Development Engineering 4 (2019) 100040

Contents lists available at ScienceDirect

Development Engineering

journal homepage: www.elsevier.com/locate/deveng

Towards a robust and affordable Automatic Weather Station

Mary Nsabagwa[a,*], Maximus Byamukama[b], Emmanuel Kondela[a], Julianne Sansa Otim[a]

[a] School of Computing and Informatics Technology, Makerere University, Kampala, Uganda
[b] School of Electrical Engineering, Makerere University, Kampala, Uganda

ARTICLE INFO

Keywords:
Automatic Weather Station
Wireless Sensor Network
Robust and affordable AWS
AWS prototype

ABSTRACT

The frequency and severity of extreme weather events have increased over the last 30 years, making predictability of weather a challenge. Weather extreme events often cause adverse impacts to lives and property. Thus, accurate and timely provision of weather data is becoming crucial to improve the skill of weather prediction and to strengthen resilience to the impacts of the adverse weather conditions. Uganda and many developing countries have challenges in acquiring accurate and timely weather data due to their sparse weather observation networks. The sparse weather observation networks are in part attributed to the high cost of acquiring an Automatic Weather Station (AWS) and limited funding to national meteorological services of the respective countries. The inability of developing countries to manufacture their own AWSs leads to high recurring costs accruing from importation and maintenance. In this study, we propose an AWS based on Wireless Sensor Networks. We plan to design three generations of the AWS prototype, the first being the subject of this paper. The purpose of this paper is therefore to evaluate the first-generation AWS prototype and to propose improvements for the second-generation, based on needs and requirements. Results from the AWS prototype data suggest improving non-functional requirements such as reliability, data accuracy, power consumption and data transmission in order to have an operational AWS. The non-functional requirements combined with cost reduction produces a robust and affordable AWS. Therefore, developing countries like Uganda will be able to acquire the AWSs in reasonable quantities, hence improvement in weather forecasts.

1. Background and motivation

Weather is the present condition of the atmosphere over a given place and time, measured in terms of variables including precipitation, temperature, wind speed and direction and humidity among others (K. et al., 2010). Weather can change over a short period of time such as hours and days. In the past, weather patterns were easily predictable based on indigenous knowledge e.g. one would tell in which months of the year rains were expected for a given place. Such methods of weather prediction have become unreliable (Hansen et al., 2012). That is, the rains come when they are least expected.

When weather conditions are observed over a long period of time, it informs the climate of a region. The climate is an average of the ensemble of weather parameters over a long period of time and the World Meteorological Organization has recommended a period of 30 years. Over the last century, a general global warming and heavy rains have been observed, leading to increased flooding and droughts in different parts of the world, which suggested changing climate and is of present concern globally (Hansen et al., 2012; Dube et al., 2016). In some regions of East Africa, seasonal rains often come early or late, are poorly distributed and often below normal (Hussein, 2011; J. Mubiru et al., 2015). Uganda being a predominantly agricultural country has faced adverse weather effects such as drought, which has caused famine in some parts of the Eastern region. The study on the economic impact of drought on agriculture in Uganda indicated a decline of 5% in the Gross Domestic Product (GDP), which also resulted in losses in the industrial sector (Kilimani et al., 2016).

In order to adapt to the changing weather patterns, many approaches have been suggested (Cairns et al., 2013; Lwasa et al., 2014; Cooper et al., 2008). One adaptation approach, which is the subject of this paper, is to improve the accuracy of weather predictions. In this era, where weather is unpredictable and indigenous knowledge about weather has failed, there is a need to have accurate and timely data collection and transmission. This necessitates robust weather instruments to provide accurate and timely weather data, which can be achieved with increased density of weather station networks, hence increased precision of readings and a better representation of areas under observation. This is however, not the case with some countries like Uganda where a sparse network of weather stations was observed (Nsabagwa et al., 2016; Sansa and Waisswa, 2012). Furthermore, many

* Corresponding author.
E-mail address: mnsabagwa@cit.ac.ug (M. Nsabagwa).

https://doi.org/10.1016/j.deveng.2018.100040
Received 16 May 2018; Accepted 29 November 2018
Available online 30 November 2018

of the weather stations in such countries are manually operated and rely on traditional means of data collection and processing. Such methods pose challenges such as delays in data delivery and human errors in data handling among others. This calls for better weather data collection techniques such as those used in Automatic Weather Stations (AWSs).

An AWS consists of sensors, which automatically collect and transmit weather data. If these AWSs are deployed in big numbers, the reliability and accuracy of their data is improved, hence accurate weather predictions. However, the high cost of the available AWSs hinder their acquisition in big numbers for some developing countries. Additionally, since most commercial AWSs are normally manufactured and assembled from outside the developing countries, their maintenance is normally costly. AWS maintenance often involves import tariffs and additional costs such as consultation fees and transport, which require high investments. Therefore, there is a need to design affordable AWSs.

To achieve the above, we designed the first-generation AWS prototype. This paper presents evaluation results of the first-generation AWS prototype, the first of three prototypes and makes recommendations for an improved second-generation prototype, which shall be assembled and deployed in Uganda, Tanzania and South Sudan.

2. Requirements specification for an Automatic Weather Station

Proper understanding of needs leads to a usable and acceptable systems. Therefore, in order to provide an AWS, which satisfies both end-user and system expectations, we analysed the requirements of a typical AWS. We categorise these requirements under functional and non-functional requirements.

2.1. AWS functional requirements

Functional requirements capture the behaviour of the system and are expressed as tasks or services of the system (Malan and Bredemeyer, 2001). Based on the definition of functional requirements, we identified the following four functional requirements. That is, an AWS should:

i. Collect weather parameters: which include precipitation, temperature, relative humidity, pressure, wind speed and many others
ii. Process captured data: Data processing involves sensor signal processing, calculating derived information such as dew point, data compression and timestamping collected data among others.
iii. Buffer data: Primary saving of data when collected before sending it to the repository, which is a location where data is stored permanently
iv. Transmit data from the AWS to the repository

2.2. Non-functional requirements

To design an acceptable AWS, it is important to have detailed understanding of the non-functional requirements. While the definition of functional requirements is precise and clear, that of non-functional requirements is usually vague (Glinz, 2007). The non-functional requirements refer to the property of the system such as constraints, system attributes, properties and restrictions. These properties must be satisfied for the system to operate as expected. Besides understanding the non-functional requirements, quantifying and measuring them is a daunting task. Proper quantification of the non-functional requirements makes evaluation of the performance of the AWS easy and leads to an acceptable system. Despite the challenges, we endeavoured to categorise and quantify the requirements based on recommendations from literature.

