technologies.

Identified Barriers to Adoption

The low-cost IOT devices central to the RRK sensor packages may require integration with existing enterprise networking infrastructure. This compatibility requirement is critical in terms of authentication of these devices on existing wired/wireless networks. In this regard, specific IOT microcontrollers and/or single board computers may have varied capability for supporting more advanced wireless authentication protocols that may require use of specialized authentication certificates. Further, adopter facilities may need to review compatibility of these devices with existing site-specific network security policies. In terms of the implementation choice for local and/or cloud based data storage, implementers may also have policies in place for determining feasibility of off-site data storage and the two architectures proposed here provide for flexibility in this regard. Additionally, integration of the technology solution would require ability to follow assembly documentation for off-the-shelf solution and an ability to write code for customized sensor solutions. These codes are used quite heavily now in practice so adopters can leverage an engaged community of developers through online technical forums. Finally, as the technology space surrounding low-cost IOT devices is characterized by rapid pace of development, adopters must also pay attention to these developments as new capabilities of these devices may offer significant advantages over the current development.

G. Workforce Development

As part of advancing workforce development in this area, ITAMCO has developed a high school scale tutorial for advanced precision manufacturing and integration of measurements with understanding of additive and subtractive processing. Figure 15 shows example screenshots of this presentation and the assignments given to students in this area. It is anticipated that this will assist with advancing knowledge in this area for measurements in manufacturing.

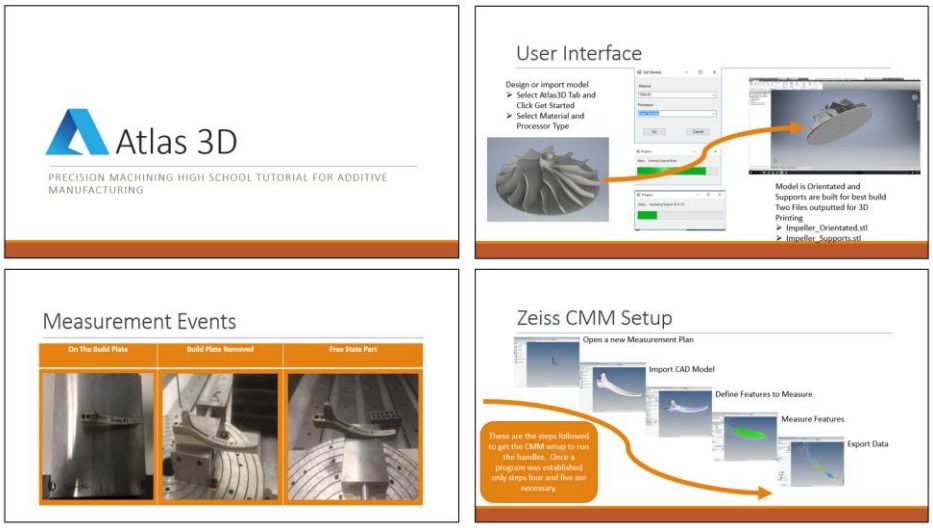


Figure 15. Workforce Development Training.

H. Conclusions and Recommendations

The flexible and reconfigurable monitoring solution developed in this DMDII-supported work provides options for low-cost (less than $1000) and significant flexibility in data sensing implementation redesign as the shop floor evolves with plant/cell reorganization, changing product mix and different needs for in process sensing. Further, this reconfigurable sensor package network provides for true scalability of cost of ownership for enabling data sensing across the shop floor and enables customer-driven selection and implementation of sensing capabilities. Practitioners will be able to follow the advancements made in this work to implement these technologies in their shop floor. It is recommended that implementers of this technology do pay close attention to advancements made continuously in these low cost IOT microprocessors and single board computers as new capabilities may provide for substantially quick changes in the landscape of internet-enabled technologies for machine monitoring.

I. Ending Financials and Labor Hour Assessment

The final budget expenditure has come from labor and material costs for Georgia Tech and ITAMCO. The Georgia Tech portion of the expenditure was $180,704.31 and the ITAMCO portion of the expenditure was $91,681.76. The final cost share from each entity are as follows: Georgia Tech ($217,512), ITAMCO ($130,841.76), Caterpillar ($70,243) and Mazak ($224,930.01). Labor hour assessment is as follows: Georgia Tech (faculty: 260 hours, student: 2080 hours), ITAMCO (manager: 774 hours, programmer: 596.5 hours), Caterpillar (senior engineer: 459.5 hours).

J. Lessons Learned

<u>Problems Encountered</u>

In implementing the low-cost IOT devices, the project team did encounter initial difficulties in connecting IOT devices to enterprise networks at the partner facility. The primary issues in this

regard surrounded IT certification policies for the IOT devices as well as lack of compatibility with the specific wireless network configuration policy for the facility. These issues were resolved in developing a wired solution for integration using Ethernet-based connectivity as well as using mobile-based hotspots for connectivity of data external to enterprise servers. It is suggested that implementers of this technology seek to understand compatibility issues with their internal IT policies at early stages of development. Additionally, the technology surrounding low-cost IOT devices is rapidly evolving and native compatibility with existing enterprise network technologies (e.g., WPA2, WEP authentication protocols) will likely be addressed as the technology matures. It is suggested that interested parties and adopters stay abreast of the connectivity capabilities of these devices and the design of their own wireless networks to understand if alternative solutions become commercially available. Further, for facilities that do not offer headless connectivity of IOT devices, such as is common in the case of web-based authentication of network devices, it is recommended that they pursue parallel networks that allow for headless connectivity as these devices are becoming very common and will be the standard for ubiquitous computing. A final implementation concern would be in the existing enterprise software framework used to consume machine status data. For implementer that require specific connectivity formats where a specific protocol (e.g., MTConnect, OPCUA, etc) is used to populate an enterprise resource planning (ERP) system, translators and interfaces would have to be designed to provide a pathway to broadcast data from databases populated by low-cost IOT devices. These existing standards require specific send/receive formats that are not natively designed to integrate with ubiquitous IOT frameworks which may incorporate hundreds to thousands of devices sending millions of messages. It is recommended that implementers with entrenched software applications review the needs for the development of such hardware/software interfaces with low-cost IOT devices.

In terms of sensor selection, the low-cost computing devices have specific compatibility requirements with power consumption from the associated sensors (e.g., bias voltage compatibility). Adopters should ensure that sensors that are selected meet the power provided by various analog and digital input/output terminals documented in the specifications for the underlying microcontroller. In terms of sensor selection relative to primary measures of performance, implementers must select sensors with appropriate sensitivity, range, and noise level as is common practice in any sensor selection scheme. The goal of the sensor selection should be to capture the full range of the signal of interest with an appropriate sensitivity and noise level to ensure the final measured electrical signal is indicative of the phenomena of interest. For example, if an accelerometer is to be used, the implementer should ensure that the vibrations of interest yield a measurable signal with no clipping (e.g., signal beyond voltage range) and sufficiently high nominal value such that the signal to noise level is useful for measurement.

Scope of Work Deviations

A no-cost extension (NCE) on this project was granted until August 2018. This NCE was to facilitate on-machine testing that was not able to occur due to a later-than-expected project kickoff.

Risks

As stated above, among the primary barriers to adoption are incompatibility with internal IT policies and equipment as well as potentially rapid enhancement/obsolescence of technology. Implementers must be aware that these devices are rapidly evolving and while the current state of implementation may be sufficient for the present application, the enhancement of the technology year over year may provide significant advantages from cost and support perspectives.

III. Appendices

List of Document Deliverables

Deliverable 1. Communications system specifications report. This document provides a full description of the wired/wireless communications platform system specifications and network setup for the retrofit kit implementation.

Deliverable 2. Legacy sensor package assessment report. This document provides design/implementation details and performance information of sensor packages selected for implementation in the wired and wireless communications platforms.

Deliverables 3 and 4. Report on installation operation feedback and on RRK system test and performance. This document, shown in Figure 13, provides details pertaining to installation of the retrofit kit systems at the project partner sites, as well as any technological issues arising during the installation process. This document also summarizes the system test and quantitative performance for the retrofit kit implementations for the project partner sites.

Deliverable 5. Transition plan and improvements report. This document, shown in Figure 14, identifies pathways for transitioning the retrofit kit system to DMDII membership and also identify areas for improvement for the final retrofit kit system.

Deliverable 6. Final Project report. The present document serves as the final report for the retrofit kit project and summarizes all data pertaining to design, implementation and performance of the retrofit kit system.

Quick guide on how to set up AWS RDS and AWS EC2 for RESTful communication with a MySQL database

❖ Communications Setup

The present project utilized two architectures: (1) flexible integrated system architecture and (2) flexible decoupled system architecture. The flexible integrated system architecture was employed at two project partner sites. For the first partner site, a Mazak SmartBox served as the connection point for a Bluetooth connection to a ITAMCO iBlue and a wired connection to a Photon/Raspberry Pi sensor retrofit kit. In this case, the Mazak SmartBox was authenticated on the enterprise wired network. For the second partner site, wired connections were made directly between the Photon/Raspberry Pi sensor retrofit kit to the enterprise wired network (without a Mazak SmartBox). MTConnect agents on the enterprise servers provided data connectivity to the sensor retrofit MTConnect adapters. The decoupled system architecture was also employed at both of these project sites. For the first partner site, these Photon sensor retrofit kits were connected to an Amazon Web Services (AWS) hosted database by connecting these sensor kits to enterprise wireless. For the second partner site, the sensor retrofit kits were found to be incompatible with the enterprise wireless protocols due to inability to support server certificate authentication (WPA2 Enterprise). In this case, an LTE mobile hotspot was used to provide direct connectivity to the AWS database. These different connectivity methods were found to be sufficient for data streaming purposes and their reliability were inherently related to the reliability of the respective wired/wireless networks. The below information provides details on setting up an AWS database instance for connectivity of the sensor retrofit kits.

❖ Setting up the Amazon Relational Database (AWS RDS)

I. How do I create and activate a new Amazon Web Services account?
https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/

II. Setting Up for Amazon RDS https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SettingUp.html

III. Creating a MySQL DB Instance and Connecting to a Database on a MySQL DB Instance
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.MySQL.html

IV. Install MySql Workbench, connect to the database, and run the following query block:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';


-- -----------------------------------------------------

-- Schema iot
```

```
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `iot` DEFAULT CHARACTER SET latin1 ;
USE `iot` ;

-- -----------------------------------------------------
-- Table `iot`.`data`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `iot`.`data` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `timestamp` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `assetId` VARCHAR(45) NOT NULL,
  `dataItemId` VARCHAR(45) NOT NULL,
  `value` VARCHAR(45) NOT NULL,
  `dataItemId2` VARCHAR(45) NOT NULL,
  `value2` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1107753
DEFAULT CHARACTER SET = latin1;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

❖ Setting up the AWS EC2 to add REST API

I. AWS Quick Start Guide: Launch a Linux Virtual Machine https://docs.aws.amazon.com/quickstarts/latest/vmlaunch/welcome.html

II. Download the api.php from:

https://github.com/mevdschee/php-crud-api

III. Open the api.php with a text editor, uncomment the following lines, and fill them appropriately:

```
// $api = new PHP_CRUD_API(array(
//         'dbengine'=>'MySQL',
//         'hostname'=>'localhost',
//         'username'=>'',
//         'password'=>'',
//         'database'=>'',
//         'charset'=>'utf8mb4'
// ));
// $api->executeCommand();
```

IV. Upload the api.php to the EC2 server and communicate with the database according to the documentation

For example:

HTTP POST to

http://ec2-xx-xxx-xxx-xxx.us-west-2.compute.amazonaws.com/api.php/data

with the body: {"assetId":"OKUMA001","dataItemId":"Temperature","value":"23.6"}

Specifications and Installation of High and Low Bandwidth Sensor Packs

❖ Legacy Sensor Package Setup

The present project utilized legacy sensor packages developed to couple flexible sensors to a variety of machine tools. Two types of sensor packages were developed for this purpose: (1) low bandwidth sensor pack, (2) high bandwidth sensor pack. Both of these sensor packs are based on an implementation using the Particle Photon base microcontroller. The primary differences in the implementations for both of these sensor packs are in the components used as well as the primary use cases of each. The low bandwidth sensor pack is designed to communicate machine state information periodically to a server database. The high bandwidth sensor pack is designed to communicate rapid changes in machine state (e.g., crashes, sudden accelerations) to a server database. The core differences are in terms of the bandwidth needed as well as the code implemented for each. The validation of these sensor packs is found above in the main body of the final project report and the below contains bill of materials information as well as source code for implementing these sensor packs by practitioners.

