CHAPTER 1
INTRODUCTION

1.1 Overview

The earth engulfs water from precipitation and it becomes a part of rivers, ground waters etc. The water at upper layer of soil surface (usually 5 to 10 cm) is known as soil moisture. Soil moisture measurement is an essential component of our irrigational lands. There is variation in soil moisture measurement which occurs due to climate variability, atmospheric changes, flood magnitude and frequencies. Direct measurements of soil moisture are labor intensive and it does not provide the accurate data of soil moisture. Therefore, we have developed an IoT based sensor which can provide us with reliable and accurate data of soil moisture. The developed soil moisture sensor can be used to measure the gravimetric content of water inside the soil depending on the resistivity or conductivity of the soil. This soil moisture sensor that can be used in many applications such as bio mediation, irrigation management, landfill management, wastewater reclamation, crop yield forecasting, issuing early warnings of droughts. The main focus of developing a sensor is judicious use of water needed in agriculture. The only way for providing optimum requirements to the crops is the accurate measurement of soil moisture. Different types of soil have different water holding capacity due to the different soil textures i.e., its different percentage of sand, silt and clay in the soil. The developed soil moisture sensor is inexpensive and is capable in providing real time data and improves the irrigation efficiency. This sensor is easy to install and requires very less maintenance. Soil Moisture Probes use Dielectric Reflectometry method for soil moisture measurement. Gravimetric Probes weighs the sample and dry it for 24 hours or use an oven and re -weigh how much moisture have been lost. This technique is very convenient and inexpensive as well as cinch to use but the time involved to evaluate soil moisture is very long. In the case of remote sensing, it is expensive and not cost-effective for collecting details for a small area. Data collection for small area, specialist training, equipment and maintenance becomes costly as compared with larger areas. Human induced errors are possible in remote sensing during the process of interpretation and analysis. Therefore, we have designed an optimized device that can notify the soil moisture value using a soil moisture sensor through cloud. It also indicates the type of area with the help of GPS module and logs the data in SD card. It also has push buttons which indicate the area we are present in; i.e. rural, urban, water bodies etc. The advantages are that it is inexpensive and there are fewer errors with optimum number of resources. It has long term stability with recalibrations and requires less maintenance. It is a device that is adaptable by any strata of the country. We can take down the moisture content by graving the soil moisture sensor into the ground or farm and get to know the water level and condition of soil at our devices like PC’s, Phones etc. The sensor measures the gravimetric content of water in the soil.

1.2 Problem Statement

IoT Based System to measure Soil Moisture using Soil Moisture sensor, GPS data logging and Cloud storage.

1.3 Motivation And Challenges

As we all know that soil moisture plays vital role in plant growth and provides them nutrients. Knowing the soil moisture status enables highly efficient irrigation, providing the water as and when required, and eliminating the wasteful use of water when irrigation is not needed. Roots can penetrate soil so that the plant can anchor itself and also so that it can access the nutrients and water stored in the soil. Therefore, an efficient device is useful in agricultural field to determine the moisture content of the soils which is inexpensive and reliable than other sensors.

1.4 Challenges

Problems faced in any technical project are never sparse. Most often an important part of the project becomes overcoming the innumerable obstacles faced. Here are some of the problems encountered on the way to finishing our project:

1. **Problem embedding the components with the Microcontroller:**

   While interfacing the components like the soil sensor, I had faced a lot of problems in getting the values being displayed on the serial monitor and LCD screen. The biggest problem faced was the alignment of the wiring. The error detection was tedious as the connections were reestablished if they were broken once at any point in Arduino. Interfacing with Arduino is not an easy task as even one misplaced wire can lead to no results.

2. **Problem in interfacing all the components all together:**

   Making all the components work together so that the final product work properly is not an easy task so many coding challenges required some hardware issues like same ports on the board that cause failure in working of the device properly. Same connection pin number resulted in not working of the system. Each component was interfaced one by one and tested for its working.

3. **Problem in soldering all the components on zero PCB**

   The biggest task was to solder all the devices without damaging them. We had to make the circuit design and then solder the wires to the devices and further to the PCB. The biggest problem faced was the loosening of the wires resulting in short circuits or open circuits. These were very difficult to detect as it hampered the outputs on the screen and LCD.

1.5 Working of the soil moisture sensor

The soil moisture device probes are immersed in the soil. It returns the soil moisture in percentage on the LCD screen. It also displays the time and date simultaneously. While being immersed into the soil it gets the data from the GPS module by providing us with the corresponding latitude and longitude of that area of action. All these data are logged in an SD card for further judicious use.

1.6 Interface

Interfacing involves combining of the components together to make a proper functioning device. We have interfaced 6 components with Arduino Uno board which is a microcontroller board based on the ATmega328P, the heart of the whole system. The other devices are interfaced to the board. We have interfaced 20 x 4 LCD to display all the readings of the moisture, date, time, longitude, latitude on it. We have used an SD card module to log all the data into its memory in a form of a file. GPS Neo 6M is used which is a GPS Module for fetching the location. We have also interfaced a set of push buttons to provide information to us with the type of land we are present in. Further, developed soil sensor is interfaced which provides us with the readings of the soil moisture when the probes are dipped into the soil. All these components are interfaced to the microcontroller ATmega328P.

1.7 Hardware Implementation

1. **Arduino Uno ATmega328P:** The Arduino Uno is a microcontroller board based on the AT-mega328P, it's the heart of the whole system.

2. Developed Soil moisture sensor: Developed Soil Moisture Probes use Dielectric Reflectometry method for soil moisture. It has a potentiometer which sets the threshold value which is compared by LM393 comparator, to determine the resistivity/conductivity of soil.

3. SD Card: it is used to store the data in a particular test file.

4. 20 X 4 LCD: This 20x4 Character LCD Display is built-in with RW1063 controller IC which are 6800, 4 line SPI or I2C interface options.

5. GPS Neo 6M: The NEO-6M GPS module is a well-performing complete GPS receiver with a built-in 25 x 25 x 4mm ceramic antenna, which provides a strong

satellite search capability. With the power and signal indicators, you can monitor the status of the module.

6. Push Buttons: A push-button (also spelled pushbutton) or simply button is a simple switch mechanism to control some aspect of a machine or a process.

1.8 Applications

Soil moisture sensor has wide variety of applications. The following are listed as follows:

1. Agriculture

Measuring soil moisture is important for agricultural applications to help farmers manage their irrigation systems more efficiently. Knowing the exact soil moisture conditions on their fields, not only are farmers able to generally use less water to grow a crop, they are also able to increase yields and the quality of the crop by improved management of soil moisture during critical plant growth stages.

2. Landscape irrigation

In urban and suburban areas, landscapes and residential lawns are using soil moisture sensors to interface with an irrigation controller. Connecting a soil moisture sensor to a simple irrigation clock will convert it into a "smart" irrigation controller that prevents irrigation cycles when the soil is already wet, e.g. following a recent rainfall event.

3. Research

Soil moisture sensors are used in numerous research applications, e.g. in agricultural science and horticulture including irrigation planning, climate research, or environmental science including solute transport studies and as auxiliary sensors for soil respiration measurements.

4. Simple sensors for gardeners

Relatively cheap and simple devices that do not require a power source are available for checking whether plants have sufficient moisture to thrive. After inserting a probe into the soil for approximately 60 seconds, a sensor indicates if the soil is too dry, moist or wet for plants.

CHAPTER 2

MICROCONTROLLER

2.1 Microcontroller

A microcontroller (sometimes abbreviated µC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

Some microcontrollers may use four-bit words and operate at clock rate frequencies as low as 4 kHz, for low power consumption (single-digit mill watts or microwatts). They will generally have the ability to retain functionality while waiting for an event such as a button press or other interrupt; power consumption while sleeping (CPU clock and most peripherals off) may be just nano watts, making many of them well suited for long lasting battery applications. Other microcontrollers may serve performance-critical roles, where they may need to act more like a digital signal processor (DSP), with higher clock speeds and power consumption.

2.1.1 Different families of microcontroller

8051

8051 manufactured in 1985. This is an 8-bit microcontroller. The width of the register represents the bit number of microcontroller. For example 89C51 has 8-bit register, so 89C51 is 8-bit microcontroller. In this you can store numbers from 0 to 255, in hexadecimal it is represented as 0x00 to 0xFF. Coming to the instruction set 8051 has 250 instructions which take 1 to 4 machine cycles to execute. The speed of the 8051 microcontroller is 1 million instructions per second. 8051 has powerful instruction set; it has commands which perform more complex calculations. The ALU of the 8051 makes computations simple. In 8051 family there is no inbuilt memory bus and A/D converters. 8051 microcontroller has 32 I/O pins, timers/counters, interrupts and UART's.

