**Software Design Description**

**For**

**Smart Water Level Detection System**

**Prepared by**

Farah Abdul Muttalib

Zulfadli Kamaruzzaman

Syahmi Khairudin

Muhammad Amirul Fahmi Abdullah

Mohamad Shahnizam Abd Saini

Ahmad Zulzhafri Zuki

Wan Zahiruddin Wan Abd Kadir

Atif Abd Razak

Table of Contents

[**1** **Introduction** 4](#_Toc484589979)

[**1.1** **Purpose** 4](#_Toc484589980)

[**1.2** **Scope** 5](#_Toc484589981)

[**1.3** **Definition, Acronyms, and Abbreviations** 6](#_Toc484589982)

[**1.4** **Overview** 7](#_Toc484589983)

[**1.5** **Reference Material** 7](#_Toc484589984)

[**2** **System Overview** 8](#_Toc484589985)

[**3** **Design Consideration** 10](#_Toc484589986)

[**3.1** **Design Assumption, Dependencies and Constraints** 10](#_Toc484589987)

[**3.1.1** **Design Assumption** 10](#_Toc484589988)

[**3.1.2** **Design Dependencies** 10](#_Toc484589989)

[**3.1.3** **Design Constraints** 11](#_Toc484589990)

[**4** **System Architectures Design** 14](#_Toc484589991)

[**4.1** **Raspberry Pi** 14](#_Toc484589992)

[**4.1.1** **System Architecture** 14](#_Toc484589993)

[**4.1.2** **System Description** 14](#_Toc484589994)

[**4.2** **Smartphone Apps** 17](#_Toc484589995)

[**4.2.1** **System Architecture** 17](#_Toc484589996)

[**4.2.2** **System Description** 17](#_Toc484589997)

[**4.3** **Webpage Django** 18](#_Toc484589998)

[**4.3.1** **System Architecture** 18](#_Toc484589999)

[**4.3.2** **System Description** 19](#_Toc484590000)

[**4.4** **Database** 20](#_Toc484590001)

[**4.4.1** **System Architecture** 20](#_Toc484590002)

[**4.4.2** **System Description** 21](#_Toc484590003)

[**5** **User Interface Design** 22](#_Toc484590004)

[**5.1** **Overview of User Interface** 22](#_Toc484590005)

[**5.1.1** **Webpage Django** 22](#_Toc484590006)

[**5.1.2** **Smartphone Apps** 23](#_Toc484590007)

[**5.2** **Interface Screens** 24](#_Toc484590008)

[**5.2.1** **Webpage Django** 24](#_Toc484590009)

[**5.2.2** **Smartphone Apps** 25](#_Toc484590010)

[**5.3** **UI Functionality** 26](#_Toc484590011)

[**5.3.1** **Webpage Django** 26](#_Toc484590012)

[**5.3.2** **Smartphone Apps** 27](#_Toc484590013)

[**6** **Detailed Design** 28](#_Toc484590014)

[**6.1** **UI Components** 28](#_Toc484590015)

[**6.1.1** **Webpage Django** 28](#_Toc484590016)

[**6.1.2** **Smartphone Apps** 31](#_Toc484590017)

[**6.2** **Webserver/Database** 31](#_Toc484590018)

[**7** **Libraries and Tools** 33](#_Toc484590019)

[**7.1** **Java (Android Studio)** 33](#_Toc484590020)

[**7.2** **Python** 34](#_Toc484590021)

[**7.2.1** **Django** 34](#_Toc484590022)

[**7.2.2** **Raspberry Pi** 34](#_Toc484590023)

[**7.3** **Database** 35](#_Toc484590024)

[**8** **Conclusion** 38](#_Toc484590025)

[**9** **Reflection** 38](#_Toc484590026)

[**10** **Appendixes** 44](#_Toc484590027)

[**10.1** **Scripts** 44](#_Toc484590028)

[**10.1.1** **Smartphone Apps** 44](#_Toc484590029)

[**10.1.2** **Webpage Django** 53](#_Toc484590030)

[**10.1.3** **Server/database** 63](#_Toc484590031)

[**10.1.4** **Raspberry Pi** 67](#_Toc484590032)

# **Introduction**

This document comprehends the software design descriptions for Smart Water Level Detection System. This document is prepared according to the “IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE 1016 – 2009”.

This document provides the details of how the Smart Water Level Detection software built and develop. The details are represented by graphical notation such as use case models, sequence diagrams, class diagram, object behavior models and other supporting design information.

## **Purpose**

The purpose of this documentation is to specify the objectives, aim and requirement for the Smart Water Level Detection System. In this documentation, it also provide the introduction and some of the analysis that can be carried out by the Smart Water Level Detection System. This project implement the concept of IoT that uses the benefits of internet.

Smart Water Level Detection System consist of 3 main parts that’s are web monitoring (Django), smartphone application and Smart Water Level Detection System Device (Raspberry Pi and water sensor).

* Smart Water Level Detection System Device – sensor and controller that bused to measure the water level at any environment it being installed
* Web monitoring – act as the database and monitoring the data that being send by the device. It also functions to create all the analysis for client to use for further actions. The web page can be access via any browser through internet.
* Smartphone apps – act as monitoring application that use data from the database to provide latest data/reading from the device.

Smart Water Level Detection System Device consist of controller and sensors that will record the manipulated data. The controller that being used is Raspberry Pi and sensor that being used is water sensor. Raspberry Pi being programmed by using Python language. The controller contained build in Wi-Fi modules that enable the controller to connect to the internet. The device offers the function to record all the data and determine the output. The data is then being sent to the web server (Django) via internet. User can install this device at any place that need to be measured the water level. Some of the most suitable place to install the device are places with heavy rain (flood detection), dam, beaches and etc.

Web monitoring is used as the database for the project and also used to monitor/analysis the data received by the device. It receive the data from the device periodically and stored it inside its database. The data also being used to generate graph and provides information and alerts user if the water level reach critical condition. Users can monitor the data real-time anywhere by accessing the web page. Multi device can be integrated within the same web page that enable users to install multiple device at multiple places and monitor all the data at single web page.

Smartphone apps gives users the same advantage to monitor the data by using their smartphones. The apps can gives the latest data that being requested from the database and visualize it on the smartphones. It becomes more convenient for users that used their smartphones daily.

All the component mentioned above used the concept of IoT that enable them to send/receive data from each other using the internet. The data stored at the database also can be downloaded into the computer for further analysis.

## **Scope**

Smart Water Level Detection System provide easy and efficient way to monitor and analysis the water level at places of these device being installed. The data can be monitored at either on any web browsers by accessing the web page or by using the smartphone apps. The water levels that being recorded by the device can be classified to several classes that represent different situation. The data also can be used to predict incoming flood or dam malfunction that will alert users so that safety measures can be taken.

Compared to the traditional way, users usually monitored by using estimation or on-site monitoring. These traditional data sometimes not accurate and wrong prediction might being made. The other advantage of using this system is that user can monitor and analyze the data from the device real-time anywhere there is internet connection. The device also will determine the critical water level and send alert to users either on the web page or on the smartphones apps. The data that being sent to the database (web page) will be stored and can be used to generate graph or data tabulation to prevent any related catastrophe. Another feature of the system is that the data inside the database can be downloaded into the computer for storage/analysis purpose.

## **Definition, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| Term | Definition |
| User | Someone who interacts with the application |
| Developer | Person of group that develop the application |
| Database | Collection of all the information monitored by this system |
| UML Diagram | Unified Modeling Language is a graphical visualization language. UML is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. It consists of a series of symbols and connectors that can be used to create process diagrams and is often used to model computer programs and workflows. |
| Software Requirements Specification (SRS) | A complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. |
| Software Design Description (SDD) | Software Design Description which is the complete description of the design of system. |
| IEEE | Institute of Electrical and Electronics Engineers |

## **Overview**

This document encompasses a design model with architectural, interface, component level and deployment representations. Design model will be contained in this document, which will be used as a medium for communicating software design information.

This document is divided into subsections to make it more understandable. Those are:

1. **Introduction:** Provide an overview of the application, describe the document structure and point the individual objectives.
2. **System Overview:** Contain general description about the system components.
3. **Design Consideration:** Provide the assumption made during the design process, dependencies and other constraints.
4. **System Architecture Design:** contains the most important diagrams of the document. Class diagrams, data flow diagrams and sequence diagrams of components are stated in this section. Also a brief explanation about the classes is mentioned.
5. **User Interface:** Contains the user interface design and some screenshots.
6. **Detailed Design:** Contains the detailed design issues and future works.
7. **Libraries and Tools:** Provides the libraries and tools that used.
8. **Conclusion:** Contains the conclusion of the SDD
9. **Reflection:** Provides what we learnt during completed this project related to subject.
10. **Appendixes:** Contains the script.

## **Reference Material**

[1] IEEE Recommended Practice for Software Design Descriptions

[2] AJCON Software Requirements Specifications Document, v1.0

[3] JavaML – A Markup Language for Java Sources,

[8] Java Reflection API, <http://download.oracle.com/javase/tutorial/reflect/index.html>

[6] Apache Tomcat Wikipedia Page, Wikipedia.org/Apache\_Tomcat

[7] Django tutorial, http://www.tutorialspoint.com/django/

# **System Overview**

Main objective of this project is to ensure the end user will always get the correct information before the actual announcement. Normally during rainy season, water damp tends to be overfilled, and in order to prevent from breaking the dam wall, the authorities has to take actions such as releasing the water from the dam to the nearest river or sea. This actions is most likely will causes of flood to area located close to the river or sea chosen. In this context, we designed this project in a manner stated in section 5 and 7.

General description on the system is further explained in the activity diagram below.

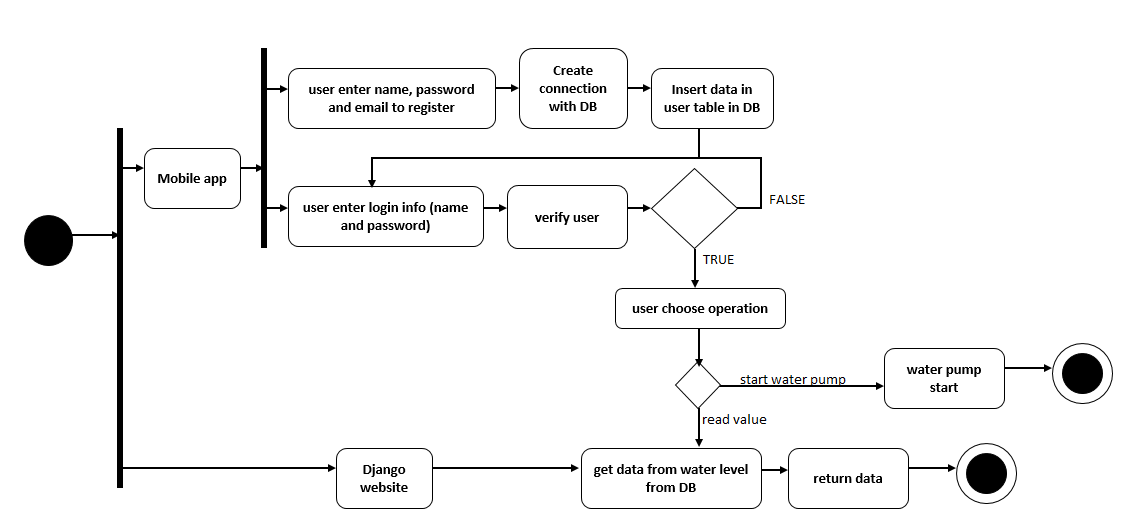


Figure 1: Activity Diagram for Smart Water Level Detection System

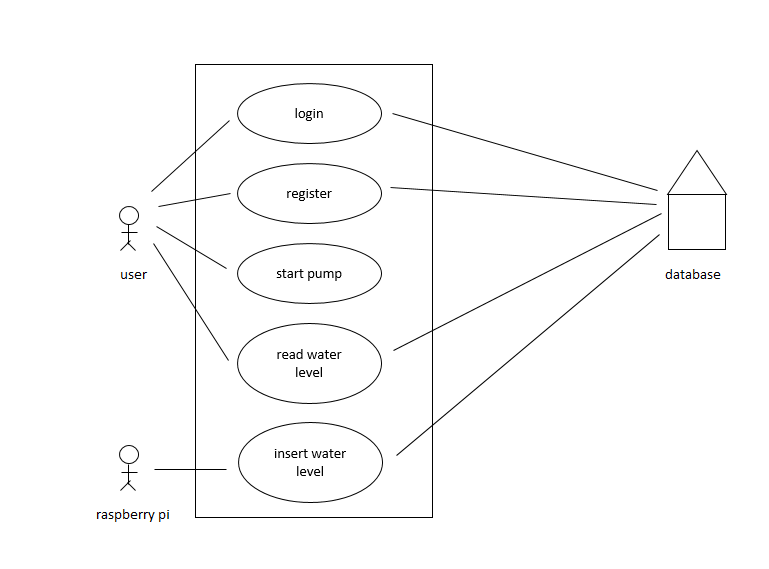


Figure 2: Class Diagram for Smart Water Level Detection System

# **Design Consideration**

## **Design Assumption, Dependencies and Constraints**

### **Design Assumption**

This project consists of three major part which are development on microprocessors, website and mobile application. With limited time constraints, some of the assumptions is made in order to ensure the project is on the right track.

