Mini Project Report on A

CROP MONITORING USING ESP32 AND LOCAL SERVER

Submitted

in partial fulfilment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY

In

Electronics and Communication Engineering

G. Rohith Vishaal-17R11A0462

D.Bhargavi-18R15A0415

T. Raju-18R15A0419

Under the Guidance of

Mrs. Padmaja A. R. L

Assistant Professo

Department of Electronics and Communication Engineering

GEETHANJALI COLLEGE OF ENGINEERING AND TECHNOLOGY,

(UGC Autonomous)

(Approved by AICTE, Permanently Affiliated to JNTUH, Accredited by NBA, Accredited by NAAC with "A" Grade and ISO 9000:2008 Certified)

Cheeryal (V), Keesara (M), Medchal Dist. Telangana – 501 301, INDIA Mobile:9391199932, Landline:040-31001618, Fax:040-24220320

email:info@gcet.edu.in, Web: http://www.geethanjaliinstitutions.com

2017-2021

GEETHANJAL COLLEGE OF ENGINEERING AND TECHNOLOGY

Department of Electronics and Communication Engineering

<u>CERTIFICATE</u>

This is to certify that the project report titled AUTOMATIC WATER DISPENSER being submitted by **G. Rohith Vishal, D. Bhargavi, T. Raju** bearing hall ticket number **17R11A0462, 1815A0415, 18R15A0419,** in partial fulfilment of the requirements for the award of the Degree of **Bachelor of Technology** in ***Electronics and Communication Engineering*** is a record of bona-fide work carried out under my guidance and supervision. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide
Mrs. Padmaja A. R. L
Assistant professor

Dr. S. Suryanarayana,
HOD, ECE

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

We, the Students of ECE department of Geethanjali College of Engineering and Technology would like to convey heartfelt thanks to **Dr. S. Udaya Kumar**, Principal of the college for the wonderful guidance and encouragement given to us to move ahead in the execution of this project.

We are highly grateful to the great personality in the field of Electronics, none other than **c** Head of the Department of **Electronics and Communication Engineering** of **GCET** for guiding and taking care of our career in this field.

We are very happy for being guided by Mrs. **Padmaja A. R. L,** Assistant Professor for his able guidance given to us to complete our technical work successfully.

Above all, we are very much thankful to the **management of Geethanjali College of Engineering & Technology** which was established by the **high profile intellectuals** for the cause of Technical Education in modern era. We wish that **GCET** sooner should become a deemed university and produce uncountable young engineers and present them to the modern technical world.

With Regards

G. Sai Rohith Vishaal (17R11A0462)

D. Bhargavi (18R15A0415)

T. Raju (18R15A0419)

ABSTRACT

The aim of this project is to design a Crop monitoring and management system for a Green House using ESP32 and a MQTT server hosted locally based on the Internet of things (IoT). A greenhouse is a covered area where plants grow and cultivate. It is also known as land of controlled crops and plants. There are some important parameters to be monitored inside the greenhouse are temperature, humidity, soil moisture, pH, luminosity using DHT11, Soil moisture sensor, pH Sensor, LDR (Light Dependent Resistor) respectively. It will start monitoring when these sensors are connected and necessary actions are performed according to the coded conditions. This project represents the technological solution to automate and improve the management of greenhouse and yield higher productivity. Internet of things (IoT) was developed for connecting a billion of devices into an internet. A huge amount of information is transferred between the electronic devices. It is a new way to interact between device and people. This shows that how the wireless communication has been for future vision in the monitoring system. Internet of things (IoT) will play a major role in day to day life in the future.

CHAPTER 1
INTRODUCTION

The world over decades has made considerable advancement in automation. Automation is employed for every sector where it is home, industry agriculture. Greenhouse is the technical approach in which farmers in the rural areas will be benefitted by automatic monitoring& control of greenhouse environment replace the direct supervisions of the human. The paper focuses on the generic architecture which can be applied for many other automation applications. The great needs are growing of crops with advancement of technology. Greenhouse are climate-controlled structure with wall &roofs &specially designed for off season growing of plants. Internet of things is one of the latest advances in information &communication technologies providing global connectivity &management of sensors devices, users with information. Temperature/Humidity Sensor, Moisture Sensor, Light Sensor efficiently inside the greenhouse by actuating a Cooling Fan, LED, Motor respectively according to the required conditions of the crops to achieve the maximum growth &yield. First let us discuss about green house & greenhouse effect. Green house is something related to a building or a place where small plants & vegetables are grown. And the area under greenhouse is covered with glass or translucent plastic roof sand this plays & important role for the vegetation in cold regions, because it is still very cold to take them to an outside environment. And now moving forward to discuss about the greenhouse effect. Greenhouse effect is simply a process in which various greenhouse gases entraps the infrared rays from the sunlight thus leading to increase of level of carbon dioxide which further helps in increasing the amount of chlorophyll and the leading to impressive plant growth & yield. And the greenhouse system helps in boosting the efficiency. And thus, our system is based to perform such activities that are to monitor& control the system from a particular place which would take care weather inside the greenhouse.

CHAPTER 1
LITERATURE SURVEY

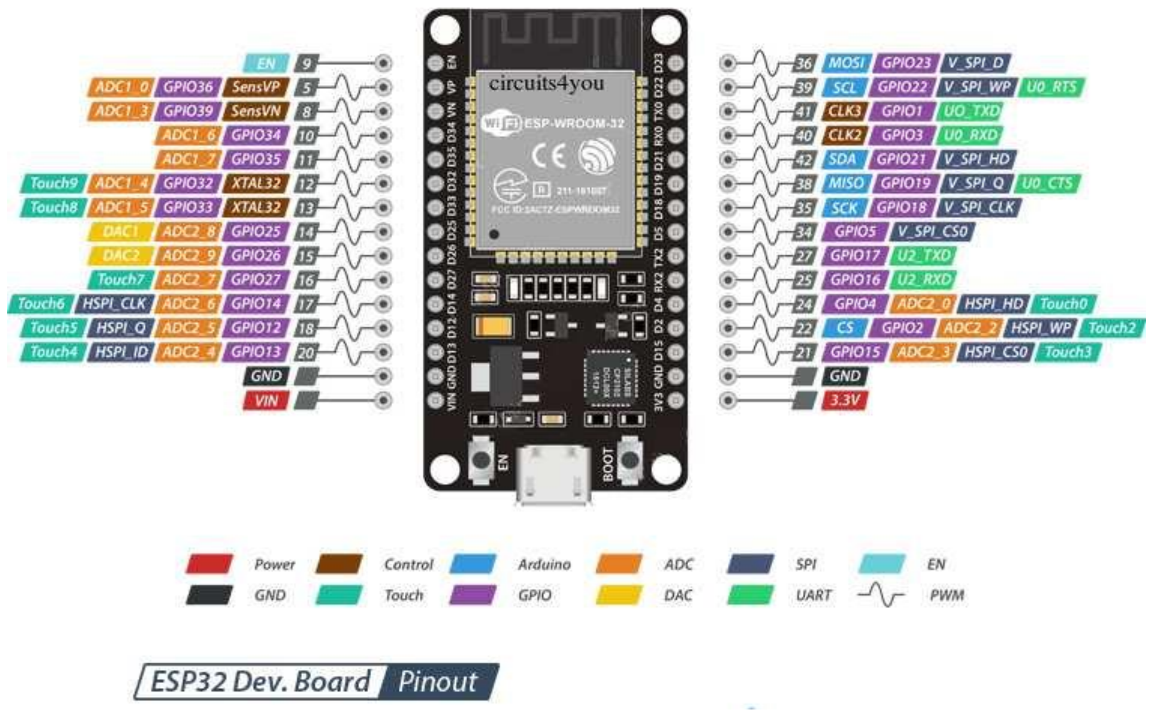
Automated greenhouse system helps the farmers by controlling the environment parameters through the environmental parameters through the internet of things (IOT) including crop health inspection using image analysis the greenhouse is generally affected by two factors: plant diseases &weather condition, which leads to the fall in production. The weather condition can be controlled through Microcontroller Unit (MCU) & the plant diseases can be monitored using image inspection system. The research recommends cheaper image evaluation framework for the plant disease can be monitored using image inspection system. The research recommends cheaper image evaluation framework for the plant diseases analysis& fully automated greenhouse data security. The prototype of the proposed system consists of Temperature/Humidity Sensor, Moisture Sensor, Light Sensor and pH Sensor. The Motor, Light, are controlled by ESP32 through an app upon reaching predetermined threshold values. The proposed architecture is equipped with embedded data security by implementing Extended Tiny Encryption Algorithms (XTEA) lastly, the agriculturists can familiarize with the recommended framework through the cloud-centred application. The autonomous frameworks permit the agriculturists to evaluate &control their greenhouse ecology remotely.

CHAPTER 3
COMPONENTS DESCRIPTION AND CIRCUIT DESIGN

3.1 COMPONENTS DESCRIPTION

3.1.1 ESP-WROOM 32

ESP32-WROOM-32 (ESP-WROOM-32) has 38 pins.



Pin Definitions

PIN DISCRIPTION

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply.
EN	3	I	Chip-enable signal. Active high.
SENSOR_VP	4	I	GPIO36, SENSOR_VP, ADC_H, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, SENSOR_VN, ADC1_CH3, ADC_H, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_

IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Ground
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14,MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2*	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3*	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD*	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/CLK*	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0*	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1*	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Ground

Functional Description

This chapter describes the modules and functions integrated in ESP32-WROOM-32 (ESP-WROOM-32).

CPU and Internal Memory

ESP32-D0WDQ6 contains two low-power Xtensa ® 32-bit LX6 microprocessors. The internal memory includes:

- 448 kB of ROM for booting and core functions.
- 520 kB (8 kB RTC FAST Memory included) of on-chip SRAM for data and instruction.

– 8 kB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.

- 8 kB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the co-processor during the Deep-sleep mode.
- 1 kbit of eFuse, of which 320 bits are used for the system (MAC address and chip configuration) and the remaining 704 bits are reserved for customer applications, including Flash-Encryption and Chip-ID.

External Flash and SRAM

ESP32 supports up to four 16-MB of external QSPI flash and SRAM with hardware encryption based on AES to protect developers' programs and data.

ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- Up to 16 MB of external flash are memory-mapped onto the CPU code space, supporting 8, 16 and 32-bit access. Code execution is supported.
- Up to 8 MB of external flash/SRAM are memory-mapped onto the CPU data space, supporting 8, 16and 32-bit access. Data-read is supported on the flash and SRAM. Data-write is supported on the SRAM.

