

Chapter 1

INTRODUCTION

This Chapter gives overview of the project, the problem statement, the proposed solution, the objective of project and about the organization of report

1.1 Overview

The advancements and developments happening in the field of science and technology has brought a tremendous change in day-to-day life of human being where everything is made smart and anything can be accessed at the finger tips using smart phones via apps web applications.

In this project, an IoT module designed and developed for the Sustain Earth company to automate the billing and metering system.

Sustain Earth uses innovative Biogas technology to provide affordable, clean cooking gas rural communities. By rebranding the now-notorious Biogas product as Gau-Gas and implementing new materials, new processes, and new technology. Sustain Earth solves the problems faced by the last generation of Biogas users and change the perception about the usefulness of Biogas.

The Project is implemented using Arduino board with various programmable sensors to check and control the power supply.

1.2 Problem Statement

The Gau-Gas plants are Installed in various rural areas and these plants are very far away to each other this makes the billing process tedious. The billing man cannot travel far distances to collect logistics usage from IoT module. The present IoT module near the plant requires internet connection to share the logistics usage with the billing and the module is not secured where any one can easily connect to module by knowing the IP address and can manipulate the data. The data literacy in rural areas is less and hence required the internet less solution. whenever the billing man goes near the plant a hotspot is created and the billing man has to connect to this hotspot and enters an ip address which displays the uptime of the system and hence whoever aware of the ip address can easily collect and modify the data.

1.3 Existing System

To get the logistics usage of the gas, the billing person connects the ESP8266 and acquire the data from the Arduino Leonardo , i.e. the ON time of the pump . He then connects this module to a IP address to send the logistics to the data server . The pump is then reset using an other IP address .



Figure 1.1: Existing System

1.4 Proposed Solution

In this project an IOT module is integrated along with each gaugas plant which are installed in different locations in rural areas. These IOT module measure the gas logistics usage by a particular user and the logistics usage are sent as a text message with authentication code to android application which acts as a gateway interface.

After validating the authentication code, the android application pushes the logistics usage into the database. The android application pushes the data only when the internet is available. An interactive web dashboard reflects the change of data in the database and it also displays the statistical display per user usage logs and it also generates a bill invoice.

The IOT module also facilitates the controlling of plant remotely where the admin can lock the module when user fails to pay the bill, the admin can also open the IoT module both lock and opening the IoT module is achieved by sending SMS to IoT module.

1.5 Organization of Report

The rest of the report is organized as follows

Chapter 2 deals with literature survey and the technologies used which is part of the project.

Chapter 3 briefs the software requirements specification, functionality, working environment and output format.

Chapter 4 gives the complete design i.e. architecture, data flow diagrams, use case diagram, module diagram and sequence diagram.

Chapter 5 deals with the implementation of the project.

Chapter 6 deals with software testing.

Chapter 7 gives the results and snapshots.

Next is the Conclusion and future work of the project. It briefly tells what the project is doing and what has been accomplished. Future work is also specified in this section.

Apart from the above, the References used are listed and the appendix provides snapshots of various interfaces used in the project.

Chapter 2

LITERATURE SURVEY

The purpose of this literature survey is to provide background information on the issues to be considered in this thesis and to emphasize the relevance of the present study. This Chapter gives the survey of associated technologies and summary of related work done in the past.

The main objective is to design an economical system to measure the amount of gau-gas consumed by the user. We use 10W AC pump to transmit gas from digester to burner. So we have two methods to measure the same: First- By using flow sensors at the output of pump, but the inconvenience is the price of the flammable gas flow measuring sensors which is too high. Second- As the flow rate of pump is constant so by measuring the duration for which the pump runs we can comfortably measure the volume of gas transfer from digester to stove.

2.1 APPROACH

Step-1

In starting we select the Arduino-Leonardo to calculate the total run time of the pump because it is economical, programming is simple by using micro USB connector and it has inbuilt EPROM memory. So by connecting the same with the pump and by calculating the ON time of the Arduino we can measure the run time of the pump as both are same.



Figure 2.1Arduino leonardo

Arduino has inbuilt millis() function which return the number of milliseconds since the Arduino board began running the current program.

Output



Figure 2.2 Output of Arduino Leonardo using serial monitor.

Disadvantage of Arduino Leonardo is, it communicates with the help of USB to serial converter and every time the Leonardo is reset it will create a new virtual COM port with a different number.

Step-2

To overcome the drawbacks of Leonardo we now use ESP8266 module which is low cost WiFi module suitable for adding WiFi functionality to an existing microcontroller project and it acts as a standalone WiFi connected device just by adding power. The main advantage of ESP8266 module is, coding can be done by using same Arduino Ide software and it also has inbuilt EEPROM memory. ESP module runs on 3.3V DC supply and does not have 5V tolerant inputs, so we need level conversion to communicate. We use Ftdi Usb To Ttl 5V 3.3V Cable to Serial Adapter Module for communicating.

ESP8266 MODULE	FTDI ADAPTOR
TX	RXD
CH_PD	PWR
RST	
3.3V	PWR
GND	GND
GPIO2	
GPIO0	GND
RX	TXD

Table 2.1 Connections

Output

Connect to SUSTAIN.EARTH and then go to <http://192.168.4.1> in a web browser to see total run time of pump.

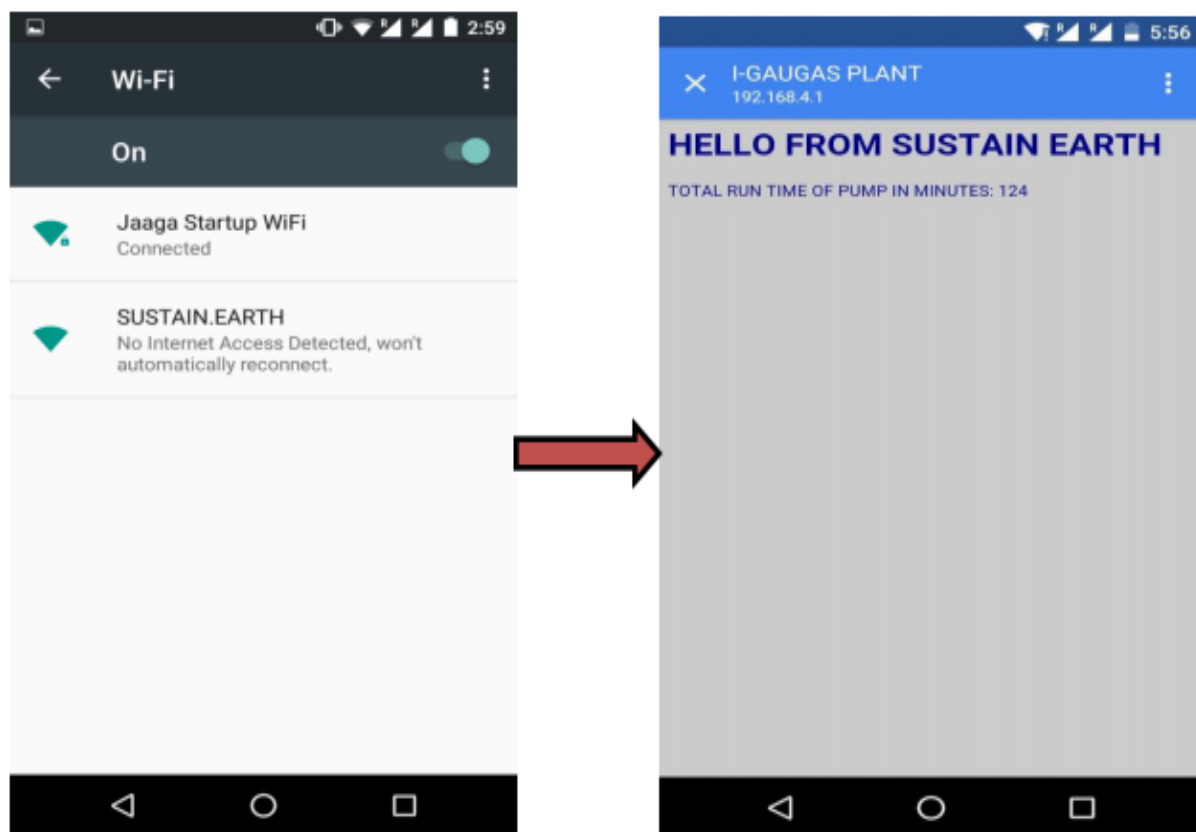


Figure 2.3 Output of ESP8266 module using smartphone

To reset the ESP8266 module go to <http://192.168.4.1/resetpump> in a web browser

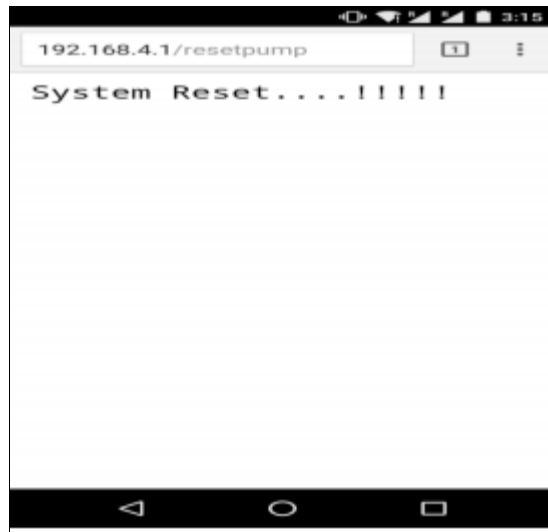


Figure 2.4 Output of ESP8266 after resetting the system.

Step-3

Now in next step, we want to connect our ESP module to the pump but the point at issue is our pump runs on 220V AC supply and ESP module runs on 3.3V DC. So we required AC to DC 220V to 3.3V Step-Down Buck Voltage Regulator Power Supply Module but same is not available in local market. So we decide to use 5V 1A AC to DC mobile adaptor and then with the help of AMS1117 voltage regulator and two 25V 100 micro-farad capacitor we further step down 5V DC to 3.3V DC.

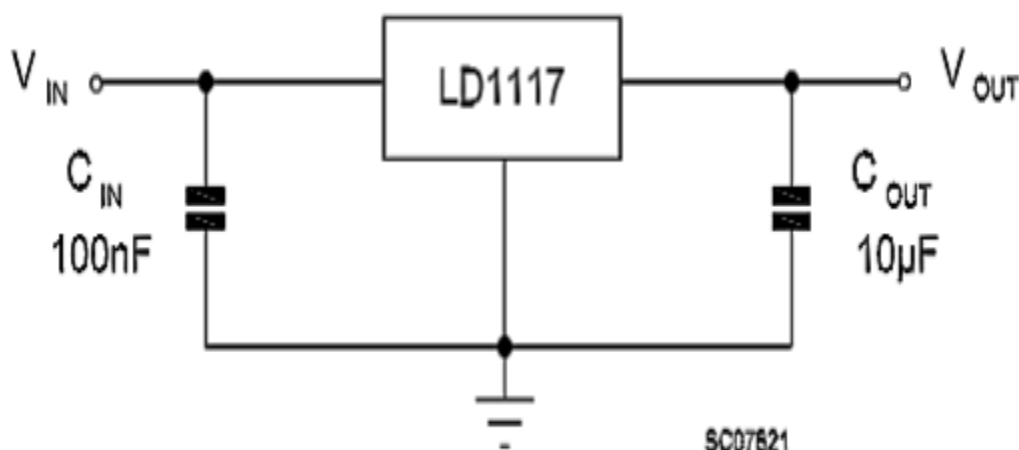


Figure 2.5 Systematic connection diagram

Step-4

Now we required proper casing to enclose ESP module and step down voltage regulator. So we select one ready to use case from local market.



Figure 2.6 Final run time calculating system.

FIELD TRIP:

This field trip was a visit to Gau gas plant which is located at Gundlakadapalli near Tirupati. Sustain Earth field Engineer had incorporated with us to the field . It was assumed that the gas would be generated and made them flow through the pipes . We thought that the system would be just like a traditional gobar gas plant . After visiting the field , the whole setup was completely different . The gas container was made up of polythene bag coated with fire proof material . The setup was maintainable and portable .

It would require 6 hours or less to fill the gas container . The gas container had a capacity of 5000 ltrs , where 2000 ltrs would be of the gau gas produced . The rest 3000 ltrs was slurry which would be left out in an outlet pipe . This slurry could be also reused as manure . The usage of only cow dung makes the gas stable . It can also be provided with the vegetable waste and other degradable wastes , but it is not stable as cow dung .

The input given to the setup was in the ratio of 50 kg cow dung to 75 kg water . The flow of gas through the pipe from the plant is always constant . There is a small unit called moisturizer which de moisturizes the humidity at every junction of the pipe . The setup is not much affected by the weather variations . During the summer it would produce much higher gas as compared to the winter . But still the ratio of the production is similar .

They use a mechanical Flow meter to measure the inflow of gas through the pipes . It costs around 4000 Rs. – 3500 Rs. Per device .

They use a step up booster pump to supply the gas to the stove . The motor was fitted with an analog regulator to provide a variable pressure to the user . The pump can run in two modes : A high flammable mode and a normal flame . The user can also control flow of gas on the regulator of the stove . Since the flow of gas and its maintenance needs to be digitalized .So , we enquired them if the regulator knob can be removed , they gave a positive reply . The implementation would be three levels of supply of gas to the stove .

The pump specification used is :

Power : 10w, Maximum Pressure (kgf/cm²)>0.12 , Maximum Vacuum(-kpa): -0.085 ,

Flow \geq 26 L/min , Load flow \geq 12(70 meter)L/min .

Presently they are taking readings based on the ESP Module and Flow Meter .

We are thinking to integrate our system along with the pump .

The readings noted down was :

Pump @ Level 1 : Single Burner ; Time : 1 minute ; 6 units ;

Pump @ Level 2 : Two Burners ; Time : 1 minutes ; 16 units ;



Figure 2.8 Gundlakadapalli



Figure 2.9 GauGas Plant at Gundlakadapalli

Chapter 3

SOFTWARE REQUIREMENTS AND SPECIFICATION

The goal of software requirement and specification is to describe what the proposed should do. It is the medium through which the user needs are accurately specified. It provides reference for the validation of the final product.

