2.9 Effect of landslides on the environment

Landslides affect the following elements of the environment,

- The topography of the earth’s surface
- The character and quality of rivers and streams and groundwater flow
- The forests that cover much of the earth’s surface
- The habitats of natural wildlife that exist on the earth’s surface, including its rivers, lake and oceans.

Large amounts of earth and organic materials enter streams as sediment as a result of this landslide and erosion activity, thus reducing the portability of water and quality of habitat for fish and wildlife. Biotic destruction by landslides is also common, widespread stripping of forest cover by mass movement has been noted in many parts of the world. Removal of forest cover impacts wildlife habitat.

The ecological role that landslides play is often overlooked. Landslides contribute to acquire and terrestrial biodiversity. Debris flows and other mass movement play an important role in supplying sediment and coarse woody debris to maintain pool/riffle habitat in streams. As disturbance agents landslides engender a mosaic of serial stages, soils and sites to forested landscapes.

Shallow, rapid landslides will only increase in scenarios of increasing rain event intensity, whereas, deep- seated mass movements will increase with seasonal increases in precipitation. Dry ravel may respond more directly to warming, increasing with sparse vegetation covers and increased frequency of fire. Climate change has the greatest impacts on landslide occurrence by modifying evapotranspiration and root strength vegetation. Evapotranspiration affects soil water recharge and subsurface, soil runoff and evaporation, ground cover, and rooting depth, these in turn affect water storage and routing in unstable sites.

2.30 Working procedure of the proposed system

The following factors were chosen for the automation of the system

- Soil moisture content
- Sudden change in the strain of the soil

Here, we are planning to install four soil moisture sensors in four selected points around the mountain and also we are planning to fix strain gauges such that each one is located in between a crack (shown in figure 2.20). Then the sensor data will be updated to a webserver through the Wi-Fi module and after that we are going to monitor the health of the every hour while updating these data to a web server (real time data monitoring). If the monitoring data exceeds a pre-determined critical conditions, the system identifies this as a possibility for a landslide and a warning will be issued via SMS and an alarm, to all the residents in that particular area.

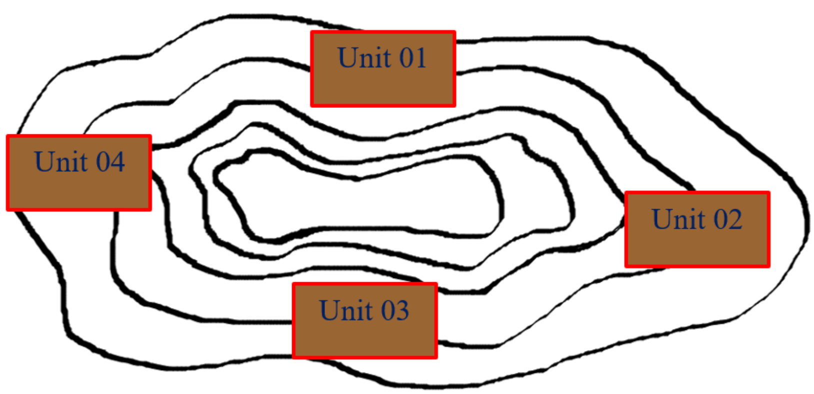


Figure 2.20 - Proposed LMEWS

Chapter 03 – System Design

3.1 Hardware Design

3.1.1 Arduino Uno

Product overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a 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 Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

Table 3.1 - Technical specification

Microcontroller	ATmega328P – 8 bit AVR family microcontroller
Operating Voltage	5V
Recommended input voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

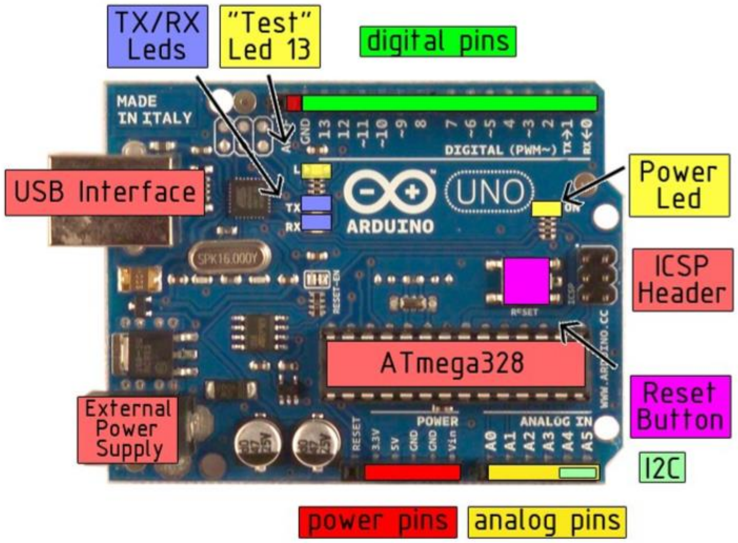


Figure 3.1 - Parts of the Arduino Uno Board

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

Table 3.2 - Pin Description

Pin Category	**Pin Name**	**Details**
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

USB overcurrent protection

The Arduino Uno has a resettable poly fuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical properties

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

3.1.2 Soil Moisture Sensor

This sensor can be used to test the moisture of soil. When the soil is having water shortage, the module output is at high level, and else the output is at low level. This module consists of a triple output mode, digital output is simple, analog output more accurate, serial output with exact readings.

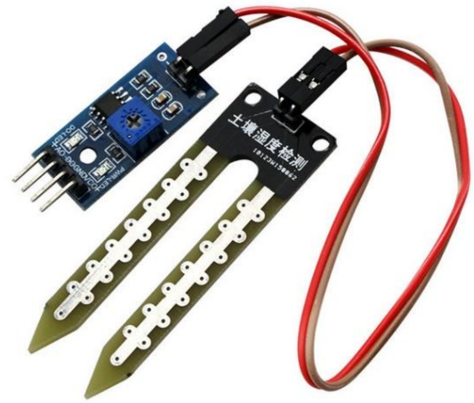


Figure 3.2 - Soil Moisture Sensor

Features

- Sensitivity adjustable.
- Has fixed bolt hole, convenient installation
- Threshold level can be configured.
- Module consists of a triple output mode, digital output is simple, analog output more accurate, serial output with exact readings.

The specifications of the soil moisture sensor FC-28 are as follows.

Table 3.3 - Specifications

Input Voltage	3.3 – 5V
Output Voltage	0 – 4.2V
Input Current	35mA
Output Signal	Both Analog and Digital

The soil Moisture sensor FC-28 has four pins.

Table 3.4 - ***Pin configuration***

Vcc	For power
A0	Analog output
D0	Digital output
GND	Ground

Working principle

The soil moisture sensor consists of two probes which are used to measure the volumetric content of water. The two probes allow the current to pass through the soil and then it gets the resistance value to measure the moisture value. When there is more water, the soil will conduct more electricity which means that there will be less resistance. Therefore, the moisture level will be higher. Dry soil conducts electricity poorly, so when there will be less water, then the soil will conduct less electricity which means that there will be more resistance. Therefore, the moisture level will be lower.

3.1.3 Wi-Fi Module ESP 8266 ESP 01

The ESP8266 ESP-01 is a Wi-Fi module that allows microcontrollers access to a Wi-Fi network. This module is a self-contained SOC (System On a Chip) that doesn't necessarily need a microcontroller to manipulate inputs and outputs as you would normally do with an Arduino. Depending on the version of the ESP8266, it is possible to have up to 9 GPIOs (General Purpose Input Output). Thus, we can give a microcontroller internet access like the Wi-Fi shield does to the Arduino, or we can simply program the ESP8266 to not only have access to a Wi-Fi network, but to act as a microcontroller as well.

Figure 3.3 - ESP8266 ESP 01 Wi-Fi Module

Table 3.5 - Pin description

No	Pin name
1	GND
2	TX0
3	GPIO2
4	CH_PD
5	GPIO0
6	RST
7	RXI
8	3V

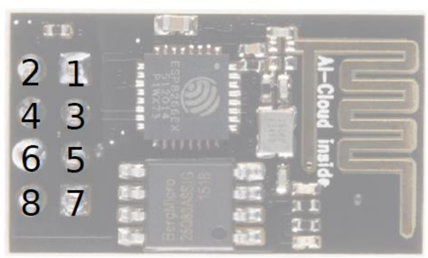


Figure 3.4 - Pin Diagram of ESP 8266

3.1.4 Strain Gauge

Strain gauge is used to measure the strain of the materials which occurs due to tension or compression. Strain gauge is an analog sensor which gives a range of values based the stress it is subjected to. It can be used in a Wheatstone bridge to measure the resulting resistance from compression or tension.

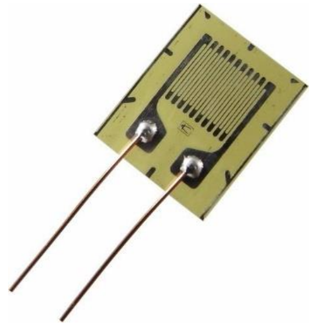
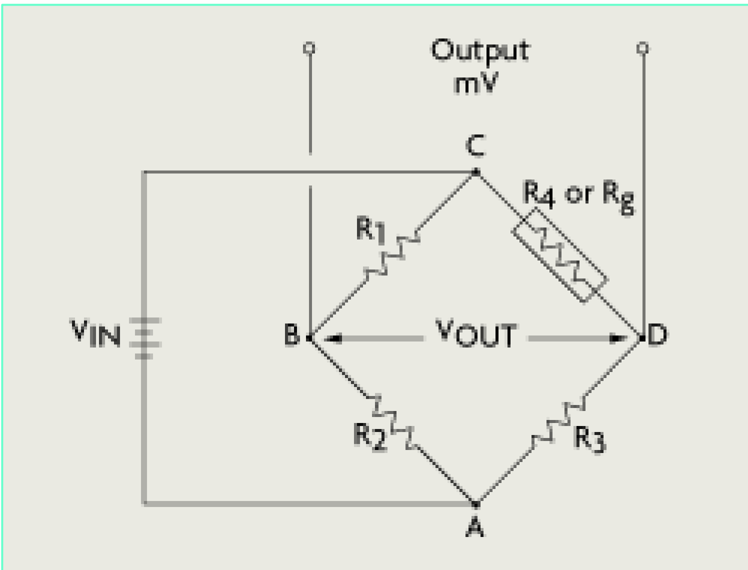


Figure 3.5 - Strain Gauge



$$V_{OUT} = V_{IN}\left[\frac{R_3}{R_3 + R_g} - \frac{R_2}{R_1 + R_2}\right]$$

Figure 3.6 – Wheatstone bridge and strain gauge

For this project we plan to place strain gauges on a pre-stressed rectangular metal plate.