* For complete design, our design assumptions can be stated as :
* This project runs on raspbian for raspberry-pi
* This project requires a power supply to run all the time
* The website can be access from anywhere without any requirement
* The android mobile app need to be installed by user
* User for mobile app need to register an account
* For final design, input of the project is current water level in the dam, user interaction on website or mobile app.
* This product can be implemented in any area that collect or stores water such as water tank or dam but requires proper precaution as the electronic circuit is not water proof.

### **Design Dependencies**

For complete design, out design dependencies can be stated as:

* The mobile application runs on android platform which is developed using Java
* The website has to be developed using python language
* The microprocessor for the project uses raspberry-pi

### **Design Constraints**

#### **Time**

As the project for course MKEL1243, this project have 8 weeks to be completed. All the members of this project has to work together to ensure the project can be done within 2 months. Agile method was used where the development of this project is performed with minimum target to be achieved each week. As the requirements is different every week, the project member will find about the latest project updates on GitHub.

#### **Performance**

Performance is the most important thing in every products, this project gives response on real time with the project maximum error of 3 seconds. The error of the reading might be not much however if the internet connection is lost, the data of water level might as well will be lost. As the server being maintained regularly, this project will work accordingly.

#### **Hardware Constraints**

##### **Raspberry Pi**

The constraint that we have when working with Raspberry Pi

1. It does not have a real-time clock (RTC) with a backup battery. The fact of missing clock can be workaround using a network time server, and most operating systems do this automatically.
2. The Raspberry Pi always boots from an SD card. It means that even a perfectly valid installation of an operating system is available on a USB stick or an external hard drive, it can’t be booted. In other words, external storage devices can be used but can’t be used to boot the Raspberry Pi.
3. It does support Bluetooth and Wi-Fi out of the box but the built in module is not working properly and needed external USB dongles. Unfortunately, most Linux distributions are still a bit picky about their hardware, so it should be first checked whether flavor of Linux supports particular device.
4. It doesn’t have built-in an Analog to Digital converter. Thus when using sensor or utilizing Raspberry Pi GPIO, especially using analog devices, external component must be used for AD conversion.
5. Its power consumption varies depending on how busy Raspberry Pi is and what peripherals are connected. Typically, the model B uses between 700-1000 mA depending on what peripherals are connected, and the model A can use as little as 500 mA with no peripherals attached. Thus, it is necessary to provide a power supply that can provide enough current to power the device plus any connected peripherals.

##### **Software constraints**

###### **Webpage Django**

As discussed from group members, Django application is used to only display a results for water tank level and the will not apply as an input to user. Therefore the main focusing constraint for Django application are:

* Medium to extract results to be display
* Type of displaying the results

From this project, the main part to handle the data is database. The database used in the project is MySQL which coming from software of Phpmyadmin. The constraint here is the data from database created from Phpmyadmin was not been able to be extract directly by entering the host, username and password such been used in MySQL. This method or extracting data directly from MySQL is also known as backdoor method. This is because the database is protected by the software. The medium used from the software is PHP file. To link Django and database is PHP file however to extract the data from PHP file will need to python script extraction coding.

There are two type of displaying data which are tabulated view and graphical view. Graphical view also consist of several types which are bar graph, pictograph, and circle graph and line graph. As the objective for Django is to display the level tank related to time and only consist of two type of data, the bar graph will be the best option. In order to display the result in graphical manners, html format will be the best option to link in to python script in Django.

###### **Smartphone Apps**

* Software must be implemented in object oriented and it must be reusable.
* System must be formed of subsystems and they should be implemented separate and understandable.
* MySQL 5.5 must be used in the database management.
* Application server is developed in java.
* Classes and their entities must be implemented according to design classes and their entities.
* Develop apps only compatible with android OS of android 4.2 jelly bean and above

#### **Design Goals and Guidelines**

##### **Portability**

This project can be implemented at any water storage systems and not limited to dam. The water level sensor will be varied based on the depth of the water storage. The water pump also can be modify to suit with the water pump installed on the water storage. The project itself is small in size and portable enough for any user to install the system themselves.

##### **Reliability**

The hardware of the project is reliable for 5 years with proper handling. As for the website and mobile app, the reliability for both of the system is failure free. The output returned by the website and mobile app will be the same as the technology developed.

##### **Correctness**

The project will run accordingly as long as all the requirements are met. The result will not depends on any environment circumstances.

# **System Architectures Design**

## **Raspberry Pi**

### **System Architecture**

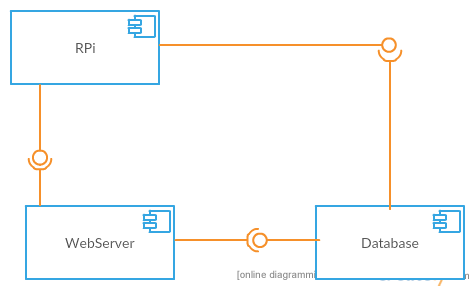


Figure 3: System Architecture for Raspberry Pi

Main concern of RPi side is to provide the sensor level value at current time and also to start and stop the pump system. For this purpose the architecture have been composed to several components: RPi Component, Webserver Component, and Database Component. Those components are interacting with each other’s. Some of them provide some interfaces to other ones, and some of them use the provided interface. Generally the interfaces provided by the other components are the methods of the classes in it. Above relations shows the relationship of all component.

### **System Description**

1. Processing Narrative for RPi system component

* This is the component that not interact directly with user. All of the component in this system is by user from external peripheral. It has simple and direct connection to serve it purposes. By this component, the main objective of creating smart water level sensor can be achieved. User can manage the system to become autonomous or manual by using this component.

1. Interface Description of RPi system component

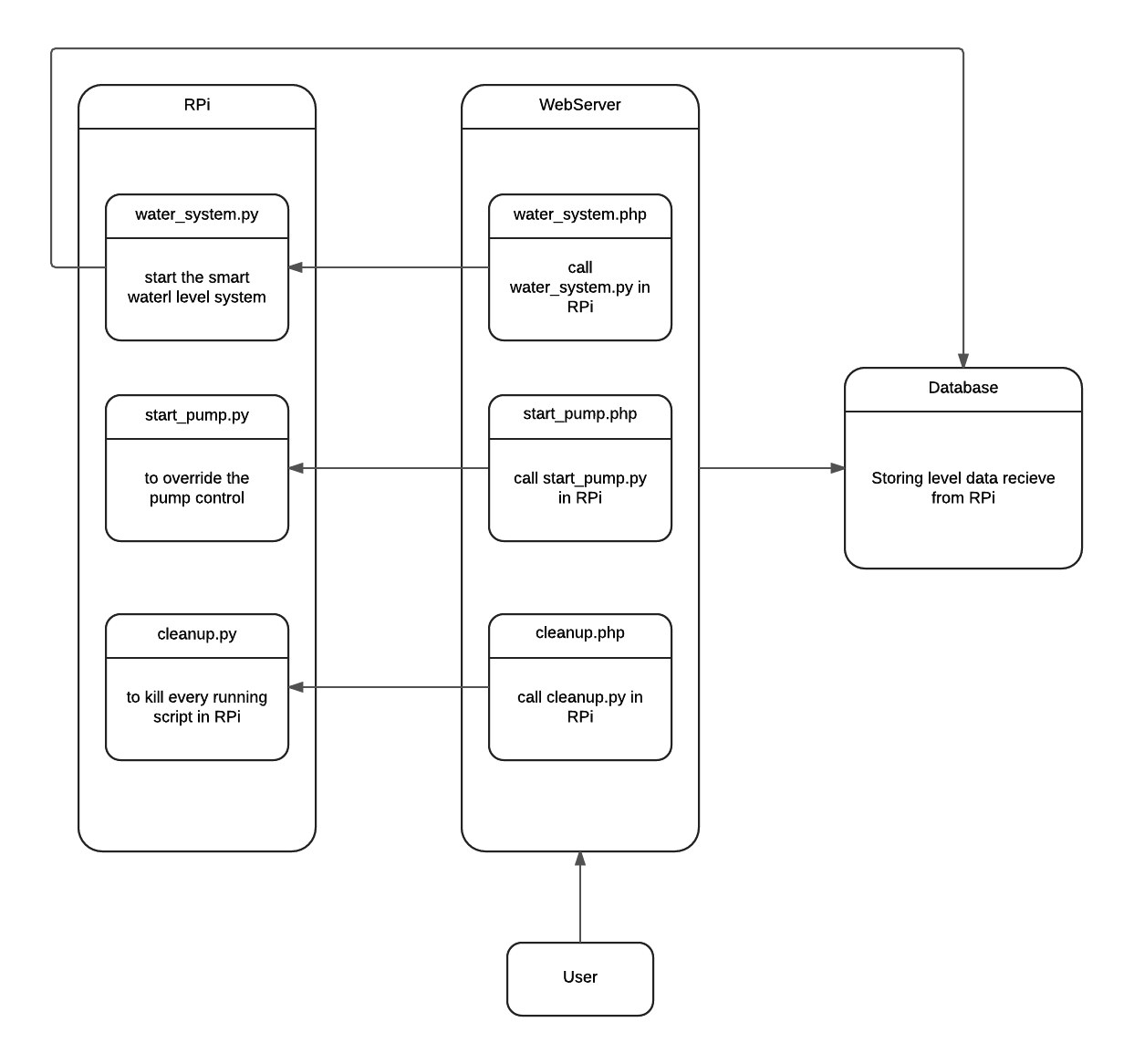


Figure 3: UML structure raspberry Pi

1. Processing detail of RPi System component

* RPi System composed by 3 component part, RPi, WebServer and Database. Each of the component indicated and described in following table:

Table 1: RPi component

|  |  |  |
| --- | --- | --- |
| File | Functions (arguments) | Purpose |
| start\_pump.py | urllib.urlopen('http://akuzul.pagekite.me/killclean.php') | Stop clean.py script via Webserver |
| urllib.urlopen('http://akuzul.pagekite.me/kill.php') | Stop all python script running inside RPi via Webserver |
| GPIO.setup(2, GPIO.OUT, initial=GPIO.LOW) | Setting RPi GPIO pin 2 as output pin |
| GPIO.output(2, GPIO.HIGH) | Passing active high to RPi GPIO pin 2 |
| water\_system.py | urllib.urlopen('http://akuzul.pagekite.me/killclean.php') | Stop clean.py script via Webserver |
| urllib.urlopen('http://akuzul.pagekite.me/killpump.php') | Stop start\_pump.py script running inside RPi via Webserver |
| GPIO.setup(2, GPIO.OUT, initial=GPIO.LOW) | Setting RPi GPIO pin 2 as output pin |
| GPIO.setup(n, GPIO.IN) | Setting RPi GPIO pin n as input pin  n= 5,6,9,10,11,12,13,14,15,16,18,19,20,21,26 |
| level=[('level','n')] | Setting level value  n= 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| level = urllib.urlencode(level) | encode URL for parameter level |
| req = urllib2.Request(url2, level) | passing encoded level to url2 |
| page=urllib2.urlopen(req).read() | read the latest passed parameter |
| except KeyboardInterrupt: | interrupt handler for water\_system.py |
| cleanup.py | urllib.urlopen('http://akuzul.pagekite.me/killpump.php') | Stop start\_pump.py script running inside RPi via Webserver |
| urllib.urlopen('http://akuzul.pagekite.me/kill.php') | Stop all python script running inside RPi via Webserver |
| GPIO.setup(2, GPIO.OUT, initial=GPIO.LOW) | Setting RPi GPIO pin 2 as output pin |
| GPIO.output(2, GPIO.LOW) | Passing low to RPi GPIO pin 2 |