3.1 General Description

3.1.1 Product Perspective

Sustain Earth uses innovative Biogas technology to provide affordable, clean cooking gas rural communities .By rebranding the now-notorious Biogas product as Gau-Gas and implementing new materials, new processes, and new technology. SustainEarth solves the problems faced by the last generation of Biogas users and change the perception about the usefulness of Biogas.

- **Gau-Gas** systems are easier to install, more efficient, easier to maintain.
- They use Internet and Mobile connectivity to collect the usage logs.
- Since these systems are present in rural areas, we cannot defy an internet solution.
- The main disadvantage is the manual collection of usage logs.
- Based on these usage logs billing is done.
- Main challenge is to reduce manual intervention and to fully automate the billing and metering of those systems.
- Finally, to devise a no Internet and easy usable solution since digital literacy of rural areas are less.

3.1.2 Product Functions

We select the Arduino Leonardo board to calculate the total run time of the pump .The Arduino Leonardo is a microcontroller board based on the ATmega32u4 . It has 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Leonardo differs from all preceding boards in that the ATmega32u4 has built-in USB communication, eliminating the need for a secondary processor. This allows the Leonardo to appear to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial / COM port.

The programming work is simpler by the usage of micro USB cable that connects to the computer or laptop having the required software and hardware specifications . The Arduino Leonardo board is used to calculate the Pump ON time and OFF time . However we only require the ON time since that is the time we need to calculate the USAGE LOGISTICS of the GAU GAS consumed by the customer .

Arduino Leonardo has an inbuilt millis() function which return the number of milliseconds since the Arduino board began running the current program . This function returns the values on a separate window called as SERIAL MONITOR with the specific frequency ,i.e. baud rate mentioned .

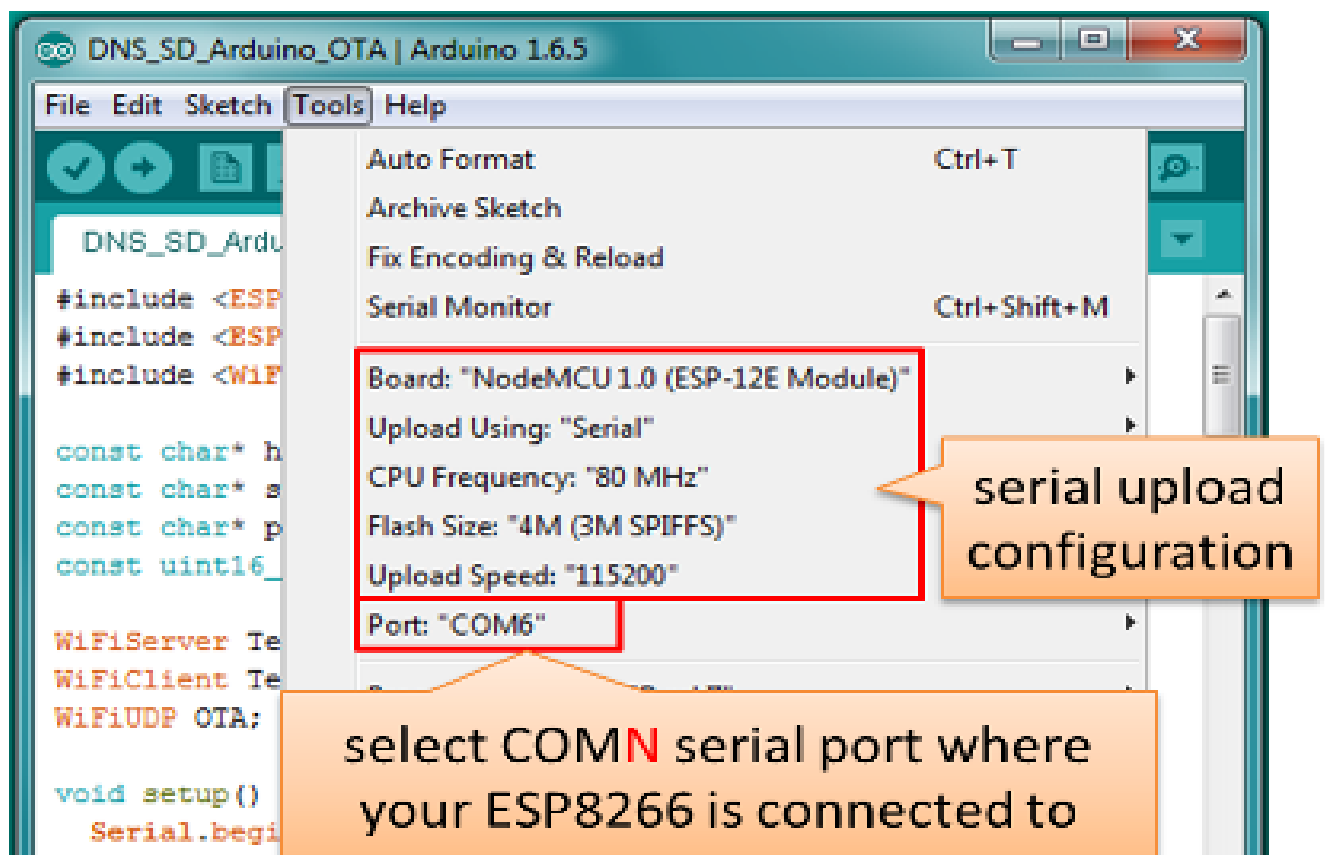


Figure 3.1 Configuration of ESP8266

3.1.3 Constraint of Arduino Leonardo

Arduino is controller dependent and its hardware capability keeps on changing .It communicates with the help of USB to serial converter and every time the Leonardo is reset it will create a new virtual COM port with a different number.So , the usage logistics of the gas cannot be retrieved when the device gets reset .To , overcome this drawback of Leonardo we use the ESP8266 WiFi module .

3.1.4 Constraint of Previous module

ESP8266 has 8 pins, 4 in the row of 2. The first pin on the top left is GND. The two pins right from the GND are GPIO 2 and 0. The pin on the top right side is the RX pin and the pin on the lower left is TX. These are the pins for communication. The middle pins on the bottom are CH_PD(chip power-down) and RST(reset).

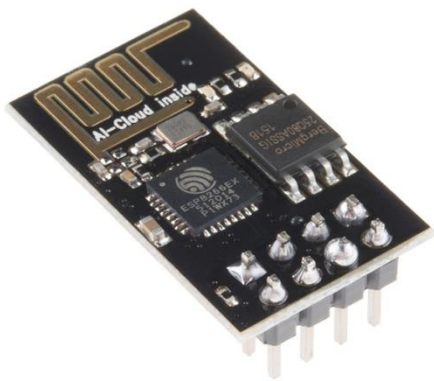


Figure 3.2 ESP8266

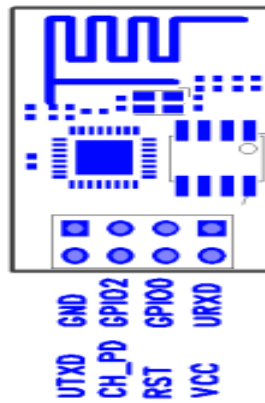


Figure 3.3 ESP8266 Pinout

The main advantage of ESP8266 module is, coding can be done by using same ARDUINO IDE software and it also has inbuilt EEPROM memory. The connections to the Arduino Leonardo are shown below as in the diagram .

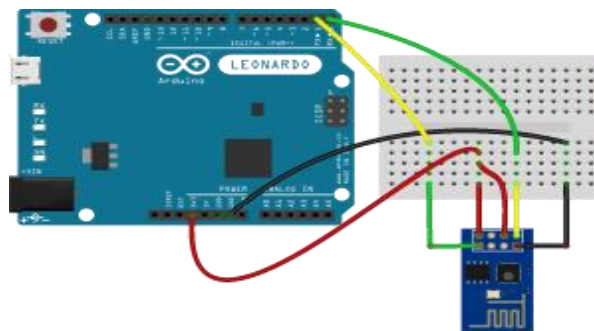


Figure 3.4 Previous Model

NOTE :The power supply to esp8266 is 3v , so the connection from arduino has to be given to the 3v power supply from the arduino.

To get the logistics usage of the gas, the billing person connects the ESP8266 and acquire the data from the Arduino Leonardo , i.e. the ON time of the pump . He then connects this module to a IP address to send the logistics to the data server . The pump is then reset using an other IP address .

3.1.5 Product Upgradation

Since the Digital literacy and Internet availability in rural areas are less, we shall replace the feature of retrieving the usage logistics of Gau Gas consumed by the customer through a SMS rather over Internet.This does not require any internet facility to send the messages, all we need is an active connection. These usage logs are encoded and then transmitted to the Database for security purposes.A graphical dashboard at the provider's side provided for easy maintenance of systems.

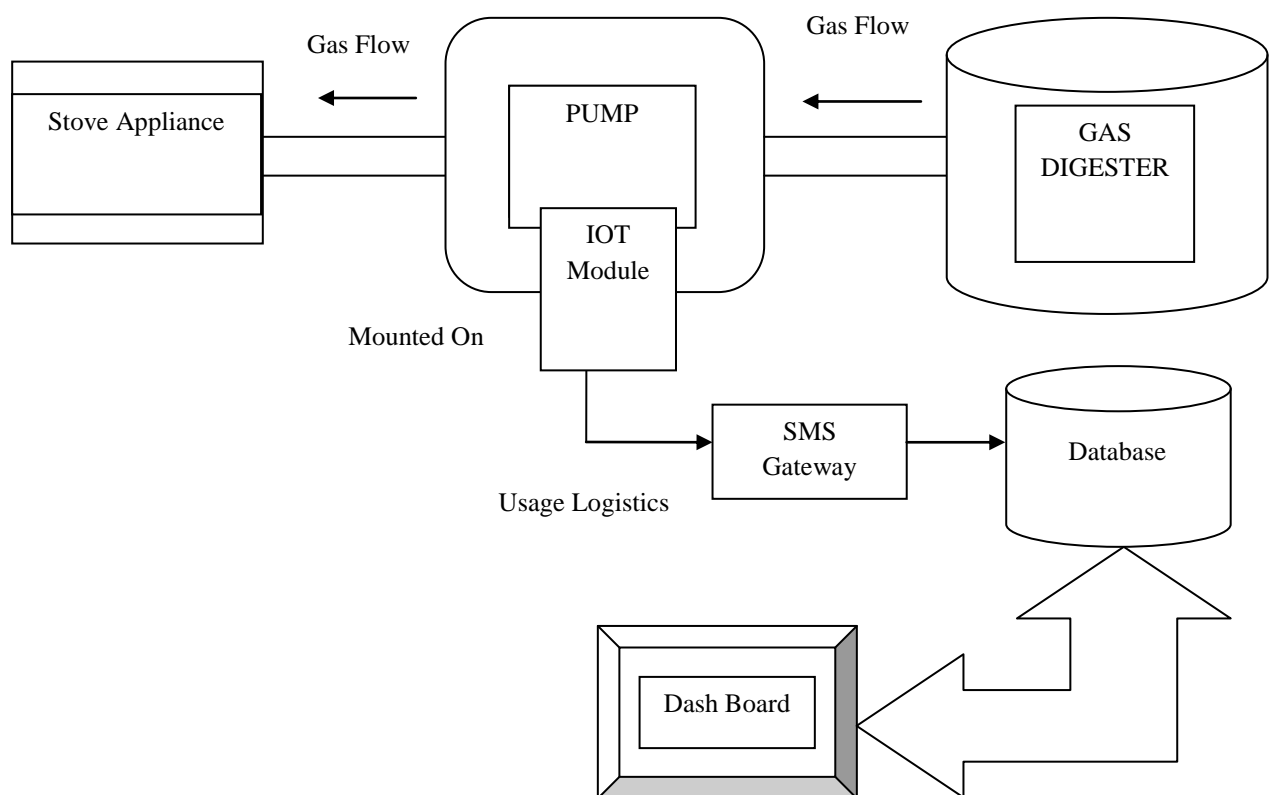


Figure 3.5 Architecture

3.2 Assumptions

- Presently they are using duration of pump runs to measure the volume of the gas transfer from digester to the store since the flow rate of pump is constant.
- We are controlling the flow of gas through regulator.
- We have provided with three levels of flow control i.e, low, medium, high.
- So whenever the regulator is turned on the timer associated with corresponding level is started, whenever the regulator is turned off the corresponding timer value is transmitted.
- This usage logs are then transmitted to the mobile application using a GSM module.
- The mobile application receives the data sent from the device and pushes the data to the database present at the server side.
- Usage logs can be transmitted per day or per hour as per provider requirements.
- If the usage logs are received from same number then it should be added to the previous logs of that number.
- Dashboard/statics display are used to represent the usage logs.
- Based on these readings the billing company can prepare the required bill for the customer.
- The user can be blocked completely from using the Gau-Gas if the bill is not paid by just sending an sms .
- If the sms is read as LOCK then the IOT module will be turned off and to again reboot the IOT module a sms OPEN should be sent .

3.3 Requirements

3.3.1 Hardware Requirements

- Arduino UNO
- GSM module SIM800A
- Current sensor ACS712
- Relay
- PVC Box , Jumper wires and power plugs.

3.3.2 Software Requirements

- Operating system
- Server Apache
- DBMS MySQL

- SDK: Android Studio
- IDE: Arduino IDE, Sublime Text
- Programming Languages: Embedded C, JAVA, PHP, MySQL, HTML 5
- Framework: GetBootstrap
- API: Google charts

3.3.3 Functional Requirements

- Switch Off and On the pump
- Logging the UP and Down Time of Pump
- Transmit the usage statistics to the mobile application through SMS.
- Receive the SMS and push the data to remote DB.
- Authenticate the data sent from application.
- Dashboard/Statistics Display
- SMS Alert to the Provider and Client.
- Generate automated bill on usage logs.
- Control over the IOT Module.