ESP32-WROOM-32 (ESP-WROOM-32) integrates 4 MB of external SPI flash. The 4-MB SPI flash can be memory- mapped onto the CPU code space, supporting 8, 16 and 32-bit access. Code execution is supported. The integrated SPI flash is connected to GPIO6, GPIO7, GPIO8, GPIO9, GPIO10 and GPIO11. These six pins cannot be used as regular GPIO.

Crystal Oscillators

The ESP32 Wi-Fi/BT firmware can only support 40 MHz crystal oscillator for now.

RTC and Low-Power Management

With the use of advanced power management technologies, ESP32 can switch between different power modes.

- Power modes

– Active mode: The chip radio is powered on. The chip can receive, transmit, or listen.

– Modem-sleep mode: The CPU is operational and the clock is configurable. The Wi-Fi/Bluetooth base- band and radio are disabled.

- Light-sleep mode: The CPU is paused. The RTC memory and RTC peripherals, as well as the ULP co-processor are running. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
- Deep-sleep mode: Only the RTC memory and RTC peripherals are powered on. Wi-Fi and Bluetooth connection data are stored in the RTC memory. The ULP co-processor can work.
- Hibernation mode: The internal 8-MHz oscillator and ULP co-processor are disabled. The RTC recovery memory is powered down. Only one RTC timer on the slow clock and some RTC GPIOs are active. The RTC timer or the RTC GPIOs can wake up the chip from the Hibernation mode.

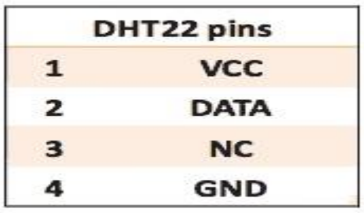
Specifications:

- Operating Power Supply: 2.7V - 3. 6 V
- Operating Current Average: 80mA
- Internal Memory: 448 KB of ROM for Booting 520 KB of on chip RAM for data and Instructions 16 KB SRAM in RTC
- Wi-Fi: 802.11b/g/n, 2.4Ghz
- Peripherals: Capacitive touch, ADCs, DACs, 12C (Inter Integrated Circuit), UART(Universal Asynchronous Receiver/Transmitter) , PWM(Pulse Width Modulation).

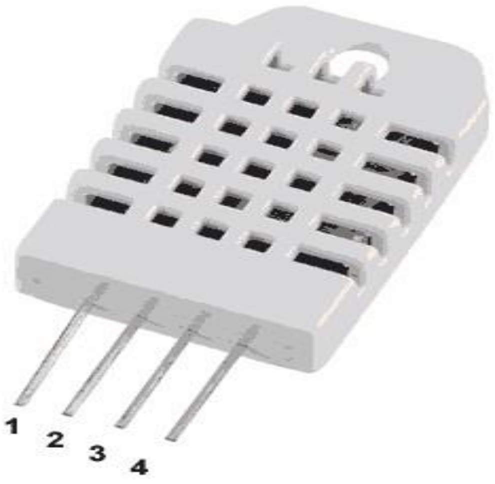
3.1.2 DHT22 Sensor

- The DHT22 is a low-cost digital temperature and humidity sensor with a single wire digital interface. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin.

- The sensor is calibrated and doesn’t require extra components so it can get the right to measuring relative humidity and temperature. It is quite simple to use but it requires careful timing to grab data. It can only get new data from it once every 2 seconds



DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



Working Principle of DHT22:

Temperature:

The temperature is measured with the help of a NTC thermistor or negative temperature coefficient thermistor. These thermistors are usually made with semiconductors, ceramic and polymers. The resistance of the device is inversely proportional with temperature and follows a hyperbolic curve. Temperature using NTC often found out Steinhart Hart equation.

Humidity:

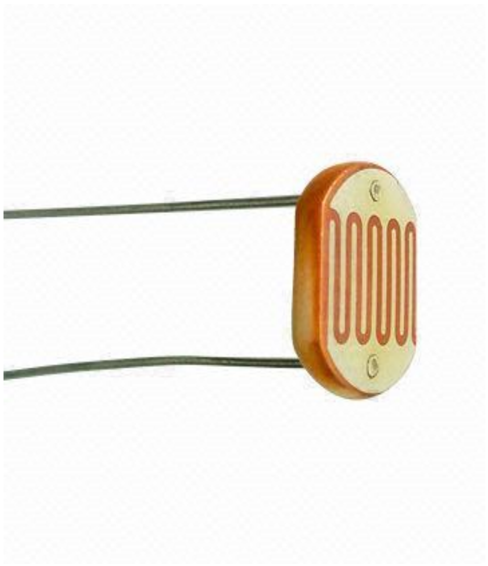
Humidity is sensed using a moisture dependent resistor. It has two electrodes and in between them there exist a moisture holding substrate which holds moisture. The Conductance and hence resistance changes with changing humidity.

Specifications:

- 3.3 to 6v power and I/O
- 1.5 Ma current use during conversion.
- 0-100% humidity readings ±0.5◦ accuracy

  Up to 0.5hz sampling rate (once every 2 seconds

3.1.3 LDR Sensor (LIGHT DEPENDENT RESISTOR)



Specifications:

Maximum power dissipation	200Mw
Maximum voltage @ 0lux	200V
Peak wavelength	600nm
Minimum resistance @ 10 lux	1.8Ω
Maximum resistance @10 lux	4.5KΩ
Dark resistance after 1 second	0.03MΩ
Dark resistance after 5 seconds	0.25MΩ

An LDR is a component that has a variable resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits. They can be described by a variety of names from light dependent resistor, LDR, photoresistor, or even photo cell, photocell or photoconductor. It is made of a high resistance semiconductor. In the dark, a photoresistor can have a resistance as high as several megohms (MΩ), while in the light, a photoresistor can have a resistance as low as a few hundred ohms.

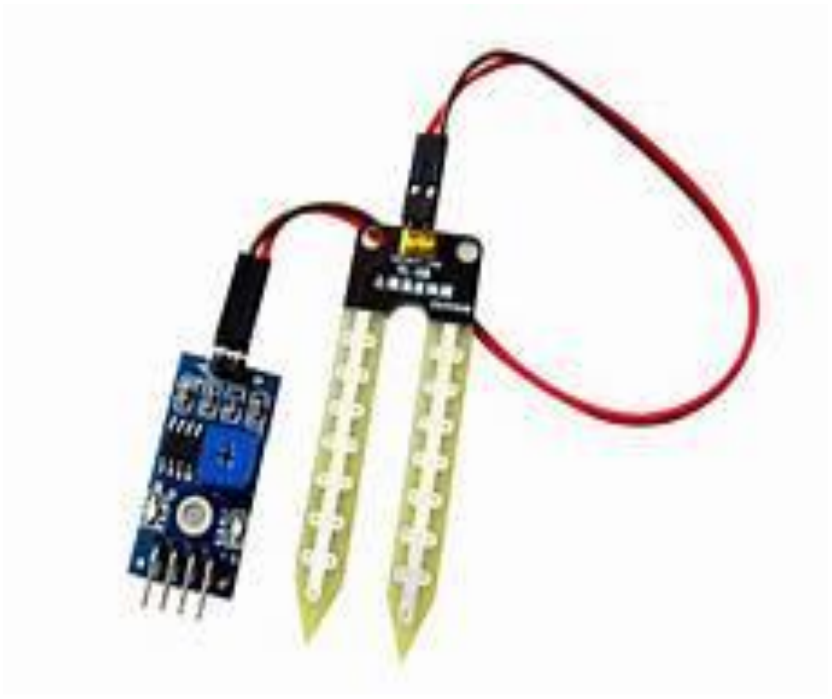
Working Principle of LDR:

- A light dependent resistor works on the principle of photo conductivity. Photo conductivity is an optical phenomenon.
- When light falls i.e. when the photons fall on the device, the electrons in the valence band of the semiconductor material are excited to the conduction band. These

photons in the incident light should have energy greater than the band gap of the semiconductor material to make the electrons jump from the valence band to the conduction band.

- Hence when light having enough energy strikes on the device, more and more electrons are excited to the conduction band which results in large number of charge carriers. The result of this process is more and more current starts flowing through the device when the circuit is closed and hence it is said that the resistance of the device has been decrease.

3.1.4 moisture sensor



Specifications:

- Range: 0 to 45% volumetric water content in soil(capable of 0 to 100% VWC with alternate calibration)
- Accuracy: ±4% typical
- Typical Resolution:0.1%
- Power: 3mA@ 5VDC
- Operating temperature: -40°C to +60°C.

The Soil Moisture Sensor uses capacitance to measure the water content of soil (by measuring the dielectric permittivity of the soil, which is a function of the water content). Simply insert this rugged sensor into the soil to be tested, and the volumetric water content of the soil is reported in percent.

Working Principle of Moisture Sensor:

- The Soil Moisture Sensor uses capacitance to measure dielectric permittivity of the surrounding medium. In soil, dielectric permittivity is a function of the water content. The sensor creates a voltage proportional to the dielectric permittivity, and therefore the water content of the soil.
- The Soil Moisture Sensor is used to measure the loss of moisture over time due to evaporation and plant, evaluate optimum soil moisture contents for various species of plants, monitor soil moisture content to control irrigation in greenhouses and enhance yield of the plant.

3.1.5 PH Sensor

Soil content and quality is a key factor of crop health. Physically, soil material and density affects water infiltration and retention. Chemically, pH of soil affects nutrient availability and uptake. Some soil characteristics are permanent qualities of a soil, and therefore part of soil selection criteria, others s by measuring soil using a soil pH meter.

pH monitoring is a crucial component of horticulture maintenance for crop health. The best pH for plants is typically can be controlled and optimized. The best way to start the process of improving soil health between 5.5 and 6.5, though some plants may thrive in more acidic or more alkaline soils. Soil nutrients are tied strongly with pH of soil, pH control maximizes the efficiency of fertilizers by controlling nutrient bioavailability pH of soils affects the presence of toxic elements structure of certain soils, and the activity of soil bacteria.

Specifications:

- Type: Sealed, gel-filled, epoxy body, Ag/AgCl.
- Shaft Diameter: 12mm OD.
- Response time: 90% of final reading in 1 second.
- Temperature Range:5 to 80℃.
- Range: pH 0-14.
- Accuracy: ±0.2 pH units.
- Iso-potential pH: pH 7(point at which temperature has no effect).

3.1.6 Servo motor

Servo motor is a tiny and light weight server motor with high output power. Servos are controlled by sending an electrical pulse of variable width or pulse width modulation (PWM), through the control wire. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. Each part of the servo motor like stator, winding, shaft, rotor, encoder makes the servo properly function. The motor with power +5V using the red wire and brown wire is ground wire connected to the ground of system, orange wire is PWM signal is given in through this wire to drive the motor.