Table 2: Webserver component

|  |  |  |
| --- | --- | --- |
| File | Functions (arguments) | Purpose |
| start\_pump.php | <?php  exec("sudo python /var/www/html/ start\_pump.py");?> | execute start\_pump.py script |
| water\_system.php | <?php  exec("sudo python /var/www/html/ water\_system.py");?> | execute water\_system.py script |
| cleanup.php | <?php  exec("sudo python /var/www/html/ cleanup.py ");?> | execute cleanup.py script |

## **Smartphone Apps**

### **System Architecture**

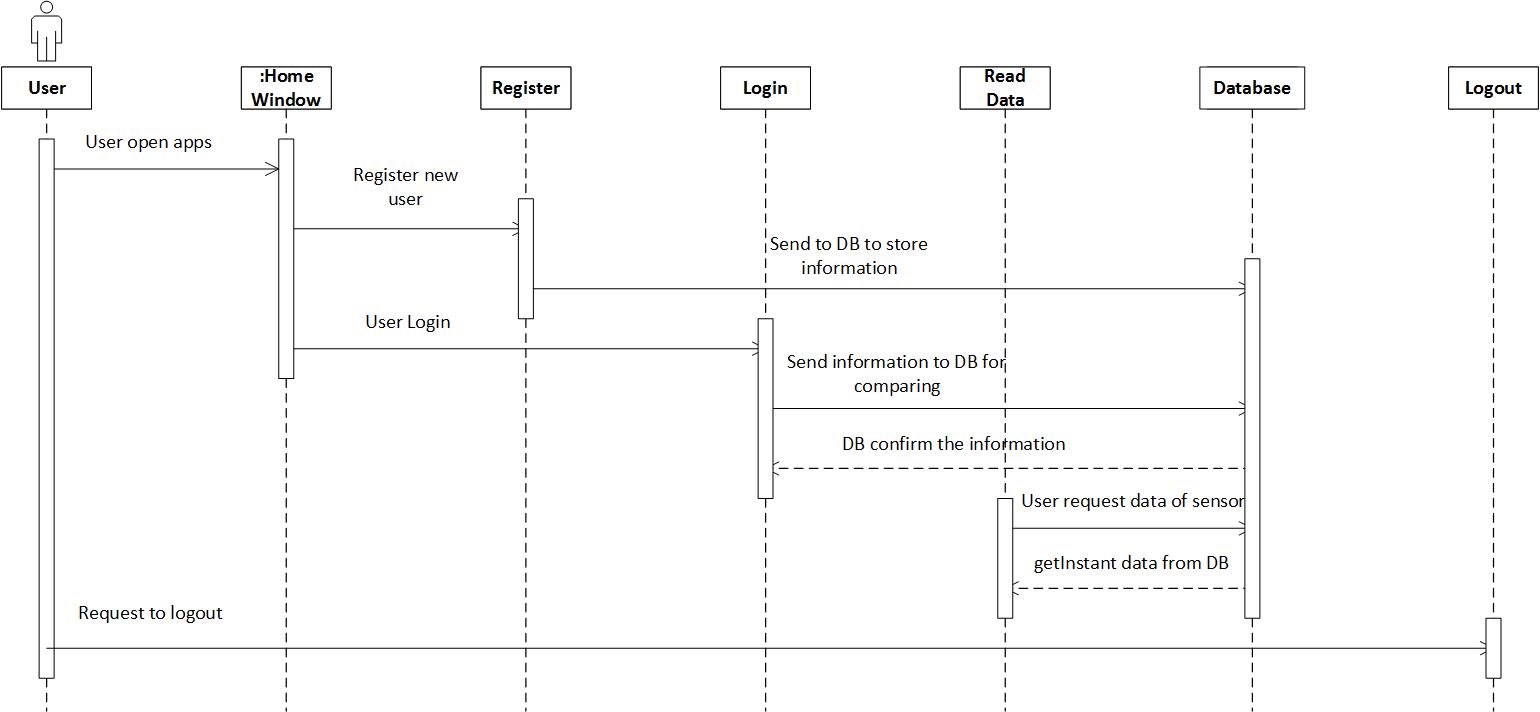


Figure 5: Sequence Diagram for Smartphone Apps

### **System Description**

Figure 5 shows the sequence diagram to show the full system architecture of Water Tank level Server System application using the android application. The flow of the diagram are, user will open the android application and it will bring user to the home page of the application. Then user will be ask to register to the system by push the register button and bring user to the register page, after finish user will automatically go to homepage that need the user to login. To login user need to fill the name and password, then system will compare the data insert with the database that store the user information. If correct user will go to the data page that will show user name and also the data from the sensor.

On the data page it have two button which are get data button and logout button. To get the data of the water level user need to push the get data button and system will request from the database to get the water level data and also water level time taken. After user are satisfied with the data, user can go to the homepage by push the logout button and then repeat the same sequence for the system. But for user that has register previously, they does not need to register again only need to login because their data had been store in the database.

## **Webpage Django**

### **System Architecture**



Figure 6: Sequence Diagram for Webpage Django

Figure 6 shows the sequence diagram to show the full system architecture of Water Tank level Server System application using Django webpage application. The basic sequence diagram will not include additional sequence diagram which in yellow box as shown in Figure 6. The additional sequence is controlled by raspberry Pi which will directly will update the PHP file based on actual water tank level. Thus every time the user insert webpage link, it will get the update result which are water level tank for each of time taken.

### **System Description**

The flow of Django application will start by user insert the server link. Figure 4.2 shows the activity diagram which provide better understanding of Django application on the water tank level server system.



Figure 7: Activity Diagram

From Figure 7, the system will be initiate once the user enter the Django link server. Then the server will be verified before reading the php file. The php file will contain the data of water tank level and time taken for each of the captured water level. Figure 8 below shows the example of data captured in PHP file. The, Django system will process the results into graphical view which been describe using html format. Lastly the graph will be presented in the webpage specified link by user.

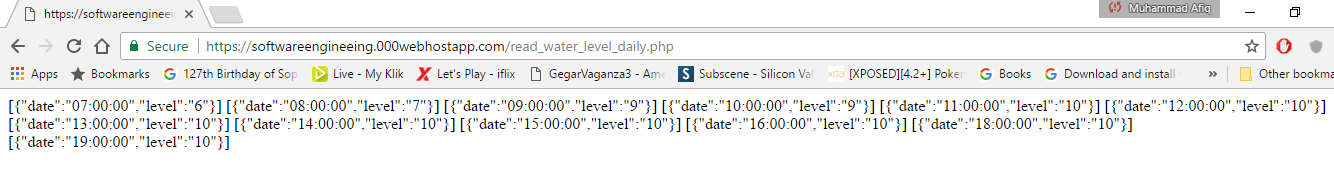


Figure 8: PHP file content

## **Database**

### **System Architecture**

The UML sequence diagram for the interaction between input and database is shown below. The actor’s interaction can be either from the raspberry pi, website or mobile application. The request is responded by specific php files in order to handle the request.

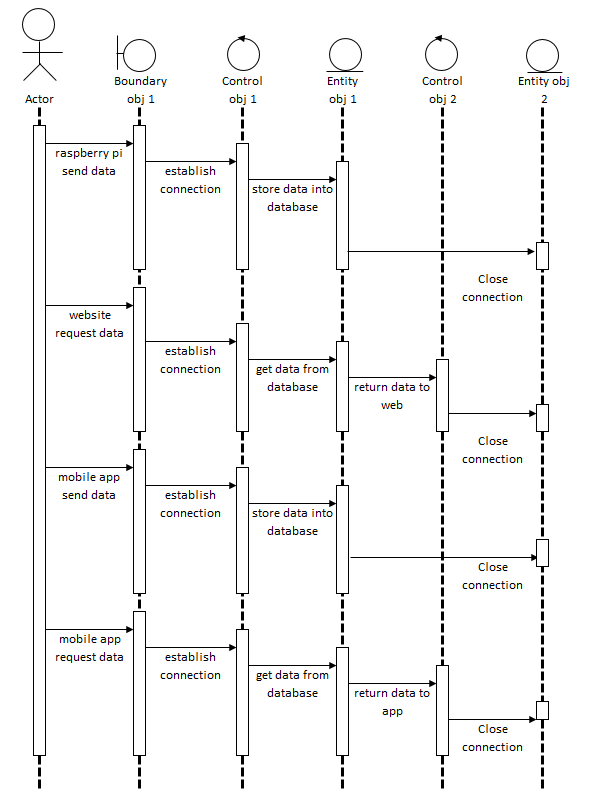


Figure 9: Sequence Diagram for Database

### **System Description**

From the UML in section d (i), the actor can send 3 different type of interaction which the first is user send interaction from raspberry pi with a set of data. As raspberry pi called the specific php file, the php will create connection to the database and request to store information in the database. Once the data is stored, the php will directly close the connection and it will not return any data to raspberry pi.

The second interaction is when the actor request information through the website or Django link. The php file will be called to create the connection. One the connection is secured, the database will returned data in form of array, consisting all the water level data for the respective date and close the connection afterwards.

The final interaction is actor request information through mobile application. The php will create connection and then get all the information needed by the application. If register button is clicked, it will call a php to create connection to database before storing the data. For login the user data will be requested from the database while for the water level system water level data will be requested.

# **User Interface Design**

## **Overview of User Interface**

### **Webpage Django**

Figure 10 below shows the overview of dataflow diagram for context diagram which will provide top overview of user interface. Figure 11 shows the level 0 which provide the database used which coming from PHP file. The Django will have one page which is main page just to provide the water level tank versus time graph.

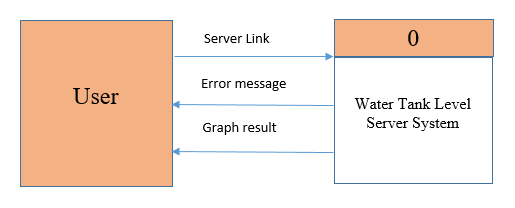


Figure 10: Context Diagram

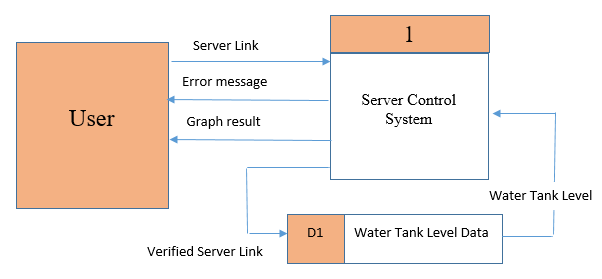


Figure 11: Level 0 Diagram

### **Smartphone Apps**

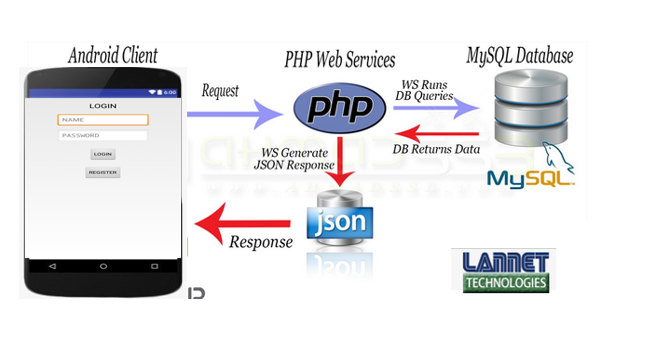


Figure 12: Overview Diagram for Smartphone Apps

Figure above shows the overview diagram of how the smartphone apps works. Basically the apps have three main side which are android client, PHP web Services and MySQL database. The system work by user insert the input on the smartphone interface. Then by pressing the required button on the smartphone, the android client side will send a request to PHP web services. At the web service, it will run the web service queries to the MySQL database. When the queries completed, the database will return the requested data to PHP web service. When web service receive the data from database it then will generate JavaScript Object Notation (JSON) response. The reason we are using JSON is because it is a lightweight data-interchange format. It also easy for humans to read and write and for machines to parse and generate it. After the JSON get generated, then the web service will send the response back to android client where it later can be display on the android interface.