AVR

AVR is an 8-bit RISC architecture microcontroller. This is available from 1996 onwards only. There are 16-bit and 32-bit microcontrollers also available in the same family. RISC means Reduced Instruction Set Computer. AVR has 140 instructions which are all 1 cycle based instructions. By default AVR microcontrollers operate with the 1 MHz clock cycle. The speed of AVR microcontroller is 12 million instructions per second. AVR family microcontroller has on-chip boot-loader. By this we can program our microcontroller easily without any external programmer. AVR controllers has number of I/O ports, timers/counters, interrupts, A/D converters, USART, I2C interfaces, PWM channels, on-chip analog comparators.

PIC

PIC (Programmable interface controller) microcontrollers are available in 3 different architectures. Those are 8-bit, 16-bit and 32-bit microcontrollers. PIC has nearly 40 instructions which all are take 4 clock cycles to execute. The speed of the PIC controller is 3 million instructions per second. The programming part of the PIC microcontroller is very hard. So those who entering into embedded world freshly this is not preferable for them. It has on-chip peripherals like SPI, ADC, I2C, UART, analog comparator, internal RC oscillator, in-system programmability, etc.

ARM

At the time of manufacturing ARM was named as Acorn RISC Machine. Later ARM limited was established in1990. From then onwards ARM renamed as Advanced RISC machine. This is the advanced RISC controller. Most of the industries get license from ARM limited. ARM has the features like load-store architecture, fixed-length instruction set and 3-address instruction format. It has 32-bit ARM instruction set and 16-bit Thumb compressed instruction set. So many on-chip peripheral are there and on-chip debugger, on-chip boot loaders, on-chip RTC, DAC also available.

2.1.2 Types of microcontrollers used in IoT projects

There are many microcontrollers available that are used as a building foundation for any IoT project. From those controllers selecting a proper controller for our application need is very important. First we have to know about what is our application, what are the requirements for that particular application. How many inputs & outputs are required for the particular application and what are the inputs and outputs. Before going to select the controller device, first we want to select proper microcontroller family. If a beginner to the embedded systems & microcontrollers, 8051 family is sufficient for you because 8051 is the basic microcontroller and is easy to learn and program. If having some knowledge about microcontrollers and programming, can select any of other family of microcontroller reaching our requirements.

The Atmel 8-bit AVR (Advanced Virtual RISC) RISC-based microcontroller is used for the project. The point of choosing this microcontroller is it is cheap and all the resources are available as open source applications.
Some of the microcontroller families and controllers:

Atmel

1. AT89 series microcontrollers
2. AT-tiny series
3. At-mega AVR
4. 4 – 512 kb program memory
5. 28 – 100-pin package
6. Extended instruction set
7. Extensive on-chip peripherals set

X-mega

1. 16 – 384 kb program memory
2. Extended performance features
3. Number of on-chip peripherals
4. Cryptography supported

2.2 ATmega328

The ATmega 328 is a single-chip microcontroller created by Atmel in the megaAVR family (later Microchip Technology acquired Atmel in 2016). It has a modified Harvard architecture 8-bit RISC processor core. The Atmel 8-bit AVR RISC-based microcontroller combines 32 KB ISP flash memory with read-while-write capabilities, 1 KB EEPROM, 2 KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. The device achieves throughput approaching 1 MIPS per MHz

ATmega328 is commonly used in many projects and autonomous systems where a simple, low-powered, low-cost micro-controller is needed. Perhaps the most common implementation of this chip is on the popular Arduino development platform, namely the Arduino Uno and Arduino Nano models.

2.3 Pin configuration of microcontroller Atmega328P

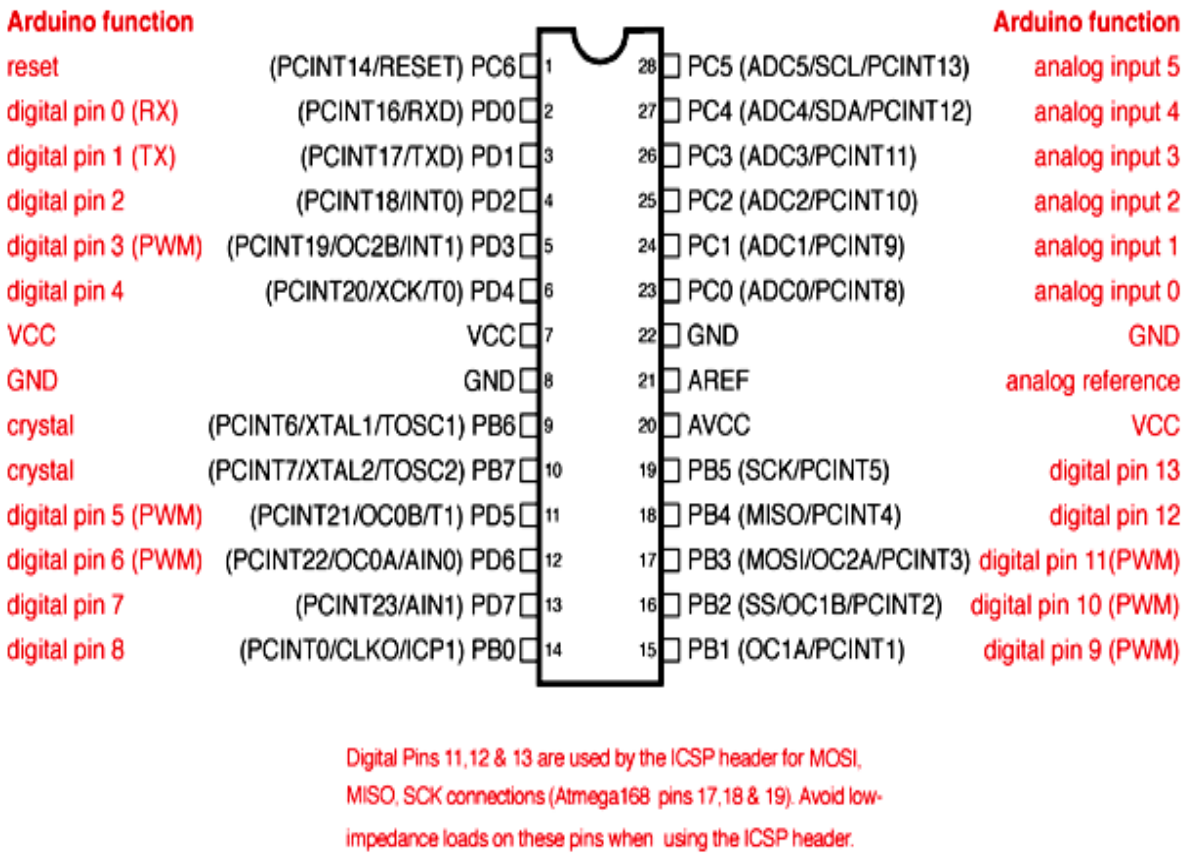

Figure 2.1 Pin allignment of Atmega328P

2.3.1 Features of Atmega328P

Table 2.1 Pin description of AT-Mega328P

Parameter	Value
CPU Type	8-bit AVR
Performance	20 MIPS at 20 MHz
Flash Memory	32KB
SRAM	2 KB
EEPROM	1 KB
Pin Count	28 or 32 pin
Maximum operating frequency	20 MHz
Number of touch channels	16
Hardware Qtouch Acquisition	No
Maximum I/O pins	23
External Interrupts	2