High Bandwidth Sensor Pack

Specifications

- Photon spec
    - Particle PØ Wi-Fi module
        - Broadcom BCM43362 Wi-Fi chip
        - 802.11b/g/n Wi-Fi
        - STM32F205RGY6 120Mhz ARM Cortex M3
        - 1MB flash, 128KB RAM
    - On-board RGB status LED (ext. drive provided)
    - 18 Mixed-signal GPIO and advanced peripherals
    - Open source design
    - Real-time operating system (FreeRTOS)
    - Soft AP setup
    - FCC, CE and IC certified

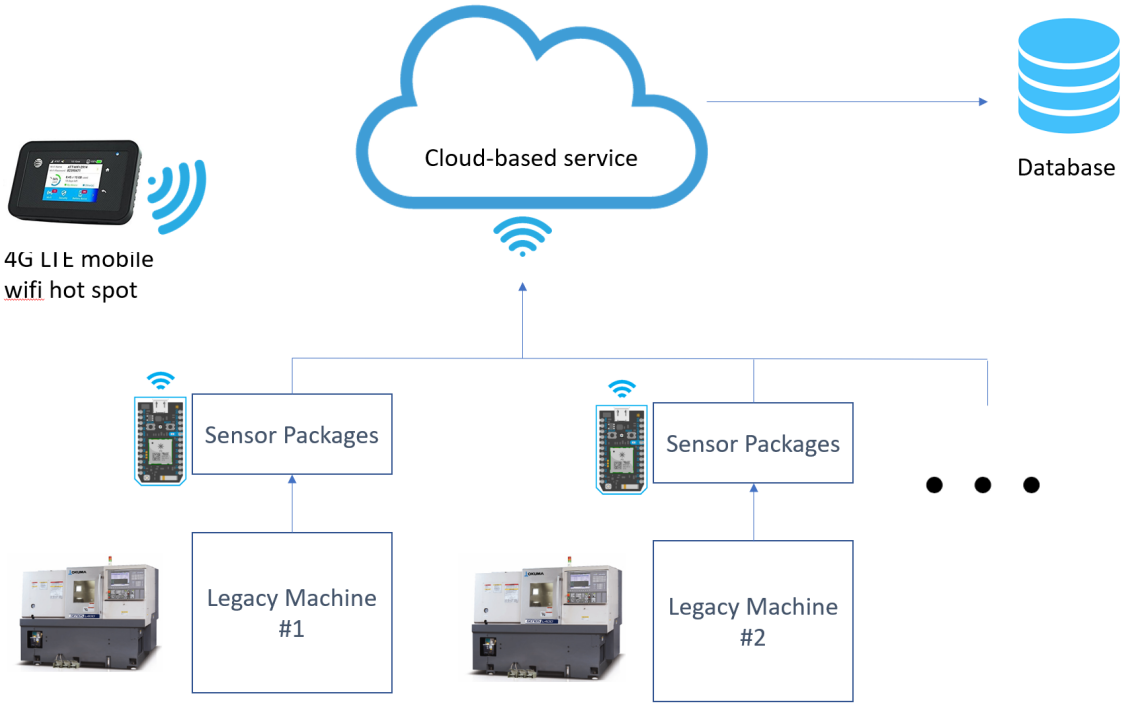
- Sensor spec
    - ADXL377
        - 3-axis accelerometer
        - +- 200g
    - MAX9814
        - Automatic Gain Control (AGC)
        - Low Input-Referred Noise Density of 30nV/√Hz
        - Low THD: 0.04% (typ)
- Assembled pack
    - Weight: 118g (w/o usb cord or adaptor)
    - Dimension (L x W x H): 7.5cm x 5.2cm x 3cm

Bill of Materials (Total Price $88.10)

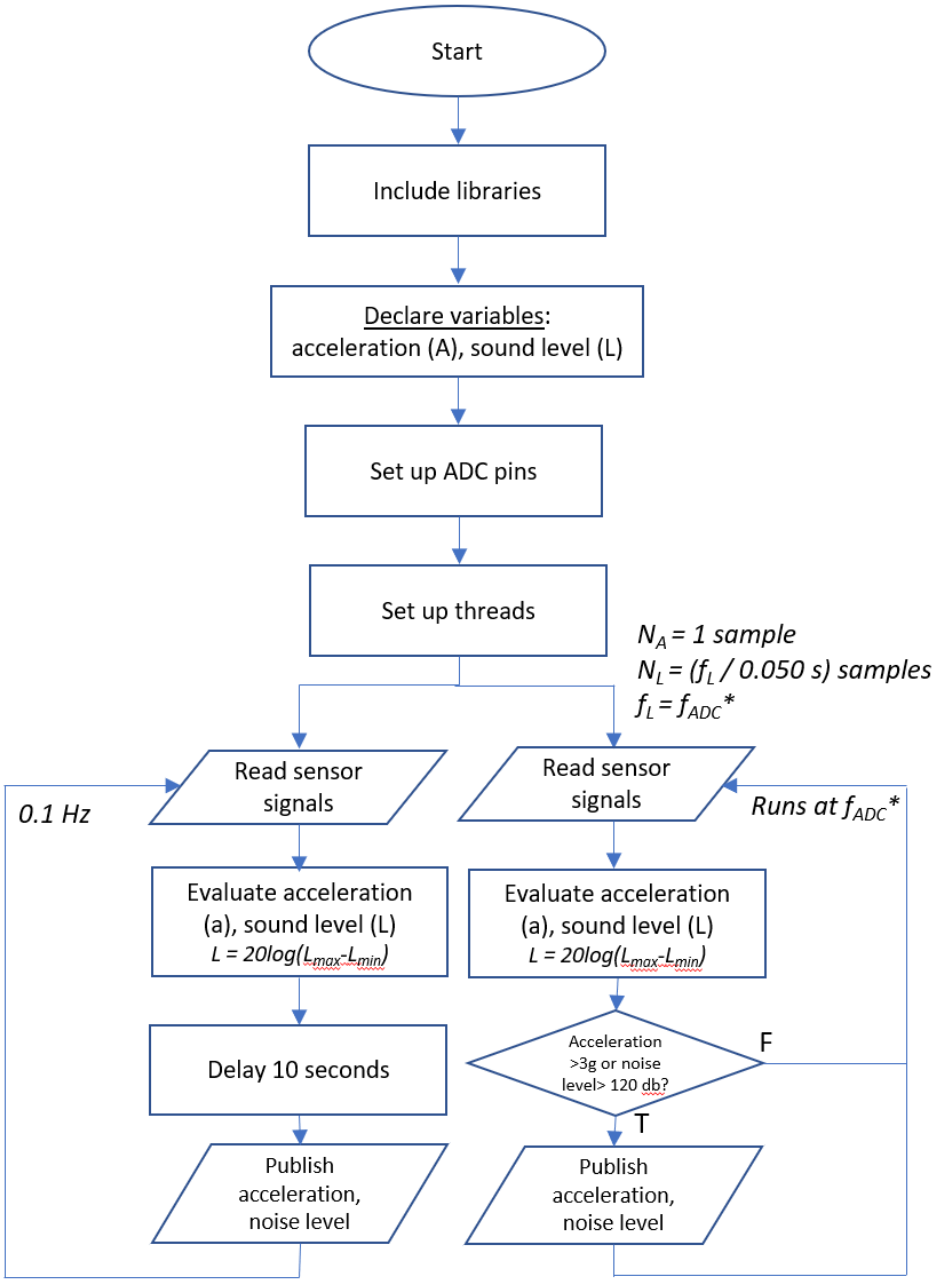
Category	Item	Type	Description	Supplier	Unit Price
Platform and Shield	PARTICLE PHOTON	Development platform	WiFi IoT Development Kit	Digikey	$19.00
	PARTICLE PHOTON BASE SHIELD	Grove Connectors for Particle	3 digital ports, 2 analog ports, 2 I2C ports and 1 UART port	Seeed	$6.90
Sensors	ADXL 377 BREAKOUT BOARD	Accelerometer for shock	3 Axis, Analog, ADXL377 - High-G Triple-Axis Accelerometer ,±200g Analog Out, 6.5mV/g sensitivity	Digikey	$24.95
	EVAl BOARD FOR MAX9814	Microphone	Electret Microphone Amplifier - MAX9814 with Auto Gain Control and Low-Noise Microphone Bias	Digikey	$7.95
Power Supply	USB 2.0 A-PLUG TO MCRO-B-PLUG	USB cable	USB 2.0 Cable A Male to Micro B Male 16.40' (5.00m) Shielded	Digikey	$9.22
	AC/DC WALL MOUNT ADAPTER 5V 10W	Power adapter	5V 10W AC/DC External Wall Mount Adapter Fixed Blade Input	Digikey	$7.38
Casing	BOX PLASTIC GRAY/CL 3.28" X 3.2"	Sensor package case	Box with Mounting Flange Plastic, Clear Cover/Door Cover Included 3.283" L x 3.196" W	Digikey	$12.70