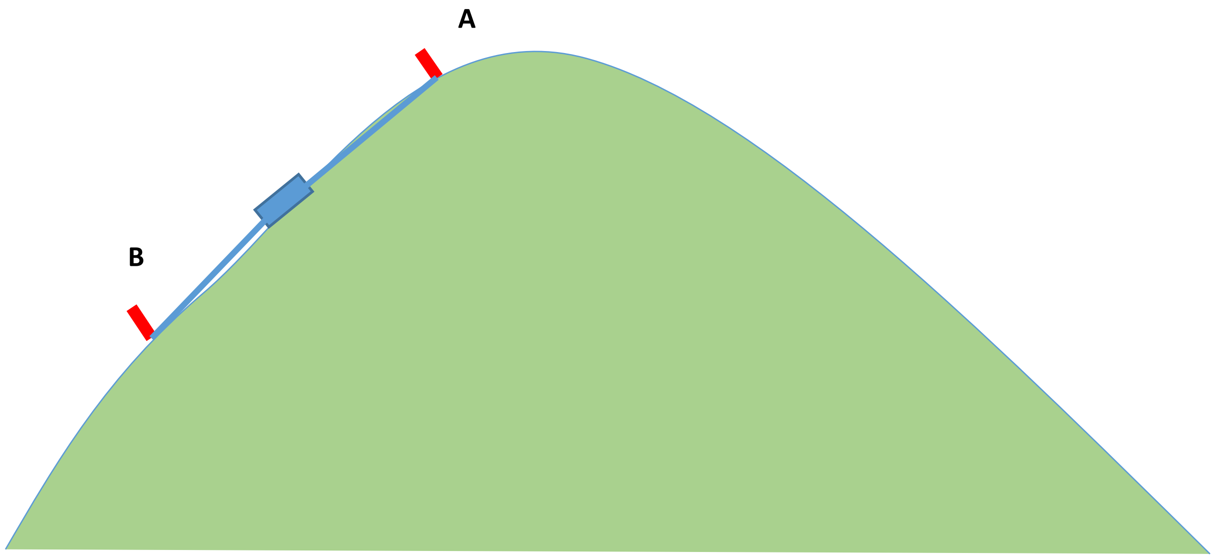


Figure 3.7 - Installation of strain gauge in mountain

As shown in the figure 3.7, two cables are attached to the two ends of the metal plate. Other ends of cables need to be connected to steel rods which will be eventually fixed firmly to the ground. When considered a landslide situation a downhill rod (B) is more likely to move compared to uphill rod (A). This will result in an alternation of the previous strain value (pre-stressed). Prolonged gradual increment in the strain value can be used to detect a possible landslide in addition to the soil moisture levels.

3.1.5 Voltage Regulator – AMS1117

General Description

The AMS1117 series of adjustable and fixed voltage regulators are designed to provide up to1A output current and to operate down to 1V input-to-output differential. The dropout voltage of the device is guaranteed maximum 1.3V, decreasing at lower load currents.

Figure 3.8 – AMS1117 Voltage Regulator

Pin Connections

3 pin fixed/adjustable version

- Ground/Adjust
- VOUT
- VIN

Features

• Three Terminal Adjustable or Fixed Voltages

• Post Regulators for Switching Supplies

• Output Current of 1A

• Operates Down to 1V Dropout

• Line Regulation: 0.2% Max.

• Load Regulation: 0.4% Max.

Applications

• High Efficiency Linear Regulators 1.5V, 1.8V, 2.5V, 2.85V, 3.3V and 5.0V

• 5V to 3.3V Linear Regulator

• Battery Chargers

• Active SCSI Terminators

3.2 Processor Unit

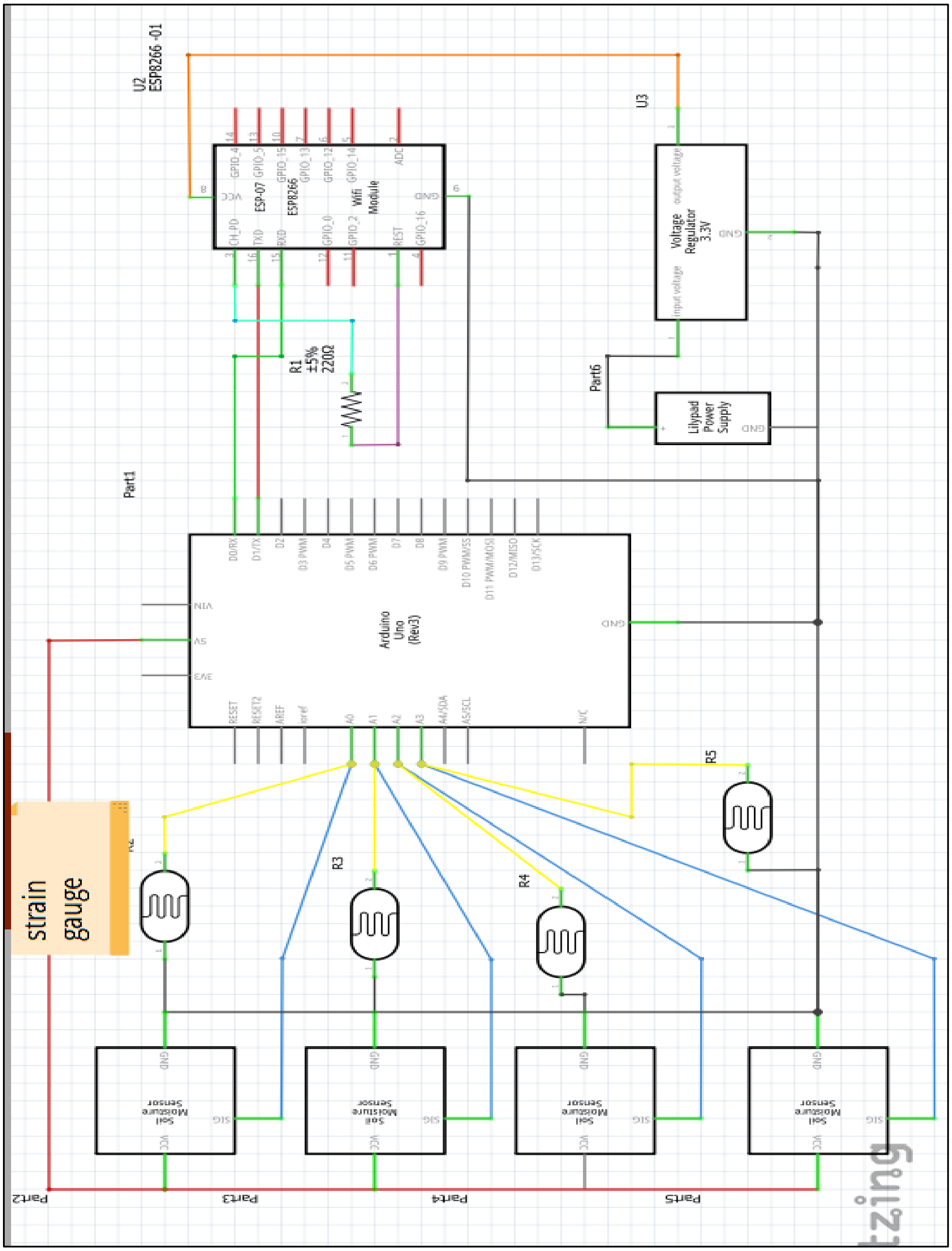


Figure 3.9 – Microcontroller pin interconnection diagram

Chapter 04 – System Implementation

4.1 Implementation of the design

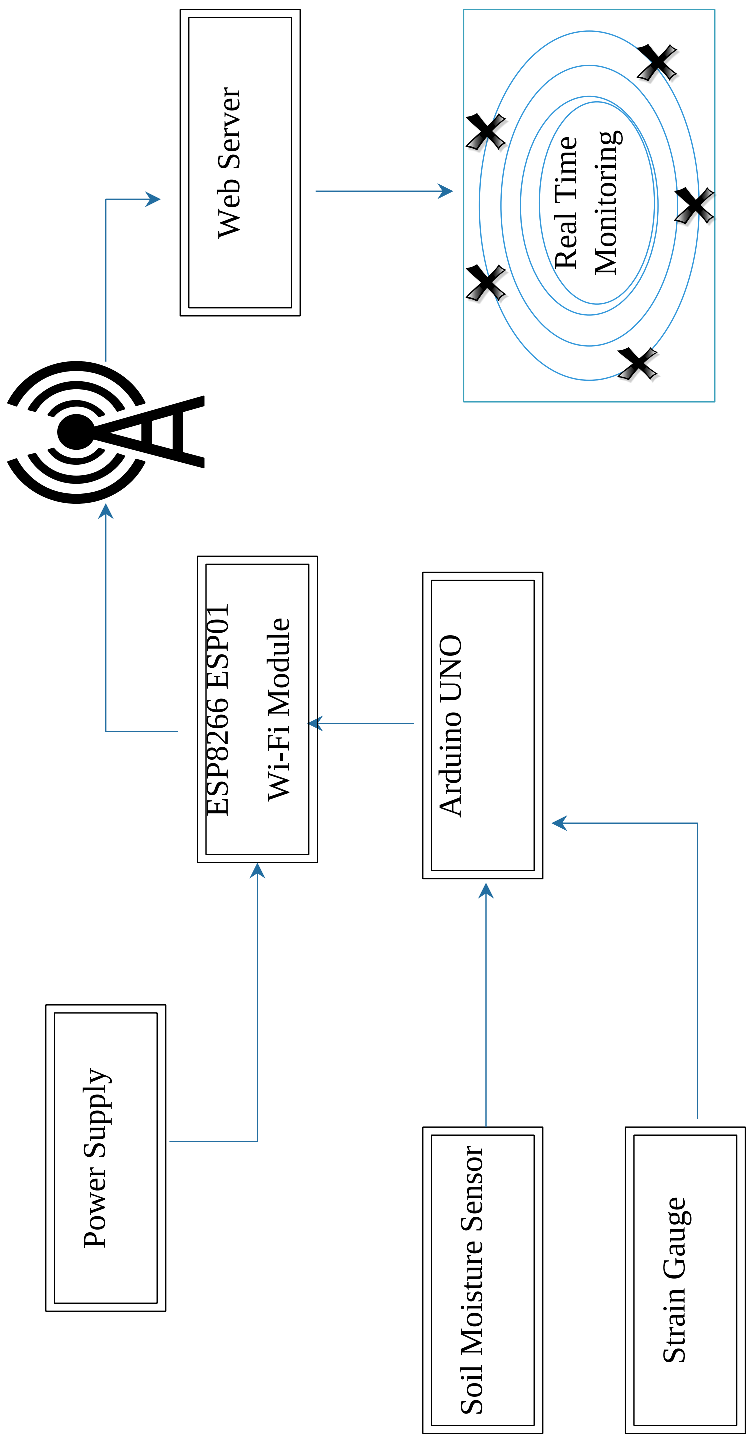


Figure 4.1 - Block diagram of the proposed system

Landslide detection and online monitoring is the main objective of the project. Each sensing unit will contain a soil moisture sensor and a strain gauge sensor. A number of such sensor units are placed on the mountain area which will possibly be more vulnerable to landslides (e.g. mountains in Kegalle district).