CHAPTER 3

COMPONENTS USED IN THE DEVELOPMENT OF SOIL MOISTURE SENSOR

3.1 Arduino Uno

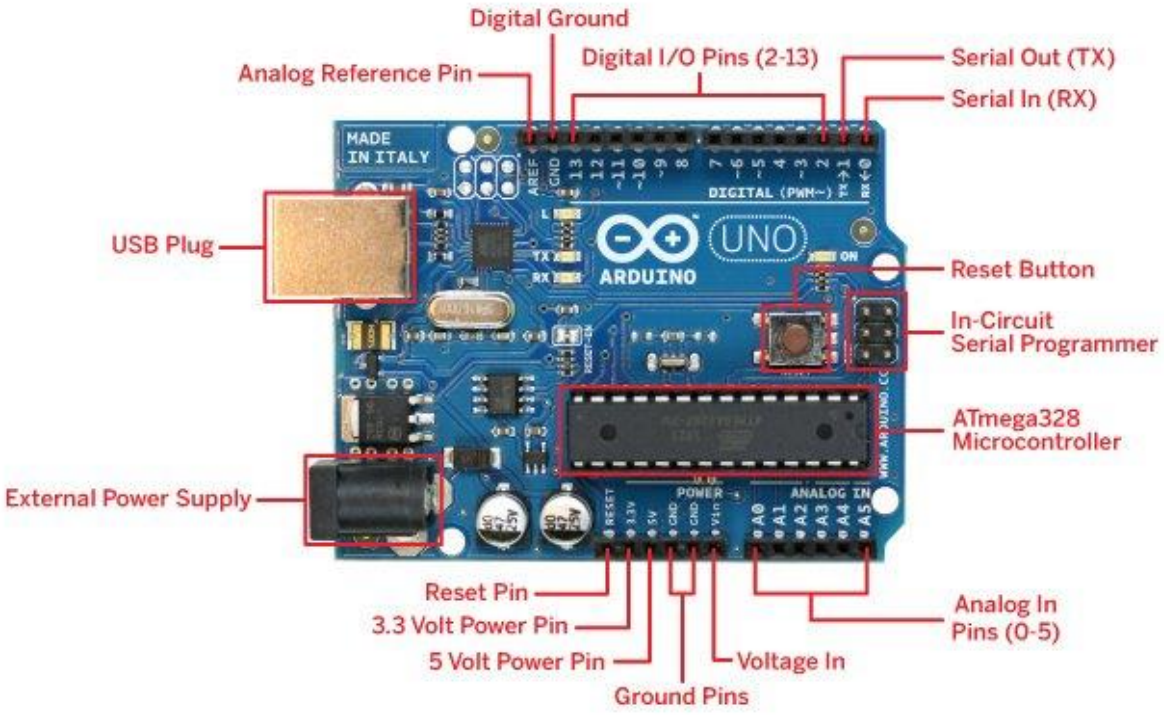

Figure 3.1. Arduino Uno

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo. The word "Uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes preprogrammed with a boot loader that allows uploading new code to it without the use of an external hardware programmer.

While the Uno communicates using the original STK500 protocol, it differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it

uses the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Voltage regulator

The Arduino Uno can be powered by USB cable or directly supplying 9-12v from the barrel jack. The circuitry operates at 5v dc which in case input more than that is regulated with the help of 7805 voltage regulator. The IC is used regulate the voltage supplied to the arduino board and manage it through processor and other elements.

Crystal Oscillator

There are certain case when the processor has to deal with time-signal issues, in order to balance it the crystal oscillator is used. The crystal oscillator is the only way the arduino is able to calculate the time. There is a number printed on the top of the crystal. The number indicates the frequency of the crystal, in most of them the frequency is 16 MHZ or 16,000,000 hertz.

Reset Button

There is a reset button given which is used to restart the program running in the Arduino Uno. There are two ways to restart the whole program.
1. You can use the default reset button.
2. You can connect your own reset button at the pin labeled as Reset.

General pin functions

LED: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it is off.

VIN: The input voltage to the Arduino/Genuino board when it is using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.

3V3: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND: Ground pins.

IOREF: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source, or enable voltage translators on the outputs to work with the 5V or 3.3V.

Special pin functions

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, under software control (using pinMode (), digitalWrite (), and digitalRead () functions). They operate at 5 volts. Each pin can provide or receive 20 mA as the recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50K ohm. A maximum of 40mA must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labeled A0 through A5; each provides 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though it is possible to change the upper end of the range using the AREF pin and the analogReference () function.
In addition, some pins have specialized functions:

Serial / UART: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.

External interrupts: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

PWM (pulse-width modulation): pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the analogWrite () function.

SPI (Serial Peripheral Interface): pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.

TWI (two-wire interface) / I²C: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.

AREF (analog reference): Reference voltage for the analog inputs.

Communication

The Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, 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. However, on Windows, an .inf file is required. Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the 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). A SoftwareSerial library allows serial communication on any of the Uno's digital pins.

Software

Arduino IDE (Integrated Development Environment) is required to program the Arduino Uno board.

Summary

Table 3.1 Arduino Uno Technical Specifications

Microcontroller	ATmega328P-8 bit AVR family
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 pins	50 mA
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Frequency(clock speed)	16 MHz

3.2 20×4 LCD(Liquid Crystal Display)

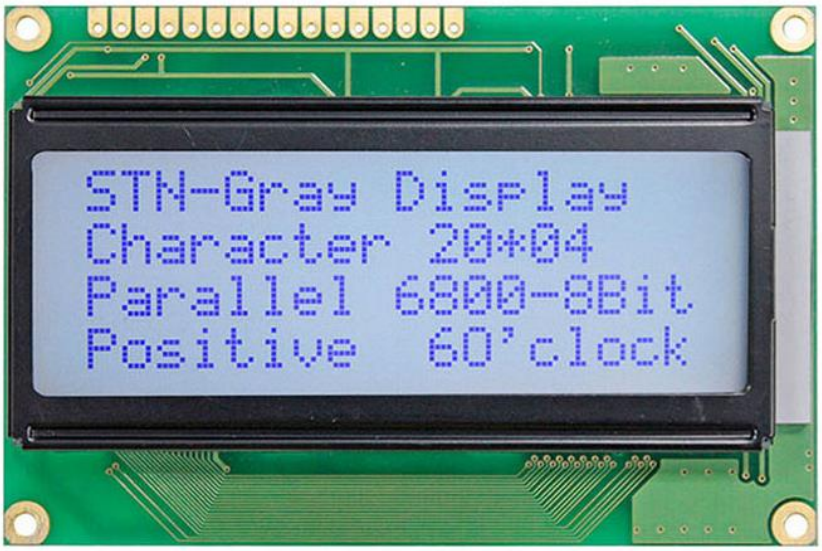

Figure 3.2 20×4 LCD

This is a 20×4 LCD Arduino compatible LCD display module with high speed I2C interface. It is able to display 20×4 characters on two lines, white characters on blue background. LCD display will run out of Arduino pin resource. It needs 6 digital pins and 2 power pin for a LCD display. This I2C 20×4 LCD display module is designed for Arduino Microcontroller. It is using I2C communication interface, with this I2C, only 2 lines(I2C) are required to display the information on any Arduino based projects. It will save at least 4 digital/ analog pins on Arduino. All connector are standard XH2.54(Breadboard type). You can connect it with jumper wire.

Specifications

- Character LCD 20x4
- Adjustable contrast
- Wide viewing angle
- 5x8 dots includes cursor
- Character size: 2.96 × 4.75(0.11 × 0.19)
- Size: 98×60×24mm
- Built-in controller (RW1063 or Equivalent)
- +5V power supply (Also available for +3V)
- Negative voltage optional for +3V power supply
- 1/16 duty cycle
- LED can be driven by PIN1, PIN2, PIN15, PIN16 or A and K
- Interface : WH2004G - 6800, WH2004G1 - SPI, WH2004G2 - I2C

3.3 NEO 6M GPS Module

Figure 3.3. NEO 6M GPS Module

The NEO-6M GPS module is a well-performing complete GPS receiver with a built-in 25 x 25 x 4mm ceramic antenna, which provides a strong satellite search capability. With the power and signal indicators, you can monitor the status of the module. The data backup battery, the module can save the data when the main power is shut down accidentally. Its 3mm mounting holes can ensure easy assembly on your aircraft, which thus can fly steadily at a fixed position, return to Home automatically, and automatic waypoint flying, etc. Or you can apply it on your smart robot car for automatic returning or heading to a certain destination, making it a real "smart" bot! The GPS modules are based on the u-blox NEO-6M GPS engine. The type number of the NEO-6M is NEO-6M-0-001, and its ROM/FLASH version is ROM 7.0.3 (PCN reference UBX-TN-11047-1). The NEO-6M module includes one configurable UART interface for serial communication, but the default UART (TTL) baud rate here is 9,600. Because the GPS signal is right-hand circular-polarized (RHCP), the style of the GPS antenna will be different from the common whip antennas used for linear polarized signals. The most popular antenna type is the patch antenna. Patch antennas are flat, generally have a ceramic and metal body, and are mounted on a metal base plate. They are often cast in a housing. The position of the antenna mounting is very crucial for optimal performance of the GPS receiver. When using the patch antenna, it should be oriented parallel to the geographic horizon. The antenna must have full view of the sky, ensuring a direct line of sight with as many visible satellites as possible.