System Architecture

1) Physical system diagram

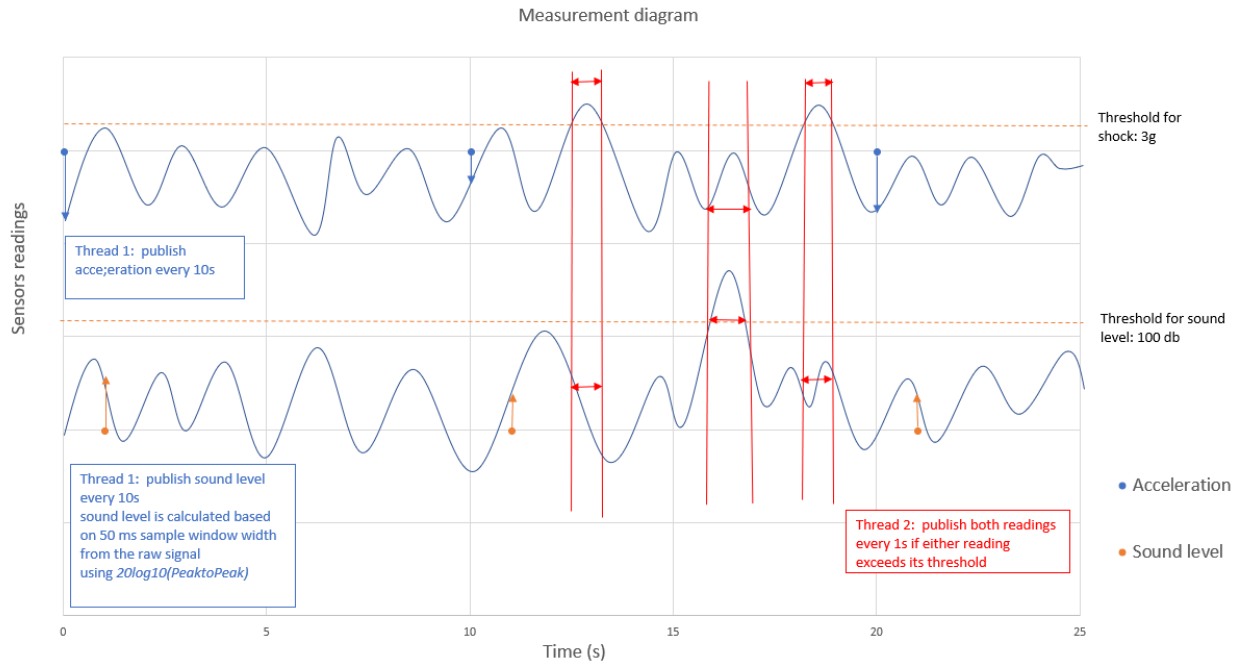


2) Program logical flowchart



*current f_{ADC} varies with max(f_{ADC}) = 1MHz

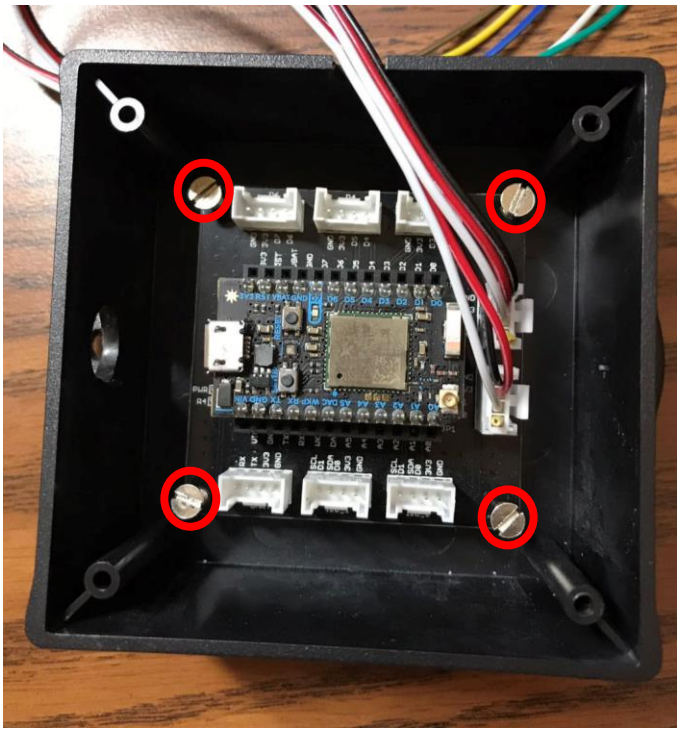
Measurement Diagram



Sensor Pack Assembly/disassembly procedure

The following assembly/disassembly procedure provides a direct guide for the user to follow in assembling hardware sensor packs. Basic tools needed include a flathead screwdriver, adhesive silicon RTV, wire strippers, and wire crimping tools. Personnel should have basic knowledge of common electrical wiring practice and is designed for a novice engineer or technician to implement. Implementers should assemble the hardware package first, then follow the below instructions to initialize the microcontroller device and deploy the code onto the microcontroller. The time to assemble a complete sensor pack, setup the microcontroller and deploy the associated code is less than one hour.

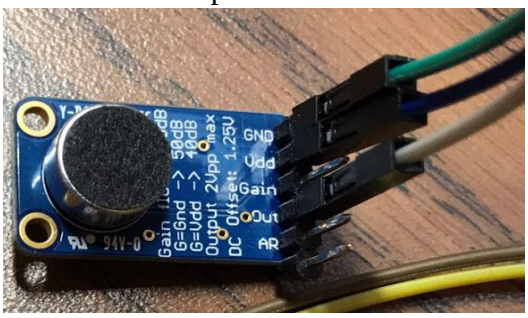
1. Adhere magnetic pads to bottom of casing using silicon RTV. These magnetic pads are to provide direct mounting capability of the overall casing to a machine chassis and/or sheet metal frame. Alternative options for mounting exist including semi-permanent adhesive (double-sided tape), suction-based mounting and use of hard fasteners to an existing hole assembly on the mounting surface for the machine. The magnetic base option is perhaps the most flexible approach and allows for complete sealing of the casing with benefit of reducing ingress of debris and fluids into the sensor pack itself.
2. Fasten Particle Groove Shield to casing using 4 machine screws. The mounting holes for the Particle Groove Shield are marked with red circles in the below figure. The mounting will secure the Particle Groove Shield to the injection molded casing (black body).



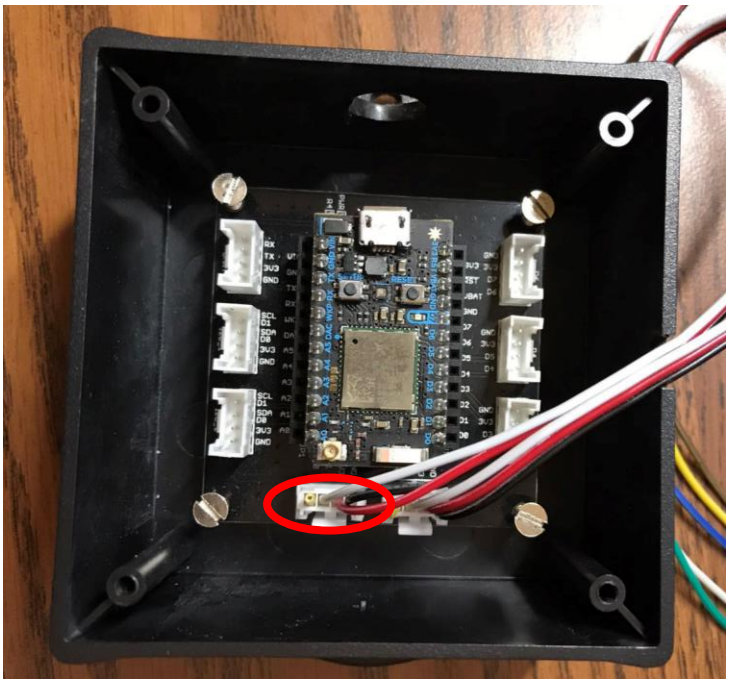
3. Plug in Photon to Groove Shield. The Photon plugs directly onto the center of the Groove Shield in the receiver holes and mounts as shown in the above figure.
4. Wire microphone as shown in the below figure and attach the wiring to the A1 analog port on the Particle Grove Shield (marked on Shield board). The wiring diagram for this connection is as follows:

   Microphone connection:
   Photon end – microphone end
   GND -- GND
   3V3 -- Vdd
   A1 -- Out

   Picture of microphone end connection:



   With the usb cord inlet hole facing upward, the clipper connects to bottom left slot.



5. Wire accelerometer as shown in the below figure and attach the wiring to the A5 analog port on the Particle Grove Shield (marked on Shield board). The wiring diagram for this connection is as follows:

   Accelerometer connection

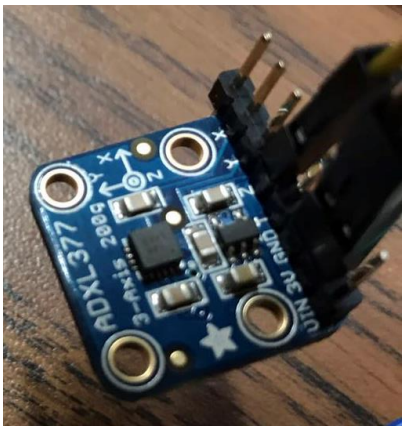
   Photon end – accelerometer end

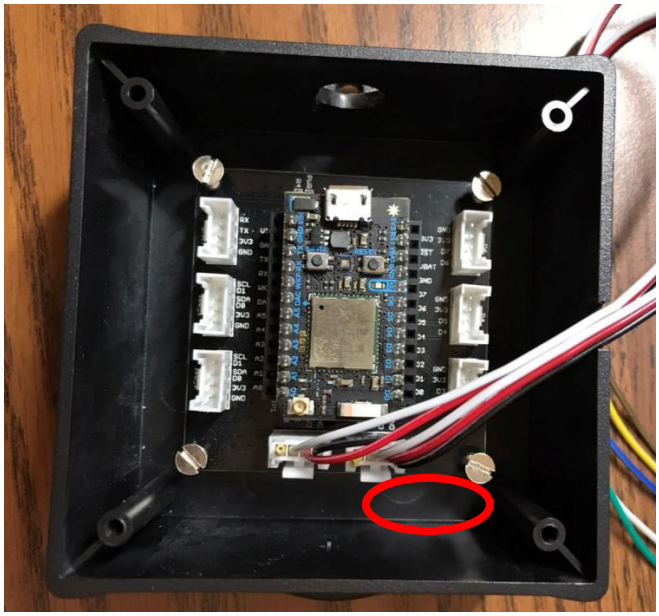
   A5 -- Z

   GND -- GND

   3V3 -- 3V

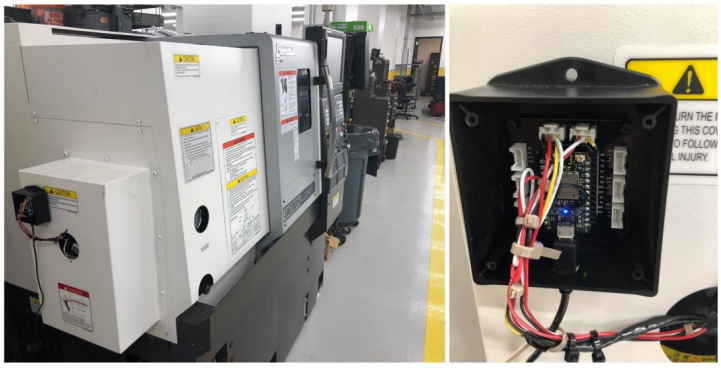
   Picture of accelerometer end connection:





With the usb cord inlet hole facing upward, the clipper connects to bottom right slot.

6. Attach sensor package to the machine to be monitored and route sensors as required to measurement position. The sensors should be mounted in an area where the sensors can detect the anomalies associated with the spindle. This can be on a exterior machine cover near the spindle motor. If possible, machine covers can be removed to place the sensors directly onto the spindle and axis motor housings. Caution should be exercised in routing of wires to the accelerometer to avoid potential snagging of wires on any translating/moving machine elements such as machine ways and slide covers. The sensor package casing should be placed on the machine per the selected attachment mode (e.g., magnetic, adhesive, mechanical fastener, suction) in an area with sufficient range to access a wireless access point. Indicator lights on the Photon microcontroller indicate access to wireless networks and connectivity to the cloud-hosted database will indicate this state. An example machine routing is shown in the below image for attachment to an Okuma Genoa lathe, where the spindle cover has been removed to allow attachment of the sensor to the spindle motor housing. The sensor kit is magnetically attached to the exterior of the machine.



Low Bandwidth Sensor Pack

Specifications

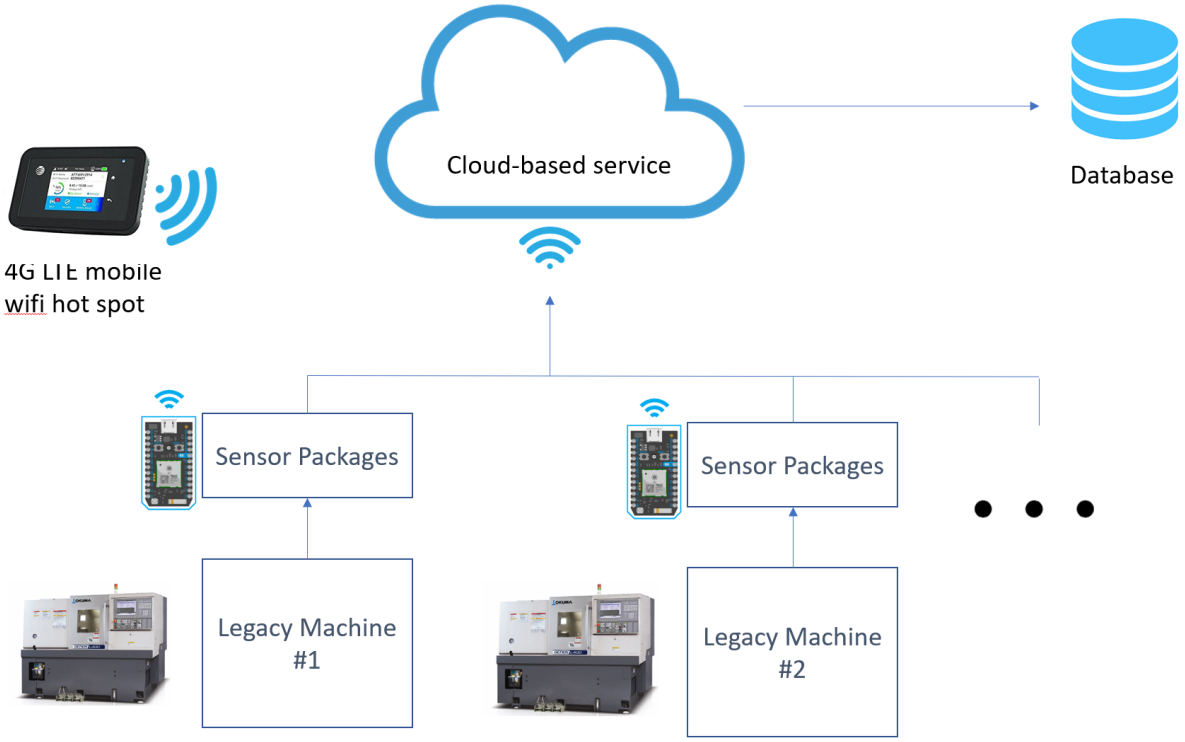
- Photon spec
    - Particle PØ Wi-Fi module
        - Broadcom BCM43362 Wi-Fi chip
        - 802.11b/g/n Wi-Fi
        - STM32F205RGY6 120Mhz ARM Cortex M3
        - 1MB flash, 128KB RAM
    - On-board RGB status LED (ext. drive provided)
    - 18 Mixed-signal GPIO and advanced peripherals
    - Open source design
    - Real-time operating system (FreeRTOS)
    - Soft AP setup
    - FCC, CE and IC certified
- Sensor spec
    - ADXL203
        - 3-axis accelerometer
        - +- 200g
    - MAX31855
        - 14-Bit, 0.25°C Resolution Converter
        - Common Thermocouple types supported
- Assembled pack
    - Weight: 173g (w/o usb cord or adaptor)
    - Dimension (L x W x H): 12cm x 6cm x 4.5cm