Servo motor to open or close the window blinds to allow sunlight or restrict it.

Specifications:

- Operating Voltage: +5V typically
- Torque: 2.5kg/cm
- Operating speed: 0.1s/60◦
- Gear Type: Plastic
- Rotation: 0◦-180◦
- Weight of motor: 9gm
- Package includes gear horns and screws

3.1.7 Submersible water motor pump

This is micro submersible w easily integrate to water system. This water pump works usi ich drain the water through its inlet and released it through er to the crops by drip irrigation method.

Specifications:

- Operating Voltage: 2.5V-3V
- Operating Current: 130 -220mA
- Flow Rate: 80 -120 L/H
- Maximum Lift: 40 -110mm
- Continuous Working Life: 500 hours
- Driving Mode: DC, Magnetic Driving
- Material: Engineering Plastic

3.1.9 Thermostat

h senses the temperature of a physical system and p emperature is maintained near a desired point. Therr by a solid stainless steel ring and works with 95% ning thermostat learns what temperatures you like , and can be controlled from anywhere over Wi-F

Specifications:

- Screen: 24-bit colour LCD

  320 * 320pixel Display

  1.75 in (44.5mm) diameter
- Battery: Built -in rechargeable lithium-ion battery
- Power: Less than 1KWh/month
- Connectivity requirement: wi-Fi internet connection
- Operating conditions: Temperature-32℉-104℉(0℃-40℃)

  Humidity- Up to 90% RH unpackaged

  Pressure -Up to 10,000

3.1.10 ARDUINO IDE

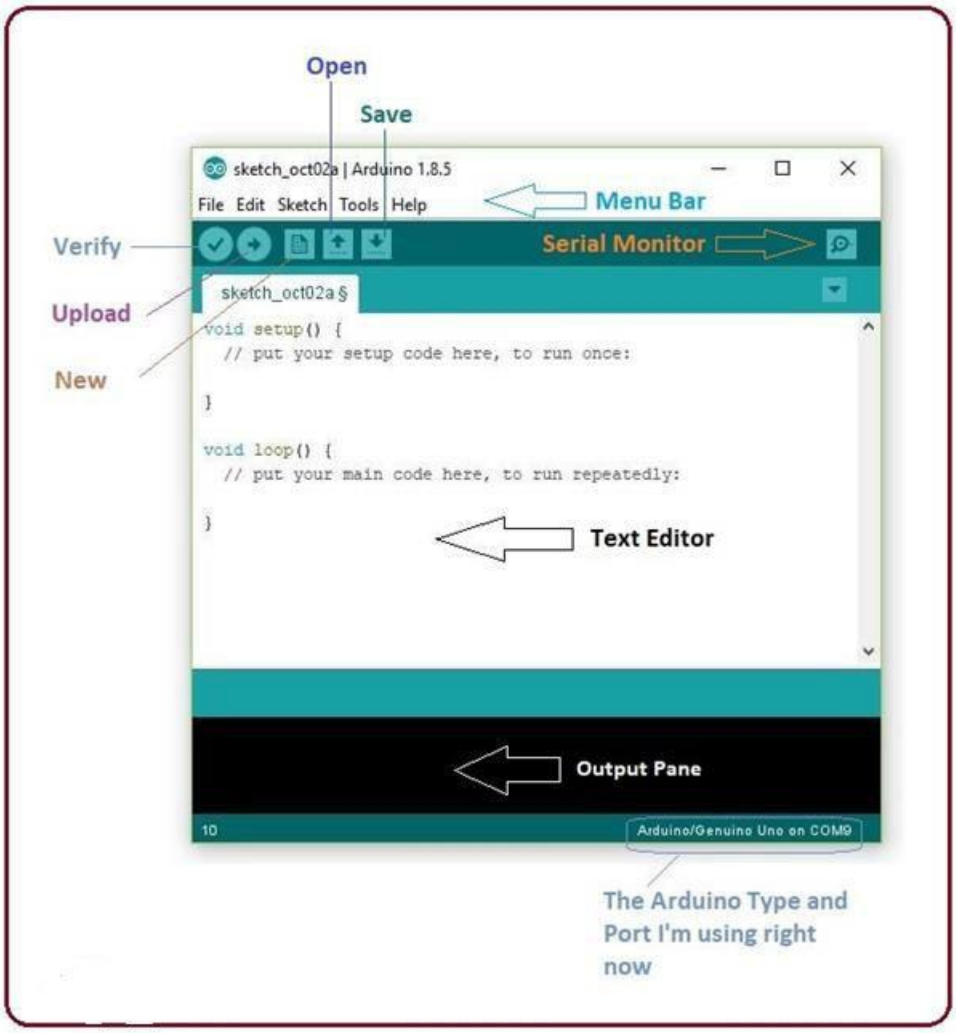
Introduction to Arduino IDE

- Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module.
- It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.
- It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.
- A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.
- Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.
- The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.
    - The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.
- This environment supports both C and C++ languages.

The IDE environment is mainly distributed into three sections

1. Menu Bar
2. Text Editor
3. Output Pane

As you download and open the IFIDE software, it will appear like an image below.

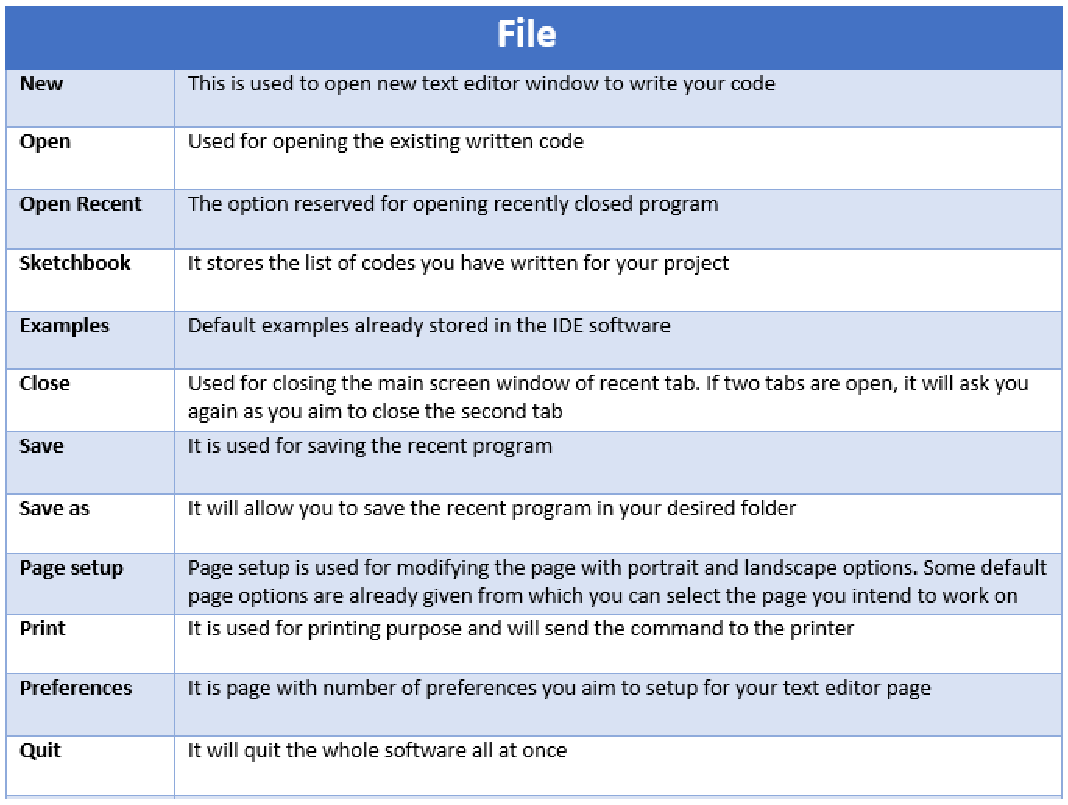

Open
Save
sketch_oct02a | Arduino 1.8.5
File Edit Sketch Tools Help
Menu Bar
Verify
Serial Monitor
Upload
sketch_oct02a §
New
Text Editor
Output Pane
10
Arduino/Genuino Uno on COM9
The Arduino Type and Port I'm using right now

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

}

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

}
```

Fig: Arduino IDE

The bar appearing on the top is called Menu Bar that comes with five different options as follow

- **File** – You can open a new window for writing the code or open an existing one. Following table shows the number of further subdivisions the file option is categorized into.

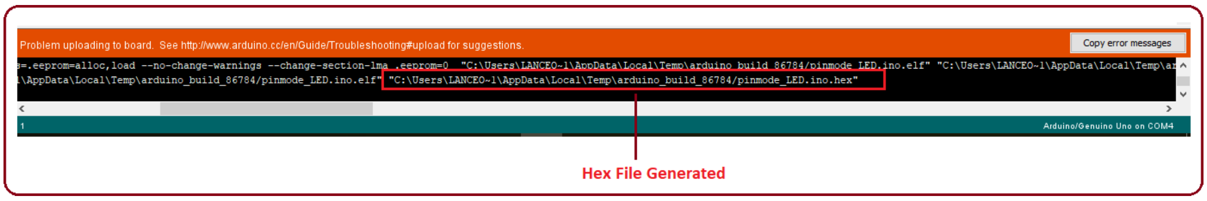


File	
New	This is used to open new text editor window to write your code
Open	Used for opening the existing written code
Open Recent	The option reserved for opening recently closed program
Sketchbook	It stores the list of codes you have written for your project
Examples	Default examples already stored in the IDE software
Close	Used for closing the main screen window of recent tab. If two tabs are open, it will ask you again as you aim to close the second tab
Save	It is used for saving the recent program
Save as	It will allow you to save the recent program in your desired folder
Page setup	Page setup is used for modifying the page with portrait and landscape options. Some default page options are already given from which you can select the page you intend to work on
Print	It is used for printing purpose and will send the command to the printer
Preferences	It is page with number of preferences you aim to setup for your text editor page
Quit	It will quit the whole software all at once

Fig: Classification of file

As you go to the preference section and check the compilation section, the Output Pane will show the code compilation as you click the upload button.

And at the end of compilation, it will show you the hex file it has generated for the recent sketch that will send to the Arduino Board for the specific task you aim to achieve.


Problem uploading to board. See http://www.arduino.cc/en/Guide/Troubleshooting#upload for suggestions.

