Smart City Solutions to Water Management using Self-Powered, Low-Cost, Water Sensors and Apache Spark Data Aggregation

W. Frank Domoney, Naseem Ramli, Salma Alarefi and Stuart D. Walker
School of Computer Science and Electronic Engineering
University of Essex, Colchester, United Kingdom
{wfdomo, nramlia, sasala,stuwal}@essex.ac.uk

Abstract—**We describe a Smart City, self-powered, water monitoring solution that uses low-cost water turbines. These can generate up to ~ 4 W electrical power whilst reporting flow rates at UK domestic water pressure levels. This permits the novel use of stand-alone, remote Apache Spark Streaming as a distributed alternative to the conventional hierarchical Smart Grid. A future Smart City will have innovative features such as high-frequency monitoring of water flows, automated leak detection and shutdown with no requirement for utility electrical power.**

Keywords—Smart City; water; turbine, Spark, Kafka

I. INTRODUCTION

Smart City schemes often feature sensors, 'big data' and advanced computing [e.g. 1]. At the city utility level, management of increasingly scarce resources is emerging as a key element [e.g. 2]. Here we focus on the energy–neutral acquisition of water consumption data. This latter information is already collected on a large scale by "dumb" and "smart" water meters [e.g. 3].In this paper, we describe a scheme which not only measures water consumption on a per-second basis but which generates sufficient electrical power (~4 W) to run an Apache Spark-based data aggregation network. All the components are modular and commercial-off-the-shelf (COTS) and U.K. Water Regulations Advisory Scheme (WRAS,[4]) approved. Consequently, installation within domestic and commercial buildings across a wide range of flow rates/heads should be reasonably straightforward.

Subsequently, this paper is organized as follows. Section II is a brief theoretical discussion of available water power and outlines the performance obtained from a prototype domestic-environment, hydro-generator. In section IIII, the Apache Spark provision is described, with section IV offering some conclusions and proposals for further work.

II. AVAILABLE HYDRO-POWER

A. Standard Newtonian power calculation

The power available from water flows, both horizontal and vertical, is well-understood [5]. For convenience, the basic Newtonian mechanical equations are reviewed here. The power P in watts obtained from a fluid (density ρ kg/m^3) flow F (m^3/sec) with head h (metres), gravitational acceleration g (m/sec^2) and turbine efficiency η ($0 \le \eta \le 1$) is given by :

$$P(W) = F \rho \eta h g \qquad (1)$$

As an example, in the case of water (ρ ~10^3 kg/m^3) with 1 bar head (a guideline U.K. minimum domestic pressure = 10.3 m head), F = 3.9 x 10^{-4} m^3/sec (a typical U.K. 15 mm pipe flow rate) and η =1, then P = 30.3 W. In practice, the available output will be much reduced but the generation of several watts electrical output power with realistic heads and flow-rates appears realistic.

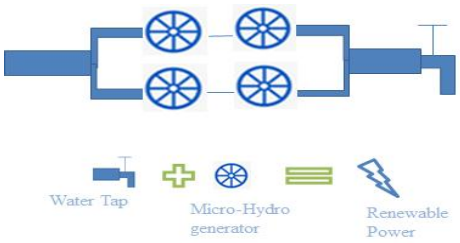
B. Practical Domestic hydro-power set-up and optimization

Figures 1a,b below show two examples of readily-available micro-hydro generators. In both case, the device has magnetic protection against excessive flow rates. The first innovation

a b

Figure 1. Micro-hydros (a) ~15V, 1 watt. (b) ~80V, 2 watts.

introduced during this study was to run turbines in series/parallel. Appropriate combinations can then be optimized for a wide range of flow rates and heads.



978-1-4673-7894-9/15/$31.00 ©2015 Crown

Figure 2. Optimum 2 x 2 configuration

Figure 2 shows the optimum 2 x2 configurations for a U. K domestic supply delivered through a 15 mm pipe. Other implementations, f or example 4 x 2 can be realized for larger flow rates /higher pressures.

