1-11-2019

Design and Evaluation of IoT-Enabled Instrumentation for a Soil-Bentonite Slurry Trench Cutoff Wall

Jeffrey C. Evans
Bucknell University, evans@bucknell.edu

Alan Marchiori
Bucknell University, amm042@bucknell.edu

Yadong Li
Bucknell University, yl5273@nyu.edu

infrastructures

MDPI

Article

Design and Evaluation of IoT-Enabled Instrumentation for a Soil-Bentonite Slurry Trench Cutoff Wall

Alan Marchiori [1,*], Yadong Li [2] and Jeffrey Evans [3]

[1] Computer Science, Bucknell University, Lewisburg, PA 17837, USA
[2] Computer Science, New York University, New York, NY 10012, USA; yl5273@nyu.edu
[3] Civil & Environmental Engineering, Bucknell University, Bucknell University, Lewisburg, PA 17837, USA; jeffrey.evans@bucknell.edu
* Correspondence: alan.marchiori@bucknell.edu

Received: 14 December 2018; Accepted: 7 January 2019 ; Published: 11 January 2019

Abstract: In this work, we describe our approach and experiences bringing an instrumented soil-bentonite slurry trench cutoff wall into a modern IoT data collection and visualization pipeline. Soil-bentonite slurry trench cutoff walls have long been used to control ground water flow and contaminant transport. A Raspberry Pi computer on site periodically downloads the sensor data over a serial interface from an industrial datalogger and transmits the data wirelessly to a gateway computer located 1.3 km away using a reliable transmission protocol. The resulting time-series data is stored in a MongoDB database and data is visualized in real-time by a custom web application. The system has been in operation for over two years achieving 99.42% reliability and no data loss from the collection, transport, or storage of data. This project demonstrates the successful bridging of legacy scientific instrumentation with modern IoT technologies and approaches to gain timely web-based data visualization facilitating rapid data analysis without negatively impacting data integrity or reliability. The instrumentation system has proven extremely useful in understanding the changes in the stress state over time and could be deployed elsewhere as a means of on-demand slurry trench cutoff wall structural health monitoring for real-time stress detection linked to hydraulic conductivity or adapted for other infrastructure monitoring applications.

Keywords: wireless sensing; reliability; instrumentation; deployment; slurry wall; bentonite

1. Introduction

In the US and worldwide, there are tens of thousands of sites were ground water is contaminated by past industrial and waste management practices. It is simply impossible, technically and economically, to remove the contaminants from the subsurface at all of these sites. Rather the approach is to remediate the site to protect the public health and the environment. Often risk assessments can be used to select the appropriate remedy [1]. One component of remedial systems is the installation of a vertical barrier to horizontal ground water flow and contaminant transport. Vertical barriers are normally part of a system that may also include a horizontal cover over the site and, at times, pumping and treatment of contaminated ground water. Vertical barriers are most often deployed circumferentially and keyed into a low permeability stratum beneath the site creating a "bathtub" effect to contain the contaminants. Low hydraulic conductivity of the vertical barrier is essential to the successful performance of the system.

In the US, the most commonly employed vertical barrier material is soil-bentonite (SB). The vertical trench is excavated by replacing the excavated soil with a slurry comprised of approximately 5% bentonite and water. The sole purpose of the slurry is to maintain trench stability. The SB backfill is then made from a mixture of excavated soil, bentonite-water slurry, and, as needed, additional

dry bentonite. The ingredients are blended to form a flowable backfill that is pushed into the trench displacing the slurry. The process is shown schematically in Figure 1 and photos from the research site described in this paper are shown in Figure 2.

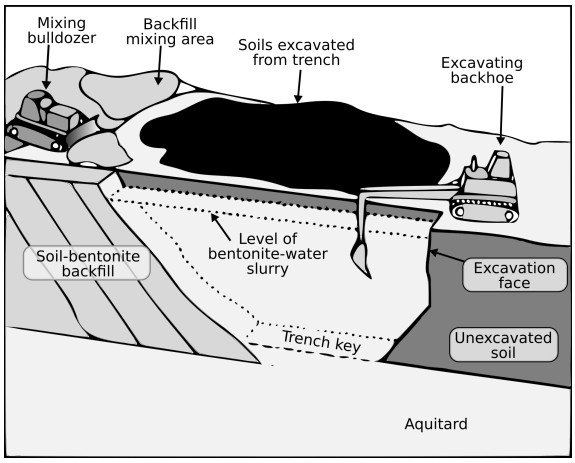


Figure 1. Illustration of the excavation and backfill process to construct a soil-bentonite slurry trench cutoff wall [2].

Figure 2. Excavation and Backfill at the test site.

Despite the fact that SB cutoff walls are commonly used, field measurement of performance of these walls is lacking [3]. Since hydraulic conductivity has been shown to be stress dependent [4,5], it is necessary to understand the state of stress in the SB cutoff wall in order to accurately determine the hydraulic conductivity. As evidenced in Figures 1 and 2, a soft compressible material is placed in a narrow trench. During consolidation of this soft material, there is stress transfer from the SB backfill to the sidewalls of the trench (the less compressible formation material). As a result the stresses within the backfill are less than geostatic, that is, less than the self-weight of the material. Models developed to predict the stresses in SB cutoff walls, include "arching" model [2] and a "lateral squeezing" model [6]. The arching model assumes that the trench walls are rigid. The lateral squeezing model assumes the sidewalls move inward and there is stress and strain compatibility between the backfill and the sidewalls. The modified lateral squeezing (MLS) model which takes into account the stress-dependent nature of SB backfill compressibility [7]. None of these existing models relate the stresses in all three directions and they are all models rather than direct in situ measures of stress, therefore, knowledge of in situ stress is required to better predict the in situ hydraulic conductivity.

In order to investigate the stress state of SB walls, a full-scale, instrumented, and internet-connected SB slurry trench cutoff wall was constructed with the support of the National Science Foundation [8]. The SB cutoff wall was outfitted with instrumentation to directly measure stresses in all three principal directions as well as vertical and lateral deformations. Data was collected

and transmitted from the site and stored in a database. A website dashboard was developed to display the sensor information.

The main contribution of this work is not any one specific component but the demonstration and evaluation of technologies used in combination to bridge the technological divide between traditional scientific instrumentation and the IoT. The current state of IoT technology is sufficiently mature that we were able to construct a highly reliable solution by following established best practices and available open-source software packages. This paper presents a model that can be followed by other researchers to reap the benefits of IoT without sacrificing scientific accuracy or data integrity. The instrumentation system coupled with modern IoT data collection and visualization has proven extremely useful in understanding the changes in the stress state over time and could be deployed elsewhere as a means of on-demand slurry trench cutoff wall structural health monitoring for real-time stress detection linked to hydraulic conductivity.

