



UNIVERSITY OF PERADENIYA
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

PROJECT PLAN DOCUMENT

FLOOD DETECTION AND SAFETY PREDICTION SYSTEM

THROUGH WATER LEVEL AND VELOCITY MONITORING

GROUP 09

E/14/080

E/14/228

E/14/240

TABLE OF CONTENTS

	Page No
Table of Figures	4
Abstract	5
Related Work	6
Background Research	7
Project Overview	8
Technical Note	
Hardware Design	9
Power supply	10
Circuit diagram	11
Ultra Sonic Sensor	12-13
Flow Rate Sensor	14-15
Arduino UNO	16
Network Design	17
Technologies Used	18
Technologies used in web and app mobile app	19
APIs Used	20-21
Interface Design	22
Database Design	23
Network Security	24
Budget	25
Timeline	26

Testing	27
Hardware Testing	27-32
Network Testing	33-38
Security Testing	39-40
User Manual	41
Web Interface	41-47
Mobile Application	48-50
Reference for further Implementation	51

TABLE OF FIGURES

	Page No
Figure 1:Project Overview	6
Figure 2:Hardware Design	7
Figure 3:Power Supply	8
Figure 4: Circuit Diagram	9
Figure 5: Working of an ultra sonic sensor	11
Figure 6: HC SR04 connected to an arduino UNO	11
Figure 7: Flow Rate Sensor	13
Figure 8: Layout of an A7 GSM module	14
Figure 9: Network Overview	15
Figure 10: Multiple nodes connecting to the central server	15
Figure 11 :MVC architecture of laravel framework	16
Figure 12: Reasons for choosing ionic	17
Figure 13: Interface Designs	20
Figure 14: Database Design	21
Figure 15: Theory behind MAC protocol	22
Figure 16: Apparatus to test the ultra sonic sensor	27
Figure 17 : Exepected value vs practical values	29
Figure 18: Error with regard to height measurement	29
Figure 19: Exepected value vs practical values	32
Figure 20: Error with regard to flow rate sensor	32
Figure 21: Time taken to process requests	35
Figure 22:Total round trip time	37
Table 1:Budget	23
Table 2:Timeline	24

Abstract

This system would monitor the status of rivers in country in real time and would help to prevent the dangers associated with water flow of river bodies of the country. This system is able to detect a change in the usual water level and speed and would therefore is able to give notifications about the safety regarding that particular water body. The system has a sensing circuit connected to a microcontroller which sense and outputs the current water level and velocity of the surface in which the signals are transmitted to a main server where the data is analyzed by comparing with previous data. The current details regarding the water bodies could is accessible to the public through a web interface and a mobile application.

Related work

Background work:

http://www.irrigation.gov.lk/index.php?option=com_riverdata&Itemid=266&lang=en

Currently, this site gives an update of the river status of Sri Lanka. However this system is based on manually collected data and is updated only once per day. By constantly keeping track of this site we found that there are certain times that the system doesn't output any data at all. This system is sufficient to get rough details about the water level of a river body but this won't clearly serve any help at detecting a risky situation prior to the occurring of an event.

Since our system is based on data collected through sensors, human force used in collecting this data manually, informing the collected data to the central office, updating these data and keeping track of them would be minimized to a larger extent through our system. Our system would be more secure, efficient and less erroneous compared with the existing manual procedure and system.

Background Research

The flood conditions can be monitored through the water level of a river and the threshold values vary from one river to river. These threshold values should be obtained from former experiences and pre-calculated data.

The drowning conditions may differ with the velocity of flowing water. Although they too differ with many other factors water velocity alone would be a good measure as an alerting signal to prevent any drowning situation. An important factor to remember is that most fatalities occur in moderate water levels and the water velocities may not seem fatal as they actually are. Therefore concluding whether a water way is safe enough or not merely by eye sight may cause destructive results. Instead a system which monitor both water level and the velocity would be able to predict the dangers and safety measures more effectively.

In waist deep water it takes roughly 2-2.5 feet per second to push a grown man over while in chest deep water it would only take 1-1.5 feet per second. According to researchers about 71% of the fatalities occur in water ways that are between 3.4 feet to 5.2feet. There's a model called drowning trap in which it states that the depth of the water, velocity and the deceptiveness of the velocity are the three major reasons that are responsible for drowning.

Project Overview

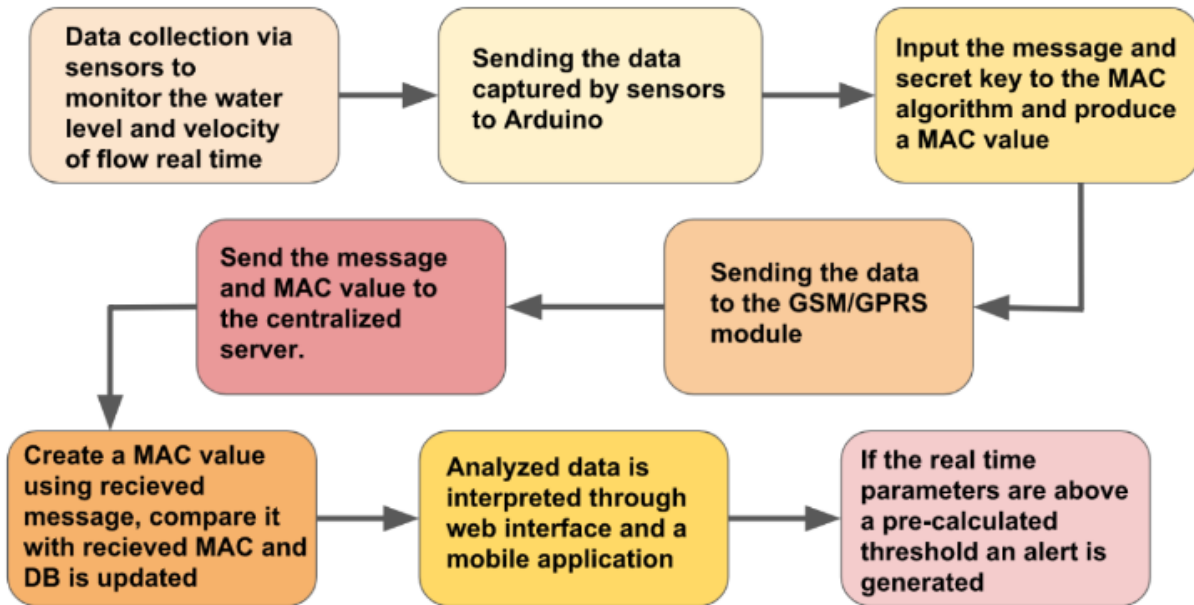


Figure1::Project Overview

Hardware Design

The hardware part containing the microcontroller, gsm module and the batteries are embedded in a box where it's top is enclosed by the solar panels. The ultra sonic sensor is attached to the bottom of the box. This part should be fixed to a bridge above the water surface. A tube extrudes from this part until it reaches below the surface of water. The flow rate sensor is attached to the bottom end of the tube in which it is connected to the arduino board. To get better results the box should be fixed about 1m above from the surface of the water.



Figure 2:Hardware Design

Power Supply

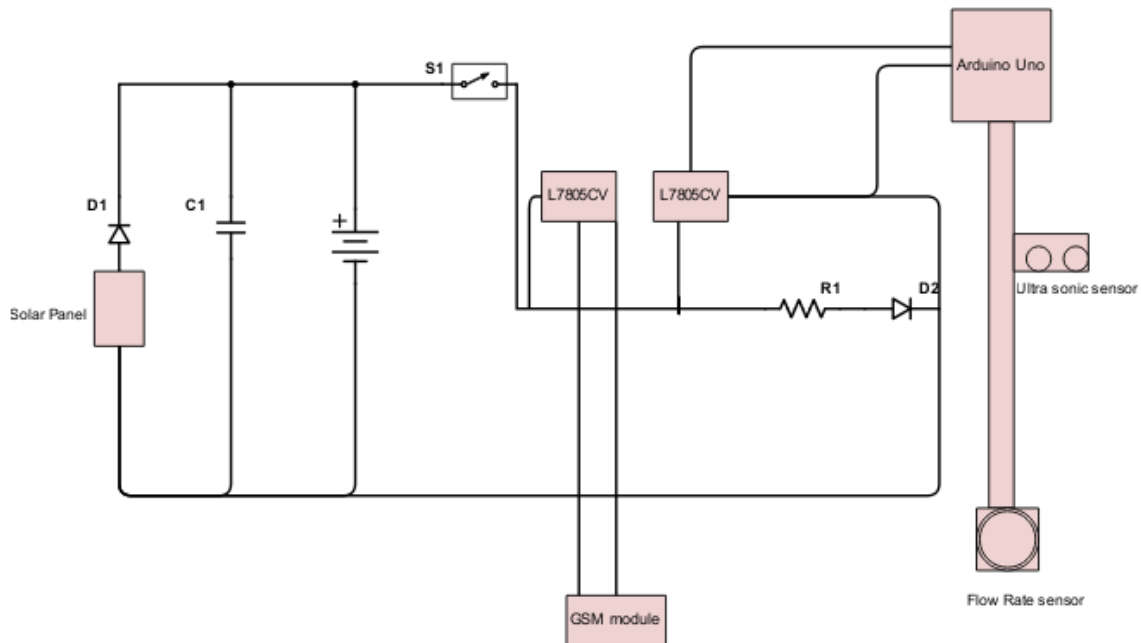


Figure 3:Power Supply

Power supplied by the solar panels=3 x 4.5 =13.5V

Power supplied by the batteries= 3 x 3.7V= 11.1 V

2 L7805CV Linear Voltage Regulators = 10V

R1 = 1k Ω

Capacitor =2200 μ F

Circuit Diagram

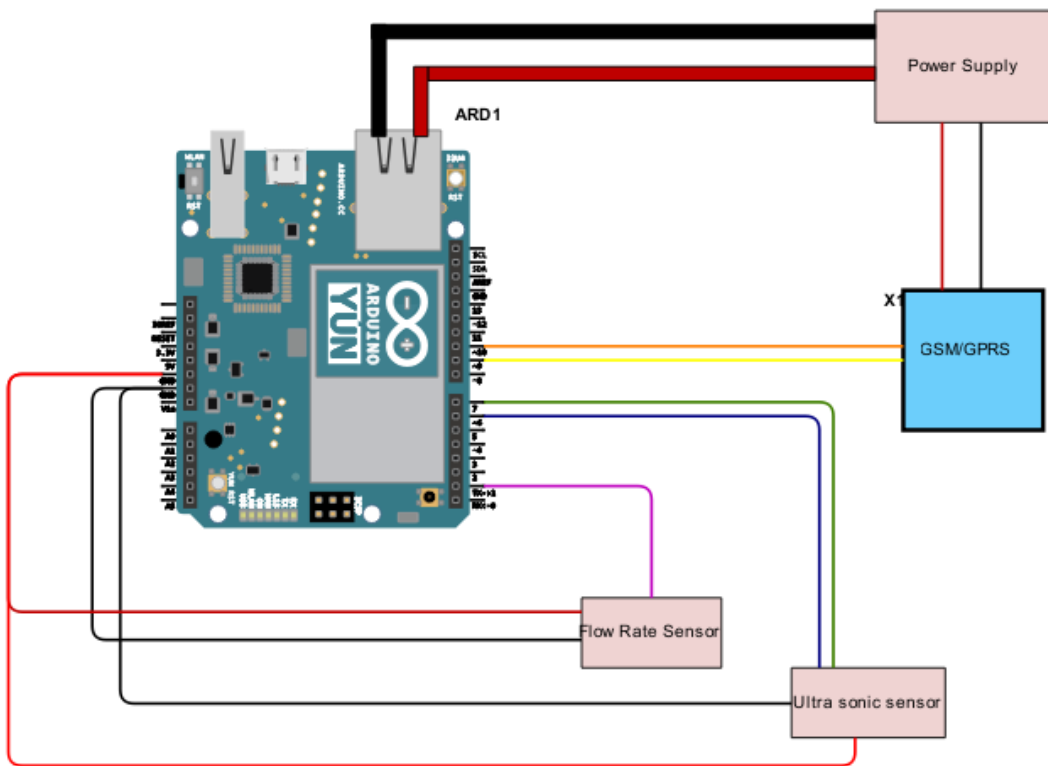


Figure 4: Circuit Diagram

Ultra sonic sensor(HC SR04)