As might be expected, the turbine or combination of turbines has an optimum resistive load value. Figure 3

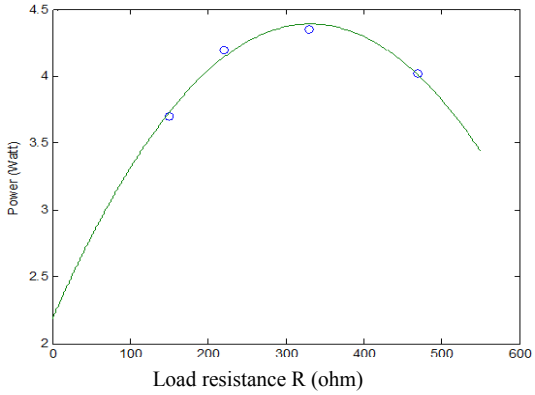


Figure 3. Optimum load investigation.

shows the results of a basic experiment designed to extract the exact matched load. The second-order fit shown in

$$P(R) = -2.0027x10^{-5} R^2 + 0.0133R + 2.184 \quad (2)$$

equation 1 above was obtained where P(R) is in watts and R is the load resistance in ohms. Eqn. 2 can be differentiated as in

$$\frac{\delta P(R)}{\delta R} = -2.0027x10^{-5} x 2R + 0.0133 = 0 \quad (3)$$

Hence R_{opt} = 328 ohm which corresponds to V_{out} = 37 V when P_{opt} =4.2 W.

C. Demonstrator

Four hydro-generators were incorporated in a complete domestic flow demonstrator as shown in figure 4 below.

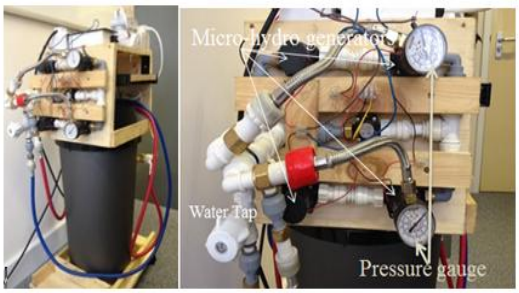


Figure 4. Complete hydro-power demonstrator

The unit was designed to emulate U.K. domestic water pressures, flow rates and demand patterns. A standard DC-DC converter (TRACO TMR 6-4813WI) was used to power a flow meter at 15 V and to be further down-converted to provide a 5V USB supply at up to 800mA output current. Figure 5 shows the set-up.

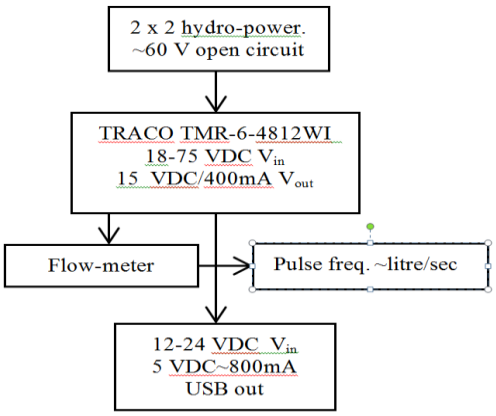


Figure 5. Power and flow-rate measurement system.

III. Data Processing

A. Functionality Description

Processing of the data from the sensors can be done using a simple development board such as the Arduino for prototyping and a microcontroller similar to Intel 8052 or one of the four bit versions in production volumes.

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read multi-format inputs: light on a sensor; a finger on a button; or a Twitter message and turn it into an output. This could be: activating a motor; turning on an LED; publishing online, etc. All these procedures are defined by a set of instructions programmed through the Arduino Software (IDE). [6]. Backhaul from the sensor can be performed using a small cell radio communication to a fibre riser in a high rise building or to a low rise building's fibre termination. Blanket coverage of all new and existing build in cities can be expected to be Fibre to the Home or Fibre to the Kerb.

A block diagram of the arrangement is shown in figure 6.