Bill of materials (Total Cost = $142.44)

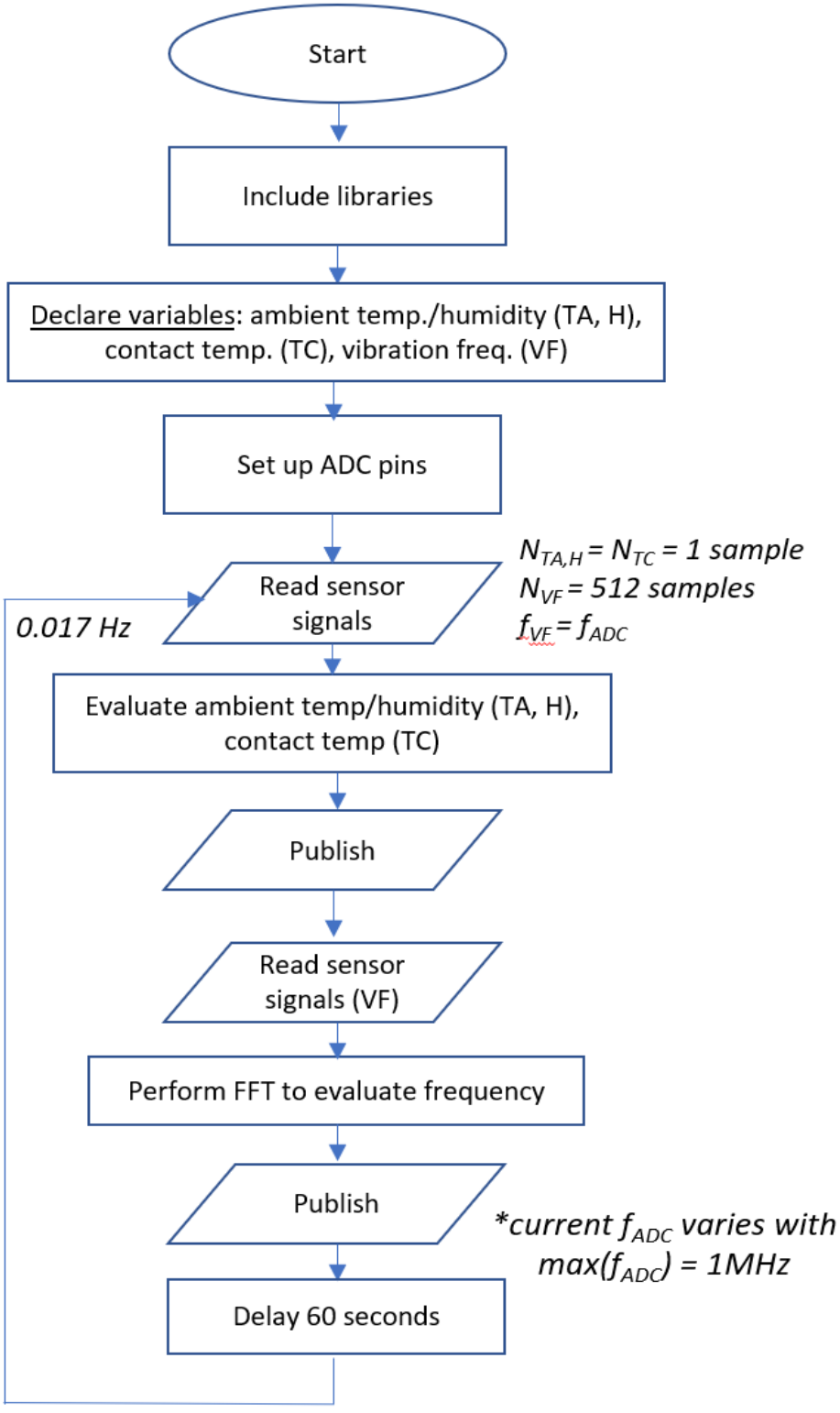
Category	Item	Type	Description	Supplier	Unit Price
Platform and Shield	PARTICLE PHOTON	Development platform	WiFi IoT Development Kit	Digikey	$19.00
	PARTICLE PHOTON BASE SHIELD	Grove Connectors for Particle	3 digital ports, 2 analog ports, 2 I2C ports and 1 UART port	Seeed	$6.90
Sensors	ADXL 203 EVAL BOARD	Accelerometer for vibration	2 Axis,Analog,ADXL203-2 Axis Accelerometer, ±1.7g Analog Out, 1000mV/g sensitivity	Digikey	$52.45
	DHT 22	Humidity/Temperature sensor	Humidity Temperature Sensor 0 ~ 100% RH Digital ±2% 2s Through Hole, -40°C ~ 80°C	Digikey	$9.95
	MAX31855 EVAL BOARD	Thermal management evaluation board	K Type Thermocouple Required	Digikey	$14.95
	THERMOCOUPLE WIRE K-TYPE 1M	K - Type Thermocouple	-73°C ~ 482°C, 3.281' (1m), Wire Terminal	Digikey	$9.99
Power Supply	USB 2.0 A-PLUG TO MCRO-B-PLUG	USB cable	USB 2.0 Cable A Male to Micro B Male 16.40' (5.00m) Shielded	Digikey	$9.22
	AC/DC WALL MOUNT ADAPTER 5V 10W	Power adapter	5V 10W AC/DC External Wall Mount Adapter Fixed Blade Input	Digikey	$7.38
Casing	BOX PLSTC GRAY/CLR 4.53"LX3.54"W	Sensor package case	Box with Mounting Flange Plastic, Clear Cover/Door Cover Included 4.528" L x 3.543" W	Digikey	$12.60

System Architecture

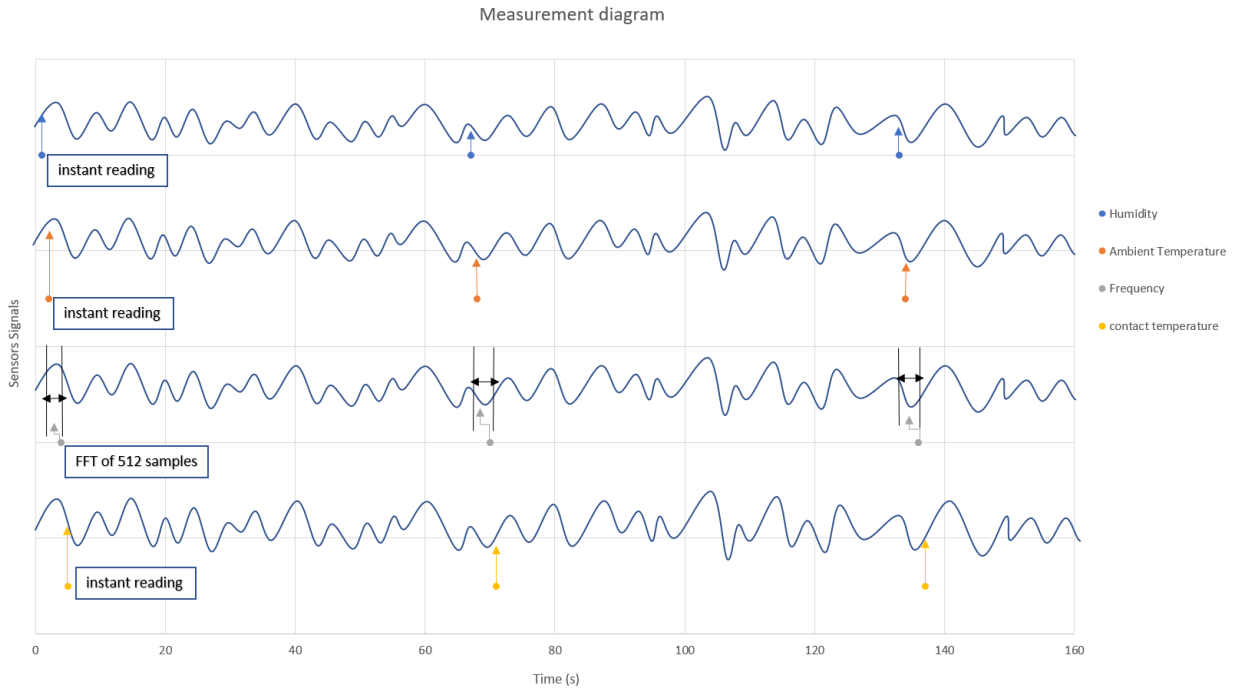
1) Physical system diagram



2) Program logical flowchart



Data measurement diagram



Period for each loop is 66 seconds, within the loop, the sensors reading is sequential, only the frequency reading is based on the FFT results of 512 samples collected in the time window shown in figure above, the other sensors readings are all instant readings.

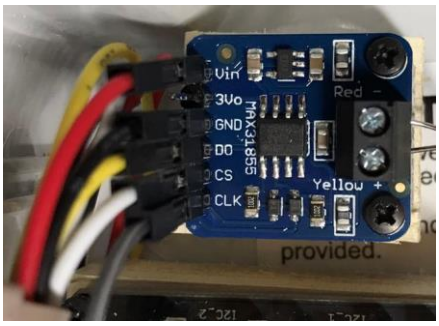
Sensor Pack Assembly/disassembly procedure

1. Adhere magnetic pads to bottom of casing using silicon RTV
2. Fasten Particle Groove Shield to a base using 4 machine screws
3. Adhere base to casing using silicon RTV
4. Plug photon into grove shield
5. Fasten breakout board to a base using 4 machine screws
6. Adhere base to casing using silicon RTV
7. Wire breakout board as show and attach to digital port on grove shield

   Photon end – thermocouple breakout board end

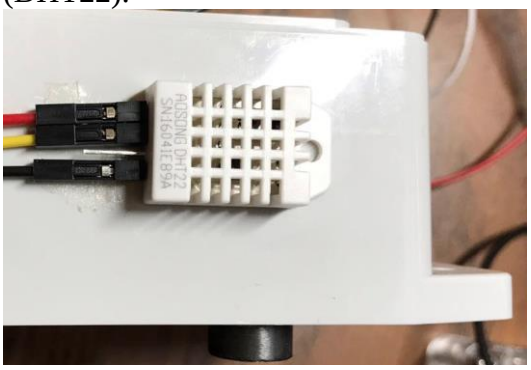
   D1 -- CLK
   D2 -- D0
   D3 -- CS
   3V3 -- Vin
   GND -- GND

   Picture of breakout broad end connection:



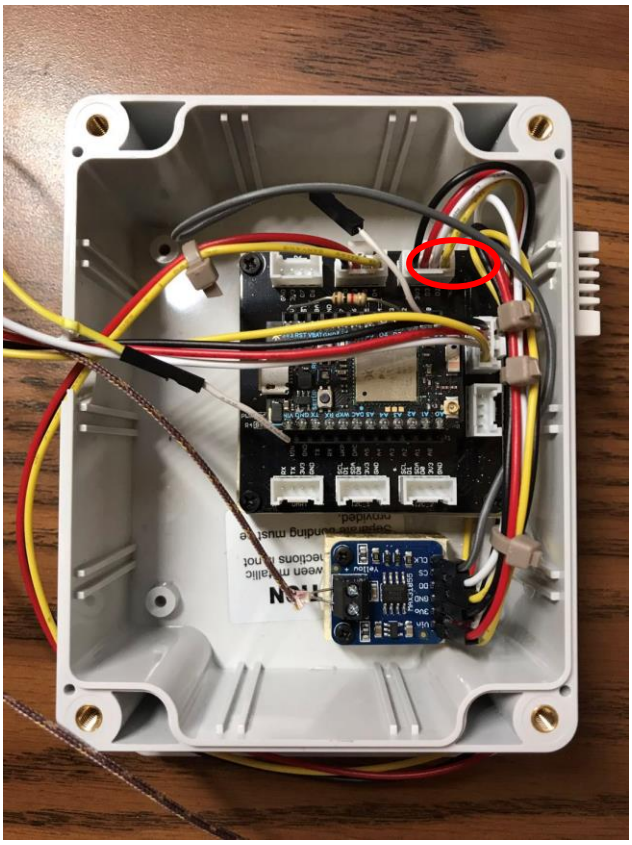
With the package placed as shown, the clip connects to upper right slot on photon shield.
Note: D1(photon) – CLK (breakout board) is a separated single wire connection.

