“IoT Based Indoor Garden System”

A report submitted to the Department of EEE, Sonargaon University of Bangladesh in partial fulfillment of the requirements for the Award of Degree of Bachelor of Science in Electrical & Electronics Engineering.

<u>Submitted by</u>

➢ Md. Rasel	ID: EEE1803014001
➢ Md. Palash Hosen	ID: EEE1501004081
➢ Md. Liton Sha	ID: EEE1701010145
➢ Md. Ekramul Islam	ID: EEE1801013171
➢ Lutfor Rahman	ID: EEE1603009098
➢ Zahanara Akter	ID: EEE1802014080

<u>Supervised by</u>

Nurul Ambia Alaul (Lecturer)

Department of Electrical & Electronics Engineering

Sonargaon University (SU)

Dhaka-1215, Bangladesh

January, 2022

Declaration

We declare that this project work entitled “**IoT Based Indoor Garden System**” is the result of our own work as cited in the references. This project has not been accepted for any degree and is not concurrently submitted in candidature for any other degree or diploma elsewhere.

Md. Rasel ID: EEE1802014001	Md. Ekramul Islam ID: EEE1801013171
Md. Palash Hosen ID: EEE1501004081	Lutfor Rahman ID: EEE1603009098
Md. Liton Sha ID: EEE1701010145	Zahanara Aktar ID: EEE1802014080

I hereby declare that I have read the project report thoroughly. In my opinion, this project is sufficient in terms of scope and quality to meet the partial requirements for the award of the B.Sc. Engineering degree in EEE.

Supervised by,

Nurul Ambia Alaul
Lecturer Dept. of EEE, SU

Approval

The Senior Project entitled **“IoT Based Indoor Garden System”** carried out by

Md. Rasel, ID: EEE1802014001,

Md. Palash Hosen, ID: EEE1501004081,

Md. Liton Sha, ID: EEE1701010145,

Md. Ekramul Islam, ID: EEE1801013171,

Lutfor Rahman, ID: EEE1603009098,

Zahanara Akter, ID: EEE1802014080

For fulfillment of the requirement for the award of B.Sc. Engineer Degree in ME was presented to the audience of the Oral Exam Committee on 10/01/2022 and has been accepted as satisfactory.

Prof.Dr.M Bashir Uddin
Professor & Department Head of EEE,SU

Nurul Ambia Alaul
Lecturer Dept. of EEE, SU

Md Ferdous Khan
Coordinator Dept. of EEE,SU

Md. Rais Uddin Mollah
Assistant Coordinator Dept. of EEE,SU

Acknowledgement

Our utmost gratitude to Allah, the almighty, without his mercy and blessing this work would not been possible. We are grateful to our supervisor **Nurul Ambia Alaul**, Lecturer, Department of Electrical & Electronics Engineering, SU for his guidance, encouragement many ways throughout his project work. We also appreciate his vision, experience, interest and support for this project which came to us with greater help to write this paper.

We are also thankful to **Teacher's Name**, Dept. of EEE, SU for his valuable suggestions. We are also thankful to our classmates for their support and also who helped us to make this project successful. Finally, we convey our heartily gratitude to our parents and family members for their moral support.

Abstract

Technology brings a remarkable advancement in every field of life, whether it is industry or agriculture. Our lives are essentially dependent on agricultural development. Researchers are working to integrate modern technologies in agriculture to develop new practices for the enhancement of healthy agriculture and production. Internet of things is a domain of computer science that provides mechanisms and techniques to interconnect a wide range of digital devices to automate the real-life systems. In big cities, peoples facing problems in their homegrown gardens regarding the maintenance and availability of proper gardeners. This research paper has proposed an IoT based approach for smart garden monitoring using Node MCU microcontroller that helps the users in identifying current parameters of temperature, moisture, and humidity of their homegrown plants and gardens. A prototype has been implemented to show the real illustration of the proposed approach. An android mobile application has been developed to display the real-time profiles of environmental factors like temperature, moisture, and humidity. With the help of this system, users will be able to treat their gardens in a better way in terms of plant health and growth. All data will be stored in a data logger (**Adafruit**) system. Downloading data is also possible. This research work replaces the need for gardeners and issues faced during the maintenance of gardens in big cities. The purpose of this research is to introduce and prosper the IoT innovation towards smart cities in our society.

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

CHAPTER 1
INTRODUCTION

1.1 Background

In the current technological era, automation rules all over the world and holds the key to radically empowering several sectors of Bangladesh economy. From manufacturing, agriculture to services and logistics, technology can enhance the capacities, efficiencies, and production quality of every human activity. Internet of things (IoT) is a technique of using computers, mobile phones, or digital devices in monitoring and controlling the simple parameters of day to day life . Using IoT concepts and knowledge, new systems will be developed based upon sensors, software, and communication protocols for automation of specific tasks. Data exchange is the key factor for IoT. The standards of our lives will be nourished by the practice of using automation for simple things.

We are living in the fourth industrial revolution in which systems are moving from manual to automatic process. It brings the concept of the smart industry and opens many research direction. Those Peoples who prefer to grow gardens and other small fruit plants in their homes often get cautious during the early stages of maintenance. As a result, the garden gets destroyed due to a lack of care. There are also weather conditions that can render the gardens lifetime short, as some crops/plants can die due to lack of moisture, severe heat, and humidity, etc. People hire the gardeners to maintain their homegrown small gardens. Most plants often get damaged due to environmental conditions and lack of proper care.

This is where the idea of an automated garden monitoring system takes place with the internet of things to overcome above- mentioned problems. The proposed system integrates all sensors and components for real-time statistics. Communication is done with wireless sensor networks. Mobile computing is an efficient technology to support the internet of things for developing real-life systems. Already, there are mobile applications that help farmers in their crop maintenance. Similar IoT based systems are designed for garden maintenance that is costly and often used one task, only temperature reading or water pouring mechanism. Our country is based upon the agriculture sector, and it needs to achieve higher benefits from other countries. A mobile application is developed to check the status of the garden for watering. This system senses the temperature and moisture with temperature and moisture sensors for big gardens. Soil and other supporting sensors are integrated over Node MCU that gathers the values of soil and transfer the information to firebase by using built-in WIFI facility. The temperature sensor, soil moisture and pump controller are integrated to develop the prototype. The system senses the moisture and humidity level of plants and provides water accordingly. All kind of this project data will store in Adafruit server for help further check condition of land.

1.2 Literature Review

Many works have been done related to smart apps to monitor plant growth. Prathibha et. al [1] used IoT to introduce smart farm monitoring program. This system consists of a microcontroller, Wi-Fi unit, network processor. By using different sensors, it tracks crop field condition. Besides that, this system monitors the soil temperature using a thermopile sensor (TMP007).

Furthermore, this system uses HDC1010 as humidity sensor to track the relative moisture of air within the farming field. This wireless monitoring of crop field is able to reduce human power, and it allows users to see the real changes in crop yield. This system is also can be applied to dependent plant which is located in the greenhouse.[2] Sehrish Munawar Cheema et.al [3] proposed a recommendation system for plant irrigation using IoT based digital solution for a home garden. This system obtains real data value from the sensors in a way to make a recommendation for irrigation scheduling, and also which plants should be grown. The microcontroller will get the data from a server and match the predefined rules to generate the plant feasibility rating. Users receive notifications through the mobile phone about the analysis, and this information is needed for them to schedule the time for watering the plants.

Pavithra et al. [4] demonstrated their automated irrigation control system using the Global System for Mobile Communication (GSM) module and microcontroller that connected with Universal Asynchronous Receiver/Transmitter (UART). The system sets the time of irrigation that depends on the temperature and humidity a sensor has obtained, as well as the types of crops. The system can irrigate the field automatically without human presence. [5] Some features in their system support water management decision which determines the time control for monitoring the tank water level and provides the exact amount of water required for plant and crop. Other than that, their system can check the temperature and humidity of soil.

Beside that, other researchers conducted by Kansara et al. [6] stated about using GSM module, sensor and microcontroller are connected using MAX232. This system uses General Packet Radio Services (GPRS) feature of a mobile phone as for a solution to control the irrigation system. The irrigation system sets as automated by using the controllers where the valve will turn ON and OFF. This way will help to apply the right amount of water and the right time that helps to improve crop performance by ensuring adequate water. This system supports some features such as support water management decision, monitors the water level in the tank and provide an accurate amount of water required and checks the soil temperature and soil humidity of the plant. However, in this project, small scale system has been developed where it helps the gardeners to monitor their plants using mobile app.

1.3 Objective

The objective of this work is:

- Design & Construction of "**IoT Based Indoor Garden system**"
- Learn the Implementation of Indoor Garden Temperature Monitoring, garden water condition in soil, sensing humidity, automatic pumping in garden System.
- Implementation of all data will record in a data logger system.

1.4 Methodology

Our used methodology for the project:

- ➢ Creating an idea for design and construction of “IoT Based Indoor Garden System”. And designing a block diagram & circuit diagram to know which components we need to construct it.
- ➢ Collecting all the components and programming the microcontroller to control the system.
- ➢ Setting up all the components in a PCB board & then soldering. Then assembling all the blocks in a board and finally running the system & checking.