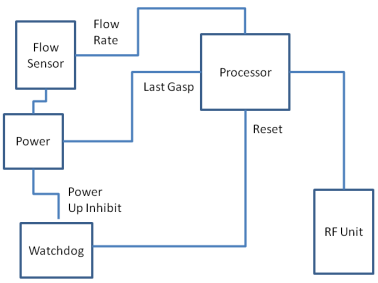


Figure 6. Simplified layout of the Processing Unit

The watchdog circuit assures a reliable reset within 50 milliseconds of the voltage reaching the desired level with subsequent triggering disabled by the processor. For each second that water is flowing, the microcontroller sends a message to an Apache Kafka Node via the RF unit for further distribution. When the water stops flowing, the power supply dies away and can either trigger a last gasp message or simply die. Absence of messages for more than a few seconds means water flow has ceased. A capacitive element can retain charge long enough to smooth out voltage fluctuations due to intermittent water flow.

A suitable transmission protocol might be Twitter [7] which is simple and readily available. Twitter addresses are preceded by an @ sign and messages can be directed to either single or multiple destinations. The extent of addresses is limited only by the number of characters available and with 140 characters available in a Twitter message, an address space of many thousands of millions (36^{24} for an address length of 24 alphanumeric characters = 2.24×10^{37}) of unique addresses is available for originating addresses.

The message set can be very limited as for example:

- Unit powered up
- Flow measured in one second
- Final Flow measured (last gasp message).

These messages may be aggregated at any chosen level from building, street, or city district for control and management.

As the value of individual messages about small water flows is low we think that there is no need for encryption of the messages which report flows. The use of machine learning techniques described below will identify anomalies such as failed sensors, leaks and attempts at theft of services.

Messages which are passed to actuators, valves and taps in the network do however need to be encrypted and transmitted securely using the usual techniques. The marginal cost of adding processing power to these high value items is low.

B. Message processing and direction

Apache Kafka [8] is a distributed publish-subscribe messaging system developed by Linkedin. It has been created with performance, availability and scalability in mind and is used as the messaging backbone at Linkedin. In combination with Apache Zookeeper it allows all the flows in a city to be managed across a distributed architecture.

Apache Spark [9] is a distributed processing environment that is designed to be used to process large volumes of data reliably using distributed clusters. It is supplanting the use of Hadoop in production environments because of its extra speed and capabilities.

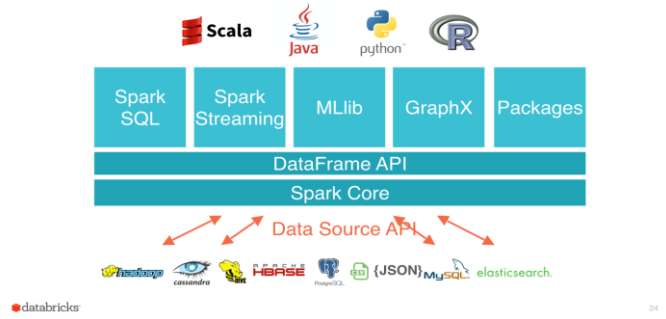


Figure 7. Spark and its components

The components of Spark are shown in figure 2 above. It consists of a Core processing element with a Dataframes API that allows access to distributed datasets. Spark divides data up into Resilient Distributed Data sets that are distributed across portions in clusters .It then has five components that allow processing of SQL Queries across the data sets, processing of streaming data by dividing the streams up into time slices and processing the resulting RDDs as static, a Machine Learning library and a Graph Processing suite. Spark applications may be programmed in Scala, Java, Python and has an R interface.

C. Streaming processing of messages

Spark Streaming allows data to be ingested from Kafka, Flume, HDFS, or a raw TCP stream, and it permits users to create a stream out of RDDs. Spark can then this as a Stream of RDDs. The creation of an individual receiver is also possible.

Fault tolerance is the capability of a system to overcome failure. Fault tolerance in Spark Streaming is similar to fault tolerance in Spark. As with RDD partitions, streams data is recomputed in case of a failure. The raw input is replicated in memory across the cluster. In case of a node failure, the data can be reproduced using the lineage. The system can recover from a failure in less than one second.

