# Real-time Data Assimilation Potential to Connect Micro-Smart Water Test Bed and Hydraulic Model

Jiada Li[1*], Shuangli Bao[2], Steven Burian[3]

**Abstract:** Recently, smart water application has gained worldwide attention, but there is a lack of understanding of how to construct smart water networks. This is partly because of the limited investigation into how to combine physical experiments with model simulations. This study aimed to investigate the process of connecting Micro-smart water test bed (MWTB) and EPANET hydraulic model, which involves experimental set-up, real-time data acquisition, hydraulic simulation, and system performance demonstration. In this study, an MWTB was established based on the flow sensing technology. The data generated by MWTB was stored in Observation Data Model (ODM) database for visualization in R-studio environment and was also archived as the input of EPANET hydraulic simulation. The data visualization fitted the operation scenarios of MWTB well. Additionally, the fitting degree between the experimental measurements and modeling outputs indicates the data from this MWTB could be compiled to hydraulic model for better understanding of smart water system.

***Keywords:*** Micro-Smart Water Test Bed, Flow Sensor, Arduino, R-studio, Database

[1*]Corresponding author, Civil Department, University of Utah (jiadali2017@gmail.com ) 201 Presidents Cir, Salt Lake City, UT 84112, U.S

[2] Civil Department, University of Utah (shuanglibao@gmail.com) 201 Presidents Cir, Salt Lake City, UT 84112, U.S

[3] Civil Department, University of Utah (steve.burian@utah.edu) 201 Presidents Cir, Salt Lake City, UT 84112, U.S

## 1 Introduction

Smart system, which was first introduced in the field of electricity, has finally reached the drinking water sector to realize the real-time data acquisition, organization, and analysis (Abu-Mahfouz *et al.* 2012)The smart water system(SWS) has received much attention in recent years due to its intelligent functions. SWS was the product of the integration of automated control technology, information communication technology (ICT) and traditional water systems, aiming to solve water issues more efficiently. SWS has been applied in many cases to address critical water problems while traditional water system can't (Dludla *et al.* 2013; Günther *et al.* 2015*)* and it has also been investigated as a potential solution for water leakage issues(Günther et al. 2014; Hatchett *et al.* 2010; Horsburgh *et al.* 2008). Despite broad application, it was that what the SWS is like and how SWS functions still confuse the public.

One way to clear this confusion is to implement the smart water system in water networks, and extensive studies are discussing how to apply smart water system into field cases. For example, (Boulos 2017) developed a cyber-physical concepts based smart flood information system, called Dayu Smart Water System. This intelligent water system created an on-site monitoring network and integrate it into rapid flood modeling to provide updated information. (Bartos *et al.* 2017) demonstrated a dynamic system-level smart stormwater system which was implemented in Anna Arbor, Michigan. However, such smart stormwater system needs long-term storm event records to display its adaptive functions,

Though the application of SWS has been started for many years, little attention was paid to the consensus of the components and the working process of SWS, which impedes the implementation of the smart water system in water networks. One way to improve the public adoption is to educate them by implementing the innovative smart water networks. By knowing SWS better, residents are more likely to accept smart water. To increase the agreement of SWS, many researchers display it as an easy-to-access way. Some studies show the installation, application, testing, interconnection and working rules for social and educational purposes(Dludla *et al.* 2013; Günther *et al.* 2015; Günther et al. 2014; Hatchett *et al.* 2010; Horsburgh *et al.* 2008).

Meanwhile, many studies utilized the experimental set-up to achieve their benefits. For instance, (Kramer 2014)introduced an experimental water distribution system following smart water network principles to configure the physical components and sensing task. Furthermore, most of the experimental studies aimed to localize the pipe leaks and to test leakage algorithm. (Kramer 2014) conducted an experimental study to measure the leakage exponents of different types of leak openings, and longitudinal and pipe materials. To analyze correlation between the reduction of leakage rate to the decrease of pressure, (Machell *et al.* 2010) installed an experimental network to adjust the geometry of the leak and also modify the material of the pipeline. Similarly, (Sonaje & Joshi 2015) constructed a polyethylene water distribution network to investigate the effects of leak area and pipe rigidity on discharge. Apart from those studies on leak area and pipe materials, the leak detection algorithm was also examed in smart water networks. (Steffelbauer *et al.*2014) proposed an efficient wireless sensor armed water distribution system intending to detect and locate leaks for long distance pipelines by combining powerful leak detection and localization algorithms. Also, (Karray *et al.* 2016) introduced a sensor-based lab smart water system called Earnpipe to optimize the reliability of the inspection and improve the accuracy of the water pipeline monitoring.

However, those studies listed above mainly focused on integrating smart water test bed with online sampling but more attention should be paid to the combination between the experimental measurements with hydraulic simulation results. The integration of physical components and simulation model will enable the engineers to localize the pipe leakage more efficiently, and also this is helpful for exhibiting what SWS can do for the community. To achieve this connection, some researchers have made initial progress which identified the future directions. (Günther *et al.* 2014) made full use of the pressure sensor alongside a calibrated hydraulic model to localize the pipe leaks by considering the demand uncertainties.

Interestingly, (Kartakis *et al.* 2015) presented a small scale testbed called WaterBox which enables to compile sensor monitoring data and actuator control algorithms in the simulation process. This work provides a typical reference for understanding how smart water network are configured. The latest study illustrated the process of constructing a laboratory scale water distribution systems to verify the pressure-dependent demands(PDD) modeling in EPANET(Walski *et al.* 2017). However, the manometers installed in this test bed shows disadvantages when compared with sensor monitoring.

To investigate how MWTB can contribute to communities, this paper explored the potential of connecting the smart water test bed and hydraulic model in experimental practice. This work was structured as four parts. The first part was to establish the "two-loop" smart water test bed including sensor installation for data collection and pipes incorporation in the hydraulic lab. The second part was to organize data by designing a relational database schema and storing the data collected in the MySQL database. The third part was to analyze data by connecting Rstudio to the database and visualizing data via R environment. The fourth part was to evaluate the system working status by operating test bed and importing the historic flow measurements to the hydraulic model. Generally, this study aims to foster the public adoption of SWS by creating MWTB. Meanwhile, the process of importing real-time data into the hydraulic model is helpful for educating people to understand how SWS is operated.

## 2 Methods and Materials

In this paper, firstly, this MWTB was designed and set up. And then, the real-time data were collected by flow sensors and transmitted to the Arduino board. The flow data was saved automatically in the Arduino SD card as txt.file. Having been transferred to CSV.file, the flow data was imported into the created MySQL Schema which composed of the Observation Database Model (ODM) (Horsburgh *et al.* 2008). Next, the database was connected with R to achieve visualization and to produce data analysis results. Finally, to test the system performance, the hydraulic modeling results were compared with the experimental flow results to by quantifying the fitting degree.

### 2.1 Micro-Smart Water Test Bed Design

This research designed the “Two-Loop” Micro-smart water test bed by using AutoCAD 2017. This “Two-Loop” water network is a “gravity drive” system composed of 1 source tank, nine pipes, six junctions, three outlets, and two valves. The design details are shown in figure1.