The structure of this paper begins with a discussion of the SB wall instrumentation in Section 2. Following this, the primary issues of Data Collection, Transport, Storage, Visualization are detailed in Sections 3–6 respectively. Results from the past two years of operation are discussed in Section 7. Finally, conclusions and future work are presented in Section 8.

2. System Design

To investigate the stress state of a full-scale SB slurry trench cutoff wall, a full-scale SB slurry trench cutoff wall was constructed with the addition of the following instrumentation as shown in Figure 3 at a test site: (1) inclinometers at eight locations immediately outside the trench to measure lateral deformations as a function of depth; (2) earth pressure cell cages to measure the three-dimensional state of stress within the backfill at three different depths; (3) paired sensors to measure moisture content and suction within the backfill; (4) settlement plates to measure vertical deformation; (5) piezometers within the wall to measure pore water pressures and in-situ hydraulic conductivity of the backfill (by slug testing); (6) monitoring wells outside the wall (adjacent to the piezometers); and (7) stainless steel electrodes for ER imaging. Power for all instrumentation at the site was provided by a 40-watt solar panel and a 24-Ah battery.

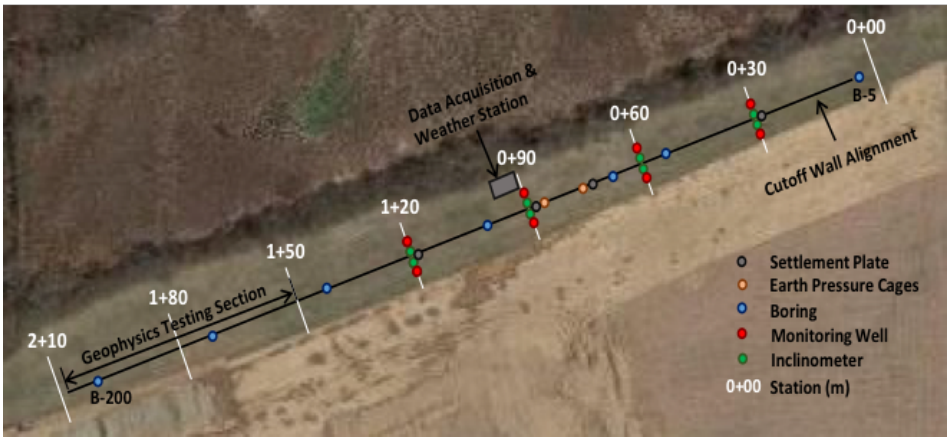


Figure 3. Location of the instrumentation at the test site.

The earth pressure cell cage, illustrated in Figure 4, includes three vibrating wire stress sensors mounted in three cardinal directions to measure vertical and horizontal (longitudinal and transverse) stresses. Each cage also has one vibrating wire piezometer (to measure pore pressure) and a biaxial tiltmeter and magnetic compass (to measure the as-placed orientation). All sensors are connected to a Kevlar reinforced cable that extends vertically out of the wall for data acquisition. The cages were originally designed for use in mine paste backfill and, therefore, have an open structure that is ideal for allowing the fluid SB backfill to fill the cage and cover the sensors. The cages were placed at different

depths within the slurry-filled trench (prior to backfill placement) and were fixed in place using a rope to the surface on a guide structure consisting of four vertical legs and cross braces. The vertical legs of the guide structure were embedded into the soil below the bottom of the trench serving to anchor the bottom of the guide structure. The top of the guide structure was affixed to ground anchors at the surface. Once backfill had displaced the slurry the gauges were released from their vertical restraints and allowed to settle freely with the backfill. The stress measurements are complemented with measurements of vertical and horizontal deformations obtained from settlement plates and inclinometers, respectively.

There were a number of difficulties to be overcome for the successful installation of the instrumentation. One constraint was the need to hold instrumentation in a specific vertical and longitudinal location while backfill (a thick viscous liquid) flowed around the cages during backfilling. While the guide structure of four pipe legs and cross-bracing was largely successful, there was elastic bending of the legs. Since this system can serve as a model for health monitoring of SB cutoff walls, future installation should consider additional cross-bracing and/or thick wall pipe for the legs. Despite the elastic deformation of the legs, the instrumentation was held in place during backfilling and, once constraints were released, moved downward with the backfill to produce the necessary stress-strain compatibility between the instrumentation cluster and the backfill. The instrumentation was hard-wired to the data acquisition system located above ground in secure indoor space (a modified steel shipping container). The wiring was all buried and encased in a protective PVC pipe and no damage occurred to the wiring system.

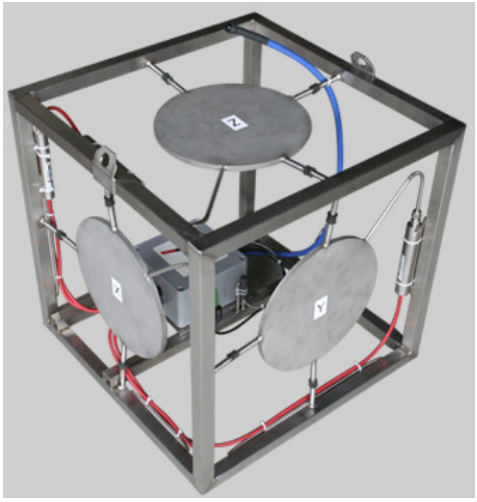


Figure 4. Earth pressure cell cage for measuring stresses in the SB slurry trench cutoff wall.

In order to remotely monitor the data measured by the instrumentation and to facilitate efficient analysis of the data, a web-based interface was desired. To collect data, a datalogger was programmed to read measurements from the instrumentation on site and coupled with a transmitter to send them continuously via radio signals to a receiver on campus as shown in Figure 5. The datalogger, along with the other instrumentation, was deployed at the site in July 2016. Correspondingly, a receiver was deployed to listen to the datalogger through an antenna on the roof of a nearby campus building. This receiver also handles ingesting the data into a time-series database. The time-series database was implemented using an existing MongoDB (https://www.mongodb.com/mongodb-3.6). The web application was implemented using the Python Flask framework (http://flask.pocoo.org/). The database and web application were hosted on a College of Engineering server. The web interface shows a list of instrumentation and displays the data in both tabular formats and in charts generated by JavaScript libraries provided by Highcharts (https://www.highcharts.com/). The web interface also allows a researcher to download data as `.csv` files for more detailed analysis.

Figure 5. Google Earth rendering of the 1.3 km wireless link from campus to the test site (Google Earth Pro v7.3.1.4507).