```
s=.eeprom=alloc,load --no-change-warnings --change-section-lma .eeprom=0  "C:\Users\LANCEO~1\AppData\Local\Temp\arduino_build_86784/pinmode_LED.ino.elf" "C:\Users\LANCEO~1\AppData\Local\Temp\ar
1\AppData\Local\Temp\arduino_build_86784/pinmode_LED.ino.elf" "C:\Users\LANCEO~1\AppData\Local\Temp\arduino_build_86784/pinmode_LED.ino.hex"
```

Fig: Hex file being generated

- **Edit** – Used for copying and pasting the code with further modification for font

- **Sketch** – For compiling and programming

- **Tools** – Mainly used for testing projects. The Programmer section in this panel is used for burning a bootloader to the new microcontroller.

- **Help** – In case you are feeling sceptical about software, complete help is available from getting started to troubleshooting.

  The **Six Buttons** appearing under the Menu tab are connected with the running program as follow.

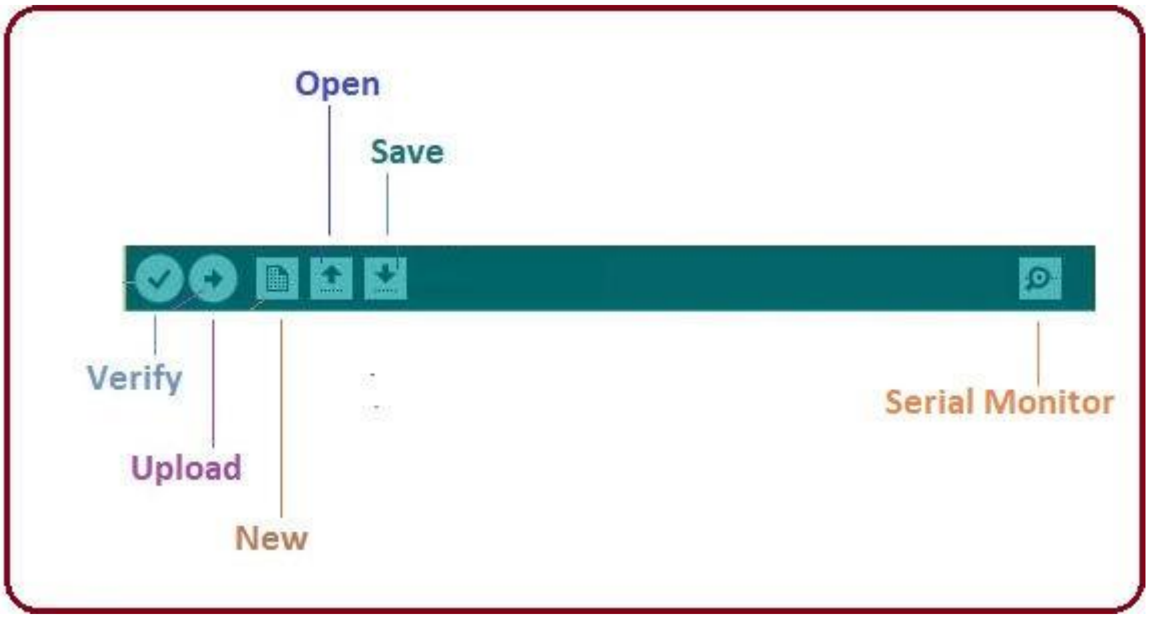
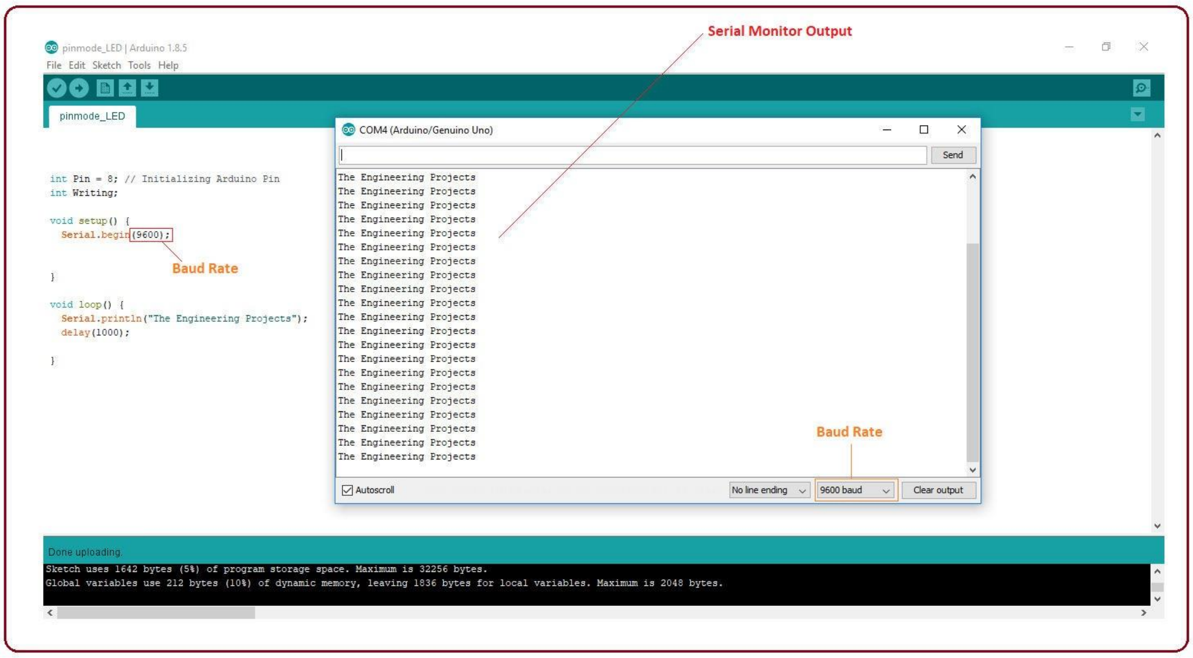


Figure 3.4 – Options for running program

- The check mark appearing in the circular button is used to verify the code. Click this once you have written your code.
- The arrow key will upload and transfer the required code to the Arduino board.
- The dotted paper is used for creating a new file.
- The upward arrow is reserved for opening an existing Arduino project.

- The downward arrow is used to save the current running code
- The button appearing on the top right corner is a **Serial Monitor** – A separate pop- up window that acts as an independent terminal and plays a vital role for sending and receiving the Serial Data. You can also go to the Tools panel and select Serial Monitor, or pressing Ctrl+Shift+M all at once will open it instantly. The Serial Monitor will actually help to debug the written Sketches where you can get a hold of how your program is operating. Your Arduino Module should be connected to your computer by USB cable in order to activate the Serial Monitor.
- You need to select the baud rate of the Arduino Board you are using right now. For my Arduino Uno Baud Rate is 9600, as you write the following code and click the Serial Monitor, the output will show as the image below.



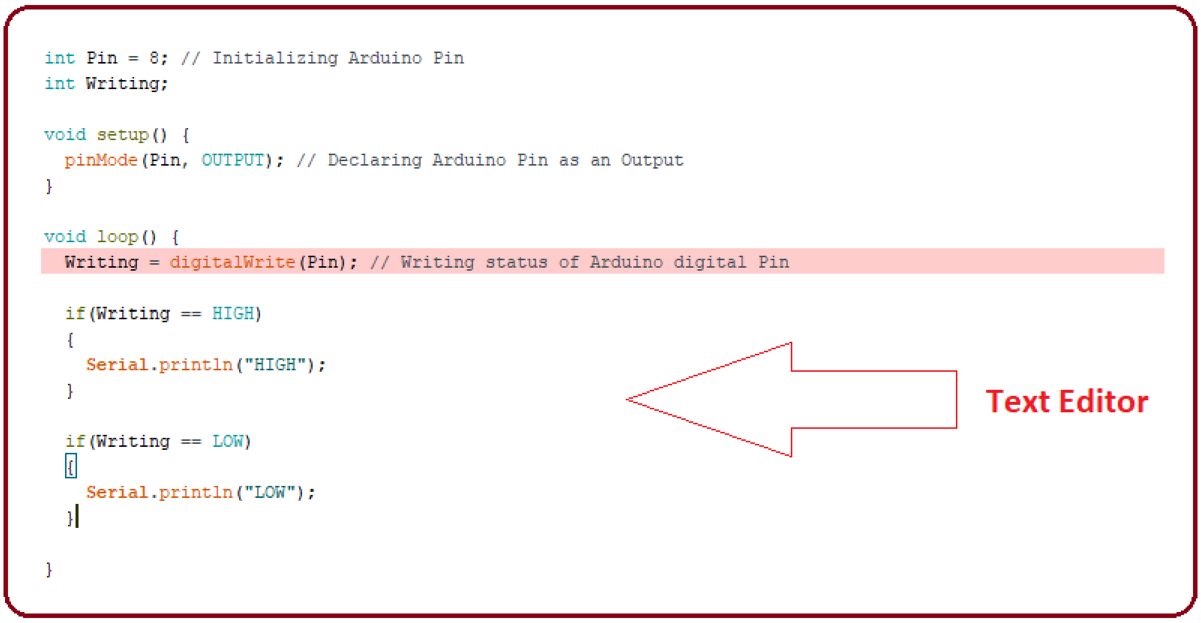
```
int Pin = 8; // Initializing Arduino Pin
int Writing;

void setup() {
  Serial.begin(9600);

}

void loop() {
  Serial.println("The Engineering Projects");
  delay(1000);

}
```

```
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
The Engineering Projects
```

```
Done uploading.
Sketch uses 1642 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 212 bytes (10%) of dynamic memory, leaving 1836 bytes for local variables. Maximum is 2048 bytes.
```

Figure 3.5 – Serial Monitor Output and Baud Rate selection

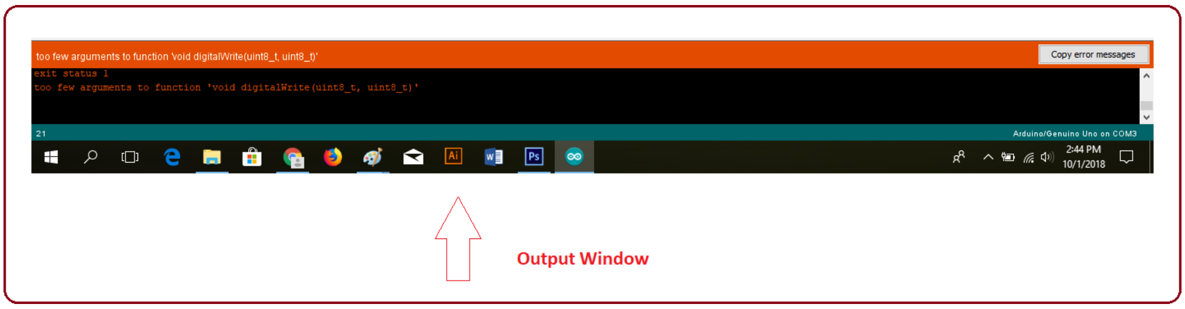
The main screen below the Menu bard is known as a simple text editor used for writing the required code.