To measure the water level we are using ultra sonic sensors which should be implemented above the water body and the sensor would output the distance from that given body to the surface of the water. Through this we could check whether there's a significant change in the water level by comparing this data with the pre-collected data. There were other sensing methods to get the water level but we chose ultra-sonic sensors as the best due to its accuracy when compared with other methods and its easy-maintainability since the components do not get in touch with the water (no rusting, less depreciation).

limitations

- The height from the surface to the fixed point can be obtained only with an accuracy of ± 1 cm. But since we do not need the minute changes of the water level and measure only the drastic changes, this error could be tolerated. With time, there might be a possibility to have a moisture layer on the face of the sensors and that'd change the density between the surface and might lead to erroneous results.
- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm/1" – 13ft
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS

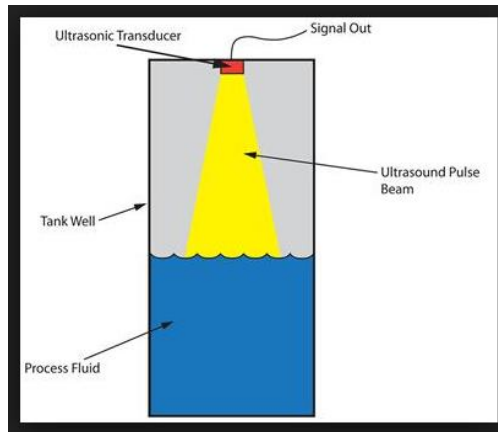


Figure 5: Working of an ultra sonic sensor

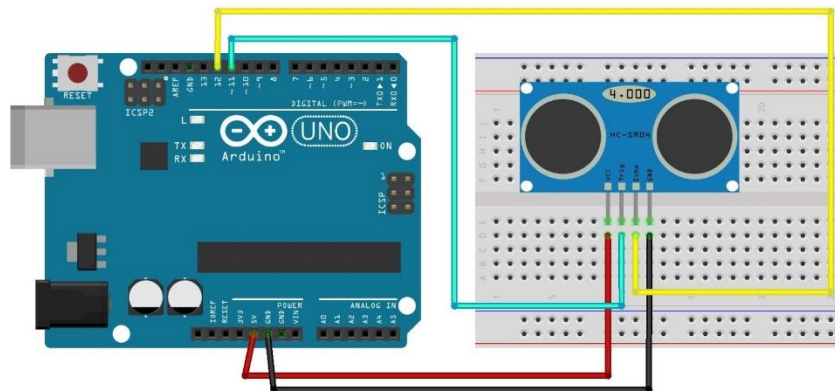


Figure 6: HC SR04 connected to an arduino UNO

Flow Rate Sensor

The flow rate sensor has to be emerged somewhere below the surface (according to the principles of hydro physics as at this point a more accurate measurement regarding flow rate of a river could be obtained) The flow meter works on the principle of the Hall effect. According to the Hall effect, a voltage difference is induced in a conductor transverse to the electric current and the magnetic field perpendicular to it. Here, the Hall effect is utilized in the flow meter using a small fan/propeller-shaped rotor, which is placed in the path of the liquid flowing. The liquid pushes against the fins of the rotor, causing it to rotate. The shaft of the rotor is connected to a Hall effect sensor. It is an arrangement of a current flowing coil and a magnet connected to the shaft of the rotor, thus a voltage/pulse is induced as this rotor rotates. In this flow meter, for every liter of liquid passing through it per minute, it outputs about 4.5 pulses. This is due to the changing magnetic field caused by the magnet attached to the rotor shaft as seen in the picture below. We measure the number of pulses using a micro-controller.

Limitations

- Since the flow rate is embedded within a tube, the friction and the cohesive forces exerted by the walls of the tube would hinder obtaining the exact flow rate at the point. But since the velocity of a flowing water body is not uniform, there's no point in trying to get the exact velocity. We can position the flow rate sensor in a place the maximum velocity could be measured (somewhere just beneath the surface) and that would be effective than getting the exact value since this measurement is going to be an average estimation of the velocity.

- Working Voltage: 5 to 18V DC (min tested working voltage 4.5V)
- Max current draw: 15mA @ 5V
- Output Type: 5V TTL
- Working Flow Rate: 1 to 30 Liters/Minute



Figure 7: Flow Rate Sensor

Arduino UNO Board

We chose Arduino UNO as the micro-controller because of its moderate price over other micro-controllers and because it serves our purpose. We just need to get the output signals of our sensors and some simple computations, so arduino UNO would suffice. (we are planning to prototype only one node, but if multiple nodes had to be built we could also go with arduino nano as it is much cheaper)

A7 GSM Module

We'd be also using an A7 GSM module to capture the signals from the micro-controller and to transfer these signals to the centralized server.

- Working voltage : 3.3V-4.2V
- Power voltage: >3.4V

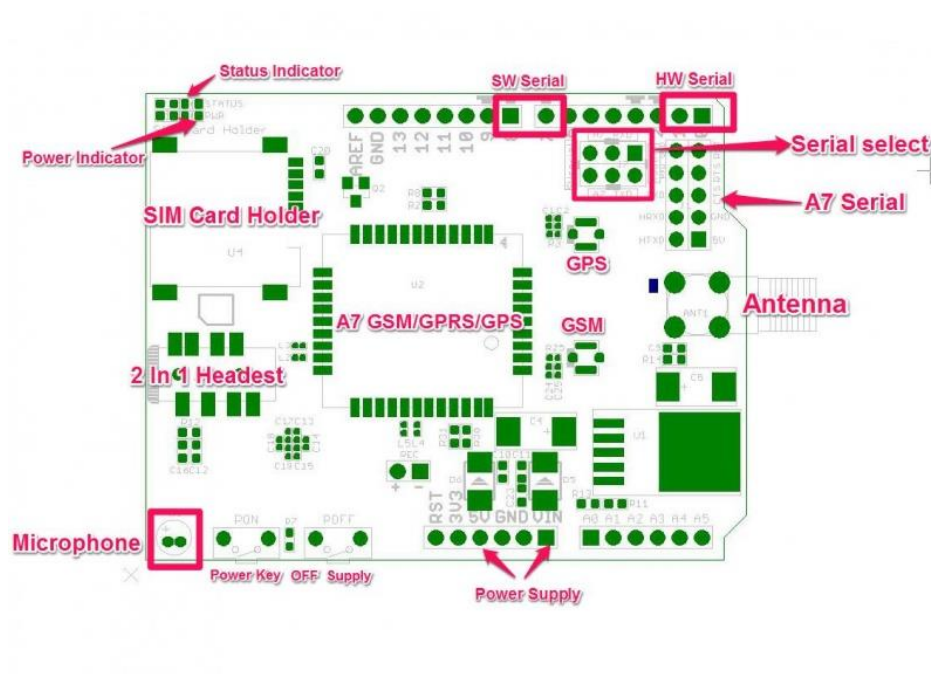


Figure 8: Layout of an A7 GSM module

Network Design

Overview

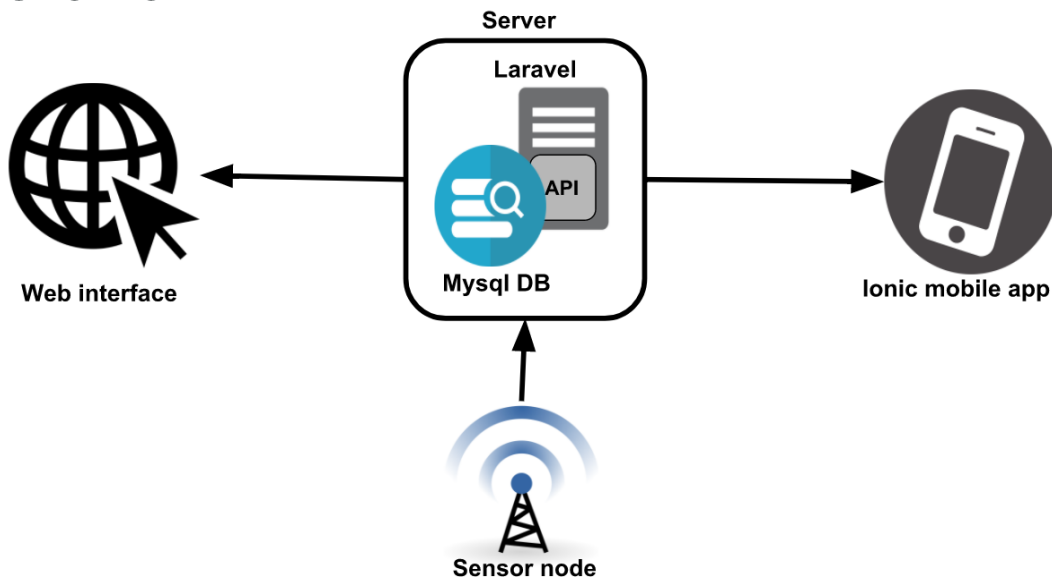


Figure 9: Network Overview

Data from multiple nodes are sent to the centralized server, in which database and other APIs are embedded within.

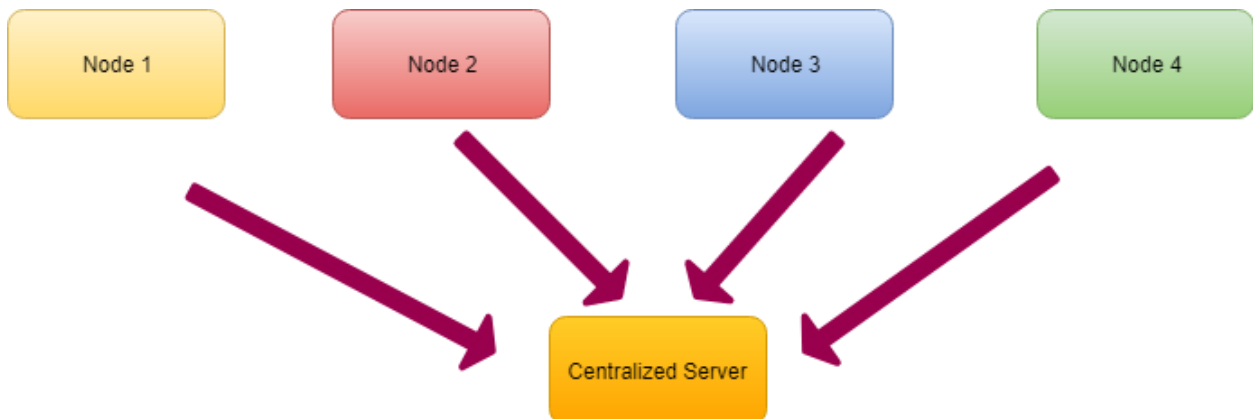


Figure 10: Multiple nodes connecting to the central server

Technologies

Laravel is basically a php framework. We mainly used laravel framework to create our application as it provides many features that are really helpful in building a real time notification system.

Some of the features to note down are ;

- Simple,fast routing engine.
- Powerful dependency injection container .
- Multiple back-ends for session and cache storage.
- Expressive, intuitive database ORM.
- Database agnostic schema migrations.
- Robust background job processing.
- Real-time event broadcasting.

Laravel also provides an attractive platform for MVC architecture.

1. Controller receives data from the GSM module
2. Controller requests data from model
3. Model returns data
4. Controller processes data
5. View receives data
6. Generated view is returned
7. Generated view is sent to the browser as a response.