1.5 Advantages

There are certainly many advantages of our project and some of the major ones have been given below:

- All data will be found in a data logger (Adafruit).
- The project is compact, cheap and user friendly.
- The whole system consumes very little energy.
- Our system is fully automatic.
- The project is temperature sense, soil moisture sense and take immediate precaution system.
- The system can be implemented anywhere with very little effort.
- Requires low maintenance.
- Without human interface the condition of garden and motor will be operate from far way from garden.

1.6 Applications

Our project has many application areas and actually we need to use it in many places to verified the soil situation, air moisture and it will be works from far way with a device. Some of the application areas of the project has been pointed out below:

- The system can be implemented in indoor garden.
- It can be implemented in rooftop garden.

1.7 Limitations

We are thinking about adding many advantages to our project but this system has some limitations. Some of the limitations are given below:

- There is no extra electricity back up in this project.
- IoT Signal may late cause it’s a demo project.

1.8 Future Scope of Work

We are thinking about adding many features to our project in the future to get more desirable outcomes. Some of the steps that we are thinking about taking are given below:

- In future, we are thinking about adding camera monitoring features to the system.
- In future, we are thinking about adding backup power system to project so that can run even when there’s no supply voltage available.

1.9 Structure of the Project

This project book consists of five chapter. The first chapter contains the statement of the introduction, our background study for the project, objectives of the study in the project and the project organization. Chapter two contains design of this project, block diagram and circuit diagram, working principle. Chapter three describes the background and real project, details of component and instrument details of the whole project. Chapter four deals with the result and discussion and shows the complete prototype of the project that we have built. In the final chapter, we discuss about future scope and conclusion of our project.

CHAPTER 2
HARDWARE ANALYSIS

2.1 Introduction

This Project has worked on two things, Hardware and Software. In this Chapter we will discuss about instrument specification, application and working procedure.

Software

- Proteus 8.9
- Arduino IDE
- Adafruit

Hardware

- SMPS
- Soil Moisture Sensor
- Temperature Sensor
- Node MCU
- Relay
- Mini Pump
- Fan

Hardware Description

2.2 Switch Mode Power Supply (SMPS)

A switched-mode power supply (switching-mode power supply, switch-mode power supply, switched power supply, SMPS, or switcher) is an electronic power supply that incorporates a switching regulator to convert electrical power efficiently Unlike a linear power supply, the pass transistor of a switching-mode supply continually switches between low-dissipation, full-on and full-off states, and spends very little time in the high dissipation transitions, which minimizes wasted energy. A hypothetical ideal switched-mode power supply dissipates no power. Voltage regulation is achieved by varying the ratio of on-to-off time (also known as duty cycles). In contrast, a linear power supply regulates the output voltage by continually dissipating power in the pass transistor. This higher power conversion efficiency is an important advantage of a switched-mode power supply.

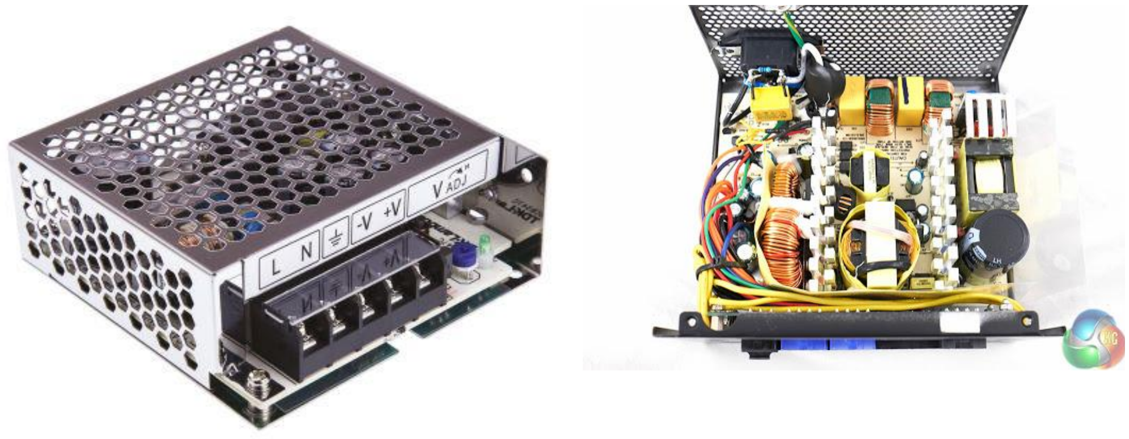


Figure 2.1: Switch Mood Power Supply (SMPS)

Switching regulators are used as replacements for linear regulators when higher efficiency, smaller size or lighter weight are required. They are, however, more complicated; their switching currents can cause electrical noise problems if not carefully suppressed, and simple designs may have a poor power factor. Switched-mode power supplies are classified according to the type of input and output voltages. The four major categories are:

- AC to DC
- DC to DC
- DC to AC
- AC to AC

A basic isolated AC to DC switched-mode power supply consists of:

- Input rectifier and filter
- Inverter consisting of switching devices such as MOSFETs
- Transformer
- Output rectifier and filter
- Feedback and control circuit

The input DC supply from a rectifier or battery is fed to the inverter where it is turned on and off at high frequencies of between 20 KHz and 200 KHz by the switching MOSFET or power transistors. The high-frequency voltage pulses from the inverter are fed to the transformer primary winding, and the secondary AC output is rectified and smoothed to produce the required DC voltages. A feedback circuit monitors the output voltage and instructs the control circuit to adjust the duty cycle to maintain the output at the desired level.

Basic working concept of an SMPS

A switching regulator does the regulation in the SMPS. A series switching element turns the current supply to a smoothing capacitor on and off. The voltage on the capacitor controls the time the series element is turned. The continuous switching of the capacitor maintains the voltage at the required level.

Design basics

AC power first passes through fuses and a line filter. Then it is rectified by a full-wave bridge rectifier. The rectified voltage is next applied to the power factor correction (PFC) pre-regulator followed by the downstream DC-DC converter(s). Most computers and small appliances use the International Electro technical Commission (IEC) style input connector. As for output connectors and pin outs, except for some industries, such as PC and compact PCI, in general, they are not standardized and are left up to the manufacturer.

There are different circuit configurations known as topologies, each having unique characteristics, advantages and modes of operation, which determines how the input power is transferred to the output. Most of the commonly used topologies such as fly back, push-pull, half bridge and full bridge, consist of a transformer to provide isolation, voltage scaling, and multiple output voltages. The non-isolated configurations do not have a transformer and the power conversion is provided by the inductive energy transfer.

Advantages of switched-mode power supplies:

- Higher efficiency of 68% to 90%
- Regulated and reliable outputs regardless of variations in input supply voltage
- Small size and lighter
- Flexible technology
- High power density

Disadvantages:

- Generates electromagnetic interference
- Complex circuit design
- Expensive compared to linear supplies

Switched-mode power supplies are used to power a wide variety of equipment such as computers, sensitive electronics, battery-operated devices and other equipment requiring high efficiency.

Switch Mode Power Supply

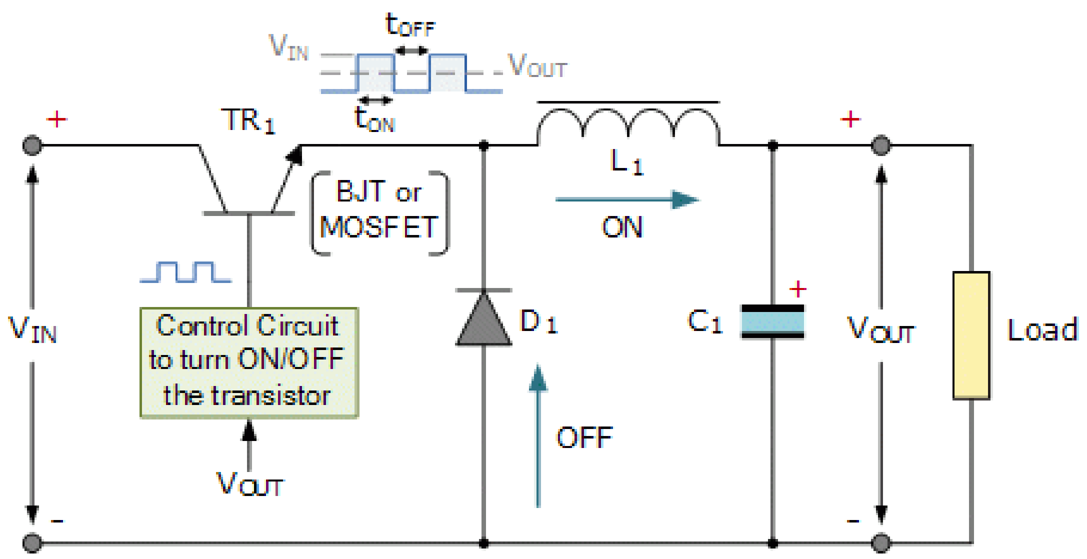


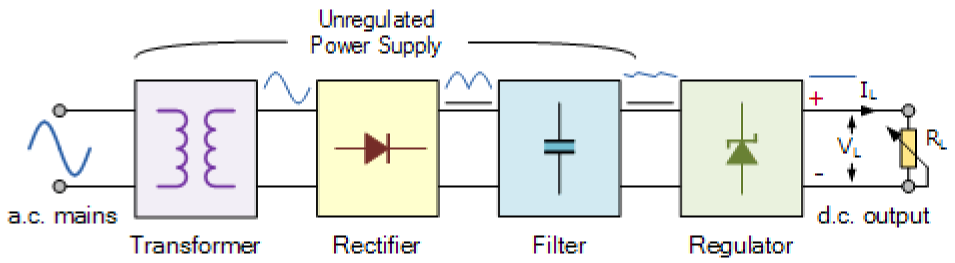
Figure 2.2: SMPS Circuit Diagram

Linear voltage IC regulators have been the basis of power supply designs for many years as they are very good at supplying a continuous fixed voltage output. Linear voltage regulators are generally much more efficient and easier to use than equivalent voltage regulator circuits made from discrete components such a Zener diode and a resistor, or transistors and even op-amps.