```
int Pin = 8; // Initializing Arduino Pin
int Writing;

void setup() {
  pinMode(Pin, OUTPUT); // Declaring Arduino Pin as an Output
}

void loop() {
  Writing = digitalWrite(Pin); // Writing status of Arduino digital Pin

  if(Writing == HIGH)
  {
    Serial.println("HIGH");
  }

  if(Writing == LOW)
  {
    Serial.println("LOW");
  }

}
```

Figure 3.6 – Text Editor in Arduino IDE

The bottom of the main screen is described as an Output Pane that mainly highlights the compilation status of the running code: the memory used by the code, and errors occurred in the program. You need to fix those errors before you intend to upload the hex file into your Arduino Module.



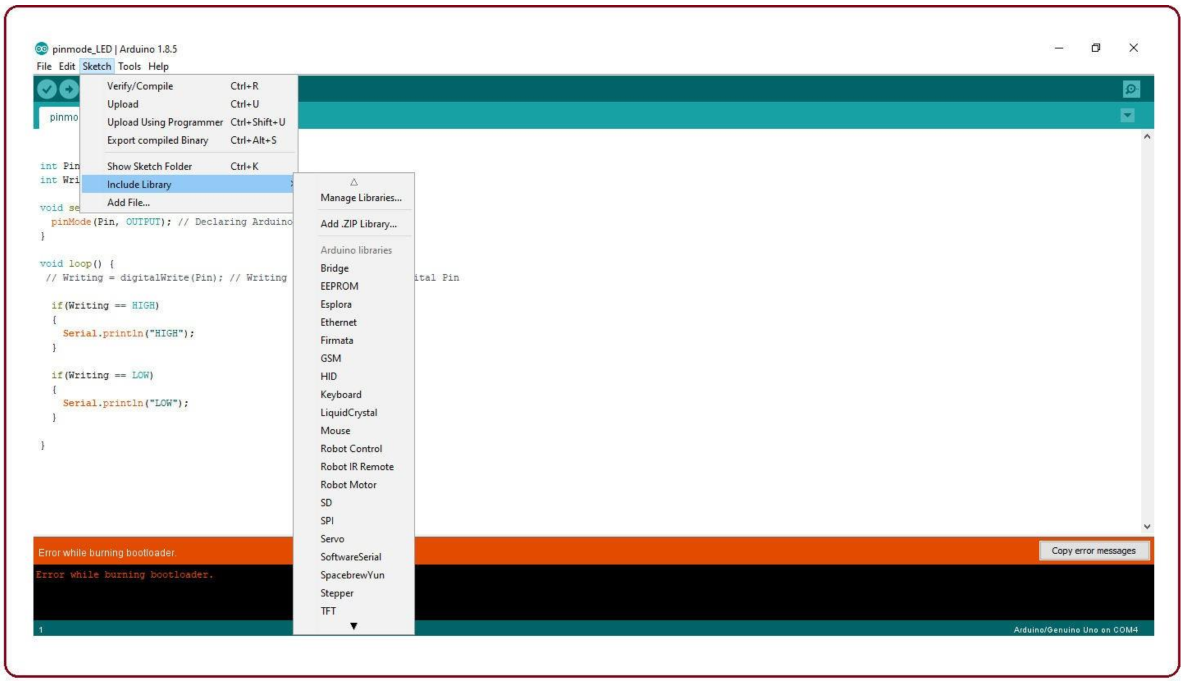
```
too few arguments to function 'void digitalWrite(uint8_t, uint8_t)'
exit status 1
too few arguments to function 'void digitalWrite(uint8_t, uint8_t)'
21                                                Arduino/Genuino Uno on COM3
```

Figure 3.7 – Output window

More or less, Arduino C language works similar to the regular C language used for any embedded system microcontroller, however, there are some dedicated libraries used for calling and executing specific functions on the board.

LIBRARIES

Libraries are very useful for adding the extra functionality into the Arduino Module. There is a list of libraries you can add by clicking the Sketch button in the menu bar and going to Include Library.



```
int Pin
int Wri

void se
  pinMode(Pin, OUTPUT); // Declaring Arduino
}

void loop() {
 // Writing = digitalWrite(Pin); // Writing                    ital Pin

  if(Writing == HIGH)
  {
    Serial.println("HIGH");
  }

  if(Writing == LOW)
  {
    Serial.println("LOW");
  }

}
```

```
Error while burning bootloader.
Error while burning bootloader.
```

Figure 3.8 – Libraries in Arduino IDE

As you click the Include Library and Add the respective library it will on the top of the sketch with a #include sign. Suppose, I Include the EEPROM library, it will appear on the text editor as

```
#include <EEPROM.h>
```

Most of the libraries are preinstalled and come with the Arduino software. However, you can also download them from the external sources.

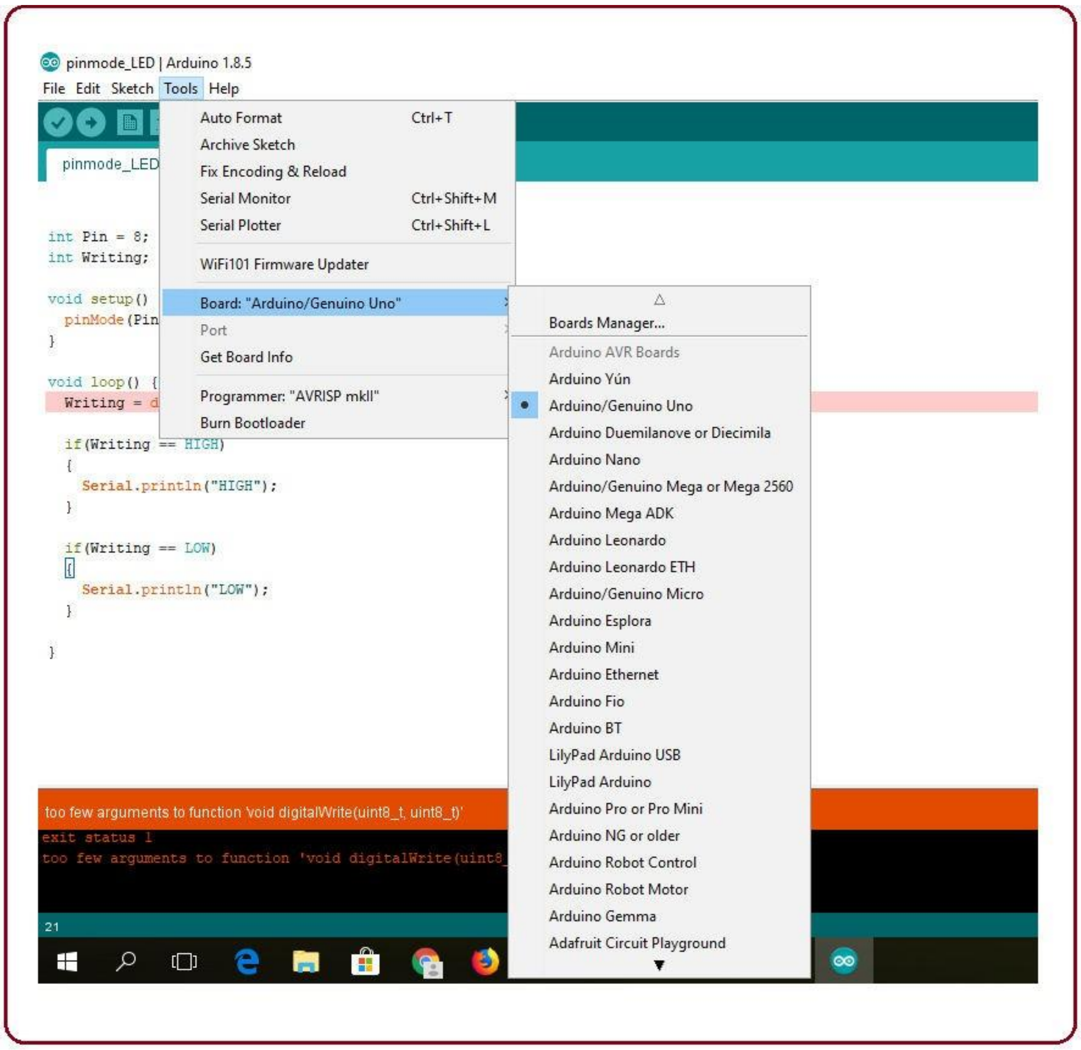
MAKING PINS AS INPUT OR OUTPUT

The digitalRead and digitalWrite commands are used for addressing and making the Arduino pins as an input and output respectively.

These commands are text sensitive i.e. you need to write them down the exact way they are given like digitalWrite starting with small “d” and write with capital “W”. Writing it down with Digitalwrite or digitalwrite won’t be calling or addressing any function.

HOW TO SELECT THE BOARD

- In order to upload the sketch, you need to select the relevant board you are using and the ports for that operating system. As you click the Tools on the Menu, it will open like the Figure below.
- Just go to the "Board" section and select the board you aim to work on. Similarly, COM1, COM2, COM4, COM5, COM7 or higher are reserved for the serial and USB board. You can look for the USB serial device in the ports section of the Windows Device Manager.
- Following Figure shows the COM4 that I have used for my project, indicating the Arduino Uno with COM4 port at the right bottom corner of the screen.

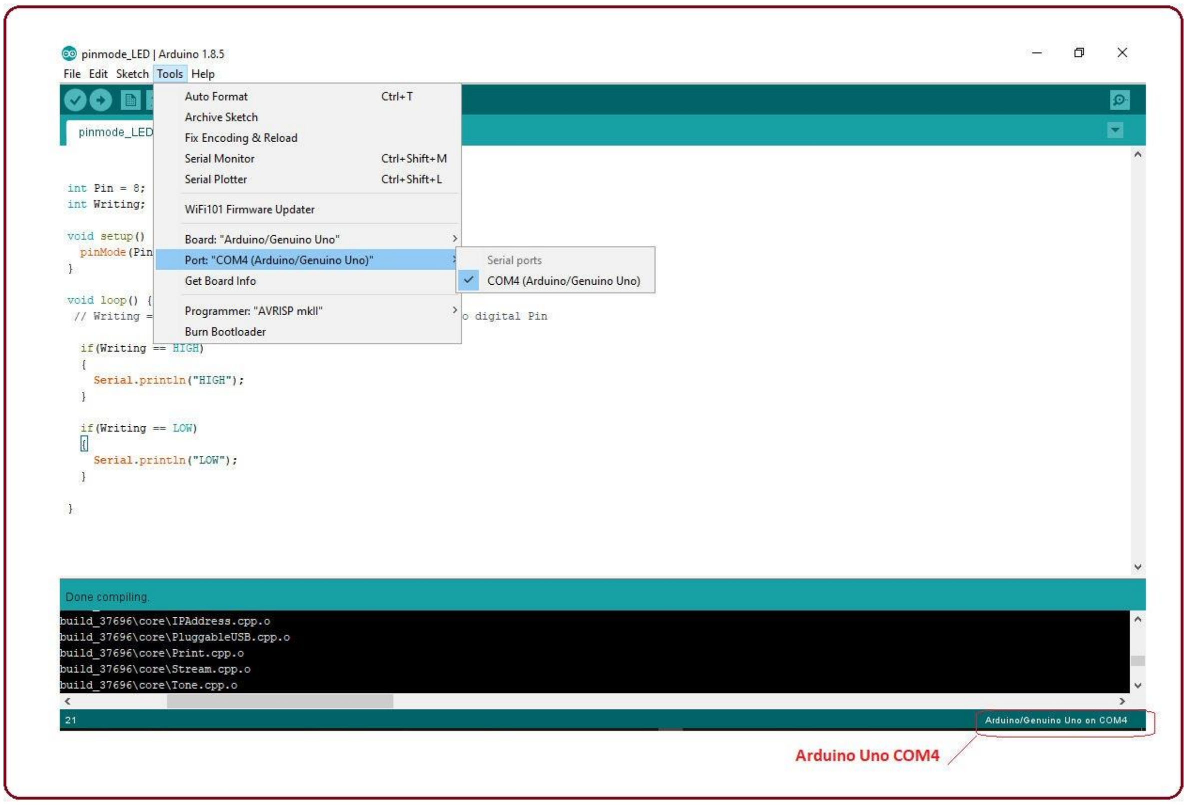


Board: "Arduino/Genuino Uno"
● Arduino/Genuino Uno