## **Interface Screens**

### **Webpage Django**

Figure 13 below present the result of water level tank for each time taken in graphical manners. The Django application only provide update results which not include user interface on the Django application. The data will be coming from PHP file.

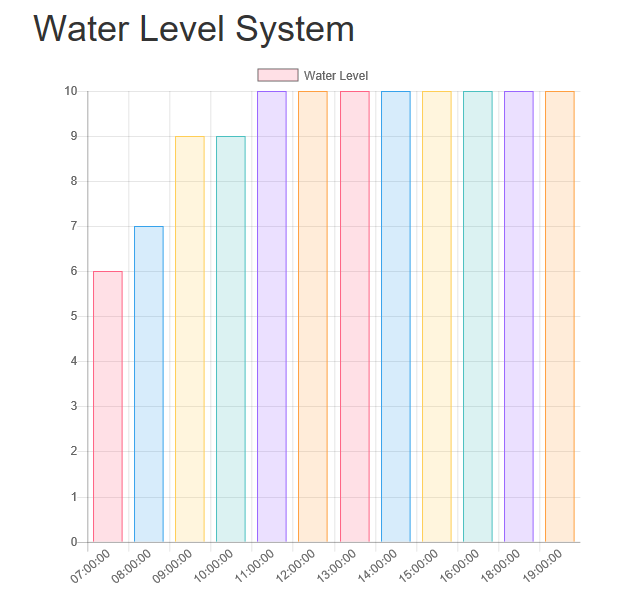


Figure 13: Water Tank Level vs Time

Based on Figure 13, the water level keep increase until the level reach 10L. This is because the water level was set to 10 which will be the limit of the tank system. The larger the data extracted the smaller graph size which will provide better explanation in term of relationship of water level tank and time taken.

### **Smartphone Apps**

For smartphone apps, we have three main interface which acts as interface between user and the application. In this apps we have login, register and Homepage screen interface. These there interface are elaborated below.

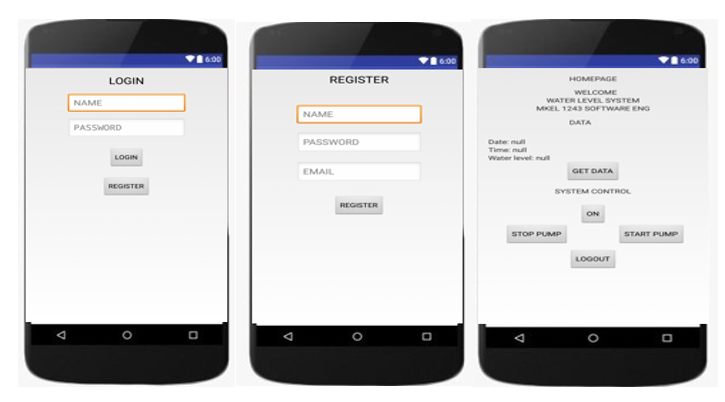


Figure 14: Smartphone Apps Interface

The login interface acts as medium before the user are authorize to enter to the main system. On this interface, user only need to specify his/her name and password correctly. If the text field is wrong, then the user need to try again until the registered username and password are correct. For register interface, it is a medium for a user to register in order to get access to the main system. It can be done by pressing the register button on the login interface. On the register interface, a user need to insert his/her name, password and email. Then all the information will be stored at the database which later need to be used as comparison purposes when the user need to login into the system.

Once the user has a registered username and password, he/she can login into the system. When the login is successful, the system will direct user to the Homepage interface where it is an interface to all the main function are located. It is where the user can retrieve the data of real time water level system and display the latest reading on the screen. The data retrieve are in the form of the latest date, time and water level. Besides that, homepage interface have a system control system interface as well where user can control the water level system manually by pressing the system control button which one of each button has their own function.

For login and register interface, all the text field has a hint for what the user need to insert so that the user does not get confused and can avoid the mistake of inserting the wrong input. However, if the user insert a wrong input, the text field will automatically clear all the wrong input and the user need to try again until the required text field’s input is correct.

## **UI Functionality**

### **Webpage Django**

The application for Django does not have user interface with the screen. The user will need to enter the link such as below:

<http://127.0.0.1:8000>

There are also additional step to perform Django run on command window. This will make sure the Django application will always enable and the link provided able to be recognized by Django. Figure 15 below shows the result by running the command below to enable Django server:

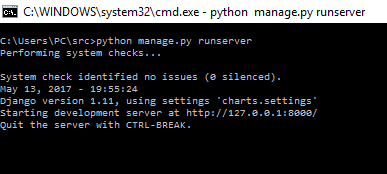


Figure 15: Enable Django run server on windows command

### **Smartphone Apps**

1. Login interface button

* Basically the login interface only have two button which are “LOGIN” and “REGISTER” button. Like what the button name tell us, the “LOGIN” button acts as login button to the main system after the user has input the registered username and password to the specify text field. If the input is correct and same with the database user info, then the user will directed to the main system which is Homepage interface screen. However, if the user does not has the registered username and password yet, he/she need to register first by pressing the “REGISTER” button where the user will be directed to register interface

1. Register interface

* Register interface only have a “REGISTER” button only. When the user already insert all the necessary text field such as username. Password and email, then the user can do the registration by pressing the “REGISTER” button. By doing that, all the user’s information will be send and stored at the database.

1. Homepage interface

* Homepage interface is where all the main function are located. It consist of “GET VALUE” button and all the system control system button such as “ON”, “STOP PUMP”, “START PUMP”. Besides that, it also a “LOGOUT” button where the user can logout from the main system and back to the login interface screen. The “GET VALUE” button acts as main button to retrieve the data in the form of data, time and water level from the water level system at the database. Every time “GET VALUE” get pressed by the user, the latest reading will keep retrieve from the database. The “ON” button acts as button to turn on the main system. “START PUMP” acts as a button to manually start the pump which can be done by a request from the user because by default the water level system’s pump only get started when the water level reach level 7 where it will start to drain the water on the container until it reach below level 7. “STOP PUMP” acts as a button to manually stop the pump when the user manually start the pump or when the pump already start to switch on by itself automatically. However, the user must be aware that by stopping the pump, the water will overflow when it reach maximum level of the container. So “STOP PUMP” button must be used with caution.

# **Detailed Design**

## **UI Components**

### **Webpage Django**

1. **Command Line : to start Django**

**Classification:** Package (Django)

**Definition:** Purpose of this package is to hold the classes that is used interaction with all classes.

**Responsibility:** This class is responsible from all interaction with user. Responsibility this package will be explained in detail by responsibilities of main window of web server, chart, dataset and all the code or classes.

**Uses/interactions:** This component uses interface for all python script to run server for web framework

**Processing:** Component initiates the system and get prepared everything for the user to let start the project of create basic web framework library.

**Interface/Exports:** the set of service provided by this component is specified the web framework library, or to start the Django framework.

1. **Application Manager Class (manage.py)**

**Classification:** Class

**Definition:** purpose of this class to initiate run of project

**Responsibilities:** this class main function of the project. When main is called the system, it is responsible from creating the framework of web server and initiating the system.

**Constraints:** there no time, memory, processor limitation. To make the class active, user need to execute or run project.

**Uses/Interactions:** this class only interaction with all script of python in Django framework

**Processing:** The manage.py class simply instantiates a Main Function to start Django framework, with command line to start project and also to run server. After execute start project using command line , the Django or project framework will created with some scripts like view.py, urls.py, setting.py, and wsgi.py.

**Interface/Exports:** this class does not provide interface or export anything to any other component.

1. **Home View Application**

**Classification**: Class (HomeView() in view.py)

**Definition:** Purpose this class to initiate the view at home page of web server.

**Responsibility:** Home View Class is API to represent the chart in home page of web server. On this class will called the chart.html to create bar chart for our project.

**Uses/interaction:** this class will interaction with html code. (chart.html)

**Processing:** Home View class simple instantiates a Home page of web server and called the chart.html to create bar chart in homepages.

**Interface/Export:** This function will export the django view to create the view of django framework.

1. **Chart Data (API VIEW) and Dataset**

**Classification:** Class (CharData(APIView) in view.py)

**Definition:** Purpose to extract the dataset and send the data information to chart.html

**Responsibility:** This class able to translate the dataset in php format from URL given. Into the text of dataset (dataset.txt). After that, this class extracted dataset.txt to translate array format. The dataset will split by date and water level. The array of dataset sent the information to template (chart.html) to represent into bar chart in homepages of webserver.

**Uses/interaction:** This class have interaction with dataset in php format and interact with template (chart.html)

**Processing:** When system is run, this class will interact with dataset and also the chart.html.

**Interface/Export:** This class export the array of dataset to template (chart.html). The interface API View used, it is import from rest framework view library.

1. **Bar Chart template (chart.html)**

**Classification:** Function (chart.html)

**Definition:** Purpose this function is to display the bar chart of dataset (date and level data information)

**Responsibilities:** This function display the dataset into bar chart format in home pages of our webserver.

**Uses/interactions:** This function interacted with the chart data class in python scripts(view.py) to collect the array of data to represent on Bar Chart in webserver

**Processing:** This function run and the dataset with different colour for every dataset.

**Interface/Export:** this function will export the dataset in Bar Chart format to home pages of our web server. And this function used template of bootstrap as UI interface. The template of bootstrap provide the chart style/format (bar chart for our project).

### **Smartphone Apps**

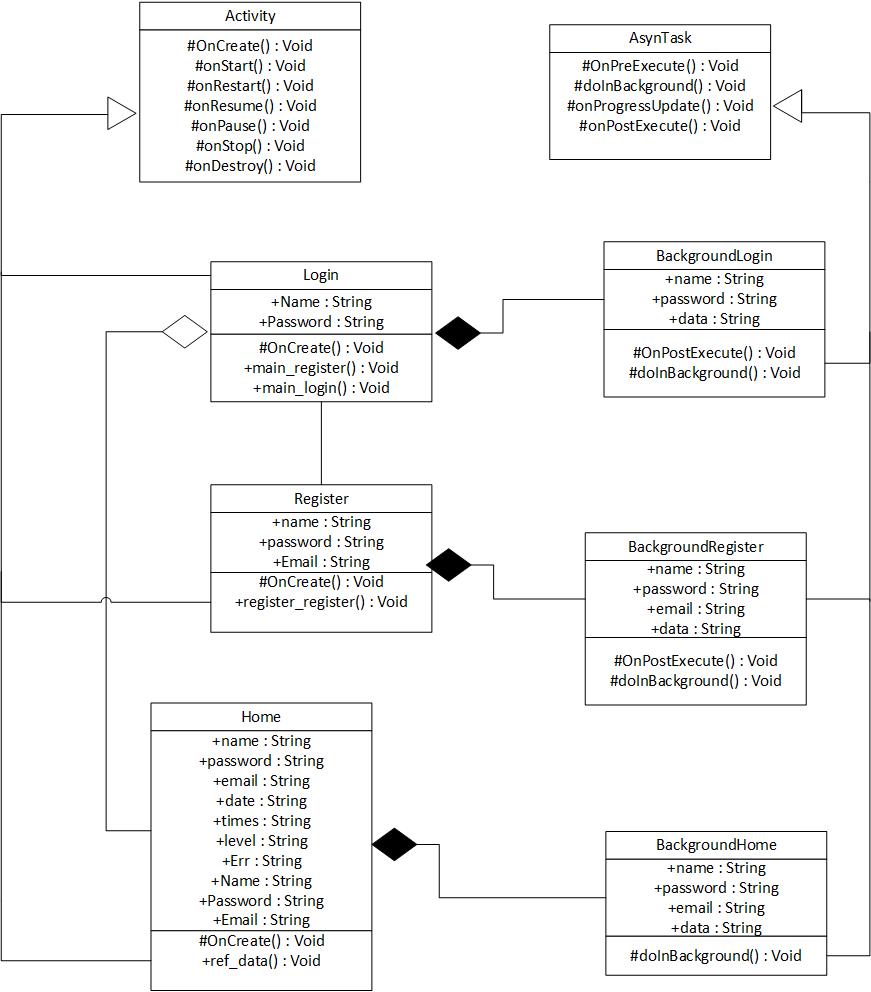


Figure 16: Smartphone Interface

## **Webserver/Database**

1. **Init.php**

Classification: File  
Definition: Purpose of this file is to initialize connection between the database and other devices such as mobile phone.   
Constraints: There is no time, memory or process limitation.  
Uses/Interactions: This function uses MainWindow class interface in order to call its initComponents() method.  
Processing: When smartphone or website requires a connection to the database, this file will be called

1. **register.php**

Classification: File  
Definition: Purpose of this file is for mobile app user to register their account.  
Constraints: There is no time, memory or process limitation.  
Uses/Interactions: This function uses MainWindow class interface in order to call its  
initComponents() method.  
Processing: When user click on register button, the php will be called to register the information into the database

1. **login.php**

Classification: File  
Definition: Purpose of this file is for mobile app user to login into their account  
Constraints: There is no time, memory or process limitation.  
Uses/Interactions: This file will be called when user click on login button in the mobile app.  
Processing: When the file is called, it will compare user’s input on mobile app login page with the database information. If the information match with each other, user will be logged into their profile, otherwise they have to give correct username and password.