Wall Construction

The coupled nature of deformation and stress behavior can be understood in the context of SB cutoff wall construction procedures. For this project, and consistent with standard industry procedures, a slurry filled trench was excavated using a Cat 330C excavator with a 0.9 m wide bucket. The slurry was made from a mixture of 5–6% sodium bentonite clay suspended in water and employed to maintain trench stability (i.e., liquid shoring). Once excavation to the design grade 7 m below ground surface was completed, backfill was added as excavation continued. The backfill was made from excavated on-site soils blended with the bentonite-water slurry to form a material with a viscous liquid consistency. The measured values of slump varied between 75 mm and 150 mm. A schematic of the process is shown in Figure 1. Since backfill is placed at a viscous liquid consistency, substantial self-weight consolidation occurs. The otherwise geostatic nature of self-weight consolidation is impacted by the relatively incompressible trench sidewalls and results in load transfer from the backfill to the sidewalls. Coupled to these two phenomena is the inward movement of the trench sidewalls to achieve the necessary stress-strain compatibility between the backfill and the adjacent formation. Finally, given void ratio changes that result from consolidation stresses, the hydraulic conductivity changes in response to the stress.

3. Data Collection

The primary goal for data collection was to ensure that at least one sample per hour from all of the sensors was reliably recorded. This sample rate was selected because the stress state in the wall is expected to change slowly over a period of days or weeks. Secondary goals were to present the data to the researchers (1) in a timely fashion on a website and (2) in a format facilitating analysis of the data. In this context, timeliness is loosely defined and up to one hour of latency is acceptable. The location of the site, less than a mile from campus as shown in Figure 5, makes the use of a low-power wireless link possible. The data collection architecture is shown in Figure 6.

Initially, a Campbell Scientific RF401A 900 MHz Spread-Spectrum Radio (https://www.campbellsci.com/rf401a) was deployed directly to the CR6 datalogger with a 3 dB Yagi antenna as shown in Figure 7. The gateway server had a matching RF401A radio. We used the PyCampbellCR1000 (https://github.com/LionelDarras/PyCampbellCR1000) software by Salem Harrache, Lionel Darras, and others to download data from the datalogger using the PakBus communications protocol in our lab experiments. However, once the datalogger was installed on site, we were initially unable to successfully download data.

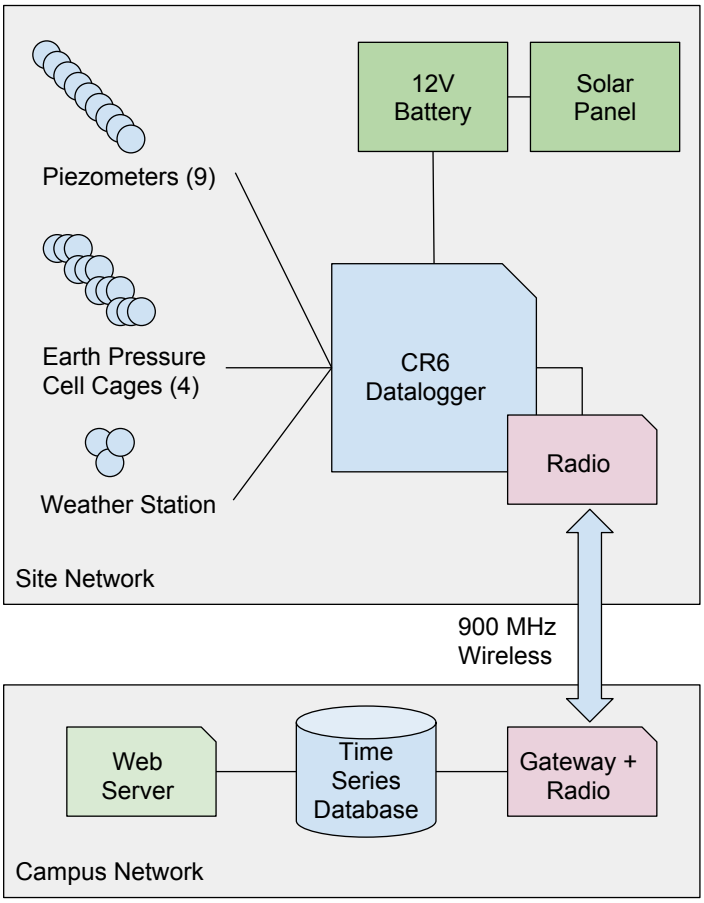


Figure 6. Diagram of the data collection, transport, storage, and visualization plan for the project.

Figure 7. Weather station with 3 dB Yagi antenna for data transmission from the site.

We began troubleshooting the problem by investigating the PakBus messages on the gateway. There was traffic in both directions but not consistently. It appeared that messages were getting dropped causing almost constant retransmissions of PakBus messages. Curious, we opened the RF401A radio to find a Digi XBee 900HP (https://www.digi.com/products/xbee-rf-solutions/sub-1-ghz-modules/xbee-pro-900hp) radio module attached to a controller board. This is a frequency hopping radio with automatic packetization supporting a maximum transmission unit (MTU) of 256 bytes. The PakBus MTU is fixed at 1500 bytes and we suspect that any one dropped PakBus fragment causes the entire message to be resent or lost altogether resulting in a unreliable communication.

To solve the problem of collecting data from the datalogger, a Raspberry Pi Model B computer was installed at the site to download data locally over the wired serial interface using PyCampbellCR1000. Power for the Raspberry Pi was provided by a 12-to-5 Volt DC/DC converter connected to the datalogger's 12 Volt battery. This increases site-energy consumption by almost 1 Watt when no other peripherals are attached to the Raspberry Pi (i.e., keyboard, display). However, this arrangement avoids sending the 1500-byte PakBus messages over the wireless link. A data collection script is run as a system cron job every 5 min. The script communicates to the logger and incrementally downloads data from all tables since the last download. The data is stored as a JSON file in a temporary directory on the Raspberry Pi's filesystem. The file is later detected and transmitted to the gateway server for processing.

This approach improved reliability by decoupling data collection and data transmission. If the wireless link were to fail, data will still be collected and stored to the Raspberry Pi's SD card as JSON files. Once the wireless link is repaired, all previously collected data would be automatically detected and uploaded to the gateway computer. As a final reliability measure, data is also retained by the datalogger itself in a circular buffer that can store about 1 week of data in our particular configuration. So even if the Raspberry Pi were down for one week, we would not lose any site data.

4. Data Transport

The data collection job on the Raspberry Pi produces JSON files that need to be transmitted to the gateway and inserted into the time series database for permanent storage. Wireless sensor networks have developed mature data collection and transport protocols [9–13]. Implementations of these network protocols are available for typical WSN platforms such as TinyOS [14] and Contiki [15]. 6LoWPAN [16] could also be leveraged for its data transport capability. However, we discount these solutions because they are typically deployed with IEEE 802.15.4 [17] radios, which would require multiple hops to reach the gateway from the deployment site. Since the site and gateway is separated by a river and private property, it would be very difficult to deploy multiple nodes along the path to create a connected network.