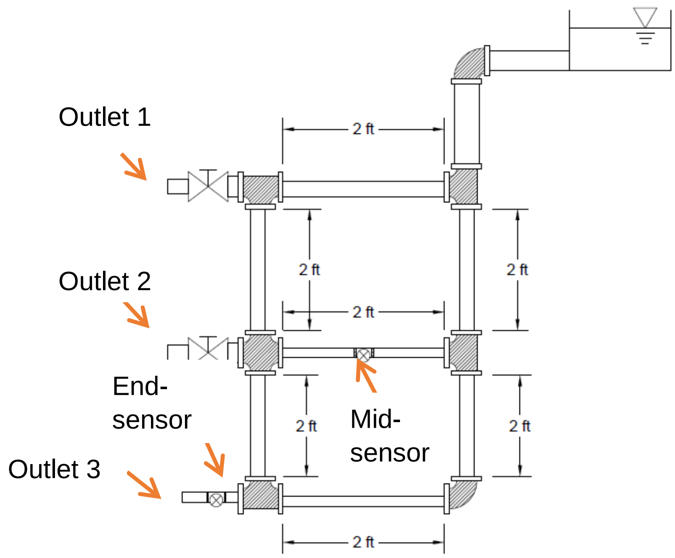



**Figure 1.** “Two-Loop” Micro-Smart Water Test Bed Designing

Additionally, two YF-S201 sensors (Sensor) shown in appendix C-figure 1 are used in this Micro-smart water system. This kind of sensor is made for detecting the gas and fluid flow rate, which sits in line with the water line and contains a pinwheel sensor to measure how much liquid or gas has moved through it. There's an integrated magnetic hall-effect sensor that outputs an electrical pulse with every revolution. Flow sensors use acoustic waves and electromagnetic fields to measure the flow through a given area via physical quantities, such as acceleration, frequency, pressure, and volume. The sensors are solidly constructed and provide a digital pulse each time an amount of water passes through the pipe, more information about this flow sensor presented in (Sensor 2016). Since the sensor is sitting in the water pipe line, to keep the lowest impact caused by the sensor, 1/2” diameter PVC pipe is designed to match the sensor inside diameter. First test sensor is placed at the most end discharge pipe, to determine the lowest head flow rate; the second sensor is placed at the center of the central connection pipe, which could observe the water flow scenario in the most “uncertain” pipe. One concern is turbulence caused by the sensors' minimum distance but this can be avoided completely in this strcuture. To make the experimental set-up fit the configuration of hydraulic model, the distances are prefereably kept as they are.

## 2.2 Hydraulic Model Establishment

The hydraulic model of this “Two-Loop” MWTB was simulated by EPANET 2.0(Rossman 2000). “Two-Loop” hydraulic model is one of the benchmark water distribution models which are open sources for education, research, and commercial purposes. This hydraulic model is a simplified model of the real micro-smart water test bed since there are some assumptions was made. For example, the elevation of nodes was assumed to be zero, and the total head of the tank was assumed to be the water level. The base demand of node 3,5 and seven were assumed regarding the different valve status. The detail of the hydraulic model can be checked in figure 2.

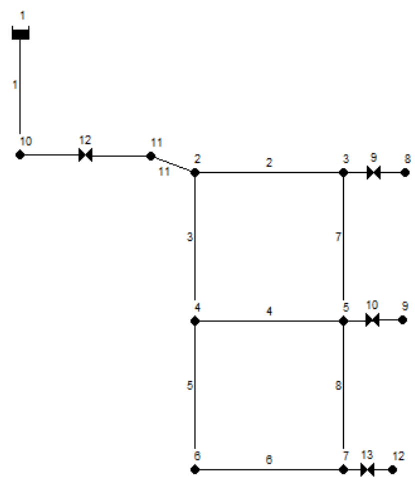


**Figure 2.** “Two-Loop” Micro-Smart Water Test Bed Hydraulic Model

## 2.3 System Setup

The system was built shown in figure 3 follow the design and framework shown in figure 6. The calculation of the system is not always matching the reality, such as the discharge flow rate is smaller than the calculation, so our team adds the extra overflow discharge pipe after purchasing the utility pump. To keep the source tank level stable under different scenarios, an automatic utility with maximum flow rate 2400 Gallons per hour and 10 ft of discharge lift was purchased and installed in the micro-smart water test bed. This pump, which was not for the system pressure adding, was used to supply water from the bottom blue tank to the top source tank. The bottom suction design filters debris and removes water down to 1/8 inch of surface and passes up to 1/8 inch solids.

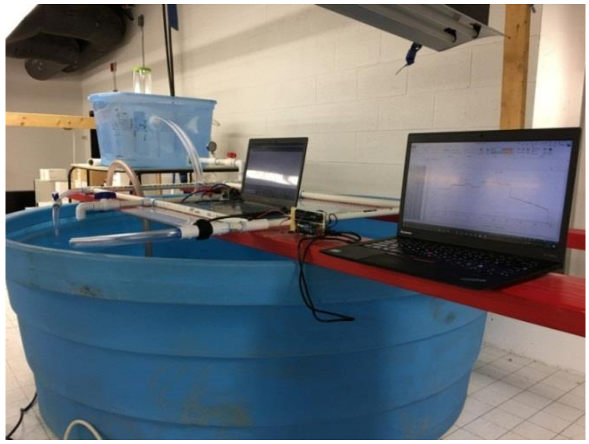


**Figure 3.** Micro-Smart Water Test Bed

The smart water system was built in the hydraulic lab at the University of Utah. The primary pipe material is PVC plastic, including the valve and the pipe cross. The resource tank is made by a customized household storage plastic tank, by making the hole and installing the pipe adapter to make the connection for the water system and the resource tank. The whole system demonstration and the details are shown in appendix A.

## 2.4 Data Acquisition

### 2.4.1 Data Collection

In data collection step, nine different scenarios shown in Table 1 were tested to get nine different data results. To test the water system testing performance, the dataset and the inputs of selected scenarios were imported into EPNET model. All the 1 to 9 scenes are listed in table 1 and figure 4, including its start time and outlet status.

**Table 1.** Testing Scenarios

| | Start Time | Outlet 1 | Outlet 2 | Outlet 3 |
|---|---|---|---|---|
| Scenario 1 | 0 s | open | open | open |
| Scenario 2 | 300 s | open | close | open |
| Scenario 3 | 600 s | close | close | open |
| Scenario 4 | 880 s | close | open | open |
| Scenario 5 | 1200 s | open | open | close |
| Scenario 6 | 1500 s | open | open | open |
| Scenario 7 | 1800 s | Half close | open | open |
| Scenario 8 | 2100 s | open | Half close | open |
| Scenario 9 | 2400 s | open | open | open |

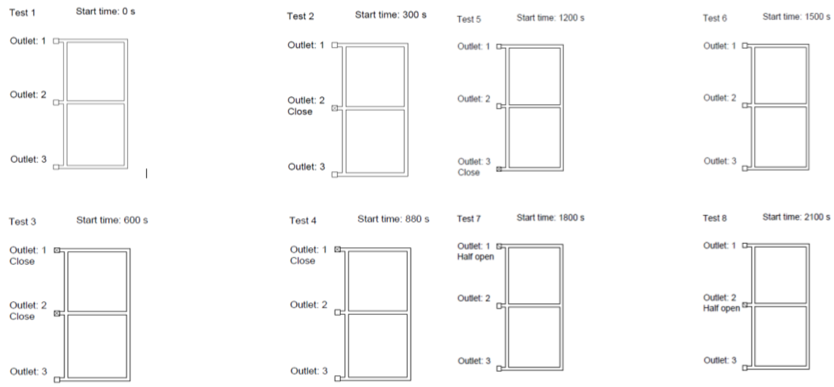



**Figure 4.** System Testing Scenarios Designing

### 2.4.2 Data Organizing

The Observations Data Model (ODM) provides a consistent format for the storage and retrieval of environmental data in a relational database (Horsburgh *et al.* 2008). To organize the flow data, a new database schema was designed, and the (Entity Relationship) ER diagram was created shown in figure 5. After that, the additional tables were established shown in Appendix B, and the flow data from the CSV file was successfully loaded and imported into these tables shown in appendix B-figures, ready for the R-studio environment retrieving.