1. **Insert\_water\_level.php**

Classification: File  
Definition: Purpose of this file is to update water level of the dam in the database  
Constraints: There is no time, memory or process limitation.  
Uses/Interactions: This file will call init.php file to establish connection and insert   
Processing: When the system is turned on, the raspberry pi will keep sending actual time value for the dam water level to the database through the php file in the web server.

1. **read\_water\_level\_daily.php**

Classification: File  
Definition: Purpose of this file is to get values for water level in the dam on the actual date  
Constraints: There is no time, memory or process limitation.  
Uses/Interactions: This file is called by website to get all the reading for water level of current date  
Processing: When the website is opened by user, it will call this php file which then establish connection using init.php. The php file will return all data with the same date as current date in array form to the the website before it is converted into bar chart.

1. **Read\_water\_level\_hourly.php**

Classification: Function  
Definition: Purpose of this file is to return the latest water level reading  
Constraints: There is no time, memory or process limitation.  
Uses/Interactions: This file is called when user successfully log into their account on mobile app.  
Processing: After user log into their account, they can choose to get the latest water level value using a button. The php file will be called and connection is established using init.php. This php file is then will returned the latest reading stored in the database to the mobile app.

# **Libraries and Tools**

## **Java (Android Studio)**

For this project, we are using android studio to develop the android apps. Due to the advantages of built in emulator inside the software, we are able to run the apps simulation with actual behavior by using the emulator. Besides that, with a variety features and function in android studio software has provide us many option to setup the layout interface.

## **Python**

### **Django**

|  |  |
| --- | --- |
| Django | Install in window.  Download here : <https://www.djangoproject.com/start/>  Version: 1.11.12 |
| Python | Used python2.7 in window:  Library : C:\Python27 |
| Linux | Used command prompt in window |

### **Raspberry Pi**

1. **requests 2.17.3**

Requests is the only Non-GMOHTTP library for Python, safe for human consumption. Requests allows you to send organic, grass-fed HTTP/1.1 requests, without the need for manual labor. There’s no need to manually add query strings to your URLs, or to form-encode your POST data. Keep-alive and HTTP connection pooling are 100% automatic. Beware that recreational use of the Python standard library for HTTP may result in dangerous side-effects, including: security vulnerabilities, verbose code, reinventing the wheel, constantly reading documentation, depression, headaches, or even death. Requests is ready for today’s web.

* International Domains and URLs
* Keep-Alive & Connection Pooling
* Sessions with Cookie Persistence
* Browser-style SSL Verification
* Basic/Digest Authentication
* Elegant Key/Value Cookies
* Automatic Decompression
* Automatic Content Decoding
* Unicode Response Bodies
* Multipart File Uploads
* HTTP(S) Proxy Support
* Connection Timeouts
* Streaming Downloads
* .netrc Support
* Chunked Requests

Requests officially supports Python 2.6–2.7 & 3.3–3.7, and runs great on PyPy.

1. **urllib/urllib2**

urllib/urllib2 is a Python module that can be used for fetching URLs, allows you access websites via your program. It defines functions and classes to help with URL actions (basic and digest authentication, redirections, cookies, etc). Through urllib, you can access websites, download data, parse data, modify your headers, and do any GET and POST requests you might need to do. Both modules can do URL request related stuff but they also have differently functionality. Some extra function are urllib2 can accept a Request object to set the headers for a URL request, meanwhile urllib provides the urlencode method which is used for the generation of GET query strings. This extra function of both module complement each other and because of that urllib and urllib2 are often used together.

1. **RPi.GPIO**

The RPi.GPIO python module offers easy access to the general purpose IO pins on the Raspberry Pi. With this module, IO pin number declaration can be done by choosing pin-numbering scheme whether using GPIO.BOARD – Board numbering scheme or GPIO.BCM – Broadcom chip-specific pin numbers. Then user can define the pin mode. Pin mode declaration can be either input or output. Furthermore, extension from pin mode declaration, user also can choose the pin mode to tie to pull up or down resistor. This make possible for the pin to act in state of active high or active low. The other function that this module offer is to add delay inside the script and to clean up every declaration that have been done on the GPIO.

## **Database**

* 1. **SQL language in MySQL database**

This project implemented MySQL for the database since this product is implementing IoT. MySQL have more flexibility and more space to store data compared to SQLite database. The language used in the database system is called structured query language (SQL) and they are used to perform activity such as create the database, create the table, insert the data in the table and many more. Some of the SQL command is shared below.

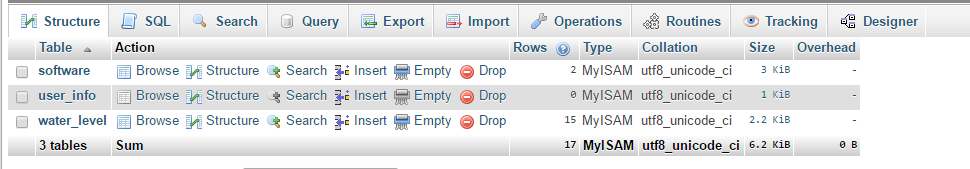
INSERT INTO `water\_level`(`id`, `now\_date`, `now\_time`, `level`) VALUES ([1],[2017-03-30],[12:10:43],[1])

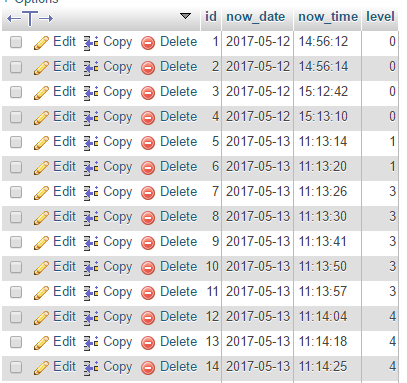
UPDATE `water\_level` SET `level`=[4] WHERE `id`==[43]

SELECT `now\_date`, `now\_time`, `level` FROM `water\_level` WHERE `id`==[40]

* + 1. phpMyAdmin

phpMyAdmin is an administrative tools to manage MySQL database. This tool gives an interface for the database which make it easier for the adminstrator to maintain the database. Other than using SQL command to perform changes on the database, phpMyAdmin allow users to do the same operation on the UI form. The database and table structure created on phpMyAdmin is shown below.





* + 1. php

php is a scripting language which can be embedded into HTML markup. This language has been used to connect between raspberry-pi, website and also mobile application with the database server. The reason php was used instead of direct interaction with the database is to prevent back door form occurring. The php file with the respective script is stored in the webserver where the database resides. The following shows some of the code for the php file.

<?php

error\_reporting(0);

$db\_name = "id1489556\_android\_database";

$mysql\_user = "id1489556\_android";

$mysql\_pass = "software";

$server\_name = "localhost";

$con = mysqli\_connect($server\_name, $mysql\_user, $mysql\_pass, $db\_name);

if(!$con){

echo '{"message":"Unable to connect to the database."}';

}

<?php

error\_reporting(0);

require "init.php";

$level = $\_POST["level"]; //water\_level

date\_default\_timezone\_set("Asia/Kuala\_Lumpur");

$time = date("H:i:s");

$sql = "INSERT INTO water\_level (now\_date, now\_time, level)VALUES (now(), '".$time."', '".$level."')";

if(!mysqli\_query($con, $sql)){

echo '{"message":"Unable to save the data to the database."}';}

else{echo '{"message":"Table updated"}';}

$conn->close();

?>

# **Conclusion**

In this document, design considerations for project Smart Water Level Detection System were dealt with. How our system work, how our system was decomposed, how these components work, their design architecture and connections, data design and flow were stated both by UML diagrams an by explanations. Moreover, user interactions were determined through user interfaces design. Libraries and tools which will be used during system development and operation were presented.

# **Reflection**

1. Farah

From this project I’ve learnt on the working operation for the database together with external devices website, mobile application and raspberry pi. The SQL command that was used in this project has open my eyes on how the IoT works. I’ve also learnt that there is a language called php that is widely used for web development and also can be used to create connection between website and external devices. This project make me understand further about how World Wide Web (WWW) operated and how we can use the information to develop Internet of Things project.

1. Atif

In this project I was assigned to handle Django webpage server. Even though the task assign to our team is to display the result from database, we put a lot of time to process the data. I learned a lot from this project on how to create database which I already created in Linux environment. I am practically used the MySQL command to handle, insert and extract the data. The task assigned to us to fully utilize python script however because we are using database from other software, directly connection to the server using python will not work. I tried that with database from Linux and it works however it is from Intel database which are confidential. I am still new to python thus a lot of python script knowledge I am able to gain.

The knowledge of extracting the data from PHP file and understanding class used in Django will be my key of learning python script. As for team perspective, we had a great team work in term of execution. Once the due date set, all of us will provide the works as planned in sprint. Communication through email and WhatsApp are sufficient enough for us as each team understand the whole concept very well. The most important part on this project for me is database team which connecting other team and provide input needed promptly. The database team will the first line support for our project when dealing with data issue or server issue.

1. Wan Zahiruddin:

In this project, I learn the how to planning my work properly with team. We have sync up meeting to achieve our objective of project. In this project, I’m involved with Django web framework, working together with Atif. My scope of work, setup Django framework and create the chart to display into webpages and Atif more on database interaction between Django and php database. And we combine our task with other teammate to execute full flow of project and tuning up if have issue on our side.

At the end of project, we know the requirement and specification if important before start the project so that, we can archive our objective properly and work together as team. The top flowchart or dataflow graph our project important to make our design simply and easy to debug. Thank you to lecturer and team. Nice to know you all.

1. Fahmi

During completed this project, there are many thing can be learnt. The knowledge of software engineering that was learnt during class was very helpful in order to develop this project. All the knowledge gain during the lectures and through self-studies can be implement during this project as practical part of study. In this project, I was assigned to understanding the operation of Django and their interface. Besides, I need to study and learn about Python. Python is new language to me. In this project, Django and Raspberry Pi need to use python.

I also assign to identify what are needed to complete the Software Description Design report (SDD). In order to make SDD, it need all knowledge of software engineering that learn during class. Use case, class diagram, ER diagram, UML and sequence diagram were use in SDD. Other that, software engineering management will be make the system more portable, maintainable and reliable. All of this to make sure the system or product is in good condition and easy to maintenance before hand over to customer. All the experience during completed this project is very valuable to innovate the new things in the future.

1. Zulfadli

This course was a really new experience for me both from learning point of view and performance wise. The practical part of the course was really helpful where ones get a chance to implement all the knowledge gain during the lectures and through self-studies. I have learnt that innovation requires collaboration, creativity, practical implementation and added value to the product. And this task becomes much easier if you have an interdisciplinary team working for a unified goal. I have truly learned a systemic approach to designing a product, that how to start from scratch and nothing to build a customer oriented or to introduce a new product into the market, how to work in teams for short span of time, how to manage and divide tasks within the group, which project management methodology to apply depending upon the nature of the project, how to filter the needs or requirements of the products, how to find basic knowledge of what already is out there available in the market. How to be creative during the synthesis of concepts for your topic or project.

I learnt that most important part of the planning stage is building project team. Generally try to establish your team as soon as possible. Identifying one or two people even during the initial stages is also possible sometimes. Appointing the team early get the most out of their ownership to the project, and maximizes what they can contribute towards the product development. The first step in building an effective project team is to create a resource plan. A resource plan requires you to understand and identify the work to be done and the human skills required to complete it. An initial plan is often a high-level outline and will be refined as you break down into parts the whole of your work.