Spark Streaming is able to process 100,000-500,000 records/node/sec.[10] If information about water consumption can be processed at one second granularity, a number of interesting options open up. Machine Learning techniques exist in Spark [11] to capture the daily behaviour of a building, street or city district. A sudden departure from the model will indicate a rupture of a water main or the tripping of sprinklers. The Spark Graph Processing facility will identify the location of leaks by means of a graph traverse [12]. Discrepancies between the sum of branch flows and the flow in water mains will indicate a slow leak or an unmonitored branch.

D. Individual building processing units

Typically the Datacentre required to support city size processing will be large and will require a substantial solar farm and very large batteries to support 24x7 operation. Both

Spark and Kafka are designed to be run on distributed processing environments. A cluster may be constructed from low-cost single board computers and networked using a ubiquitous optical fibre network. Examples include Joshua Kiepert's Raspberry Pi-based cluster [13] although the boards offered by Parallella [14] may have far better energy performance that the notoriously energy hungry Raspberry Pi - based solution. The rising water main will have enough flow in general to keep their clusters powered. Again, the option of battery backup exists to store power to keep the building systems supplied when no water is flowing. These batteries can be recharged when major water flows resume.

IV. CONCLUSIONS AND FURTHER WORK

This paper has outlined the main features of a Smart City solution to water management. All the components are COTS and are priced for the mass market. Full functionality of the hydro-generation scheme has been demonstrated with power of ~4 W and self-powered flow rate monitoring available in the domestic scenario. Further work will be concerned with developing a realistic model of a tower block as well as programming the associated Arduino unit. A sizing exercise is needed to identify an ideal address length. The Kafka and Spark elements will then be incorporated into the prototype once these have been built and suitable load generators used to simulate a city. Finally, measurement of the power consumption of the Raspberry Pi Cluster can be compared with an equivalent Parallella cluster running the same algorithm to identify the ideal configuration.

References

[1] A. Caragliu, C. Del Bo and P. Nijkamp, “Smart cities in Europe” Journal of Urban Technology, vol.18, issue 2,pp.65-82, 2011.

[2] R. A. Stewart, R. Willis, D. Giurco, K. Panuwatwanich and Guillermo Capati,”Web-based management system: linking smart metering to the futre of urban water planning” Australian Planner,vol.47, issue 2, pp.66-74, 2010.

[3] S.C. Hsia, S.W.Hsu. and Y.J. Chang, “Remote monitoring and smart sensing for water meter system and leakage detection” IET Wirel. Sens. Syst., vol. 2, issue . 4, pp. 402–408, 2012

[4] Water Regulations Advisory Scheme (2015) [Online]. Available: https://www.wras.co.uk/approvals/what_is_a_wras_approval/

[5] A. Harvey, Micro-Hydro Design Manual: a guide to small-scale water power schemes. Intermediate Technology Publications,London,1993.

[6] Arduino LLC (2014) [Online]. Available: https://www.arduino.cc/en/Guide/Introduction

[7] A Prescott. (2012) Twitter format(7) man page [Online]. Available: http://aprescott.github.io/twitter-format/twitter-format.7

[8] M. Kjetland (2013, 13 March) Booster conference, Bergen, Norway. [Online]. Available: https://vimeo.com/62298867

[9] Korau, Kowinski Wendell and Zaharia (2015 February) " Learning Spark" O'Reilly Media

[10] A Kharbanda (2015, 23 April) [Online]. Available: http://opensource.com/business/15/4/guide-to-apache-spark-streaming

[11] Apache Foundation Spark (2015 July 15) [Online]. Available: http://spark.apache.org/docs/latest/mllib-guide.html .

[12] Apache Foundation Spark (2015 July 15) [Online]. Available: https://spark.apache.org/docs/latest/graphx-programming-guide.html

[13] J. Kiepert (2013, May 22) [Online]. Available: http://coen.boisestate.edu/ece/files/2013/05/Creating.a.Raspberry.Pi-Based.Beowulf.Cluster_v2.pdf

[14] Parallella Inc: (2014) The Board. [Online]. Available: https://www.parallella.org/board/