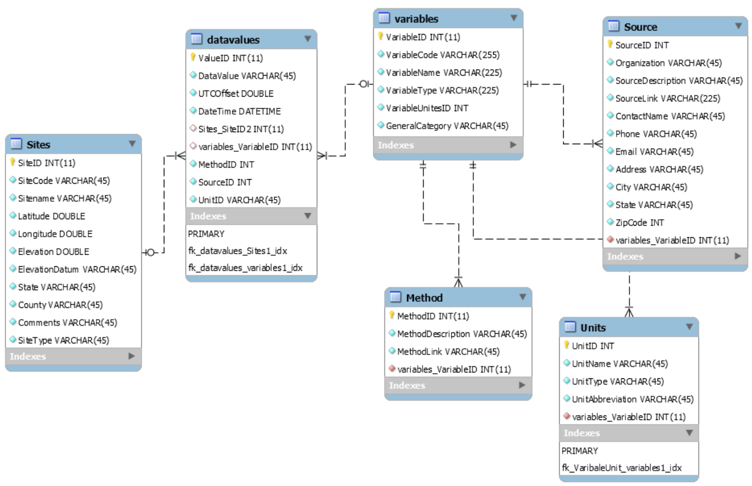



**Figure 5.** Database Design

### 2.4.3 Data Visualizing

Of Data visualization, ggplot2 package in R-studio platform was adopted to retrieve data from the SQL database and make plots for all scenarios. Also, the R-studio has used to directly plot graphs for each classified scenarios like scenarios 1, 2 and 3.

Overall, based on the methods and materials above, the framework of MWTB for the education was organized in figure 6. According to the framework, the process starts with the Arduino programming and then the UNO board setting up. After that, the measurements obtained by YF – S201 flow sensor would be saved in storage card in

UNO board for MySQL database storing and organizing. Finally, the real-time data saved in the database will be extracted by the Rstudio and be visualized in Rstudio environment. All the codes of this framework are open source, and the details can be checked in appendix D.

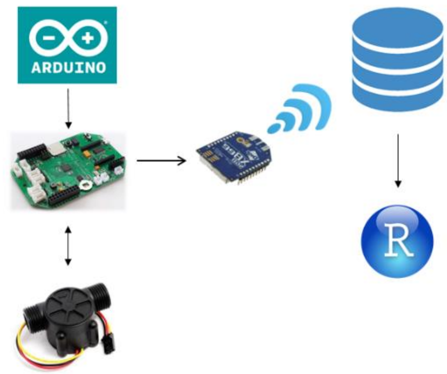



**Figure 6.** Micro-smart Water System Framework

## 3 Results

### 3.1 Data visualization under different Scenarios

By querying the MySQL database, R studio was utilized to make plots for all scenarios shown in figure 7. However, it seems a little ambiguous to analyze the data visualization of the whole scenarios just one time. To solve this issue and to better analyze the graphic results, all the scenarios were categorized by every three scenarios.

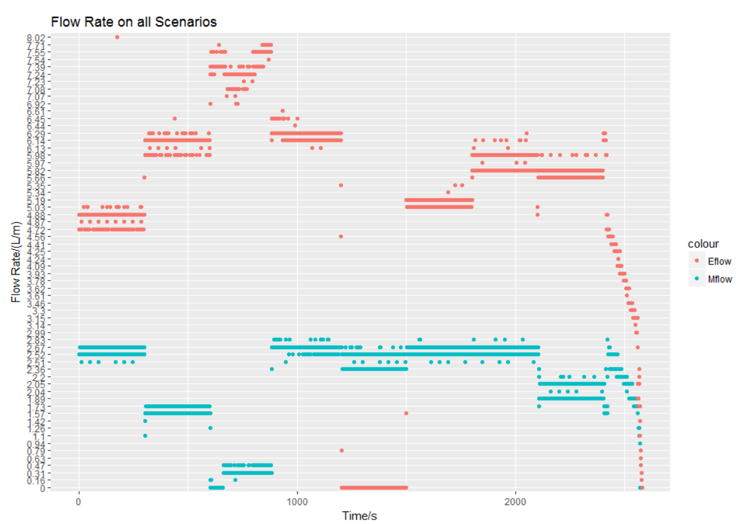



**Figure 7.** Data Visualization for All Scenarios

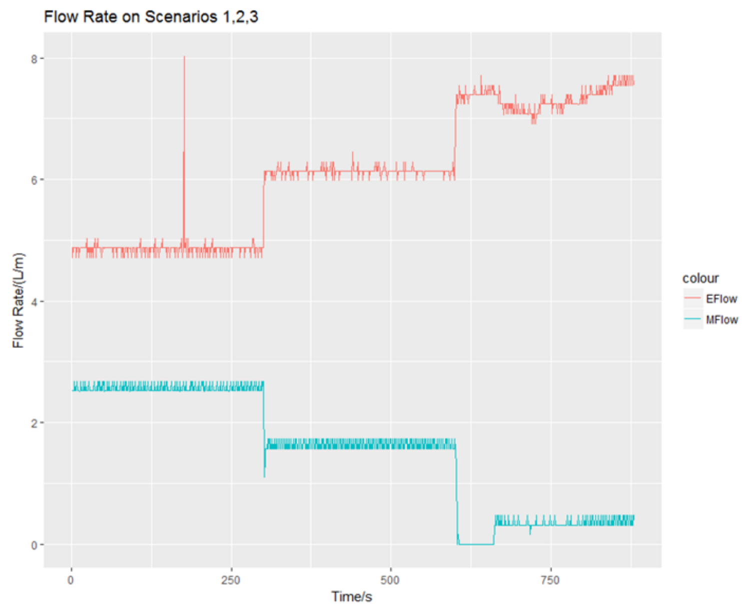



**Figure 8**. Data Visualization under Scenarios 1,2,3

According to the figure 8, from scenario 1 to 2, and 3, the flow rate from End sensor increases step by step, although there are some errors or fluctuation in the EFlow curve. The reason for this is that because of the number of opening outlets declines, which increases the pressure of water discharge in outlet 3. In contrast, the flow rate from Mid sensor decreases scenario by scenario, because once all the outlets were closed one by one, the flow going through the mid pipe would be reduced. Fortunately, the flow rate from mid sensor is more stable than that from end sensor on each scenario.

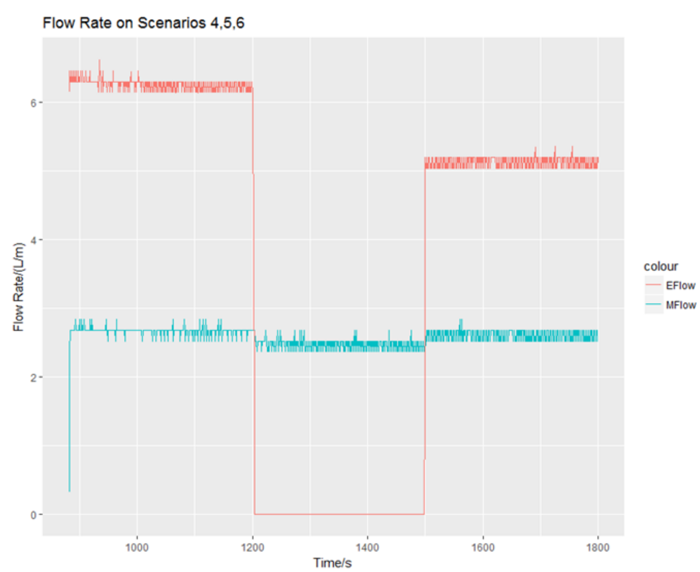



**Figure 9.** Data Visualization under Scenarios 4,5,6

According to the figure 9, from scenario 4 to 5, and 6, the flow rate from the mid sensor can keep almost stable because the outlet 2 is always open. However, the flow rate from the end sensor suddenly goes down to zero in scenario five once the outlet three was closed. The operation of MWTB can be tracked by flow sensors timely. Therefore, the change of flow rate regarding the outlets' status indicates that the flow sensors installed in the micro smart water test bed have reasonable sensitivity.