Features

- A complete GPS module with an active antenna integrated, and a built-in EEPROM to save configuration parameter data.

- Built-in 25 x 25 x 4mm ceramic active antenna provides strong satellite search capability.
- Equipped with power and signal indicator lights and data backup battery.
- Power supply: 3-5V
- Default baud rate: 9600bps.
- Interface: RS232 TTL

3.4 SD Card Module

Figure 3.4. SD Card module

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.

Storing data is one of the most important parts of every project. There are several ways to store data according to the data type and size. SD and micro SD cards are one of the most practical ones among the storage devices, which are used in devices such as mobile phones, minicomputers and etc.

Specifications

- Operating voltage 5V
- SPI Communication method
- SD card Socket
- Supports FAT16 and FAT32
- Support 2gb to 4gb

Pin Wiring to Arduino Uno

Table 3.2 Pin Wiring

SD Card Module	Wiring to Arduino Uno
VCC	3.3V or 5V
CS	4
MOSI	11
CLK	13
MISO	12
GND	GND

3.5 Developed Soil Moisture Sensor

Figure 3.5 Developed soil moisture sensor

Soil moisture sensors measure the volumetric water content in soil. Since the direct gravimetric measurement of free soil moisture requires removing, drying, and

weighing of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content.
The relation between the measured property and soil moisture must be calibrated and may vary depending on environmental factors such as soil type, temperature, or electric conductivity. Reflected microwave radiation is affected by the soil moisture and is used for remote sensing in hydrology and agriculture. Portable probe instruments can be used by farmers or gardeners.
Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks.
We have developed a soil moisture sensor by cutting two steel rods of equal length that has been attached to the comparator with specified width between the two probes. The developed IoT based sensor provides us with reliable and accurate data of soil moisture. The developed soil moisture sensor can be used to measure the gravimetric content of water inside the soil depending on the resistivity or conductivity of the soil. The developed soil moisture sensor is inexpensive and is capable in providing real time data and improves the irrigation efficiency. This sensor is easy to install and requires very less maintenance. The soil moisture sensor uses two probes to measure the presence of water content in soil. It has a potentiometer which sets the threshold value which is compared by LM393 comparator, to determine the resistivity/conductivity of soil. The resistivity will be less for high moisture content in soil and vice –versa.

Pin configuration of the Soil Moisture Sensor
This module uses 4 pins:

- VCC pin is used for power
- A0 pin is an analog output
- D0 pin is a digital output
- GND pin is a Ground

This module also includes a potentiometer that will fix the threshold value, & the value can be evaluated by the comparator-LM393.

Working Principle

Soil Moisture Probes use Dielectric Reflectometry method for soil moisture measurement. It uses gravimetric method and mainly utilizes capacitance to gauge the water content of the soil (dielectric permittivity). The working of this sensor can be done by inserting this sensor into the earth and the status of the water content in the soil can be reported in the form of a percent.

Specifications

The specification of this sensor includes the following:

- The required voltage for working is 5V
- The required current for working is <20mA
- Type of interface is analog
- The required working temperature of this sensor is 10°C~30°C

3.6 Push Buttons

Figure 3.6. Push Button

The pushbutton is a component that connects two points in a circuit when you press it. We connect three wires to the Arduino board. The first goes from one leg of the pushbutton through a pull-up resistor (here 2.2 KOhms) to the 5 volt supply. The second goes from the corresponding leg of the pushbutton to ground. The third connects to a digital i/o pin (here pin 7) which reads the button's state.
When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to 5 volts (through the pull-up resistor) and we read a HIGH. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to ground, so that we read a LOW. (The pin is still connected to 5 volts, but the resistor in-between them means that the pin is "closer" to ground.)

The push button in our project plays a vital role in operating a menu. It has 4 options i.e. rural area, urban area, water bodies and barren land. It detects the area on the button being pressed and then displays the location of that particular area. It provides the latitude and longitude of that area. On the LCD the menu gets displayed with its corresponding options and when we select an option using push button, it displays the type of land we are in with its corresponding location coordinates on the LCD.

Chapter 4

IoT based system to measure Soil Moisture

4.1 Introduction

As we know that soil moisture measurement is an essential component of our topography. There is variation in soil moisture measurement occurs due to climate variability, atmospheric changes, flood magnitude and frequencies. So, we use soil moisture sensor as direct measurements of soil moisture are labor intensive and it does not have accuracy and reliability. Therefore, we have developed an IoT based sensor which overcomes all these problems faced. The developed soil moisture sensor can be used to measure the gravimetric content of water inside the soil depending on the resistivity or conductivity of the soil. This soil moisture sensor has many applications such as bio mediation, irrigation management, landfill management, wastewater reclamation, crop yield forecasting, issuing early warnings of droughts. The main focus of developing a sensor is judicious use of water needed in agriculture. The only way for providing optimum requirements to the crops is the accurate measurement of soil moisture. Different types of soil have different water holding capacity due to the different soil textures i.e., its different percentage of sand, silt and clay in the soil. The developed soil moisture sensor is inexpensive and is capable in providing real time data and improves the irrigation efficiency. This sensor is easy to install and requires very less maintenance. The soil moisture sensor is tested with 16 soil samples. It is calibrated with respect to the responses to natural drying and wetting by using water balance and infiltration. The sensor responds to most of the precipitation events. . Human induced errors are possible in remote sensing during the process of interpretation and analysis. Therefore, we have designed an optimized device indicates the type of area with the help of GPS module and logs the data in SD card. The push buttons in the device tells us the area that we are present in with their corresponding location coordinates. The advantages are that it is inexpensive and there are fewer errors with optimum number of resources. It has long term stability with recalibrations and requires less maintenance. It is a device that is adaptable by any strata of the country. We can take down the moisture content by graving the soil moisture sensor into the ground or farm and get to know the water level and condition of soil at our devices like PC's, Phones etc. The sensor measures the gravimetric content of water in the soil. In this chapter we will learn about the actual working of device.

4.2 Working of the soil moisture sensor

The soil moisture sensor is designed in such a way that when the probes are put into the soil, it will give the soil moisture in the percentage form. It measures the soil moisture by the gravimetric method. The values of the moisture as well as the date and time are mentioned. It also gives us the location of the area where the soil moisture is

being tested by giving the latitude and longitude of the place. Further, the flow chart of the device is discussed to have a clearer picture on methodology.

4.3 Usage as an effiecient device

Agricultural land is where the need of proper measurement of soil moisture is mandatory for efficient farming. So, we have designed an efficient device that is portable to anyone. It can be carried anywhere and the soil moisture will be shown onto the LCD. It will give accurate readings under any geographical condition. It is tested under various conditions to give accurate and precise readings. It is tested with water samples as well as with 16 soil samples of diverse areas. It has consistent readings as it has very less RMSE (Root mean square error) value. We have interfaced it with micro SD card logger and GPS gives the location and the type of area we are present along with its latitude and longitude. It is more efficient to use this sensor as it is inexpensive and reliable than other sensors.

4.4 Comparison with other sensors

There are different types of Soil Moisture Sensors with the different technologies. Some of them are listed as follows:

- **Time Domain Reflectometry (TDR) Sensors**
  These devices require a device to generate the electronic pulse and need to be carefully calibrated in order to precisely measure the amount of time it takes for the pulse to propagate down the line and back again. They are also sensitive to the saline content of salt and relatively expensive compared to some measurement methods.
- **Frequency Domain Reflectometry (FDR) Sensors**
  There are many soil moisture probes on the market today that use the Frequency Domain Reflectometry method (FDR) of soil measurement. This method of measurement also uses an oscillator to propagate an electromagnetic signal through a metal tine or other wave guide, but with this method, the difference between the output wave and the return wave frequency is measured to determine soil moisture.
- **Gravimetric developed sensor**
  Gravimetric soil moisture measurement involves taking a sample of the soil from the site, weighing the sample, drying it in an oven for 24 hours and then re-weighing it to determine how much water was lost. Soil Moisture Probes use Dielectric Reflectometry method for soil moisture measurement. This soil measurement technique is inexpensive and easy to execute. In the case of remote sensing, it is expensive and not cost-effective for collecting details for a small area. The advantages are that it is inexpensive and there are fewer errors with optimum number of resources. It has long term stability with

recalibrations and requires less maintenance. Other sensors just provide us with the moisture percentage, but our developed sensor will provide us with the information as to the type of land we are present and also provide us with the current location with date and time. The data will also be logged into the SD card. This makes our device more efficient.