On the other hand we also explored commercial point-to-point WiFi solutions (e.g., Ubiquiti LiteBeam (https://www.ubnt.com/), MikroTik (https://mikrotik.com/), etc.). These solutions deliver hundreds of megabits per second data rates over ranges up to 15 km. This is orders of magnitude more than is required for our application, which comes at a price. In this case, the cost is energy consumption up to 10-Watts. Additionally, these solutions required 120 Volt AC power or 24/48 Volts for PoE. This would necessitate another DC-DC converter or power inverter at the site in addition to extra batteries and solar panels.

To avoid the significant energy cost, we elected to re-purpose the existing equipment we had. Namely, the Digi XBee 900HP radio modules and 900-MHz Yagi antennas. To transport the data we built a lightweight point-to-point reliable data transport protocol that we call the XBee Transfer Protocol.

The XBee Transfer Protocol (XTP) was inspired by ZMODEM [18] and similar classic network protocols. It is a simple protocol for reliably transferring files between two XBee radios that is written in Python using the Python-XBee package (https://github.com/niolabs/python-xbee) and is made available as open source software (https://github.com/amm042/XbeeTransferProtocol). The entire

protocol implementation is approximately 400 lines of Python split evenly between the sending and receiving code.

The receiver periodically broadcasts a beacon for discovery. Once the receiver has been discovered by the sender, the protocol breaks up a file into chunks and data fragments for transmission as shown in Figure 8. The file is first broken into fixed-sized chunks up to 64 KB. Each chunk is then split into fragments smaller than the MTU of the radio for transmission. The fragment size can vary depending on the XBee variant, but is typically 127 bytes. This two-level structure enables transferring large files while still allowing efficient 3-byte data fragment headers. Correctness is ensured at all levels. Fragments are checked by the radio's underlying per-packet CRC. Each reconstructed chunk must also pass a 32-bit CRC. Finally, the resulting file must have the same MD5 hash for a transfer to be successful.

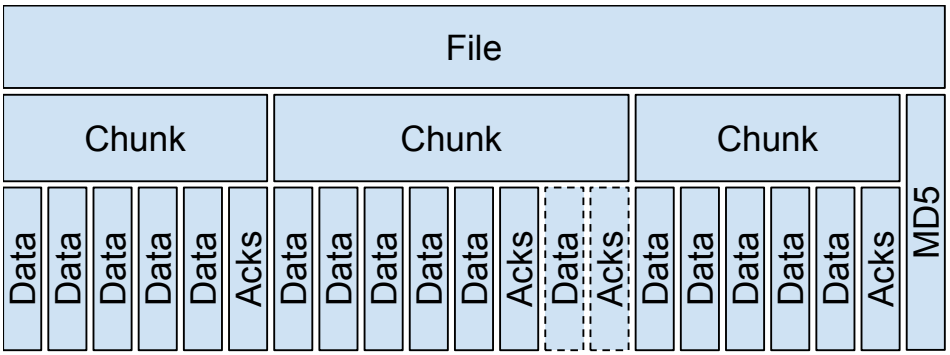


Figure 8. The XBee Transfer Protocol breaks a file into chunks with batched acknowledgments per chunk. Lost data packets are retransmitted (dotted lines) until all data fragments in each chunk are received. A final MD5 hash check provides a second-level verification of the transfer.

Transmitting a chunk is initiated by the sender transmitting a send request containing an 8-bit packet identifier followed by four 32-bit unsigned values defining (1) byte offset of the chunk in the file, (2) the total file size in bytes, (3) the number of fragments in the chunk, and (4) the 32-bit CRC of the chunk. Following this header is the filename encoded as a UTF-8 string. Since the offset and file size are 32-bits, the maximum supported file size is 4 GiB. The send request packet length is 17 bytes plus the length of the encoded filename.

The receiver acknowledges the send request and creates a zeroed bit array containing one bit for each fragment in the chunk. The size of the bit array depends on the number of fragments in the chunk, however, this bit array must be smaller than the MTU of the radio minus all headers, so that it can be transmitted in the payload of a single packet. For common low-power radios with a maximum payload size of 100 bytes, the resulting effective maximum number of fragments is 800 yielding a maximum chunk size of 80 KB.

After the send request is acknowledged, the sender transmits all data fragments in the chunk without any individual acknowledgments or additional error checking. We assume the radio hardware supports minimal error detection sufficient to reject most corrupt packets. As a result, the fragment header is only three-bytes long which maximizes efficiency. The header is comprised of a one-byte packet type plus a two-byte fragment identifier. When a fragment is received from the radio, the data is stored and the corresponding bit in the bit array is set by the receiver. After all data fragments are sent, the sender then transmits an acknowledgment request.

Upon receiving the acknowledgment request, the receiver transmits the bit array containing the received fragments. In response, the sender retransmits any data fragments where the corresponding bit in the array is not set followed by another acknowledgment request. The process is repeated until all fragments in the current chunk have been acknowledged.

The sender then continues the transfer by sending the remaining chunks in the file using the same process with a new send request message specifying the same filename and the appropriate beginning offset to each chunk.

As a final check, after all chunks have been successfully received, the sender transmits a MD5 check message containing the filename and MD5 hash. The receiver computes the MD5 hash on the local file and compares it to the sender's transmitted hash. If they match, the file was successfully transferred. If not, the file transfer failed and the file is deleted. The retriever's MD5 hash is also sent back to the sender. If they do not match, the transfer is started over from the beginning by the sender. This additional check is necessary because most low-power radios (e.g., IEEE 802.15.4) use 16-bit checksums which has the possibility of false positive results.

All messages other than the data fragments are considered control messages. Data fragments are sent without MAC-level acknowledgements and rely on XTP's batched acknowledgement mechanism. The control messages are sent using the radio's MAC-level acknowledgement mechanism. On the 900HP radio module, the default value is 10 retransmissions. On the series 1 radio, the default value is 3 retransmissions. In addition to these retransmissions, XTP will retry each control message 15 times. If XTPs control message retransmissions are exhausted, XTP will abort the transfer and report failure.

The protocol is intended to be very lightweight and does not have any inherent security features. Malicious messages could be constructed to corrupt or otherwise manipulate file transfers. However, security can be implemented with additional radio features. In our deployment, the radios are configured with a unique channel hopping sequence and encryption using the radio's built in 128-bit AES encryption for all packets making unauthorized access or manipulation of site data extremely difficult.