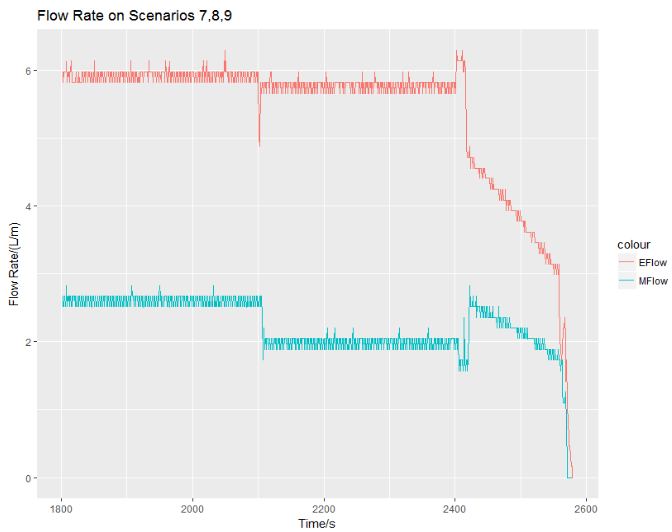



**Figure 10.** Data Visualization under Scenarios 7,8,9

According to the figure 10, from scenario 7 to 8, and 9, when the outlet 1 or outlet 2 is open half, the flow rate from mid sensor shows little drop while the flow rate from end sensor presents a little larger decrease. This phenomenon is because the flow going through mid sensor depends more on the status of outlet 2. But both sensors installed in the system shows sensitive action to the little change of outlets' status. In a word, given the above graphic analysis, the system performance test shows that the flow sensors installed in the Micro-smart water system have reasonable sensitivity to the system operation or status change.

### 3.2 EPNET Model Verification

A hydraulic model was established by considering the structure of MWTB, aiming to evaluate the performance of connecting hydraulic model and experimental set-up. The experimental flow measurements of outlet three was imported into node 7, and then run the model to obtain the flow rate of pipe four which is regarding to the flow rate of the mid sensor, like the examples of figure 11 of scenario 2 and figure 12 of scenario 3. Finally, the newly produced flow rate in pipe four would be compared with the existing experimental measurements from the mid sensor. The modeling outputs can be seen in table 2.

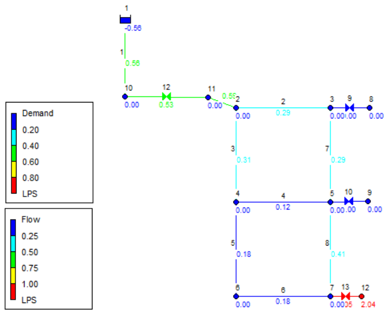


**Figure11.** Scenario 2 modeling output

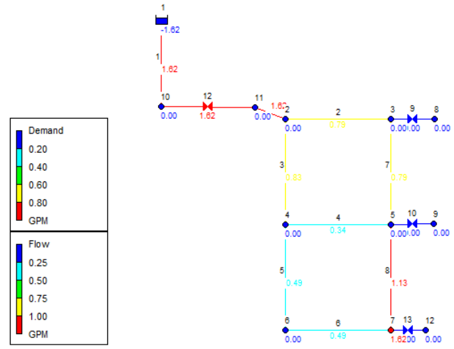


**Figure12.** Scenario 3 modeling output

## 4 Discussion

Summarizing figure 8 to 10, it can be known that the MWTB can produce required data which matches up different scenarios. This work indicates this experimental MWTB is sensitive to water network operation since each operating scenario can be presented accurately in the plots. Such kind of sensitivity can be utilized to demonstrate how SWS work for the public; this is because the engineers can teach people how to infer the inner pipe condition by watching the change of data visualization.

**Table 2.** Model Verification Results

| Scenarios | Pressure | Mid Sensor Measurements(gpm) | Mid outlet Modeling Output(GPM) | Fitting Degree |
|---|---|---|---|---|
| 1 | T1 | 0.66 | 0.27 | 0.590909 |
| 2 | T2 | 0.4147 | 0.34 | 0.18013 |
| 3 | Changing | 0.081 | 0.043 | 0.469136 |
| 4 | T4 | 0.705 | 0.34 | 0.51773 |
| 5 | T5 | 0 | 0 | 0 |
| 6 | T6 | 0.705 | 0.29 | 0.588652 |

Note: Fitting Degree = (Measurement-Modeling Output)/ Measurement

Referring to table 2, the column of “Relative errors” demonstrates that the experimental measurements from mid sensor match a lot the mid outlet modeling output under scenario 2, 3 and 5. Conversely, on scenarios 1 and 6, there is a larger difference between measurements and modeling outputs. Interestingly, all outlets are open on scenarios 1 and six while at least one outlet was closed on other scenarios. It implies that experimental measurements fit modeling outputs reliably and confirms the above result that the hydraulic model can adapt more to change of Micro water test bed.

Overall, the testing result and performance evaluation are showing the MWTB works well. This real-time connection between experimental practice and hydraulic simulation project is performed as expected. Furthermore, the MWTB can be expanded to a large scale with multiple loops in the future when needed, and this model as a basic unit will be applied at somewhere in the city water system.

## 5 Conclusions

In this paper, there are five tasks which are mainly accomplished. 1) Set-up a experimental MWTB in lab; 2) Install and collect flow data by using flow sensor and Arduino sketch; 3)Establish a database to store and organized the experimental measurements; 4) Utilize R to connect MySQL, and visualize flow data from ODM database; 5)Evaluate the performance of MWTB by quantifying fitting degree between measurements and modeling output. One key point is that the data visualization illustrates that MWTB has reasonable sensitivity to the system operation. Such kind of sensitivity has the potential for being used for detecting pipe leakage in the future. Additionally, the experimental measurements fitting the modeling output results show MWTB has a reliable performance analysis under different scenarios. This analysis verified that MWTB could be used an education practice to show how SWS work for flow monitoring and also leakage detection. One future work will be focusing on expanding the scale of MWTB with multiple loops to verify the network performance. Another future direction will be demonstrating a broader application of smart water techniques in water resources field such as pipe leakage detection, stream flow monitoring or system operation efficiency improvement.

## References


Abu-Mahfouz, A. M., Steyn, L. P., Isaac, S. J., & Hancke, G. P. (2012, January). The multi-level infrastructure of interconnected testbeds of large-scale wireless sensor networks (MI2T-WSN). In Proceedings of the International Conference on Wireless Networks (ICWN) (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Arduino,(2018): https://www.arduino.cc/en/guide/introduction

Bartos, M., Wong, B., and Kerkez, B., 2018. Open storm: a complete framework for sensing and control of urban watersheds. *Environmental Science: Water Research & Technology*, *4*(3), pp.346-358.

Boulos, P.F., 2017. Smart water network modeling for sustainable and resilient infrastructure. *Water Resources Management*, *31*(10), pp.3177-3188.

Dludla, A.G., Abu-Mahfouz, A.M., Kruger, C.P., and Isaac, J.S., 2013, May. Wireless sensor networks testbed: ASNTbed. In *IST-Africa Conference and Exhibition (IST-Africa), 2013* (pp. 1-10). IEEE.

ePro Labs. Sensor. 2016: https://wiki.eprolabs.com/index.php?title=Flow_Sensor_YF-S201. (accessed 19 Jan 2019)

Günther, M., Camhy, D., Steffelbauer, D., Neumayer, M. and Fuchs-Hanusch, D., 2015. Showcasing a smart water network based on an experimental water distribution system. *Procedia Engineering*, *119*, pp.450-457.

Günther, M., Steffelbauer, D., Neumayer, M. and Fuchs-Hanusch, D., 2014. Experimental setup to examine leakage outflow in a scaled water distribution network. *Procedia engineering*, *89*, pp.311-317.