```
int Pin = 8;
int Writing;

void setup()
  pinMode(Pin
}

void loop() {
  Writing = d

  if(Writing == HIGH)
  {
    Serial.println("HIGH");
  }

  if(Writing == LOW)
  {
    Serial.println("LOW");
  }

}
```

```
too few arguments to function 'void digitalWrite(uint8_t, uint8_t)'
exit status 1
too few arguments to function 'void digitalWrite(uint8_
```

Figure 3.9 – Board selection


pinmode_LED | Arduino 1.8.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM4 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM4 (Arduino/Genuino Uno)
pinmode_LED
Arduino Uno COM4

```
int Pin = 8;
int Writing;

void setup()
  pinMode(Pin
}

void loop() {
 // Writing =                                  o digital Pin

  if(Writing == HIGH)
  {
    Serial.println("HIGH");
  }

  if(Writing == LOW)
  {
    Serial.println("LOW");
  }

}
```

```
Done compiling.
build_37696\core\IPAddress.cpp.o
build_37696\core\PluggableUSB.cpp.o
build_37696\core\Print.cpp.o
build_37696\core\Stream.cpp.o
build_37696\core\Tone.cpp.o
21                                                Arduino/Genuino Uno on COM4
```

Figure 3.10 – Port selection

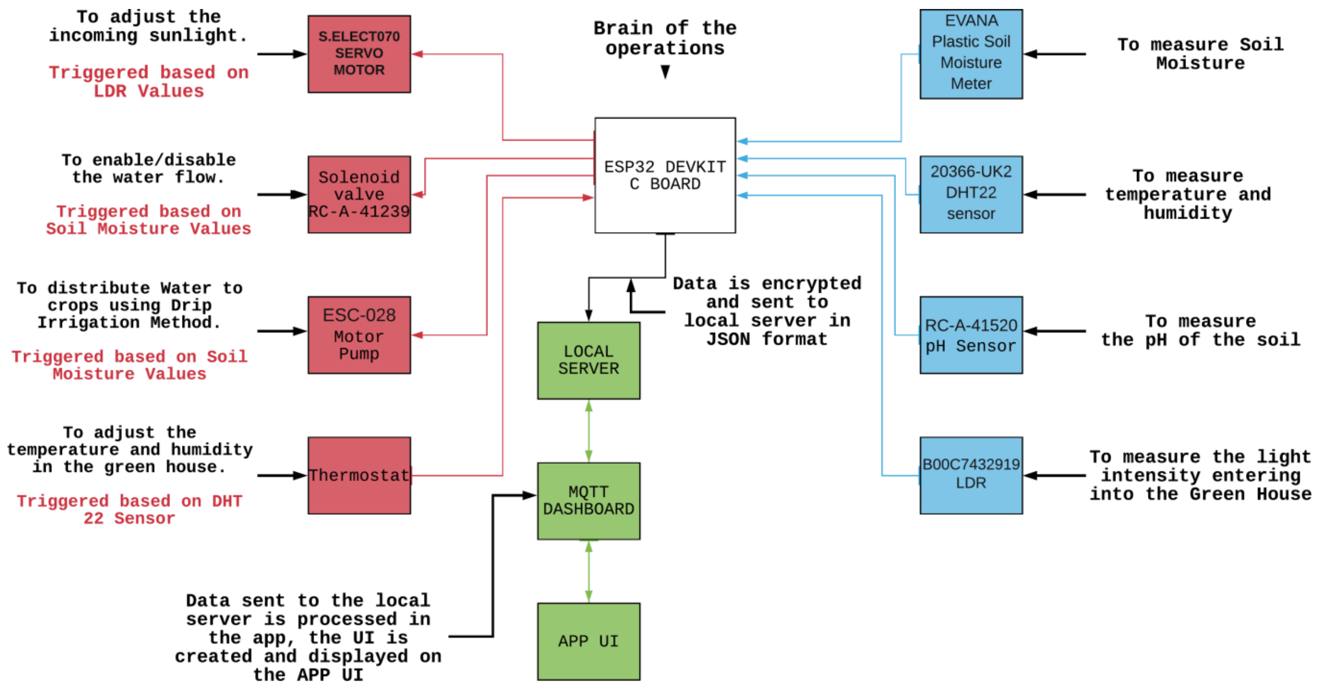
- After correct selection of both Board and Serial Port, click the verify and then upload button appearing in the upper left corner of the six button section or you can go to the Sketch section and press verify/compile and then upload.
- The sketch is written in the text editor and is then saved with the file extension .ino.

- Once you upload the code, TX and RX LEDs will blink on the board, indicating the desired program is running successfully.

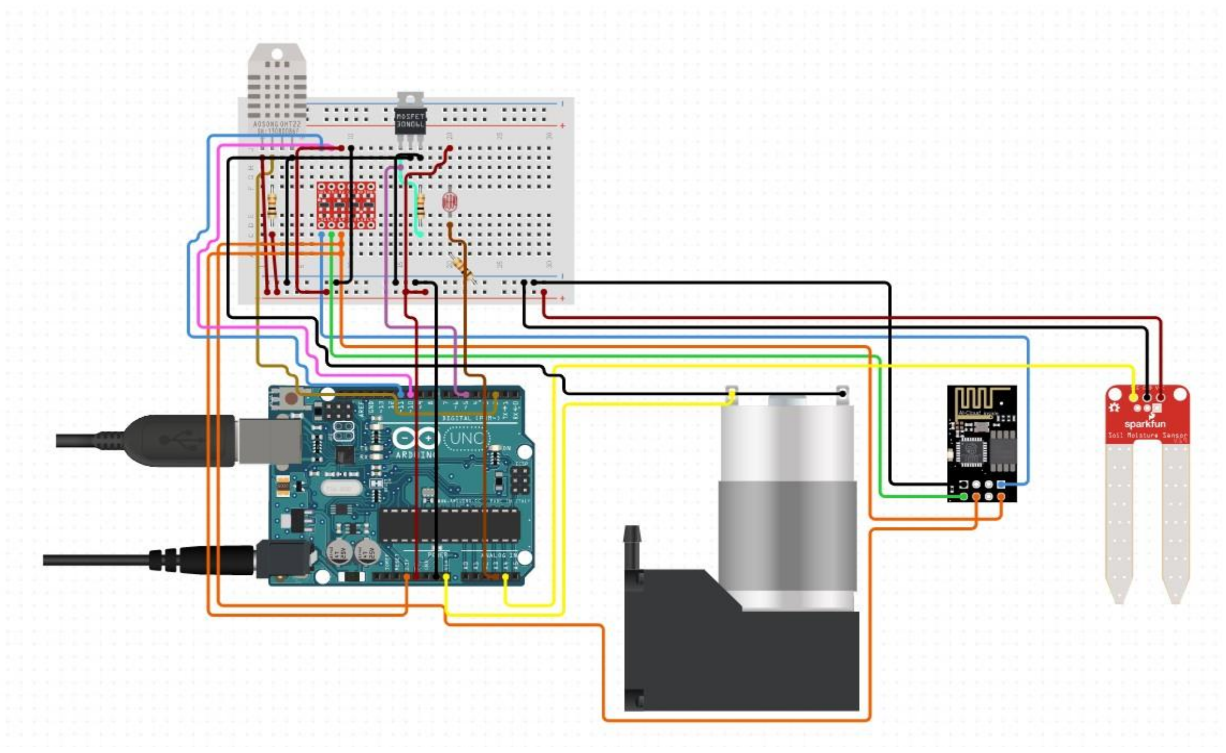
○ **Note**: The port selection criteria mentioned above is dedicated for Windows operating system only, you can check this Guide if you are using MAC or Linux.

The amazing thing about this software is that no prior arrangement or bulk of mess is required to install this software, you will be writing your first program within 2 minutes after the installation of the IDE environment.

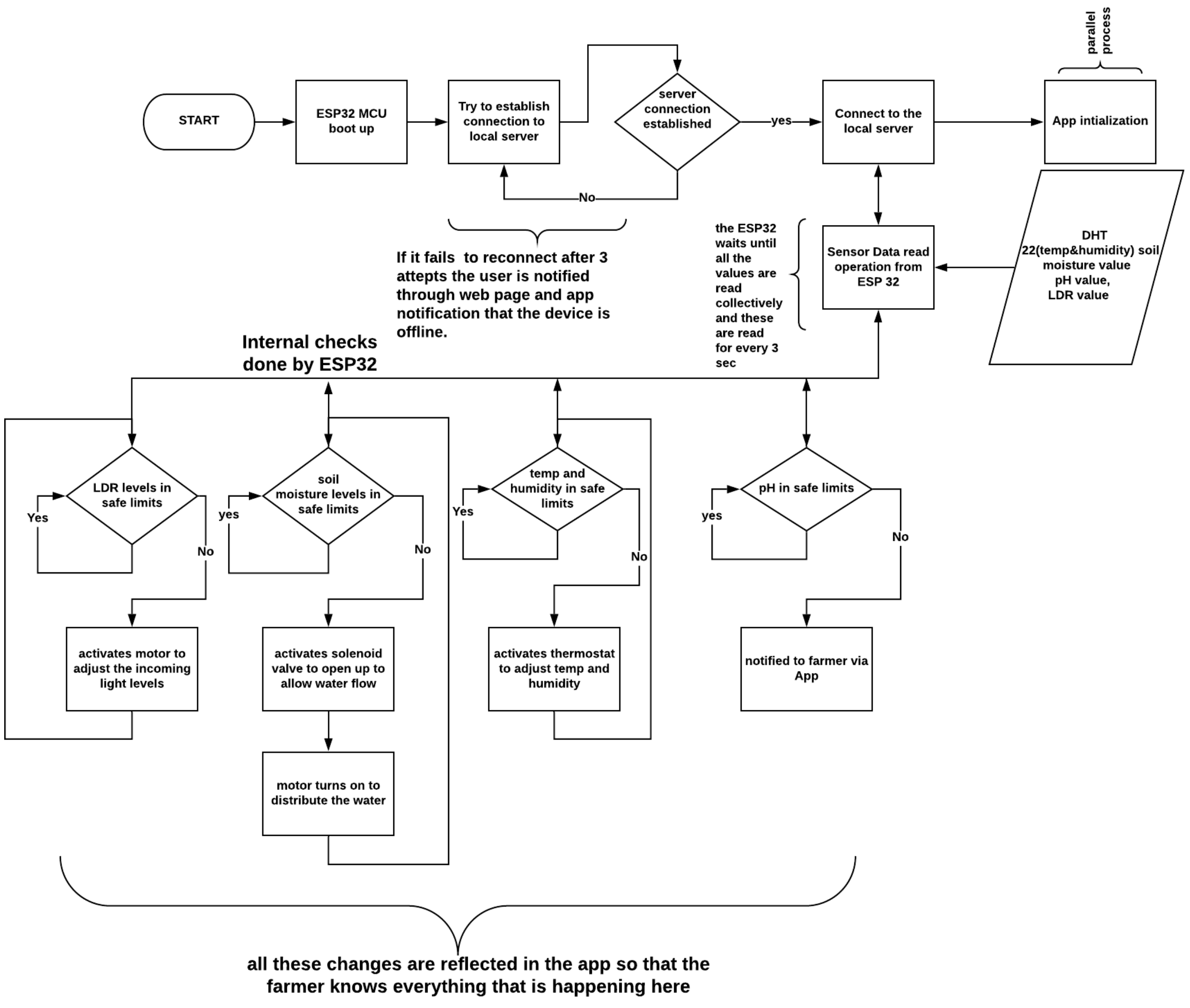
3.2 BLOCK DIAGRAM


To adjust the incoming sunlight.
Triggered based on LDR Values
S.ELECT070 SERVO MOTOR
To enable/disable the water flow.
Triggered based on Soil Moisture Values
Solenoid valve RC-A-41239
To distribute Water to crops using Drip Irrigation Method.
Triggered based on Soil Moisture Values
ESC-028 Motor Pump
To adjust the temperature and humidity in the green house.
Triggered based on DHT 22 Sensor
Thermostat
Brain of the operations
ESP32 DEVKIT C BOARD
Data is encrypted and sent to local server in JSON format
LOCAL SERVER
MQTT DASHBOARD
APP UI
Data sent to the local server is processed in the app, the UI is created and displayed on the APP UI
EVANA Plastic Soil Moisture Meter
To measure Soil Moisture
20366-UK2 DHT22 sensor
To measure temperature and humidity
RC-A-41520 pH Sensor
To measure the pH of the soil
B00C7432919 LDR
To measure the light intensity entering into the Green House

3.3 CIRCUIT DIAGRAM



3.4 FLOW CHART



START
ESP32 MCU boot up
Try to establish connection to local server
server connection established
yes
No
Connect to the local server
App intialization
parallel process
If it fails to reconnect after 3 attepts the user is notified through web page and app notification that the device is offline.
the ESP32 waits until all the values are read collectively and these are read for every 3 sec
Sensor Data read operation from ESP 32
DHT 22(temp&humidity) soil moisture value pH value, LDR value
Internal checks done by ESP32
LDR levels in safe limits
Yes
No
activates motor to adjust the incoming light levels
soil moisture levels in safe limits
yes
No
activates solenoid valve to open up to allow water flow
motor turns on to distribute the water
temp and humidity in safe limits
Yes
No
activates thermostat to adjust temp and humidity
pH in safe limits
yes
No
notified to farmer via App
all these changes are reflected in the app so that the farmer knows everything that is happening here

3.5 PROGRAMMING CODE