3.3.4 Non – Functional Requirements

Performance

The system must be interactive and the delays involved must be less .So in every action-response of the system, there are no immediate delays. In case of opening windows forms, of popping error messages and saving the settings or sessions there is delay much below 2 seconds, In case of opening databases, sorting questions and evaluation there are no delays and the operation is performed in less than 2 seconds for opening ,sorting, computing, posting > 95% of the files. Also when connecting to the server the delay is based editing on the distance of the 2 systems and the configuration between them so there is high probability that there will be or not a successful connection in less than 20 seconds for sake of good communication.

Safety

Information transmission should be securely transmitted to server without any changes in information.

Reliability

As the system provide the right tools for discussion, problem solving it must be made sure that the system is reliable in its operations and for securing the sensitive details.

Availability

If the SMS service gets disrupted while sending information to the server, the information can be send again for verification.

Chapter 4

DESIGN

In this chapter, the architectural design is presented first and then the functional architecture overview, followed by a detailed description of module diagrams, sequence diagrams and use case diagrams and the functionality table. In our design we are not measuring or controlling the gas flow or pressure rather we are controlling the step up pump which pumps the gas from the gas digester to the burner at a constant rate. Our design results are calibrated against results using a mechanical flow meter and design results almost closer to the reality. The design of the system is discussed below

4.1 Architecture

4.1.1 IOT module

The below illustrates an overview of the sensors connected to Arduino Uno. The system along with programmable sensors like current sensor, humidity sensor, relay, SD card module, GSM module is connected to Arduino Uno. GSM module begins and checks whether the signal is available. Once signal is available, the GSM module will check for any unread messages. If there are no unread messages then the relay will be in HIGH state. The counter starts once the regulator is turned on. After the usage, a message will be sent to the registered number by the GSM module. The counter has three parts i.e., HIGH, LOW, MEDIUM. To determine these values we use current sensor in order to differentiate the current to these three parts. The regulator used is an analogue regulator, and once the current is in that range respective counters will be turned on. The current sensors senses the current regulated through the regulator knob integrated along with the stove replacing the traditional regulator knob present in the stove. The different ranges of currents are specified indicating the low, medium, high modes of regulation and the counter starts and the runtime of burner in particular mode is sent as usage statistics to the android application along with the predefined authentication key. It is of the form J1T*@***\$,2,12,0,32,0,41,0 each of the usage values are zero separated.

Now, if there is an unread message as LOCK the relay will be turned off indicating no further usage of the gas and position of the message will be set. This indicates that the user has paid

his bill. If any message other than LOCK is sent, the relay won't be turned off and the pump will be still being on. Once the customer pays his bill, OPEN message is sent in order to turn on the relay and the received position will be equals to 2. The message will then be cleared.

The Arduino will be reset in order to get the values again. If the message read is anything other than OPEN, the module will still be in locked state. The position will be deleted including the message. The relay will only be turned on when the message received is OPEN. We can remotely control the gau-gas plant through this iot module. Since we are controlling pump itself it is not possible to tamper the iot module and use the gas. The Arduino operates from 5-12v and relay work under 5v and gsm approximately uses 9-12v and it varies. The gsm, adapter runs on the ac current.

4.1.2 Android application

The android application acts as SMS gateway. The android application completely eliminates the SMS gateway which saves up to 2000rs. The registered mobile number is an android device that contains the android application. The android application is installed on the registered or on the company person's mobile phone. The android application reads all the SMS that are received by the mobile phone. Once the SMS read by the android application it notifies in the toast form.

The android application runs as a service on the mobile phone i.e., it runs in the background. The android application validates the all inbox SMS if the authentication code is present or not. Special permission needs to be granted to the application for updating the values to the database.

Once the message is received from the IOT module, the android application first checks for authentication code. Once the authentication code is matched, the usage statistics in the form of SMS is parsed and then stored in the database. Only the message from particular registered user and with the particular format is stored in database.

The android application checks the internet availability if the internet is not available then the data is stored in local database of the mobile phone, once the internet is available application pushes data to database. The data is stored in the local database.

The format of SMS that the android application validates is:

AUTHENTICATION_CODE, PLANT_ID, USAGE_UNITS_OF_GAS_IN_HIGH_MODE,
0,USAGE_UNITS_OF_GAS_IN_MEDIUM_MODE,0,
USAGE_UNITS_OF_GAS_IN_LOW_MODE.

i.e.,J1T*@***\$,2,12,0,32,0,41,0

4.1.3 Web application

The web application is an interactive dashboard, providing various functionality to the admin. These functionalities include add plant, add user, generate bill, usage logs for single user. These functionalities are designed using html, google charts, php etc... the web application reflects the change in the data present in database to graphical representation and it offers additional functionalities.

Firstly when user enters URL: <http://gaugas.vadiedu.com> the web application displays the index page with about us, team, architecture and other details. When user clicks on the login tab a login form is displayed asking the login credentials of the admin and it authenticates the login details. Then map views of the plants that are set in different remote places are displayed.

One of the user ids is selected from the dropdown tab displayed. Then it displays per user usage details and the details of the plant such as the plant id, plant active status, user details such as user id, name, address, and the usage units in centimetre cube.

The runtime of the burner i.e., usage logs is converted into cubic centimetre. Two different interactive dynamic graphs for usage units of a user in daily basis in centimetre cube and also in the uptime of the burner in minutes is displayed from day user gets gas connection.

A button is given to generate a invoice bill based on users usage logs is present and upon click it generates the bill for a user usage of gas for previous month in pdf format.

Separate functionalities, forms for adding new plant and new users to existing plants are present which takes user address, user name details, plant location, plant id etc.... as the form input.

A functionality to revise/update the bill rate is given for admin in settings tab.

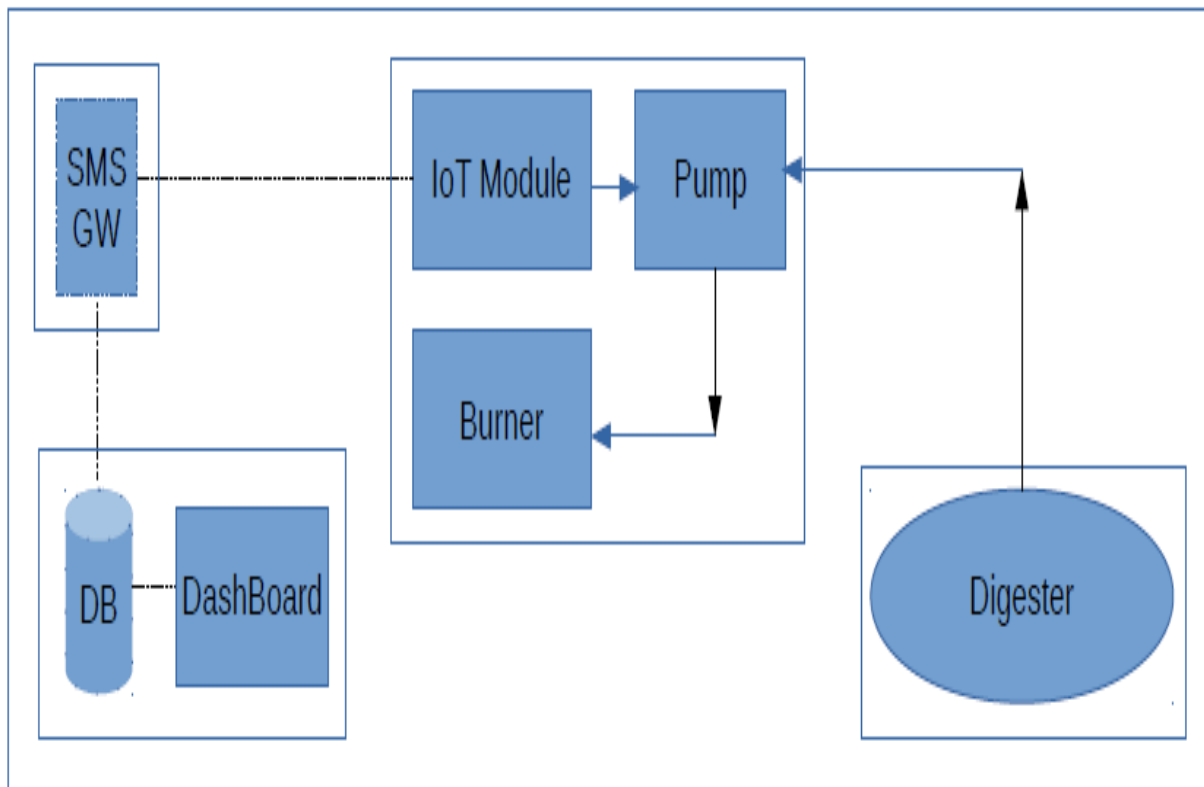


Figure 4.1 Iot Based Metering & Billing Automation System Architecture

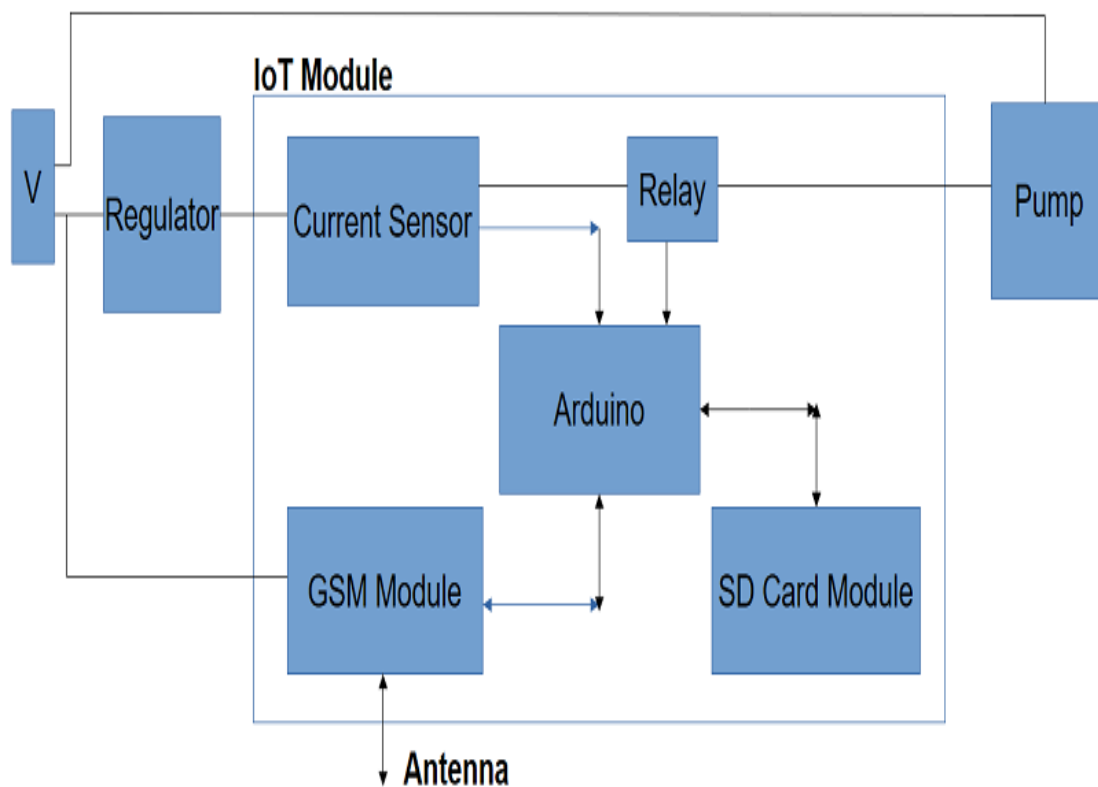


Figure 4.2 IOT module Architecture

4.2 Data Flow Diagrams

Data flow diagrams are graphical representation of the flow of data through an information system. It explains the main and actual logic for each and every component.

4.2.1 User Interaction Phase in Web application

The below Figure shows the data flow diagram for the user interaction phase.

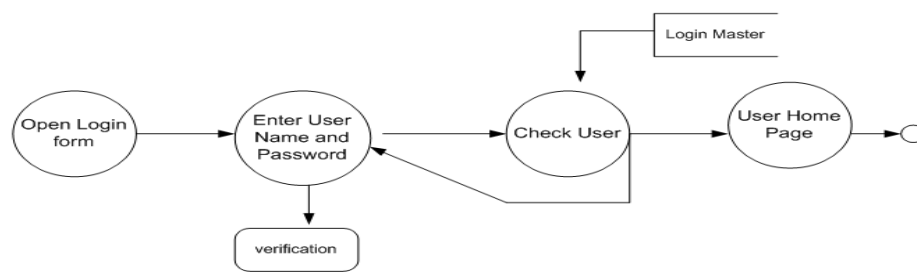


Figure 4.3User interaction phase in web application

- The user login page is displayed to the user
- Username and password is entered.
- User is validated and dashboard is displayed.

4.2.2 Flow inside IoT module

- When the regulator is turned on corresponding usage value is recorded.
- Relay is used to control arduino remotely.
- After the usage, when the regulator is turned off, the usage statistics is sent using GSM module.