Hatchett, S., Boccelli, D., Uber, J., Haxton, T., Janke, R., Kramer, A., Matracia, A. and Panguluri, S., 2010. How accurate is a hydraulic model?. In *Water Distribution Systems Analysis 2010* (pp. 1379-1389).

Horsburgh, J.S., Tarboton, D.G., Maidment, D.R. and Zaslavsky, I., 2008. A relational model for environmental and water resources data. *Water Resources Research*, *44*(5).

Karray, F., Garcia-Ortiz, A., Jmal, M.W., Obeid, A.M. and Abid, M., 2016. Earnpipe: A testbed for smart water pipeline monitoring using wireless sensor network. Procedia Computer Science, 96, pp.285-294.

Kartakis, Sokratis, Edo Abraham, and Julie A. McCann. "Waterbox: A testbed for monitoring and controlling smart water networks." In *Proceedings of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks*, p. 8. ACM, 2015.

Kramer, M., 2014. Enhancing sustainable communities with green infrastructure. *Office of Sustainable Communities, US Environmental Protection Agency: Washington, DC, USA*, p.66.

Machell, J., Mounce, S.R., and Boxall, J.B., 2010. Online modeling of water distribution systems: a UK case study. *Drinking Water Engineering and Science*, *3*, pp.21-27.

Rossman, L.A., 2000. EPANET 2: users manual.

Sonaje, N.P. and Joshi, M.G., 2015. A review of modeling and application of water distribution networks (WDN) software. *Int. J. of Tech. Research and Appl*, *3*(5), pp.174-178.

Steffelbauer, D., Neumayer, M., Günther, M. and Fuchs-Hanusch, D., 2014. Sensor placement and leakage localization considering demand uncertainties. *Procedia engineering*, *89*, pp.1160-1167.

Walski, T., Blakley, D., Evans, M. and Whitman, B., 2017. Verifying Pressure Dependent Demand Modeling. *Procedia Engineering*, *186*, pp.364-371.

**Appendix A: Acronyms**

Micro-smart water test bed (MWTB);

Observation Data Model (ODM);

Smart Water System(SWS), Pressure-dependent Demands(PDD);

Information Communication Technology (ICT).

**Appendix B: Design Drawings**

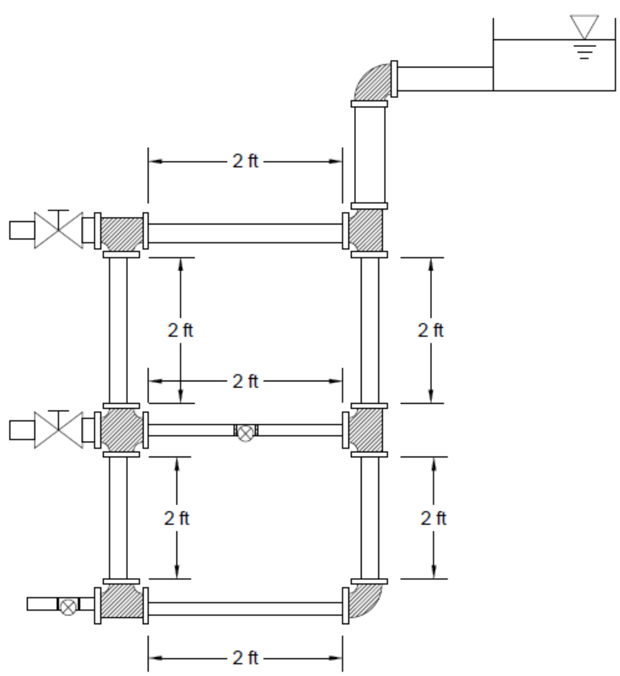



**Figure 1.** Water system with pipe length

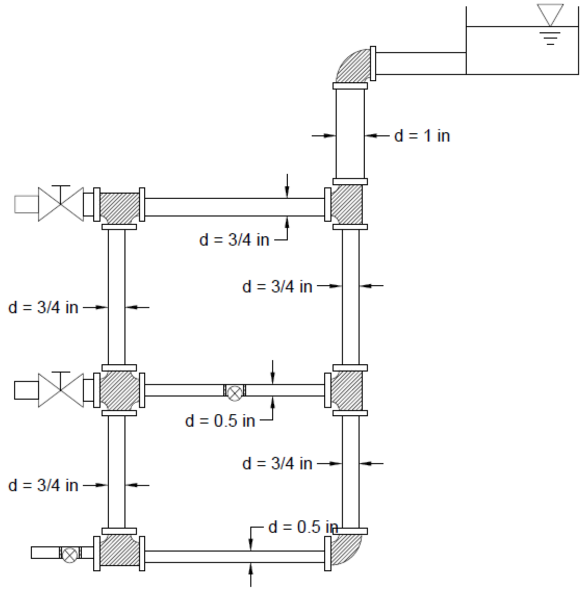
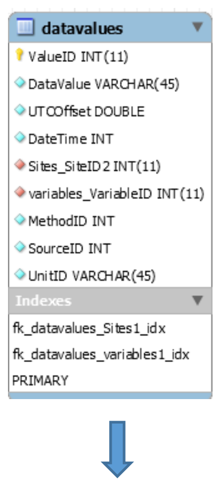
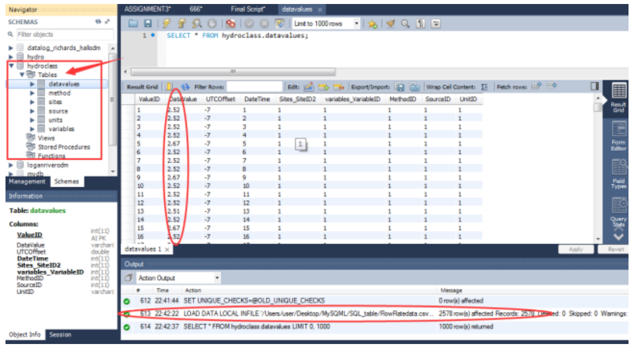