The most popular linear and fixed output voltage regulator types are by far the positive output voltage series, and the negative output voltage series. These two types of complementary voltage regulators produce a precise and stable voltage output ranging from about 5 volts up to about 24 volts for use in many electronic circuits. There is a wide range of these three-terminal fixed voltage regulators available each with its own built-in voltage regulation and current limiting circuits. This allows us to create a whole host of different power supply rails and outputs, either single or dual supply, suitable for most electronic circuits and applications. There are even variable voltage linear regulators available as well providing an output voltage which is continually variable from just above zero to a few volts below its maximum voltage output.

Most D.C power supplies comprise of a large and heavy step-down mains transformer, diode rectification, either full-wave or half-wave, a filter circuit to remove any ripple content from the rectified D.C producing a suitably smooth D.C voltage, and some form of voltage regulator or stabilizer circuit, either linear or switching to ensure the correct regulation of the power supplies output voltage under varying load conditions. Then a typical D.C power supply would look something like this:

Typical DC Power Supply



These typical power supply designs contain a large mains transformer (which also provides isolation between the input and output) and a dissipative series regulator circuit. The regulator circuit could consist of a single zener diode or a three-terminal linear series regulator to produce the required output voltage. The advantage of a linear regulator is that the power supply circuit only needs an input capacitor, output capacitor and some feedback resistors to set the output voltage.

2.3 Soil Moisture Sensor Module

This **soil moisture sensor module** is used to detect the moisture of the soil. It measures the volumetric content of water inside the soil and gives us the moisture level as output. The module has both digital and analog outputs and a potentiometer to adjust the threshold level.

Table 01: Soil Moisture Sensor Pin Description

Pin Name	Description
VCC	The VCC pin powers the module, typically with +5V
GND	Power Supply Ground
DO	Digital Out Pin for Digital Output.
AO	Analog Out Pin for Analog Output

Moisture Sensor Module Features & Specifications

- Operating Voltage: 3.3V to 5V DC
- Operating Current: 15mA
- Output Digital - 0V to 5V, Adjustable trigger level from preset
- Output Analog - 0V to 5V based on infrared radiation from fire flame falling on the sensor
- LEDs indicating output and power
- PCB Size: 3.2cm x 1.4cm
- LM393 based design
- Easy to use with Microcontrollers or even with normal Digital/Analog IC
- Small, cheap and easily available

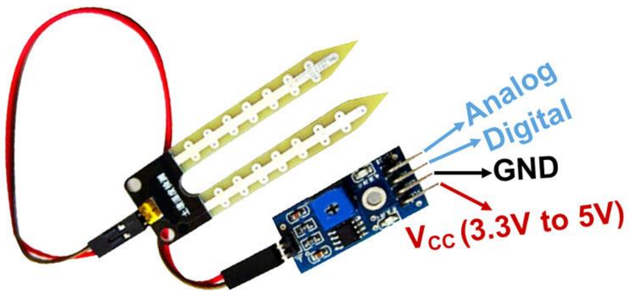


Figure 2.3: Soil Moisture Sensor

2.4 Temperature Sensor

The DHT11 is a basic, ultra-low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It's fairly simple to use, but requires careful timing to grab data.

DHT11 Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

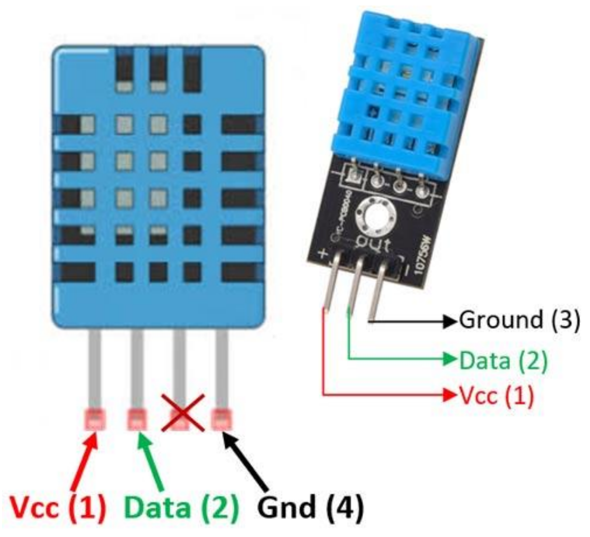


Figure 2.4: DHT11 Temperature Sensor

2.5 Node MCU

Node MCU is an open-source firmware for which open-source prototyping board designs are available. The name "Node MCU" combines "node" and "MCU" (micro-controller unit). The term "Node MCU" strictly speaking refers to the firmware rather than the associated development kits. Both the firmware and prototyping board designs are open source. The firmware uses the Lua scripting language.

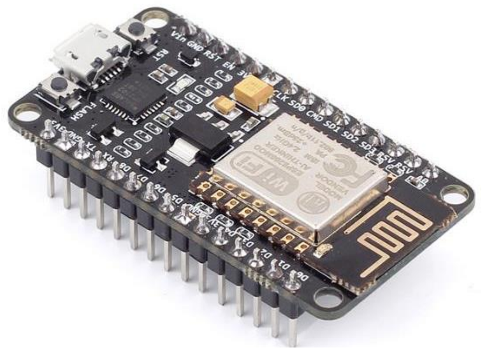


Figure 2.5: Node MCU

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications.

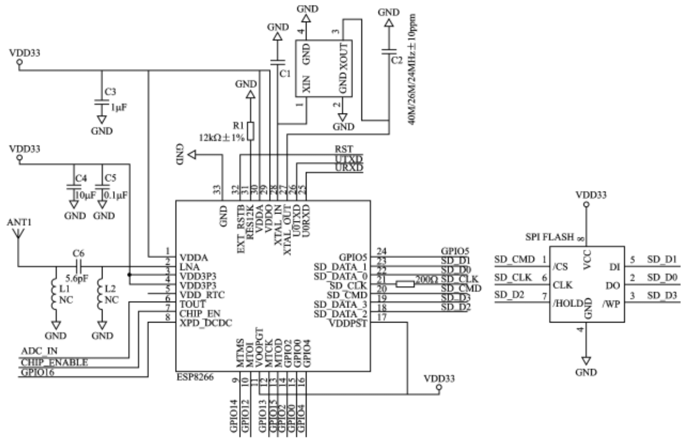


Figure 2.6: Node MCU Schematic Diagram

This an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Express if Systems, and hardware which is based on the ESP-12 module. The term "Node MCU" by default refers to the firmware rather than the development kits. The firmware uses the Luascripting language. It is based on the eLua project, and built on the Espress if Non-OS SDK for ESP8266. Node MCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects). Node MCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub.