We made a breakdown of the non-functional requirements using the IEEE specification (IEEE, 1998), which provides the following four sub categories.

i. External Interfaces
ii. Attributes
iii. Design Constraints
iv. Performance Constraints

The external interfaces answer questions of how the system software interacts with people, hardware and software systems. The attributes are concerned with issues like portability, correctness, maintainability and security among others. The design constraint issues involve answering questions on whether there are any required standards in effect, implementation language, policies, resource limits and operating environment. The performance requirements are concerned with issues of speed, availability, response and recovery time of various software functions among others. In the proceeding sub sections, we delve into the non-functional requirements of an AWS under the four IEEE categories.

2.2.1. AWS external interfaces

Our AWS is composed of wireless sensor nodes, which collect, process, buffer and transmit weather data to a storage location referred to as a repository. A node is a device that is capable of performing some processing, gathering sensory information and communicating with other connected nodes on the network. One special node known as a sink collects data from other nodes and transmits it to a remote repository via a gateway.[1] Once at the repository, additional processing may take place. All sensor nodes are powered by solar panels or grid power depending on the location.

Although the AWS automatically collects and transmits data, it requires interfaces through which its operations can be monitored and configured. That is, monitoring individual sensor nodes as well as the whole AWS. A sensor node may have an interface to which connections are made in order to debug or configure it through serial communication.

Support software such as web services (HTTP) on the AWS gateway enable remote access to data but drains the limited wireless sensor resources. In cases where the AWS is off-grid or powered by batteries, running such services on the AWS may lower its efficiency. Fig. 1 is a block diagram of the AWS interfaces and Fig. 2 shows the web visualization of the first-generation AWS prototype data. The wireless sensor nodes and gateway are part of the AWS. The AWS interfaces with the repository and transmission entities through uplink options, which may be Ethernet, fiber optic, Wi-Fi, GSM/GPRS, satellite, VHF/UHF/SHF, sneaker net or copper cable among others.

2.2.2. Attributes

Attributes are properties that define the quality of the system. It is impossible to exhaust all attributes of the system. Additionally, quantitatively evaluating these requirements is a hard task. In Crnkovic and Larsson (2004), attributes are quantified based on properties of components including run-time and life cycle properties. Run-time properties are measured during the execution of the system while lifecycle attributes characterise different phases of development and maintenance of the system. Other classifications of attributes are given in Musa and Alkhateeb (2013). Among the categories, IEEE gives a complete classification of attributes, a reason we chose it for our AWS. Table 1 gives a breakdown of the quality attributes based on ISO 9126 Software characteristics. The table also indicates the main category of non-functional requirements in which the attribute belongs. Some functional attributes such as suitability are non-quantifiable.

Benchmarking of the AWS with a standard instrument is important in determining the accuracy of the AWS. Proper sensor calibration can also ensure data readings are correctly made. Furthermore, AWS siting

[1] The gateway is a device, which acts as the bridge between the WSN and other networks or Internet.

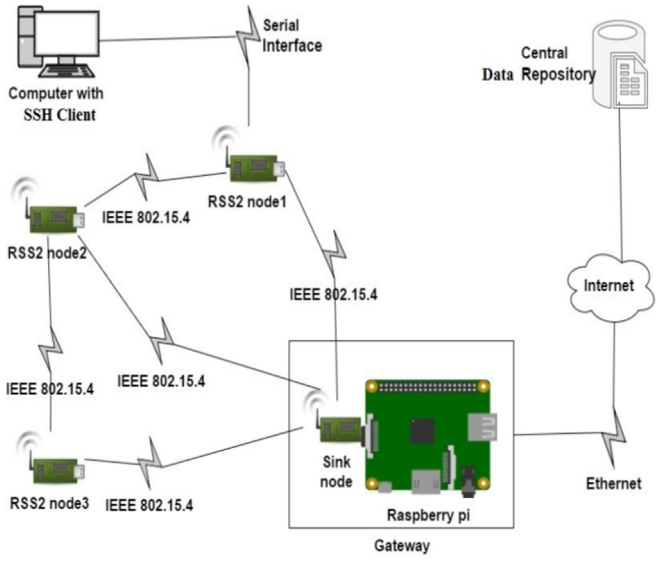

Fig. 1. Block Diagram of an AWS architecture and its external interfaces. The AWS consists of wireless sensor nodes, which collect and transmit data to a sink node connected to a gateway, a device, which transmits data to a repository.

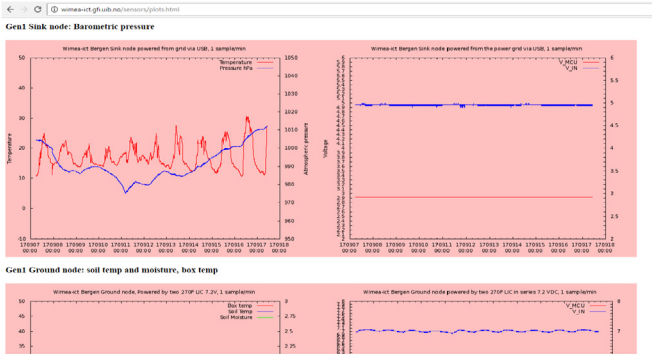

Fig. 2. Screenshot of a web visualization of the sensor node plots. The plots are re-created every 1 min, plotting the most recently generated 10,000 records using gnuplot, using cron tab. Accessed from First-generation AWS Prototype Web Link (2017).

Table 1
AWS Non-functional requirements using ISO 9126 Software Characteristics.