4.5 Flowchart of the device

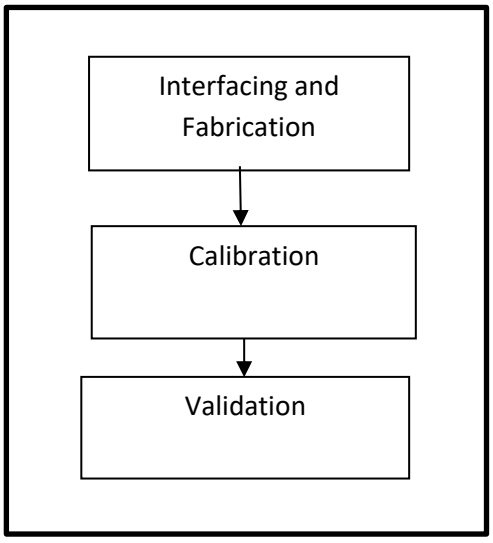

Figure 4.1 Outline of the device

4.5.1 Interfacing and Fabrication of device

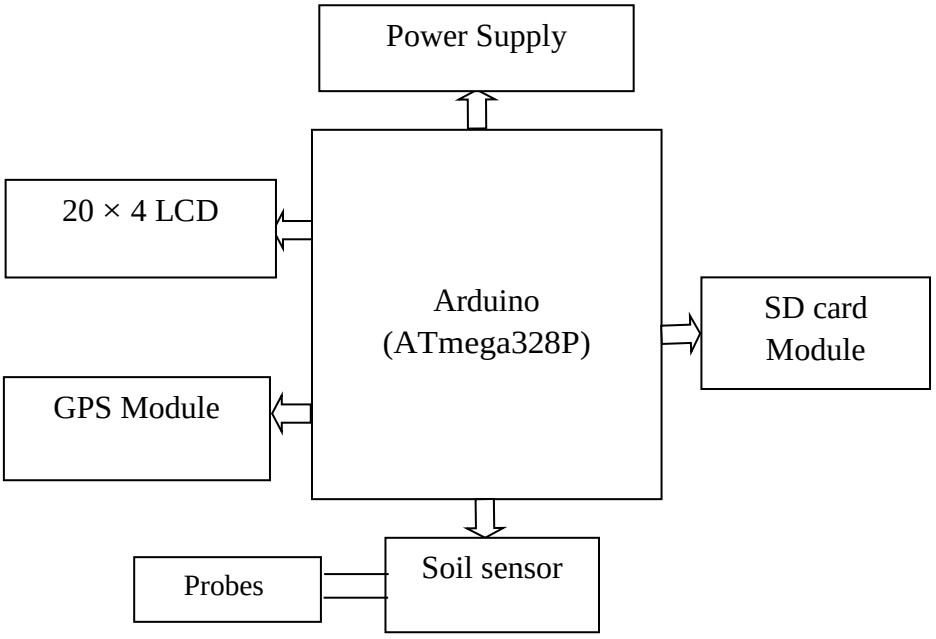

Figure 4.2 Block diagram of the device

4.5.2 Calibration of the device

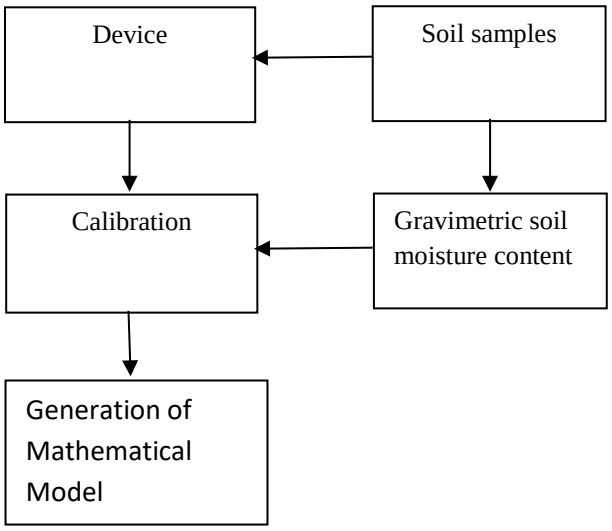

Figure 4.3 Steps for calibration of Device

4.5.3 Validation of the device

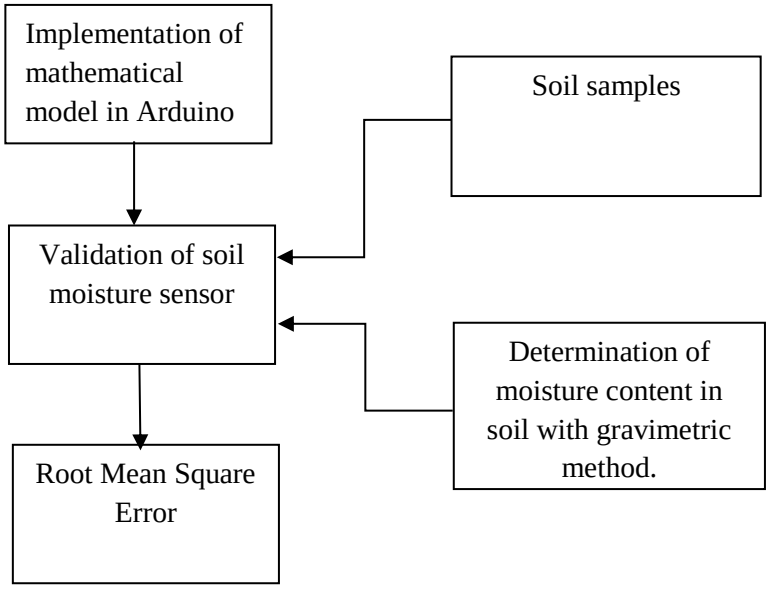

Figure 4.4 Steps for validation of Device

4.6 Methodology

Following steps explains the methodology used in the development of IoT based soil moisture monitoring system. We refer to above figures as the flowchart of methodology.

Step 1: The first step involves the interfacing of the components with the Arduino UNO board and formation of soil moisture sensor device. In this step, we develop an Arduino code for interfacing all the components with each other. The hardware device has been implemented in a compact form on zero PCB by wire connections and soldering.

Step 2: This step involves the calibration of the device. We will prepare different soil samples with different moisture level in soil. These soil samples will be used to determine the relationship between the soil moisture and designed sensor reading and provided by the soil moisture sensor. The ground truth soil moisture will be measured with gravimetricmethod.

Step 3: In this step, we will develop an empirical relationship between soil moisture measured with gravimetric method and digitized reading provided by the developed moisture sensor. The developedempirical equation will be implemented in Arduino to provide direct soil moisture reading in percentage.

Step 4: In this step, we will validate the developed soil moisture measuring device. We will determine the root mean square error (RMSE) value. This helps in detecting the errors taking place in the calibration of soil moisture sensor. If the RMSE is low, it means that the device is accurate and reliable. Therefore, it can be used for observing the soil moisture value with longitude and latitude.

CHAPTER 5

SOFTWARE IMPLEMENTATION ON HARDWARE

5.1 Introduction

Software is a combination of various programs, a program is a combination of different instructions and hardware needs instructions in order to operate. Thus, we design a program in software to operate the hardware according to our needs. The program is embedded into the microcontroller which is responsible for the functionality of the peripherals such as (Sensors, GPS, LCD LCD etc.). In this chapter we will come across the software that we have used during the span of the completion of project.

5.2 Arduino IDE 1.8.9

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.
The source code for the IDE is released under the GNU General Public License, version 2. The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.