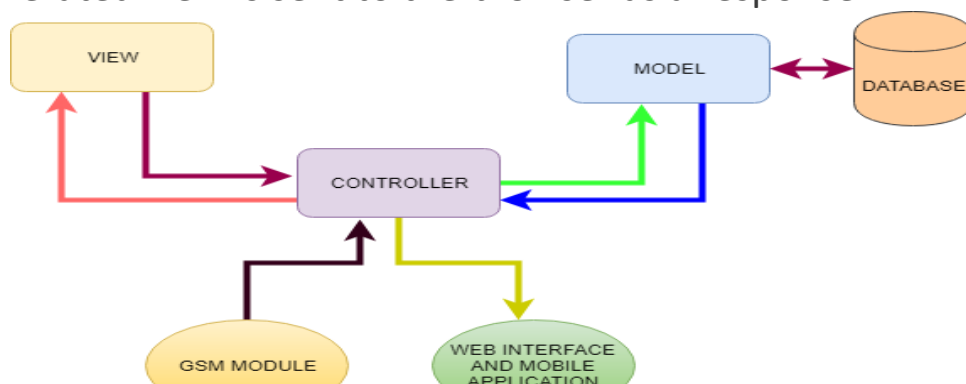
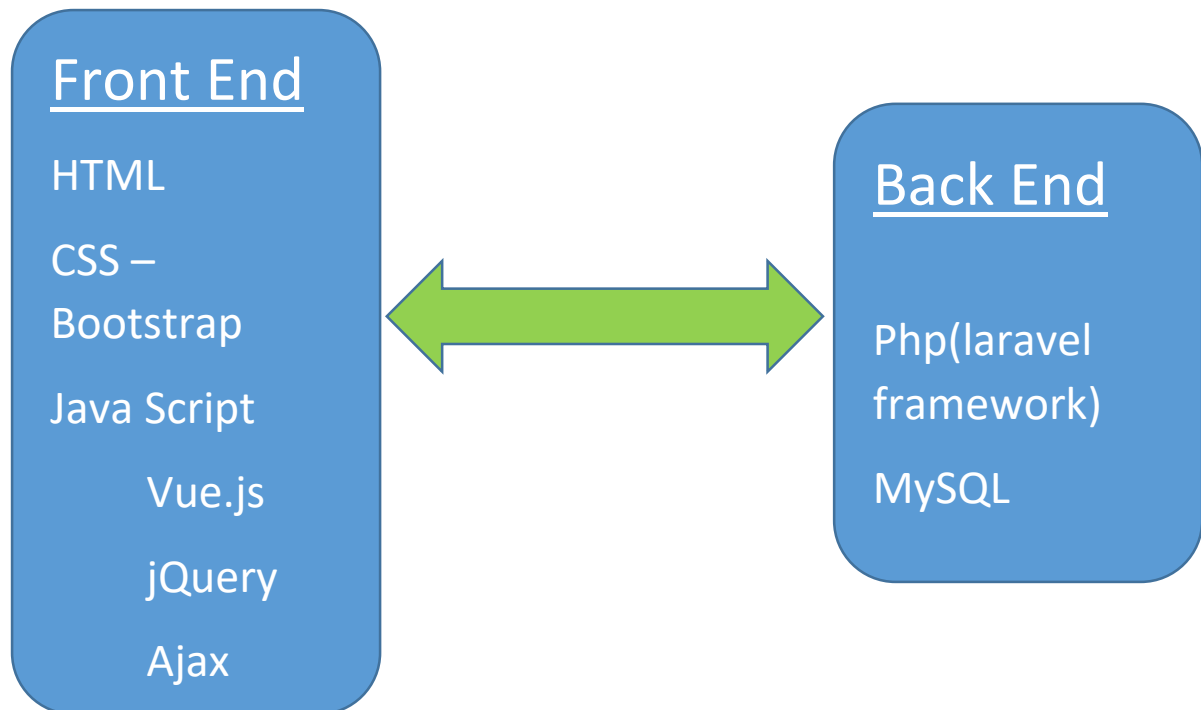


Figure 11 :MVC architecture of laravel framework

Technologies used in the web application



Technologies used in Mobile Application

We mainly used the ionic framework as it is compatible with both android and ios.



Figure 12: Reasons for choosing ionic

Application Platform Interfaces (APIs) Used

RESTful API (Laravel)

It is an application program interface that uses HTTP requests to GET, PUT, POST and DELETE data.

A RESTful API -- also referred to as a RESTful web service -- is based on representational state transfer technology, an architectural style. One of the key advantages of REST APIs is that they provide a great deal of flexibility. Data is not tied to resources or methods, so REST can handle multiple types of calls, return different data formats and even change structurally with the correct implementation of hypermedia.

Pusher API

Pusher is a simple hosted API for quickly, easily and securely adding real time bi-directional functionality via WebSockets to web and mobile apps, or any other Internet connected device. Pusher offers a rich suite of libraries that can be used within our applications. This makes the real time notification system much easier.

Google Maps API

Google Maps API allows display the location of each location of nodes in our web application and also the user can search nodes according to his/her location.

We use Postman to test and manage our APIs. Reasons for using postman are;

1. It has an easy to use interface
2. It's automation capabilities - It helps to automate the process of making API requests and testing API responses, allowing developers to establish a very efficient workflow
3. History/Auto complete
4. Easy organization - Postman allows API calls to be organized into groups that can be saved as "collections." Folders can be added to collections allowing API calls to be further organized into sub-collections. Collections and folders are especially useful when consuming many APIs and regularly testing a large number of API calls. Collections make it possible for developers to find and reuse specific API requests quickly.
5. Response viewer
6. Test Editor and runner

Interface Design

FLOOD DETECTION SYSTEM ABOUT WATER LEVEL CONTACT CREATE NODE DHANUSHKI ▾

Station Name	River	Alert Level (m)	Minor Flood Level (m)	Major Flood Level (m)	Current Water Level (m)	Velocity (m/s)	Condition
Deraniyagala	Kelani	4.88	5.79	6.36	7	9	Flood
Norwood	Kelani	1.5	2	2.15	3	0	Flood
Holombuwa	Kelani	3	3.35	4.87	5	2	Flood
Nagalagam Street	Kelani	1.22	1.52	2.13	0.48	0.36	Normal
Kitulgala	Kelani	2	3	5	10	7	Flood
Glencorse (Awissawella)	Kelani	15.5	16.76	19.81	3	800	Normal
Hanwella	Kelani	7	8	10	0	2	Normal
Nawalapitiya	Mahaweli	3	4	4.5	7	4	Flood

Copyright © FDS 2018

FLOOD DETECTION SYSTEM ABOUT WATER LEVEL CONTACT CREATE NODE DHANUSHKI ▾

Deraniyagala River: Kelani

Alert Level

4.88

Minor Flood Level

5.79

Major Flood Level

6.36

Current Level

7

Current Velocity



9

Latitude

6.9269106738286

Longitude

80.336648008984

Copyright © FDS 2018

FLOOD DETECTION SYSTEM ABOUT WATER LEVEL CONTACT CREATE NODE DHANUSHKI ▾

Station name:

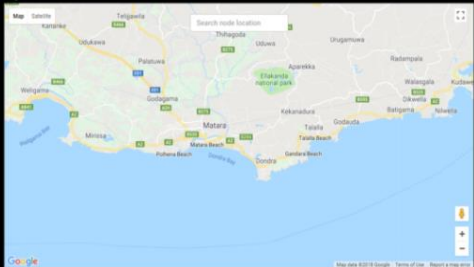
River:

Alert Level:

Minor Flood Level (m):

Major Flood Level (m):

Map



Longitude:

Latitude:

Insert image: No file chosen

Copyright © FDS 2018

Figure 13: Interface Designs

Database Design

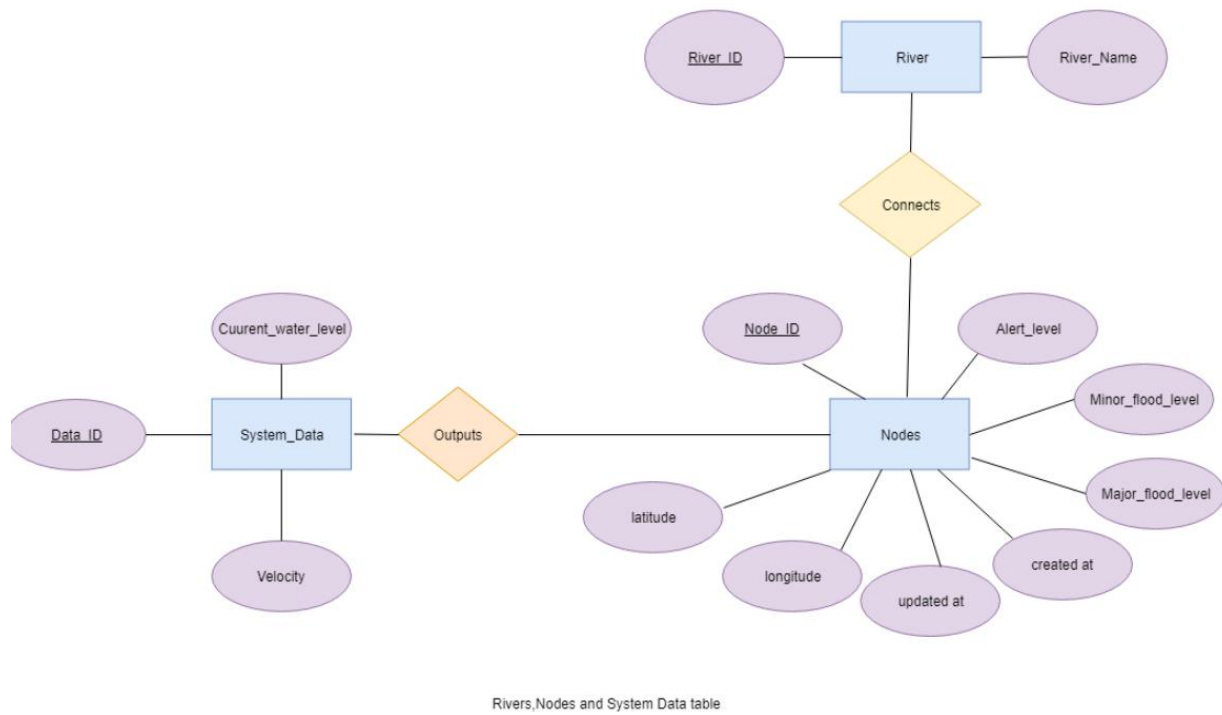
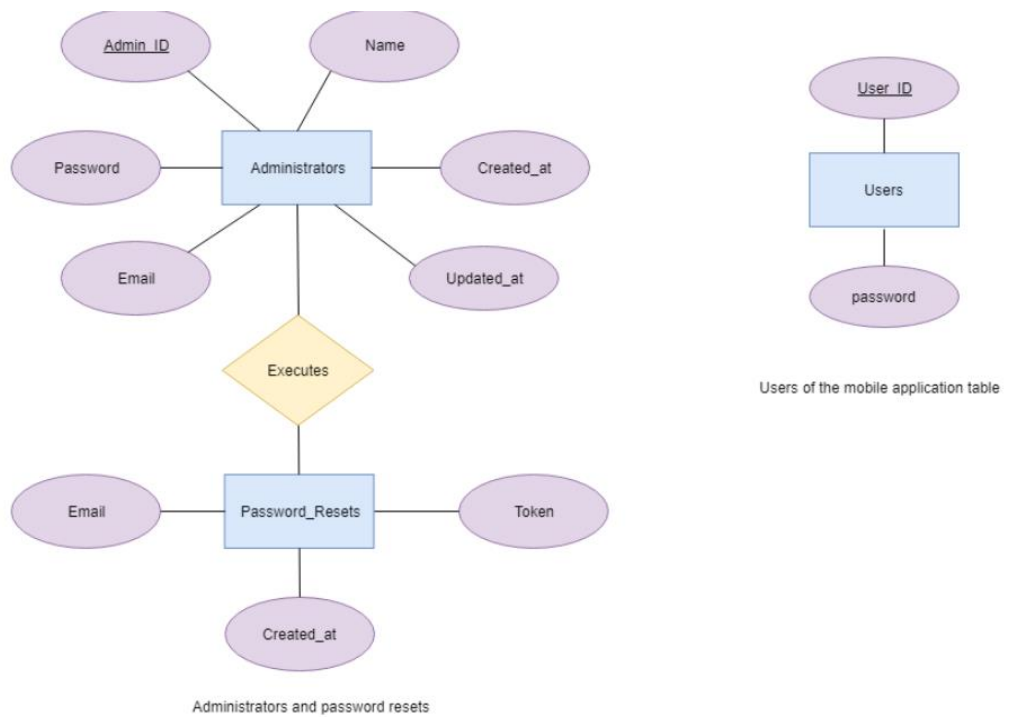


Figure14: Database Design

Network Security

Since our system doesn't contain any sensitive data, and is intended to be available to the public, encrypting the data would not serve any purpose. Instead we are focusing on taking suitable precautions against any 3rd party manipulating and corrupting our data. To overcome this issue we are mainly focusing on 3 aspects.