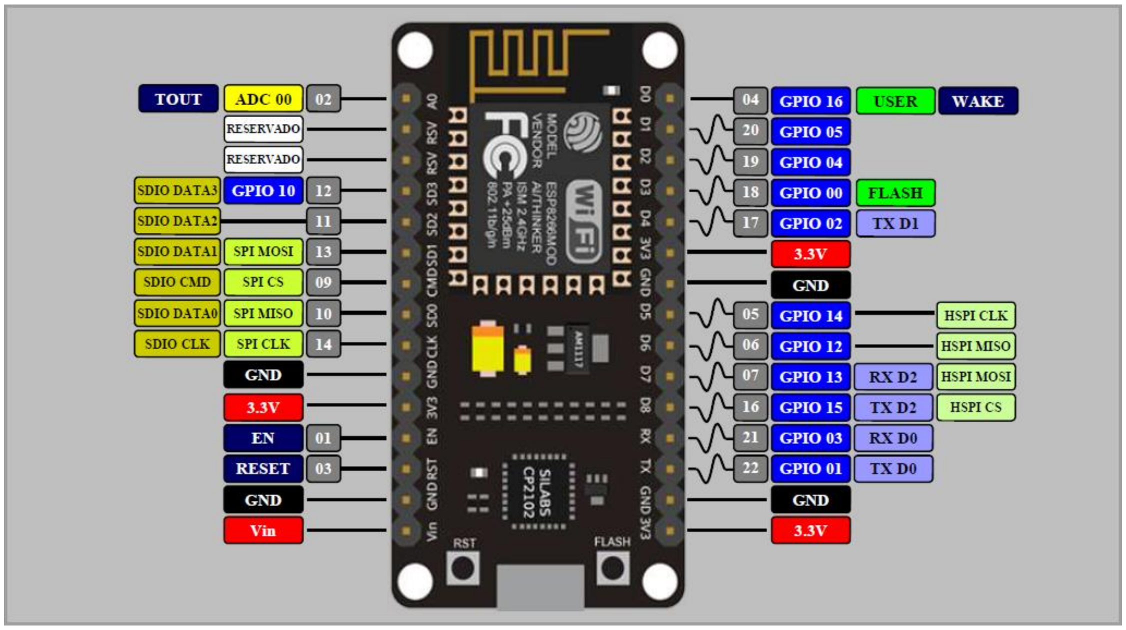


Figure 2.7: Node MCU Pin Out

Node MCU V3 ESP8266 ESP-12E is Wi-Fi development board that helps you to prototype your IoT product with few Lua script lines, or through Arduino IDE. The board is based on ESP8266 ESP-12E variant, unlike other ESP-12E, you won’t need to buy a separate breakout board, USB to serial adapter, or even solder it to a PCB to get started, you will only need a USB cable (Micro USB).

Features

1. Communication interface voltage: 3.3V.
2. Antenna type: Built-in PCB antenna is available.
3. Wireless 802.11 b/g/n standard
4. Wi-Fi at 2.4GHz, support WPA / WPA2 security mode
5. Support STA/AP/STA + AP three operating modes
6. Built-in TCP/IP protocol stack to support multiple TCP Client connections (5 MAX)
7. D0 ~ D8, SD1 ~ SD3: used as GPIO, PWM, IIC, etc., port driver capability 15mA
8. AD0: 1 channel ADC
9. Power input: 4.5V ~ 9V (10VMAX), USB-powered
10. Current: continuous transmission: ≈70mA (200mA MAX), Standby: <200uA
11. Transfer rate: 110-460800bps
12. Support UART / GPIO data communication interface
13. Remote firmware upgrade (OTA)
14. Flash size: 4MByte.

2.6 Mini Pump:

DC 3-6V Mini Micro Submersible Water Pump for fountain, garden and controlled water hydroponic systems.

Technical Specifications:

- DC Voltage: 2.5-6V
- Maximum lift: 40-110cm / 15.75″-43.4″
- Flow rate: 80-120L/H
- Outside diameter of water outlet: 7.5mm / 0.3″
- Inside diameter of water outlet: 5mm / 0.2″
- Diameter: Approx. 24mm / 0.95″

- Length: Approx. 45mm / 1.8″
- Height: Approx. 30mm / 1.2″
- Material: engineering plastic

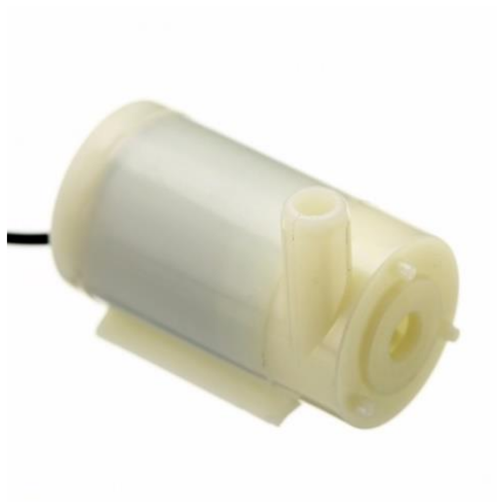


Figure 2.8: Mini Water Pump

2.7 BC547 Transistor

Table 02: Transistor Pin Configuration

Pin Number	Pin Name	Description
1	Collector	Current flows in through collector
2	Base	Controls the biasing of transistor
3	Emitter	Current Drains out through emitter

BC547 Transistor Features

- Bi-Polar NPN Transistor
- DC Current Gain (h_{FE}) is 800 maximum
- Continuous Collector current (I_C) is 100mA
- Emitter Base Voltage (V_{BE}) is 6V

- Base Current(I_B) is 5mA maximum
- Available in To-92 Package

Brief Description on BC547

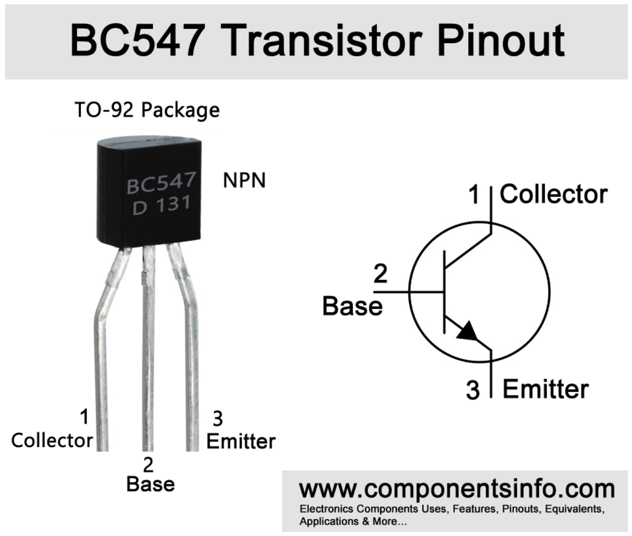


Figure 2.9: Transistor Pin Out

BC547 is a NPN transistor hence the collector and emitter will be left open (Reverse biased) when the base pin is held at ground and will be closed (Forward biased) when a signal is provided to base pin. BC547 has a gain value of 110 to 800, this value determines the amplification capacity of the transistor. The maximum amount of current that could flow through the Collector pin is 100mA, hence we cannot connect loads that consume more than 100mA using this transistor. To bias a transistor we have to supply current to base pin, this current (I_B) should be limited to 5mA.

When this transistor is fully biased then it can allow a maximum of 100mA to flow across the collector and emitter. This stage is called **Saturation Region** and the typical voltage allowed across the Collector-Emitter (V_{CE}) or Base-Emitter (V_{BE}) could be 200 and 900 mV respectively. When base current is removed the transistor becomes fully off, this stage is called as the **Cut-off Region** and the Base Emitter voltage could be around 660 mV.

BC547 as Switch

When a transistor is used as a switch it is operated in the **Saturation and Cut-Off Region** as explained above. As discussed a transistor will act as an Open switch during Forward Bias and as a Closed switch during Reverse Bias, this biasing can be achieved by supplying the required amount of current to the base pin. As mentioned the biasing current should maximum of 5mA. Anything more than 5mA will kill the Transistor; hence a resistor is always added in series with base pin. The value of this resistor (R_B) can be calculated using below formulae.

$$\mathbf{R_B = V_{BE} / I_B}$$

Where, the value of V_{BE} should be 5V for BC547 and the Base current (I_B depends on the Collector current (I_C). The value of I_B should not exceed mA.

BC547 as Amplifier

A Transistors acts as an Amplifier when operating in **Active Region**. It can amplify power, voltage and current at different configurations.

Some of the configurations used in amplifier circuits are

1. Common emitter amplifier
2. Common collector amplifier
3. Common base amplifier

Of the above types common emitter type is the popular and mostly used configuration. When uses as an Amplifier the DC current gain of the Transistor can be calculated by using the below formulae

$$\textbf{DC Current Gain} = \textbf{Collector Current } (\mathbf{I_C}) / \textbf{Base Current } (\mathbf{I_B})$$

Applications

- Driver Modules like Relay Driver, LED driver etc..
- Amplifier modules like Audio amplifiers, signal Amplifier etc..
- Darlington pair

2D model of the component

If you are designing a PCD or Perf board with this component then the following picture from the Datasheet will be useful to know its package type and dimensions.

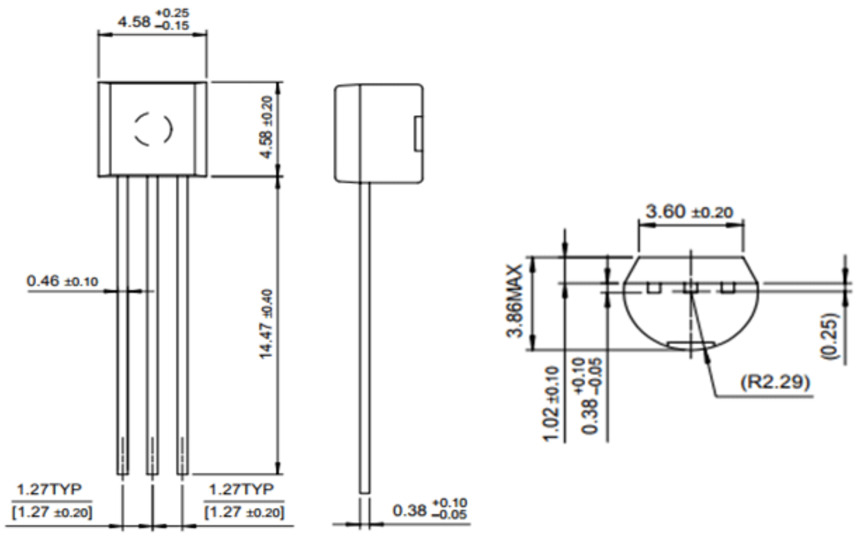


Figure 2,10: Transistor 2d Model

2.8 Capacitor

Capacitor is an electronic component that stores electric charge. The capacitor is made of 2 close conductors (usually plates) that are separated by a dielectric material. The plates accumulate electric charge when connected to power source. One plate accumulates positive charge and the other plate accumulates negative charge. The capacitance is the amount of electric charge that is stored in the capacitor at voltage of 1 Volt. The capacitance is measured in units of (F).The capacitor disconnects current in direct current (DC) circuits and short circuit in alternating current (AC) circuits.