Communication is an essential part of team work. Without successful communication, it is very difficult to achieve the desired result. When information is shared effectively, the workload is divided between the team members and task at hand becomes easier for all. With so many available social apps these days in the market it’s really easier for the teams to get started and have effective communication right way. It is essential that team communication occurs throughout the project in order to minimize confusion and unnecessary delays. Team members more freely share their ideas, thoughts and opinions, thus offering additional opportunities for innovation and creativity. Messages to be communicated become simplified when there is effective communication. Chances for misunderstandings are minimized, if not altogether excluded. Team attention remains on the project, completing tasks and such activities, and whatever energy is necessary in the best interest of the project.

Besides that, I also learnt that time management is play an important role in team project. Being an engineering student itself, I tend to make everything during the last hour because I need a drive and motivation to complete the task. But throughout this project, I have been exposed to some sort of management method such as agile or scrum, and I find it useful to adopt that kind of method into my daily routine to complete my task. This method allowed me to set a target for myself and accomplish everything within the time range, therefore it is very helpful.

After all said and done, I believe that it so much things to learn and a room for me to grow. Therefore I hope that in future when I involve in this kind of project, I applied every knowledge that I have gain during accomplishing this project to be a better team member in every aspect that the team needed me to. Like Tupac Amaru Shakur said “***You either evolve or you disappear”***.

1. Zulzhafri

      Throughout the project progress, I have learnt many things that is related to software engineering. The task that was assigned to me was to design and configure the Raspberry Pi as the controller for the system. The capability of Raspberry Pi proves that this controller suitable to use in any IoT project because the various functions of the board. Moreover, the board was built in wireless module. The board also easy to be used as the Raspberry Pi provide open source library that can be used by all people. Based on the project done, the controller manage to collect the data that is the water level and sent it to the database. To control the system, external control to the smartphone apps was successfully configured. There are three mode that are start system, start pump and stop system. When the system does not need to be used, users can stop the system to enable idle mode.

     The knowledge of software engineering was successfully implemented inside the project. The usage of use case, class diagram, sequence diagram, ER diagram and UML did helps in creating project that are maintainable, improved security, reliable and portable. Besides that, the benefit of using the knowledge was shown in the document SRS and SDD. As an engineer and programmer, software engineering is important in order to produce a good project or product. There are significant improvement can be made to the project by other researcher that interested in this project. Due to time constraint and fund, we manage to complete the project with highest efficiency with given constraints.

1. Syahmi

Throughout the project, there are a lot of new experience and skill that I have learn especially in term of software development and management. Working with many people in one project does not an easy task to do unless if everyone has practiced software engineering skillset in their working routine because every task given is related with the software planning, and how we do the task efficiently among our team members. So it is importance to have these three main criteria which are systematic, disciplined and quantifiable approach to make sure all the planning and development achieve the required project’s objective for every milestone. Besides that, in this project we have encounter quite a few problematic problem especially when integrating each part of the submodule projects into one system. However, with the teamwork and discussion approach we successfully solve the problem with an efficient effort.

1. Nizam

In this software engineering subject had a lot of topic that can make me a good engineer. This subject was to produce an engineer that can manage a project with a team within the dateline. With this subject I had learn on how to manage my time to follow the dateline and also communicate with other teammate in order to sync with the project update.

I had learn a lot of topic that relate to my daily work environment, such as how to create solution for the problem that I face, write the flow on how to solve the problem. In this subject it has one big assignment that need me to tag team with other seven people to complete this assignment. The assignment was about to create IOT system that useful to other people. As a team we decide to create a system that can monitor water level and control the water pump.

In this big team of eight, we had divide in several small group to manage four part of the assignment. The part consist were database, android application, Django webserver and hardware. I had been assigned to do the android application for the IOT system. This was not new to me but I need to do the android application into another level. Because before this I just use appinventor to create my android application, but now it need to be create with another platform and the platform that I used was android studio. Android studio was a new thing for me and I need to study on how to use it. I had some obstacle because I do not have compatible laptop that can run the android studio and I need to find new laptop. This take me few week to find new laptop and finally I manage to borrow from my friend. I manage to finish creating the android application within three week time by other teammate help.

So in final week before presentation we manage to sync and combine all part to create a complete system without fatal error. During completion of my part, many things that I learn such as time management, communication skills, good programming style and many more.What can I conclude, this subject and assignment had teach me a lot of skill in creating high quality solution.

# **Appendixes**

## **Scripts**

https://github.com/SoftwareEgineeringWaterDetectionSystem/Smart\_Water\_Level\_Detection\_System

### **Smartphone Apps**

* Login.java (131 lines)

|  |
| --- |
| package com.example.nano\_android;  import android.app.Activity;  import android.content.Context;  import android.content.Intent;  import android.os.AsyncTask;  import android.os.Bundle;  import android.support.v7.app.AlertDialog;  import android.view.View;  import android.widget.EditText;  import org.json.\*;  import java.io.\*;  import java.net.\*;  public class login extends Activity {  EditText name, password;  String Name, Password;  String Names = null, Passwords = null;  Context ctx=this;  String NAME=null, PASSWORD=null, EMAIL=null;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(com.example.nano\_android.R.layout.login);  name = (EditText) findViewById(com.example.nano\_android.R.id.main\_name);  password = (EditText) findViewById(com.example.nano\_android.R.id.main\_password);  }  public void main\_register(View v){  startActivity(new Intent(this,register.class));  }  public void main\_login(View v){  Name = name.getText().toString();  Password = password.getText().toString();  BackGround b = new BackGround();  b.execute(Name, Password);  }  class BackGround extends AsyncTask<String, String, String> {  @Override  protected String doInBackground(String... params) {  String name = params[0];  String password = params[1];  String data="";  int tmp;  try {  URL url = new URL("https://softwareengineeing.000webhostapp.com/login.php");  String urlParams = "name="+name+"&password="+password;  HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();  httpURLConnection.setDoOutput(true);  OutputStream os = httpURLConnection.getOutputStream();  os.write(urlParams.getBytes());  os.flush();  os.close();  InputStream is = httpURLConnection.getInputStream();  while((tmp=is.read())!=-1){  data+= (char)tmp;  }  is.close();  httpURLConnection.disconnect();  return data;  } catch (MalformedURLException e) {  e.printStackTrace();  return "Exception: "+e.getMessage();  } catch (IOException e) {  e.printStackTrace();  return "Exception: "+e.getMessage();  }  }  @Override  protected void onPostExecute(String s) {  String err=null;  try {  System.out.println("start read");  JSONObject root = new JSONObject(s);  JSONObject user\_data = root.getJSONObject("user");  NAME = user\_data.getString("name");  PASSWORD = user\_data.getString("password");  EMAIL = user\_data.getString("email");  if ((PASSWORD.equalsIgnoreCase(Password)) && (NAME.equalsIgnoreCase(Name))){  System.out.println("Im in");  Intent i = new Intent(ctx, home.class);  i.putExtra("name", NAME);  i.putExtra("password", PASSWORD);  i.putExtra("email", EMAIL);  // i.putExtra("err", err);  finish();  startActivity(i);  } else {  Intent i = new Intent(ctx, login.class);  name.setText("");  password.setText("");  startActivity(i);  }  }catch (JSONException e) {  e.printStackTrace();  //err = "Exception: "+e.getMessage();  }  }  } |

* Register.java ( 93 lines)

|  |
| --- |
| package com.example.nano\_android;  import android.R;  import android.app.Activity;  import android.content.Context;  import android.content.Intent;  import android.os.AsyncTask;  import android.os.Bundle;  import android.view.View;  import android.widget.Button;  import android.widget.EditText;  import android.widget.Toast;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;  import java.net.HttpURLConnection;  import java.net.MalformedURLException;  import java.net.URL;  public class register extends Activity {  EditText name, password, email;  String Name, Password, Email;  Context ctx=this;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(com.example.nano\_android.R.layout.register);  name = (EditText) findViewById(com.example.nano\_android.R.id.register\_name);  password = (EditText) findViewById(com.example.nano\_android.R.id.register\_password);  email = (EditText) findViewById(com.example.nano\_android.R.id.register\_email);  }  public void register\_register(View v){  Name = name.getText().toString();  Password = password.getText().toString();  Email = email.getText().toString();  BackGround b = new BackGround();  b.execute(Name, Password, Email);  }  class BackGround extends AsyncTask<String, String, String> {  @Override  protected String doInBackground(String... params) {  String name = params[0];  String password = params[1];  String email = params[2];  String data="";  int tmp;  try {  URL url = new URL("https://softwareengineeing.000webhostapp.com/register.php");  String urlParams = "name="+name+"&password="+password+"&email="+email;  HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();  httpURLConnection.setDoOutput(true);  OutputStream os = httpURLConnection.getOutputStream();  os.write(urlParams.getBytes());  os.flush();  os.close();  InputStream is = httpURLConnection.getInputStream();  while((tmp=is.read())!=-1){  data+= (char)tmp;  }  is.close();  httpURLConnection.disconnect();  return data;  } catch (MalformedURLException e) {  e.printStackTrace();  return "Exception: "+e.getMessage();  } catch (IOException e) {  e.printStackTrace();  return "Exception: "+e.getMessage();  }  }  @Override  protected void onPostExecute(String s) {  if(s.equals("")){  s="Data saved successfully.";  Intent i = new Intent(ctx, login.class);  finish();  startActivity(i);  }  Toast.makeText(ctx, s, Toast.LENGTH\_LONG).show();  }  }  } |

* Home.java ( 237 lines)