8. Wire DHT22 as shown and attach to digital port on particle grove shield. The white module on the outer surface of the package is the temperature and humidity sensor (DHT22).



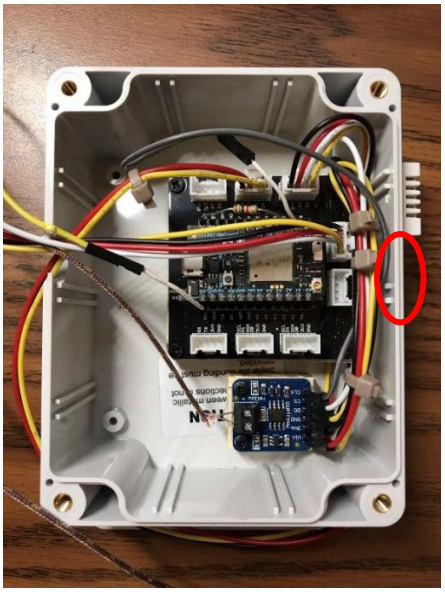
Photon end	– DHT22 end (top to bottom)
3V3	-- 1
D4	-- 2
GND	-- 4

With the package placed as shown, the clip connects to the slot in red circle. Note: There is a resistor connected from 3V3 to D4 for signal pullup for DHT22



9. Wire accelerometer as shown and attach to digital port on Particle grove shield

   Photon end -- accelerometer end
   A5 -- Y
   GND -- G
   Vin -- V+
   With photon placed as shown, the clip connects to the slot in red circle. Note: Vin (Photon end) to V+ is a separated single wire connection.



10. Close the lid and tighten 4 screws

11. Attach sensor package to the machine to be monitored and route sensors as required to measurement position. The sensors should be mounted in an area where the sensors can detect the anomalies associated with the spindle. This can be on a exterior machine cover near the spindle motor. If possible, machine covers can be removed to place the sensors directly onto the spindle and axis motor housings. Caution should be exercised in routing

of wires to the accelerometer to avoid potential snagging of wires on any translating/moving machine elements such as machine ways and slide covers. The sensor package casing should be placed on the machine per the selected attachment mode (e.g., magnetic, adhesive, mechanical fastener, suction) in an area with sufficient range to access a wireless access point. Indicator lights on the Photon microcontroller indicate access to wireless networks and connectivity to the cloud-hosted database will indicate this state.

Getting started with Particle Photon (Wi-Fi enabled microcontroller)

The following steps are a quick starting guide on how to set up a Particle Photon. For detailed instructions please visit: https://docs.particle.io/quickstart/photon/.

- **Power On Your Device**
  - After unboxing the Photon, power it up with a micro USB cable connected to a computer or a 5V power supply adapter
  - As soon as it is plugged in, the RGB LED on your device should begin blinking blue. If your device is not blinking blue, hold down the SETUP button.
- **Install the Particle app to your smartphone**
  - You can download the app on either Android or iOS smartphones with the links below:
  - **https://itunes.apple.com/us/app/particle-iot/id991459054?mt=8**
  - **https://play.google.com/store/apps/details?id=io.particle.android.app&hl=en_US**
- **Connect your Particle Photon to the Internet using your smartphone**
  - Using the app, create a Particle account and follow the instructions to connect to your Photon and provide the Wi-Fi credentials. Once connected, the Photon will remember the information and will connect to Wi-Fi automatically. If the device is successfully connected to the router and to the Internet, you should see a breathing cyan color on the RGB LED.
- **Upload the sketch to the Particle Photon**
  - Go to build.particle.io with your computer and login with your Particle credentials
  - To double check if everything is working fine, you can use an example project from the Particle IDE e.g. “Blink an LED”, and Flash your Particle Photon. If the code is successfully gets flashed to the microcontroller, you should see a Blue LED on the Photon blinking every second
  - Now, create a blank app and copy paste the provided code in the next section to publish the desired sensor data to the cloud. In order to add a new header/cpp file (.h/.cpp) you can click on the plus sign on the top right corner of the Particle IDE.
  - By Flashing the code to the Photon, you should start receiving the data to the specified database
- **Troubleshooting**
  - If you faced any difficulties during the setting up process, you can troubleshoot the issue by visiting the following link: https://docs.particle.io/tutorials/device-os/led/photon

Source Code

The following sections of code will provide functionality for: (1) a high-bandwidth sensor pack designed to detect rapid acceleration events associated with spindle crashes and/or tool impacts and (2) a low-bandwidth sensor pack designed to detect spindle states and communicate other sensor information (e.g., motor temperature, air temperature.humidity) at a low measurement frequency. The algorithm and logical design of each of the code sets is explained in the flowcharts provided above. The primary differences between these two sensor packs are in the ability of the high-bandwidth sensor pack to detect events that occur on a rapid timescale, whereas the low-bandiwidth pack simply polls the sensors with a longer time interval where detection of rapid events is not critical. The code blocks are provided in a fully deplorable form and the user should follow the instructions on the previous page to initialize the microcontroller platform and access the development environment where the code following this in the (1) high bandwidth sensor pack and (2) low bandwidth sensor pack sections can be pasted for web-based deployment of the code to the microcontroller. It should be noted that the following each of the code sections (e.g., high bandwidth, low bandwidth) are to each be deployed on a dedicated microcontroller and are not designed to provide dual functionality to a single microcontroller..

1) High bandwidth sensor pack

```
// This #include statement was automatically added by the Particle IDE.
#include "publishMQTT.h"
#include "math.h"
double thresholdacc = 3;//change this value for alert threshold of accelerometer, in g
double thresholdnoise = 100;//change this value for alert threshold of noise level, in dB
#define mic A1
#define acc A5
String assetId = "OkumaGenosSensors";
Timer timer(10000, print_every_ten_second);
double accg = 0.0;
double noise = 0.0;
void setup() {
   pinMode(mic,INPUT);
   pinMode(acc,INPUT);
   timer.start();
}
void loop()
{
   accg = analogRead(5);
   accg = (accg/4095) * 1.7;
   //accg = ((accg-2000)/4096*3300/7.2);
   noise = noiseValue();
   noise = noise*0.75;
  // //------------------------------------------------------------------ noise and acceleration threshold
   if(accg>=thresholdacc || noise>=thresholdnoise)
   {
```

```
      Particle.publish("acc", String(accg));
      publishMQTT(assetId, "acc", String(accg));
    }
    delay(1000);
}
// publish a message every 10 seconds
void print_every_ten_second()
{
    Particle.publish("acc",String(accg));
    Particle.publish("soundlevel",String(noise));
}
float noiseValue()
{
  int sampleWindow = 50; // Sample window width. 50ms = 20Hz
  int signalMax = 0;
  int signalMin = 4096;
  unsigned long startMillis = millis();
  unsigned int sample;
  unsigned int peakToPeak;
  while (millis() - startMillis < sampleWindow) {
    sample = analogRead(A1);
    if (sample < 4096) {
      if (sample > signalMax) {
        signalMax = sample;
      } else if (sample < signalMin) {
        signalMin = sample;
      }
    }
  }
  peakToPeak = signalMax - signalMin;
  return 20 * log(peakToPeak);
}
```

2) Low bandwidth sensor pack

```
// This #include statement was automatically added by the Particle IDE.
#include "publishMQTT.h"
// This #include statement was automatically added by the Particle IDE.
#include "arduinoFFT.h"
// This #include statement was automatically added by the Particle IDE.
#include <HttpClient.h>
// This #include statement was automatically added by the Particle IDE.
#include <Adafruit_MAX31855.h>
// This #include statement was automatically added by the Particle IDE.
#include <HC_SR04.h>
// This #include statement was automatically added by the Particle IDE.
#include <PietteTech_DHT.h>
```

```
// This #include statement was automatically added by the Particle IDE.
#include "insertToDatabase.h"
unsigned long delaytime = 20000; //change delaytime to change the delay time between each loop in microseconds, use this variable to change the publish rate to database
String assetId = "OKUMAGENOS";// change this variable for the assetID
#define SAMPLES 512             //change this variable for sampling size of FFT, Must be a power of 2
#define SAMPLING_FREQUENCY 500 //change this variable for changing the sampling frequency of FFT, Hz, must be less than 10000 due to ADC
//Ultrasonic sensor: trigpin to D4, echopin to D5
//DHT22 Sensor: DHT pin to D6
/////////set up variables of thermocoupler//////
int thermoCLK = D1;
int thermoCS = D3;
int thermoDO = D2;
Adafruit_MAX31855 thermocouple(thermoCLK, thermoCS, thermoDO);
double cm = 0.0;
double inches = 0.0;
// int trigPin = D4;
// int echoPin = D5;
// HC_SR04 rangefinder = HC_SR04(trigPin, echoPin, 5, 250);
#define DHTTYPE  DHT22       // Sensor type DHT11/21/22/AM2301/AM2302
#define DHTPIN   D4 //6         // Digital pin for communications
PietteTech_DHT DHT(DHTPIN, DHTTYPE);
int n;
arduinoFFT FFT = arduinoFFT();
unsigned int sampling_period_us;
unsigned long microseconds;
double vReal[SAMPLES];
double vImag[SAMPLES];
void setup()
{
   Serial.begin(9600);
//   Spark.variable("cm", &cm, DOUBLE);
//   Spark.variable("inches", &inches, DOUBLE);
  sampling_period_us = round(1000000*(1.0/SAMPLING_FREQUENCY));
}
void loop()
{
   int result = DHT.acquireAndWait(1000); // wait up to 1 sec (default indefinitely)
//  switch (result) {
//   case DHTLIB_OK:
//     Serial.println("OK");
//     break;
//   case DHTLIB_ERROR_CHECKSUM:
//     Serial.println("Error\n\r\tChecksum error");
```

```
//     break;
//   case DHTLIB_ERROR_ISR_TIMEOUT:
//     Serial.println("Error\n\r\tISR time out error");
//     break;
//   case DHTLIB_ERROR_RESPONSE_TIMEOUT:
//     Serial.println("Error\n\r\tResponse time out error");
//     break;
//   case DHTLIB_ERROR_DATA_TIMEOUT:
//     Serial.println("Error\n\r\tData time out error");
//     break;
//   case DHTLIB_ERROR_ACQUIRING:
//     Serial.println("Error\n\r\tAcquiring");
//     break;
//   case DHTLIB_ERROR_DELTA:
//     Serial.println("Error\n\r\tDelta time to small");
//     break;
//   case DHTLIB_ERROR_NOTSTARTED:
//     Serial.println("Error\n\r\tNot started");
//     break;
//   default:
//     Serial.println("Unknown error");
// break;
//   }

//   Particle.publish("Air Humidity ",String(DHT.getHumidity()));
  insertToDatabase(assetId, "Humidity", String(DHT.getHumidity()));
  //publishMQTT(assetId, "Humidity", String(DHT.getHumidity()));
  Serial.println("Humidity," + String(DHT.getHumidity()));
  delay(1000);
//   Particle.publish("Ambient Temperature: ",String(DHT.getCelsius()));
  insertToDatabase(assetId, "Ambient-Temperature", String(DHT.getCelsius()));
  //publishMQTT(assetId, "Ambient Temperature", String(DHT.getCelsius()));
  Serial.println("Ambient-Temperature," + String(DHT.getCelsius()));
  delay(1000);
  n++;
     /*SAMPLING*/
   for(int i=0; i<SAMPLES; i++)
   {
      microseconds = micros();
      double value = analogRead(A5);//Overflows after around 70 minutes!
      value = (((value*5.0)/4095) - 2.50 );
      vReal[i] =value;
      vImag[i] = 0;
      while(micros() < (microseconds + sampling_period_us)){
      }
   }
```

```
    /*FFT*/
   FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
   FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);
   FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);
   double peak = FFT.MajorPeak(vReal, SAMPLES, SAMPLING_FREQUENCY);
   /*PRINT RESULTS*/
   // Particle.publish("frequency",String(peak));     //Print out what frequency is the most
dominant.
   insertToDatabase(assetId, "Frequency", String(peak));
   //publishMQTT(assetId, "Frequency", String(peak));
   Serial.println("Frequency," + String(peak));
   delay(1000);
  double c = thermocouple.readCelsius();
  if (isnan(c)) {
   //  Particle.publish("Something wrong with thermocouple!");
  }
  else {
   //  Particle.publish("contact temperature", String(c));
    insertToDatabase(assetId,"Contact-Temperature", String(c));
    //publishMQTT(assetId,"Contact Temperature", String(c));
    Serial.println("Contact-Temperature," + String(c));
    delay(1000);
  }
   // cm = rangefinder.getDistanceCM();
//   Particle.publish("liquid level (cm)", String(cm));
//   insertToDatabase(assetId, "liquidLevel", String(cm));
  delay(delaytime);
}
```