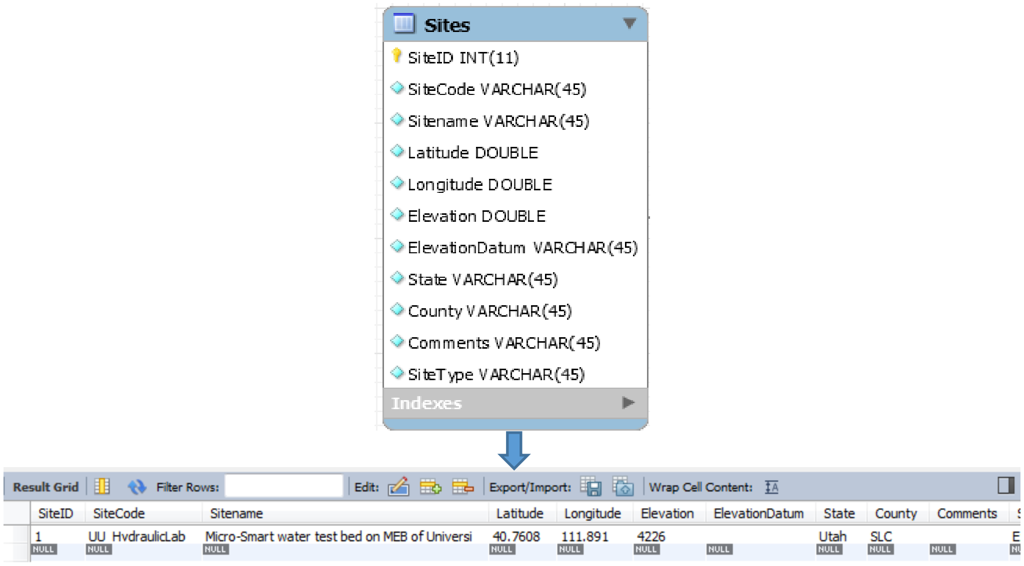

**Figure 2.** Water system with pipe diameter

## Appendix B: Database Creation Figures

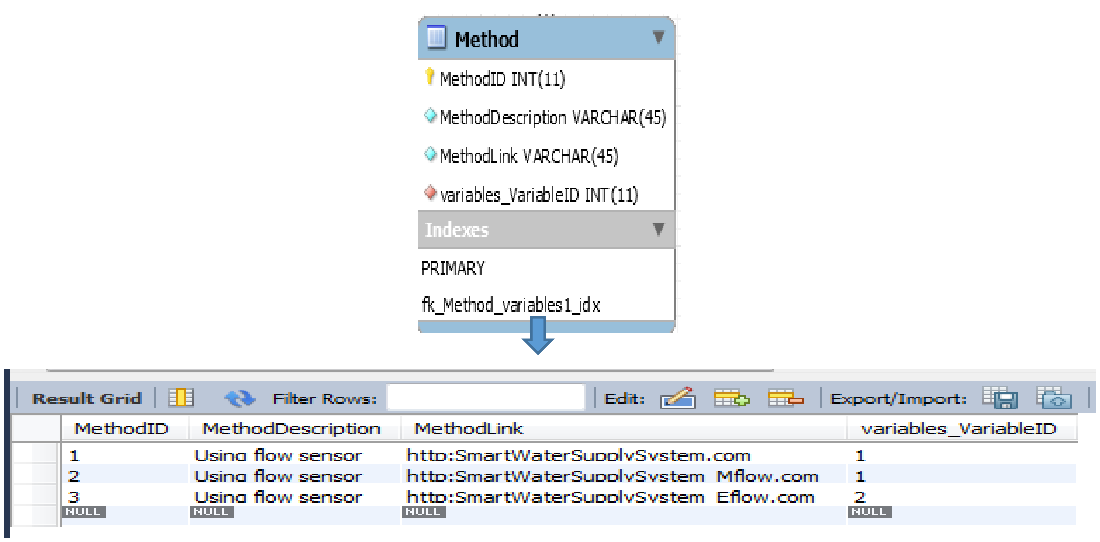




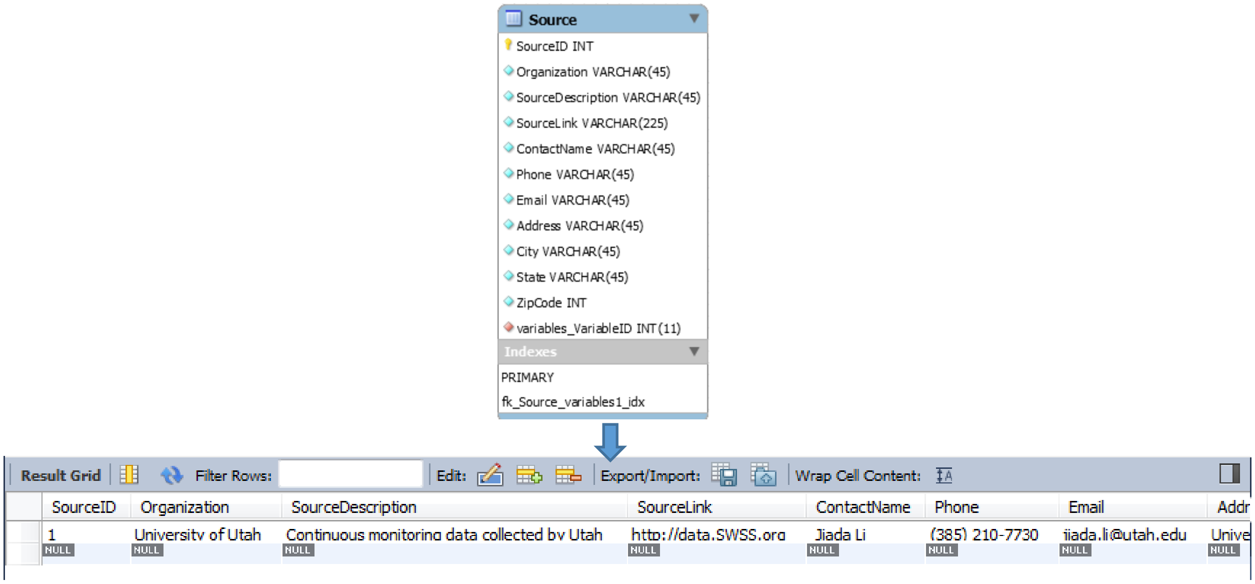
**(a) Datavalues**



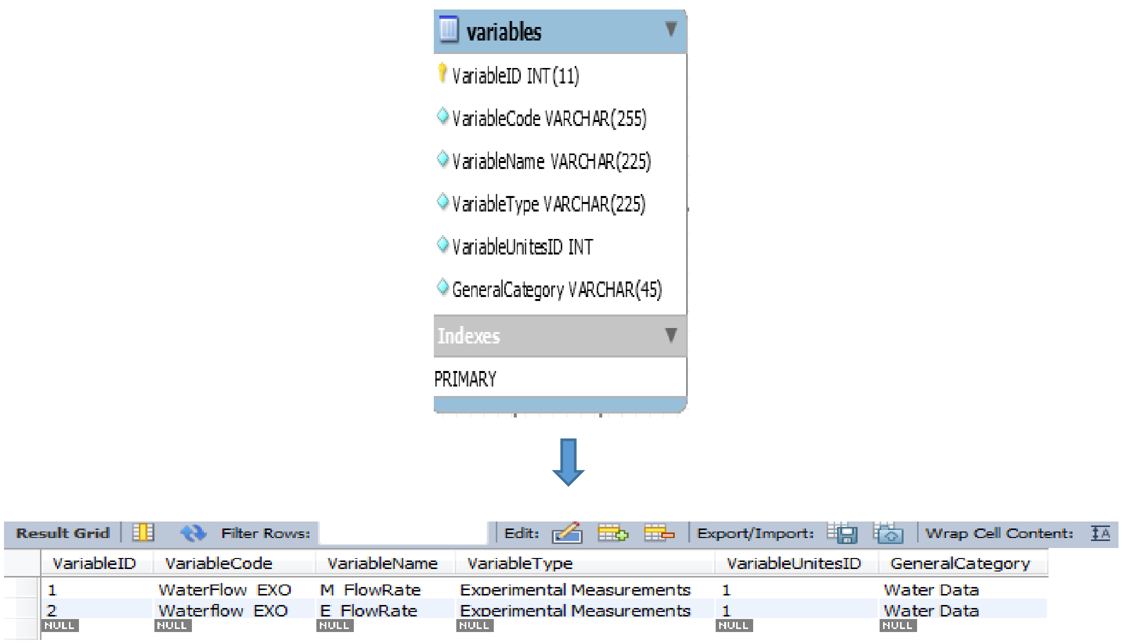

**(b) Sites**



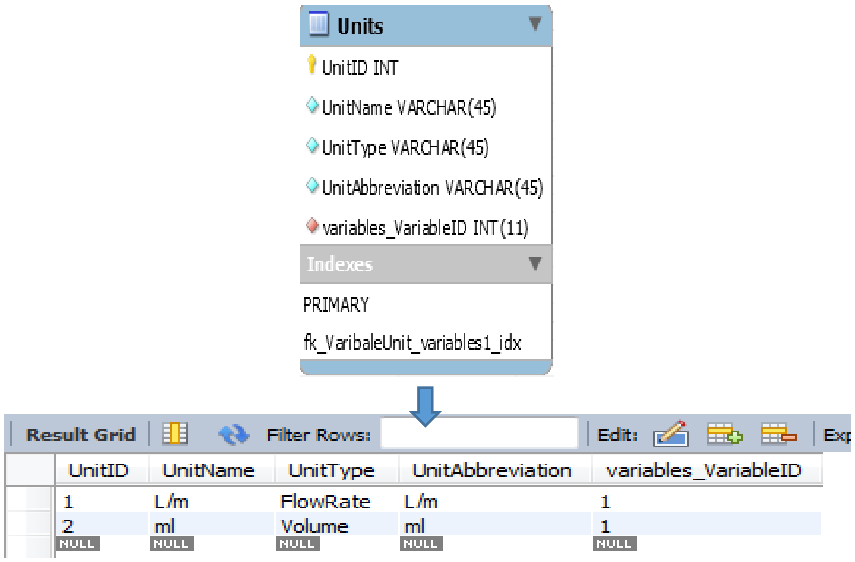

**(c) Methods**




**(d) Source**




**(e) Variables**




| UnitID | UnitName | UnitType | UnitAbbreviation | variables_VariableID |
|---|---|---|---|---|
| 1 | L/m | FlowRate | L/m | 1 |
| 2 | ml | Volume | ml | 1 |
| NULL | NULL | NULL | NULL | NULL |

**(f) Units**

**Figure1 (a,b,c,d,e,f).** Data Importing

## Appendix C: Smart Components Figures

**Figure 1** YF – S201 flow sensor


**Figure 2** Arduino Board

## Appendix D: Codes

**Arduino Codes** (128 lines)

```
/* Test version 2, with volume measurement
*/
byte sensorInterrupt = 0;  // 0 = digital pin 2
// The hall-effect flow sensor outputs approximately 4.5 pulses per second per
// litre/minute of flow.
float calibrationFactor = 6.35;

volatile byte pulseCount;
float flowRate;
unsigned int flowMilliLitres;
unsigned long totalMilliLitres;
unsigned long countTime;

// Include the SD library
#include <SD.h>
// Initialize some variables for use in my main loop
int recordNum = 1;
float someValue = 1.0;
void setup()
{
  // Initialize a serial connection for reporting values to the host
```

```
  Serial.begin(38400);
  Serial.print("Time (s),Flow rate (L/m), Current Flow (ml/s), Water Volum (ml)");
  pulseCount        = 0;
  flowRate          = 0.0;
  flowMilliLitres   = 0;
  totalMilliLitres  = 0;
  countTime         = 0;

  // The Hall-effect sensor is connected to pin 2 which uses interrupt 0.
  // Configured to trigger on a FALLING state change (transition from HIGH
  // state to LOW state)
  attachInterrupt(sensorInterrupt, pulseCounter, FALLING);

  // Initialize the SD Card
  pinMode(10, OUTPUT); // Set the pinMode on digital pin 10 to OUTPUT
  // send a message to the serial port and exit the program
  if (!SD.begin(10)){
    Serial.println("SD card initialization failed!");
    return;
  }

  // Initialization of the SD card was successful, and open a file
  Serial.println("SD card initialization done.");
  File myFile = SD.open("test.txt", FILE_WRITE);
  // Print a header line to the file with column names
  myFile.println("Recording Time, Flow rate (L/m), Flow Rate (ml/s), volume (ml)");
  // Close the file
  myFile.close();
}

/**
 * Main program loop
 */
void loop()
{
```

```
   if((millis() - countTime) > 1000)    // Only process counters once per second
  {
    detachInterrupt(sensorInterrupt);
    flowRate = ((1000.0 / (millis() - countTime)) * pulseCount) / calibrationFactor;
    countTime = millis();
    flowMilliLitres = (flowRate / 60) * 1000;
    // The millilitres passed in this second to the cumulative total
    totalMilliLitres += flowMilliLitres
    unsigned int frac;

/* //Print the flow rate for this second in litres / minute
    Serial.print(countTime); // print time
    Serial.print(int(flowRate));  // Print the integer part of the variable
    Serial.print(".");             // Print the decimal point
    //Determine the fractional part. The 10 multiplier gives us 1 decimal place.
    frac = (flowRate - int(flowRate)) * 10;
    Serial.print(frac, DEC) ;      // Print the fractional part of the variable
    Serial.print("L/min,");