- Activating an SSL certificate for the web interface
- User Authentication
- Authenticating the message from the microcontroller using message authentication code (MAC)

MAC

A message authentication code (often called MAC) is a block of a few bytes that is used to authenticate a message. The receiver can check this block and be sure that the message hasn't been modified by the third party

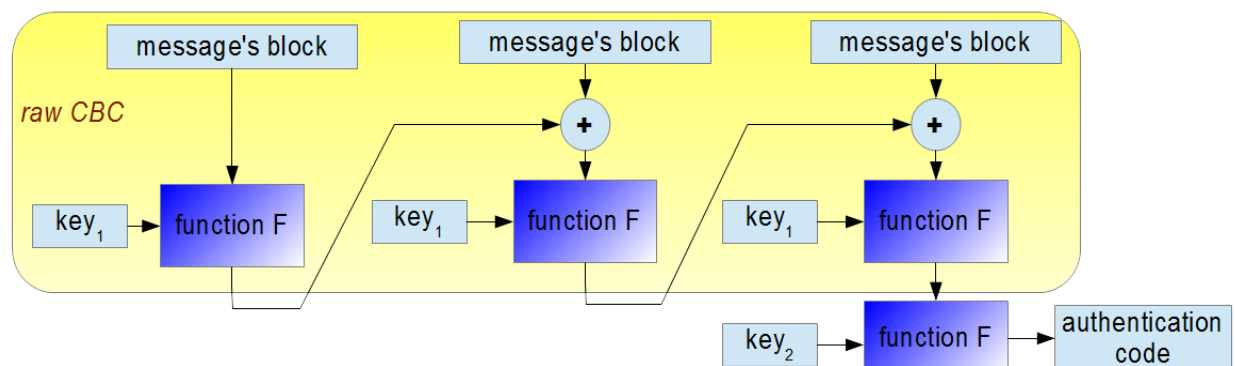


Figure 15: Theory behind MAC protocol

Budget

Table 1: Budget

Arduino mega	Rs 1110
SIMCOM SIM900 Quad-band GSM GPRS Shield for Arduino	Rs 3777
ultrasonic sensor hc-sr04	Rs 300
Flow rate sensor	Rs 800
Other wires and stuff	Rs 1000
Total Amount	Rs 6987

Timeline

Table 2:Timeline

Description	1	2	3	4	5	6	7	+6	8	9	10	11	12	13	14
Project selection, Problem identification, feasibility analysis and background study															
Planning and designing of the embedded system, required network for communications, servers and security aspects															
Testing with different components of the initial plan, implementing the hardware, developing the communication network considering security aspects, developing the web interface and the mobile application while making relevant changes to overcome the limitations of the initial plan															
Testing and other further implementations															
Project completion and documentation															

Testing

The testing was done in three main streams hardware testing, network testing and security testing. After being tested separately all the components were integrated and tested again for their proper functionality.

Hardware testing

Ultrasonic sensor



Figure 56: Apparatus to test the ultra sonic sensor

A set up was implemented as shown in the above diagram. The height from the ultra-sonic sensor to the surface of water was measured by a meter ruler. Then this value was compared with the value given as an output by the arduino UNO board which is connected to the ultrasonic sensor. Three values were taken for each height value and the mean was calculated as the final value to minimize the experimental errors.

Height(cm)	Obtained Values			Avg Value	Error
	Case 1	Case 2	Case 3		
10	12	16	14	14	4
20	16	22	24	24	4
30	26	28	26	26	-4
40	42	43	44	43	3
50	53	53	52	53	3
60	59	62	64	63	3
70	71	70	70	70	0
80	81	81	80	79	-1
90	90	90	90	91	1
100	100	100	100	99	-1
110	110	109	110	110	0
120	124	123	121	119	-1
130	131	132	131	131	1
140	141	142	144	139	-1
150	152	152	150	150	0
160	163	162	162	159	-1
170	173	172	171	171	1
180	184	183	182	182	2
190	192	192	190	191	1
200	201	202	201	202	2

Absolute value of maximum error=4cm

Absolute value of minimum error=0

Error at the end points were higher than the middle points. So when fixing the apparatus it should be fixed at a height of about 1m above the water surface.

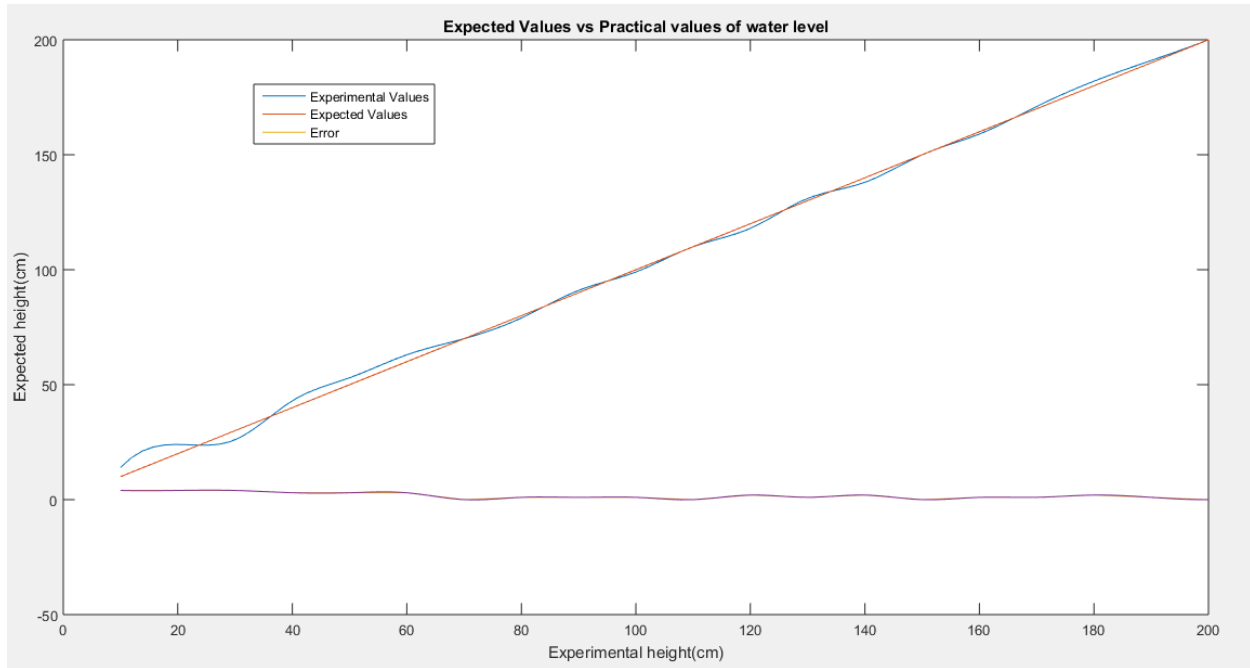


Figure 17:Expected value vs practical values

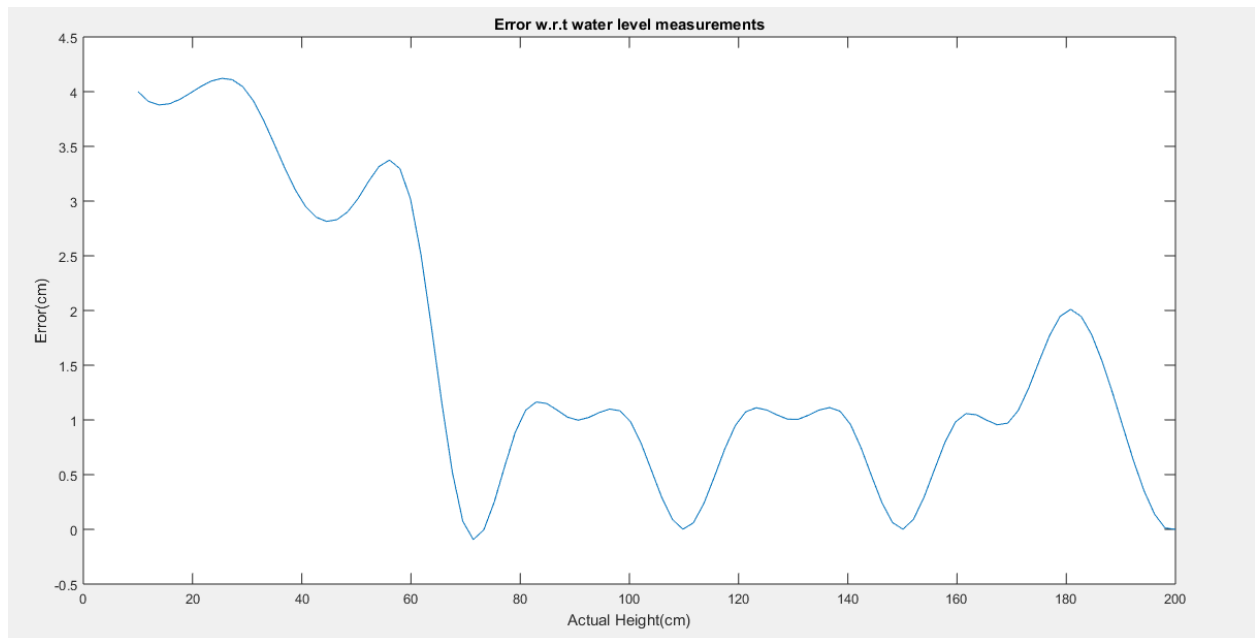


Figure 18:Error with regard to height measurements

Flow Rate Sensor

The flow rate sensor was measured by comparing the values outputted by the flow rate sensor we used with a flow meter of a higher accuracy. Both flow rate sensor and the flow meter were emerged in a stream and the values were checked in several places with different velocities. The flow meter had an accuracy of $\pm 5\%$ which was higher than the accuracy of the flow rate sensor we used, which according to the manufacture's guidelines were $\pm 10\%$. The practical values we obtained also gave an accuracy about $\pm 10\%$.