Attribute	Description	Main category	How to achieve the attribute
Suitability	AWS appropriateness to objectives	Functionality	Proper understanding of requirements
Accuracy	Correctness of results	Functionality	Proper sensor selection and benchmarking, Proper AWS siting
Interoperability	Working well with other components	Functionality	Thorough testing (component and system)
Security[a]	Access control	Functionality	Data encryption
Fault Tolerance	Ability to withstand failure	Reliability	Choosing robust components, protect parts from adverse environment
Understandability	Ease of understanding	Reliability	Plug and play facilities, provide user-friendly AWS with accompanying instructions
Learnability	Ease of learning	Usability	Plug and play facilities, simple interfaces, user-friendly instructions
Operability	Ease to operate in any environment	Usability	Test performance under varying conditions over prolonged periods of time
Time Behaviour	Response time and throughput	Efficiency	Minimize processing complexity and data delivery time
Resource Utilization	Efficient use of resources	Efficiency	Minimize energy consumption, memory, CPU processing
Analysability	Ease to identify cause of failure	Maintainability	Provide debugging interfaces, facilitate remote monitoring
Changeability	Effort to change system	Maintainability	Use plug-in features to ease addition of new components
Adaptability	Ability to change to new specifications	Portability	Use plug and play features, Use open source software
Installability	Ease to install	Portability	Plug and play facilities, User manual
Replaceability	Plug and Play aspects	Portability	Plug and play hardware

[a] Data security is given a low priority because accessing small bits of weather data poses no security threat.

may affect data accuracy and should be checked to ensure that it conforms to standards.

Resource utilization is a major concern in the AWS design since sensor nodes are resource constrained. The resources include memory, processing power and energy and data delivery timeliness among others. In regards to timeliness, collected data must be delivered immediately it is collected if it is to be useful. Table 2 gives sample wireless sensor node AWS hardware resource limits of an ATmega128RFA1 (Atmel, 2011) microprocessor-based wireless sensor node, which is what we use in the first-generation AWS prototype.

Table 2
AWS resource requirements.

Resource	Max Available
EEPROM	4K Bytes
Flash	128K Bytes
SRAM	16K Bytes
CPU Active Mode Current	4.1 mA
CPU Deep sleep Mode Current	Less than 250ns
Transmit & receive current	12.5 and 14.5 mA
Frame buffer size	128 Bytes

Hardware features and resource limits have a direct effect on the software size. For example, the firmware size should be less than the size of flash memory. Additionally, the amount of time CPU is active should be minimized to conserve energy. This is because the node consumes less energy when in deep sleep mode than in active mode. In deep sleep mode, neither transmission nor reception occurs. Since transmission and reception of data frames consumes much energy, their frequency may be minimized to conserve energy. IEEE 802.15.4 protocol frames, due for transmission must be less than or equal to 128 bytes to avoid buffer overflows, which normally leads to data loss. If the data size is greater than 128 bytes, it must be split in sizes that fit in the available buffers.

2.2.3. Design constraints

The AWS design should comply with WMO[2] standards provided in the guide for meteorological instruments and methods of observation (World Meteorological Orga, 2008). The guidelines include accuracy of measurements, maximum height requirements and measurement methods among others. Failure to follow such standards may lead to inaccurate readings. We follow the suggested standards when deploying the AWS.

Another design constraint for the AWS is affordability. In order to make the AWS affordable, approaches such as using locally available parts and free and open source software may be employed. Free and

[2] The World Meteorological Organization, which provides a framework for international cooperation in the development of meteorology and operational hydrology and their practical application.

open source software helps to eliminate software-related costs such as licences and subscriptions.

The AWS design choices may be limited by constraints of the components. Components such as wireless sensor nodes are constrained in resources such as memory and computational power. The environment in which the AWS operates may also place additional limitations on the AWS. For instance, AWSs, which are deployed in remote off-the-grid locations, may suffer from insufficient power supplies as opposed to those on the grid. Since data transmission is an energy-consuming activity in wireless sensor networks, the frequency of transmissions must be regulated.

2.2.4. Performance constraints

We evaluate performance in terms of speed and response time of the AWS. Delays may arise at any point of executing the AWS core functions. That is, during data collection, processing or transmission. During data collection, sensors may fail to collect data due to insufficient power, hardware or software faults. In such a case, the only measure is to ensure that general maintenance of sensors such as cleaning and replacements are performed regularly. Remote monitoring of power supplies if used can assist in ensuring that power problems are fixed in time to avoid sensor shut down.

Data processing may involve data aggregation and time-stamping among others. Processing must be performed in a short time to shorten delays in data delivery. Additionally, processing must consume minimal computational resources such as memory, processor time and power. Data aggregation in particular requires packets to wait for others so as to be combined to reduce the number of transmissions. Such a method may be applicable in cases where weather data is required after a long time interval. For example, temperature readings may be required after 1 h while wind readings may be required more frequently.

We use IEEE 802.15.4 (IEEE Standard, 2006) protocol for inter-sensor communication. Performance of the protocol degrades with poor link quality and low Received Signal Strength Indicator (RSSI). The RSSI is affected by many factors, many of which are non-design issues and may vary with time and environmental conditions (Wennerstrom et al., 2013). We recommend checking RSSI values during deployment to establish the best locations for placement of nodes and especially the sink node or gateway.

Data transmission from the AWS to the repository is facilitated by uplinks, which are interfaces that enable access to Internet. Choosing an uplink option over the other depends on resource availability and application requirements. For the case of Uganda, mobile Internet plans were given in Uganda Communications Com and Uganda (2017) gives a map of Uganda, indicating the fibre optic cable coverage. Table 3 gives a SWOT analysis of the AWS uplink options.

Sneakernet is the cheapest uplink option. However, since data is required with minimal delays, sneaker net may only apply when the AWS is in an accessible location, there are insufficient or no funds or in cases where timeliness in data delivery is of no concern. Since available resources and location determine the cost of setting up the uplink, a design that supports all uplink options gives a user a chance to select and use the best option for a particular deployment. We therefore left the option of choosing the most preferred uplink option open. Table 4 shows estimated uplink set up and operational costs.