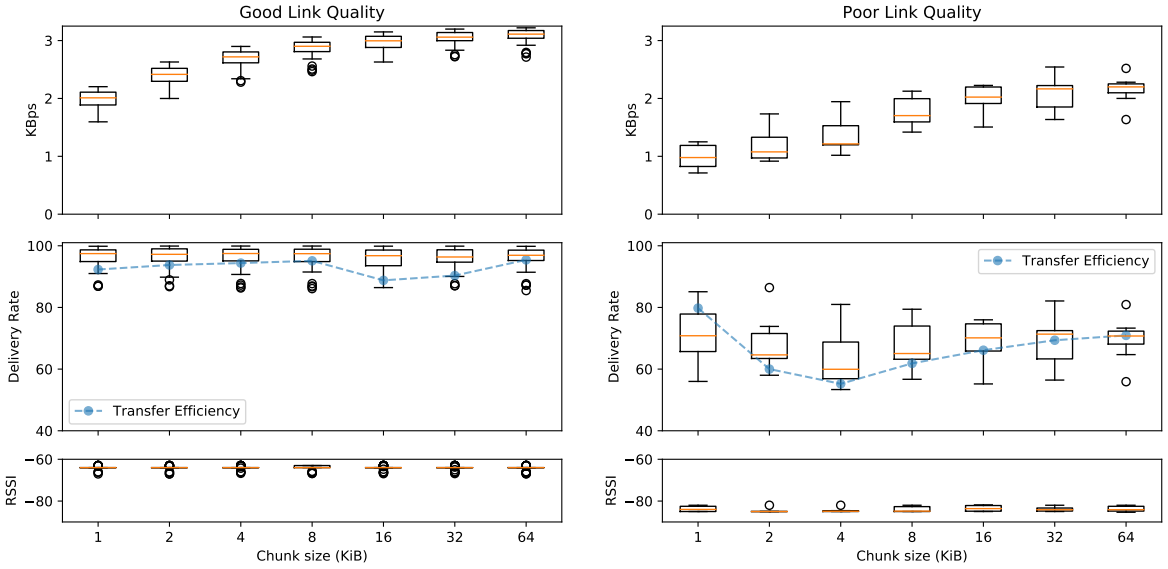
4.1. XTP Evaluation

The XTP is designed to be simple, reliable, and efficient. Each data fragment is sent with minimal overhead (3 bytes), relying on the radio's internal data integrity checks for each data fragment while still verifying the final result with a MD5 hash. Batching acknowledgments for data fragments significantly improves performance. All control messages (i.e., all messages other the data fragments) are transmitted with per-packet acknowledgments and the default number of retransmissions for the given radio hardware. If a control message is lost, the XTP transfer fails and the application must restart the transfer. The XBee serial API is relatively consistent between different models of Digi XBee radios, the only difference being the addition or removal of radio-specific commands. For example, on the 900HP channel hopping is managed differently than the 802.15.4 radios because they operate on different bands with different regulatory requirements. As a result, we were able to test on XBee 802.15.4 series 1 modules (https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-802-15-4) and deploy to the site with the 900HP modules without any changes in the XTP protocol.

We evaluated the performance of the protocol by repeatedly transferring a 1 MiB file between two XBee 802.15.4 series 1 radios with two different link conditions. The first, with good link quality, had a median RSSI of −64 dBm. The second, with poor link quality, had median RSSI of −84 dBm, only 8 dBm greater than the −92 dBm sensitivity of the radio.

For each test, we used various chunk sizes from 1 KB to 64 KB and repeated each test at least seven times. The radios were configured without encryption and communicated to the host at 38,400 baud, limiting the throughput to a theoretical maximum of 4.8 KBps. This baud rate was selected for the 900HP module, which has a configurable RF datarate of 10–20 kbps (depending on the hardware version) in high-sensitivity mode. The XBee series 1 radios use a fixed RF data rate of 250 kbps. As a result, it is trivial to achieve higher throughput by increasing the serial baud rate and enabling per-packet acknowledgements and retransmissions using the series 1 radio. For example, the radio could perform multiple retransmissions at the RF data rate before the next packet is received over the serial interface.

Figure 9 shows the achieved data rate, fragment delivery rate, and RSSI values under both link conditions. The fragment delivery rate is the percentage of transmitted data fragments that were received aggregated over each chunk. The blue dashed line on the delivery rate plot is the computed protocol efficiency. This is computed from the total number of bytes transmitted during the file transfer divided by the size of the file. This value includes all control packets and retransmissions. Under the poor link conditions, several transfers failed and were restarted. Failed transfers are excluded from these results.



(**a**) Good link quality (RSSI $>$ -70 dBm). (**b**) Poor link quality (RSSI $\leq$ -70 dBm).

Figure 9. Achieved data throughput transferring 1 MiB of random data between two XBee 802.15.4 series 1 radios with XTP and various chunk sizes over both good and poor quality links.

These results show that increasing the chunk size from 1 KB to 64 KB increased throughput by over 50% regardless of the link quality, however, the link quality did impact the achieved throughput as expected. These results also demonstrate that batched acknowledgements provide high throughput on good links and balanced throughput and reliability on poor links.

We attempted to benchmark even worse links but we were not able to create a link of consistent quality below this level. At the limits of the radio's sensitivity, transfers will fail and have to be restarted. This results in extremely low throughput and high variance. Although reliability is a goal of XTP, it is not suited to operate over extremely poor links with very high packet loss (e.g., $>$50%). It is designed to operate efficiently over wireless links where bi-directional communication is generally possible using standard per-packet acknowledgements. In higher loss situations, we suggest improving link quality through other means to achieve reliable communication and then applying XTP for data transfer. This could include using higher powered radios, directional antennas, reducing the data rate to improve receiver sensitivity, or using more advanced signal encoding with forward error correction.

4.2. XTP Extensions

In the SB slurry wall deployment, we used XTP for point-to-point data transfer. The protocol is not limited to point-to-point data collection and could be used to transfer data from multiple data sources at a single receiver in a start topology. This would rely on the radio's underlying MAC protocol for collision avoidance. Typical MAC protocols are either scheduled or unscheduled. Unscheduled protocols use carrier sensing to detect other transmissions and use various algorithms to minimize collisions. Schedule protocols, such as TSMP [19], generate collision-free schedules for all nodes.

XTP could be layered above both classes of MAC protocols to provide additional reliability when transferring large files. However, for best results, the MAC protocol should support the selection of both reliable and unreliable quality-of-service. XTP uses the MAC protocol's reliability mechanism for control messages and it's own reliability mechanism for data messages.