Flow Rate Sensor		
Expected Value(ml/s)	Obtained Value(ml/s)	Error
50	52	2
55.5	54	-1.5
51	52	1
54	53	-1
78	77	-1
79	78	-1
80.1	80	-0.1
80.5	82	1.5
125	125	0
127.5	128	0.5
129	129	0
135	134	-1
221.5	220	-1.5
223	222	-1
225	225	0
228	228	0
352	352	0
351.5	353	1.5
354	356	2
356.5	358	1.5
458	464	6
459	465	6
500	504	4
502.5	505	2.5

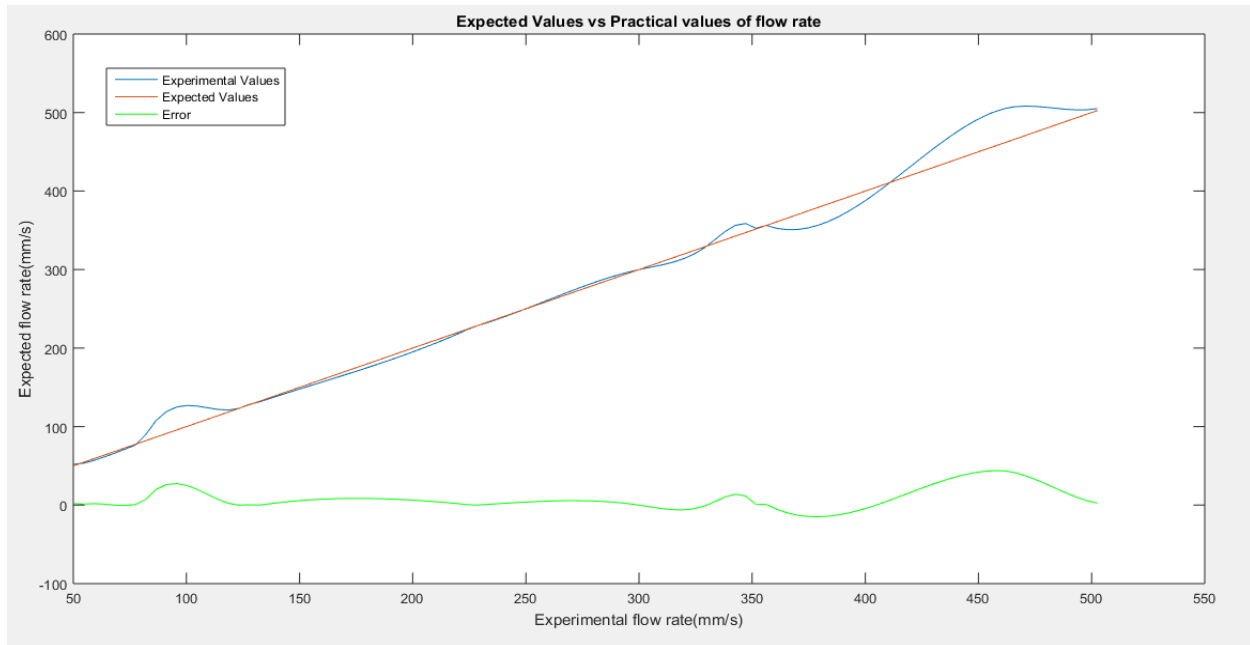


Figure 19: Expected values vs practical values

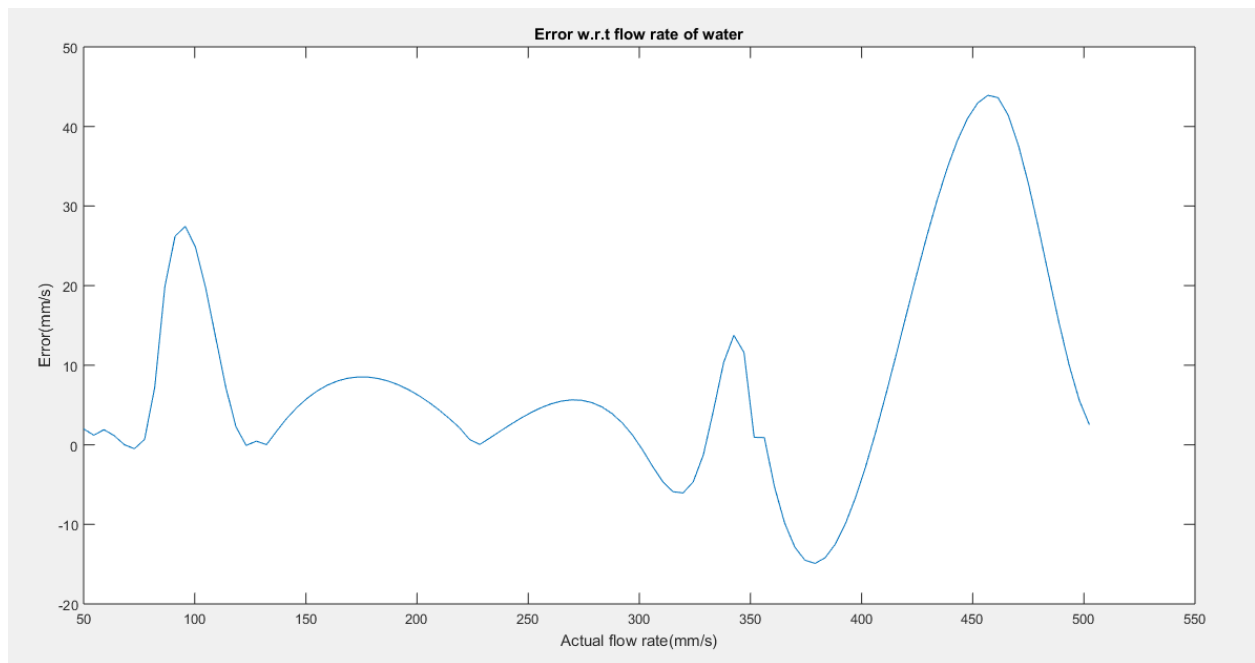


Figure 20: Error of the flow rate sensor

Maximum error = 6 ml/s

Minimum error = 0

Network testing

The central server is hosted at <https://flood.codechilli.lk> and several tests were used to test whether the networking aspect was functioning as expected. We used the postman API to test the condition of our network.

Test1

We sent a number of get requests to our server to calculate how much time it took to update the website once a request was sent to the server.

Test Case	Time taken(ms)
1	271
2	546
3	428
4	1654
5	601
6	454
7	4518
8	840
9	1302
10	1354
11	493
12	1382
13	924
14	610
15	229
16	523
17	1494
18	224
19	356
20	294

21	3324
22	298
23	1846
24	280
25	388
26	371
27	388
28	844
29	1470
30	351
31	247
32	602
33	441
34	232
35	329
36	334
37	336
38	365
39	335
40	743
41	729
42	415
43	500
44	257
45	233
46	434
47	415
48	727
49	835
50	264

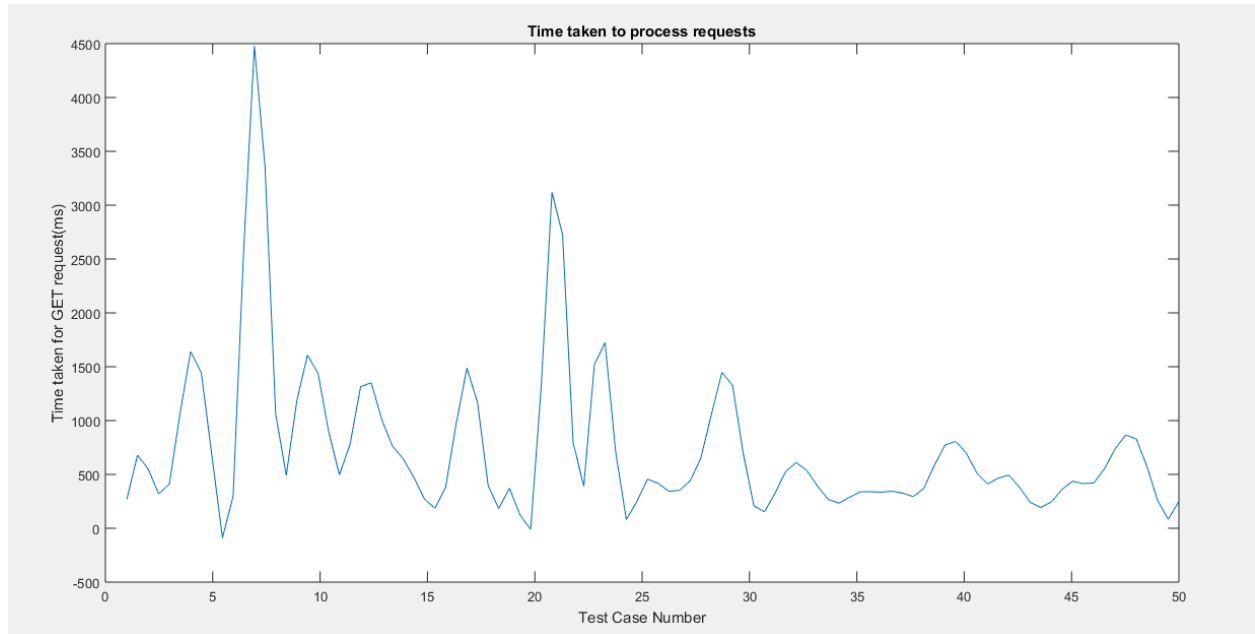


Figure 21:time taken to process requests

Maximum value=4518 ms

Minimum value =224 ms

Range =4294ms

Average time taken to process a request : 736.6ms

Standard Deviation : 785.90ms

Test 2

We conducted another test to experiment the total round trip time (the time taken to transmit data from a node to the server plus the time taken to update the server) to measure the efficiency of our service.

Test Case	time(ms)
1	61200
2	72000
3	79260
4	114480
5	70020
6	87360
7	73800
8	58800
9	94020
10	73800
11	70680
12	60720
13	61920
14	59760
15	69000
16	53400
17	47340
18	63360
19	72000
20	68700

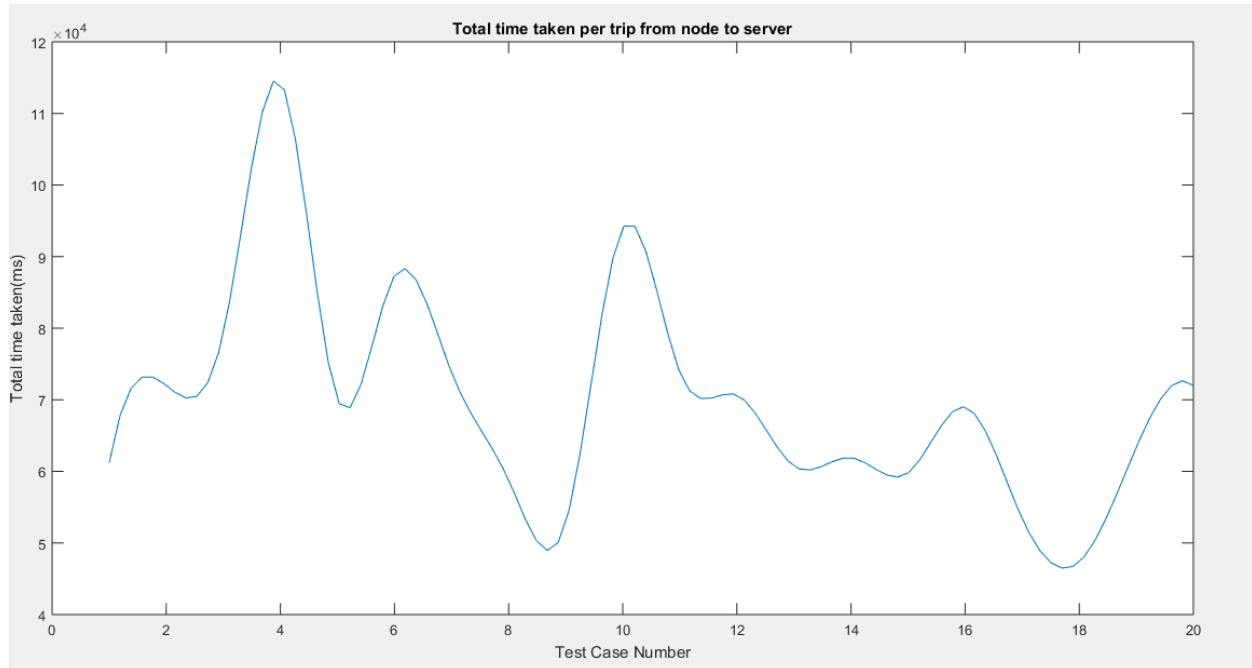
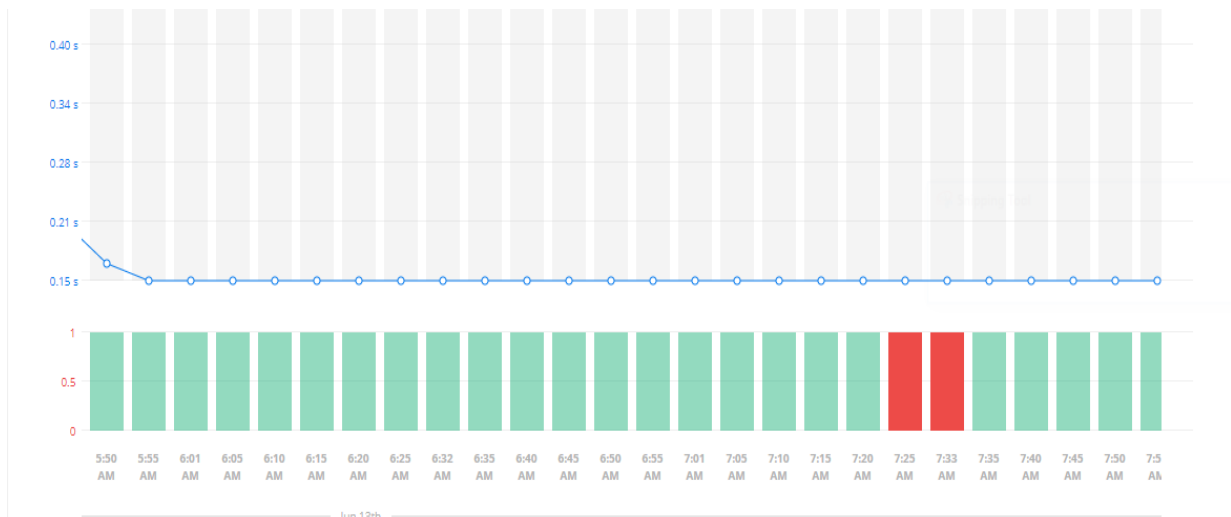


Figure 22: Total round trip time

Maximum time=114480ms
Minimu time=53400ms
Range=61080ms
Standard Deviation : 81.32s
Average : 69.7s

Test 3

In order to measure the network traffic we used the same API to constantly send 20 requests each 5 minutes and check whether any problems occur. Here, we observed all the packets were transferred without any issue except for two occasions. Later we found out that the two occasions were due to a network error in the local machine and was no effect in the system.