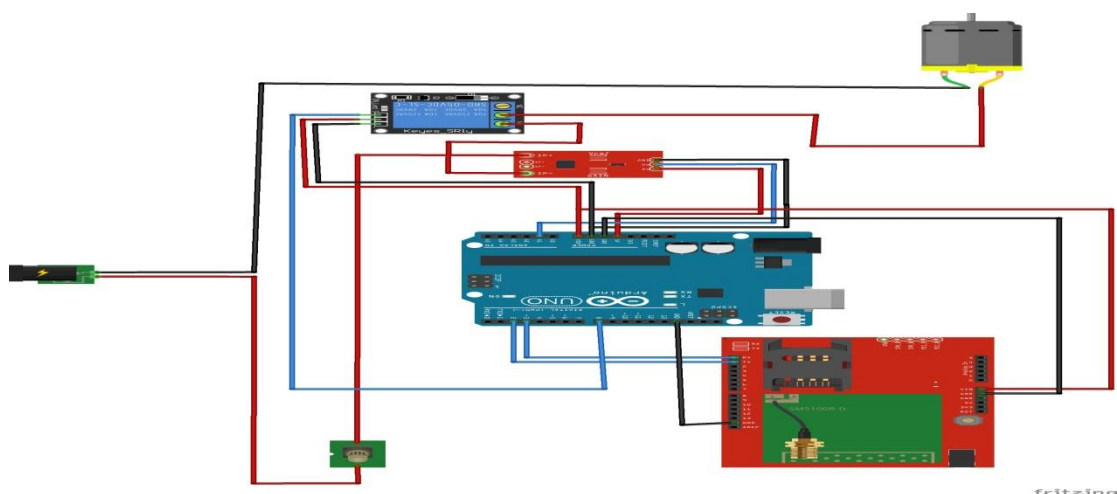


Figure 4.4 IOT pin diagram

4.3 Activity Diagram of overall billing and metering mechanism

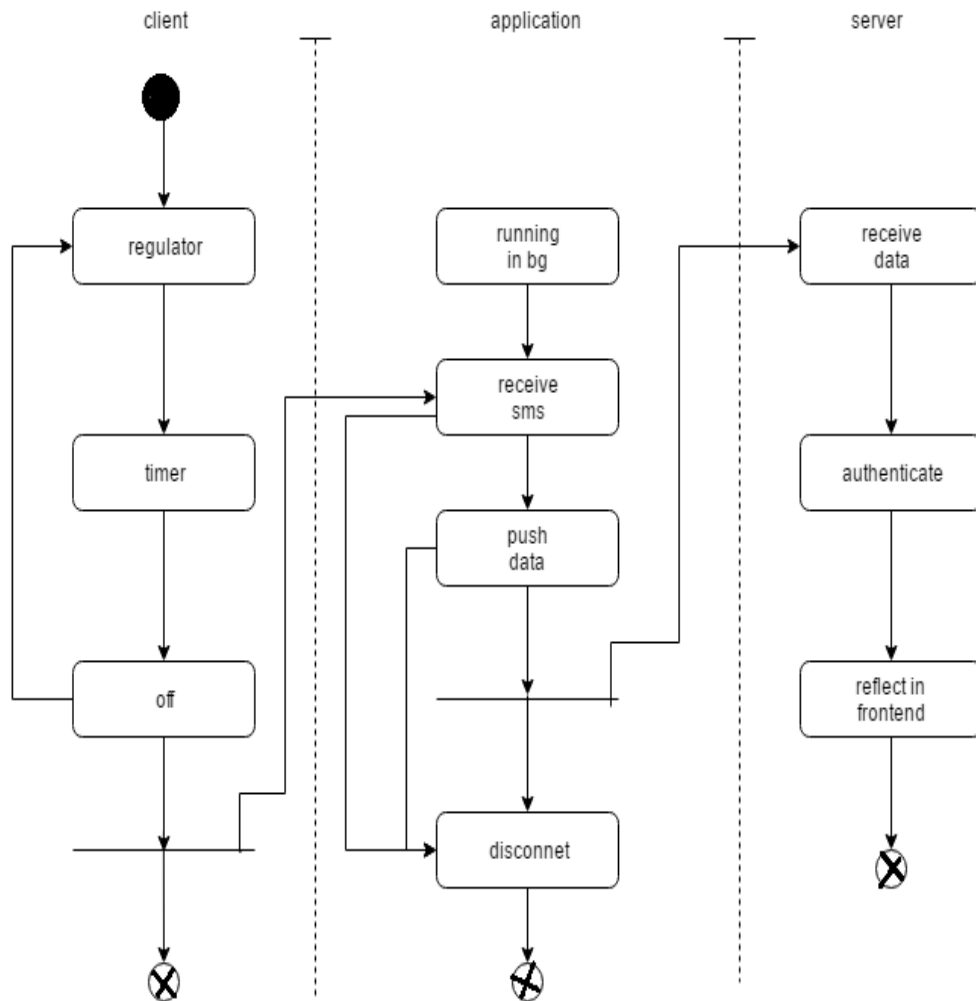


Figure 4.5 Activity diagram of sending SMS from module to Application

- When the regulator is on the timer in background is running.
- When client turns off the regulator the regulator uptime is sent along with the authentication to the android application.
- The mobile application receives the SMS sent by the IOT module.
- The mobile application in the company persons mobile is running in background and validates SMS sent by the Gau gas plant.
- The mobile application then pushes the data sent by the IOT module to the database once internet is available.
- The dashboard fetches the data from the database and authenticates displays on the interactive web page along with the additional functionalities.

4.4 ER Diagram of the database used in billing and metering mechanism

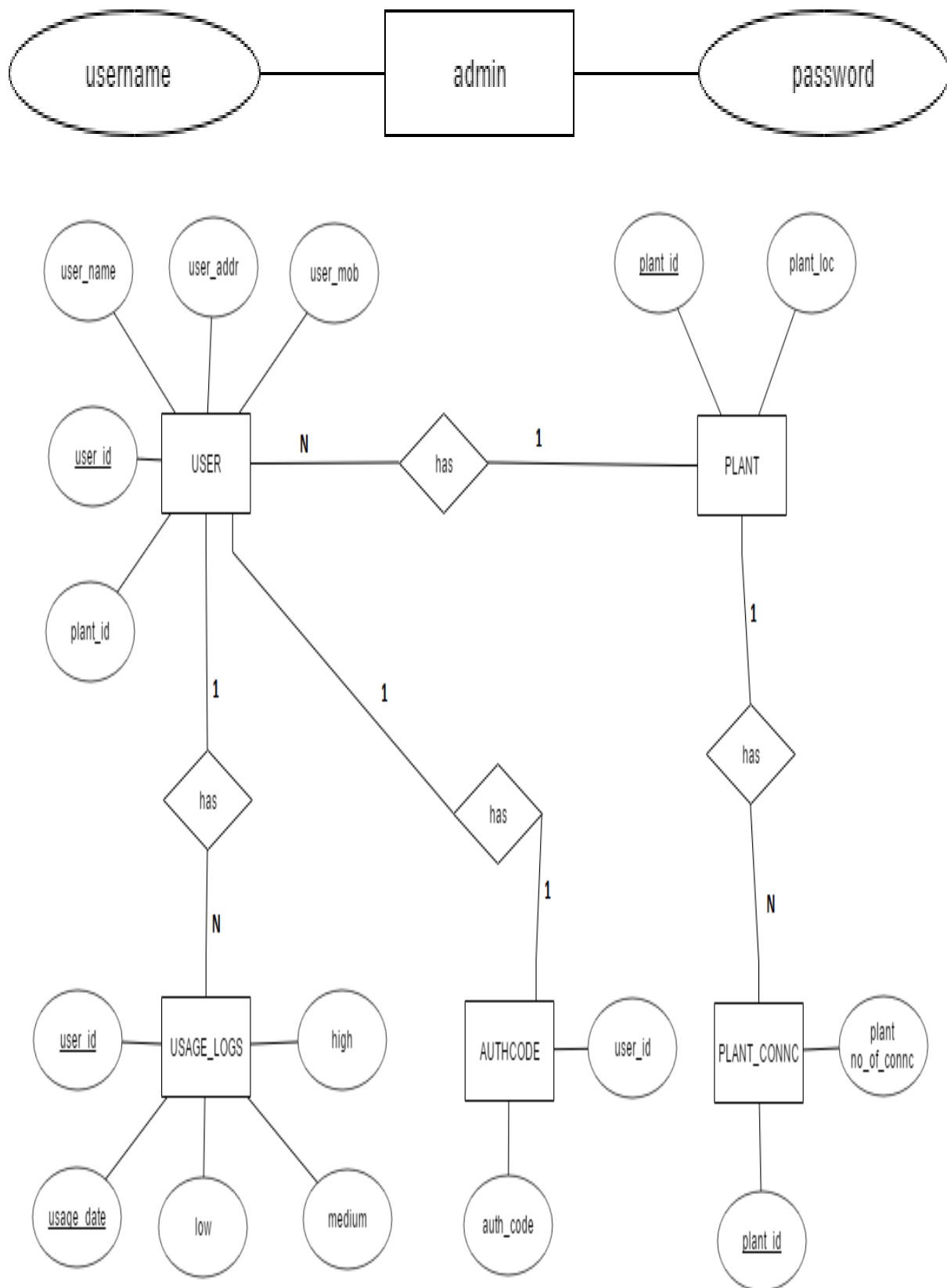


Figure 4.6 ER diagram of database

4.5 Database design

The database gaugas comprises of five tables running in apache server .

The five tables are:

- User
- Plant
- Usage logs
- Authcode
- Admin
- The user table is used to store the user complete details like user id, user name, contact number, contact address and the pant id to which the user is connected. The plant table is used to store the details of plant plant id, plant location and total number of connections to each plant.
- Usage logs is used to update live usage logistics data from plant, the table stores detail of user id, usage date, usage vales in three different modes of low, medium and high.
- Authcode table has user id and associated authentication code for each user.

Admin table has two columns of user name and password, This table is accessed during login time to authenticate registered user to dashboard.

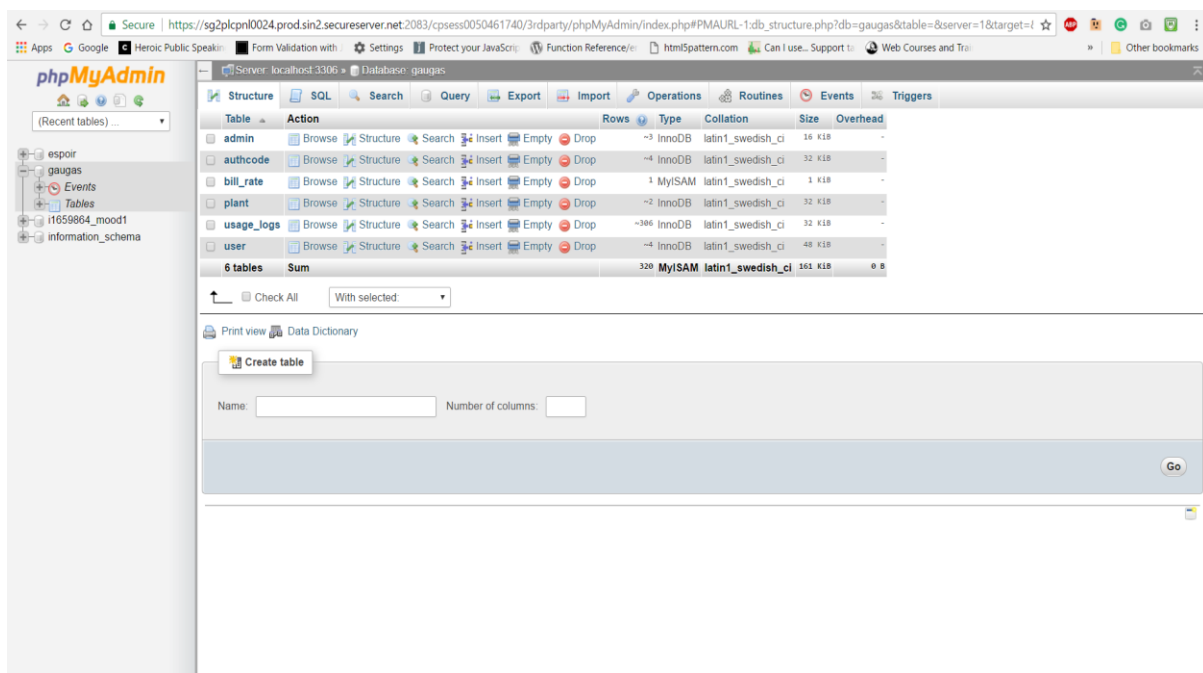


Figure 4.7 C-panel Database

4.6 Functionality table

IOT module	<ul style="list-style-type: none"> • Senses the regulator current. • Records the regulator usage in different levels. • Sends the usage logs in the form of SMS along with auth code. • Locking and open mechanism i.e., remote controlling plant.
Android application	<ul style="list-style-type: none"> • Runs in background as a service. • Receives the incoming messages from the IOT module • Validates the message for auth code and sends/pushes those messages i.e., usage logs of user in database. • Check if internet is available, if not store in local database.
Web application	<ul style="list-style-type: none"> • The dashboard fetches the data from the database and authenticates displays on the interactive web page along with the additional functionalities. • Functionality to generate bill. • Functionality to add user, plant. • Functionality to update price. • Graphical interactive display of per user per day usage logs in both cc and minutes.

Table 4.1 Functions

Chapter 5

Implementation

5.1 Agile Software Development

Agile development methodology provides opportunities to assess the direction of a project throughout the development lifecycle. This is achieved through regular cadences of work, known as sprints or iterations, at the end of which teams must present a potentially shippable product increment.

The main principles of Agile methodology are

- Customer Involvement.
- Pair programming.
- Refactoring.
- Incremental development.

In our project we are going to use Agile methodology for project management

5.1.1 Customer Involvement

Gau Gas systems are using by many people in various parts of our country like Tirupati and Gujarat.

We had an official meeting with the Gau Gas team where we Explained our solution to current system problem .

5.1.2 Pair Programming

We have been split into two teams where one team working with Arduino to make the controller board and the other will be working on SMS oriented solution and statistics display.

5.1.3 Incremental development

Gau Gas systems is a live working system our aim is to make it completely automation eliminating the manual interaction .

5.1.4 Refactoring

Improving objective attributes of code (length, duplication, coupling and cohesion, cyclomatic complexity) that correlate with ease of maintenance

5.2 Sensors and Modules

A current sensor is a device that detects electric current(AC or DC) in a wire, and generates a signal proportional to it. The generated signal could be analog voltage or current or even digital output. It can be then utilized to display the measured current in an ammeter or can be stored for further analysis in a data acquisition system or can be utilized for control purpose.