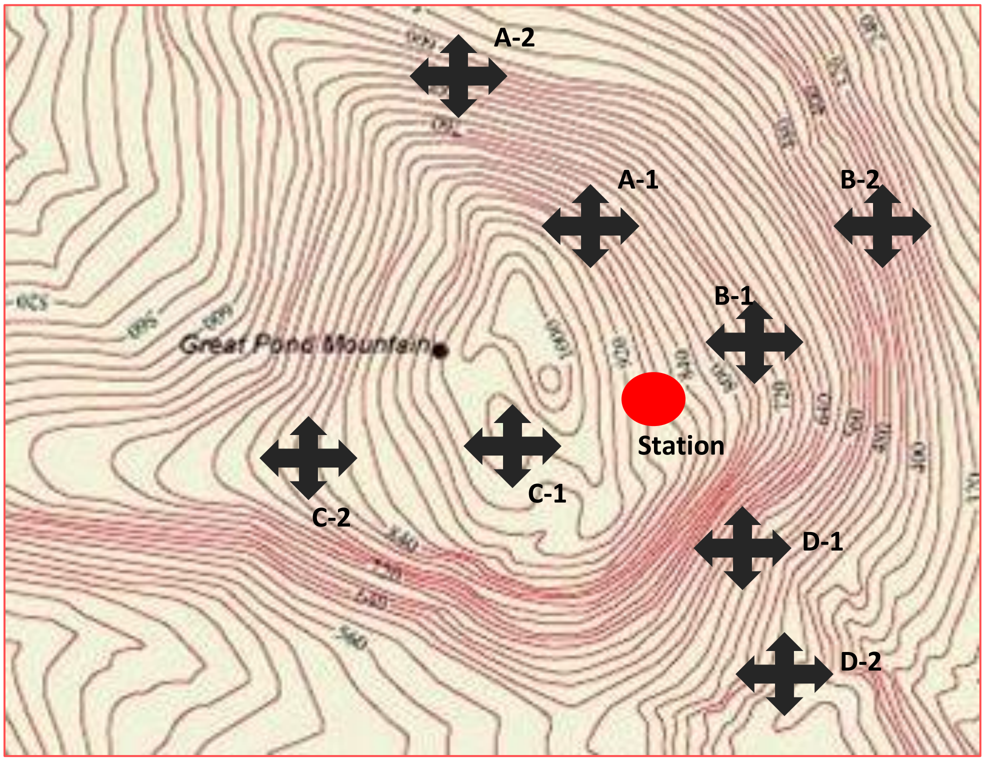


Figure 4.2 - Contour Map

As shown in the contour map, sensor units can be positioned in number of directions of the mountain. Signal and cables from the sensor units (e.g. A-1 to D-2) are brought together at the station on the mountain. Sensor values will be transmitted to an IOT website (in this project www.Thingspeak.com) by connecting to the internet through a Wi-Fi connection.

4.2 Software Implementation

4.2.1 Microcontroller - Arduino Uno to ATmega328 Pin Mapping

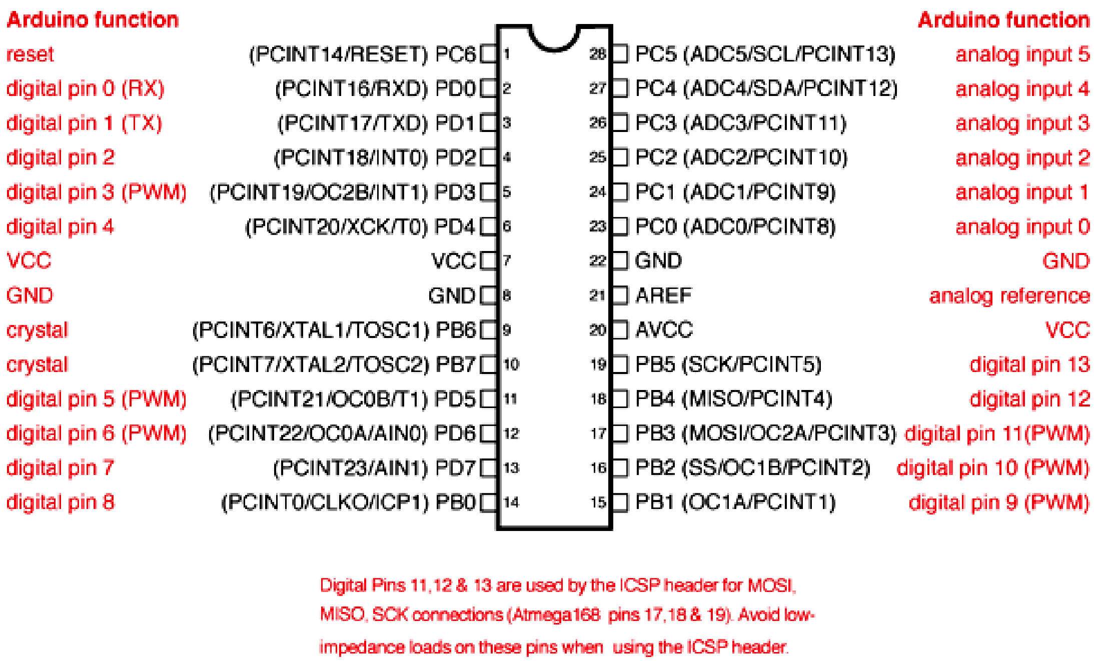


Figure 4.3 - Microcontroller pin mapping diagram

4.2.2 Arduino IDE Environment

The Arduino Software (IDE) contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. Arduino IDE can be run on Windows or Linux on either 32 bit or 64 bit Application Size. We used Arduino 1.8.5 Software to build our software. Basically this software enables better and assisted code editing, compiling and debugging. Arduino graphical user interface will give you different panes and options for creating and executing programs in addition to make relevant configurations. It requires Java Runtime Environment. This comes with a board management module, where users can select the board they want to work with at the moment.

Features

- Sketchbook
- Sketches Management
- Sketch Editing Tools
- Libraries
- Serial Monitor
- User Preferences
- Project Documentation
- Sketches sharing
- Auto Format
- Sketch Archive
- Fix Encoding & Reload
- Board Selection & Management
- Port Menu
- Programmer Functions
- Burn Boot-loader

4.2.3 Programming Languages

Programming in C is a tremendous asset in those areas where you may want to use assembly language, but would rather keep it as a simple to write and easy to maintain program. It has been said that a program written in C will pay a premium of a 50 to 100% increase in running time, because no language is as compact or fast as Assembly language. However, the time saved in coding can be voluminous, making it the most desirable language for many programming chores.

In Arduino IDE we used C language to communicate with different modules such as ESP8266 WIFI Module and soil moisture sensor. First we write our program suitable for that component and compile it. Then we uploaded into our arduino board and test whether it is working or not.

Also we used python little bit to monitor the values from Arduino IDE serial monitor through this ThingSpeak online web monitoring system.

4.2.4 ThingSpeak Online Web Monitoring

ThingSpeak is the online web monitoring systems. It enables sensors, instruments and websites to send data to the cloud where it is stored in either a private or a public channel. It gets data in three ways. Such as JSON, XML and CSV. Values in the XML file can also be used for online monitoring purpose. Once data is in a ThingSpeak channel, you can analyse and visualize it, calculate new data, or interact with social media, web service and other devices.

4.2.5 How monitoring was done with Soil Moisture Sensor

After connected with the arduino board it communicates with the computer through serial monitor of Arduino IDE. Then we checked the graphs by doing it practically. When values changed due to the moisture the graph also changed in parallel. In this project a python program is used to check the values in the XML file. Python program checks the XML value every few minutes and checks whether the values have reached to a critical level.


```
▼<channel>
   <id type="integer">498392</id>
   <name>sensors</name>
   <latitude type="decimal">0.0</latitude>
   <longitude type="decimal">0.0</longitude>
   <field1>Field Label 1</field1>
   <created-at type="dateTime">2018-05-17T05:49:24Z</created-at>
   <updated-at type="dateTime">2018-05-21T19:09:01Z</updated-at>
   <last-entry-id type="integer">196</last-entry-id>
 ▼<feeds type="array">
   ▼<feed>
      <created-at type="dateTime">2018-05-21T11:25:44Z</created-at>
      <entry-id type="integer">97</entry-id>
      <field1>1022</field1>
    </feed>
   ▼<feed>
      <created-at type="dateTime">2018-05-21T11:26:02Z</created-at>
      <entry-id type="integer">98</entry-id>
      <field1>1023</field1>
    </feed>
   ▼<feed>
      <created-at type="dateTime">2018-05-21T11:26:19Z</created-at>
      <entry-id type="integer">99</entry-id>
      <field1>1021</field1>
    </feed>
```


Figure 4.4 - Monitoring of data in XML form

These highlighted values can be marked as soil moisture sensor readings. Also the graph of soil moisture readings can be shown like this.