3) Supporting files

ArduinoFFT.cpp

```
/*
        FFT libray
        Copyright (C) 2010 Didier Longueville
        Copyright (C) 2014 Enrique Condes
        This program is free software: you can redistribute it and/or modify
        it under the terms of the GNU General Public License as published by
        the Free Software Foundation, either version 3 of the License, or
        (at your option) any later version.
        This program is distributed in the hope that it will be useful,
        but WITHOUT ANY WARRANTY; without even the implied warranty of
        MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
        GNU General Public License for more details.
        You should have received a copy of the GNU General Public License
        along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
```

```
#include "arduinoFFT.h"

arduinoFFT::arduinoFFT(void)
{ // Constructor
	#warning("This method is deprecated and will be removed on future revisions.")
}
arduinoFFT::arduinoFFT(double *vReal, double *vImag, uint16_t samples, double
samplingFrequency)
{// Constructor
	this->_vReal = vReal;
	this->_vImag = vImag;
	this->_samples = samples;
	this->_samplingFrequency = samplingFrequency;
	this->_power = Exponent(samples);
}
arduinoFFT::~arduinoFFT(void)
{
// Destructor
}
uint8_t arduinoFFT::Revision(void)
{
	return(FFT_LIB_REV);
}
void arduinoFFT::Compute(double *vReal, double *vImag, uint16_t samples, uint8_t dir)
{
	#warning("This method is deprecated and will be removed on future revisions.")
	Compute(vReal, vImag, samples, Exponent(samples), dir);
}

void arduinoFFT::Compute(uint8_t dir)
{// Computes in-place complex-to-complex FFT /
	// Reverse bits /
	uint16_t j = 0;
	for (uint16_t i = 0; i < (this->_samples - 1); i++) {
		if (i < j) {
			Swap(&this->_vReal[i], &this->_vReal[j]);
			if(dir==FFT_REVERSE)
				Swap(&this->_vImag[i], &this->_vImag[j]);
		}
		uint16_t k = (this->_samples >> 1);
		while (k <= j) {
			j -= k;
			k >>= 1;
		}
		j += k;
```

```
	}
	// Compute the FFT  /
	double c1 = -1.0;
	double c2 = 0.0;
	uint16_t l2 = 1;
	for (uint8_t l = 0; (l < this->_power); l++) {
		uint16_t l1 = l2;
		l2 <<= 1;
		double u1 = 1.0;
		double u2 = 0.0;
		for (j = 0; j < l1; j++) {
			for (uint16_t i = j; i < this->_samples; i += l2) {
				uint16_t i1 = i + l1;
				double t1 = u1 * this->_vReal[i1] - u2 * this->_vImag[i1];
				double t2 = u1 * this->_vImag[i1] + u2 * this->_vReal[i1];
				this->_vReal[i1] = this->_vReal[i] - t1;
				this->_vImag[i1] = this->_vImag[i] - t2;
				this->_vReal[i] += t1;
				this->_vImag[i] += t2;
			}
			double z = ((u1 * c1) - (u2 * c2));
			u2 = ((u1 * c2) + (u2 * c1));
			u1 = z;
		}
		c2 = sqrt((1.0 - c1) / 2.0);
		if (dir == FFT_FORWARD) {
			c2 = -c2;
		}
		c1 = sqrt((1.0 + c1) / 2.0);
	}
	// Scaling for reverse transform /
	if (dir != FFT_FORWARD) {
		for (uint16_t i = 0; i < this->_samples; i++) {
			this->_vReal[i] /= this->_samples;
			this->_vImag[i] /= this->_samples;
		}
	}
}

void arduinoFFT::Compute(double *vReal, double *vImag, uint16_t samples, uint8_t power,
uint8_t dir)
{	// Computes in-place complex-to-complex FFT
	// Reverse bits
	#warning("This method is deprecated and will be removed on future revisions.")
	uint16_t j = 0;
	for (uint16_t i = 0; i < (samples - 1); i++) {
```

```
        if (i < j) {
                Swap(&vReal[i], &vReal[j]);
                if(dir==FFT_REVERSE)
                        Swap(&vImag[i], &vImag[j]);
        }
        uint16_t k = (samples >> 1);
        while (k <= j) {
                j -= k;
                k >>= 1;
        }
        j += k;
}
// Compute the FFT
double c1 = -1.0;
double c2 = 0.0;
uint16_t l2 = 1;
for (uint8_t l = 0; (l < power); l++) {
        uint16_t l1 = l2;
        l2 <<= 1;
        double u1 = 1.0;
        double u2 = 0.0;
        for (j = 0; j < l1; j++) {
                 for (uint16_t i = j; i < samples; i += l2) {
                                uint16_t i1 = i + l1;
                                double t1 = u1 * vReal[i1] - u2 * vImag[i1];
                                double t2 = u1 * vImag[i1] + u2 * vReal[i1];
                                vReal[i1] = vReal[i] - t1;
                                vImag[i1] = vImag[i] - t2;
                                vReal[i] += t1;
                                vImag[i] += t2;
                 }
                 double z = ((u1 * c1) - (u2 * c2));
                 u2 = ((u1 * c2) + (u2 * c1));
                 u1 = z;
        }
        c2 = sqrt((1.0 - c1) / 2.0);
        if (dir == FFT_FORWARD) {
                c2 = -c2;
        }
        c1 = sqrt((1.0 + c1) / 2.0);
}
// Scaling for reverse transform
if (dir != FFT_FORWARD) {
        for (uint16_t i = 0; i < samples; i++) {
                 vReal[i] /= samples;
                 vImag[i] /= samples;
```

```
                }
        }
}

void arduinoFFT::ComplexToMagnitude()
{ // vM is half the size of vReal and vImag
        for (uint16_t i = 0; i < this->_samples; i++) {
                this->_vReal[i] = sqrt(sq(this->_vReal[i]) + sq(this->_vImag[i]));
        }
}

void arduinoFFT::ComplexToMagnitude(double *vReal, double *vImag, uint16_t samples)
{       // vM is half the size of vReal and vImag
        #warning("This method is deprecated and will be removed on future revisions.")
        for (uint16_t i = 0; i < samples; i++) {
                vReal[i] = sqrt(sq(vReal[i]) + sq(vImag[i]));
        }
}

void arduinoFFT::Windowing(uint8_t windowType, uint8_t dir)
{// Weighing factors are computed once before multiple use of FFT
// The weighing function is symetric; half the weighs are recorded
        double samplesMinusOne = (double(this->_samples) - 1.0);
        for (uint16_t i = 0; i < (this->_samples >> 1); i++) {
                double indexMinusOne = double(i);
                double ratio = (indexMinusOne / samplesMinusOne);
                double weighingFactor = 1.0;
                // Compute and record weighting factor
                switch (windowType) {
                case FFT_WIN_TYP_RECTANGLE: // rectangle (box car)
                        weighingFactor = 1.0;
                        break;
                case FFT_WIN_TYP_HAMMING: // hamming
                        weighingFactor = 0.54 - (0.46 * cos(twoPi * ratio));
                        break;
                case FFT_WIN_TYP_HANN: // hann
                        weighingFactor = 0.54 * (1.0 - cos(twoPi * ratio));
                        break;
                case FFT_WIN_TYP_TRIANGLE: // triangle (Bartlett)
                        weighingFactor = 1.0 - ((2.0 * abs(indexMinusOne - (samplesMinusOne /
2.0))) / samplesMinusOne);
                        break;
                case FFT_WIN_TYP_BLACKMAN: // blackmann
                        weighingFactor = 0.42323 - (0.49755 * (cos(twoPi * ratio))) + (0.07922 *
(cos(fourPi * ratio)));
                        break;
```

```
                case FFT_WIN_TYP_FLT_TOP: // flat top
                        weighingFactor = 0.2810639 - (0.5208972 * cos(twoPi * ratio)) +
(0.1980399 * cos(fourPi * ratio));
                        break;
                case FFT_WIN_TYP_WELCH: // welch
                        weighingFactor = 1.0 - sq((indexMinusOne - samplesMinusOne / 2.0) /
(samplesMinusOne / 2.0));
                        break;
                }
                if (dir == FFT_FORWARD) {
                        this->_vReal[i] *= weighingFactor;
                        this->_vReal[this->_samples - (i + 1)] *= weighingFactor;
                }
                else {
                        this->_vReal[i] /= weighingFactor;
                        this->_vReal[this->_samples - (i + 1)] /= weighingFactor;
                }
        }
}
void arduinoFFT::Windowing(double *vData, uint16_t samples, uint8_t windowType, uint8_t
dir)
{// Weighing factors are computed once before multiple use of FFT
// The weighing function is symetric; half the weighs are recorded
        #warning("This method is deprecated and will be removed on future revisions.")
        double samplesMinusOne = (double(samples) - 1.0);
        for (uint16_t i = 0; i < (samples >> 1); i++) {
                double indexMinusOne = double(i);
                double ratio = (indexMinusOne / samplesMinusOne);
                double weighingFactor = 1.0;
                // Compute and record weighting factor
                switch (windowType) {
                case FFT_WIN_TYP_RECTANGLE: // rectangle (box car)
                        weighingFactor = 1.0;
                        break;
                case FFT_WIN_TYP_HAMMING: // hamming
                        weighingFactor = 0.54 - (0.46 * cos(twoPi * ratio));
                        break;
                case FFT_WIN_TYP_HANN: // hann
                        weighingFactor = 0.54 * (1.0 - cos(twoPi * ratio));
                        break;
                case FFT_WIN_TYP_TRIANGLE: // triangle (Bartlett)
                        weighingFactor = 1.0 - ((2.0 * abs(indexMinusOne - (samplesMinusOne /
2.0))) / samplesMinusOne);
                        break;
                case FFT_WIN_TYP_BLACKMAN: // blackmann
```

```
                        weighingFactor = 0.42323 - (0.49755 * (cos(twoPi * ratio))) + (0.07922 *
(cos(fourPi * ratio)));
                        break;
                case FFT_WIN_TYP_FLT_TOP: // flat top
                        weighingFactor = 0.2810639 - (0.5208972 * cos(twoPi * ratio)) +
(0.1980399 * cos(fourPi * ratio));
                        break;
                case FFT_WIN_TYP_WELCH: // welch
                        weighingFactor = 1.0 - sq((indexMinusOne - samplesMinusOne / 2.0) /
(samplesMinusOne / 2.0));
                        break;
                }
                if (dir == FFT_FORWARD) {
                        vData[i] *= weighingFactor;
                        vData[samples - (i + 1)] *= weighingFactor;
                }
                else {
                        vData[i] /= weighingFactor;
                        vData[samples - (i + 1)] /= weighingFactor;
                }
        }
}

double arduinoFFT::MajorPeak()
{
        double maxY = 0;
        uint16_t IndexOfMaxY = 0;
        //If sampling_frequency = 2 * max_frequency in signal,
        //value would be stored at position samples/2
        for (uint16_t i = 1; i < ((this->_samples >> 1) + 1); i++) {
                if ((this->_vReal[i-1] < this->_vReal[i]) && (this->_vReal[i] > this-
>_vReal[i+1])) {
                        if (this->_vReal[i] > maxY) {
                                maxY = this->_vReal[i];
                                IndexOfMaxY = i;
                        }
                }
        }
        double delta = 0.5 * ((this->_vReal[IndexOfMaxY-1] - this->_vReal[IndexOfMaxY+1]) /
(this->_vReal[IndexOfMaxY-1] - (2.0 * this->_vReal[IndexOfMaxY]) + this-
>_vReal[IndexOfMaxY+1]));
        double interpolatedX = ((IndexOfMaxY + delta)  * this->_samplingFrequency) / (this-
>_samples-1);
        if(IndexOfMaxY==(this->_samples >> 1)) //To improve calculation on edge values
                interpolatedX = ((IndexOfMaxY + delta)  * this->_samplingFrequency) / (this-
>_samples);
```

```
	// retuned value: interpolated frequency peak apex
	return(interpolatedX);
}

double arduinoFFT::MajorPeak(double *vD, uint16_t samples, double samplingFrequency)
{
	#warning("This method is deprecated and will be removed on future revisions.")
	double maxY = 0;
	uint16_t IndexOfMaxY = 0;
	//If sampling_frequency = 2 * max_frequency in signal,
	//value would be stored at position samples/2
	for (uint16_t i = 1; i < ((samples >> 1) + 1); i++) {
		if ((vD[i-1] < vD[i]) && (vD[i] > vD[i+1])) {
			if (vD[i] > maxY) {
				maxY = vD[i];
				IndexOfMaxY = i;
			}
		}
	}
	double delta = 0.5 * ((vD[IndexOfMaxY-1] - vD[IndexOfMaxY+1]) /
(vD[IndexOfMaxY-1] - (2.0 * vD[IndexOfMaxY]) + vD[IndexOfMaxY+1]));
	double interpolatedX = ((IndexOfMaxY + delta)  * samplingFrequency) / (samples-1);
	if(IndexOfMaxY==(samples >> 1)) //To improve calculation on edge values
		interpolatedX = ((IndexOfMaxY + delta)  * samplingFrequency) / (samples);
	// returned value: interpolated frequency peak apex
	return(interpolatedX);
}

uint8_t arduinoFFT::Exponent(uint16_t value)
{
	#warning("This method will not be accessible on future revisions.")
	// Calculates the base 2 logarithm of a value
	uint8_t result = 0;
	while (((value >> result) & 1) != 1) result++;
	return(result);
}

// Private functions

void arduinoFFT::Swap(double *x, double *y)
{
	double temp = *x;
	*x = *y;
	*y = temp;
}
```