Table 3
22 SWOT analysis of AWS uplink options.

Uplink	Strengths	Weaknesses	Opportunities	Threats
Copper	Low set up cost	Short distance connections, Low bandwidth, low speed	If AWS is close to an office, setup costs are eliminated.	Affected by temperature and human accidents.
Fibre Optic	Low operational cost, low cost of ownership, high bandwidth, reliable, fast speeds	High set up costs in areas without fibre, not readily available, Low coverage especially in Africa	Very cheap if infrastructure is available	May be affected by cable cuts during road repairs
GPRS/GSM	High coverage, No set up costs	Costly to send huge data volumes	Can take advantage of promotional offers	Network Failure especially in times when mobile users are busy
Satellite	Big geographical coverage	High operational costs	Excellent for long distance communication	Prohibitive costs
VHF/UHF/SHF	Low operational costs,	Requires Licences to operate, expensive to set up	The most feasible solution for areas without any form of infrastructure	Interferences for poor instalments
Sneakernet (Physically copy and transfer data using media of choice)	Low cost	Needs physical human presence hence infeasible in hard to reach areas, Delays in data delivery	The cheapest form of transmitting weather data	Places may be inaccessible

3. AWS design

In this section, we present the first-generation AWS prototype design. The AWS prototype is based on Wireless Sensor Networks, communicating wirelessly using IEEE 802.15.4 protocol. The choice of sensors used was a decision made by meteorological experts (Björn Pehrson, 2014). Since WMO recommends measuring various weather parameters at specific distances from the ground, we placed various nodes at three different heights from the ground. That is, 10 m, 2 m and near the ground. We also added a sink node to receive data from all

Table 4
Estimated Connection and operational costs in USD for uplink options.

Uplink option	Cost/month/1 GB	Set up cost
Ethernet	5	150
Fibre Optic	250-450 per 512 Kbps	1500
GSM/GPRS	5.5–10	28
Satellite	5–250	5000

other nodes. The sink node is connected to a gateway, a Raspberry Pi, which transmits data to a repository and was placed approximately 80 m from the other three nodes. We selected Ethernet as an uplink option because the AWS was close to a networked building, which made connection easy and cheap. Each node had its own power supply and was equipped with a radio transceiver to transmit data to the sink node. Table 5 shows the AWS prototype parameters measured.

The gateway, runs a web server that stores AWS prototype database, a text file containing rows of data in form of key-value pairs. The database file can be downloaded via HTTP, using a browser of choice. The remote repository to which data should permanently be stored, was only receiving and storing data in a text file although we plan to implement a relational database for efficient querying. Besides weather parameters, the database also stores parameters such as date and time of data collection, time zone, latitude and longitude, status of sensor nodes and wireless signal information. The AWS is designed in a way that new sensors are automatically detected by neighbouring nodes.

Remote monitoring of the AWS is performed using web browser and data received via TCP ports at specific intervals. Real-time plots of the collected data show variation of link quality, weather parameters and voltages over time.

4. First-generation AWS prototype evaluation

We deployed the prototype and monitored it for a period of 5 months. This period gave us an opportunity to understand the behaviour of the AWS in both hot and cold weather conditions. Our AWS was deployed alongside the official weather station for benchmarking purposes. Engineers and meteorologists monitored the AWS prototype performance. Meteorologists recommended sensors for use and performed data analysis to determine sensor accuracy. In cases where we used multiple sensors to measure a single parameter, analysis results helped to select the best sensor for use in the second-generation AWS prototype. The engineers were tasked with assembling, deploying, maintaining and monitoring the AWS to ensure continuous service.

We evaluated the AWS performance in relation to the non-functional requirements identified. Issues of concern include resource limits like power consumption, space requirements, AWS cost and robustness of the system among other things.

We assessed whether the AWS performed the required functions but most importantly whether it satisfied the non-functional requirements specified in section 2. We noted that the AWS prototype performed the four functional requirements. However, some performance parameters needed improvement for the second-generation AWS prototype to achieve better performance. Below are the questions we had to answer during the evaluation process:-

i. Does the AWS prototype possess the necessary quality attributes?
ii. Are the design constraints satisfied?
iii. Is the AWS prototype performance satisfactory?

4.1. Quality attributes

Table 6 shows the quality attribute benchmarks we evaluated.

4.2. Design constraints

We made a breakdown of the design constraints in three categories, namely:- affordability, standards and constraints. Affordability is concerned with all forms of costs associated with producing, operating and maintaining the AWS. We expect the AWS costs to be lower than those of the existing commercial AWSs since we plan to use free and open source software in the later generations of the AWS. The nodes use contiki (Dunkels et al., 2004), which is a lightweight and flexible embedded operating system. The small size of contiki image enables it to fit on the available 128K of the node's flash memory. Additionally, contiki is free and open source, attracting a wide range of developers.

Because of our operating system choice, we use C programing, a language used in developing contiki operating system. In case any other operating system is to be used, the supported language may be adopted.

We emphasize the use of locally available parts to further lower the cost of production.

4.3. Performance constraints

We evaluated performance in terms of the AWS timeliness in data delivery and response time in order to ensure that response time is minimized and data is delivered in real-time.

5. Results

In this section, we present results of our evaluation in regards to the main categories of non-functional requirements i.e. attributes, design and performance constraints.

5.1. Quality attributes

5.1.1. Reliability

Reliability is an important quality attribute of an AWS. We evaluated the AWS reliability through assessing the ratio of downtime and uptime. We computed the expected number of packets from 7th July 10:27 to 4th December and got the percentage packet loss from the actual number of packets received during the time. Table 7 shows the packet delivery ratio of the four nodes from 7th July to 4th December.

5.1.2. Data accuracy

A comparison of first-generation AWS prototype data with an already benchmarked AWS, while showing a good correlation, still exhibits some degree of variance in some regions. Fig. 3 shows a scatter plot of solar insolation recorded by the BPW20R sensor of the AWS prototype versus a standard AWS from 7th July to 27th August.