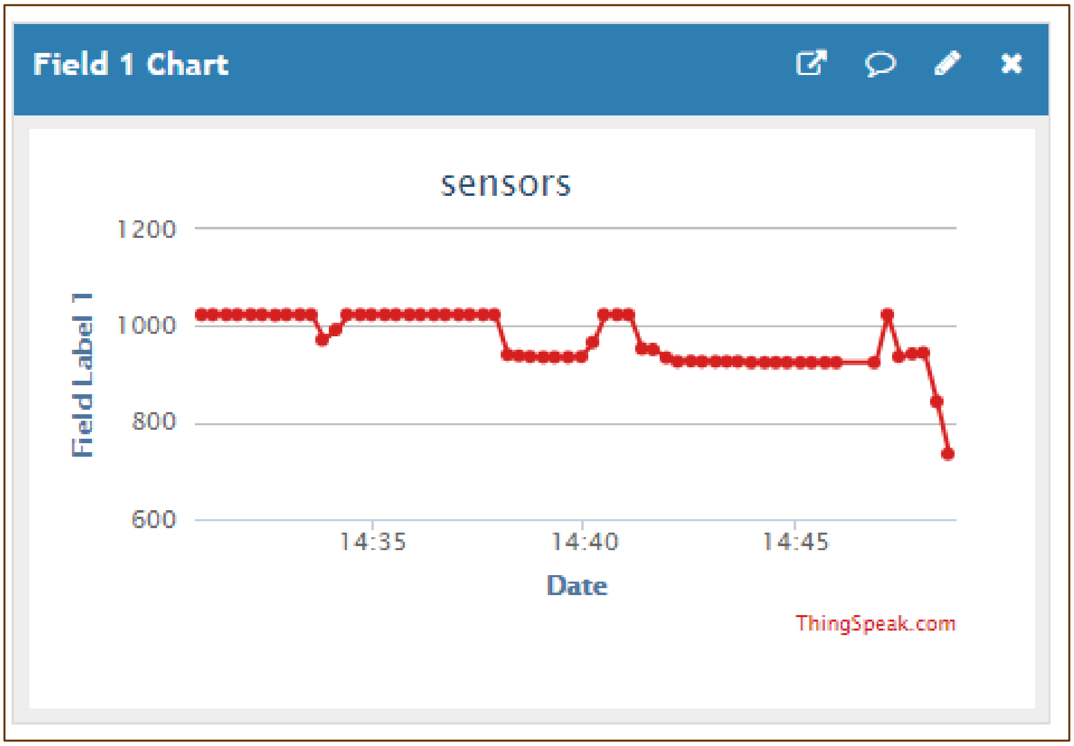


Figure 4.5 - Thingspeak data monitoring

4.2.6 How Arduino Board communicating with Personal Computer

The Arduino Uno has a number of facilities for communicating with a computer, another arduino or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

4.2.7 How Arduino Board Communicating with ESP 8266 WIFI MODULE

To check the connectivity of the ESP8266 WIFI Module, we connect it to arduino board correctly. Then in arduino IDE we can check whether it is connected or not by opening serial monitor and type "**AT**" in the message field and press enter, it should respond with OK. To join the network we can type "**AT+CWJAP***" and type Wi-Fi-name and password in the message field.

Afterwards we enter data real time to our thingspeak account as our online server and uploaded software to the arduino board. When uploading have to change Tx-2 and Rx-3 in the code.

4.2.8 How Arduino Board Communicating with Soil Moisture Sensor

Soil moisture sensor changes its values between 0 to 1023. When it is dry it shows value 1023 value in arduino serial monitor. And when its apply with water it shows below 1023 value. After verifying the code, upload it to the board and open the serial monitor. You will see the sensor data on the monitor being changed when you dip the sensor leads in water and when dry**. You can use these values as threshold if you intend to trigger an action bases on these values.**

4.2.9 Benefits

Ability to read soil volumetric water content directly

No special maintenance necessary

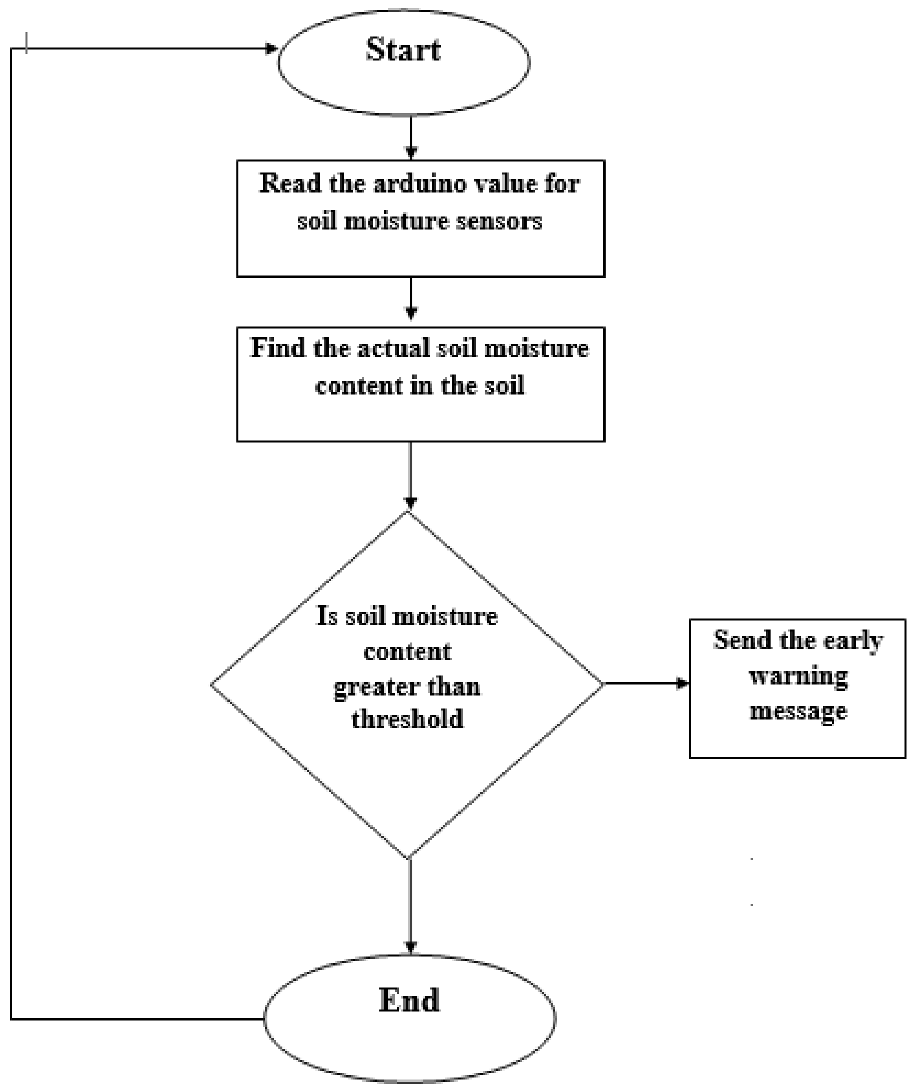
Highly accurate when sensors are installed properly in good contact with soil.