    // Print the number of litres flowed in this secon
    Serial.print(flowMilliLitres);
    Serial.print("mL/S,");
    // Print the cumulative total of litres flowed since starting
    Serial.print(totalMilliLitres);
    Serial.println("mL");
*/
    // Reset the pulse counter so we can start incrementing again
    pulseCount = 0;

    // Enable the interrupt again now that we've finished sending output
    attachInterrupt(sensorInterrupt, pulseCounter, FALLING);
    // Open the file
  File myFile = SD.open("test.txt", FILE_WRITE);
  String dataRecord = String(countTime/1000) + ", " +
String(flowRate)+","+String(flowMilliLitres)+","+String(totalMilliLitres);
```

```
  myFile.println(dataRecord);
  Serial.println(dataRecord);

  // Close the file
  myFile.close();
  }
}

/*
Insterrupt Service Routine
 */
void pulseCounter()
{
  // Increment the pulse counter
  pulseCount++;
}
```

**R codes** (70 lines)

**1) SQL Retrieving and Plotting All Scenarios**

```
#Load RMySQL package
library(RMySQL)
d <- dbDriver("MySQL")
#Conect R with MySQL
con <- dbConnect(d,user='root',password='kd542507',host='localhost')
#Ask the permission for using hydroclass odm
sqlstmtdb <- dbSendQuery(con,"Use hydroclass;")
#query SQL temp data from Loganriver near Tony Grove
sqlstmt1 <- dbSendQuery(con, "SELECT DateTime, DataValue FROM DataValues
            WHERE
            Sites_SiteID2 = 1 AND
            variables_VariableID =1 AND DataValue <> -9999
            ORDER BY DateTime" )
#Use dbFetch functions to execute SQL queries and return results
Mflow <- dbFetch(sqlstmt1, n=-1)
```

```
names(Mflow) <- c("time","Mflow")
sqlstmt2 <- dbSendQuery(con, "SELECT DateTime, DataValue FROM DataValues
                    WHERE
                    Sites_SiteID2 = 1 AND
                    variables_VariableID = 2 AND DataValue <> -9999
                    ORDER BY DateTime" )
Eflow <- dbFetch(sqlstmt2, n=-1)

names(Eflow) <- c("time","Eflow")
Series <- merge(Mflow,Eflow,by="time")
library(ggplot2)
m <- ggplot(data =
Series)+geom_point(aes(x=time,y=Mflow,color="Mflow"))+geom_point(aes(x=time,y=Eflow,color="Eflow"))
m <- m+labs(title="Flow Rate on all Scenarios ", x="Time/s", y="Flow Rate/(L/m)")
print(m)
```

**2) Plotting for Classified Scenarios**

```
#Scenarios 1,2.3
C1 <- read.csv('C:/Users/user/Desktop/combination1.csv')
names(C1) <- c("Time","MFlow","Mflow","MVolume","EFlow","Eflow","EVolume")
library(ggplot2)
l <- ggplot(data =
C1)+geom_line(aes(x=Time,y=MFlow,color="MFlow"))+geom_line(aes(x=Time,y=EFlow,color="EFlow"))
l <- l+labs(title="Flow Rate on Scenarios 1,2,3", x="Time/s", y="Flow Rate/(L/m)")
print(l)

#Scenarios 4,5,6
C2 <- read.csv('C:/Users/user/Desktop/combination2.csv')
names(C2) <- c("Time","MFlow","Mflow","MVolume","EFlow","Eflow","EVolume")
library(ggplot2)
m <- ggplot(data =
C2)+geom_line(aes(x=Time,y=MFlow,color="MFlow"))+geom_line(aes(x=Time,y=EFlow,color="EFlow"))
m <- m+labs(title="Flow Rate on Scenarios 4,5,6", x="Time/s", y="Flow Rate/(L/m)")
print(m)

#Scenarios 7,8,9
C3 <- read.csv('C:/Users/user/Desktop/combination3.csv')
```

```
names(C3) <- c("Time","MFlow","Mflow","MVolume","EFlow","Eflow","EVolume")
library(ggplot2)
n <- ggplot(data C3)+geom_line(aes(x=Time,y=MFlow,color="MFlow"))+geom_line(aes(x=Time,y=EFlow,color="EFlow"))
n <- n+labs(title="Flow Rate on Scenarios 7,8,9", x="Time/s", y="Flow Rate/(L/m)")
print(n)
```