Device Name	Current Sensor
Function	Sensing Current at regular intervals

Description: This function is used to detect AC and DC currents based on RMS value



Figure 5.1 ACS 712

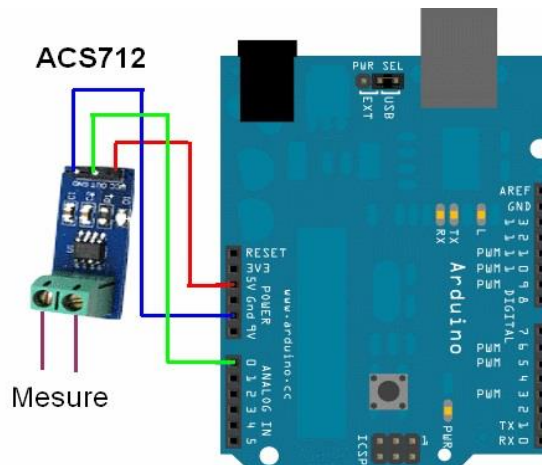


Figure 5.2 ACS 712 connected to Uno

ACS 712 allows measurement of current -direct or alternating- flowing in a conductor. The desired measured current generates a magnetic field which the sensor converts to a proportional output voltage, using the Hall effect. This voltage in turn is read by a microcontroller system through an A/D converter to calculate its peak value and the corresponding RMS value of the load current.

- There are 3 versions for the ACS712 sensor, for ranges of 5, 20 and 30 amperes.
- Using the ACS 712 current sensor we always will be able to get the **root mean square** value of the current signal either AC or DC. The **root mean square** (abbreviated **RMS** or **rms**) is defined as the square root of mean square.

- For a cyclically alternating electric current, RMS is equal to the value of the direct current that would produce the same average power dissipation in a resistive load.

Connections

- Signal Conditioning Circuit Board input side GND, Signal, and +5VDC lines to GND, OUT and VCC pins of ACS712 module.
- Signal Conditioning Circuit Board output side GND +5VDC lines to 5VDC power source.
- Signal Conditioning Circuit Board output side GND and A/D pin lines to Multimeter set to VDC range.
- Extension cord female plug to a variable AC load - I used a 3 way light bulb (50W, 200W & 250W).
- Extension cord male plug.to Kill A Watt meter (set to AC Current range) which is then plugged into AC wall outlet.

Test method

Gradually increase the AC load. The VDC at the multimeter should increase as the AC is increased, as should the Kill A Watt meter Amp readings.Adjust the Trim Pot of the Signal Conditioning Circuit Board so that at with no AC load the VDC signal is around zero. You may not be able to get it all the way down to zero VDC.

Calibration

Using the above test setup, apply a variable AC load and measure AC current load (AMPS) and the VDC signal output. This is your calibration data, with the volts being the "X" value and amps the "Y" value. Plug this data into a spreadsheet or a pocket calculator with linear regression function to determine the trend line equation.

Device Name	Relay
Function	It is an electrically operated switch.

Description: It is activated by a current or signal in one circuit to open or close another circuit.

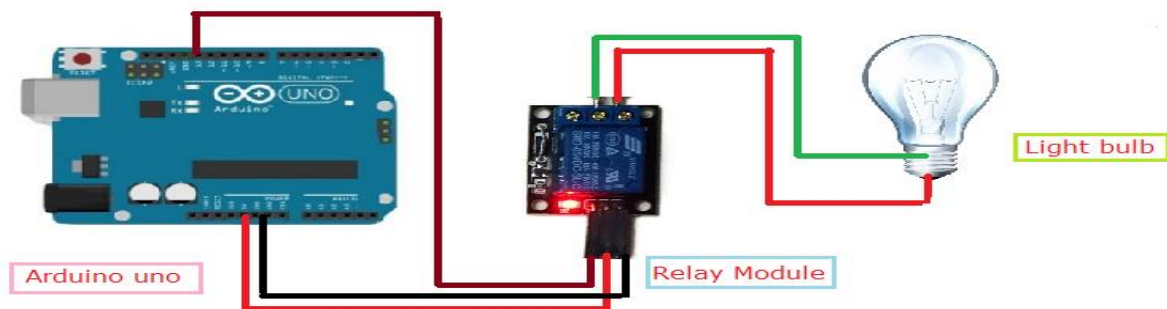


Figure 5.3 Relay Integration

- AC is alternating current 220v (india) which powers the ac lights. Arduino cannot control high volt n amp, but a relay can do this job, which is the sole design of it. so we are using relay as switch to control high power devices.
- NO NC and COM in relay?
- C = Common Connection → it is the center terminal.
 NC = Normally Closed Connection → It is a switch & no connection between Com & NO
 NO = Normally Open Connection → It is always in contact with COM, even when relay is not powered.

Connections

- Hot line from supply is connected to COM
- Supply line to the Ac light is connected to NO
- Gnd or - or other terminal in light is connected directly.

Test Method and Calibration

The Red light on the Relay board turns on when power is applied (via the VCC pin). When power is applied to one of the Channel pins, the respective green light goes on, plus the relevant relay will switch from NC to NO. When power is removed from the channel pin, the relay will switch back to NC from NO. In this sketch we see that power is applied to both LEDs in the setup() method. When there is no power applied to the CH1 pin, the yellow LED will be on, and the Green LED will be off. This is because there is a break in the circuit for the green LED. When power is applied to CH1, the relay switches from NC to NO, thus closing the circuit for the green LED and opening the circuit for the yellow LED. The green LED turns on, and the yellow LED turns off.

Device Name	GSM Module
Function	Specialized type of modem which accepts a SIM card, and operates over a subscription to a mobile operator, just like a mobile phone.

Description: This function is used to check whether the module is getting signal or not.

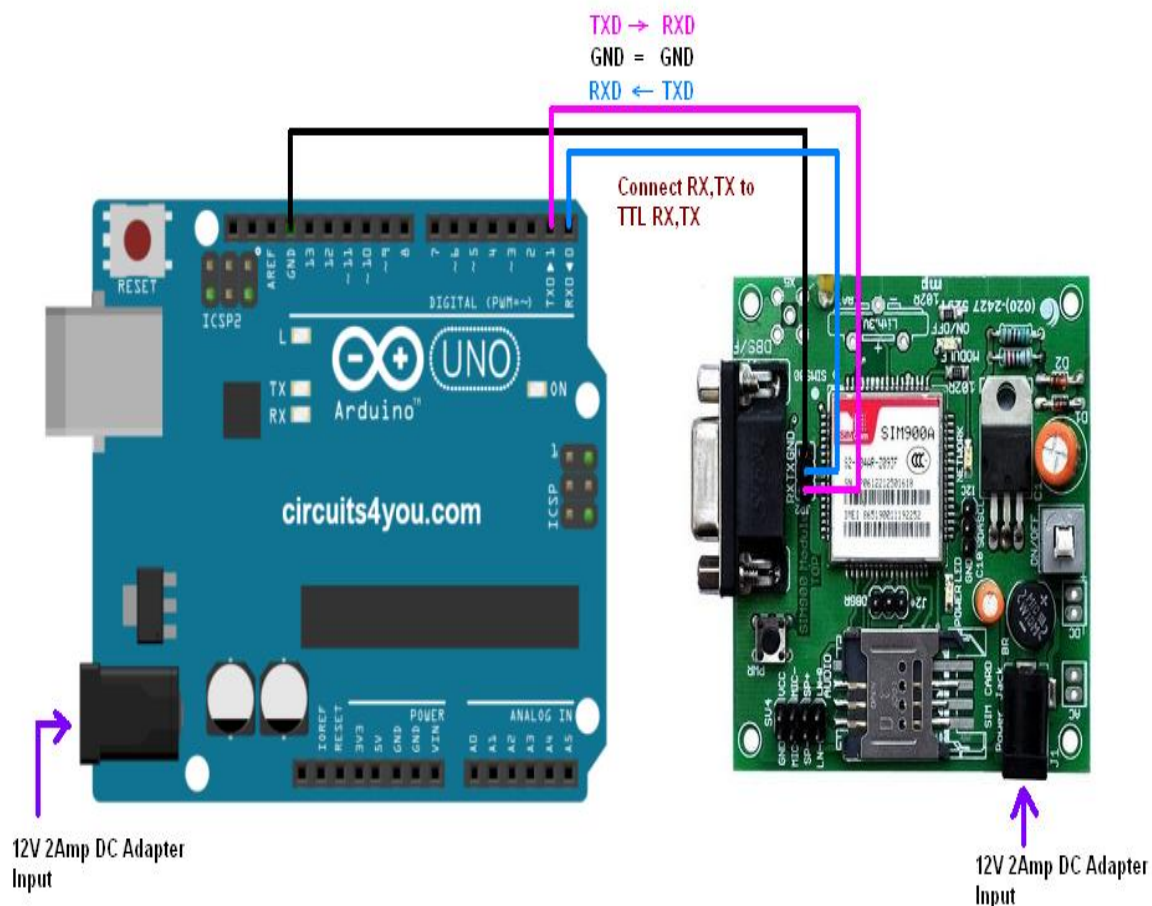


Figure 5.4 GSM SIM 900A Integration with Uno

SIM900A Modem is built with Dual Band GSM/GPRS based SIM900A modem from SIMCOM. It works on frequencies 900/ 1800 MHz. SIM900A can search these two bands automatically. The frequency bands can also be set by AT Commands. The baud rate is configurable from 1200-115200 through AT command. The GSM/GPRS Modem is having internal TCP/IP stack to enable you to connect with internet via GPRS. SIM900A is an ultra compact and reliable wireless module. This is a complete GSM/GPRS module in a SMT type and designed with a very powerful single-chip processor integrating AMR926EJ-S core, allowing you to benefit from small dimensions and cost-effective solutions.

Specification

- Dual-Band 900/ 1800 MHz
- GPRS multi-slot class 10/8GPRS mobile station class B
- Dimensions: 24*24*3 mm
- Weight: 3.4g
- Control via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- Supply voltage range : 5V
- Low power consumption: 1.5mA (sleep mode)
- Operation temperature: -40°C to +85 °

Power requirements of GSM module

GSM modules are manufactured by different companies. They all have different input power supply specs. You need to double check your GSM modules power requirements. In this tutorial, our gsm module requires a 12 volts input. So we feed it using a 12V,1A DC power supply. Other gsm modules which require 15 volts and some other which needs only 5 volts. They differ with manufacturers. If you are having a 5V module, you can power it directly from Arduino's 5V out.

Booting Up the GSM

- Insert the SIM card to module and lock it.
- Connect the adapter to module and turn it ON!
- Now wait for some time (say 1 minute) and see the blinking rate of 'status LED' (GSM module will take some time to establish connection with mobile network).
- Once the connection is established successfully, the status LED will blink continuously every 3 seconds.

Basic AT Command

- To change sms sending mode :**AT+CMGF=1**

```
mySerial.println("AT+CMGF=1");
```

- To read SMS in text mode :**AT+CNMI=2,2,0,0,0**

```
mySerial.println("AT+CNMI=2,2,0,0,0");
```


- To make a call :**ATD+60XXXXXXXXXX;** //replace X with number you want to call, change +60 to your country code

```
mySerial.println("ATD+60XXXXXXXXXX;");
```

- To disconnect / hangupcall :**ATH**

```
mySerial.println("ATH");
```

- To redial :**ATDL**

```
mySerial.println("ATDL");
```

- To receive a phone call : **ATA**

```
mySerial.println("ATA");
```

Device Name	GSM Module
Function	Specialized type of modem which accepts a SIM card, and operates over a subscription to a mobile operator, just like a mobile phone.

Description: This function is used for calling to other mobile phone with sim.

Power requirements of GSM module

GSM modules are manufactured by different companies. They all have different input power supply specs. You need to double check your GSM modules power requirements. In this tutorial, our gsm module requires a 12 volts input. So we feed it using a 12V,1A DC power supply. Other gsm modules which require 15 volts and some other which needs only 5 volts. They differ with manufacturers. If you are having a 5V module, you can power it directly from Arduino's 5V out.

Booting Up the GSM

- Insert the SIM card to module and lock it.
- Connect the adapter to module and turn it ON!
- Now wait for some time (say 1 minute) and see the blinking rate of 'status LED' (GSM module will take some time to establish connection with mobile network).

- Once the connection is established successfully, the status LED will blink continuously every 3 seconds.

Function – Calling

- As you key-in **c : to make a call**, gsm will read the ATD command and make a call to a phone number you have upload in your source code.
- When you key-in **h : to disconnect/hangup call**, gsm will read the ATH command and disconnect the connection.
- When you key-in **e : to redial**, gsm will read the ATDL command and redialing previous number it has called.
- When there is an incoming call, you can see RING printed on serial monitor and you can click **i : to receive a call** and GSM's ATA command will be carried out and you will be connected to a call connection.

Device Name	GSM module
Function	Specialized type of modem which accepts a SIM card, and operates over a subscription to a mobile operator, just like a mobile phone.

Description: This function is used to send sms to other mobile phones and receive sms.

Power requirements of GSM module

GSM modules are manufactured by different companies. They all have different input power supply specs. You need to double check your GSM modules power requirements. In this tutorial, our gsm module requires a 12 volts input. So we feed it using a 12V,1A DC power supply. Other gsm modules which require 15 volts and some other which needs only 5 volts. They differ with manufacturers. If you are having a 5V module, you can power it directly from Arduino's 5V out.

Booting Up the GSM

- Insert the SIM card to module and lock it.
- Connect the adapter to module and turn it ON!
- Now wait for some time (say 1 minute) and see the blinking rate of 'status LED' (GSM module will take some time to establish connection with mobile network).

- Once the connection is established successfully, the status LED will blink continuously every 3 seconds.

Function -Send and Receive SMS

- Key-in s to send SMS. Receptient's number and text message printed on serial monitor.
NOTE : You can edit the receptient's phone number and text message on your source code.
- When gsm receive a message, text message and number will be printed on serial monitor.

Device Name	SD Card Module
Function	Secure Digital (SD) is a non-volatile memory card format developed by the <i>SD Card Association</i> (SDA) for use in portable devices.

Description: To initialize the SD card module.