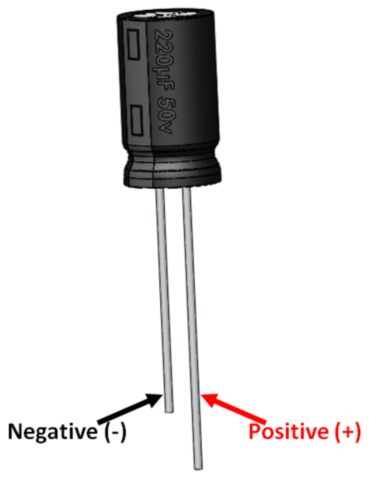


Figure 2.11: Capacitor Pin Out

Pin Configuration

The **Electrolytic Capacitors** have polarity. Meaning they have a positive and negative pin. The pin which is long is the positive pin and the pin which is short is the negative pin**.** You can also identify the polarity using the negative strip on the capacitor label. As shown in the picture above the negative pin will be directly under the negative symbol.

Features

- Capacitor Type - Electrolytic
- Has a high range of capacitance value starting from 0.01uF to 10000uF
- Has a high range of voltage value starting from 16V to 450V
- Can withstand a maximum of 105°C temperature

Other types of Capacitors

Ceramic Capacitor, Box Capacitor, Variable Capacitor.

Capacitor parameters selection

Ever wondered about the types of Electrolytic capacitors available in market and how to select one for your project? Electrolytic Capacitors can be classified based on two main parameters. One is their **Capacitance(C-Farad)** itself and the other is its **Voltage (V-Volts)** rating.

Capacitor is a passive component which can store a charge (Q). This charge (Q) will be a product of the value of capacitance (C) and the voltage (V) applied to it. The value of the capacitance and Voltage of a capacitor will be mentioned on its label.

Hence the amount of charge a capacitor can be found using the value of Voltage (V) and Capacitance (C) of the capacitor.

$$\mathbf{C = Q \times V}$$

Precaution

While using an Electrolytic capacitor care should always be taken to connect the positive pin to the positive of the circuit and the negative pin to the negative of the circuit. Also the voltage appearing across the capacitor terminals should always be less than the rated capacitor voltage (V). Failing to do so will lead to abnormal heating of the capacitor and might even burst.

Capacitor in series and parallel

In most of the circuits the value of the capacitance need not be exactly the same value specified in the circuit. A higher value of capacitance will generally not affect the performance of the circuit. However, the value of voltage should be the same or higher than the specified value to prevent the risk mentioned in precaution above. In that case, if you do not have the exact value you can use to capacitors in series or parallel to attain the desired value.

When two capacitors are *connected in parallel* then, the value of the capacitance(C) gets directly added up and the rated voltage (V) is remains the same in parallel as shown in the picture below.

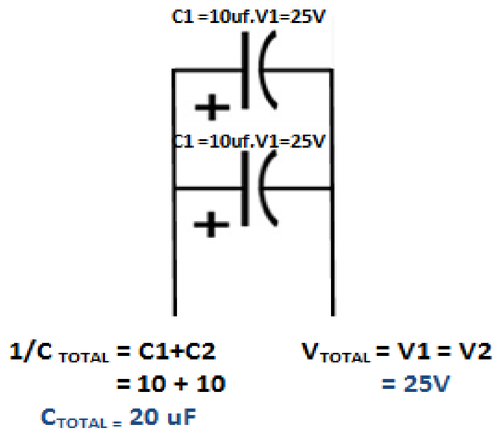
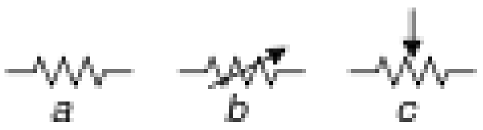


Figure 2.12: Capacitor Operation

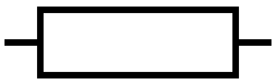
2.9 Resistor

A **resistor** is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses. High-power resistors that can dissipate many watts of electrical power as heat, may be used as part of motor controls, in power distribution systems, or as test loads for generators. Fixed resistors have resistances that only change slightly with temperature, time or operating voltage. Variable resistors can be used to adjust circuit elements (such as a volume control or a lamp dimmer), or as sensing devices for heat, light, humidity, force, or chemical activity.

Two typical schematic diagram symbols are as follows:



(a) resistor, (b) rheostat (variable resistor), and (c) potentiometer



IEC resistor symbol

The notation to state a resistor's value in a circuit diagram varies.

One common scheme is the RKM code following IEC 60062. It avoids using a decimal separator and replaces the decimal separator with a letter loosely associated with SI prefixes corresponding with the part's resistance. For example, *8K2* as part marking code, in a circuit diagram or in a bill of materials (BOM) indicates a resistor value of 8.2 kΩ. Additional zeros imply a tighter tolerance, for example *15M0* for three significant digits. When the value can be expressed without the need for a prefix (that is, multiplicator 1), an "R" is used instead of the decimal separator. For example, *1R2* indicates 1.2 Ω, and *18R* indicates 18 Ω.

Specifications:

- Resistance: 220 Ohms
- Power (Watts): 0.25W, 1/4W
- Temperature Coefficient: 350ppm/Celcius
- Tolerance: +/- 5%
- Case: Axial
- Size: 0.094" Dia x 0.248" L (2.40mm x 6.30mm)

CHAPTER 3

SOFTWARE IMPLEMENTATION

3.1 Arduino Software

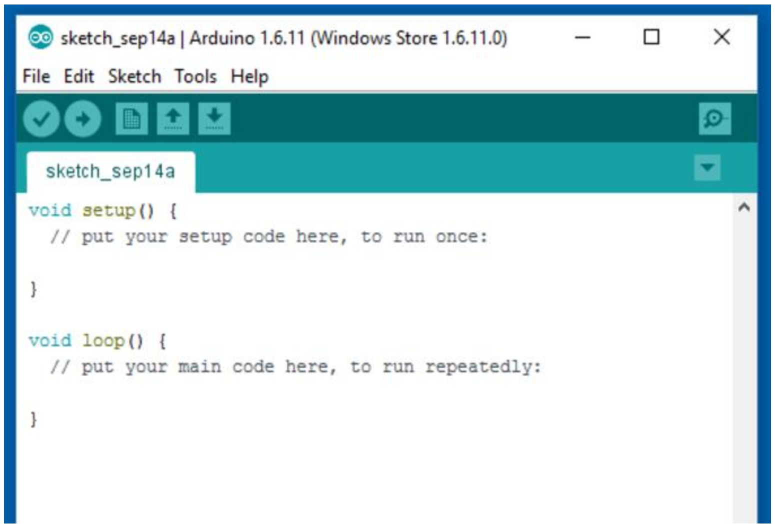
The digital microcontroller unit named as Arduino Nano can be programmed with the Arduino software IDE. There is no any requirement for installing other software rather than Arduino. Firstly, Select "Arduino Nano from the Tools, Board menu (according to the microcontroller on our board). The IC used named as ATmega328 on the Arduino Nano comes pre burned with a boot loader that allows us to upload new code to it without the use of an external hardware programmer.

Communication is using the original STK500 protocol (reference, C header files). We can also bypass the boot loader and programs the microcontroller through the ICSP (In Circuit Serial Programming) header. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available. The ATmega16U2/8U2 is loaded with a DFU boot loader, which can be activated by:

On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

The Arduino Nano is one of the latest digital microcontroller units and has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL at (5V) with serial communication, which is available on digital pins 0 -(RX) for receive the data and pin no.1 (TX) for transmit the data. 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 .in file is required. 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). A Software Serial library allows for serial communication on any of the Nano's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus. Arduino programs are written in C or C++ and the program code written for Arduino is called sketch. The Arduino IDE uses the GNU tool chain and AVR Lab to compile programs, and for uploading the programs it uses argued. As the Arduino platform uses Atmel microcontrollers, Atmel's development environment, AVR Studio or the newer Atmel Studio, may also be used to develop software for the Arduino.


sketch_sep14a | Arduino 1.6.11 (Windows Store 1.6.11.0)
File Edit Sketch Tools Help
sketch_sep14a

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 3.1: Arduino Software Interface IDE

The Arduino Integrated Development Environment - or 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. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Writing Sketches

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.