|  |
| --- |
| package com.example.nano\_android;  import android.R;  import android.content.Context;  import android.content.Intent;  import android.app.Activity;  import android.net.Uri;  import android.os.AsyncTask;  import android.os.Bundle;  import android.provider.ContactsContract;  import android.view.View;  import android.widget.Button;  import android.widget.TextView;  import org.json.JSONException;  import org.json.JSONObject;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;  import java.net.HttpURLConnection;  import java.net.MalformedURLException;  import java.net.URL;  public class home extends Activity {  Context ctx=this;  String DATES = null, TIMES = null, LEVEL = null;  String name, password, email, dates, times, level, Err;  String Name, Password, Email;  TextView nameTV, emailTV, passwordTV, datesTV, timesTV, levelTV, err, wel;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(com.example.nano\_android.R.layout.home);  nameTV = (TextView) findViewById(com.example.nano\_android.R.id.home\_name);  emailTV = (TextView) findViewById(com.example.nano\_android.R.id.home\_email);  passwordTV = (TextView) findViewById(com.example.nano\_android.R.id.home\_password);  datesTV = (TextView) findViewById(com.example.nano\_android.R.id.home\_dates);  timesTV = (TextView) findViewById(com.example.nano\_android.R.id.home\_times);  levelTV = (TextView) findViewById(com.example.nano\_android.R.id.home\_level);  err = (TextView) findViewById(com.example.nano\_android.R.id.err);  wel = (TextView) findViewById(com.example.nano\_android.R.id.wel);  Intent intent = getIntent();  name = intent.getStringExtra("name");  password = intent.getStringExtra("password");  email = intent.getStringExtra("email");  Err = intent.getStringExtra("err");  dates = intent.getStringExtra("dates");  times = intent.getStringExtra("times");  level = intent.getStringExtra("level");  wel.setText("WELCOME ");  nameTV.setText(""+name);  passwordTV.setText("WATER LEVEL SYSTEM");  emailTV.setText("MKEL 1243 SOFTWARE ENG");  err.setText(Err);  datesTV.setText("Date: " + dates);  timesTV.setText("Time: " + times);  levelTV.setText("Water level: " + level);  Button button = (Button) findViewById(com.example.nano\_android.R.id.home\_value);  button.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  ret\_data();  }  });  Button button2 = (Button) findViewById(com.example.nano\_android.R.id.home\_logout);  button2.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  Intent i = new Intent(ctx, login.class);  finish();  startActivity(i);  }  });  Button buttonOnSystem = (Button) findViewById(com.example.nano\_android.R.id.home\_on);  buttonOnSystem.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View arg) {  Intent i2 = new Intent(ctx, home.class);  Intent i = new Intent(Intent.ACTION\_VIEW, Uri.parse("http://akuzul.pagekite.me/water\_system.php"));  startActivity(i);  finish();  try {  Thread.sleep(5000);  } catch (InterruptedException e) {  e.printStackTrace();  }  finish();  startActivity(i2);  ret\_data();  }  });  Button buttonPump = (Button) findViewById(com.example.nano\_android.R.id.home\_start);  buttonPump.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View arg) {  Intent i = new Intent(Intent.ACTION\_VIEW, Uri.parse("http://akuzul.pagekite.me/start\_pump.php"));  startActivity(i);  finish();  try {  Thread.sleep(5000);  } catch (InterruptedException e) {  e.printStackTrace();  }  Intent i3 = new Intent(ctx, home.class);  startActivity(i3);  ret\_data();  }  });  Button buttonStopSystem = (Button) findViewById(com.example.nano\_android.R.id.home\_stop);  buttonStopSystem.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View arg) {  Intent i = new Intent(Intent.ACTION\_VIEW, Uri.parse("http://akuzul.pagekite.me/cleanup.php"));  startActivity(i);  finish();  try {  Thread.sleep(5000);  } catch (InterruptedException e) {  e.printStackTrace();  }  Intent i4 = new Intent(ctx, home.class);  startActivity(i4);  ret\_data();  }  });  }  public void ret\_data(){  Name = name.toString();  Password = password.toString();  Email = email.toString();  System.out.println(Name);  System.out.println(Password);  System.out.println(Email);  home.BackGround b = new home.BackGround();  b.execute(Name, Password, Email);  }  class BackGround extends AsyncTask<String, String, String> {  @Override  protected String doInBackground(String... params) {  String name = params[0];  String password = params[1];  String email = params[2];  System.out.println("debug");  String data = "";  int tmp;  try {  URL url = new URL("https://softwareengineeing.000webhostapp.com/rpi\_data\_home.php");  String urlParams = "";  System.out.println("debug2");  HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();  System.out.println("debug3");  httpURLConnection.setDoOutput(true);  System.out.println("debug4");  OutputStream os = httpURLConnection.getOutputStream();  System.out.println("debug5");  os.write(urlParams.getBytes());  System.out.println("debug6");  os.flush();  System.out.println("debug7");  os.close();  InputStream is = httpURLConnection.getInputStream();  while ((tmp = is.read()) != -1) {  data += (char) tmp;  }  System.out.println(data);  is.close();  httpURLConnection.disconnect();  try {  System.out.println("start read");  JSONObject root = new JSONObject(data);  JSONObject user\_data = root.getJSONObject("user");  DATES = user\_data.getString("now\_date");  TIMES = user\_data.getString("now\_time");  LEVEL = user\_data.getString("level");  System.out.println(DATES);  System.out.println(TIMES);  System.out.println(LEVEL);  Intent i = new Intent(ctx, home.class);  i.putExtra("dates", DATES);  i.putExtra("times", TIMES);  i.putExtra("level", LEVEL);  System.out.println("debug9");  i.putExtra("name", name);  i.putExtra("password", password);  i.putExtra("email", email);  finish();  startActivity(i);  } catch (JSONException e) {  e.printStackTrace();  }  return (data);  } catch (MalformedURLException e) {  e.printStackTrace();  return "Exception: " + e.getMessage();  } catch (IOException e) {  e.printStackTrace();  return "Exception: " + e.getMessage();  }  }  }  } |

### **Webpage Django**

<>.py

* Manage.py ( 22 lines)

|  |
| --- |
| #!/usr/bin/env python  import os  import sys  if \_\_name\_\_ == "\_\_main\_\_":  os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "charts.settings")  try:  from django.core.management import execute\_from\_command\_line  except ImportError:  # The above import may fail for some other reason. Ensure that the  # issue is really that Django is missing to avoid masking other  # exceptions on Python 2.  try:  import django  except ImportError:  raise ImportError(  "Couldn't import Django. Are you sure it's installed and "  "available on your PYTHONPATH environment variable? Did you "  "forget to activate a virtual environment?"  )  raise  execute\_from\_command\_line(sys.argv) |

* Setting.py (120lines)

|  |
| --- |
| Django settings for charts project.  Generated by 'django-admin startproject' using Django 1.10.6.  For more information on this file, see  https://docs.djangoproject.com/en/1.10/topics/settings/  For the full list of settings and their values, see  https://docs.djangoproject.com/en/1.10/ref/settings/  """  import os  SECURE\_HSTS\_SECONDS = 1  SECURE\_CONTENT\_TYPE\_NOSNIFF = True  SECURE\_BROWSER\_XSS\_FILTER = True  SECURE\_SSL\_REDIRECT = True  SESSION\_COOKIE\_SECURE = True  X\_FRAME\_OPTIONS = 'DENY'  SECURE\_HSTS\_SUBDOMAINS = True  CSRF\_COOKIE\_SECURE = True  SECURE\_HSTS\_PRELOAD = True  SECURE\_HSTS\_INCLUDE\_SUBDOMAINS = True  ALLOWED\_HOSTS = ['\*']  # Application definition  INSTALLED\_APPS = [  'django.contrib.admin',  'django.contrib.auth',  'django.contrib.contenttypes',  'django.contrib.sessions',  'django.contrib.messages',  'django.contrib.staticfiles',  'rest\_framework',  #'rest\_auth',  ]  MIDDLEWARE = [  'django.middleware.security.SecurityMiddleware',  'django.contrib.sessions.middleware.SessionMiddleware',  'django.middleware.common.CommonMiddleware',  'django.middleware.csrf.CsrfViewMiddleware',  'django.contrib.auth.middleware.AuthenticationMiddleware',  'django.contrib.messages.middleware.MessageMiddleware',  'django.middleware.clickjacking.XFrameOptionsMiddleware',  ]  ROOT\_URLCONF = 'charts.urls'  TEMPLATES = [  {  'BACKEND': 'django.template.backends.django.DjangoTemplates',  'DIRS': [os.path.join(BASE\_DIR, 'templates')],  'APP\_DIRS': True,  'OPTIONS': {  'context\_processors': [  'django.template.context\_processors.debug',  'django.template.context\_processors.request',  'django.contrib.auth.context\_processors.auth',  'django.contrib.messages.context\_processors.messages',  ],  },  },  ]  WSGI\_APPLICATION = 'charts.wsgi.application'  # Database  # https://docs.djangoproject.com/en/1.10/ref/settings/#databases  DATABASES = {  'default': {  'ENGINE': 'django.db.backends.sqlite3',  'NAME': 'id1489556\_water\_level\_reading',  'HOST': 'softwareengineeing.000webhostapp.com',  'PORT': 'localhost',  'USER': 'id1489556\_admin',  'PASSWD': '12345',  }  }  # Password validation  # https://docs.djangoproject.com/en/1.10/ref/settings/#auth-password-validators  AUTH\_PASSWORD\_VALIDATORS = [  {  'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  },  {  'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  },  {  'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  },  {  'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  },  ]  # Internationalization  # https://docs.djangoproject.com/en/1.10/topics/i18n/  LANGUAGE\_CODE = 'en-us'  TIME\_ZONE = 'UTC'  USE\_I18N = True  USE\_L10N = True  USE\_TZ = True  # Static files (CSS, JavaScript, Images)  # https://docs.djangoproject.com/en/1.10/howto/static-files/  STATIC\_URL = '/static/' |

* urls.py (12 lines)

|  |
| --- |
| from django.conf.urls import url  from django.contrib import admin  from .views import HomeView, get\_data, ChartData  urlpatterns = [  url(r'^$', HomeView.as\_view(), name='home'),  url(r'^api/data/$', get\_data, name='api-data'),  url(r'^api/chart/data/$', ChartData.as\_view()),  url(r'^admin/', admin.site.urls),  ] |

* view.py ( 76 lines)

|  |
| --- |
| from django.contrib.auth import get\_user\_model  from django.http import JsonResponse  from django.shortcuts import render  from django.views.generic import View  #import requests  from rest\_framework.views import APIView  from rest\_framework.response import Response  #import urllib.request  import pdb  import os  import sys  import json  import urllib  import urllib2  User = get\_user\_model()  class HomeView(View):  def get(self, request, \*args, \*\*kwargs):  return render(request, 'charts.html', {"customers": 10})  def get\_data(request, \*args, \*\*kwargs):  data = {  "sales": 100,  "customers": 10,  }  return JsonResponse(data) # http response  class ChartData(APIView):  authentication\_classes = []  permission\_classes = []  """  def get(self, request, format=None):  qs\_count = User.objects.all().count()  labels = ["1pm", "2pm", "3pm", "4pm", "5pm", "6pm"]  default\_items = [qs\_count, 89, 70, 75, 90, 50]  data = {  "labels": labels,  "default": default\_items,  }  print (data)  return Response(data)  """  def get(self, request, format=None):  qs\_count = User.objects.all().count()  url = "https://softwareengineeing.000webhostapp.com/read\_water\_level\_daily.php"  phpcontent = urllib2.urlopen(url, timeout=None)  phpcontent = phpcontent.read()  file = open("database.txt", "w+")  print >> file, phpcontent  file.close()  database = open("database.txt", "r+")  date\_ = []  levels = []  for item in database:  item = item.strip()  if "date" in item:  raw\_data = item  data\_ = json.loads(raw\_data)  print (data\_[0]["date"])  date = data\_[0]["date"]  print (data\_[0]["level"])  level = data\_[0]["level"]  date\_.append(date)  levels.append(level)  data = {  "labels": date\_,  "default": levels,  }  print (data)  return Response(data) |

* wsgi.py (4 lines)

|  |
| --- |
| import os  from django.core.wsgi import get\_wsgi\_application  os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "charts.settings")  application = get\_wsgi\_application() |

<>.html

* base.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1">  <title>Water Level System</title>  {% include 'base/css.html' %}  {% include 'base/bootstrap\_defaults.html' %}  </head>  <body>  <div class='container'>  {% block content %}  {% endblock content %}  </div>  {% include 'base/js.html' %}  <script>  $(document).ready(function(){  {% block jquery %}{% endblock %}  })  </script>  </body>  </html> |

* base\_ori.html

|  |
| --- |
| {% extends 'base.html' %}  <script>  {% block jquery %}  var endpoint = '/api/chart/data/'  var defaultData = []  var labels = [];  $.ajax({  method: "GET",  url: endpoint,  success: function(data){  labels = data.labels  defaultData = data.default  setChart()  },  error: function(error\_data){  console.log("error")  console.log(error\_data)  }  })  function setChart(){  var ctx = document.getElementById("myChart");  var ctx2 = document.getElementById("myChart2");  var myChart = new Chart(ctx2, {  type: 'bar',  data: {  labels: labels,  datasets: [{  label: '# of Votes',  data: defaultData,  backgroundColor: [  'rgba(255, 99, 132, 0.2)',  'rgba(54, 162, 235, 0.2)',  'rgba(255, 206, 86, 0.2)',  'rgba(75, 192, 192, 0.2)',  'rgba(153, 102, 255, 0.2)',  'rgba(255, 159, 64, 0.2)'  ],  borderColor: [  'rgba(255,99,132,1)',  'rgba(54, 162, 235, 1)',  'rgba(255, 206, 86, 1)',  'rgba(75, 192, 192, 1)',  'rgba(153, 102, 255, 1)',  'rgba(255, 159, 64, 1)'  ],  borderWidth: 1  }]  },  options: {  scales: {  yAxes: [{  ticks: {  beginAtZero:true  }  }]  }  }  });  var myChart = new Chart(ctx, {  type: 'polarArea',  data: {  labels: labels,  datasets: [{  label: '# of Votes',  data: defaultData,  backgroundColor: [  'rgba(255, 99, 132, 0.2)',  'rgba(54, 162, 235, 0.2)',  'rgba(255, 206, 86, 0.2)',  'rgba(75, 192, 192, 0.2)',  'rgba(153, 102, 255, 0.2)',  'rgba(255, 159, 64, 0.2)'  ],  borderColor: [  'rgba(255,99,132,1)',  'rgba(54, 162, 235, 1)',  'rgba(255, 206, 86, 1)',  'rgba(75, 192, 192, 1)',  'rgba(153, 102, 255, 1)',  'rgba(255, 159, 64, 1)'  ],  borderWidth: 1  }]  },  options: {  scales: {  yAxes: [{  ticks: {  beginAtZero:true  }  }]  }  }  });  }  // var ctx = document.getElementById("myChart");  {% endblock %}  </script>  {% block content %}  <div class='row'>  <div class='col-sm-12' url-endpoint='{% url "api-data" %}'>  <h1>Software Engineering Project</h1>  <div class='col-sm-6'>  <canvas id="myChart" width="400" height="400"></canvas>  </div>  <div class='col-sm-6'>  <canvas id="myChart2" width="400" height="400"></canvas>  </div  </div>  </div>  {% endblock content %} |