```
// Necessary Libraries
#include <AsyncMqttClient.h>
#include <WiFi.h>
extern "C" {
#include "freertos/FreeRTOS.h"
#include "freertos/timers.h"
}

// For Temperature and Humidity
#include <DHT.h>

// For Servo that is the window motor
#include <ESP32Servo.h>


// WiFi configuration
#define WIFI_SSID  "Router"
#define WIFI_PASSWORD  "aishwarya@007"



// MQTT configuration
// make sure it is an static IP
// If the IP is not static set it up as a static IP in the router configuration
// If you using a cloud MQTT broker type in the domain name
// #define MQTT_HOST "io.adafruit.com"
#define MQTT_HOST "192.168.31.169"
// This port is insecure(not encrypted) but if your running it locally it's okay if your wifi
is not hacked
```

```
// If you are using a cloud broker make sure you use an SSL connection.
//        https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-the-
mosquitto-mqtt-messaging-broker-on-ubuntu-16-04
#define MQTT_PORT 1883
// Setting up authentication for mosquitto broker
//       https://medium.com/@eranda/setting-up-authentication-on-mosquitto-mqtt-broker-
de5df2e29afc
#define MQTT_USERNAME "rohvis"
#define MQTT_PASS "incorrect007"


// MQTT Topics
// Sensor topics
#define MQTT_PUB_TEMP            "greenhouse/feeds/temperature"
#define MQTT_PUB_HUM             "greenhouse/feeds/humidity"
#define MQTT_PUB_LDR1            "greenhouse/feeds/ldr1"
#define MQTT_PUB_LDR2            "greenhouse/feeds/ldr2"
#define MQTT_PUB_SOIL_MOISTURE   "greenhouse/feeds/soil-moisture"
#define MQTT_PUB_PH_LEVEL        "greenhouse/feeds/pH-level"

// Threshold Settings topics
#define MQTT_SUB_TEMP_LOW_SET        "greenhouse/feeds/temperature-low-set"
#define MQTT_SUB_TEMP_HIGH_SET       "greenhouse/feeds/temperature-high-set"
#define MQTT_SUB_HUM_LOW_SET         "greenhouse/feeds/humidity-low-set"
#define MQTT_SUB_HUM_HIGH_SET        "greenhouse/feeds/humidity-high-set"
#define MQTT_SUB_LDR1_LOW_SET        "greenhouse/feeds/ldr1-low-set"
#define MQTT_SUB_LDR1_HIGH_SET       "greenhouse/feeds/ldr1-high-set"
#define MQTT_SUB_LDR2_LOW_SET        "greenhouse/feeds/ldr2-low-set"
#define MQTT_SUB_LDR2_HIGH_SET       "greenhouse/feeds/ldr2-high-set"
```

```
#define MQTT_SUB_PH_LEVEL_LOW_SET       "greenhouse/feeds/pH-level-low-set"
#define MQTT_SUB_PH_LEVEL_HIGH_SET      "greenhouse/feeds/pH-level-high-set"
#define MQTT_SUB_SOIL_MOISTURE_LOW_SET  "greenhouse/feeds/soil-moisture-
low-set"
#define MQTT_SUB_SOIL_MOISTURE_HIGH_SET "greenhouse/feeds/soil-moisture-
high-set"

// User Actions topics
#define     MQTT_SUB_WATER_PUMP_SWITCH     "greenhouse/feeds/water-pump-
switch"
#define MQTT_SUB_WINDOW_SWITCH     "greenhouse/feeds/window-switch"
#define MQTT_SUB_MANUAL_OVERRIDE   "greenhouse/feeds/manual-override"


// Feedback topics
#define MQTT_PUB_STATUS          "greenhouse/feeds/status"
#define MQTT_PUB_STATUS1         "greenhouse/feeds/status1"

// Board : https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/
// Pin Configuration for Sensors
#define DHTPIN      26
#define SOILMOISTURE 35
#define PH          17
#define LDR1        32
#define LDR2        33


// Pin Configuration for actuators
#define WATERPUMP    12
#define WINDOW_MOTOR 14
```

```
#define DHTTYPE DHT11
// create a dht object
DHT dht(DHTPIN, DHTTYPE);



// Initializing a MQTT client Object
AsyncMqttClient mqttClient;

// Creates a new software timer instance and returns a handle by which the timer can be referenced.
// Doc:https://www.freertos.org/FreeRTOS-timers-xTimerCreate.html
TimerHandle_t mqttReconnectTimer;
TimerHandle_t wifiReconnectTimer;

// Stores last time when data was published
unsigned long previousMillis = 0;
unsigned long previousSubMillis = 0;
// Interval at which to publish the sensor readings
const long interval = 5000;
const long subscribeInterval = 2000;



//  Variables to hold sensor readings
float temperature = 26;
float humidity = 62;
int soilMoisture = 55;
int ldrOne = 62, ldrTwo = 75;
float pH = 6;
```

```
// Variables to hold theresholds from the app
int tempLowThreshold = 25, tempHighThreshold = 40;
int humLowThreshold = 30, humHighThreshold = 70;
int soilMoistureLowThreshold = 50, soilMoistureHighThreshold = 65;
int ldrOneLowThreshold = 20, ldrOneHighThreshold = 95;
int ldrTwoLowThreshold = 20, ldrTwoHighThreshold = 95;
int pHLowThreshold = 5, pHHighThreshold = 12;