Save

Saves your sketch.

Serial Monitor

Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

File

New

Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.

Open

Allows to load a sketch file browsing through the computer drives and folders.

Open Recent

Provides a short list of the most recent sketches, ready to be opened.

Sketchbook

Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.

Examples

Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.

Close

Closes the instance of the Arduino Software from which it is clicked.

Save

Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.

Save as...

Allows to save the current sketch with a different name.

Page Setup

It shows the Page Setup window for printing.

Print

Sends the current sketch to the printer according to the settings defined in Page Setup.

Preferences

Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.

Quit

Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

Edit

Undo/Redo

Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.

Cut

Removes the selected text from the editor and places it into the clipboard.

Copy

Duplicates the selected text in the editor and places it into the clipboard.

Copy for Forum

Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.

Copy as HTML

Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.

Paste

Puts the contents of the clipboard at the cursor position, in the editor.

Select All

Selects and highlights the whole content of the editor.

Comment/Uncomment

Puts or removes the // comment marker at the beginning of each selected line.

Increase/Decrease Indent

Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.

Find

Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.

Find Next

Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.

Find Previous

Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

Sketch

Verify/Compile

Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

Upload

Compiles and loads the binary file onto the configured board through the configured Port.

Upload Using Programmer

This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a Tools -> Burn Bootloader command must be executed.

Export Compiled Binary

Saves a .hex file that may be kept as archive or sent to the board using other tools.

Show Sketch Folder

Opens the current sketch folder.

Include Library

Adds a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

Add File...

Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar.

Tools

Auto Format

This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.

Archive Sketch

Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

Fix Encoding & Reload

Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

Serial Monitor

Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

Board

Select the board that you're using. See below for descriptions of the various boards.

Port

This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

Programmer

For selecting a hardware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a boot loader to a new microcontroller, you will use this.

Burn Boot loader

The items in this menu allow you to burn a boot loader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

Help

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

Find in Reference

This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Beginning with version 1.0, files are saved with a .ino file extension. Previous versions use the .pde extension. You may still open .pde named files in version 1.0 and later, the software will automatically rename the extension to .ino.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

Uploading

Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx , /dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

Third-Party Hardware

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

For details on creating packages for third-party hardware, see the Arduino IDE 1.5 3rd party Hardware specification.

Serial Monitor

This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

3.2 Proteus Software

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronics design engineers and technicians to create schematics and electronics prints for manufacturing printed circuit boards.

The first version of what is now the Proteus Design Suite was called PC-B and was written by the company chairman, John Jameson, for DOS in 1988. Schematic Capture support followed in 1990 with a port to the Windows environment shortly thereafter. Mixed mode SPICE Simulation was first integrated into Proteus in 1996 and microcontroller simulation then arrived in Proteus in 1998. Shape based auto routing was added in 2002 and 2006 saw another major product update with 3D Board Visualization. More recently, a dedicated IDE for simulation was added in 2011 and MCAD import/export was included in 2015. Support for high speed design was added in 2017. Feature led product releases are typically biannual, while maintenance based service packs are released as required.

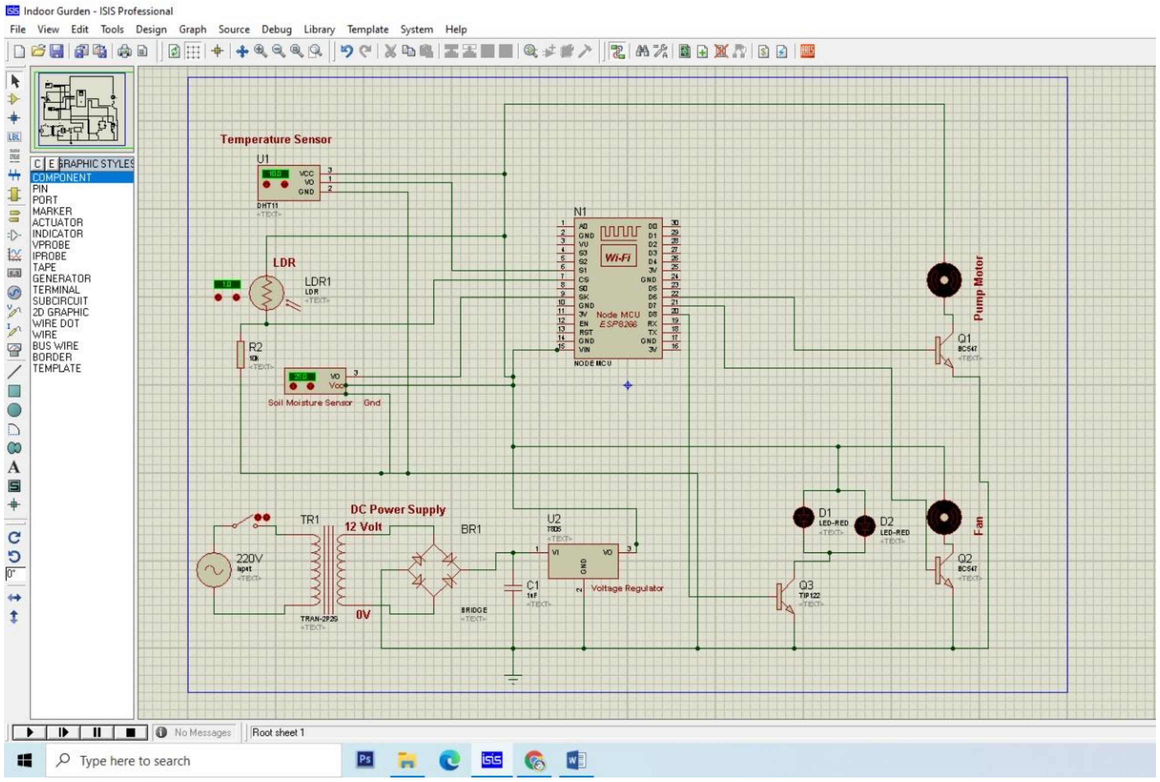


Figure 3.2: Our Project Design in Proteus Software

3.3 Adafruit Server:

Adafruit IO is a system that makes data useful. Our focus is on ease of use, and allowing simple data connections with little programming required. On an Arduino there are two different libraries you can use to access Adafruit IO. One library is based on the REST API, and the other library is based on the MQTT API. The difference between these libraries is that MQTT keeps a connection to the service open so it can quickly respond to feed changes. The REST API only connects to the service when a request is made so it's a more appropriate choice for projects that sleep for a period of time (to reduce power usage) and wake up only to send/receive data. If you aren't sure which library to use, try starting with the Adafruit MQTT library below.

Adafruit MQTT Client Library To use Adafruit IO with the MQTT protocol on an Arduino you can use the Adafruit MQTT Arduino library (https://adafru.it/fp6). This is a general-purpose MQTT library for Arduino that's built to use as few resources as possible so that it can work with platforms like the Arduino Uno. Unfortunately, platforms like the Trinket 3.3V or 5V (based on the ATtiny85) have too little program memory to use the library--stick with a Pro Trinket or better.

Adafruit IO

Having a better understanding about the Internet of Things and Cloud computing, let's now go over what Adafruit IO is about, and how it works. With the rise in digital transformations, IoT deployments in the cloud have become more popular. By deploying IoT solutions on the cloud, we have the following benefits:

- **Cost -** Reduces the cost of computing and storage by using various cloud services.
- **Scalability -** The "pay-as-you-go" pricing model allows a flexible pay model, also allowing scalability of the application.
- **Data control** - Data backup and recovery with high security.
- **Server uptime -** Allows very minimum or no downtime, with high server availability.

Adafruit IO is one such cloud provider focusing more on IoT deployments on the cloud. Adafruit IO supports different hardware like Raspberry PI, ESP2866, and Arduino. IoT developers prefer Adafruit IO over other IoT cloud providers for the following reasons:

- **Powerful API -** Provides us libraries for various programming languages, which also provides the built-in user interface support.
- **Dashboard -** Understanding data via charts and graphs enables us to make better decisions.
- **Privacy -** Data is secured in the cloud platform with better encryption algorithms.
- **Documentation & Community -** Many blogs with amazing community support allows continuous developments of the products.

Objective

we worked with NodeMCU on blinking lights of built-in LEDs. In this article, to demonstrate Adafruit IO works, we will send (publish) the LED brightness readings to the Adafruit IO cloud via Arduino IDE (written in C), and receive (subscribe) them via a Python server.

Installation

Follow the installation guide to get setup:

- Sign up by creating a new account in Adafruit IO.
- Make note of your private key by heading over to the “My Key” section.
- Then, “Create a new dashboard” with the desired name.
- In local setup, install a Python package called Adafruit_IO using pip install Adafruit_IO.
- Similarly, make sure to go through the previous article on how to set up the existing development environment.

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 Block Diagram

In this project, we have used a SMPS which has a Transformer to step down the input ac supply to get our desired value and then using the rectifier circuit and filter inside we get a dc output. Then the regulator IC has been used to output a regulated 5V so that we can use it to run the Node MCU.