Continuous measurement at same location

Chapter 05 – System Analysis and Testing

5.1 System Analysis

5.1.1 Fault analysis flow chart



5.1.2 System Testing

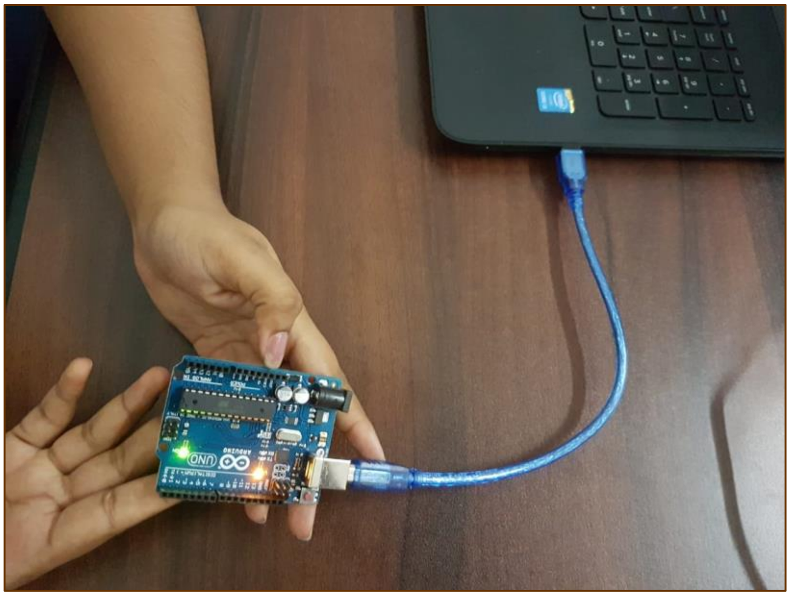


Figure 5.1 – Connecting Arduino board to the laptop

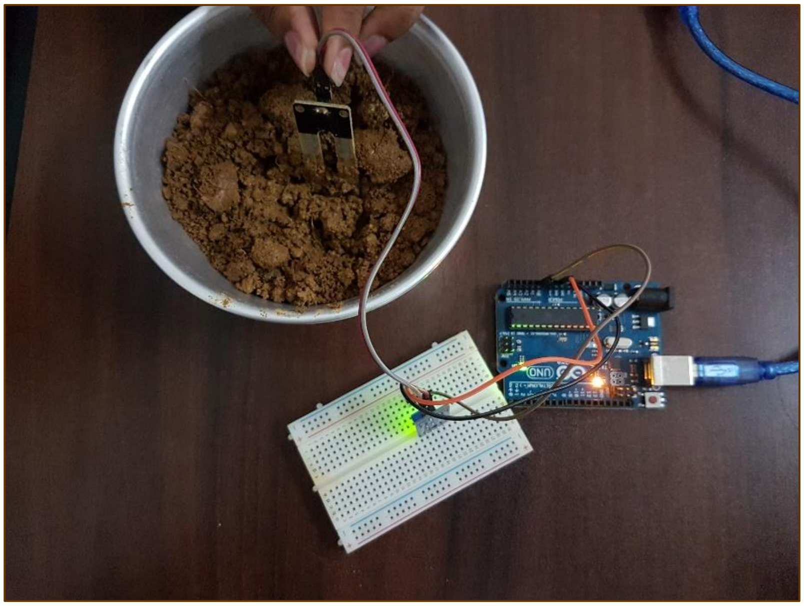


Figure 5.2 – Connecting soil moisture sensor with arduino

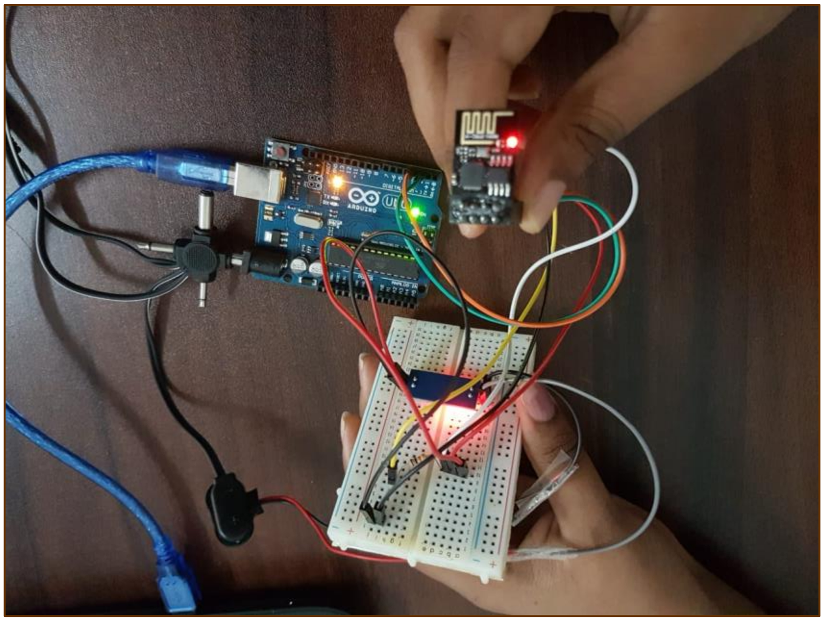


Figure 5.3 - Connecting Wi-Fi module with arduino board and power pack

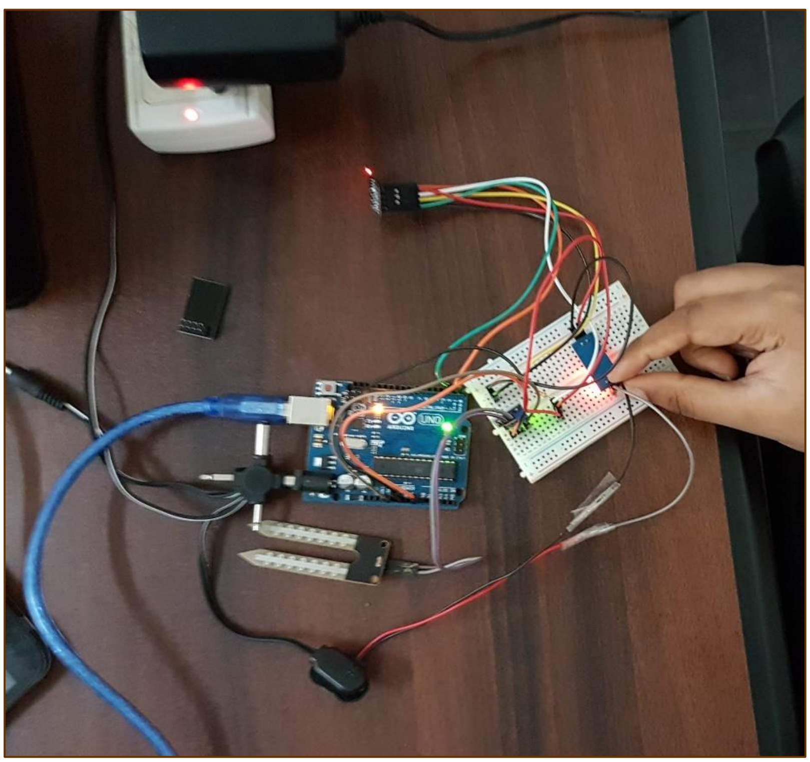


Figure 5.4 - Circuit after all the connections

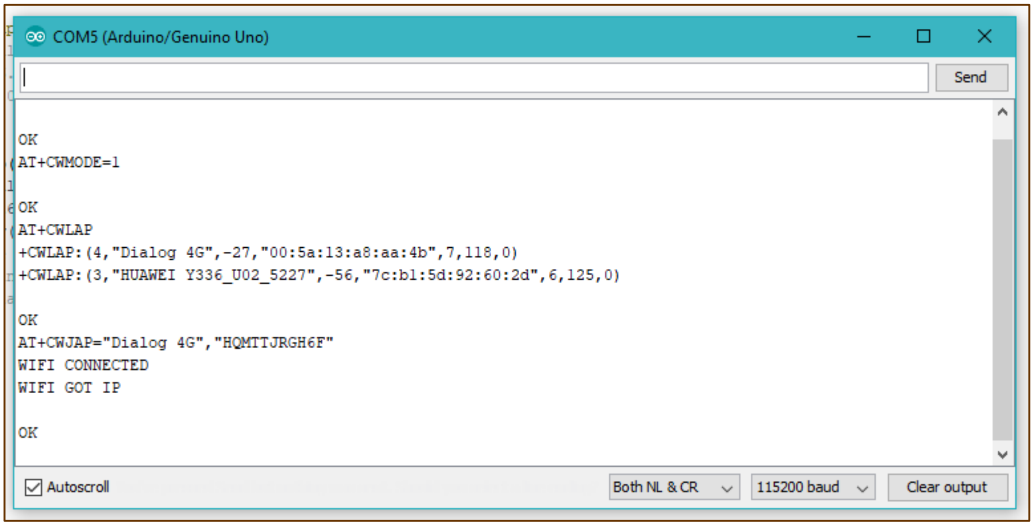


Figure 5.5 - Configuration of Wi-Fi Module

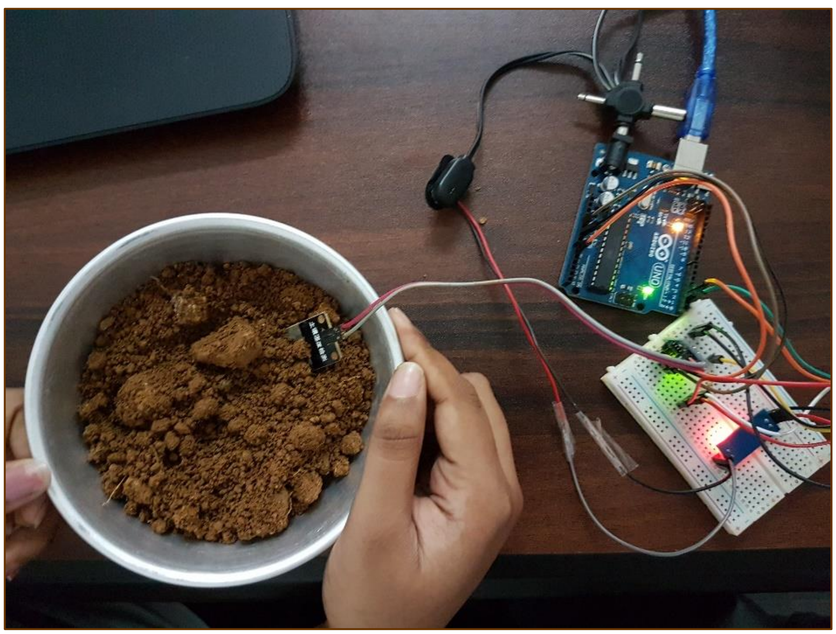


Figure 5.6 - Testing the system

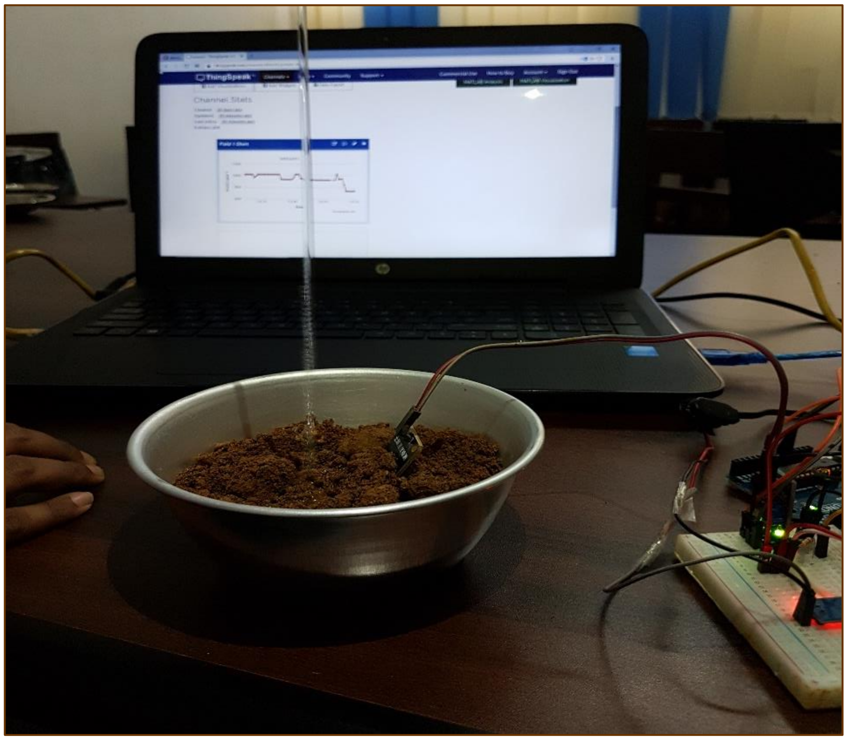


Figure 5.7 - Testing the system with the variation of soil moisture

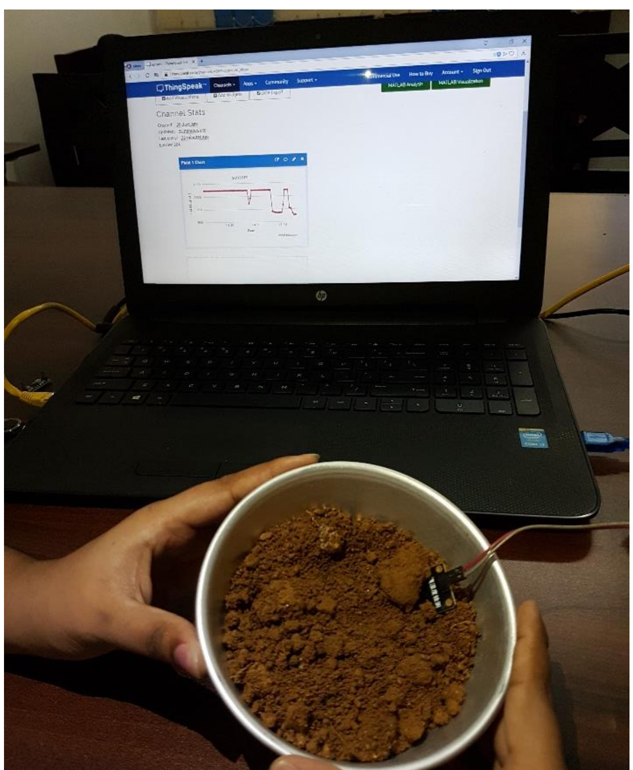


Figure 5.8 - Variation of the graph with respect to the change in soil moisture

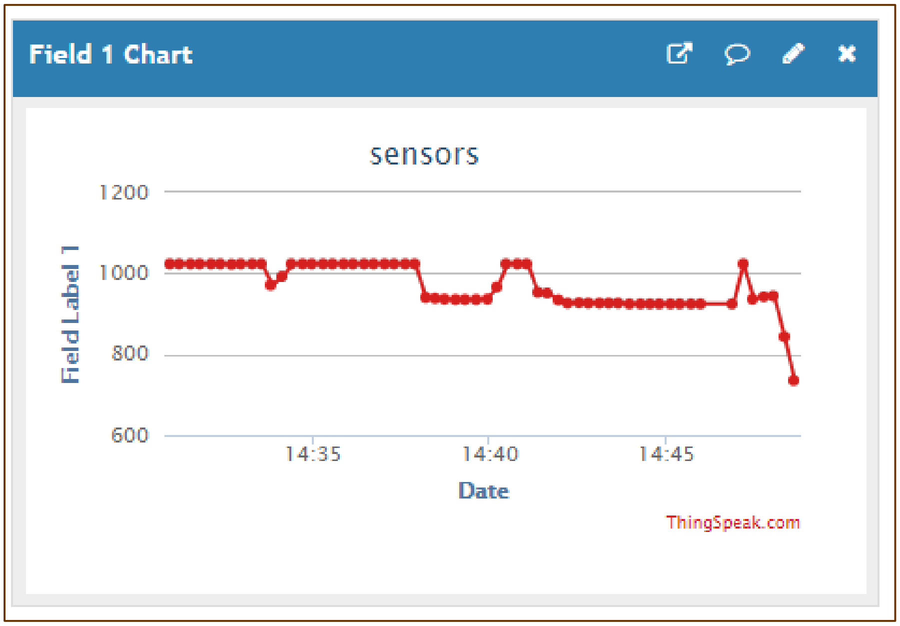


Figure 5.9 - Updating of the soil moisture vs. time on thingspeak

5.2 Testing of sensors

In this project we wanted to find out a mathematical relationship between soil moisture variations with the addition of unit amount of water. Since the data we needed was not available in NBRO, We had to carry out certain experiments. Following are the details of those experiments.