Figs. 4 and 5 shows the scatter plots for temperature readings of the temperature readings of against a benchmarked garden AWS with and

Table 5
First-generation AWS prototype weather parameters measured (sensors are specified in Björn Pehrson (2014)).

Node	Location with Ref to the ground(m)	Weather Parameter
Ground	0	Soil Temperature, Soil Moisture
2m	2	Temperature, Humidity
10m	10	Wind Speed, Wind Direction, Humidity, Solar Insolation
Sink	Depends on Good Signal Value	Temperature, atmospheric pressure, Humidity

Table 6
AWS Quality attribute benchmarks.

Attribute	Resource/Property	Limits
Accuracy	Proper sensor selection, ideal AWS environment	Less than ± 0.5
Fault Tolerance	Choosing robust components, protect parts from adverse environment	Not less than a Year warranty
Time Behaviour	Processing time data transmission time	Deliver data in less than a minute
	Data transmission	
Resource Utilization	Energy consumption	Current draw in active state: 10 mA
	memory	Firmware size < 128 KB
	CPU processing power	

Table 7
AWS packet delivery ratio (Ratio of packets Received).

Node	Expected	Received	%received	%Dropped
2M	216812	171732	79.21	20.79
10M	433624	313741	72.35	27.64
GND	216812	181099	83.53	16.47
Sink	216812	215055	99.19	0.81

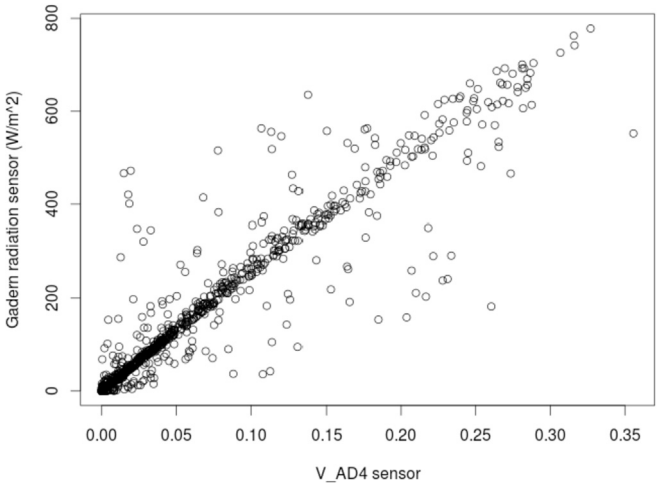

Fig. 3. A scatter plot of the AWS solar irradiance against a benchmarked AWS. A correlation of 0.65 was observed from the first-generation AWS prototype data compared with the standard AWS.

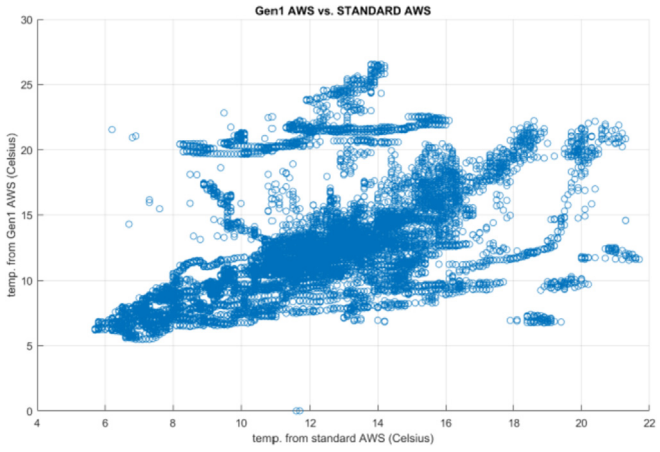

Fig. 4. A scatter plot of the AWS SHT25 temperature readings against a benchmarked garden AWS.

without outliers respectively. The correlation value of the SHT25 temperature readings between the first-generation prototype and the standard benchmarked AWS was 0.4137. There were many outliers that

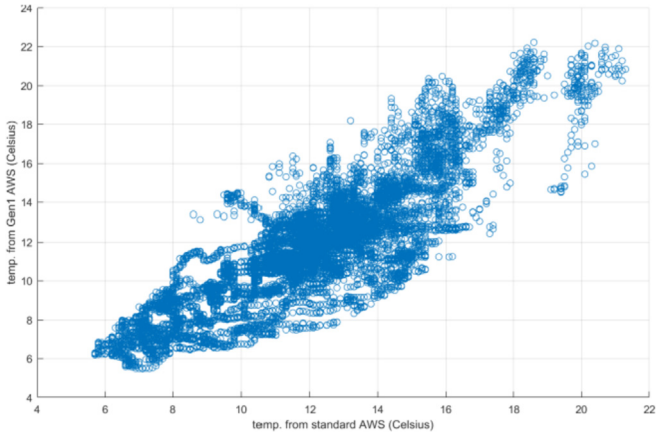

Fig. 5. A scatter plot of the AWS SHT25 temperature readings against a benchmarked garden AWS without outliers (temperature range min −4.5 °C, max 29.2 °C and Relative Humidity of min 17.99 and max 104.26).

can be caused by a number of reasons. Primarily, interferences at the ADC channel pin can have a sudden and large impact on instantaneous values recorded by the microcontrollers. Low power to the sensor nodes can also affect the accuracy of the ADC. We however observed that low power was not an issue. Generally, the first-generation AWS prototype gave higher temperature readings mainly because the sensor was enclosed in a box with limited aeration. This was in comparison to the benchmarked AWS which had a well aerated radiation shield.

We further removed all outlier values that fell within a difference of 5 degrees Celsius from each other, hence retaining over 80% of the original data points. The scatter from the resulting data shows a stronger positive correlation of 0.86.

There were variations in the supply voltages of the nodes. The variations were significant for the three nodes including 2m, 10m and gnd nodes because they relied on solar power, which was intermittent while the sink node was directly connected to grid power, which was more stable. Fig. 6 shows the variation of supply voltage of all nodes over a period of 300 hours.