Security Testing

Test 1

We have used the Message Authentication Code (MAC) to prevent any third party manipulating the data in a disruptive manner. There's a mac value produced at the node and that value is only updated in the server side if and only if the mac value produced at the server side is compatible with the mac value from the node.

We have used a secret key and anyone who is trying to decrypt the mac code with the secret key may have to run 2^{97} queries only with a success probability of 0.87%. This proves that decrypting the mac code would be hardly possible thus ensuring the security aspect and the reliability. The test cases we conducted proved to be of high reliability and all of the test cases passed.

Mock Server	Correct value for mac	Mock value for mac	
11/2/3.0	14933a7067d59cb77466d3d360fce61c	14933a7067d59cb77466d3d360fce61d	Failed
12/4/5.0	55f8efaa89d1894ec474ed8404ea2fb3	65f8efaa89d1894ec474ed8404ea2fb3	Failed
13/2.5/3	0f7ed67bd7c2a4184dfe3872e3569e5e	0f7ed67bd7c2a4184dfdhgD2e3569e5e	Failed
14/3/1.1	13b8d9325e5f95f15ca6a70736ef97f4	13b8d9325e5f95f15ca6a70736ef97f4ghgh	Failed
11/5/9.0	c2e0ed54e9c7e051170b19b20c4a1000	c2e0ed54e9c7e051170b19b20ckjkk1000	Failed
12/3/4.0	5c30b99baa194de4d9c374bba9ed611f	5c30b99baa194de4d9c374bba9ejkhjjkk	Failed
11/2/3.0	14933a7067d59cb77466d3d360fce61c	14933a7067d59cb77466d3d360fce61c	Success
12/4/5.0	55f8efaa89d1894ec474ed8404ea2fb3	55f8efaa89d1894ec474ed8404ea2fb3	Success
13/2.5/3	0f7ed67bd7c2a4184dfe3872e3569e5e	0f7ed67bd7c2a4184dfe3872e3569e5e	Success
14/3/1.1	13b8d9325e5f95f15ca6a70736ef97f4	13b8d9325e5f95f15ca6a70736ef97f4	Success
11/5/9.0	c2e0ed54e9c7e051170b19b20c4a1000	c2e0ed54e9c7e051170b19b20c4a1000	Success
12/3/4.0	5c30b99baa194de4d9c374bba9ed611f	5c30b99baa194de4d9c374bba9ed611f	Success

Test 2

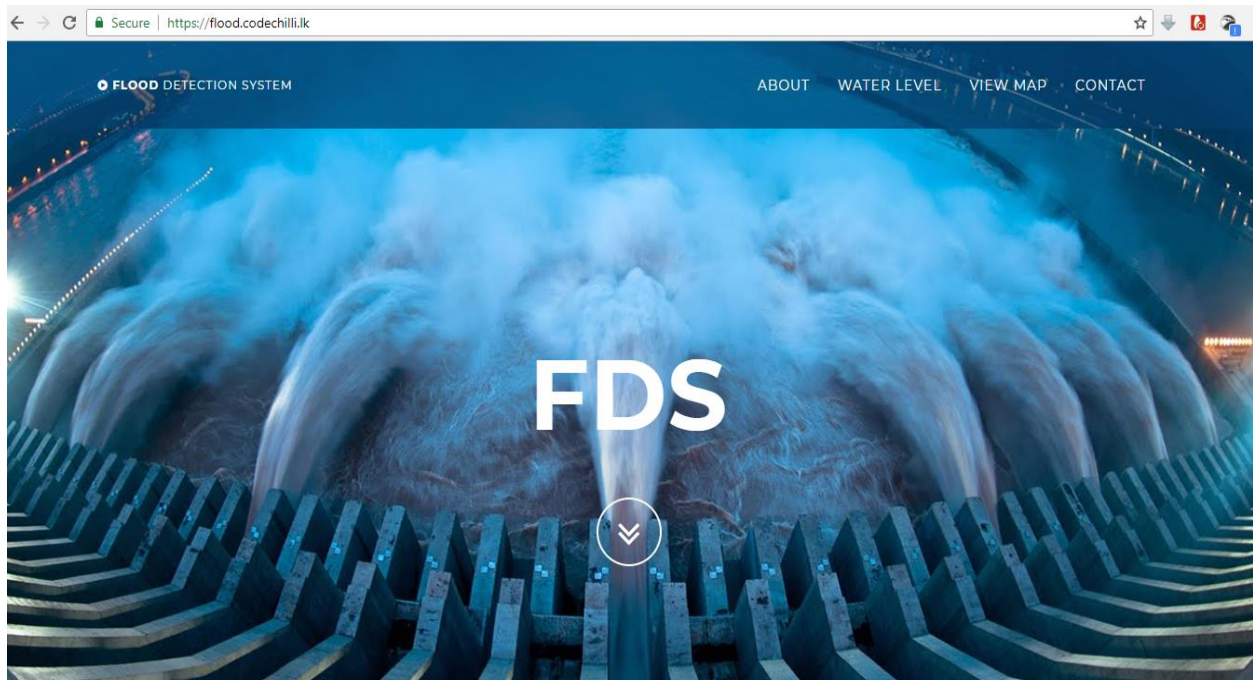
The application had two interfaces one being the public interface and the other one being the admin interfaces. Only the administrators are authenticated to do the modifications to the data. The public can only view the data. We have authenticated the administrative staff who are eligible to do certain changes and we have tested our application for several times to check whether this feature works correctly. The application showed to give the expected results in every test case.

User manual

Web Interface

The public interface can be accessed by anyone at

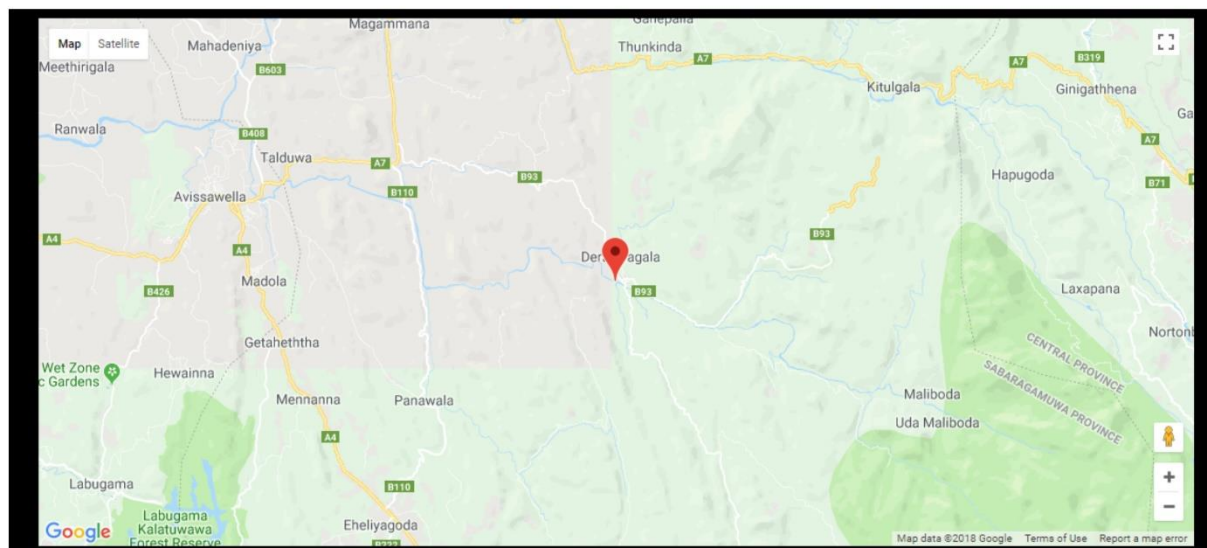
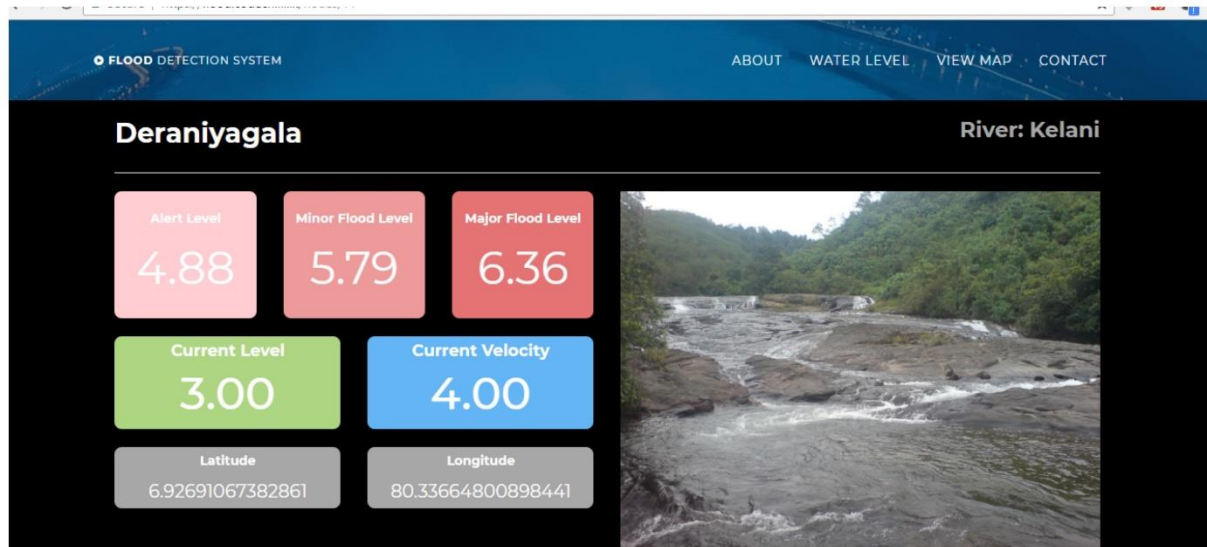
<https://flood.codechilli.lk/>