Here, we tested an actual soil sample from kendawa (recommended potential landslide site by NBRO) and Aranayake.

In the first attempt on Kendawa soil sample, the readings were not sufficient for our purpose and therefore we have carried out a second attempt too.

5.2.1 Experiment results on Kendawa soil sample – First attempt

Table 5.1 – Results of Kendawa soil sample – First attempt

X = arduino reading	Y = actual soil moisture reading
944	23.8
947.8	26.72
924.6	29.66
547.8	31.78
503.2	36.27
524.6	38.77
521.6	41.75
493.6	43.19
431.2	47.64
416.4	49.51
425	52.06

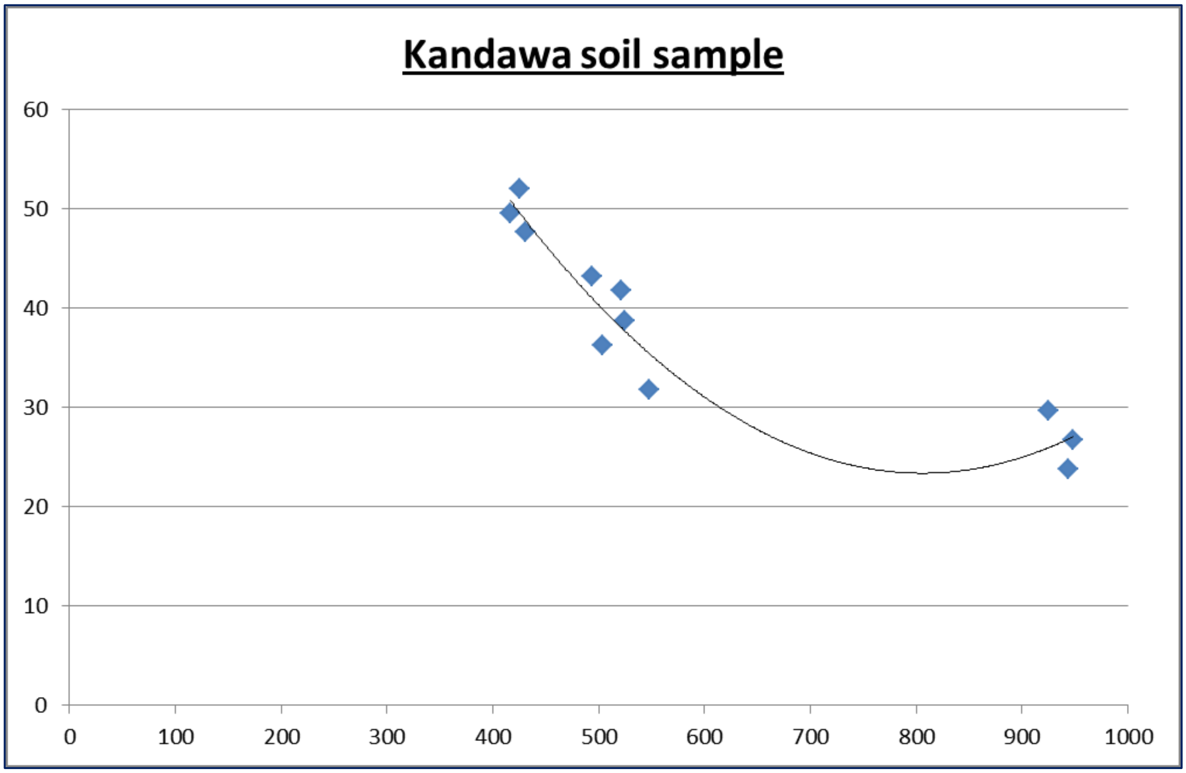


Figure 5.10 - Kandawa first attempt

5.2.2 Experiment results on Kendawa soil sample – Second attempt

Table 5.2 – Results of Kendawa soil sample – Second attempt

X = arduino reading	Y = actual soil moisture reading
966	0.24
826.8	0.28
467	0.33
425.4	0.38
440.2	0.42
425.4	0.49
395.6	0.55
399.2	0.61

389.4	0.66
393.4	0.72
394.4	0.81
389.2	0.83
415.6	0.9
375.8	0.91
357.8	1.03
389.2	1.11
385.2	1.24
387.4	1.37
388.2	1.47
392	1.67

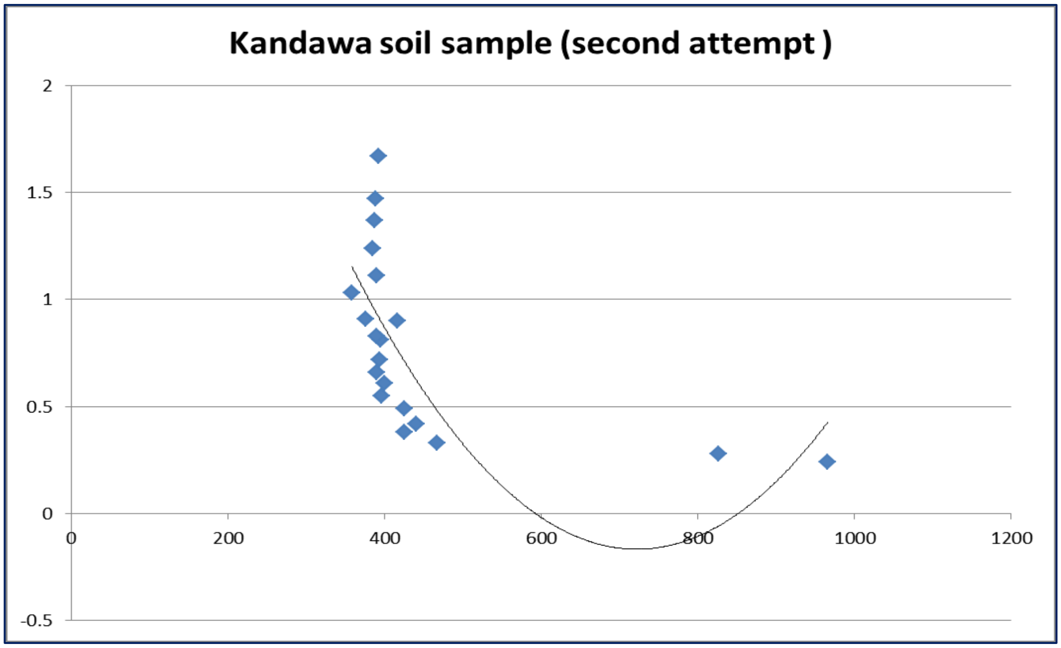


Figure 5.11 - Kandawa second attempt

5.2.3 Experiment results on Aranayake soil sample

Table 5.3 – Results of Aranayake soil sample

X = arduino reading	**Y = actual soil moisture reading**
1001	0.14
967.8	0.21
559.6	0.26
477.8	0.34
467.8	0.4
462.2	0.46
484.6	0.46
474.5	0.58
447	0.64
487	0.68
452	0.97
371	1.08
296	1.05
304	0.91
301	0.94