5.2. Design constraints

5.2.1. Power consumption

Power consumption is a design constraint, which affects the lifetime of the AWS. The AWS nodes are powered by solar panels. Since solar panel energy is a limited resource, the selected nodes should consume little power. Typically, each sensor node consumes about 45 μW. This is very low power consumption in comparison to what other peers have achieved in this area. One node can run for over 6 weeks on a single charge of a 270 F lithium-ion capacitor (Robert Olsson, 2015). Weather stations with one solar panel have a single point of failure in regards to energy harvesting and are thus less reliable. Also, the use of miniature solar panels reduces attractiveness to vandals, which has been proven to be a major concern in developing countries (Nsabagwa et al., 2016).

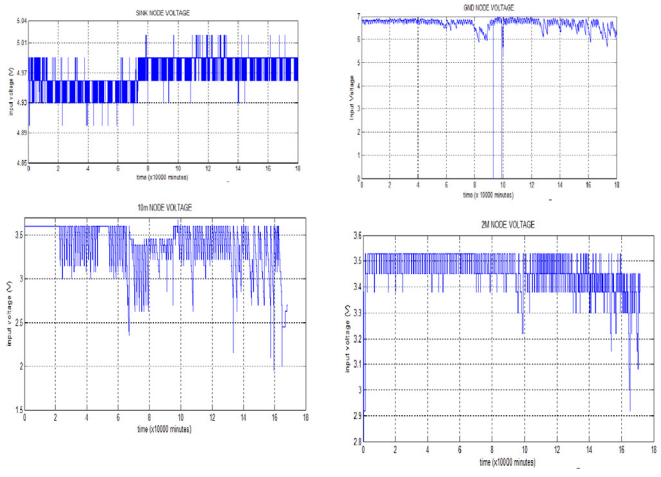

Fig. 6. Clock-wise from Top left are the supply voltages for sink, gnd, 10m and 2m nodes. The sink node had the most stable voltage since it was powered from the grid. On the other hand, 10m suffered from power problems due to the power supply failure, hence going to as low as 1.9 V and sometimes off.

5.2.2. Data buffering and transmission

Sensor nodes transmit data to the sink node without buffering it. In the event of the sink failure, all data lost is never recovered. In order to support data buffering at the sensor node, there are TX/RX buffers, which take a maximum of 128 bytes. Since a single IEEE 802.15.4 frame may take up to 127 bytes, the space becomes insufficient. The challenge with the current design is that whenever the gateway goes off, all data from other nodes is lost.

A total of n rows were recorded every minute, each row taking an average of y bytes. The file therefore increases by $(n.y)$ bytes every minute if all nodes transmit. The amount of buffer memory in bytes required per node in a month, which has d days is given by equation *(1)*.

$$size = n.\ y.\ 60.24.\ d \tag{1}$$

In regards to our AWS, n = 6, y = 192 bytes. In a month, of 30 days, the required raw data bandwidth is 497 MB. The data was sent in its raw form, without any form of compression because compressing it may need waiting for data to accumulate, causing delays in delivery.

Besides the raw data, additional bandwidth overhead is experienced in terms of software installation and upgrades. Additional bandwidth is also required for upload traffic such as accessing data via HTTP. That is, in form of downloading the AWS prototype data file and accessing plots. This extra bandwidth may amount to 500 MB, making the required monthly bandwidth go up to 1 GB.

5.2.3. Learnability and maintainability

Below are the steps required to set up the AWS for operation: -

i. Assembling power supplies for the three nodes
ii. Powering the nodes
iii. Configuring the nodes
iv. Attaching nodes to the stand
v. Configuring the gateway
vi. Deploying
vii. Continuous monitoring AWS operation

Each of the steps involves various levels of complexity and expertise. Node configuration is performed via a serial command-line interface. That is, one has to physically connect the node, which increases maintenance costs in case the AWS is in a remote location. Additionally, users require prior basic networking knowledge to perform the configurations.

Table 8
Cost estimates for Gen 1 AWS.

Category	Equipment	Quantity	Unit Cost	Total Cost
Production	RS Motes	4	67	268
	White enclosures	4	7	28
	Temperature Sensor Radiation shield	1	120	120
	Photo Diodes	1	23	23
	Daughter Cards	3	49	147
	Power Supply Unit	4	57	228
	Raspberry Pi	1	96	96
	Modem	1	40	40
	Memory Card (4 GB)	1	5	5
	Diffusor (3 mm)	1	6	6
	Wind Gauge	1	170	170
	Soil Moisture sensor	1	57	57
	Soil Temperature Sensor	1	12	12
	Stand fabrication + Material	1	300	300
	Solar Panels	1	100	100
	Other Consumables	1	200	200
Operation	Internet Subscription/Month			~1 GB
	Power (on-grid stations)/ Month			
Maintenance	Replacement of spare parts			
	Transport			
Total				**1800**

Gateway configuration requires technical knowledge including setting up TCP clients, Internet connectivity, web server configuration, cron tabs and many other services that facilitate remote access. For new users, setting up these many services makes learning a complicated process, hence a hard to learn process, calling for an easier interface.

5.2.4. AWS costs

The cost of production, operation and maintenance has an impact on the affordability of the AWS. For that reason, we strive to keep the cost as low as possible. One way of lowering the cost is through promoting the use of free and open-source software, which we plan to provide in the second-generation prototype. Table 8 gives an estimate of the cost of the AWS. We have eliminated operational and maintenance costs because they vary with service provider and AWS type. Our AWS cost has been estimated at USD 1,800, which is lower than a commercial AWS such as (Scientiic Sales, 2017), which are approximately USD 7000.

5.3. Performance

In regards to the AWS response time, data is delivered in less than a second after it is collected, which is an indication of good performance. Data speeds may be affected by the network speed, which is a non-design aspect.

6. Discussion

There are several issues of the prototype, which need to be improved in order to get an affordable and robust AWS.