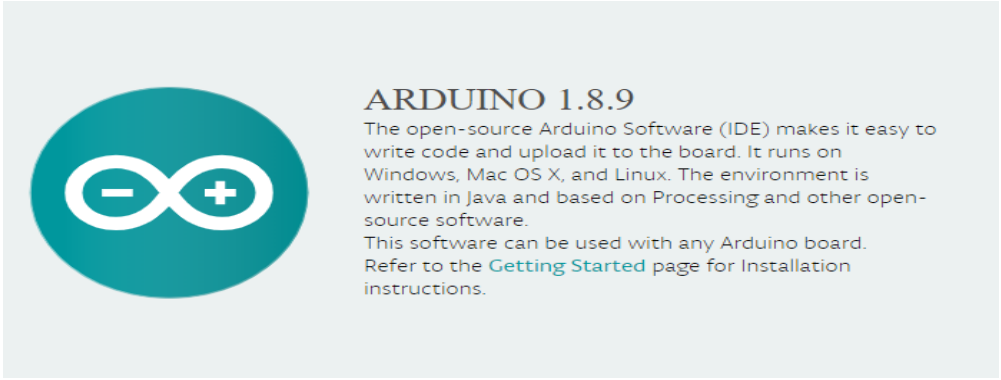

Figure 5.1. Arduino IDE software

Figure 5.2. Arduino IDE console

5.3 Code Analysis of the project

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <SD.h>
#include "RTClib.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

int RXPin = 4;
int TXPin = 3;

int GPSBaud = 9600;

int sensor_pin = A0;

int var;

int output_value=0 ;
```

```
// Create a TinyGPS++ object
TinyGPSPlus gps;


// Create a software serial port called "gpsSerial"

SoftwareSerial gpsSerial(RXPin, TXPin);

File myFile;

RTC_DS1307 RTC;

LiquidCrystal_I2C lcd(0x27, 16, 2);

// Define constants

#define menuButton 2
#define menuSelect 5
#define menuSave 6
#define debounceTimeout 50

int menuButtonPreviousState = LOW;
int menuSelectPreviousState = LOW;
int menuSavePreviousState = LOW;

// Define variables

long int lastDebounceTime;
bool urban = true;
bool waterbodies = true;
bool rural = true;
bool barrenland = true;

// Menu options
char * menuOptions[] = {"urban", "water bodies","rural","barren land"};
bool featureSetting[] = {false,false,false,false};
bool menuMode = false;
bool menuNeedsPrint = false;

int optionSelected = 0;
```

```
// Setup function

void setup()
{
 lcd.backlight();
  RTC.begin();
  Wire.begin();
  lcd.init();

Serial.begin(9600); // connect serial

 gpsSerial.begin(GPSBaud);
Serial.print("Initializing SD card...");

  if (!SD.begin(10)) {

    Serial.println("initialization failed!");
    return;
  }


  lcd.setCursor(0,0);
  lcd.print("SD card begin");
  Serial.println("SD card begin");
   delay(5000);


 if (! RTC.isrunning()) {

    lcd.println("RTC is NOT running!");
    RTC.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }


    DateTime now = RTC.now();

    lcd.setCursor(0,0);


    lcd.print(now.year(), DEC);
    lcd.print('/');
```

```
  lcd.print(now.month(), DEC);
  lcd.print('/');

  lcd.print(now.day(), DEC);

   lcd.print(' ');    lcd.print(now.hour(), DEC);
   lcd.print(':');

   lcd.print(now.minute(), DEC);


   Serial.print("Latitude: ");

   lcd.setCursor(0,1);

  lcd.print(gps.location.lat(), 6);

   Serial.println(gps.location.lat(), 6);

   lcd.print(',');

   Serial.print("Longitude: ");
   lcd.print(gps.location.lng(), 6);

  Serial.println(gps.location.lng(), 6);
   delay(5000);


 pinMode(menuSelect, INPUT);
 pinMode(menuSave, INPUT);
 pinMode(menuSelect, INPUT);
 Serial.begin(9600);
}


// Function to return the current selected option

 char *ReturnOptionSelected(){
 char *menuOption = menuOptions[optionSelected];


// Return optionSelected
 return menuOption;
}
```

```
// Function to return status of current selected option

  char *ReturnOptionStatus(){
  bool optionSetting = featureSetting[optionSelected];
  char *optionSettingVal;

if (optionSetting == false){
    optionSettingVal = "False";
  }else{
    optionSettingVal = "True";
  }

  // Return optionSetting

  return optionSettingVal;
}

// Function to toggle current option

bool ToggleOptionSelected(){
  featureSetting[optionSelected] = !featureSetting[optionSelected];
  return true;
}


 void displayInfo()
{
  if (gps.location.isValid())
  {
    Serial.print("Latitude: ");
    Serial.println(gps.location.lat(), 6);
    Serial.print("Longitude: ");
    Serial.println(gps.location.lng(), 6);
    }
  else
  {
    Serial.println("Location: Not Available");
  }


  Serial.println();
  Serial.println();

}
```

```
// The main loop

   void loop(){


  // Read the buttons

  int menuButtonPressed = digitalRead(menuButton);
    int menuSelectPressed = digitalRead(menuSelect);
    int menuSavePressed = digitalRead(menuSave);

  // Get the current time
  long int currentTime = millis();

  if(menuButtonPressed == LOW && menuSelectPressed == LOW &&
menuSavePressed == LOW){

    //Reset the count time while button is not pressed

    lastDebounceTime = currentTime;
    menuButtonPreviousState = LOW;
    menuSelectPreviousState = LOW;
    menuSavePreviousState = LOW;
  }

  if(((currentTime - lastDebounceTime) > debounceTimeout)){

    // If the timeout is reached, button pressed!</p><p>    // menuButton is pressed,
provide logic

    // Only fires when the button has previously been released
    if((menuButtonPressed == HIGH) && (menuButtonPreviousState == LOW)){

      if(menuMode == false){
        menuMode = true;

        // Let the user know

        lcd.clear();
        lcd.setCursor(0,0);
        Serial.println("Menu is active");
        lcd.print("Menu is active");
      }else if (menuMode == true && optionSelected < 4){
```

```
// Change option if menu is active
      optionSelected = optionSelected + 1;
    }
    else if (menuMode == true && optionSelected >= 4){

// Reset option
      optionSelected = 0;
    }

    // Print the menu
    menuNeedsPrint = true;

 // Toggle the button prev. state to only display menu
    // if the button is released and pressed again

    menuButtonPreviousState = menuButtonPressed; // Would be HIGH
  }

 // menuSelect is pressed, provide logic

  if((menuSelectPressed == HIGH) && (menuSelectPreviousState == LOW)){

    if(menuMode){
      // Change the selected option
      // At the moment, this is just true/false
      // but could be anything

      bool toggle = ToggleOptionSelected();
         if(toggle){
      menuNeedsPrint = true;
      }
     else
     {
       Serial.print("Something went wrong. Please try again");
      }
    }

// Toggle state to only toggle if released and pressed again
    menuSelectPreviousState = menuSelectPressed;
  }
  if((menuSavePressed == HIGH) && (menuSavePreviousState == LOW)){

    // Exit the menu
    // Here you could do any tidying up
```

```
  // or save to EEPROM
  menuMode = false;
  Serial.println("Menu exited");
  lcd.clear();
  lcd.print("Menu exited");

  // Toggle state so menu only exits once

   menuSavePreviousState = menuSavePressed;
  }
 }

// Print the current menu option active, but only print it once

  if(menuMode && menuNeedsPrint){
  // We have printed the menu, so unless something
  // happens, no need to print it again

    menuNeedsPrint = false;
   char *optionActive = ReturnOptionSelected();
   char *optionStatus = ReturnOptionStatus();
   Serial.print("Selected: ");
   Serial.print(optionActive);
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print(optionActive);
   Serial.print(": ");
   Serial.print(optionStatus);
   Serial.println();

   if(menuSelectPressed == HIGH){
  output_value=0;

   while (gpsSerial.available() > 0)

   if (gps.encode(gpsSerial.read()))
  displayInfo();

  // If 5000 milliseconds pass and there are no characters coming in
  // over the software serial port, show a "No GPS detected" error

   if (millis() > 5000 && gps.charsProcessed() < 10)
   {
     lcd.println("No GPS detected");
```

```
 Serial.println("No GPS detected");
}

 myFile = SD.open("test.txt", FILE_WRITE);
lcd.clear();
DateTime now = RTC.now();


 Serial.print(" ");

  myFile.print("DATE:");
  myFile.print(now.year(), DEC);
  myFile.print('/');
  myFile.print(now.month(), DEC);
  myFile.print('/');
  myFile.print(now.day(), DEC);
  myFile.println(' ');
  myFile.print("TIME:");
  myFile.print(now.hour(), DEC);
  myFile.print(':');
  myFile.print(now.minute(), DEC);
  myFile.print(':');
  myFile.print(now.second(), DEC);
  myFile.println(' ');

   lcd.setCursor(0,0);