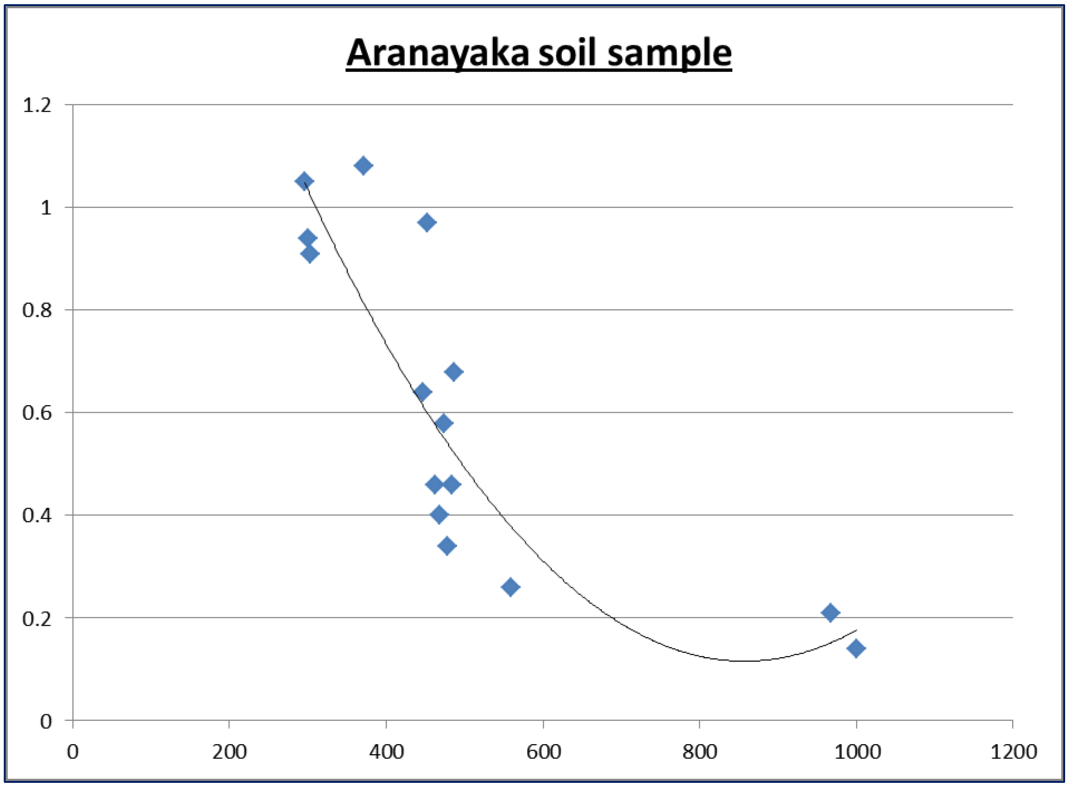


Figure 5.12 – Aranayake test sample

5.3 SMS Warning

In ThingSpeak IoT, the sensor value variation with time can be monitored. As shown in the figure below a graph is auto generated using the values fetched from the sensor unit.

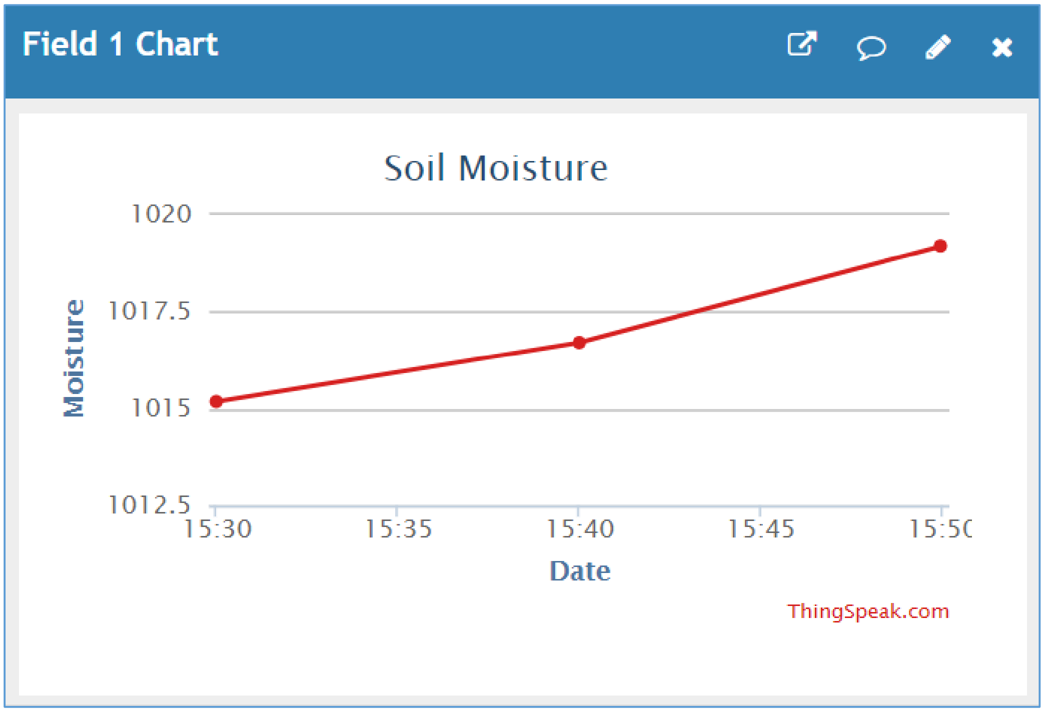


Figure 5.13 – Thingspeak sensor monitoring

ThingSpeak lets to get data in three ways: json, xml and csv. Values in the xml file can also be used for online monitoring purpose. If the moisture value drops below 500 that can be considered as a critical situation with a huge rainfall (further dropping of soil moisture can be dangerous). It is not practical to keep a human looking at the graph continuously to issue a warning message when the moisture values drop below 500. Therefore, a dedicated PC should be there to server as a server for the whole monitoring process.

In this project a python program is used to check the values in the xml file. Python program checks the xml value every few minutes and checks whether the values have reached to a critical level.

```
  Measure Soil Moisture Data and send it to Thingspeak
</description>
<latitude type="decimal">0.0</latitude>
<longitude type="decimal">0.0</longitude>
<field1>Moisture</field1>
<created-at type="dateTime">2018-04-15T07:19:49Z</created-at>
<updated-at type="dateTime">2018-05-21T10:23:10Z</updated-at>
<last-entry-id type="integer">285</last-entry-id>
▼<feeds type="array">
 ▼<feed>
    <created-at type="dateTime">2018-05-21T09:51:31Z</created-at>
    <entry-id type="integer">186</entry-id>
    <field1>1023</field1>
  </feed>
 ▼<feed>
    <created-at type="dateTime">2018-05-21T09:51:48Z</created-at>
    <entry-id type="integer">187</entry-id>
    <field1>1023</field1>
  </feed>
 ▼<feed>
    <created-at type="dateTime">2018-05-21T09:52:05Z</created-at>
    <entry-id type="integer">188</entry-id>
    <field1>1003</field1>
  </feed>
 ▼<feed>
    <created-at type="dateTime">2018-05-21T09:52:23Z</created-at>
    <entry-id type="integer">189</entry-id>
```


Moisture value in xml

Figure 5.14 – XML data

When the moisture value reaches a critical number a warning message should be sent to the civilians living in the region. To send bulk messages we need to connect with a SMS portal (e.g. Dialog, Mobitel). However, it is quite costly to use such a portal for this kind of testing purpose. Moreover, Sri Lankan mobile network companies don’t give access to the individual entities because of the possible misuse for unethical purposes. A free SMS service available in the internet was used since there were no free SMS portal to send bulk messages. A method published in the GitHub repository worked acceptably well for testing purposes (https://gist.github.com/six519/61d30310da7ded740e6a). In this method a web SMS service called http://www.afreesms.com was utilized. Below shows the steps which were automated using a web driver.

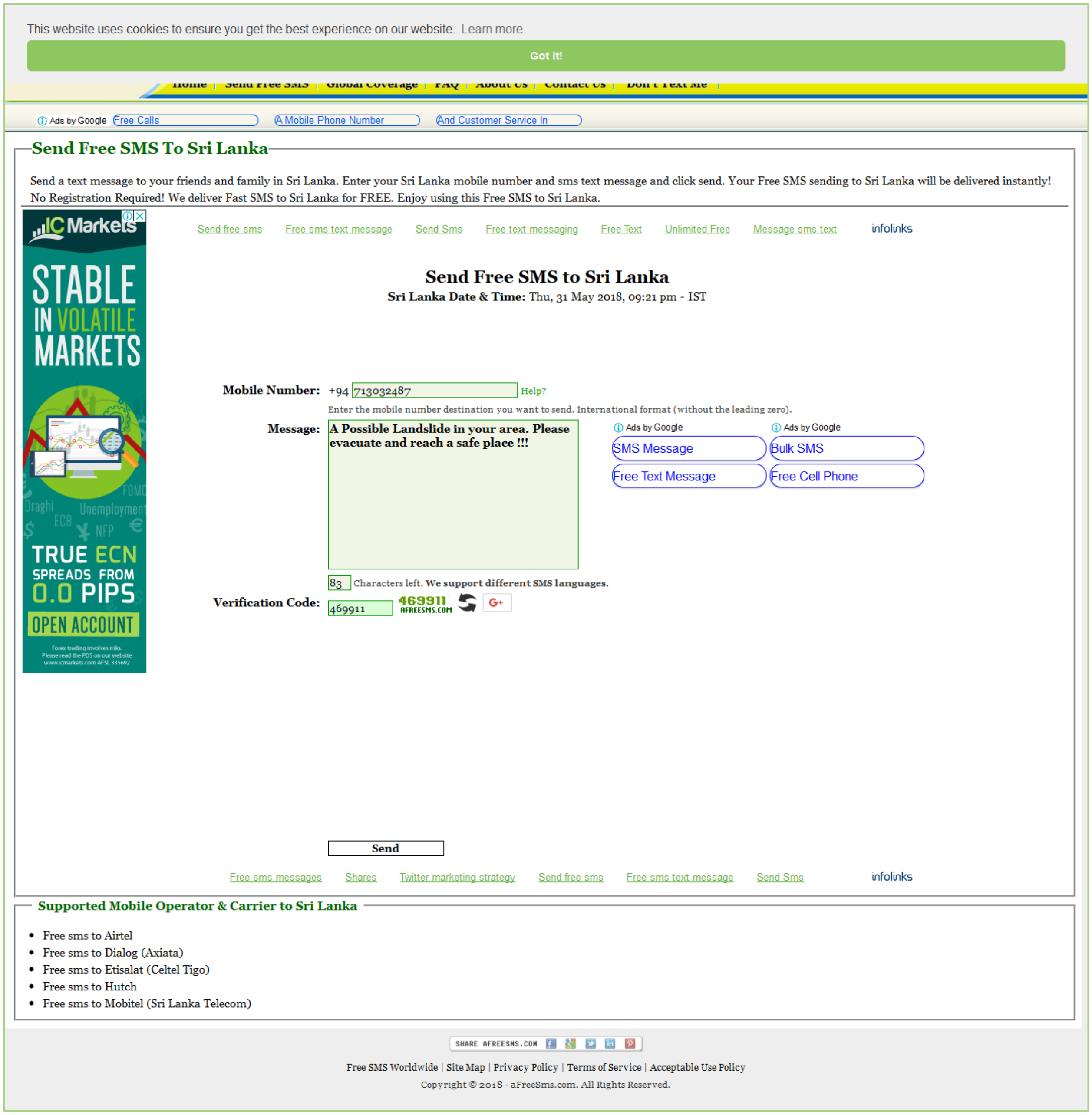


Figure 5.15 – Free SMS website

5.4 Cost Analysis

Table 5.4 - Cost analysis of the prototype

CATEGORY	QUANTITY	UNIT COST (Rs.)	COST(Rs.)
Arduino uno	1	925	925
soil moisture sensor	1	110	110
strain guage	1	170	170
voltage regulator	1	80	80
ESP 8266 wifi module	1	425	425
power supply	1	650	650
TOTAL			2360

Table 5.5 - Total cost analysis

CATEGORY	QUANTITY	UNIT COST (Rs.)	COST(Rs.)
Arduino uno	1	925	925
soil moisture sensor	4	110	440
strain guage	4	170	680
voltage regulator	1	80	80
ESP 8266 wifi module	1	425	425
power supply	1	650	650
PCB design			600
TOTAL COST			**3800**

So from the total cost analysis of our final product, it is clear that we have to spend Rs.3800/- for the system. But when we consider the above given details, and compare the amount of money that the government has to spend when a landslide disaster occurs and the amount of money that has to be spent on this system, the cost for this system is a fair value. So our final product is a cost effective one.