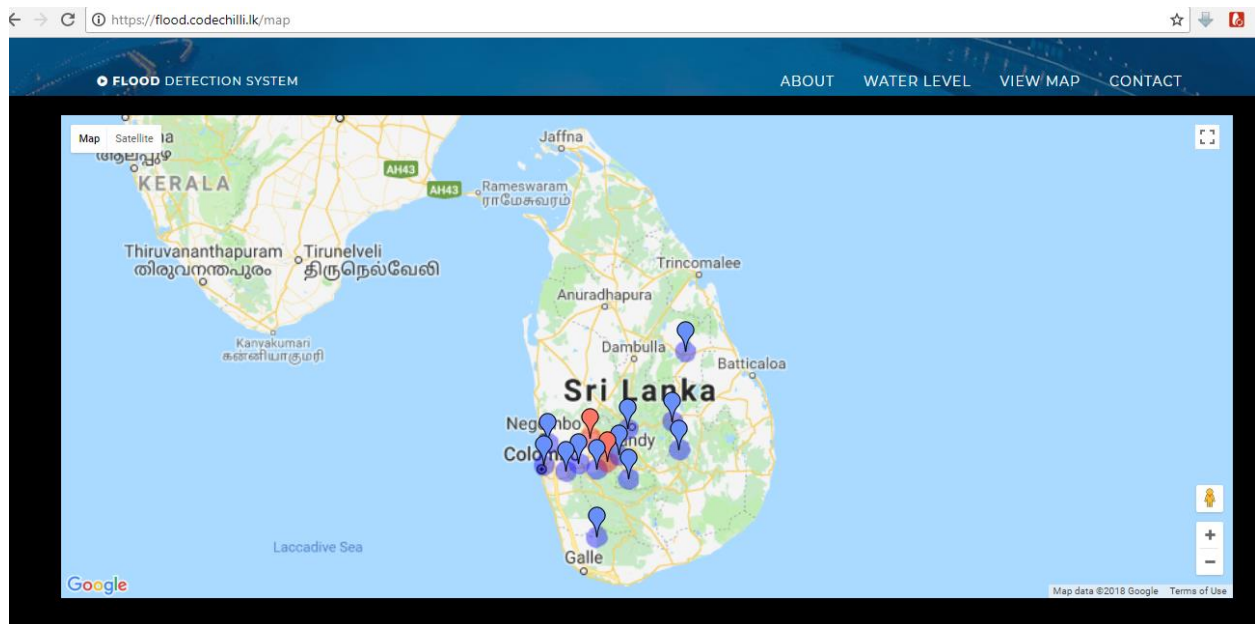
Once a person is accessed to this page he/she can view the water levels of different rivers and water bodies through Water Level tab on the top of the page.

Station Name	River	Alert Level (m)	Minor Flood Level (m)	Major Flood Level (m)	Current Water Level (m)	Velocity (m/s)	Condition
Deraniyagala	Kelani	4.88	5.79	6.36	3.00	4.00	Normal
Norwood	Kelani	1.50	2.00	2.15	1.00	0.50	Normal
Holombuwa	Kelani	3.00	3.35	4.87	3.50	0.70	Flood
Nagalagam Street	Kelani	1.22	1.52	2.13	1.20	0.36	Normal
Kitulgala	Kelani	2.00	3.00	5.00	2.50	0.39	Flood
Hanwella	Kelani	7.00	8.00	10.00	0.00	0.50	Normal
Nawalapitiya	Mahaweli	3.00	4.00	4.50	2.50	0.40	Normal
Peradeniya	Mahaweli	5.00	7.00	9.00	0.50	0.22	Normal
Glencourse	Kelani	15.50	16.76	19.81	13.30	0.70	Normal
Weragantota	Mahaweli	5.00	6.00	8.00	4.20	0.50	Normal
Taldena	Mahaweli	3.00	4.00	5.00	0.00	0.00	Normal
Manampitiya	Mahaweli	3.50	3.90	4.00	0.00	0.00	Normal
Dunamale	Attanagalu Oya	3.30	4.40	5.50	0.00	0.00	Normal

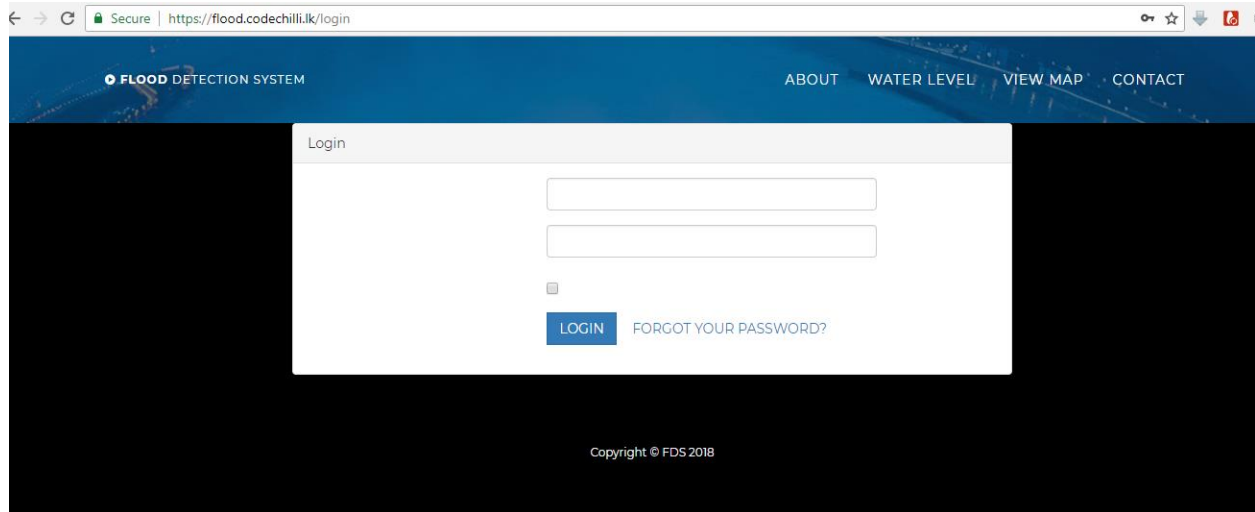
If a user wants to view more details about a specific station (water body) he/she can click on the station name and find more details.



You can also view the overall map from the view map tab on the top of the home page. In the map, the water bodies that are safe would be shown in blue while the flood conditioned locations are shown in red color.



Only the administrators that are authenticated can make modifications in the public interface. If you are already authenticated as an admin <https://flood.codechilli.lk/login> and the following page will pop up.



If the correct credentials are supplied, a new web page will load.

Station Name	River	Alert Level (m)	Minor Flood Level (m)	Major Flood Level (m)	Current Water Level (m)	Velocity (m/s)	Condition
Deraniyagala	Kelani	4.88	5.79	6.36	3.00	4.00	Normal
Norwood	Kelani	1.50	2.00	2.15	1.00	0.50	Normal
Holombuwa	Kelani	3.00	3.35	4.87	3.50	0.70	Flood
Nagalagam Street	Kelani	1.22	1.52	2.13	1.20	0.36	Normal
Kitulgala	Kelani	2.00	3.00	5.00	2.50	0.39	Flood
Hanwella	Kelani	7.00	8.00	10.00	0.00	0.50	Normal
Nawalapitiya	Mahaweli	3.00	4.00	4.50	2.50	0.40	Normal
Peradeniya	Mahaweli	5.00	7.00	9.00	0.50	0.22	Normal
Glencourse	Kelani	15.50	16.76	19.81	13.30	0.70	Normal
Weragantota	Mahaweli	5.00	6.00	8.00	4.20	0.50	Normal
Taldena	Mahaweli	3.00	4.00	5.00	0.00	0.00	Normal
Manampitiya	Mahaweli	3.50	3.90	4.00	0.00	0.00	Normal

An admin can create new nodes using the create node tab.

WATER LEVEL MONITORING SYSTEM ABOUT WATER LEVEL VIEW MAP CONTACT CREATE NODE DHAN

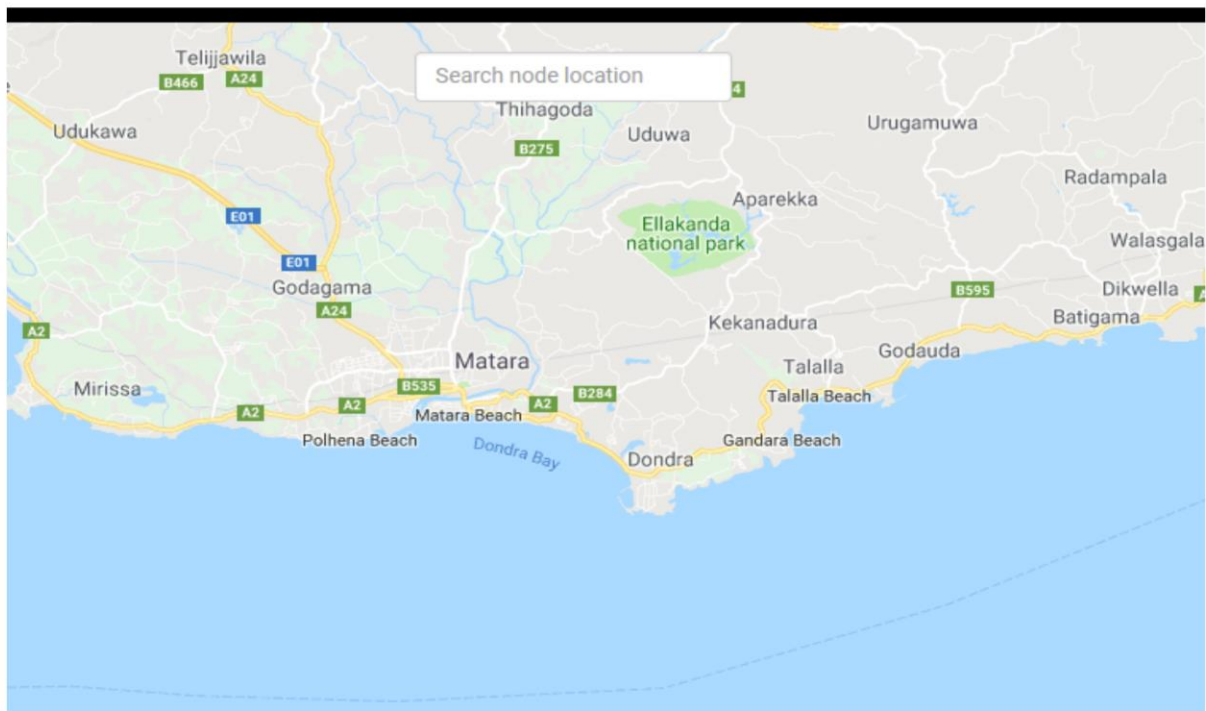
Station name:

River:

Alert Level:

Minor Flood Level (m):

Major Flood Level (m):



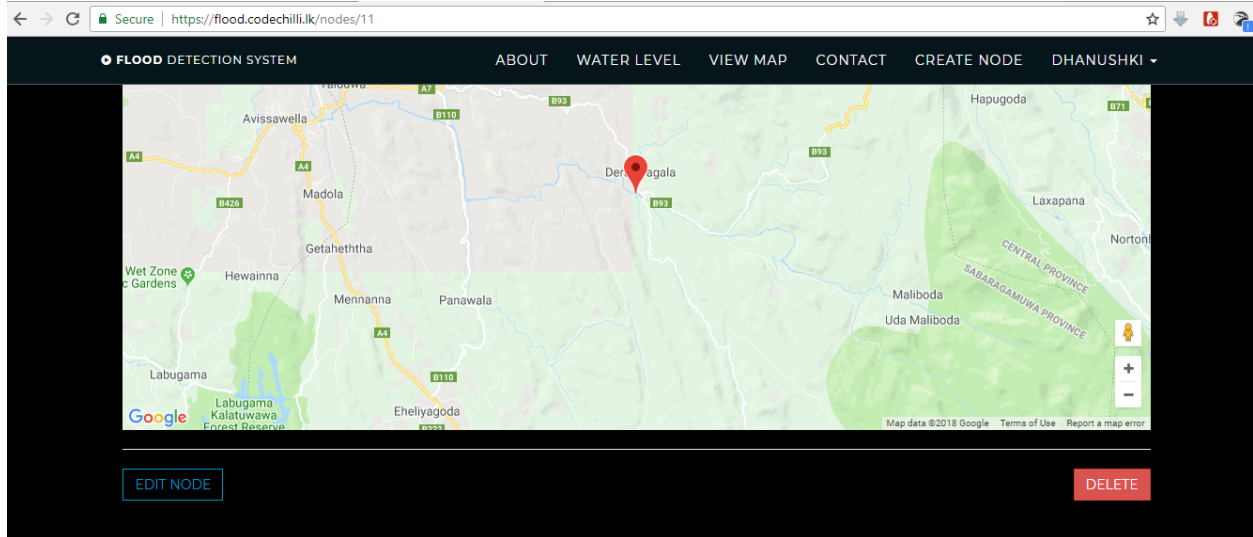
Google Map data ©2018 Google Terms of Use Report a map error