  Serial.print("Latitude: ");
  Serial.println(gps.location.lat(), 6);

   lcd.print(gps.location.lat(), 6);
   myFile.print("Latitude: ");
    myFile.println(gps.location.lat(), 6);

  lcd.print(",");
             Serial.print("Longitude:");
            Serial.println(gps.location.lng(), 6);
             lcd.println(gps.location.lng(), 6);
             myFile.print("Longitude: ");
             myFile.println(gps.location.lng(), 6);

           for(int i=1;i<=10;i++)
           {
```

```
            var=analogRead(sensor_pin);
            output_value=output_value+var;
 }
output_value=output_value/10;

Serial.print("Moisture : ");
          Serial.println(output_value);
           lcd.setCursor(0,1);
           lcd.print(output_value);

            myFile.print("Moisture : ");
            myFile.println(output_value);

           myFile.close();
          }
     }
 }
```

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Calibration of sensor

In measurement technology and metrology, calibration is the comparison of measurement values delivered by a device under test with those of a calibration standard of known accuracy. Such a standard could be another measurement device of known accuracy, a device generating the quantity to be measured such as a voltage, a sound tone, or a physical artifact, such as a meter ruler.

The outcome of the comparison can result in one of the following:

- no significant error being noted on the device under test
- a significant error being noted but no adjustment made
- an adjustment made to correct the error to an acceptable level

The term "calibration" means just the act of comparison and does not include any subsequent adjustment. The calibration standard is normally traceable to a national standard held by a national metrological body.

Calibration is a need to form any device with proper accuracy, reliability and stability. It is essential for a device to give accurate readings to draw out conclusions based on that. Errors should be minimized for a proper device working. It gives a guarantee that the readings are accurate at any geographical condition. Calibration makes the device consistent when used by different operators under various environmental conditions. So, to make the device stable and steady we should calibrate it. The definition of calibration includes the word “documented.” This means that the calibration comparison must be recorded. Calibration is essential as measuring devices are installed in the field in different environments and industries. It means they face various challenges, such as abrasion, vibration, and abrupt changes of temperature, harsh environments and much more. These environments will affect the measuring device, making it necessary to calibrate in order to verify the accuracy and, if necessary, adjust the field instrument to measure with the accuracy required for the application.

On top of that, when you have a device working correctly, it will positively affect working process because all the measurements will be correct. Lastly, calibration brings positive advantages, such as the reduction of variation with respect to the technical specification as well as preventive maintenance. It also ensures the traceability of the measurements.

Therefore, after interfacing all the devices we calibrated our developed soil moisture device under various climatic and environmental conditions. First of all, we have calibrated the developed soil moisture sensor under pure water condition. We kept the probes of the soil moisture sensor in pure water for a certain time and observed the reading which is shown in Figure 6.1 . The developed soil moisture sensor converts the voltage value across the probe into bits. It is a 10 bits sensor therefore, it provides the reading ranging from 0-1024. This range helps us in finding the percentage of moisture in soil. We have taken many observations by dipping the probes in water to observe the consistency of the range of the sensor in pure water.

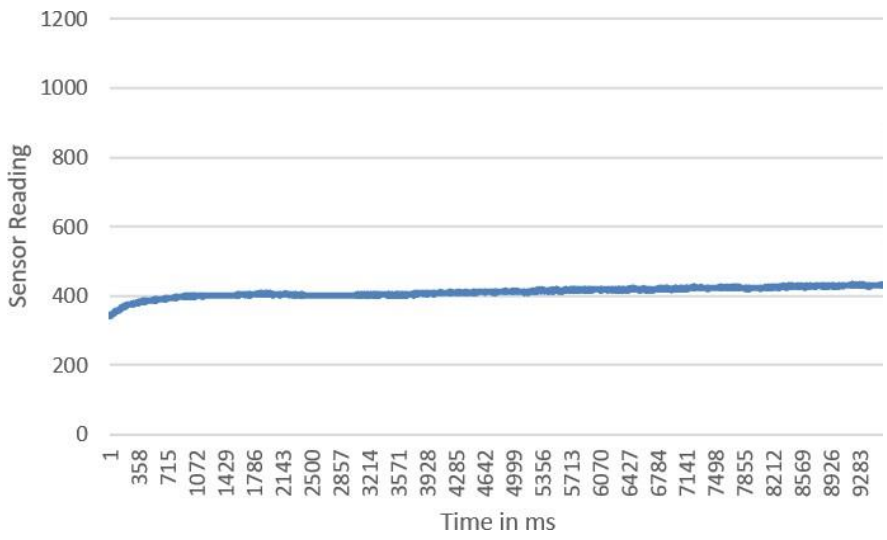

Figure 6.1. Graph between Sensor Reading and Time during which the probe is in water

From the Fig 6.1, we may observe that the soil sensor reading becomes constant after approximately 580ms. Therefore, in our developed device, we have provided a delay of 500 ms to have accurate reading of soil moisture.

The next part of calibration takes place by testing our soil sensor with different soil samples. We took 16 soil samples having different soil moisture and labeled them as sample 1 to 16 as shown in Figure 6.2.

Figure 6.2 Soil samples taken for calibration

To determine the relationship between digitized readings of the soil moisture device and actual gravimetric soil moisture, we evaluated the soil moisture of all the 16 samples with gravimetric method. In gravimetric method the percentage soil moisture is evaluated with the help of Equation (1)

$$m_g = ((W_{wet} - W_{dry}) \div W_{dry}) \times 100\% \quad (1)$$

Where, m_g is the percentage of gravimetric soil moisture content. The moisture from the soil has been removed by keeping the soil sample in microwave oven for 20 minutes at 800 W. W_{wet} is the weight of the soil with moisture content. W_{dry} is the weight of soil without any moisture content. Table I shows the soil moisture measured with the developed sensor. Figure 6.3 shows the graph between readings of soil moisture measuring device and gravimetric soil moisture. Linear Regression method was carried out to determine the relationship between digitized reading and gravimetric soil moisture content. A linear relationship was obtained with coefficient of determination R^2 = 0.83. Equation (2) shows the empirical relationship between the gravimetric soil moisture and the average range of the soil sensor reading.

$$y = -26\, m_g + 1000 \quad (2)$$

Where, y is the average range of soil sensor reading and m_g is the percentage of gravimetric soil moisture content. Finally, Equation (2) has been implemented in Arduino to provide gravimetric soil moisture content.

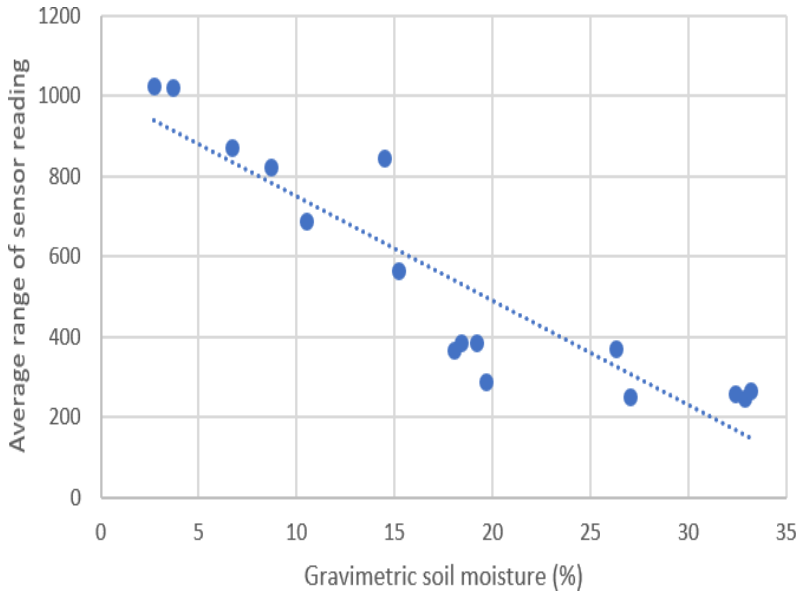

Figure 6.3 Plot between the Range of sensor reading and gravimetric soil moisture.

Table 6.1 Calibration of different soil samples

Sample	Weight (dry soil)gm	Water in ml	Weight(wet) (a)	Weight(after 2 days) (b)	((a-b)/a)*100 %	Sensor range(d)	Oven dry weight (c)	((b-c)/b)*100%	Average(d)
1	263.0	10	272.9	263.7	3.38	1022-1022	256.5	2.72	1022
2	262.2	20	281.1	264.4	5.59	1019-1019	255.5	3.72	1019
3	261.3	30	194.1	273.5	6.99	872-869	255.2	6.72	869.5
4	261.5	40	300.1	279	6.97	859-781	254.8	8.73	820
5	262.8	50	310.2	284.8	9.5	739-640	254.9	10.51	684.4
6	262.5	60	324.0	297.6	8.13	848-841	254.4	14.54	844.5
7	261.4	70	330.0	303.9	7.92	581-545	257.1	15.22	563
8	263.5	80	346.1	317.4	8.27	381-383	259	18.41	382
9	263.9	90	359.9	330.3	8.2	352-382	270.7	18.06	366