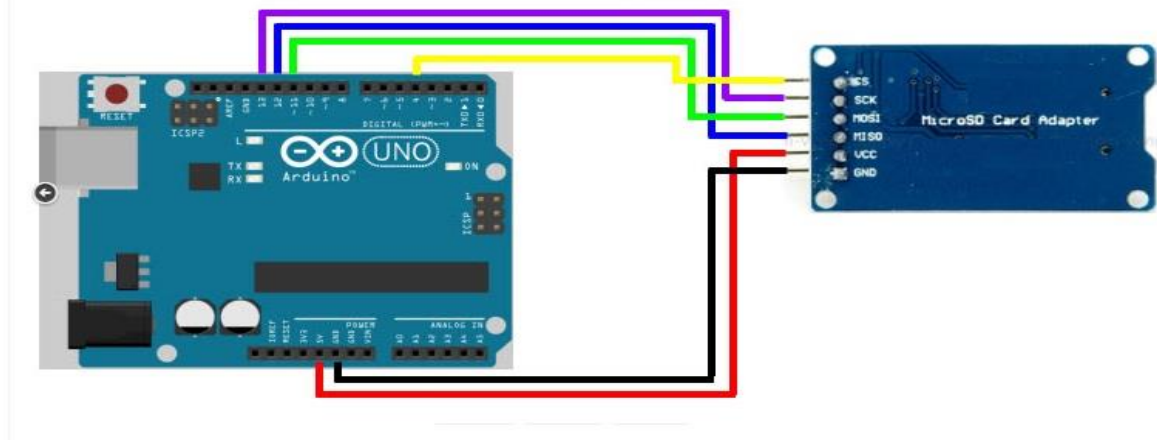


Figure 5.5 SD card module connected to Uno

The SD Card Module is a simple solution for transferring data to and from a standard SD card. The pinout is directly compatible with Arduino, but can also be used with other microcontrollers. This module has SPI interface which is compatible with any sd card and it use 5V or 3.3V power supply which is compatible with Arduino UNO/Mega. SD module has various applications such as data logger, audio, video, graphics.

Connections

- Vcc - 5V
- Gnd – Gnd

- MOSI - pin 11
- MISO - pin 12
- CLK - pin 13
- CS - pin 4

Meanings of the pin designations

- CS = Chip Select
- MOSI = Master Out, Slave In
- MISO = Master In, Slave Out
- SCK = Slave Clock

Initial Bootup & Read/Write from SD card

- In the setup(), we call SD.begin(), naming pin 4 as the CS pin. This pin varies depending on the make of shield or board you are using.
- In setup(), create a new file with SD.open() named "test.txt". FILE_WRITE enables read and write access to the file, starting at the end. If a file "test.txt" was already on the card, that file would be opened.
- Name the instance of the opened file "myFile".
- Once opened, use myFile.println() to write a string to the card, followed by a carriage return. Once the content is written, close the file.
- Again, open the file with SD.open(). Once opened, ask the Arduino to read the contents of the file with SD.read() and send them over the serial port. After all the contents of the file are read, close the file with SD.close().

5.2.1 Circuit Diagram

- The various signals that passes through the IOT module can be seen .
- The components are Arduino uno , Gsm module , Current sensor board , relay switch board , power plug , regulator and finally the pump .
- Uno acts as a stopwatch and collector of the usage logs .
- Gsm module is used to send the stastics recorded by the uno .
- Current sensor is used to analyse the current flow by the regulator .
- Relay is used as a Lock to turn on/off the IoT Module if the user has not paid the bill .
- Finally the power plug which is used to power up the IoT module.

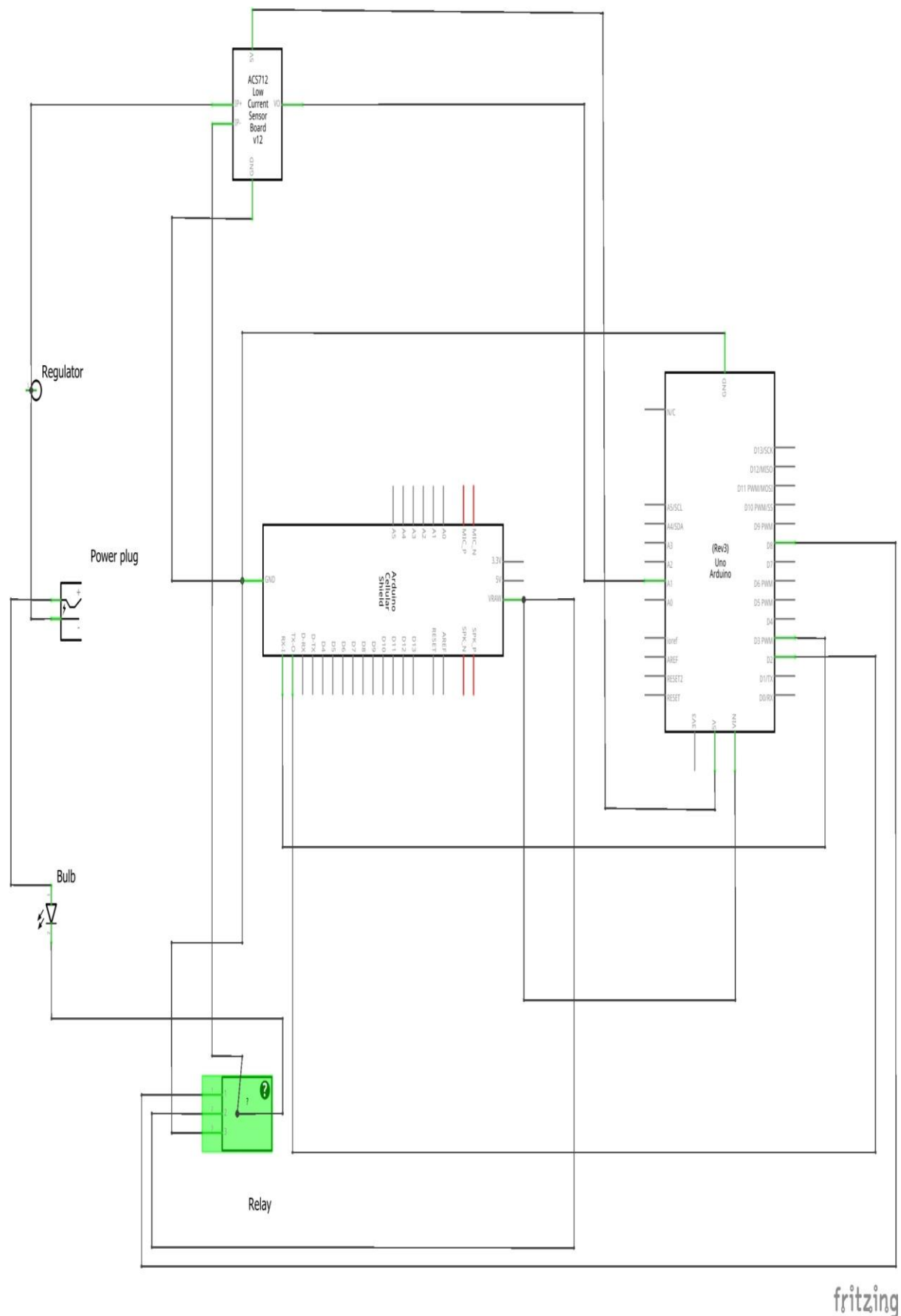


Figure 5.6 Pin Diagram

5.3 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support

5.3.1 Project Structure

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files. All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

- **manifests:** Contains the AndroidManifest.xml file.
- **java:** Contains the Java source code files, including JUnit test code.
- **res:** Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown.

We can also customize the view of the project files to focus on specific aspects of our app development.

For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

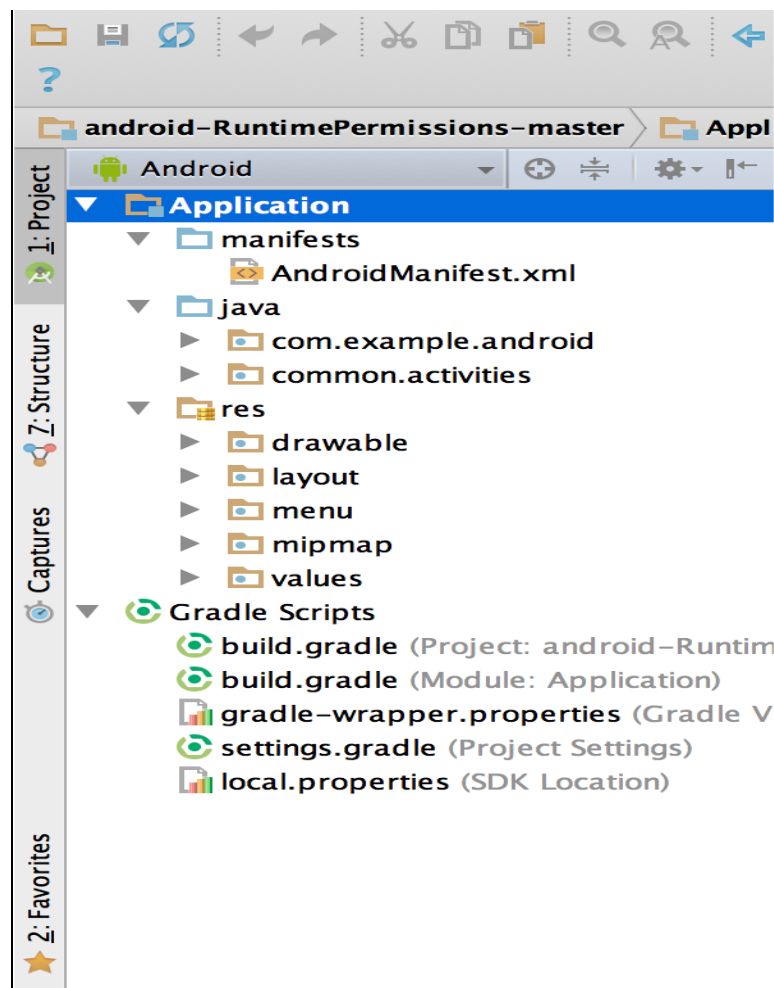


Figure 5.7 Project Structure In Android Studio

5.3.2 User Interface

- The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.

- The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
- The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
- The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
- The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.

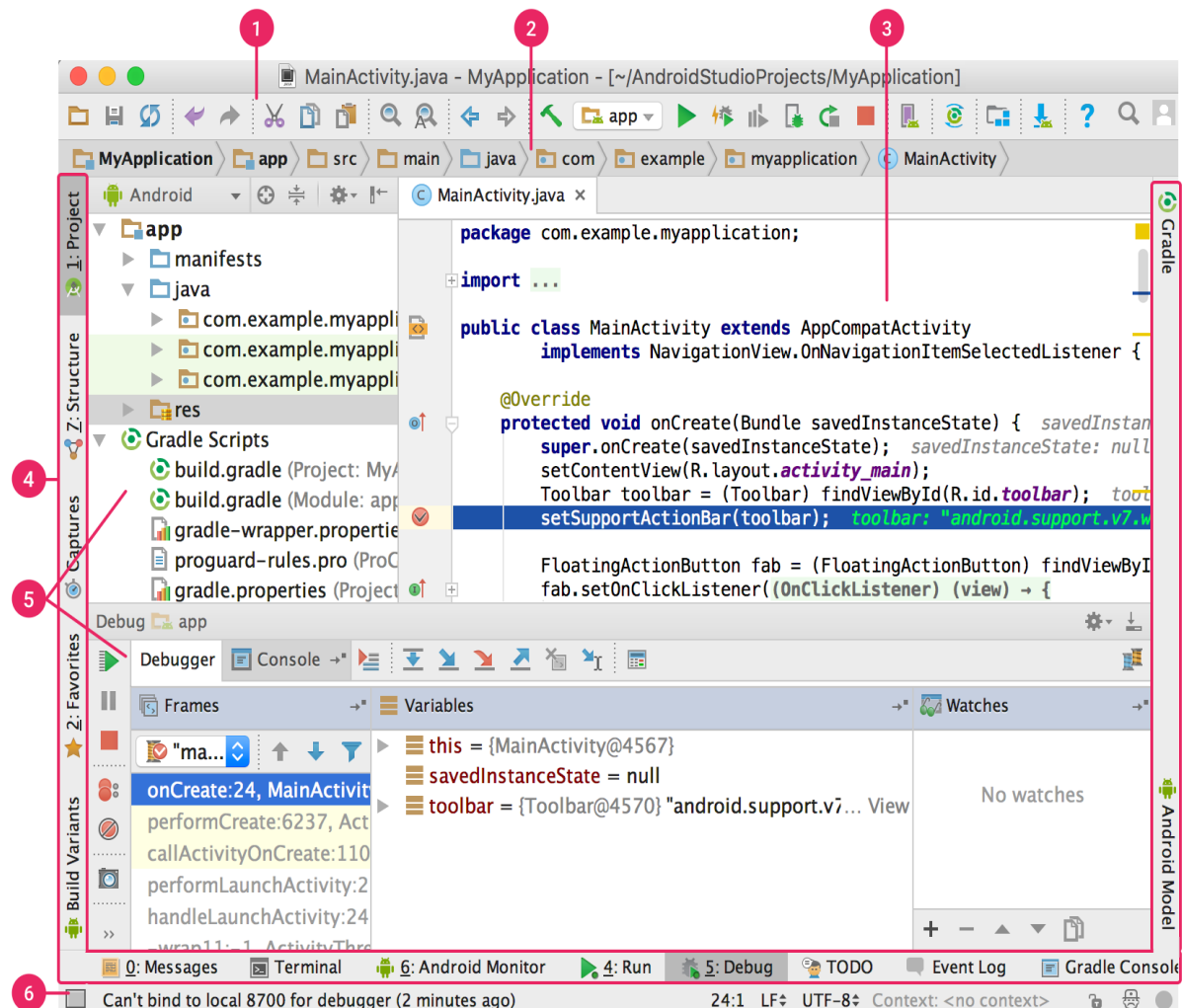


Figure 5.8 User Interface in Android

5.3.3 Gradle Build System

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the Android plugin for Gradle. This build system runs as an integrated tool from the Android Studio menu, and independently from the command line.

You can use the features of the build system to do the following:

- Customize, configure, and extend the build process.
- Create multiple APKs for your app, with different features using the same project and modules.
- Reuse code and resources across source sets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files. Android Studio build files are named `build.gradle`. They are plain text files that use Groovy syntax to configure the build with elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

5.3.4 Build Variants

The build system can help you create different versions of the same application from a single project. This is useful when you have both a free version and a paid version of your app, or if you want to distribute multiple APKs for different device configurations on Google Play.