ArduinoFFT.h

```
/*
        FFT libray
        Copyright (C) 2010 Didier Longueville
        Copyright (C) 2014 Enrique Condes
        This program is free software: you can redistribute it and/or modify
        it under the terms of the GNU General Public License as published by
        the Free Software Foundation, either version 3 of the License, or
        (at your option) any later version.
        This program is distributed in the hope that it will be useful,
        but WITHOUT ANY WARRANTY; without even the implied warranty of
        MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
        GNU General Public License for more details.
        You should have received a copy of the GNU General Public License
        along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
#ifndef arduinoFFT_h /* Prevent loading library twice */
#define arduinoFFT_h
#ifdef ARDUINO
        #if ARDUINO >= 100
                #include "Arduino.h"
        #else
                #include "WProgram.h" /* This is where the standard Arduino code lies */
        #endif
#else
        #include <stdlib.h>
        #include <stdio.h>
        #ifdef __AVR__
                #include <avr/io.h>
        #endif
        #include <math.h>
        #include "defs.h"
#endif
#define FFT_LIB_REV 0x14
/* Custom constants */
#define FFT_FORWARD 0x01
#define FFT_REVERSE 0x00
/* Windowing type */
#define FFT_WIN_TYP_RECTANGLE 0x00 /* rectangle (Box car) */
#define FFT_WIN_TYP_HAMMING 0x01 /* hamming */
#define FFT_WIN_TYP_HANN 0x02 /* hann */
#define FFT_WIN_TYP_TRIANGLE 0x03 /* triangle (Bartlett) */
#define FFT_WIN_TYP_BLACKMAN 0x04 /* blackmann */
#define FFT_WIN_TYP_FLT_TOP 0x05 /* flat top */
#define FFT_WIN_TYP_WELCH 0x06 /* welch */
/*Mathematial constants*/
```

```
#define twoPi 6.28318531
#define fourPi 12.56637061
class arduinoFFT {
public:
	/* Constructor */
	arduinoFFT(void);
	arduinoFFT(double *vReal, double *vImag, uint16_t samples, double
samplingFrequency);
	/* Destructor */
	~arduinoFFT(void);
	/* Functions */
	uint8_t Revision(void);
	uint8_t Exponent(uint16_t value);
	void ComplexToMagnitude(double *vReal, double *vImag, uint16_t samples);
	void Compute(double *vReal, double *vImag, uint16_t samples, uint8_t dir);
	void Compute(double *vReal, double *vImag, uint16_t samples, uint8_t power, uint8_t
dir);
	double MajorPeak(double *vD, uint16_t samples, double samplingFrequency);
	void Windowing(double *vData, uint16_t samples, uint8_t windowType, uint8_t dir);
	void ComplexToMagnitude();
	void Compute(uint8_t dir);
	double MajorPeak();
	void Windowing(uint8_t windowType, uint8_t dir);
private:
	/* Variables */
	uint16_t _samples;
	double _samplingFrequency;
	double *_vReal;
	double *_vImag;
	uint8_t _power;
	/* Functions */
	void Swap(double *x, double *y);
};

#endif
```

Installation and Performance Report

Georgia Tech Precision Machining Research Consortium

I. Installation Report

Stand-alone wireless and wired plug-and-play sensor packages were developed for retrofit of legacy machines. The wireless sensor packages were built upon a Particle Photon device, which provides for low-cost connectivity for sensors to the cloud architecture in Figure 1. In some cases, the bandwidth required for sensing is greater than that capable within the architecture and for these the Particle Photon is used as an intermediary local processing unit for analysis before publishing the result to the cloud database. The project targeted development of sensor packages that can be integrated into a system network to specifically measure spindle/drive health (vibration), part/coolant/component temperature, coolant pH, tool wear and machine utilization. To enable a wired-based connection, the Particle Photon was paired to a Raspberry Pi platform across a USB interface. Figure A1 shows an RRK sensor package kit as assembled and deployed on an Okuma Genos lathe. The kit is contained within an IP-rated water-resistant housing and is affixed to the machine using a magnetic base. The kit is affixed outside of the spindle motor housing with accelerometers and temperature sensors led into the motor housing and attached to the outer motor housing casing. Figure A2 shows an RRK sensor package kit deployed at the Caterpillar facility.

To accommodate attachment of sensors to the machine tools of interest the project team removed exterior machine element covers to mount accelerometers and temperature sensors directly to the spindle housing. As in Figure 1, the spindle motor cover was removed and the accelerometer mounted inside the outer cover with the sensor pack attached to the exterior of the outer cover using magnetic attachment methods. It should be noted that the ideal location of the sensor of interest is closest to the phenomena to be measured. In the case of the accelerometer, measurement near the spindle motor and/or tooling would provide the maximum possible signal from the sensor. In some cases, the machine itself transmits these vibrations strongly through the machine casting and can be measured on the outer covers/panels of the machine. However, in other cases where machines may be very stiff platforms, location of the accelerometer may need to be close to the motor itself. Implementers should consult with internal maintenance teams on safe routing of cables if it is needed to access and route sensors toward the interior of the machine frame.

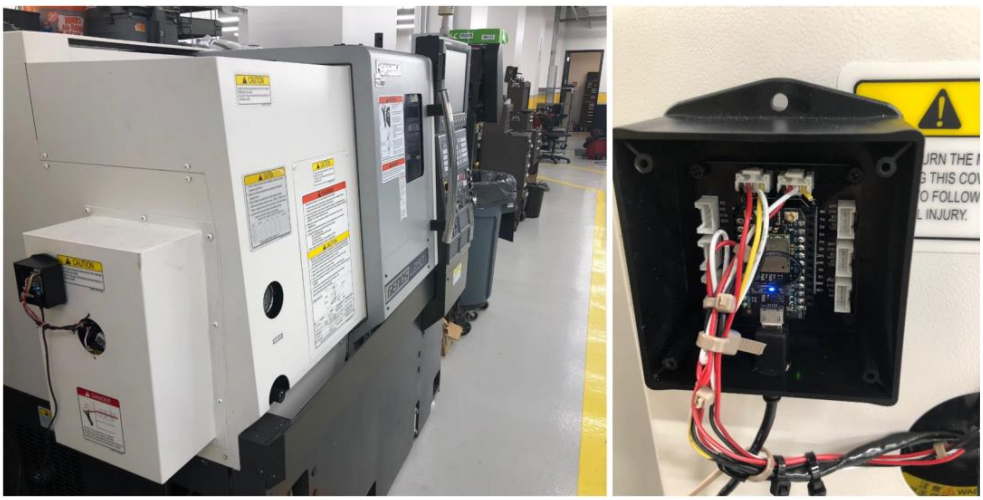


Figure A1. Photon-based reconfigurable retrofit kit sensor package at GT.

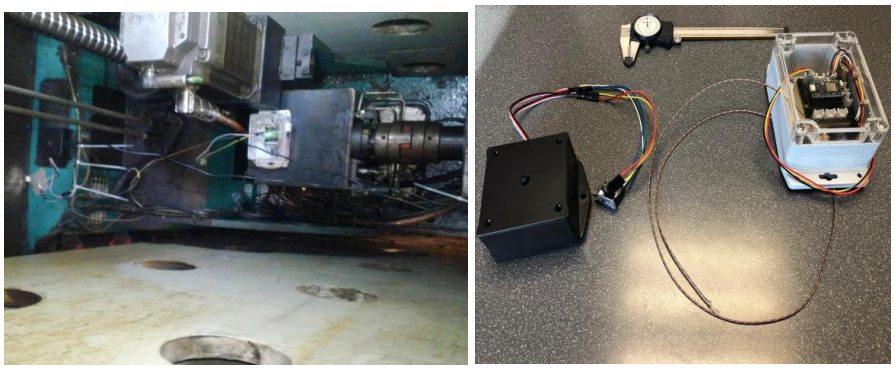


Figure A2. Photon-based reconfigurable retrofit kit sensor packages at Caterpillar.

In addition to determining ideal placement of sensors on the machine itself, connectivity of the microcontrollers to the networking infrastructure is important. Implementers should determine compatibility of the microcontroller with the wireless networking equipment within the facility prior to installation. Further, the strength of this network in terms of signal coverage should be verified before installing the sensor hardware.

II. Performance Report

The primary metrics for the RRK kit pertain to sensor validation as well as data requirements for data storage. Figure A3 shows test data from a controlled frequency measurement test of spindle operations on a machine tool at Georgia Tech using the RRK sensor package kit as installed from the previous reporting period. The test involves running the spindle motor of the machine at a range of spindle frequencies. The data shows that the RRK sensor package accelerometer was able to accurately measure the preset spindle motor frequency, which demonstrates its capability for tracking usage of the spindle. Figure A4 shows test data from controlled fluid level monitoring sensors as well as coolant temperature sensors. In this case, the fluid level monitoring sensor is an ultrasonic sensor and from the test data, the measured and actual changes to fluid level were identical. Coolant temperature was monitored using a thermocouple and validated in a boiling test. From the data, the boundary conditions do correspond to the expected behavior. Figure A5 shows results from a current monitoring test as well as a pressure sensor test. In this regard, the data also validate the sensor behavior.