For deployments over a larger distance, it might be necessary to construct a multi-hop network. A large number of ad-hoc network routing protocols exists that could be layered below XTP for network routing, such as RPL [20]. For a small number of hops, utilizing relay nodes could be another solution. In the simplest form, this would be a full node that first receives a file, buffers it, and then transmits the received file to the next host. This could be achieved without modification to XTP, however, this would increase the latency of the transfer. If this additional latency was unacceptable in the target application, the node could relay on a per-chunk or even per-fragment basis with minimal effort by developing an XTP-aware relay application.

5. Data Storage

The datalogger produces data organized in a tabular format with one row generated per sampling interval. The data collection process transforms this format into JSON objects which are compressed and transported to the gateway computer via the data transmission protocol. At the gateway, data is extracted from the the compressed JSON files and inserted into the time series database. The JSON files are retained as a backup. Although many specialized time series databases exist [21], such as InfluxDB and OpenTSDB, we chose to store the data in a standard MongoDB database. The two primary considerations were (1) this project wasn't expected to produce sufficiently large amounts of data to justify the use of a specialized database and (2) we are more familiar with the MongoDB API.

Storing time series data in a document-based database takes a bit of planning to achieve good performance. We followed the suggestions from a MongoDB webinar on this topic [22] and use three levels of aggregation: hourly, weekly, and monthly.

To insert a sample in the database, we first find or create a document for the given sensor, year, month, day, and hour. This document has two arrays containing all of the raw samples and timestamps received from the sensor in the particular hour. The hourly documents also contain the average sensor reading over the one-hour interval. This average is updated as data is inserted.

Once the hourly document is updated, the corresponding weekly document as specified by sensor, year, month, and week number, is created or updated. The weekly document contains The sensor, year, month, week, and average reading over that week. This update is done with a single MongoDB aggregation operation that averages the matching daily average readings in the week. Although this operation accesses $24 \times 7 = 168$ documents, it has not caused a performance problem.

Finally, the monthly document is created or updated. This operation takes the sensor, year, and month field and stores the aggregate of all of the weekly sensor readings in the given month. This is again performed using a single MongoDB aggregation operation to compute the average from the corresponding weekly documents. This operation accesses fewer than 5 documents in the worst case.

Data Access API

To facilitate the easy use of data from the document-oriented data in MongoDB, a REST-based web service was created using Python Flask (http://flask.pocoo.org/) to transform the data from the document-oriented structure back into a time series. This web service has a simple API with two primary methods. Both methods require the sensor name, start date/time, and stop date/time of interest. The first method returns all of the raw samples in the interval for the sensor. This method is used for raw data downloads only. The second method determines the most appropriate resolution of data to return based on the length of the interval and is used by the web interface.

To determine the appropriate resolution to return, we first compute the number of hours in the interval. If it is less than a threshold (currently 500), the hourly aggregate data is returned. If there are more samples in the interval, the number of weeks is computed. If this is less than the threshold,

weekly aggregates are returned. If this still results in too many samples, the monthly aggregates are returned. This logic helps ensure the user is never overwhelmed by too much data while still allowing close inspection of short intervals.

6. Data Visualization

A dynamic website was created using HTML5, CSS3, and JavaScript. The Bootstrap dashboard template (https://getbootstrap.com/docs/4.1/examples/dashboard/) served as our starting point. The sensors are grouped by experimental function so that users can more easily find the most relevant data.

Navigating to a group of sensors gives the option to view data from each individual sensor in tabular, chart, or CSV format. The CSV format returns the raw data while the tabular and chart view uses the automatic data resolution provided by the data access API. Charts are generated with the Highcharts (https://www.highcharts.com/products/highcharts/) library. Figure 10 shows the output of one sensor over a one-month period. The user can select the desired range of data and navigate to other sensors in this group on this page. Clicking the toolbar on the left will change the sensor group.

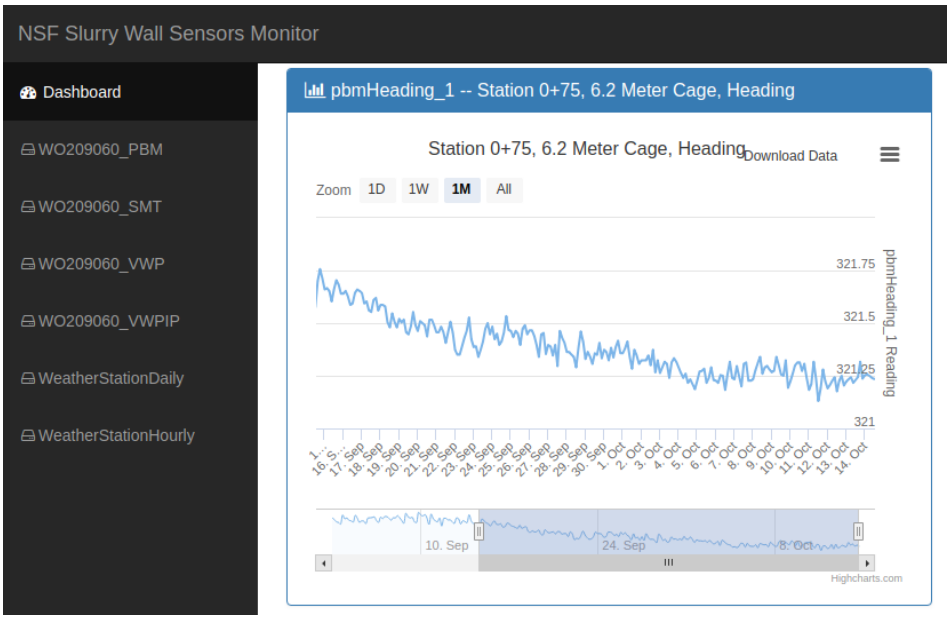


Figure 10. The web-based dashboard plotting data from one sensor in a group of sensors.

7. System Evaluation

After more than 2 years of operation, we have collected over 1 GB of raw sensor data using this system. The system was designed to ensure that at least one sample per hour is recorded from each sensor. To ensure this minimum level of data, the datalogger is programmed to sample the primary sensors 4 times per hour. The secondary, non-essential sensors such as weather and battery voltage, are sampled once per hour. Accordingly, the database should receive 11,453 raw samples per day. Using this number as a benchmark, the actual data yield over the deployment is shown in Figure 11. Data from July 2016 is available, however, has been omitted in our analysis because the datalogger program was modified several times in this period and as a result the number of samples received varied.