Longitude :

Latitude :

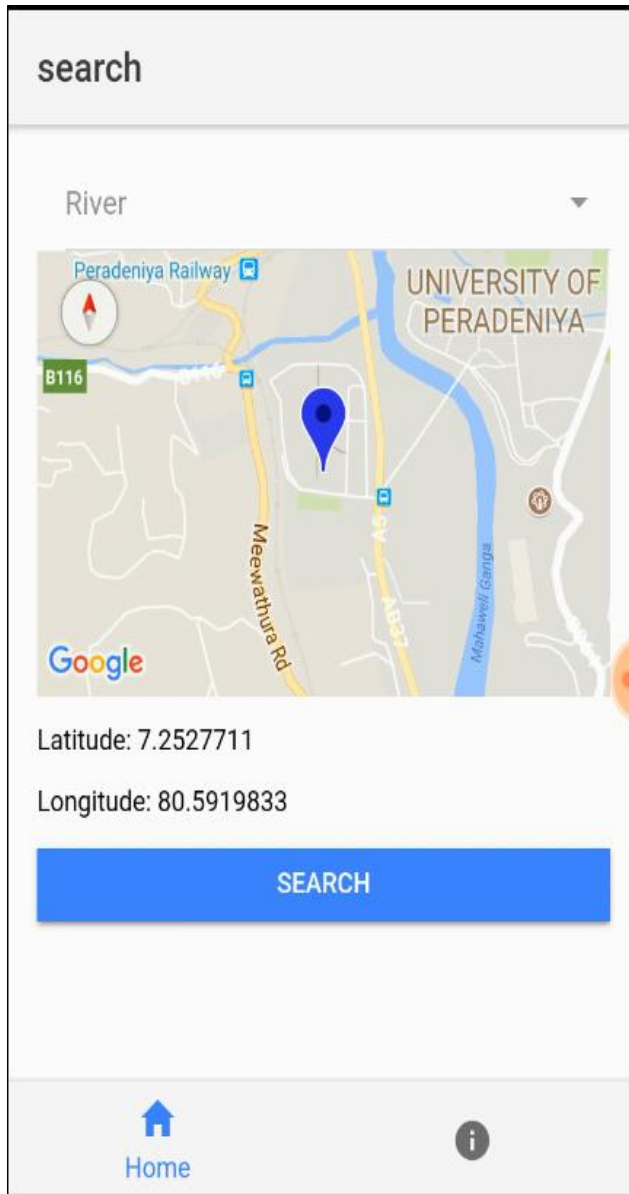
Insert image : No file chosen

An admin can also make changes in a particular node using EDIT NODE and DELETE NODE.

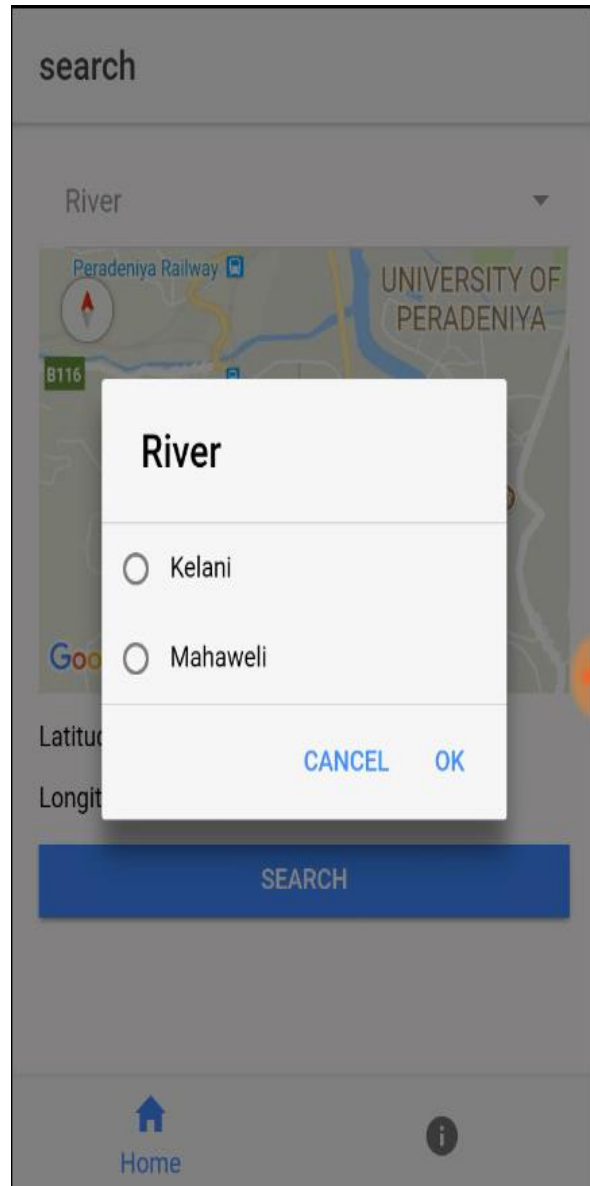


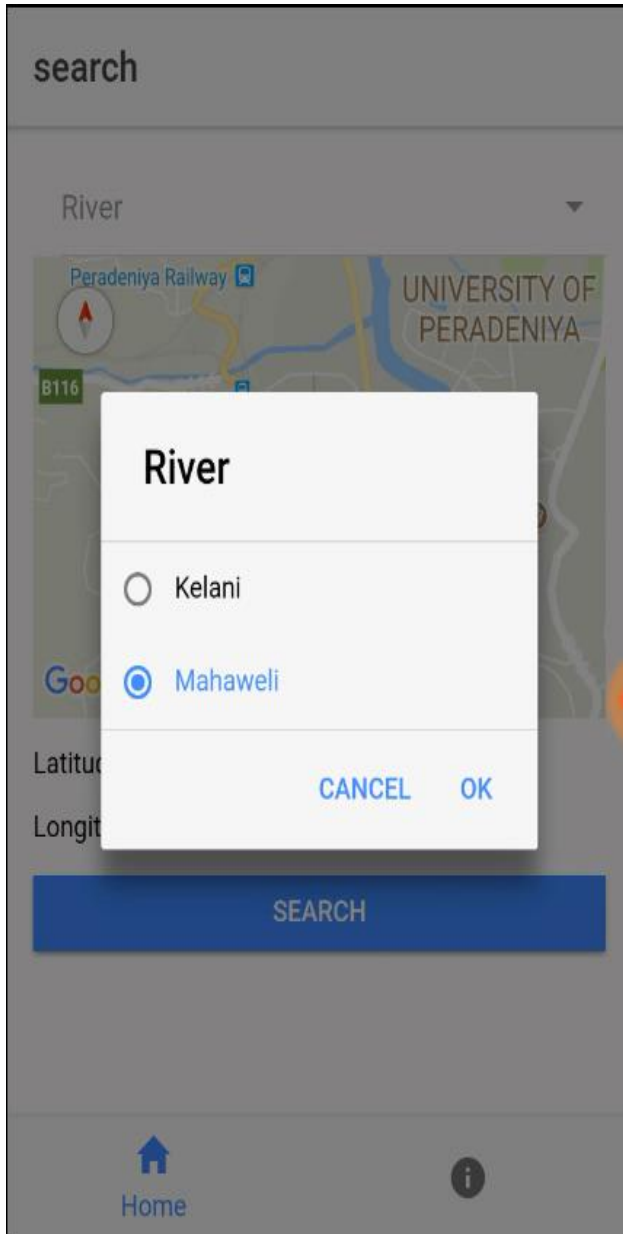
Mobile Application

Step 1

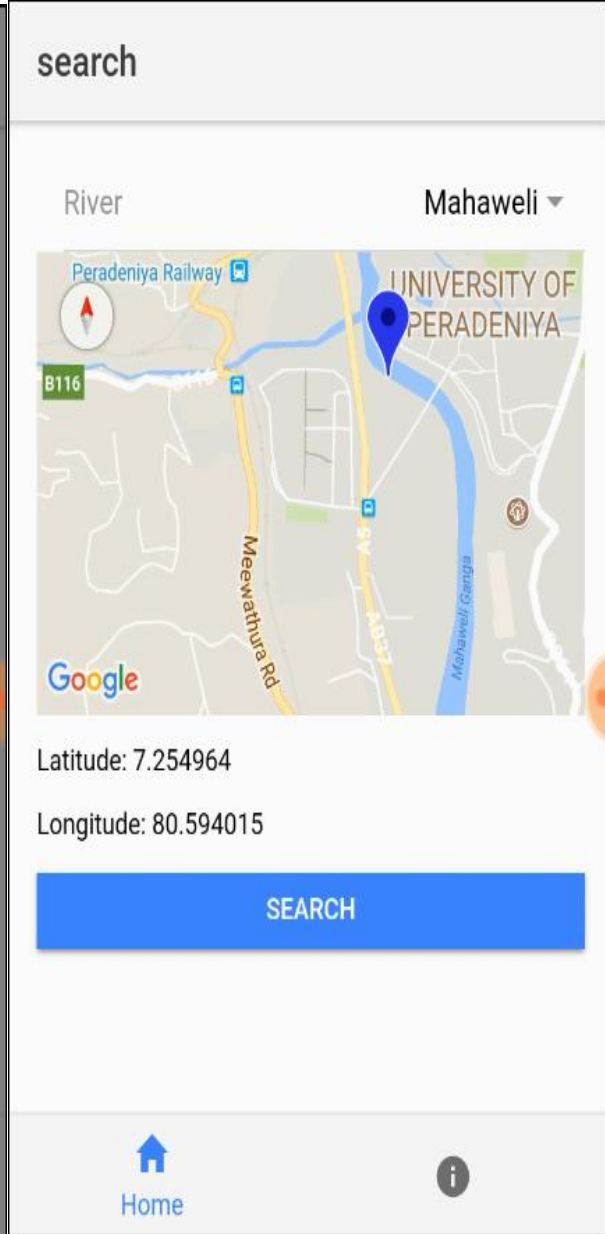


Step 2






Step 3



Step 4

search





Latitude: 7.254964

Longitude: 80.594015

SEARCH

Current water level: 4.80

Current velocity: 0.50

 Home 

Result

Reference for further Implementation

Source code for the web interface can be found at

<https://github.com/DhanushkiMapitigama/Unified-Project.git>

Cloning a laravel project

- Clone your project
- Go to the folder application using `cd` command on your cmd or terminal
- Run `composer install` on your cmd or terminal
- Copy `.env.example` file to `.env` on the root folder. You can type `copy .env.example .env` if using command prompt Windows or `cp .env.example .env` if using terminal, Ubuntu
- Open your `.env` file and change the database name (`DB_DATABASE`) to whatever you have, username (`DB_USERNAME`) and password (`DB_PASSWORD`) field correspond to your configuration. By default, the username is `root` and you can leave the password field empty. (**This is for Xampp**)
By default, the username is `root` and password is also `root`. (**This is for Lamp**)
- Run `php artisan key:generate`
- Run `php artisan migrate`
- Run `php artisan serve`
- Go to `localhost:8000`

More details of how to clone a laravel project can be found at <https://devmarketer.io/learn/setup-laravel-project-cloned-github-com/>

The source code for the mobile application can be found at

<https://github.com/DhanushkiMapitigama/fds-app/tree/development>

The source code to be inserted to the arduino board is found at

<https://github.com/DhanushkiMapitigama/Unified-Project/tree/shanakamunasinghe-patch-3/test2>

The codes used for test cases

Flow rate sensor : <https://github.com/DhanushkiMapitigama/Unified-Project/blob/shanakamunasinghe-patch-3/FlowRateSensor.ino>

Ultrasonic sensor : https://github.com/DhanushkiMapitigama/Unified-Project/tree/shanakamunasinghe-patch-3/level_meter