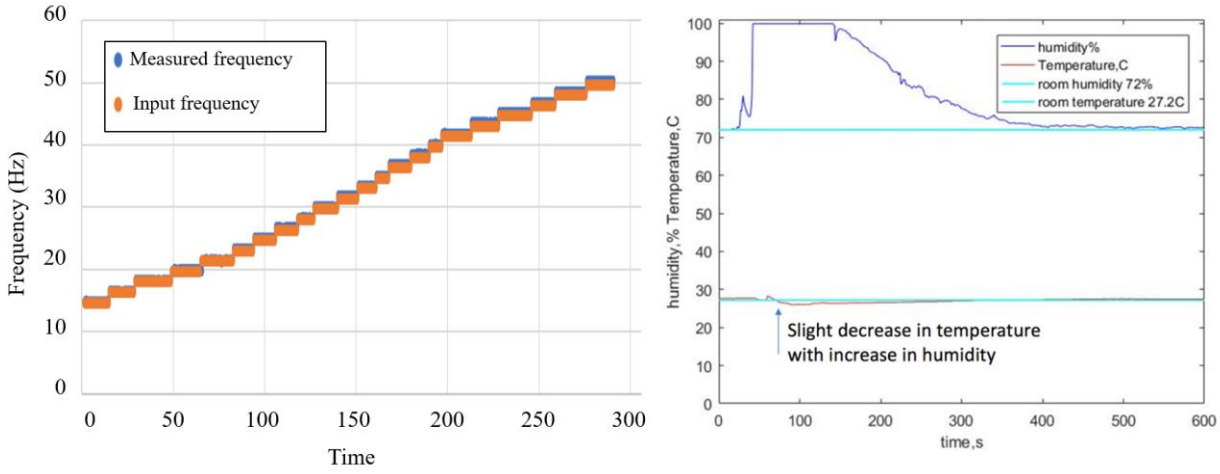


Figure A3. Sensor validation for spindle speed and humidity measurement.

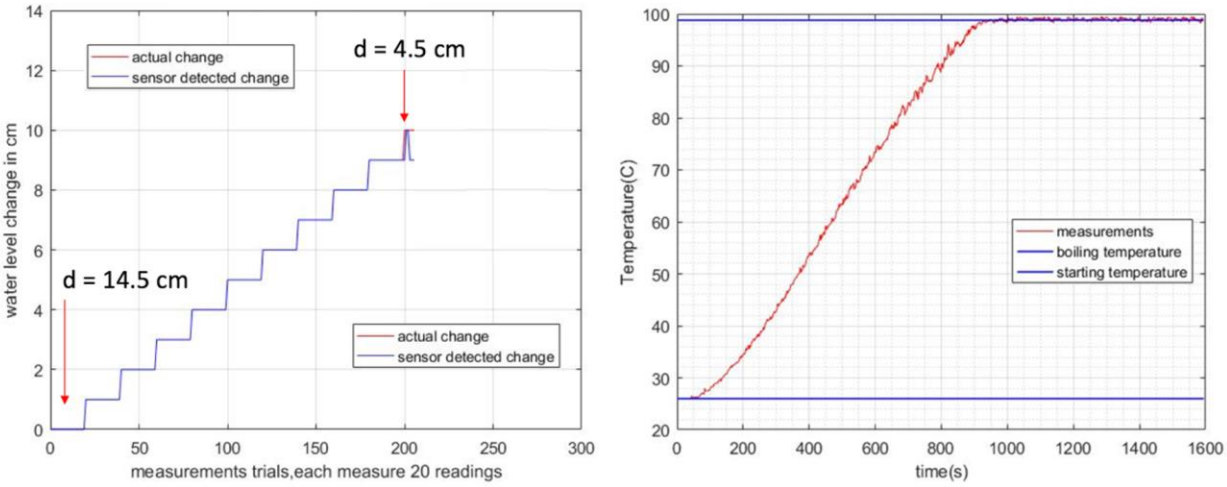


Figure A4. Sensor validation for fluid level and temperature monitoring.

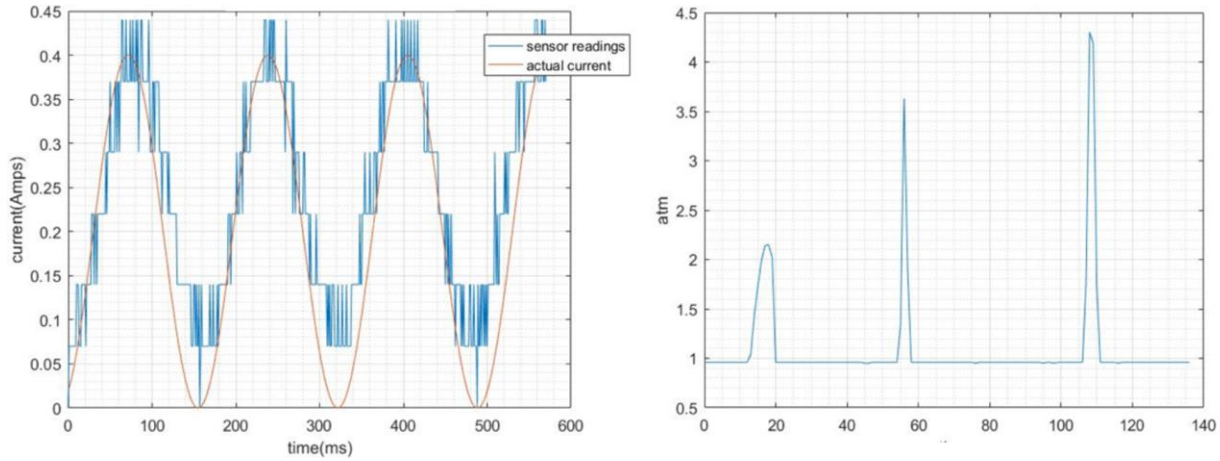


Figure A5. Sensor validation for current and pressure measurement.

Additionally, estimation of data storage requirements for the AWS database depending on the implementation of sensor retrofits at manufacturing facilities. Figure A6 shows the data requirements for bandwidth and data storage for a range of sensors both on a daily (left) and annual (right) data requirement perspective. From the figure, the data required is scales linearly with the number of sensors and it is clear that data requirements in the TB range may be required depending on the number of sensors implemented at a facility. This data will be useful for requirements planning for implementers of RRK sensor packages.

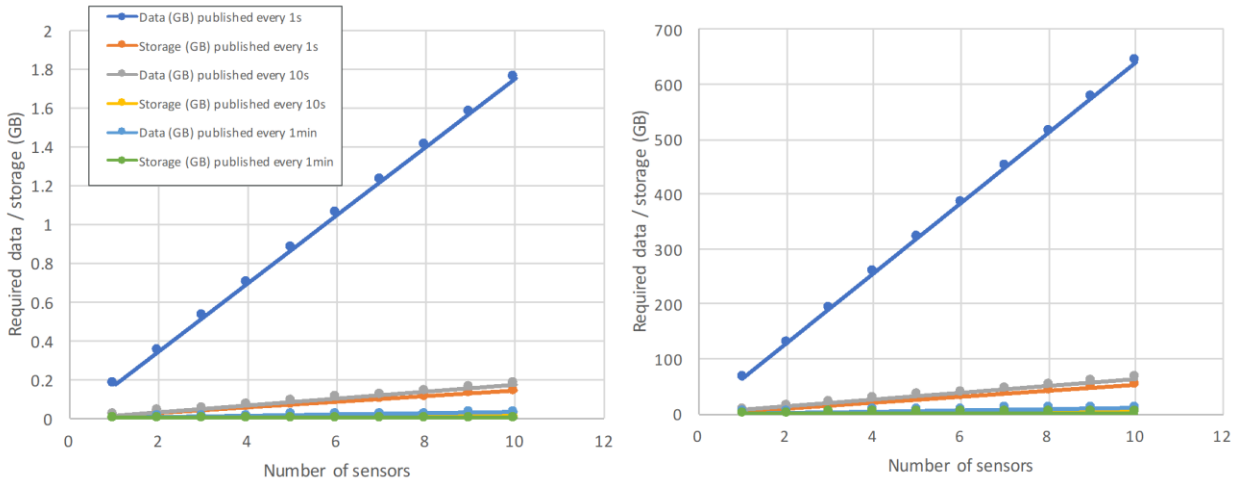


Figure A6. Data requirements for reconfigurable retrofit kit sensor packages.

In terms of ROI, the cost basis for implementation is approximately $300/sensor package. In this regard, the base implementation cost is far below the $1000/machine implementation cost. Implementation of the iBlue and Mazak Smartbox would have to consider the actual retail prices of these hardware. For the Smartbox, an implementation across multiple machines would enable a $1000/machine implementation cost.

Transition Plan and Improvements Report

Georgia Tech Precision Machining Research Consortium

I. Transition Plan

The flexible integrated architecture as well as the decoupled integrated architecture follows a strategy to provide flexibility based on industrially hardened communications network hubs and wireless sensor packages. Each of the individual systems components are mature from an operational perspective (TRL6-7) and their integration as a system for the retrofit concept is also demonstrated in a relevant production environment (TRL6). The key technical advances that were made during this project include demonstration of prototype systems-level integration and true flexibility of data sensing for customer-specific applications using low-cost, industry-standard high security integration platforms. The open nature of the data transfer, particularly for the decoupled integrated architecture, enables remote data analysis as well as plug-and-play incorporation of new models and sensors into the overall diagnostic system architecture. Ability to analyze data remotely enables business models wherein third party suppliers might provide diagnostic/prognostic services to companies from larger enterprises to small/medium enterprises.

The project has resulted in development of a RRK sensor package that can be integrated into commercial industrial solutions. Industry partners involved in development of technology solution (ITAMCO) as well as implementation of the technology at their facilities (Caterpillar) have provided useful perspectives on the capabilities of the system. The project team has future plans to further develop additional edge computing applications using the base retrofit kits. To complement the data connectivity, the project team will develop cloud-based analytics-based methods for processing of retrofit kit sensor data. The project team will consult with existing firms and new startup firms interested in developing commercial offering for retrofit kits and web applications. Transition opportunities with companies interested in developing and commercializing sensor packs are welcome. Elements of our team have had discussions with others in this regard.

According to the IDC Worldwide Semiannual Internet of Things Spending Guide (2017), IOT spending in the manufacturing area is predicted to be $183B by 2021. The core technology of the RRK sensor package is approximately $300/package in terms of cost of materials. Further, the components that the machine monitoring and RRK sensor packages are built upon are commercially available items that are in high supply. Implementers can easily purchase base components in small lot sizes and at low cost. It is anticipated that the market for these technologies is significant considering that a sensor retrofit solution at this price level does not currently exist in the market. The unique element is how to integrate these to achieve ubiquitous sensing for manufacturing. Companies that exist in this market include technology developers with expensive and cost-intensive data collection devices. Low cost computing for manufacturing is an open market. In applying the technical advancements made in this project, future development and/or implementation of these technologies can be made by new startup firms or existing firms interested in offering/adoption of these technologies.

II. Improvements

The low-cost IOT devices central to the RRK sensor packages may require integration with existing enterprise networking infrastructure. This compatibility requirement is critical in terms of

authentication of these devices on existing wired/wireless networks. In this regard, specific IOT microcontrollers and/or single board computers may have varied capability for supporting more advanced wireless authentication protocols that may require use of specialized authentication certificates. Further, adopter facilities may need to review compatibility of these devices with existing site-specific network security policies. In terms of the implementation choice for local and/or cloud based data storage, implementers may also have policies in place for determining feasibility of off-site data storage and the two architectures proposed here provide for flexibility in this regard. Additionally, integration of the technology solution would require ability to follow assembly documentation for off-the-shelf solution and an ability to write code for customized sensor solutions. These codes are used quite heavily now in practice so adopters can leverage an engaged community of developers through online technical forums. Finally, as the technology space surrounding low-cost IOT devices is characterized by rapid pace of development, adopters must also pay attention to these developments as new capabilities of these devices may offer significant advantages over the current development.

The RRK sensor package developed in this project represents an implementable and scalable solution for machine retrofit at this instance in time. As the landscape of IOT technologies rapidly changes on a monthly to yearly basis primarily due to the rise of consumer IOT devices, the industrial IOT sector can and must take advantage of the associated benefits and be comfortable with constant iteration of the sensor platform due to rapid technology obsolescence. Primary improvements to the existing solution for the RRK sensor packages would address the limitations associated with the current implementation as in the previous paragraph. As the underlying microcontroller platforms mature and increase in compatibility with common enterprise-level wireless network protocols, adopters should seek to include these platforms, especially in headless-type integration schemes, which do not require the inclusion of a visualization device for navigating web-based enterprise authentication schemes. Additionally, the use of mesh-enabled WiFi microcontroller devices is increasing and can be used to chain RRK sensor packages across long facility distances where deployment of wireless technologies may not be possible. A rapid development in the sensor market also will provide a natural pathway to improve the current sensor pack technology developed in this project. Finally, the use of rechargeable power technologies with low power consumption microcontrollers can be explored for long-duration truly wireless implementations of the RRK sensor package.