**Database Codes** (200 lines)

**1) ER diagram codes:**

```
-- MySQL Script generated by MySQL Workbench
-- Thu Dec  7 12:00:30 2017
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema hydroclass
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `hydroclass` DEFAULT CHARACTER SET utf8 ;
USE `hydroclass` ;

-- -----------------------------------------------------
-- Table `hydroclass`.`Sites`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `hydroclass`.`Sites` (
  `SiteID` INT(11) NOT NULL AUTO_INCREMENT,
  `SiteCode` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
  `Sitename` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
  `Latitude` DOUBLE NOT NULL,
  `Longitude` DOUBLE NOT NULL,
  `Elevation` DOUBLE NOT NULL,
```

```
  `ElevationDatum` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
  `State` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
  `County` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
  `Comments` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
  `SiteType` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
  PRIMARY KEY (`SiteID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_unicode_ci;

-- -----------------------------------------------------
-- Table `hydroclass`.`variables`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `hydroclass`.`variables` (
 `VariableID` INT(11) NOT NULL AUTO_INCREMENT,
`VariableCode` VARCHAR(255) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
`VariableName` VARCHAR(225) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
`VariableType` VARCHAR(225) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
`VariableUnitesID` INT NOT NULL,
`GeneralCategory` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
PRIMARY KEY (`VariableID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_unicode_ci;

-- -----------------------------------------------------
-- Table `hydroclass`.`datavalues`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `hydroclass`.`datavalues` (
 `ValueID` INT(11) NOT NULL AUTO_INCREMENT,
 `DataValue` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
`UTCOffset` DOUBLE NOT NULL,
`DateTime` INT NOT NULL,
```

```
`Sites_SiteID2` INT(11) NOT NULL,
`variables_VariableID` INT(11) NOT NULL,
 `MethodID` INT NOT NULL,
`SourceID` INT NOT NULL,
`UnitID` VARCHAR(45) NOT NULL,
INDEX `fk_datavalues_Sites1_idx` (`Sites_SiteID2` ASC),
INDEX `fk_datavalues_variables1_idx` (`variables_VariableID` ASC),
PRIMARY KEY (`ValueID`),
CONSTRAINT `fk_datavalues_Sites1`
FOREIGN KEY (`Sites_SiteID2`)
REFERENCES `hydroclass`.`Sites` (`SiteID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_datavalues_variables1`
 FOREIGN KEY (`variables_VariableID`)
REFERENCES `hydroclass`.`variables` (`VariableID`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_unicode_ci;


-- -----------------------------------------------------
-- Table `hydroclass`.`Method`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `hydroclass`.`Method` (
  `MethodID` INT(11) NOT NULL AUTO_INCREMENT,
  `MethodDescription` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
  `MethodLink` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8_unicode_ci' NOT NULL,
  `variables_VariableID` INT(11) NOT NULL,
  PRIMARY KEY (`MethodID`),
  INDEX `fk_Method_variables1_idx` (`variables_VariableID` ASC),
  CONSTRAINT `fk_Method_variables1`
```

```
    FOREIGN KEY (`variables_VariableID`)
    REFERENCES `hydroclass`.`variables` (`VariableID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_unicode_ci;

-- -----------------------------------------------------
-- Table `hydroclass`.`Units`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `hydroclass`.`Units` (
  `UnitID` INT NOT NULL AUTO_INCREMENT,
  `UnitName` VARCHAR(45) NOT NULL,
  `UnitType` VARCHAR(45) NOT NULL,
  `UnitAbbreviation` VARCHAR(45) NOT NULL,
  `variables_VariableID` INT(11) NOT NULL,
  PRIMARY KEY (`UnitID`),
  INDEX `fk_VaribaleUnit_variables1_idx` (`variables_VariableID` ASC),
  CONSTRAINT `fk_VaribaleUnit_variables1`
    FOREIGN KEY (`variables_VariableID`)
    REFERENCES `hydroclass`.`variables` (`VariableID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `hydroclass`.`Source`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `hydroclass`.`Source` ;

CREATE TABLE IF NOT EXISTS `hydroclass`.`Source` (
  `SourceID` INT NOT NULL AUTO_INCREMENT,
  `Organization` VARCHAR(45) NOT NULL,
```

```
  `SourceDescription` VARCHAR(45) NOT NULL,
  `SourceLink` VARCHAR(225) NOT NULL,
  `ContactName` VARCHAR(45) NOT NULL,
  `Phone` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `Address` VARCHAR(45) NOT NULL,
  `City` VARCHAR(45) NOT NULL,
  `State` VARCHAR(45) NOT NULL,
  `ZipCode` INT NOT NULL,
  `variables_VariableID` INT(11) NOT NULL,
  PRIMARY KEY (`SourceID`),
  INDEX `fk_Source_variables1_idx` (`variables_VariableID` ASC),
  CONSTRAINT `fk_Source_variables1`
   FOREIGN KEY (`variables_VariableID`)
   REFERENCES `hydroclass`.`variables` (`VariableID`)
 ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

INSERT INTO `units` (`UnitID`,`UnitName`,`UnitType`,`UnitAbbreviation`,`variables_VariableID`) VALUES
(1,'L/m','FlowRate','L/m',1);
INSERT INTO `units` (`UnitID`,`UnitName`,`UnitType`,`UnitAbbreviation`,`variables_VariableID`) VALUES
(2,'ml','Volume','ml',1);

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

**2) Data Loading codes**

```
USE hydroclass;

#load sites
LOAD DATA LOCAL INFILE '/Users/user/Desktop/MySQML/SQL_table/sites.csv'
INTO TABLE sites
```

```
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(SiteCode,SiteName,Latitude,Longitude,Elevation,ElevationDatum,State,County,Comments,SiteType);

#load variables
LOAD DATA LOCAL INFILE '/Users/user/Desktop/MySQML/SQL_table/variables.csv'
INTO TABLE variables
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(VariableCode,VariableName,VariableType,VariableUnitesID,GeneralCategory);

#load methods
LOAD DATA LOCAL INFILE '/Users/user/Desktop/MySQML/SQL_table/methods.csv'
INTO TABLE method
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(MethodDescription,MethodLink,variables_VariableID);

#load sources
LOAD DATA LOCAL INFILE '/Users/user/Desktop/MySQML/SQL_table/sources.csv'
INTO TABLE source
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(Organization,SourceDescription,SourceLink,ContactName,Phone,Email,Address,City,State,ZipCode);

#load datavalues-Mflow
LOAD DATA LOCAL INFILE '/Users/user/Desktop/MySQML/SQL_table/MFlowRatedata.csv'
```

```
INTO TABLE datavalues
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(DataValue,UTCOffset,DateTime,Sites_SiteID2,variables_VariableID,MethodID,SourceID,UnitID);

#load datavalue-Eflow
LOAD DATA LOCAL INFILE '/Users/user/Desktop/MySQML/SQL_table/EFlowRatedata.csv'
INTO TABLE datavalues
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(DataValue,UTCOffset,DateTime,Sites_SiteID2,variables_VariableID,MethodID,SourceID,UnitID);
```