6.1. Power consumption

The prototype power consumption can be reduced by changing the gateway, the raspberry Pi. The gateway design, which is based on an operating system incurs overhead due to the high-level software stacks. This is as opposed to the bare metal embedded applications. The Raspberry Pi requires 700 mA current draw in active mode (Vladimir Vujovic, 2014). On the other hand, sensor nodes draw approximately 8 mA when in active mode. The power consumption can be reduced in two ways in order to exploit even the low levels of solar irradiance such as during cloudy days. First, by changing the gateway platform from the

raspberry pi to a more power-lean one while retaining the functionality. Secondly, by trading some functionality for lower power performance; for example, by eliminating the need for remote access via SSH, reducing the frequency of data transmission or by completely doing away with an operating system and using a custom embedded platform such as a low power microcontroller. These two options can be combined to save even more power. Duty cycling methods can also be looked into to reduce the sensor node power consumption.

For areas with grid power, power consumption is not a big concern. Embedded computers like the Raspberry Pi would take up to 40 days to consume 1 kWh of electricity. In areas without access to grid power however, such as rural areas in Africa, the need for low power devices is extremely important. In such cases, gateway functionality would have to be implemented on low power microcontrollers, with custom power management programs such as sleeping during inactive states. The ATMEGA128RFA1, the same used in the motes, is a very good contender to act as a sink node and gateway at the same time.

6.2. Power supply

The power supply design is essential for battery-powered AWS components such as sensor nodes and low-power gateway microcontrollers. The power supply design for grid powered AWS are trivial. This system consists of the energy-harvesting unit, which is a solar panel in most cases, the battery and some power electronics. The power supply design varies greatly depending on the intensity and duration of sunshine and the nominal voltage of the selected battery technology.

In areas with poor sunshine, such as the frigid zones, the solar panel power rating must be much larger than the rated power of the consuming component. As such, the supply voltages from the panels are usually high and must be stepped down. In our AWS prototype, the panels were rated 21 V and we used a step-down converter from the input range 4.5–25VDC from the solar panels to 3.6VDC, to charge a super capacitor powering the node.

In deployments with high solar intensity and duration, smaller solar panels have been tested. Some of these generate voltages as low as 2.5 V. These can be stepped up to the battery voltage, such as 3.6 V. The use of these power electronics is essential because the intermittent sunshine doesn't ensure constant voltages on the solar panels.

The battery technology to use is also a major consideration. On one hand, traditional battery technologies such as lithium-ion batteries have high energy densities and high internal resistances and as such, are hard to charge when sunshine is intermittent. On another hand, emerging technologies like Lithium-ion capacitors (LICs) have extremely low internal resistance and can charge in seconds. Also, compared to batteries, LICs have a smaller energy density. However, with regards to the AWS application, the energy density from LICs is sufficient to provide power for several days to the motes. Their use to power a low-power gateway is currently untested.

6.3. Data transmission

The first-generation AWS prototype uses raspbian, a Linux-like operating system. The gateway supports a number of communication devices. The first-generation AWS prototype uses Ethernet but can also use USB dongles for Ethernet, Wi-Fi or cellular communication. This sub-problem is concerned with two things. First is choosing the most reliable and cost -effective way of transmitting data from the AWS to the central repository given the available resources at the AWS installation site.

A choice has to be made between cellular network connection, other terrestrial wireless options (VHF/UHF/SHF), copper wire, optic fibre, satellite, sneakernet or a combination of some of these. The selection has to be made while keeping in mind the reliability of the selected data transmission option, the short and long term costs of using and maintaining the said option as well as the power consumption, data integrity and other issues.

Second is the consideration that a change in the gateway device could imply adjustments in the uplink devices to be used. Less advanced embedded systems without in-built controllers for standard interfaces such as USB or Ethernet will require external controllers and, hence, developing drivers for these controllers.

An assessment of data stored in the database file reveals duplication of data. Data including time zone, Latitudes and Longitudes and MAC address, which never changes for a given node and location, wastes bandwidth. Sending such duplicated data wastes resources including power, bandwidth and storage space. Other parameters such as TTL, RSSI and LQI may only be transmitted if there is a change in the value. Doing so shall lower costs of transmitting data.

6.4. Weather parameters

Rainfall measurements, which are very important especially for East Africa, are missing in the first-generation AWS prototype. We shall investigate and choose a rain sensor for inclusion in the second generation AWS prototype.

6.5. Reliability

Temperature data correlations were non-linear especially for July. This was because the AWS prototype temperature sensor was not enclosed in a radiation shield, which made heating of the sensor due to the direct sun and increasing the values of the readings.

In regards to the number of packets received, the sink node dropped the least number of packets while 10 m node lost the most. The sink node's high packet reception rate was because it was directly attached to the gateway and as such dropped no packets during transit. This is as opposed to the other nodes, which were situated close to 80 m from the gateway. Whenever the RSSI environmental conditions changed, packets would be dropped. Besides dropping packets in transit, the 10m node suffered from power failures due to faults in the circuitry, which often caused it to go off. Furthermore, the wind sensors attached to the 10m node required higher voltages that was available.

7. Conclusions and future work

Automatic Weather Stations play a major role in weather information management since they provide timely and reliable data, hence higher chances of accurate weather predictions. It is however challenging to achieve the timeliness and accuracy if AWSs are costly or non-robust. In this paper, we have evaluated our first-generation AWS prototype, which is the first of three prototypes to be designed and tested. We assessed functional and non-functional requirements of an AWS, and used them to benchmark performance of our first-generation AWS prototype, which indicated a shortfall of some of the non-functional requirements. Thus, improvements are required in the second-generation prototype. We hope by producing affordable and robust weather stations, the country shall be able to increase the density of the weather stations, hence increase in weather data for research purposes and improved accuracy in forecasting. Researchers may evaluate the impact of the AWSs by assessing the number of operational WIMEA-ICT AWSs installed in the country as well as the amount of weather data the AWSs contribute towards the forecasting process. Increased access to weather information by researchers fosters innovation in weather related services, which in the end improves lives of people and businesses. Next we intend to undertake the following:

- Setting up second-generation AWS prototypes in Uganda, Tanzania and South Sudan to test in these operating environments and weather conditions
- Designing low-power gateway, which eliminates high power-consuming tasks and handles basic functions such as data collection and

transmission

- Research on using smaller solar panels that are less attractive to vandals.
- Developing a free and open-source wireless sensor node application to enable customization e.g. data buffering at node level.
- Include a rain gauge.
- Provide for a radiation shield to protect the temperature sensor from the direct sun which results into wrong readings.
- Work with meteorological authorities in benchmarking the AWS.