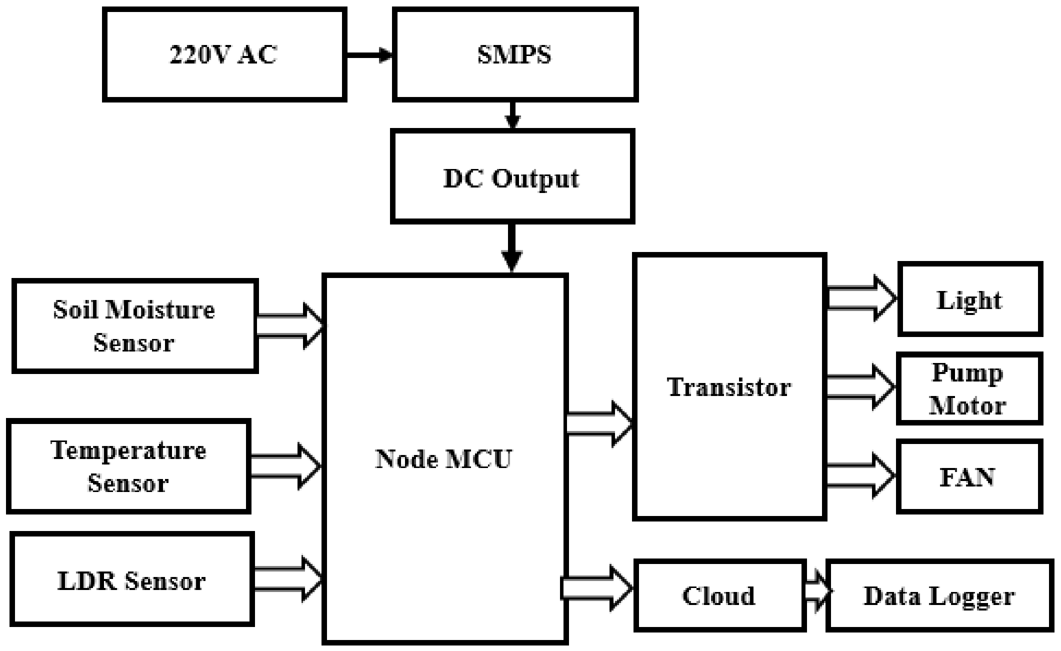


Figure 4.1: Block Diagram of IoT Based Indoor Garden System

4.2 Circuit Diagram

The schematic diagram here is representing the electrical circuit and the components of the project. Here we have used standardized symbols and lines.

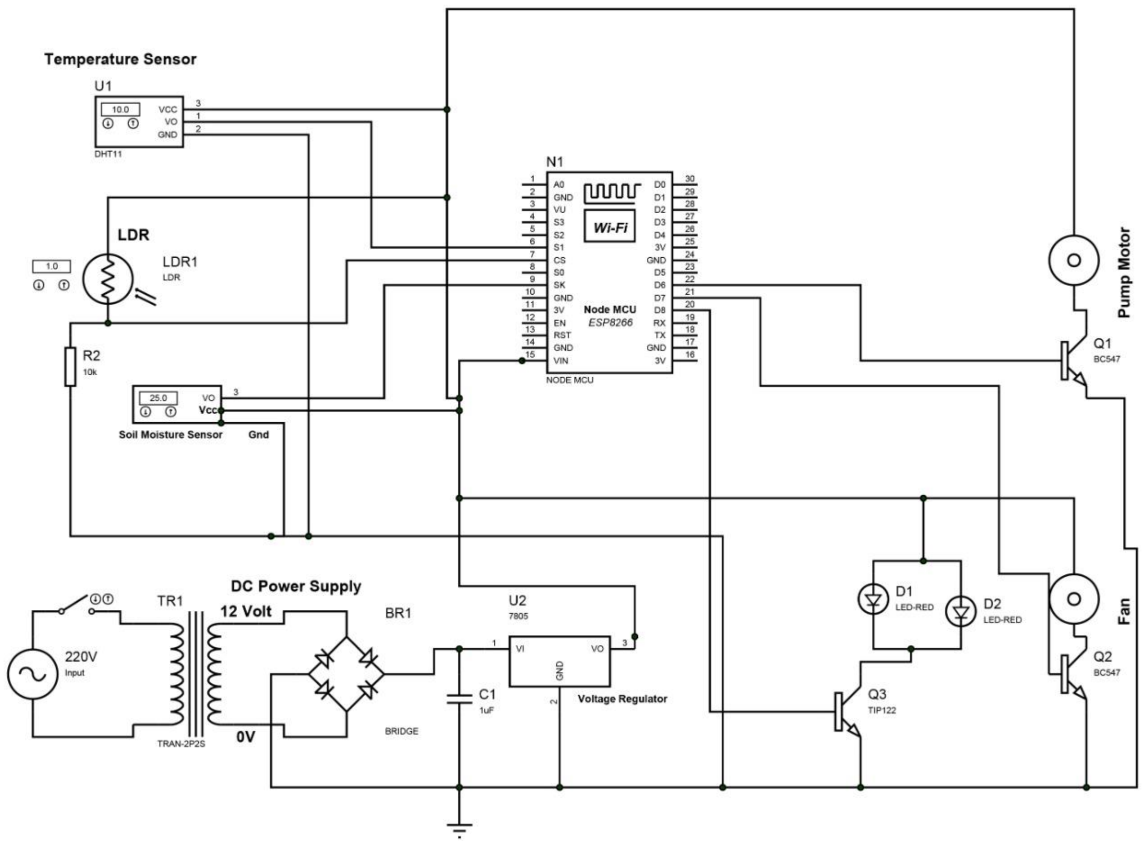


Figure 4.2: Circuit Diagram of IoT Based Indoor Garden System.

4.3 Working Principle

This is IoT Based Indoor Garden System. The main brain of our system is the Node MCU. The way of whole project works is that we take 220V (rms) ac power from the supply voltage and then feed it to a Switch Mode Power Supply or in short SMPS module. The SMPS simply converts the 220V ac to a pure dc of 5V. We will use this 5V dc output from the SMPS to run our Node MCU, Sensor and other units.

A soil moisture sensor is use here for measure the moisture of soil in garden, DHT11 is a Temperature sensor, it measure the indoor garden temperature. On the other hand, for measuring temperature these sensors use a NTC temperature sensor or a thermistor. A thermistor is actually a variable resistor that changes its resistance with change of the temperature. Here also used relay and many other things. This projects is mainly works to detect temperature, when temperature will extend a limit then fan will be On and start cooling inside of the garden. And soil moisture sensor senses the soil moisture. If there moisture is low, then relay will on and start the pump motor automatically. After pumping when soil will over moisture it will stop sensor by send a signal in Node MCU. And User is also sensing this reading in phone by the use of Adafruit server. To communicate over Adafruit server it's the function of sending message over the IoT. All information will save in a data logger system. We will download from this server anytime when we want.

4.4 The Project Prototype

The complete prototype of our project is shown below:

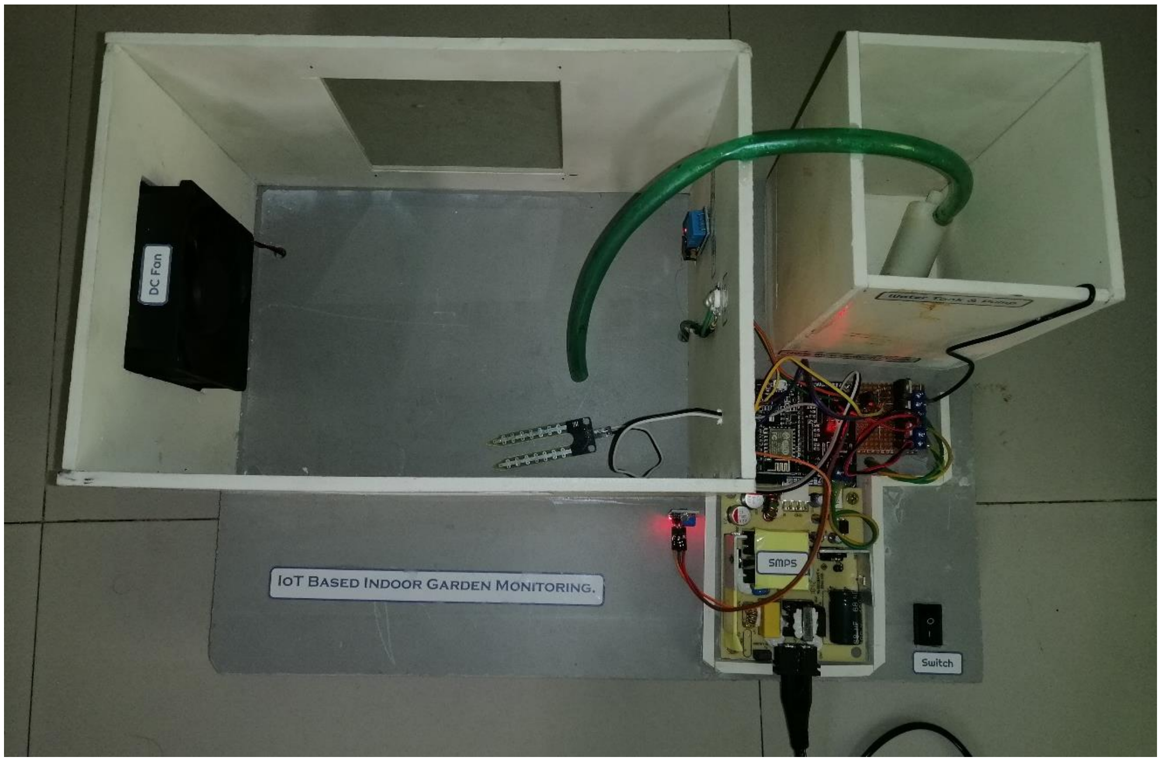


Figure 4.3: The Complete Prototype of the Project

4.5 Cost Analysis

In the below table we have summarized our project expenditure.