* charts.html

|  |
| --- |
| {% extends 'base.html' %}  <script>  {% block jquery %}  var endpoint = '/api/chart/data/'  var defaultData = []  var labels = [];  $.ajax({  method: "GET",  url: endpoint,  success: function(data){  labels = data.labels  defaultData = data.default  setChart()  },  error: function(error\_data){  console.log("error")  console.log(error\_data)  }  })  function setChart(){  var ctx = document.getElementById("myChart");  var ctx2 = document.getElementById("myChart2");  var myChart = new Chart(ctx2, {  type: 'bar',  data: {  labels: labels,  datasets: [{  label: 'Water Level',  data: defaultData,  backgroundColor: [  'rgba(255, 99, 132, 0.2)',  'rgba(54, 162, 235, 0.2)',  'rgba(255, 206, 86, 0.2)',  'rgba(75, 192, 192, 0.2)',  'rgba(153, 102, 255, 0.2)',  'rgba(255, 159, 64, 0.2)',  'rgba(255, 99, 132, 0.2)',  'rgba(54, 162, 235, 0.2)',  'rgba(255, 206, 86, 0.2)',  'rgba(75, 192, 192, 0.2)',  'rgba(153, 102, 255, 0.2)',  'rgba(255, 159, 64, 0.2)'  ],  borderColor: [  'rgba(255,99,132,1)',  'rgba(54, 162, 235, 1)',  'rgba(255, 206, 86, 1)',  'rgba(75, 192, 192, 1)',  'rgba(153, 102, 255, 1)',  'rgba(255, 159, 64, 1)',  'rgba(255,99,132,1)',  'rgba(54, 162, 235, 1)',  'rgba(255, 206, 86, 1)',  'rgba(75, 192, 192, 1)',  'rgba(153, 102, 255, 1)',  'rgba(255, 159, 64, 1)'  ],  borderWidth: 1  }]  },  options: {  scales: {  yAxes: [{  ticks: {  beginAtZero:true  }  }]  }  }  });  }  // var ctx = document.getElementById("myChart");  {% endblock %}  </script>  {% block content %}  <div class='row'>  <div class='col-sm-12' url-endpoint='{% url "api-data" %}'>  <h1>Water Level System</h1>  <div class='col-sm-6'>  <canvas id="myChart2" width="20" height="20"></canvas>  </div>  </div>  </div>  {% endblock content %} |

### **Server/database**

* Ini.php ( 14 lines)

|  |
| --- |
| <?php  error\_reporting(0);  $db\_name = "id1489556\_android\_database";  $mysql\_user = "id1489556\_android";  $mysql\_pass = "software";  $server\_name = "localhost";  $con = mysqli\_connect($server\_name, $mysql\_user, $mysql\_pass, $db\_name);  if(!$con){  echo '{"message":"Unable to connect to the database. Please try again"}';  }  ?> |

* Init\_rpi.php (15 lines)

|  |
| --- |
| <?php  error\_reporting(0);  $db\_name = "id1489556\_software\_eng";  $mysql\_user = "id1489556\_admin";  $mysql\_pass = "software";  $server\_name = "localhost";  $con = mysqli\_connect($server\_name, $mysql\_user, $mysql\_pass, $db\_name);  if(!$con){  echo '{"message":"Unable to connect to the database."}';  }  ?> |

* Login.php (16 lines)

|  |
| --- |
| <?php  error\_reporting(0);  require "init\_rpi.php";  $sql = "SELECT \* FROM `register\_1` WHERE name='$\_POST[name]' AND password='$\_POST[password]'";  $result = mysqli\_query($con, $sql);  $response = array();  while($row = mysqli\_fetch\_array($result)){  $response = array("id"=>$row[0],"name"=>$row[1],"password"=>$row[2],"email"=>$row[3]);  }  echo json\_encode(array("user"=>$response));  ?> |

* Register.php (15 lines)

|  |
| --- |
| <?php  error\_reporting(0);  require "init\_rpi.php";  $name = $\_POST["name"];  $password = $\_POST["password"];  $email = $\_POST["email"];  $sql = "INSERT INTO `register\_1` (`id`,`name`, `password`, `email`) VALUES (NULL, '".$name."', '".$password."', '".$email."');";  if(!mysqli\_query($con, $sql)){  echo '{"message":"Unable to save the data to the database."}';  }  ?> |

* Rpi\_data\_home.php (11 lines)

|  |
| --- |
| <?php  error\_reporting(0);  require "init\_rpi.php";  $result = mysqli\_query($con, $sql);  $response = array();  while($row = mysqli\_fetch\_array($result)){  $response = array("id"=>$row[0],"now\_date"=>$row[1],"now\_time"=>$row[2],"level"=>$row[3]);  }  echo json\_encode(array("user"=>$response));  ?> |

* **insert\_water\_level.php**

|  |
| --- |
| <?php  error\_reporting(0);  require "init\_rpi.php";  $level = $\_POST["level"]; //water\_level  //$level = "2";  date\_default\_timezone\_set("Asia/Kuala\_Lumpur");  $time = date("H:i:s");  echo $time;  $sql = "INSERT INTO water\_level (now\_date, now\_time, level)  VALUES (now(), '".$time."', '".$level."')";  //echo $sql ;  if(!mysqli\_query($con, $sql)){  echo '{"message":"Unable to save the data to the database."}';  }  else{  echo '{"message":"Table updated"}';  }  $conn->close();  ?> |

* **read\_water\_level\_daily.php**

|  |
| --- |
| <?php  error\_reporting(0);  require "init\_rpi.php";  $sql = "SELECT \* FROM `water\_level` WHERE now\_date = CURDATE()";  $result = mysqli\_query($con, $sql);  if ($result->num\_rows > 0) {  // output data of each row  while($row = $result->fetch\_assoc()) {  echo '[{"date":"' . $row["now\_date"]. '","level":"' . $row["level"]. '"'. "}]\r\n";    }  } else {  echo "0 results";  }  ?> |

* **read\_water\_level\_hourly.php**

|  |
| --- |
| <?php  error\_reporting(0);  require "init\_rpi.php";  $sql = "SELECT \* FROM `water\_level` WHERE id=( SELECT MAX(id) FROM `water\_level`)";  $result = mysqli\_query($con, $sql);  $response = array();  while($row = mysqli\_fetch\_array($result)){  $response = array("now\_date"=>$row[1],"now\_time"=>$row[2],"level"=>$row[3]);  echo json\_encode(array($response));  }  ?> |

### **Raspberry Pi**

* **water\_system.py**

|  |
| --- |
| import RPi.GPIO as GPIO  import urllib  import urllib2  import requests  from time import sleep  GPIO.setmode(GPIO.BCM)  urllib.urlopen('http://akuzul.pagekite.me/killpump.php')  urllib.urlopen('http://akuzul.pagekite.me/killclean.php')  try:    while 1:    url2 = 'http://softwareengineeing.000webhostapp.com/insert\_water\_level.php'  count = 0  GPIO.setup(2, GPIO.OUT, initial=GPIO.LOW)  sleep(0.05)  GPIO.setup(14, GPIO.IN)  GPIO.setup(15, GPIO.IN)  GPIO.setup(18, GPIO.IN)  GPIO.setup(12, GPIO.IN)  GPIO.setup(16, GPIO.IN)  GPIO.setup(20, GPIO.IN)  GPIO.setup(10, GPIO.IN)  GPIO.setup(9, GPIO.IN)  GPIO.setup(11, GPIO.IN)  GPIO.setup(5, GPIO.IN)  GPIO.setup(6, GPIO.IN)  GPIO.setup(13, GPIO.IN)  GPIO.setup(19, GPIO.IN)  GPIO.setup(26, GPIO.IN)  GPIO.setup(21, GPIO.IN)  if(GPIO.input(14) == GPIO.LOW):  GPIO.output(2, GPIO.HIGH)  level=[('level','15')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 15  print("level=%d")%count  sleep(1)  elif(GPIO.input(15) == GPIO.LOW):  GPIO.output(2, GPIO.HIGH)  level=[('level','14')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 14  print("level=%d")%count  sleep(1)  elif(GPIO.input(18) == GPIO.LOW):  GPIO.output(2, GPIO.HIGH)  level=[('level','13')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 13  print("level=%d")%count  sleep(1)  elif(GPIO.input(12) == GPIO.LOW):  GPIO.output(2, GPIO.HIGH)  level=[('level','12')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 12  print("level=%d")%count  sleep(1)  elif(GPIO.input(16) == GPIO.LOW):  GPIO.output(2, GPIO.HIGH)  level=[('level','11')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 11  print("level=%d")%count  sleep(1)  elif(GPIO.input(20) == GPIO.LOW):  GPIO.output(2, GPIO.HIGH)  level=[('level','10')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 10  print("level=%d")%count  sleep(1)  elif(GPIO.input(10) == GPIO.LOW):  GPIO.output(2, GPIO.HIGH)  level=[('level','9')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 9  print("level=%d")%count  sleep(1)  elif(GPIO.input(9) == GPIO.LOW):  level=[('level','8')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 8  print("level=%d")%count  sleep(1)  elif(GPIO.input(11) == GPIO.LOW):  level=[('level','7')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 7  print("level=%d")%count  sleep(1)  elif(GPIO.input(5) == GPIO.LOW):  #GPIO.output(2, GPIO.HIGH)  level=[('level','6')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 6  print("level=%d")%count  sleep(1)  elif(GPIO.input(6) == GPIO.LOW):  level=[('level','5')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 5  print("level=%d")%count  sleep(1)  elif(GPIO.input(13) == GPIO.LOW):  level=[('level','4')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 4  print("level=%d")%count  sleep(1)  elif(GPIO.input(19) == GPIO.LOW):  level=[('level','3')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 3  print("level=%d")%count  sleep(1)  elif(GPIO.input(26) == GPIO.LOW):  level=[('level','2')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 2  print("level=%d")%count  sleep(1)  elif(GPIO.input(21) == GPIO.LOW):  level=[('level','1')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  count = 1  print("level=%d")%count  sleep(1)  else:  level=[('level','0')]  level = urllib.urlencode(level)  req = urllib2.Request(url2, level)  page=urllib2.urlopen(req).read()  print page  #url = "http://softwareengineering.000webhostapp.com/try\_pali.php"  #response = urllib.urlopen(url).read()  #print response  print("level=%d")%count  sleep(1)  except KeyboardInterrupt:  print ("\nCtrl-C pressed. Program exiting...")  finally:  GPIO.cleanup() # run on exit |

* **start\_pump.py**

|  |
| --- |
| import RPi.GPIO as GPIO  import urllib  import urllib2  import requests  from time import sleep  GPIO.setmode(GPIO.BCM)  try:  urllib.urlopen('http://akuzul.pagekite.me/killclean.php')  sleep(1)  urllib.urlopen('http://akuzul.pagekite.me/kill.php')  sleep(1)  while 1:  GPIO.setup(2, GPIO.OUT, initial=GPIO.LOW)  sleep(0.05)  GPIO.output(2, GPIO.HIGH)  sleep(1)  except KeyboardInterrupt:  print ("\nCtrl-C pressed. Program exiting...")  finally:  GPIO.cleanup() # run on exit |

* **cleanup.py**

|  |
| --- |
| import RPi.GPIO as GPIO  import urllib  import requests  from time import sleep  GPIO.setmode(GPIO.BCM)  try:  urllib.urlopen('http://akuzul.pagekite.me/killpump.php')  urllib.urlopen('http://akuzul.pagekite.me/kill.php')  while 1:  GPIO.setup(2, GPIO.OUT, initial=GPIO.LOW)  GPIO.output(2, GPIO.LOW)  sleep(0.05)  sleep(1)  finally:  GPIO.cleanup() # run on exit |