Following satisfactory developments on this AWS, we plan to mass-produce and deploy 70 units in the East African region. This deployment will lead to a denser weather station network hence increased precision of weather readings and accuracy in weather predictions. Other developing regions with similar challenges will have access to the free and open-source designs and software of this AWS hence can take advantage to customise their own AWS and share in its benefits. We hope to improve on the volume of weather data, hence accurate and timely weather forecasts.

Conflicts of interest

There is no conflict of Interest that we are aware of.

Acknowledgements

We would like to acknowledge the financial support of NORAD (Agreement number UGA-13/0018). We are also grateful to the meteorological services of Uganda, Tanzania, Norway and South Sudan for the technical support given, Björn Pehrson and Robert Olsson for the technical input and Isaac Mugume and Lawrence Ssanyu for proof-reading this document.

References

Atmel, 2011. Microcontroller with Low Power Transceiver for ZigBee and ATmega256RFR2 ATmega128RFR2 ATmega64RFR2 1 Pin Configurations, vol. 3. pp. 1–14.

Björn Pehrson, R.O., 2014. WIMEA-ICT research component 3. In: Automated Weather Station (AWS) Development and Network Densification.

Cairns, J.E., et al., Jun-2013. Adapting maize production to climate change in sub-Saharan Africa. Food Secur. 5 (3), 345–360.

Cooper, P.J.M., Dimes, J., Rao, K.P.C., Shapiro, B., Shiferaw, B., Twomlow, S., Jun. 2008. Coping better with current climatic variability in the rain-fed farming systems of sub-Saharan Africa: an essential first step in adapting to future climate change? Agric. Ecosyst. Environ. 126 (1–2), 24–35.

Crnkovic, I., Larsson, M., 2004. Classification of quality attributes for predictability in component-based systems. J. Econom. 3549, 257–278.

Dube, T., Moyo, P., Ncube, M., Nyathi, D., 2016. The impact of climate change on agro-ecological based livelihoods in Africa: a review. J. Sustain. Dev. 9 (1), 256.

Dunkels, A., Grönvall, B., Voigt, T., 2004. Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proceedings - Conference on Local Computer Networks. LCN, pp. 455–462.

First-generation AWS Prototype Web Link.

Glinz, M., Oct. 2007. On non-functional requirements. In: 15th IEEE Int. Requir. Eng. Conf. (RE 2007), pp. 21–26.

Hansen, J., Sato, M., Ruedy, R., 2012. Perception of climate change. Proc. Natl. Acad. Sci. U. S. A. 109 (37) E2415–23.

Hussein, M.A., 2011. Climate change impacts on East Africa. Mitig. Adapt. Strategies Glob. Change 589–601.

IEEE, 1998. IEEE Recommended Practice for Software Requirements Specifications, vol. 1998.

IEEE Standard, 2006. IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-local and Metropolitan Area Networks-specific Requirements-Part 15.4 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std. 802.15.4-2006. pp. 1–26.

J. Mubiru, Drake N., Kyazze, Florence B., Radeny, Maren, Zziwa, Ahamada, Lwasa, James, Kinyangi, 2015. Climatic trends, risk perceptions and coping strategies of smallholder farmers in rural Uganda. In: CGIAR Res. Progr. Clim. Chang. Agric. Food Secur..

K, D., 2010. In: In: Pike, R.G., Redding, T.E., Moore, R.D., Winkler, R.D., Bladon (Eds.), Compendium of Forest Hydrology and Geomorphology in British Columbia, LMH 66, vol. 2 Volume 2 of 2.

Kilimani, N., Van Heerden, J., Roos, L., Van Heerden, J., Bohlmann, H., Roos, L., 2016. Counting the Cost of Drought Induced Productivity Losses in an Agro-based Economy : the Case of Uganda the Case of Uganda. no. June. .

Lwasa, S., Mugagga, F., Wahab, B., Simon, D., Connors, J., Griffith, C., 2014. Urban and peri-urban agriculture and forestry: transcending poverty alleviation to climate change mitigation and adaptation. Urban Clim. 7, 92–106.

Malan, R., Bredemeyer, D., 2001. Functional requirements and use cases. Architect. Resour. 1–10.

Musa, K., Alkhateeb, J., 2013. Quality model based on COTS quality. Int. J. Software Eng. Appl. 4 (1), 3–11.

Nsabagwa, O. Mary, Byamukama, Maximus, Otim, Julianne Sansa, Richard, 2016. Network densification strategies for automatic weather stations: challenges and opportunities for Uganda. In: IST-Africa Week Conference.

Robert Olsson, B.P., 2015. Powering devices using ultra-capacitor batteries. In: AFRICON, 2015.

Sansa, J.O., Waisswa, M., 2012. On improvement of the weather information management in Uganda. In: Proc. Rep. 5th UbuntuNet Alliance Annu. Conf. pp. 9–14.

Scientiic Sales. Available: https://www.scientificsales.com.

Uganda, N.I.T.A., 2017. The Scope and Benefits of the NBI/EGI to Uganda and the Neighbouring Countries. pp. 8.

Uganda Communications Commission, Mobile Internet Explained.

Vladimir Vujovic, M.M., 2014. Raspberry pi as a wireless sensor node: performances and constraints. In: Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on.

Wennerstrom, H., Hermans, F., Rensfelt, O., Rohner, C., Norden, L.A., 2013. A long-term study of correlations between meteorological conditions and 802.15.4 link performance. In: 2013 IEEE International Conference on Sensing, Communications and Networking. SECON 2013.

World Meteorological Organization, 2008. 2008th ed. Guide of Meteorological Instruments and Methods of Observation, vols. I & II World Meteorological Organization no. 8.