Table 3: Cost of Components with Price

No.	**Product Name**	**Specification**	**Qty. Price**	**Unit Price (Taka)**	**Total Price (Taka)**
01	SMPS	5V 5Amp	1	450	450
02	Soil Moisture Sensor		1	230	230

03	Temperature Sensor	DHT11	1	180	180
04	Node MCU	ESP8266	1	550	550
05	Transistor		2	50	100
06	Mini pump		1	350	350
07	Fan	12V	1	150	300
08	Others				1500
				Total =	3,660/=

CHAPTER 5

DISCUSSION & CONCLUSION

5.1 Discussion

In this project we collect all of the instruments, make a circuit design and connect all of these elements sequentially. IoT based Indoor Garden is made for the automation gardening purpose, which is mainly used in automatic daily life. This project has some advantages, that are it’s a ecofriendly project, low management cost, installation cost is low. Anywhere you can install by little effort. It's mainly needed for city gardens where men are so busy with their work. In this situation nursing a garden is so tough. It's a huge change in our daily life.

5.2 Conclusion

In this proposed smart irrigation system, an automatic mode of operation is designed for watering purposes. With this type of device, no extra assistance is needed. It works perfectly in the absence of the owner by detecting the soil condition through a moisture sensor and according to the condition of the soil, the Node MCU runs the irrigation system. It also notifies its user about the current status of both soil and motor. This method of watering the soil is very helpful for farmers especially in the rural areas because of its reasonable cost. To test the performance of the irrigation system in a garden, both dry and wet soils are arranged. This proposed system works as per expectation level by providing maximum automation without wasting any water. It is also tested that; the system is able to run the water pump without the sunlight for a few hours with the stored energy in the batteries. Before starting the operation, the moisture sensor needs to be checked, temperature sensor needs to be check for start the fan and all information are stored in a data logger (Adafruit) system.

Reference

[1] Usama Abdullah, Ayesha Ali (2014). GSM Based Water level and Temperature Monitoring System. International Journal of Recent Development in Engineering and Technology. Volume 3, Issue 2, August 2014.

[2] Asaad Ahmed Mohammed ahmed Eltaieb, Zhang Jian Min, “Automatic Water Level Control System”, International Journal of Science and Research (IJSR)2013

[3] Nivit Yadav, “CPCB Real Time Water Quality Monitoring", port: Centre for Science and Environment,2013.

[4] B. K. Bose, “Global warming: energy, environmental pollution, and the impact of power electronics,” IEEE Ind. Electron. Mag., vol. 4, no. 1, pp. 6–17, Mar. 2010.

[5] K. Ahmed, J. Paul, M. M. Rahman, A. Shufian, M. S. Tanvir, and M. M. I. Sagor, “Automatically controlled energy conservation system for corporate office based on microcontroller,” IEEE Int. Conf. Adv. Sci., Eng. Robot. Technol., Dhaka, Bangladesh, May 3–5, 2019, in press.

[6] A. Shufian, M. M. Rahman, K. Ahmed, R. Islam, M. Hasan, and T. Islam, “Design and implementation of solar power wireless battery charger,” IEEE Int. Conf. Adv. Sci., Eng. Robot. Technol., Dhaka, Bangladesh, May 3–5, 2019, in press.

[7] M. R. Habib, K. Ahmed, N. Khan, M. R. Kiran, M. A. Habib, M. T. Hasan, and O. Farrok, “PID controller based automatic solar powerdriven grass cutting machine,” IEEE Int. Conf. Comput., Commun., Chemical, Mater. Electron. Eng., Rajshahi, Bangladesh, Jul. 11–12, 2019, in press.

[8] Website: https://www.toppr.com/guides/biology/crop-production and management /irrigation/. [Accessed: 27-Jul-2019].

[9] M. Monica, B. Yeshika, G. S. Abhishek, H. A. Sanjay, and S. Dasiga, “IoT based control and automation of smart irrigation system,” IEEE Int. Conf. Recent Innovations Signal Process. Embedded Syst., Bhopal, India, Oct. 27–29, 2017, pp. 601–607.

[1] Usama Abdullah, Ayesha Ali (2014). GSM Based Water level and Temperature Monitoring System. International Journal of Recent Development in Engineering and Technology. Volume 3, Issue 2, August 2014.

Appendix

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include "DHT.h"

// DHT 11 sensor
#define DHTPIN D5
#define moisture D7
#define motor D8
#define ldr D1
#define light D2
#define fan D6

#define DHTTYPE DHT11

// WiFi parameters
#define WLAN_SSID       "Rasel."
#define WLAN_PASS       "123456789"

// Adafruit IO
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "kibrea"
#define AIO_KEY         "aio_VgYy968mQ9zYFgWrahxB5VPm2rAo"
// DHT sensor
```

```
DHT dht(DHTPIN, DHTTYPE, 11);

// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

// Setup feeds for temperature & humidity
Adafruit_MQTT_Publish temperature = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/temperature");
Adafruit_MQTT_Publish humidity = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/humidity");
Adafruit_MQTT_Publish Motoronoff = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/Motoronoff");
Adafruit_MQTT_Publish lightonoff = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/lightonoff");

/************************** Sketch Code ***********************************/

void setup() {

  // Init sensor
  dht.begin();
pinMode(D7,INPUT);
pinMode(D8,OUTPUT);
pinMode(D1,INPUT);
pinMode(D2,OUTPUT);
pinMode(D6,OUTPUT);

  Serial.begin(115200);
  Serial.println(F("Adafruit IO Example"));

  // Connect to WiFi access point.
  Serial.println(); Serial.println();
  delay(10);
  Serial.print(F("Connecting to "));
  Serial.println(WLAN_SSID);

  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(F("."));
  }
  Serial.println();

  Serial.println(F("WiFi connected"));
  Serial.println(F("IP address: "));
  Serial.println(WiFi.localIP());

  // connect to adafruit io
  connect();

}

void loop() {
  // ping adafruit io a few times to make sure we remain connected
  if(! mqtt.ping(3)) {
    // reconnect to adafruit io
    if(! mqtt.connected())
      connect();
  }

  // Grab the current state of the sensor
  int humidity_data = (int)dht.readHumidity();
  int temperature_data = (int)dht.readTemperature();
  int moisture_sensor = digitalRead(moisture);
    int ldr_sensor = digitalRead(ldr);

    Serial.print(humidity_data);

 int temperature_data1 = temperature_data;
    Serial.println(temperature_data1);
  if (temperature_data1 >34){
      digitalWrite(fan,HIGH);
  }
    if (temperature_data1 <34){
      digitalWrite(fan,LOW);

  }
  if (! temperature.publish(temperature_data1))
    Serial.println(F("Failed to publish temperature"));
  else
```

```
    Serial.println(F("Temperature published!"));

  if (! humidity.publish(humidity_data))
    Serial.println(F("Failed to publish humidity"));
  else
    Serial.println(F("Humidity published!"));

if(moisture_sensor==HIGH){
  digitalWrite(motor,HIGH);
}
if(moisture_sensor==LOW){
  digitalWrite(motor,LOW);
}

  if (! Motoronoff.publish(moisture_sensor)) {
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }

if(ldr_sensor==HIGH){
  digitalWrite(light,HIGH);
}
if(ldr_sensor==LOW){
  digitalWrite(light,LOW);
}
if (! lightonoff.publish(ldr_sensor)) {
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }
  // Repeat every 10 seconds
  delay(10000);

}

// connect to adafruit io via MQTT
void connect() {

  Serial.print(F("Connecting to Adafruit IO... "));

  int8_t ret;

  while ((ret = mqtt.connect()) != 0) {

    switch (ret) {
      case 1: Serial.println(F("Wrong protocol")); break;
      case 2: Serial.println(F("ID rejected")); break;
      case 3: Serial.println(F("Server unavail")); break;
      case 4: Serial.println(F("Bad user/pass")); break;
      case 5: Serial.println(F("Not authed")); break;
      case 6: Serial.println(F("Failed to subscribe")); break;
      default: Serial.println(F("Connection failed")); break;
    }

    if(ret >= 0)
```

```
      mqtt.disconnect();

    Serial.println(F("Retrying connection..."));
    delay(5000);

  }

  Serial.println(F("Adafruit IO Connected!"));

}
```

Thank You