5.3.5 Layouts

- **LinearLayout** is a view group that aligns all children in a single direction, vertically or horizontally.
- **RelativeLayout** is a view group that displays child views in relative positions.
- **TableLayout** is a view that groups views into rows and columns.
- **AbsoluteLayout** enables you to specify the exact location of its children.
- The **FrameLayout** is a placeholder on screen that you can use to display a single view.

5.3.6 AndroidManifest.xml

- Add permission to receive, read and write SMS to the application.
- Give the permission to Connect the android application to the Internet through wifi or cellular data.
- Then add the broadcast receiver with an intent-filter to receiving SMS.

5.3.7 SmsBroadcastReceiver.java

- This is a BroadcastReceiver which receives the intent-filtered SMS messages.
- onReceive, we extract the SMS message bundle and show a toast message and also update the UI by adding the SMS message to the SMS inbox list.
- When a SMS message arrives, the inbox is automatically refreshed.
- The incoming SMS will then be sent across the POST URL , over Internet once the authentication code is confirmed .

5.3.8 SmsActivity.java

- This is the main Android activity of the SMS application.
- It acts as the SMS inbox.
- It has a ListView to show the SMS messages.
- onCreate we read all the messages present in the internal SMS inbox from its Uri and show them using the ListView.

5.3.9 activity_sms.xml

- Text View Shows the received number , and the message with content wrap .
- List View is used to separate the each message received into a list .

5.4 Web Application

5.4.1 Server Apache

- Apache is the most widely used web server software.
- Developed and maintained by Apache Software Foundation, Apache is an open source software available for free.
- It runs on 67% of all web servers in the world. It is fast, reliable, and secure.
- It can be highly customized to meet the needs of many different environments by using extensions and modules.
- Most WordPress hosting providers use Apache as their web server software. However, WordPress can run on other web server software as well.
- A web server is the software that receives your request to access a web page. It runs a few security checks on your HTTP request and takes you to the web page.

5.4.2 Database-MySQL

- MySQL is the most popular Open Source Relational SQL Database Management System.
- MySQL is one of the best RDBMS being used for developing various web-based software applications.
- MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company.

5.4.3 Sublime Text

Sublime Text is a proprietary cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages, and its functionality can be extended by users with plugins, typically community-built and maintained under free-software licenses.

5.4.4 Front End

Web application Frontend is developed using HTML , CSS, Java Script Technologies to make web application responsive to provide an optimal viewing experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices (from mobile phones to desktop computer monitors) Bootstrap frame work is implemented. .Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

To make the application more interactive to user google charts have been used to show the statistical graphs and responsive maps.

5.4.5 Back End

The Technologies used to develop the backend part of the application is PHP and MySQL.The application is hosted and running in an apache server.

core computational logic of a website is developed using php scripting language. The scripting are written in MySQLi procedural way. various PHP built in functions are used to manipulate the database entries and to update the tables.

The Database and the corresponding tables are created using graphical user interface of apache server.

Chapter 6

SOFTWARE TESTING

Software Testing is a process of investigation conducted for the purpose to provide stakeholders with information about quality of the software or hardware product and even service under test. Even testing can provide an independent, objective view of the product to allow the business to understand and risk involved in the implementation of software development.

In our project for testing purpose we have followed manual testing method. For this purpose we have replaced the pump with a 10v bulb. The testing was done using bulb as a load and regulating the flow of current to find out suitable required range of current. It is shown below:

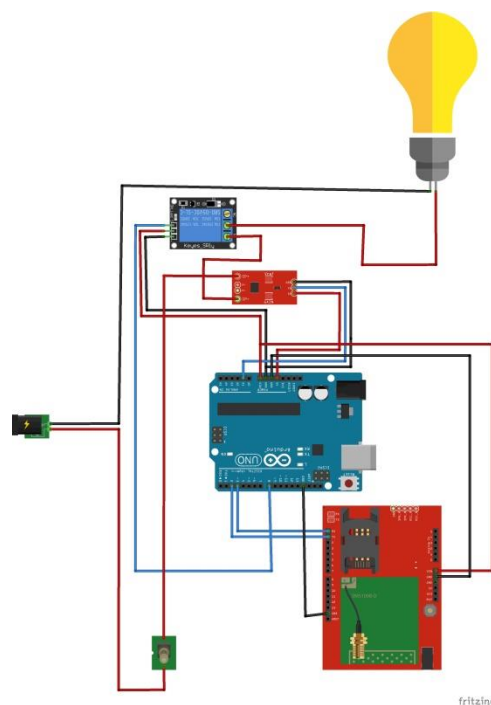


Figure 6.1: circuit diagram with bulb

For calibration purpose we made use of flow meter which measures the flow of gas within the pipe. With flow meter as a measuring unit we calculated the time taken for the gas to flow for different interval of time. It is as follows:

TEST 1:

Number of Test case	Time	Cubic centimeter
Test case 1	60 seconds	6.9
Test case 2	60 seconds	6.9
Test case 3	59 seconds	6.7

Table 6.1: Calibration for 60 seconds**TEST 2:**

Number of Test case	Time	Cubic centimeter
Test case 1	300 seconds	34.9
Test case 2	300 seconds	34.1
Test case 3	300 seconds	33.9

Table 6.2: Calibration for 300 seconds**TEST 3:**

Number of Test case	Time	Cubic centimeter
Test case 1	600 seconds	69
Test case 2	600 seconds	68.7
Test case 3	600 seconds	69.2

Table 6.3: Calibration for 600 seconds

By the above calibration we measure the amount of gas flow in a given particular time.

The process of testing can be grouped from the perspective they are added in the development of software product, or by indicating the level of testing. The important level during any process of development is as: system testing, unit testing, and integration testing.

- Unit Testing
- Integration Testing and
- System testing.

Unit Testing

- Unit test focuses verification effort on the smallest unit of software design component or module.

Test Case ID	Unit Test Case 1
Description	Controlling pump using regulator
Input	Current
Expected Output	Pump turn off when regulator supplies zero current
Actual Output	Pump gets turned off using regulator
Remarks	Pass

Table 6.4: Verification of Unit Test Case for controlling pump

Test Case ID	Unit Test Case 2
Description	Calculating total run time of pump
Input	Current
Expected Output	Number of seconds the pump was running.
Actual Output	Total seconds the pump in ON state.
Remarks	Pass

Table 6.5: Verification of Unit Test Case for calculating run time

Test Case ID	Unit Test Case 3
Description	Transmit usage statistics to mobile application.
Input	Usage log.
Expected Output	Mobile application receiving SMS from IoT module.
Actual Output	Mobile application receives SMS.
Remarks	Pass

Table 6.6: Verification of Unit Test Case for sending usage logs

Test Case ID	Unit Test Case 4
Description	Pushing the received SMS to remote DB.
Input	SMS from IoT module.
Expected Output	Database should be updated with new values.
Actual Output	SMS is pushed into database and database is updated.
Remarks	Pass

Table 6.7: Verification of Unit Test Case for pushing data to database

Test Case ID	Unit Test Case 5
Description	Authenticate data sent from mobile application.
Input	SMS with authentication code.
Expected Output	Data should be authenticated and pushed to database.
Actual Output	Data is authenticated and pushed to database.
Remarks	Pass

Table 6.8: Verification of Unit Test Case for authentication of data

Test Case ID	Unit Test Case 6
Description	Remotely controlling IoT module.
Input	SMS from admin.
Expected Output	IoT modules should stop the flow of gas.
Actual Output	IoT module stops the flow of gas.
Remarks	Pass

Table 6.9: Verification of Unit Test Case for remote control of IoT module

Integration Testing

- Integration tests are designed to test integrated software components to determine if they actually run as one program.

Test Case ID	Integration Test Case 1
Description	Turning of the pump and getting the log.
Input	SMS with authentication code.
Expected Output	The pump should be off and the logs should be displayed.
Actual Output	The pump was off and the logs were displayed.
Remarks	Pass

Table 6.10: Verification of Integration Test Case for getting log

Test Case ID	Integration Test Case 2
Description	Sending usage logs to mobile application and pushing it to database
Input	SMS sent from IoT module.
Expected Output	SMS should be received and database should be updated.
Actual Output	SMS is received and the database is updated.
Remarks	Pass

Table 6.11: Verification of Integration Test Case pushing data

Test Case ID	Integration Test Case 3
Description	Retrieving data from database and generating bill.
Input	Usage logs.
Expected Output	Dashboard should be updated and bill should be generated.
Actual Output	Dashboard is updated and bill gets updated.
Remarks	Pass

Table 6.12: Verification of Integration Test Case for generating bill

System Testing

- Testing performs a very critical role for quality assurance and for ensuring reliability of the software. During testing, the program should be tested with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as expected. Testing forms the first step in determining the errors in a program. The success of testing in revealing errors in program depends critically on the test cases.
- Testing a large system is a complex activity and hence it has to be broken into smaller activities, due to this incremental testing is generally performed for project in which components and sub-systems of a system are tested separately before integrating them to form the system for system testing. Integration of various components of the system is an important issue that the testing phase has to deal with.

Test Case ID	System Test Case 1
Description	Controlling the pump and sending logs to mobile application.
Input	Usage logs.
Expected Output	Logs should be updated in database.
Actual Output	Logs gets updated in database.
Remarks	Pass

Table 6.13: Verification of System Test Case for receiving logs

Test Case ID	System Test Case 2
Description	Generating and controlling IoT module remotely.
Input	Usage logs and SMS.
Expected Output	Bill should be generated and IoT module should be locked .
Actual Output	Bill is generated and IoT module is locked.
Remarks	Pass

Table 6.14: Verification of System Test Case for bill generation .

Chapter 7

RESULTS AND SNAPSHOTS

This chapter includes pictures of the output of all the functional requirements and non functional requirements.

The functional requirements achieved are

- Transmitting the usage stastics to the remote DB through SMS
- Interactive charts and graphs to show Statistics of usage logs
- Generate automated bill on usage logs.
- Remotely controlling the IoT Module.
- Adding new user and new plant to the database.