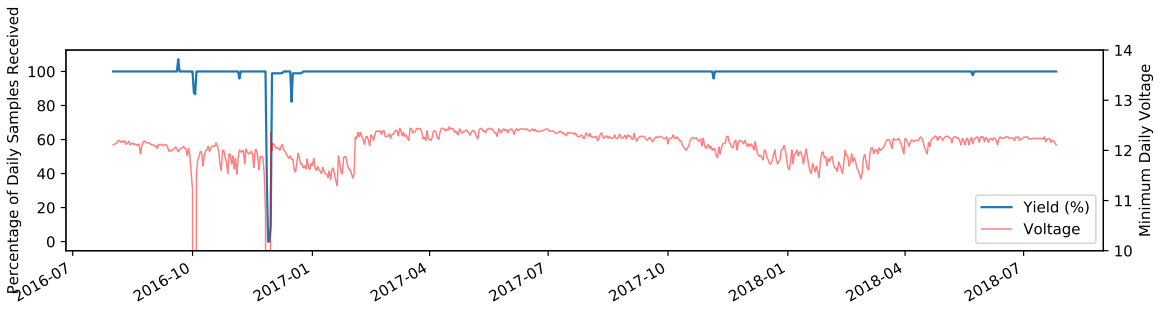


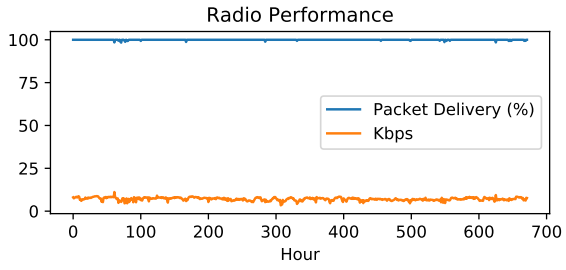
Figure 11. Raw data yield per day at the database and minimum battery voltage (to show battery failure) from August 2016 through July 2018.

Over the 722-day evaluation period, the database received 99.42% of the expected data. There was one day early in the test where extra samples were received. Early in the deployment data was also collected manually, which required disconnecting the Raspberry Pi from the datalogger. If the Raspberry Pi was downloading at the same time the download progress is not saved. After this event, we made sure to stop the Raspberry Pi before disconnecting it. There were two periods in late 2016 that the battery on the datalogger died causing loss of data for various periods over 6-days. As a result, the original 24 Ah battery was replaced with a 113 Ah battery eliminating this problem.

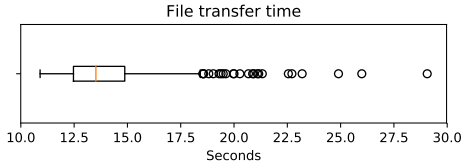
Excluding the dates with power failures, the database recorded 99.95% of the expected raw data. The exact cause of the remaining data loss was investigated and the lost data was consistently from the weather station not providing a set of hourly readings. One possible explanation is that the weather station's temperature/humidity sensor was the only sensor to use the SDI-12 serial protocol and this sensor was also duty-cycled sensors to save power. Since SDI-12 is an active protocol, it is possible this sensor failed to respond in time and recordings were skipped. We were unable to verify this explanation. However, because all other sensors reported correctly at these times we believe this data was never received by the datalogger. To avoid this we could replace the sensor or disable duty-cycling in the future.

To investigate the performance of XTP under the deployed conditions, we added performance monitoring in September 2018 to all XTP file transfers. Figure 12a shows the hourly radio performance in terms of packet (fragment) delivery rate and aggregate throughput. The average packet delivery was 99.97% and the median throughput was 7.13 Kbps. Figure 12c shows the hourly radio utilization is under 1% with median 0.38%. These results show that transporting data from the site is a relatively easy task. We did not evaluate the overall data latency as the XTP latency is significantly less than the latency caused by the data collection job. The distribution of file transfer duration is shown in Figure 12b. The data collection job runs every 5-min, which is much grater than the typical transfer time.

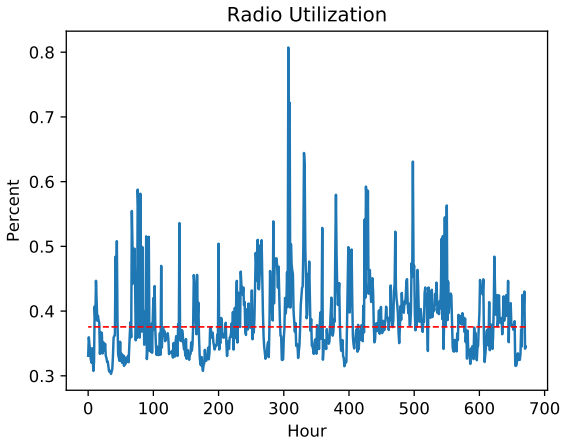
Overall, we conclude there was no data loss due to a failure of the data collection, transport, or storage systems developed for this project. We believe this is a result of our intentional focus on redundancy and separation of tasks within the system. If a data collection fails, it can be retired as the data is buffered on the datalogger. The data transmissions are checked at multiple levels and if a transmission fails, it starts over until it succeeds buffering incoming data in the meantime. Data is stored in the time series database and archive data is also stored both on the SD card at the site and on the gateway computer's hard drive for recovery. Fortunately, we have not had to use these archives but having them provides extra safeguards.



(**a**) Hourly packet delivery rate and effective transfer rate.



(**b**) Distribution of file transfer times from the site.



(**c**) Hourly radio utilization.

Figure 12. Radio performance measurements from the site from September 2018.

8. Conclusions and Future Work

In this project we successfully connected an industrial datalogger to a modern data storage and web-based visualization pipeline to achieve our goals of high reliability and the timely presentation of remote site data in a format facilitating analysis of the data. Throughout this process we relied heavily on open-source software and systems and released our own point-to-point reliable data transport protocol as open-source. This project can be used as a model giving credibility and validation to support other projects adapting other legacy scientific instrumentation onto the internet of things.

We conclude that this system can be used in the context of on-demand slurry trench cutoff wall structural health monitoring system for real-time stress detection linked to hydraulic conductivity. For on-demand monitoring of slurry trench cutoff wall projects in homogeneous soil conditions, the number of sensor cages could be reduced from four to one located near the bottom of the trench. This one instrumentation cluster coupled with one set of inclinometers and piezometers located at the same station as the instrumentation cluster would provide real-time monitoring of stresses which are directly linked to changes in hydraulic conductivity. This in situ measurement approach is a significant improvement over current quality control measures that are predominantly ex-situ.

In the future, we would like to generalize this approach to make it easier to apply to new projects. The currently deployed system has software that runs (1) at the site, (2) on the gateway, and (3) on the server. Each of these currently needs to be manually installed and configured for the whole system to operate. The software running on the Raspberry Pi at the site could be provided as a complete system image file that could be used to download from any compatible datalogger. The gateway software could be provided in a similar manner. However, because the web interface was developed specifically for this project, for each new project a new web interface would have to be manually created. However, the datalogger has sufficient metadata to automate this process. Our long-term vision is a plug-and-play solution to collecting, transporting, storing, and visualizing data from any supported datalogger.