10	262.7	100	367.5	338.7	7.83	375-396	273.6	19.23	385.5
11	264.4	110	371.7	341.7	8.04	283-289	274.5	19.68	286
12	260.7	120	389.5	358.1	8.05	371-365	263.9	26.32	368
13	262.9	130	389.2	364.9	8.34	230-270	266.2	27.05	250
14	261.1	140	406.3	372.2	8.39	220-269	249.9	32.87	244.5
15	262.1	150	415.5	383.4	7.73	252-265	259.1	32.48	258.5
16	261.9	160	424.3	389.2	8.28	260-267	259.9	33.22	263.5

6.2 Validation of the sensor

Validation is a testing process by which you prove ("validate") that the device you've built works for the end user as intended. Or we can say validation is "establishing by objective evidence that device specifications conform to user needs and intended use". Validation is the assurance that a product, service, or system meets the needs of the device working. It often involves acceptance and suitability.

We have found out the validity of the soil moisture sensor by testing it further in various different types of soil of varied moisture content. A comparison is done between the gravimetric soil moisture measured with developed device and gravimetric method. If the graph is approximately linear, then we get approved of the reliable working of our device. Figure 6.4 shows a plot between observed value and measured value. Further, we find out the Root mean square error (RMSE) value with the help of equation (3).

$$RMSE = \sqrt{\sum_{i=1}^{n} (^{(M_i - O_i)^2}/_{n})} \quad (3)$$

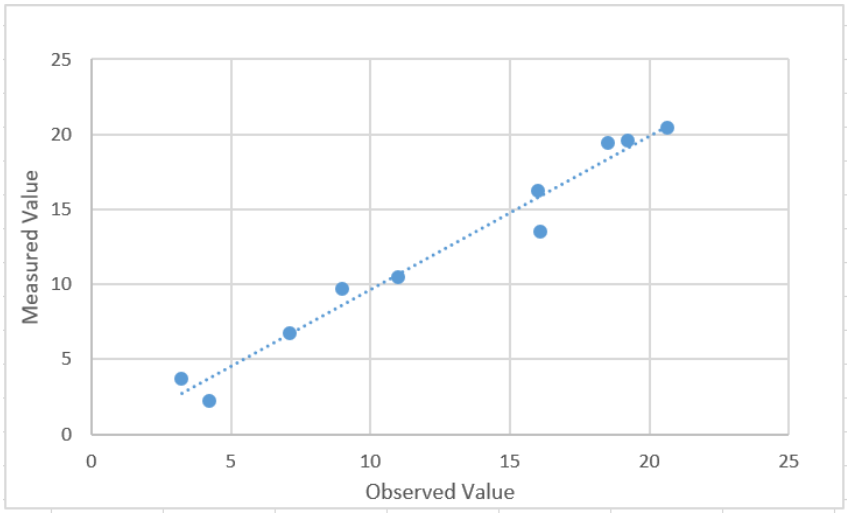

Figure 6.4. Graph of Validation of soil moisture sensor

We observed that the Measured and Observed value are almost close to the linear line, which states that the device has less RMSE error. It is up to 1.52. Hence, the calibration and validation of the device is done successfully.

6.3 Performance Analysis

The soil moisture sensor is calibrated and validated successfully. It has least error values due to which it is efficient and reliable in providing accurate data. It has no effect in the performance due to any geographical factors. The device ensures the display of soil moisture in terms of percentage. It also mentions the type of land we are present in with the location inculcating longitude and latitude. This portable device is a boon to agriculture and other irrigational systems.

CHAPTER 7

CONCLUSION

The soil moisture sensor is developed and interfaced with various components. It is calibrated to give accurate and reliable values of moistures under varied geographical conditions. We have interfaced it with micro SD card logger and push buttons to get all the data stored in SD card. GPS gives the location and the type of area we are present along with its latitude and longitude. The device is especially useful in agricultural field to determine the moisture content of the soil. It is more efficient to use this sensor as it is inexpensive and reliable than other sensors. The working model of the device is shown in Figure 7.1.

Figure 7.1 Working model of the developed soil moisture sensor

7.1 Outcomes of this project

1. Skills and self-confidence in coding and working with software like Arduino IDE were developed.
2. Skills to solder devices and design a well performing circuit got developed.
3. The device developed has been designed for many applications and not limited to measuring the soil moisture measurement.
4. This device is a necessity for small crop farmers and other people who cannot afford expensive sensors like TDR and FDR soil sensors. As, this device is error free and inexpensive.

CHAPTER 8

FUTURE WORK

In the future use or work I would like to extend this project to a cloud platform where we can store the data on a cloud system and get all the data on our phones. This will insure more vigilance and security towards the device functioning.

8.1 Approaches for future work

1. We need Arduino MKR 1000 Wi-Fi module which is a powerful board that helps in adding Wi-Fi connectivity to the devices on Arduino IoT cloud platform.
2. Then the linking of data from device to an IoT cloud platform is needed which will be further handled from a cell phone.
3. We need to develop an app which will help us monitor the soil moisture sensor device.
4. This would make the device not only portable but it will help us save the resources more efficiently.
5. Further, we need to design it using ORCAD software to make a compact device which can be further used in a more professional form.

PUBLICATION

The following report has been published as a thesis in the Springer Journal.
Following is the Topic and Conference name:

TOPIC: IoT based System to measure Soil Moisture sensor, GPS data logging and Cloud storage.
CONFERENCE NAME: International conference on Innovative Computing and Communication (ICICC-2020)

REFERENCES

[1] C. Pathe, W. Wagner, D. Sabel, M. Doubkova, and J. B. Basara, “Using ENVISAT ASAR global mode data for surface soil moisture retrieval over Oklahoma, USA,” IEEE Trans. Geosci. Remote Sens., vol. 47, no.2, pp. 468–480, 2009.

[2] N. Pierdicca, L. Pulvirenti, and C. Bignami, “Soil moisture estimation over vegetated terrains using multitemporal remote sensing data,” Re-mote Sens. Environ., vol. 114, pp. 440– 448, 2010.

[3] J. D. Bolten, W. T. Crow, Z. Xiwu, T. J. Jackson, and C. A. Reynolds, “Evaluating the utility of remotely sensed soil moisture retrievals for operational agricultural drought monitoring,” IEEE J. Sel. Topics Appl.Earth Observ. Remote Sens. (JSTARS), vol. 3, no. 1, pp. 57–66, 2010.

[4] Wen, M.M., Liu, G., Horton, R. and Noborio, K., “An in-situ probe-spacing-correction thermo-TDR sensor to measure soil water content accurately”, European journal of soil science, 69(6), pp.1030-1034, 2018.

[5] Oates, M.J., Ramadan, K., Molina-Martínez, J.M. and Ruiz- Canales, A., “Automatic fault detection in a low-cost frequency domain (capacitance based) soil moisture sensor.” Agricultural water management, 183, pp.41-48, 2017.

[6] Alejandro Egido, Marco Caparrini, Giulio Ruffini, SimonettaPaloscia, Emanuele Santi, Leila Guerriero, NazzarenoPierdicca andNicolas Floury, “Global Navigation Satellite Systems Reflectometryas a Remote Sensing Tool for Agriculture", Remote Sensing, vol.4 , pp. 2356-2372, 2012.

[7] K. M. Larson, Eric E. Small, Ethan Gutmann, Andria Bilich, Penina Axelrad and John Braun, "Using GPS Multipath to Measure Soil Moisture Fluctuations: Initial Results", GPS Solutions, vol.12, Issue 3, pp 173–177, 2008.

[8] Valery U. Zavorotny, Kristine M. Larson, John J. Braun, Eric E.Small, Ethan D. Gutmann, and Andria L.Bilich, "A Physical Model for GPS Multipath Caused by Land Reflections: Toward Bare Soil Moisture Retrievals", IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 3, pp 100 – 110, 2010.

[9] Singh, P. and Saikia, S., December, “Arduino-based smart irrigation using water flow sensor, soil moisture sensor, temperature sensor and ESP8266 WiFi modul”. In 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), pp. 1-4, 2016.