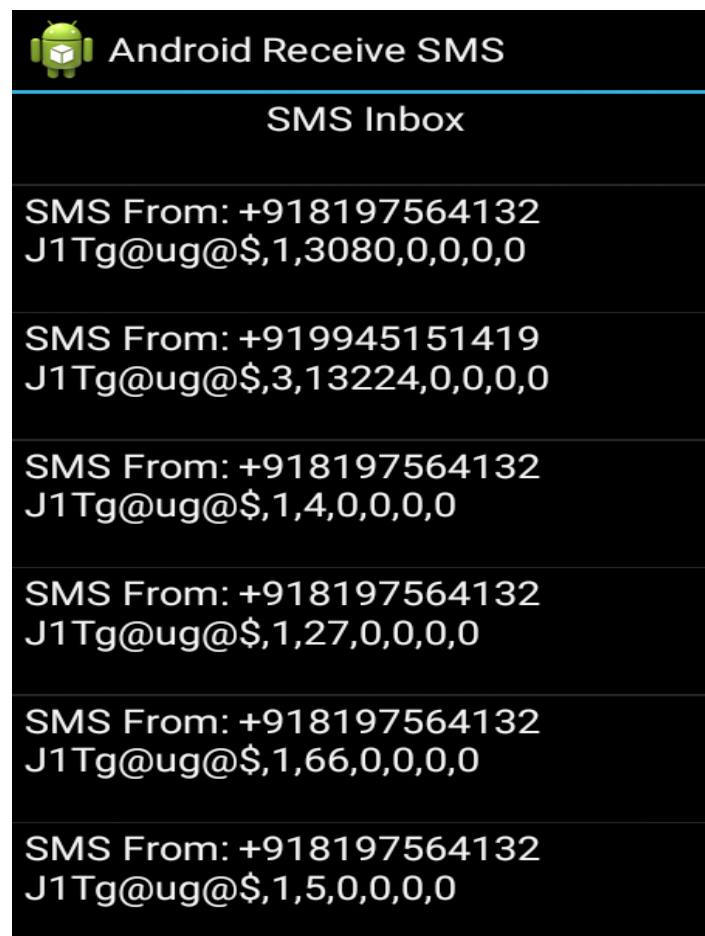


Figure 7.1Android application

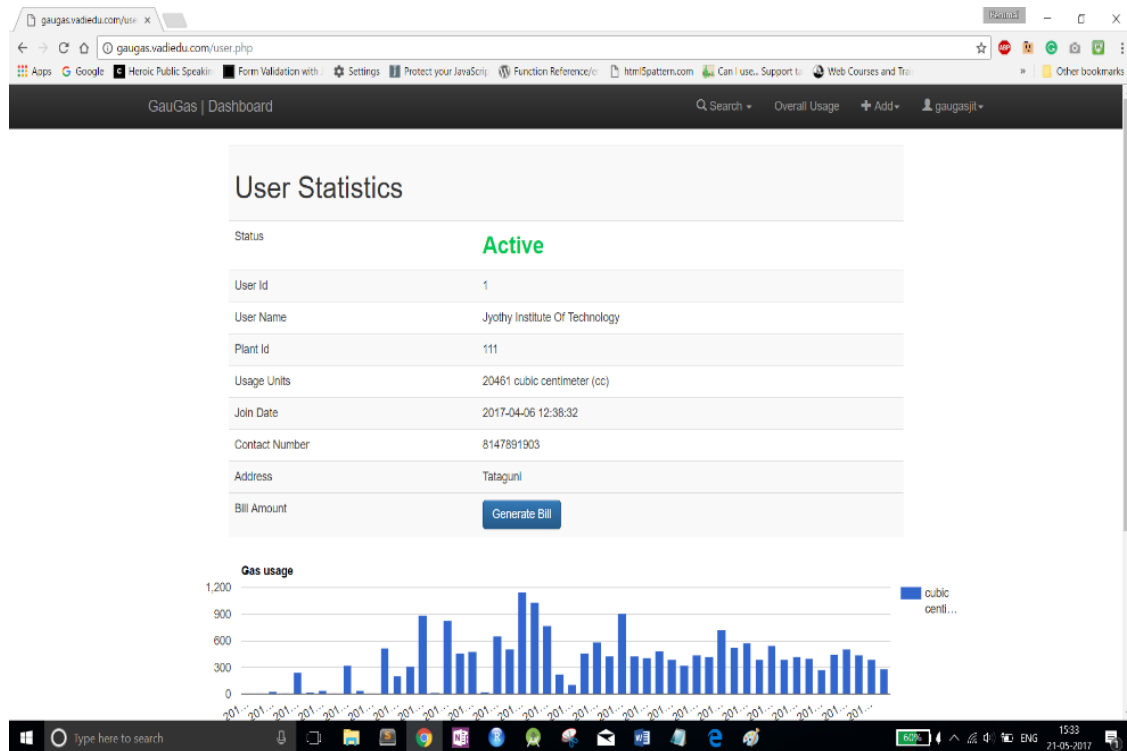


Figure 7.2 User details

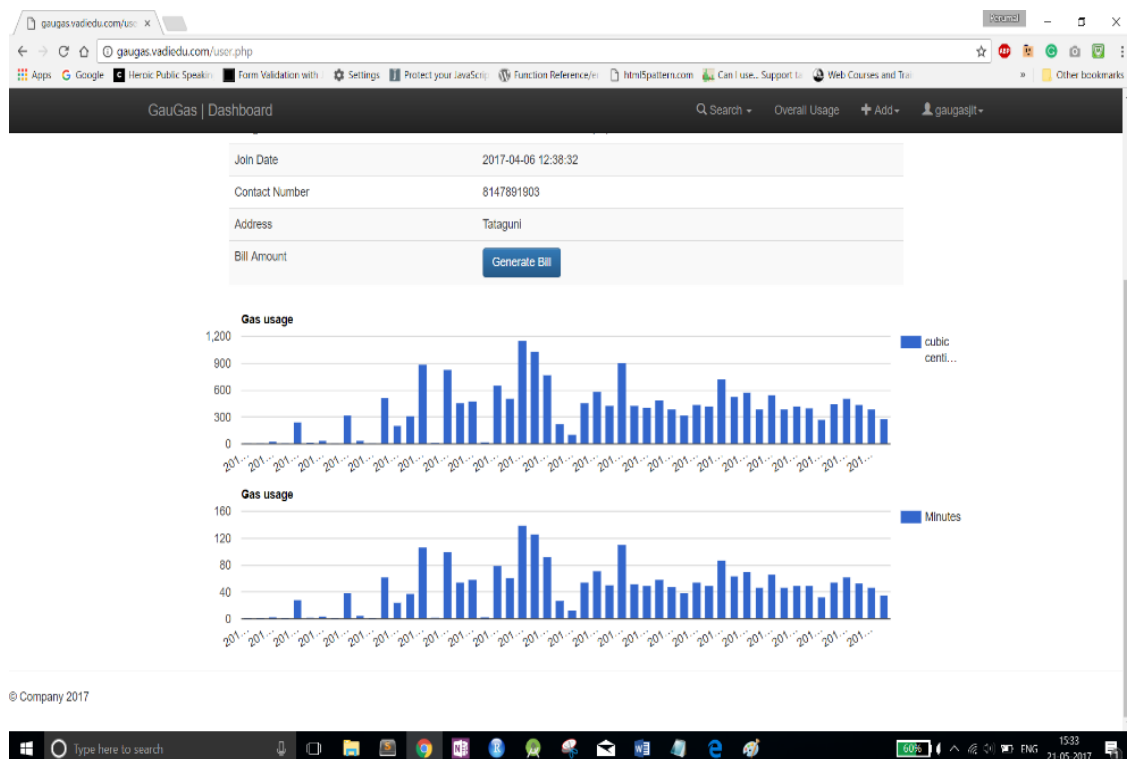


Figure 7.3 usage chart

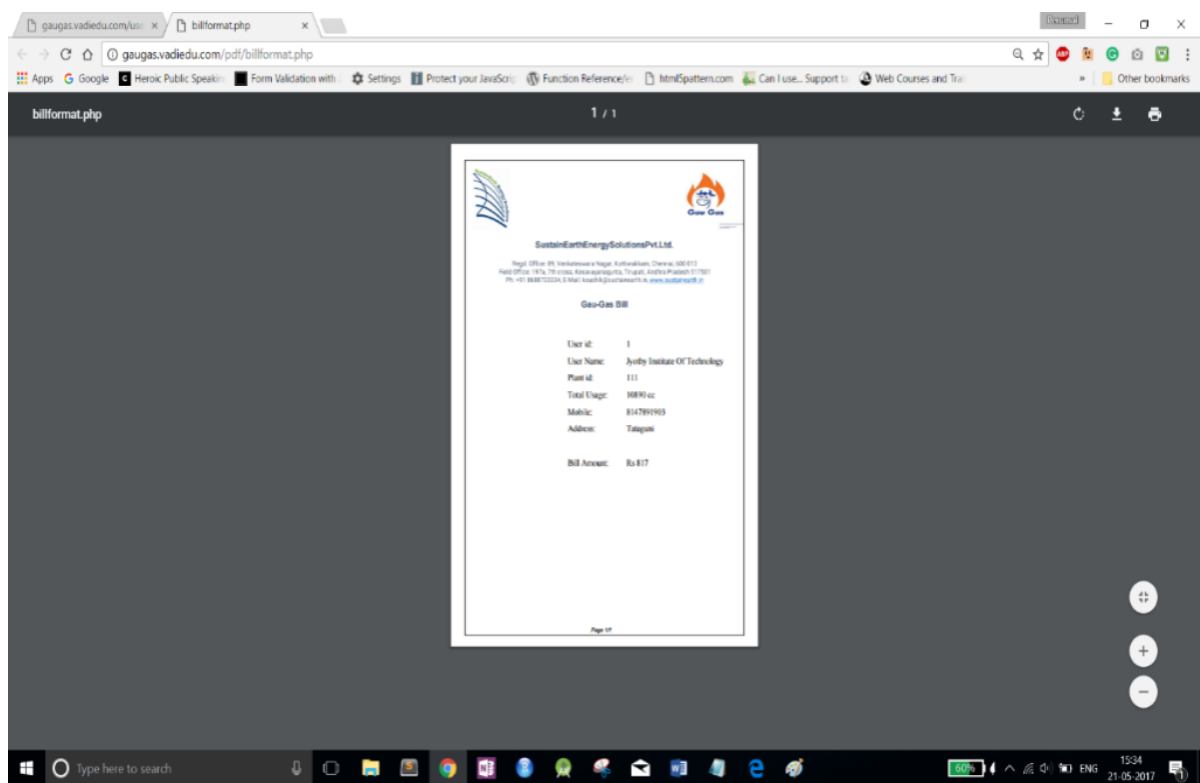


Figure 7.4 Bill



Figure 7.5 Mixing cow dung and water to feed plant



Figure 7.6 Through away prototype



Figure 7.7 Installing the GauGas plant at ground level in JIT



Figure 7.8 IoT module setup at JIT hostel Kitchen



Figure 7.9 Completely filled Gau gas plant at JIT

GauGas


HOMEABOUT USTEAMCONTACT USARCHITECTURELOGIN

Please log in

User Name

Password


submit



SustainEarth Energy Solutions

1st Floor, #197 A, 7th Cross Kesavayanagunta,
Tirupati - 517501 Andhra Pradesh, India

+91 868 873 3334



Jyothy Institute of Technology

Thataguni, off Kanakapura Road,
Bengaluru - 560 082 Karnataka, India

+91 814 789 1903

GauGas© 2017

Figure 7.10 Login Page

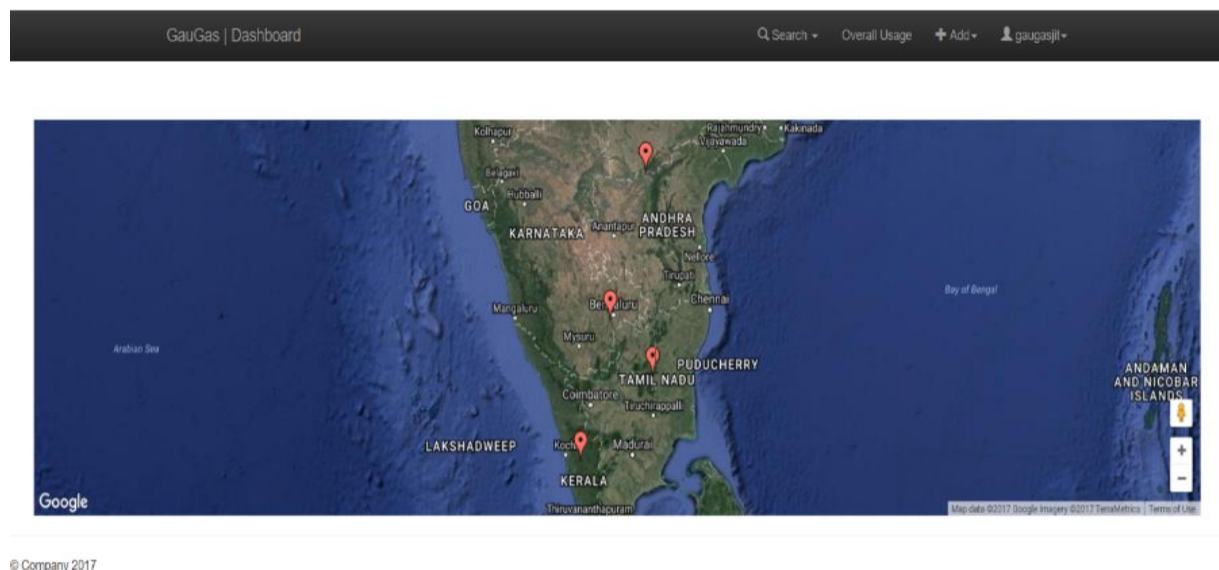


Figure 7.11 Dashboard

The screenshot shows a web browser window with the address bar displaying 'gaugas.vadiedu.com/adduser.php'. The page title is 'GauGas | Dashboard'. The form contains the following fields:

- User Id: Enter User Id
- User Name: Enter user name
- User Address: Enter user Address
- Contact Number: Enter user Number
- Plant Id: Enter plantid

A 'Submit' button is located at the bottom of the form. The footer of the page reads '© Company 2017'. The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray with the date '21-05-2017' and time '15:35'.

Figure 7.12 Add new user

The screenshot shows a web browser window with the address bar displaying 'gaugas.vadiedu.com/addplant.php'. The page title is 'GauGas | Dashboard'. The form contains the following fields:

- plant Id: Enter plant Id
- plant loc: Enter plant location
- Number of connections: Enter no. of connections

A 'Submit' button is located at the bottom of the form. The footer of the page reads '© Company 2017'. The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray with the date '21-05-2017' and time '15:35'.

Figure 7.13 Add new plant

Chapter 8

CONCLUSION AND FUTURE WORK

The main objective of the project is to develop a IoT module which automate the metering and billing process without the use of internet in plant location.

I had taken a wide range of literature review in order to achieve all the tasks, where I came to know about some of the products that are existing in the market. I made a detailed research in that path to cover the loop holes that existing systems are facing and to eradicate them in our IoT module. In the process of research, I came to know about the latest technologies. the product has been successfully developed in terms of extendibility, portability, and maintainability and tested in order to meet all requirements that are

- Authentication
- Integrity
- Confidentiality

These are the three basic concepts for the secure communication over a network.

8.1 Future Work

8.1.1 LCD Display

The IoT module can have an LCD display showing logistic usage and the details of the IoT module.

8.1.2 Touch Pad

The user can use the touch pad to control the flow of gas to various levels of low, medium and high instead of analog regulator. The touch pad can also be used during the installation time to configure the IoT module.

8.1.3 SD Card

The SD Stores the logistics usage of plant and stores in a specific format such that the data can be fetched by the admin by sending an SMS to the module.

8.1.4 Payment Gateway

Implementing SMS based smart payment gateway option for the rural areas people such there will be no difficulties in making the bill payment.

8.1.5 Remote Configuration Of IoT Module

The should be able to check the condition of the module and also to configure the module remotely by sending and receiving the SMS from the module.

REFERENCES

- [1] www.henrysbench.capnfatz.com/henrys-bench/arduino-current-measurements/the-acs712-current-sensor-with-an-arduino/
- [2] Reading Current sensor values through Arduino uno and displaying the values on the serial monitor [https:// www.electrodragon.com/acs712-current-sensor-read/](https://www.electrodragon.com/acs712-current-sensor-read/)
- [3] Belal Khan , automatic refresh of data without refreshing on web page ,April 28 2015 <https://www.simplifiedcoding.net/php-mysql-insert-into-database-tutorial/>
- [4] Interfacing the Gsm module to begin and get connected to the network for calling and receiving the sms <https://www.instructables.com/id/Interfacing-SIM900A-GSM-Modem-with-Arduino/>
- [5] Android tutorial for learning about how database works in the android application, <https://developer.android.com/training/basics/data-storage/databases.html>
- [6] To read and delete the sms based on their position in the stored messages in the sim <http://www.learnerswings.com/2014/03/read-sms-message-by-at-commands-using.html>
- [7] <http://www.circuitstoday.com/interface-gsm-module-with-arduino>
- [8] Paper on Receiving the sms and displaying it in the application using a list view , June 05 2015 , <http://javapapers.com/android/android-receive-sms-tutorial/>
- [9] Technical report on economics of gau gas plant by D.K.Desai GSFC Professor , IIM Ahmedabad , July 1974
- [10] Online monitoring and control of the biogas process By Kanokwan Boe , Institute of Environment & Resources , Technical University of Denmark , May 2006
- [11] General Theory of Security and a Study Case in Internet of Things Yixian Yang, Haipeng Peng, Lixiang Li and Xinxin Niu , July 2015
- [12] Lysis: a platform for IoT distributed applications over socially connected objects Roberto Girau, Salvatore Martis, Luigi Atzori, Senior Member, IEEE Department of Electrical and Electronic Engineering - University of Cagliari, 09123 Cagliari, Italy