Chapter 06 – Conclusion

Sri Lanka faces natural disasters such as floods, landslides, cyclones, drought, lightning strikes etc. very often. Out of them, landslides have become a major topic nowadays with the incidents such as landslide at Aranayake and landslide occurred in May, 2017. Government has taken many steps to avoid the loss of lives but sometimes they fail due to various reasons. When studying this problem, we saw that Sri Lanka lacks a solution like real time early warning, to ensure the safety of the people who are already in landslides prone areas. Therefore, our aim was to implement a **"Landslides Monitoring and Early Warning System"**, which is capable in identifying the changes in some selected factors which confirms the occurrence of a landslide beforehand and warns the people in the considered area through an alarm system and a SMS system.

Though our implemented system is not actually the system what we planned to implement, it is more close to the planned one. We were unable to implement the planned system due to certain difficulties like, we obtained the data which we needed to automate the system from NBRO, but unfortunately there were not much of data available to meet our requirement. As a result, we had to conduct certain experiments in order to obtain some data like soil moisture. To do so we had to spend much time. Because of that at the end we didn't have much time to automate the system with our experimental data. Also we planned to implement a prototype to demonstrate our system but with the less time we had, Dr. Premasiri, a senior lecturer, department of Earth Resource Engineering university of Moratuwa, recommended to not to implement a prototype but to proceed with the electronic system.

Through this project, we were able to get knowledge on new technologies and operation of different type of electronic equipment. Also we were able to learn some legal aspects often used when dealing with different organizations and we got the opportunity to know higher officials in different organizations under different fields.

We hope that our landslides monitoring and early warning system will be helpful in making lives of residents in prone areas, a better one.

References

US department of the interior, 2008, The landslides handbook – A guide to understanding landslides, viewed on 05th April, 2018 from

Arduino Uno data sheet viewed on 30th April, 2018 from https://www.farnell.com/datasheets/1682209.pdf

ESP Wi-Fi module, Ai Thinker viewed on 30th April, 2018 from https://ecksteinimg.de/Datasheet/Ai-thinker%20ESP-01%20EN.pdf

Soil moisture operation, Instructables viewed on 10th April, 2018 from http://www.instructables.com/id/Arduino-Soil-Moisture-Sensor/

Abbreviations

Acronym	Definition
LMEWS	Landslides Monitoring and Early Warning System
NBRO	National Building Research Organization
SMS	Short Message Service
USA	United States of America
PDNA	Post Disaster Needs Assessments
JICA	Japan International Cooperation Agency
GDP	Gross Domestic Product
DEM	Digital Elevation Model
XML	eXtensible Markup Language
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
USB	Universal Serial Bus
ICSP	In Circuit Serial Programming
AC	Alternating Current
DC	Direct Current
FTDI	Future Technology Devices International
SRAM	Static Random Access Memory
EEPROM	Electrically Erasable Programmable Read Only Memory
GND	Ground
TTL	Time To Live
SPI	Serial Peripheral Interface
MOSI	Master Out Slave In
SS	Slave Select
MISO	Master In Slave Out
SCK	Serial Clan Killer
LED	Light Emitting Diode
TWI	T Wave Inversion

SDA	Screen Design Aid
SCA	Service Component Architecture
AREF	Ampicillin - Resistant Enterococcus Faecium
ESP	Electro Selective Pattern
SOC	System On a Chip
GPIO	General Purpose Input Output
TX	Transmit
RX	Receive
RST	Reset
AMS	Additional Member System
SCSI	Small Computer System Interface
IOT	Internet Of Things
IDE	Integrated Design Environment
JSON	Java Script Object Notation
CSV	Comma Separated Values
UART	Universal Asynchronous Receive- Transmitter

Appendix 01- Wi-Fi Module Communication

CODE 01

```
void setup() {
  /*Serial.begin(115200);
  ESP8266.begin(115200); // Change this to the baudrate used by ESP8266
  delay(1000); // Let the module self-initialize*/
}

void loop() {
  //Serial.println("Sending an AT command...");
  //ESP8266.println("AT");
  //delay(1000);
  /*while (ESP8266.available()){
    String inData = ESP8266.readStringUntil('\n');
    Serial.println("Got reponse from ESP8266: " + inData);
  } */
}
```

CODE 02

```
#include <SoftwareSerial.h>



int soil;



// replace with your channel's thingspeak API key
String apiKey = "4CUKRDC7W8PFSFI2";
```

```
// connect 2 to TX of ESP
// connect 3 to RX of ESP
SoftwareSerial ser(2,3); // RX, TX

// this runs once
void setup() {

  // enable debug serial
  Serial.begin(115200);
  // enable software serial
  ser.begin(115200);

  // reset ESP8266
  //ser.println("AT+RST");
}



// the loop
void loop() {
  soil = analogRead(A0);
  esp_8266();
}



void esp_8266()
{
 // convert to string
```

```
String soil_send = String(soil);
//Serial.print(soil_send);
// TCP connection
String cmd = "AT+CIPSTART=\"TCP\",\"";
cmd += "184.106.153.149"; // api.thingspeak.com
cmd += "\",80";
ser.println(cmd);

if(ser.find("Error")){
  Serial.println("AT+CIPSTART error");
  return;
}

// prepare GET string
String getStr = "GET /update?api_key=";
getStr += apiKey;
getStr +="&field1=";
getStr += soil_send;
getStr += "\r\n\r\n";

// send data length
cmd = "AT+CIPSEND=";
cmd += String(getStr.length());
ser.println(cmd);

if(ser.find(">")){
  ser.print(getStr);
```

```
  }
  else{
    ser.println("AT+CIPCLOSE");
    // alert user
    Serial.println("AT+CIPCLOSE");
  }

  // thingspeak needs 15 sec delay between updates
  delay(16000);
}
```

APPENDIX 02- SOIL MOISTURE SENSOR COMMUNICATING

```
#include <SoftwareSerial.h>



int soil;



// replace with your channel's thingspeak API key
String apiKey = "4CUKRDC7W8PFSFI2";

// connect 2 to TX of ESP
// connect 3 to RX of ESP
SoftwareSerial ser(2,3); // RX, TX

// this runs once
void setup() {
```

```
  // enable debug serial
  Serial.begin(115200);
  // enable software serial
  ser.begin(115200);

  // reset ESP8266
  //ser.println("AT+RST");
}


// the loop
void loop() {
  soil = analogRead(A0);
  esp_8266();
}


void esp_8266()
{
 // convert to string
  String soil_send = String(soil);
  //Serial.print(soil_send);
  // TCP connection
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com
  cmd += "\",80";
```

```
ser.println(cmd);

if(ser.find("Error")){
  Serial.println("AT+CIPSTART error");
  return;
}

// prepare GET string
String getStr = "GET /update?api_key=";
getStr += apiKey;
getStr +="&field1=";
getStr += soil_send;
getStr += "\r\n\r\n";

// send data length
cmd = "AT+CIPSEND=";
cmd += String(getStr.length());
ser.println(cmd);

if(ser.find(">")){
  ser.print(getStr);
}
else{
  ser.println("AT+CIPCLOSE");
  // alert user
  Serial.println("AT+CIPCLOSE");
}
```

```
  // thingspeak needs 15 sec delay between updates
  delay(16000);
}
```

Apprndix 02- Soil Moisture Sensor Communication

Code 1

```
#include <SoftwareSerial.h>



int soil;



// replace with your channel's thingspeak API key
String apiKey = "4CUKRDC7W8PFSFI2";

// connect 2 to TX of ESP
// connect 3 to RX of ESP
SoftwareSerial ser(2,3); // RX, TX

// this runs once
void setup() {

  // enable debug serial
  Serial.begin(115200);
  // enable software serial
  ser.begin(115200);
```

```
  // reset ESP8266
  //ser.println("AT+RST");
}


// the loop
void loop() {
  soil = analogRead(A0);
  esp_8266();
}


void esp_8266()
{
 // convert to string
  String soil_send = String(soil);
  //Serial.print(soil_send);
  // TCP connection
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // api.thingspeak.com
  cmd += "\",80";
  ser.println(cmd);

  if(ser.find("Error")){
    Serial.println("AT+CIPSTART error");
    return;
```

```
 }

 // prepare GET string
 String getStr = "GET /update?api_key=";
 getStr += apiKey;
 getStr +="&field1=";
 getStr += soil_send;
 getStr += "\r\n\r\n";

 // send data length
 cmd = "AT+CIPSEND=";
 cmd += String(getStr.length());
 ser.println(cmd);

 if(ser.find(">")){
   ser.print(getStr);
 }
 else{
   ser.println("AT+CIPCLOSE");
   // alert user
   Serial.println("AT+CIPCLOSE");
 }

 // thingspeak needs 15 sec delay between updates
 delay(16000);
}
```

Appendix 03- Python code of converting sauce file into XML format

Code 1

```
#Thingspeak_crawl
import bs4 as bs
import urllib.request as ureq
import pandas as pd

sauce = ureq.urlopen('https://thingspeak.com/channels/476078/feed.xml').read()
soup = bs.BeautifulSoup(sauce,'xml') #specify the parser

for label in soup.find_all('field1')[1:]:
            print(label.text)
```