Author Contributions: Conceptualization, J.E. and A.M.; methodology, J.E. and A.M.; software, Y.L. and A.M.; validation, J.E. and A.M.; formal analysis, A.M.; investigation, J.E. and A.M.; resources, J.E.; data curation, A.M.; writing—original draft preparation, J.E. and A.M.; writing—review and editing, J.E. and A.M.; visualization, A.M.; supervision, J.E. and A.M.; project administration, J.E.; funding acquisition, J.E.

Funding: This research was funding by the National Science Foundation under award number 1463198.

Acknowledgments: The authors would also like to acknowledge the contributions of CETCO which donated the bentonite for this project, Central Builders Supply which provided on-site support during construction and the study location, and Geo-Solutions, Inc. which performed the construction. The project would not have been possible without the support of these companies. In addition, the authors acknowledge the Jeffrey C Evans Geotechnical Engineering Laboratory endowment, established through the generosity of Michael and Laureen Costa, which was used to provide financial support for the students and faculty. Special thanks are also given to James Gutelius, Director of Bucknell University's Civil Engineering Laboratories, for his numerous contributions to this project.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results

References

1. Lagrega, M.D.; Evans, J.C. *Hazardous Waste Management*, 2nd ed.; McGraw-Hill: New York, NY, USA, 2001.
2. Evans, J.C.; Costa, M.J.; Cooley, B. *The State-of-Stress in Soil-Bentonite Slurry Trench Cutoff Walls*; Number 46 in Geotechnical Special Publication (GSP); ASCE: Reston, VA, USA, 1995; pp. 1173–1191.
3. Council, N.R. *Assessment of the Performance of Engineered Waste Containment Barriers*; The National Academies Press: Washington, DC, USA, 2007. [CrossRef]
4. Evans, J.C., Hydraulic Conductivity of Vertical Cutoff Walls (STP1142). In *Hydraulic Conductivity and Waste Contaminant Transport in Soils*; American Society for Testing and Materials: West Conshohocken, PA, USA, 1994; pp. 79–94.
5. Evans, J.C.; Huang, H. Hydraulic Conductivity of Soil-Bentonite Slurry Walls. In Proceedings of the Geo-Chicago 2016: Sustainable Geoenvironmental Systems, Chicago, IL, USA, 14–18 August 2016; pp. 548–557. [CrossRef]
6. Filz, G.M. Consolidation stresses in soil-bentonite back-filled trenches. In Proceedings of the 2nd International Congress on Enviornmental Geotechnics, Osaka, Japan, 5–8 November 1996; pp. 497–502.
7. Ruffing, D.G.; Evans, J.C.; Malusis, M.A. *Prediction of Earth Pressures in Soil-Bentonite Cutoff Walls*; Number GSP 199 in GeoFlorida 2010 Advances in Analysis, Modeling and Design; ASCE: Reston, VA, USA, 2010; pp. 2416–2425.
8. Malusis, M.; Evans, J.; Jacob, R.W.; Ruffing, D.; Barlow, L.; Marchiori, A.M. Construction and Monitoring of an Instrumented Soil-Bentonite Cutoff Wall: Field Research Case Study. In Proceedings of the 29th Central Pennsylvania Geotechnical Conference, Hershey, PA, USA, 25–27 January 2017.
9. Gnawali, O.; Fonseca, R.; Jamieson, K.; Moss, D.; Levis, P. Collection Tree Protocol. In Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09), Berkeley, CA, USA, 4–6 November 2009.
10. Wan, C.Y.; Campbell, A.T.; Krishnamurthy, L. PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks. In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '02), Atlanta, GA, USA, 28 September 2002; ACM: New York, NY, USA, 2002; pp. 1–11. [CrossRef]
11. Kim, S.; Fonseca, R.; Dutta, P.; Tavakoli, A.; Culler, D.; Levis, P.; Shenker, S.; Stoica, I. Flush: A Reliable Bulk Transport Protocol for Multihop Wireless Networks. In Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys '07), Sydney, Australia, 4–9 November 2007; ACM: New York, NY, USA, 2007; pp. 351–365. [CrossRef]
12. Shon, T.; Choi, H. Towards the Implementation of Reliable Data Transmission for 802.15.4-Based Wireless Sensor Networks. In Proceedings of the 5th International Conference on Ubiquitous Intelligence and Computing (UIC '08), Oslo, Norway, 23–25 June 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 363–372. [CrossRef]
13. Marchiori, A.; Han, Q. PIM-WSN: Efficient multicast for IPv6 wireless sensor networks. In Proceedings of the 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Lucca, Italy, 20–24 June 2011; pp. 1–6. [CrossRef]
14. Levis, P.; Madden, S.; Polastre, J.; Szewczyk, R.; Woo, A.; Gay, D.; Hill, J.; Welsh, M.; Brewer, E.; Culler, D. TinyOS: An operating system for sensor networks. In *Ambient Intelligence*; Springer: Berlin, Germany, 2004.

15. Dunkels, A.; Gronvall, B.; Voigt, T. Contiki—A Lightweight and Flexible Operating System for Tiny Networked Sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN '04), Tampa, FL, USA, 16–18 November 2004.
16. Montenegro, G.; Hui, J.; Culler, D.; Kushalnagar, N. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, 2007. Available online: https://www.rfc-editor.org/rfc/rfc4944.txt (accessed on 1 July 2018).
17. Gutierrez, J.A.; Callaway, E.H.; Barrett, R. *IEEE 802.15.4 Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensor Networks*; IEEE Standards Office: New York, NY, USA, 2003.
18. Forsberg, C. The ZMODEM Inter Application File Transfer Protocol. 1988. Available online: http://gallium.inria.fr/~doligez/zmodem/zmodem.txt (accessed on 1 July 2018).
19. Pister, K.S.J.; Doherty, L. TSMP: Time Synchronized Mesh Protocol. In Proceedings of the IASTED International Symposium on Distributed Sensor Networks (DSN08), Orlando, FL, USA, 16–18 November 2008.
20. Winter, T.; Thubert, A.B.P.; Clausen, T.; Hui, J.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J. *RPL: IPv6 Routing Protocol for Low Power and Lossy Networks,(RFC 6550)*; IETF ROLL WG, Technical Report; IETF: Fremont, CA, USA, 2012.
21. Jensen, S.K.; Pedersen, T.B.; Thomsen, C. Time Series Management Systems: A Survey. *IEEE Trans. Knowl. Data Eng.* 2017, *29*, 2581–2600. [CrossRef]
22. Biow, C. MongoDB for Time Series Data. 2015. Available online: https://www.mongodb.com/presentations/mongodb-time-series-data (accessed on 1 July 2018).