// setting ideal safe values
int tempSafe = (tempLowThreshold + tempHighThreshold)/2;
int humSafe = (humLowThreshold + humLowThreshold)/2;
int soilSafe = ( soilMoistureLowThreshold + soilMoistureHighThreshold)/2;
int ldrOneSafe =  (ldrOneLowThreshold + ldrOneHighThreshold)/2;
int ldrTwoSafe =  (ldrOneLowThreshold + ldrOneHighThreshold)/2;
int pHSafe = (pHLowThreshold+pHHighThreshold)/2;
boolean soilSafeState = false;

//User Actions
boolean manualOverride = false;

// first status
boolean initialStatus = true;

// Servo object for the window switch
Servo windowMotor;
```

```
void connectToWifi() {
  Serial.println("[ACTION] -- Connecting to Wi-Fi...");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
}

void connectToMqtt() {
  Serial.println("[ACTION] -- Connecting to MQTT...");
  mqttClient.connect();
}

void WiFiEvent(WiFiEvent_t event) {
  switch (event) {
    case SYSTEM_EVENT_STA_GOT_IP:
      Serial.println("[STATUS] -- WiFi connected");
      Serial.println("[INFO] -- IP address: ");
      Serial.println(WiFi.localIP());
      connectToMqtt();
      break;

    case SYSTEM_EVENT_STA_DISCONNECTED:
      Serial.println("[ALERT] -- WiFi connection lost!");
      // Ensure we don't connect to MQTT broker when there is no WiFi
      xTimerStop(mqttReconnectTimer, 0);
      xTimerStart(wifiReconnectTimer, 0);
      break;
  }
}


void onMqttConnect(bool sessionPresent) {
  Serial.println("[STATUS] -- Connected to MQTT.");
  Serial.print("[STATUS] -- Session Present:");
  Serial.println(sessionPresent);

  // When we connect to the MQTT we subscribe to topics

  uint16_t packetIdSub1 = mqttClient.subscribe(MQTT_SUB_TEMP_LOW_SET, 1);

  uint16_t packetIdSub2 = mqttClient.subscribe(MQTT_SUB_TEMP_HIGH_SET, 1);

  uint16_t packetIdSub3 = mqttClient.subscribe(MQTT_SUB_HUM_LOW_SET, 1);

  uint16_t packetIdSub4 = mqttClient.subscribe(MQTT_SUB_HUM_HIGH_SET, 1);

  uint16_t                    packetIdSub5                    =
mqttClient.subscribe(MQTT_SUB_SOIL_MOISTURE_LOW_SET, 1);

  uint16_t                    packetIdSub6                    =
mqttClient.subscribe(MQTT_SUB_SOIL_MOISTURE_HIGH_SET, 1);

  uint16_t packetIdSub7 = mqttClient.subscribe(MQTT_SUB_LDR1_LOW_SET, 1);

  uint16_t packetIdSub8 = mqttClient.subscribe(MQTT_SUB_LDR1_HIGH_SET, 1);

  uint16_t packetIdSub9 = mqttClient.subscribe(MQTT_SUB_LDR2_LOW_SET, 1);

  uint16_t packetIdSub10 = mqttClient.subscribe(MQTT_SUB_LDR2_HIGH_SET, 1);

  uint16_t                    packetIdSub11                   =
mqttClient.subscribe(MQTT_SUB_PH_LEVEL_LOW_SET, 1);
```

```
  uint16_t                                    packetIdSub12                                    =
mqttClient.subscribe(MQTT_SUB_PH_LEVEL_HIGH_SET, 1);

  uint16_t                                    packetIdSub13                                    =
mqttClient.subscribe(MQTT_SUB_MANUAL_OVERRIDE, 1);

  uint16_t                                    packetIdSub14                                    =
mqttClient.subscribe(MQTT_SUB_WATER_PUMP_SWITCH, 1);

  uint16_t  packetIdSub15  =  mqttClient.subscribe(MQTT_SUB_WINDOW_SWITCH,
1);



}


void onMqttDisconnect(AsyncMqttClientDisconnectReason reason) {
  Serial.println("[ALERT] -- Disconnected from MQTT.");
  if (WiFi.isConnected()) {
    // If the connection is lost to the MQTT.We check if the WiFi connection is present or
not
    // Then we re-attempt to connect to the MQTT
    xTimerStart(mqttReconnectTimer, 0);
  }
}


void onMqttSubscribe(uint16_t packetId, uint8_t qos) {
  Serial.println("[STATUS] -- Subscribe acknowledged.");
  Serial.print("[INFO] -- packet ID:");
  Serial.print(packetId);
  Serial.print("[INFO] QOS: ");
  Serial.println(qos);

}


void onMqttPublish(uint16_t packetId) {
  Serial.printf("[INFO] -- Data Published with Packet ID:%d\n", packetId);
}


String valueExtract(char* payload, size_t len) {
  String temp = "";
  for (int i = 0; i <= len; i++) {
    temp += String((char)payload[i]);
  }
  Serial.println(temp);
  return temp;
}

void sendStatusFeedback(String msg, int channel) {
  if (channel == 1)
    mqttClient.publish(MQTT_PUB_STATUS, 1, true, String(msg).c_str());
  if (channel == 2)
    mqttClient.publish(MQTT_PUB_STATUS1, 1, true, String(msg).c_str());
}

void onMqttMessage(char* topic, char* payload, AsyncMqttClientMessageProperties
properties, size_t len, size_t index, size_t total) {
```

```
Serial.println("[STATUS] -- Publish received.");
Serial.print("[INFO] -- topic: ");
Serial.println(topic);
String value = valueExtract(payload, len);
Serial.print(String(topic) + ":" + String(value));

if (String(topic) == MQTT_SUB_TEMP_LOW_SET)
  tempLowThreshold = value.toInt();

if (String(topic) == MQTT_SUB_TEMP_HIGH_SET)
  tempHighThreshold = value.toInt();

if (String(topic) == MQTT_SUB_HUM_LOW_SET)
  humLowThreshold = value.toInt();

if (String(topic) == MQTT_SUB_HUM_HIGH_SET)
  humHighThreshold = value.toInt();

if (String(topic) == MQTT_SUB_SOIL_MOISTURE_LOW_SET)
  soilMoistureLowThreshold = value.toInt();

if (String(topic) == MQTT_SUB_SOIL_MOISTURE_HIGH_SET)
  soilMoistureHighThreshold = value.toInt();

if (String(topic) == MQTT_SUB_LDR1_LOW_SET)
  ldrOneLowThreshold = value.toInt();

if (String(topic) == MQTT_SUB_LDR1_HIGH_SET)
  ldrOneHighThreshold = value.toInt();
```

```
if (String(topic) == MQTT_SUB_LDR2_LOW_SET)
  ldrTwoLowThreshold = value.toInt();

if (String(topic) == MQTT_SUB_LDR2_HIGH_SET)
  ldrTwoHighThreshold = value.toInt();

if (String(topic) == MQTT_SUB_PH_LEVEL_LOW_SET)
  pHLowThreshold = value.toInt();

if (String(topic) == MQTT_SUB_PH_LEVEL_HIGH_SET)
  pHHighThreshold = value.toInt();

if (String(topic) == MQTT_SUB_MANUAL_OVERRIDE) {
  if (value[0] == 't') {
    manualOverride = true;
    Serial.printf("[INFO] -- Manual Ovveride ON");
    sendStatusFeedback("Manual Override ON!", 1);
  }
  else {
    manualOverride = false;
    Serial.printf("[INFO] -- Manual Ovveride OFF");
    sendStatusFeedback("Manual Override OFF!", 1);
  }
}

if (String(topic) == MQTT_SUB_WATER_PUMP_SWITCH) {
  if (value[0] == 't' && manualOverride) {
    Serial.printf("[ACTION] -- MOTOR ON");
    sendStatusFeedback("MOTOR ON", 2);
    waterPump(true);
  }
  else if (value[0] == 'f' && manualOverride) {
    Serial.printf("[ACTION] -- MOTOR ON");
    sendStatusFeedback("MOTOR OFF", 2);
    waterPump(false);
  }
  else {
    Serial.printf("[ALERT] -- Cannot execute action! Manual Ovveride is OFF");
    sendStatusFeedback("Cannot execute action! Manual Override OFF!", 1);
  }
}

if (String(topic) == MQTT_SUB_WINDOW_SWITCH) {
  if (value[0] == 't' && manualOverride) {
    sendStatusFeedback("Opening windows", 2);
    Serial.printf("[ACTION] -- OPENING WINDOWS");
    windows(true);
  }
  else if (value[0] == 'f' && manualOverride) {
    sendStatusFeedback("Closing windows", 2);
     Serial.printf("[ACTION] -- CLOSING WINDOWS");
    windows(false);
  }
  else {
    Serial.printf("[ALERT] -- Cannot execute action! Manual Ovveride is OFF");
    sendStatusFeedback("Cannot execute action! Manual Override OFF!", 1);
  }
}
```

```
}
void setSafeValues(){
  //ideal mid values
tempSafe = (tempLowThreshold + tempHighThreshold)/2;
humSafe = (humLowThreshold + humLowThreshold)/2;
soilSafe = ( soilMoistureLowThreshold + soilMoistureHighThreshold)/2;
ldrOneSafe =  (ldrOneLowThreshold + ldrOneHighThreshold)/2;
ldrTwoSafe =  (ldrOneLowThreshold + ldrOneHighThreshold)/2;
pHSafe = (pHLowThreshold+pHHighThreshold)/2;
}

void getSensorData(boolean randomMode) {

  if (randomMode) {
    temperature = random(25, 31);
    humidity = random(60, 96);
    pH = random(5, 15);
    ldrOne = random(65, 100);
    ldrTwo = random(65, 100);
    soilMoisture = random(60, 96);

  }
  else {
    delay(500);
    temperature = dht.readTemperature();
    delay(500);
    humidity = dht.readHumidity();
    delay(500);


    if (isnan(temperature) || isnan(humidity))
      Serial.println(F("[ALERT] -- Failed to read from DHT sensor!"));

    int ldrOneRead = analogRead(LDR1);
    delay(500);
    ldrOne = map(ldrOneRead, 0, 4095, 1, 100);
    delay(500);


    int ldrTwoRead = analogRead(LDR2);
    delay(500);
    ldrTwo = map(ldrTwoRead, 0, 4095, 1, 100);
    delay(500);



    int soilRead = analogRead(SOILMOISTURE);
    soilMoisture = 100 - map(soilRead, 0, 4095, 1, 100);
    delay(500);


    pH = random(4, 15);


  }
}
```

```
void thresholdChecks() {
  if (initialStatus) {
    sendStatusFeedback("ALL OK!", 2);
    initialStatus = false;
  }
  if (temperature < tempLowThreshold) {
    Serial.println("[ALERT] -- Temperature below Lower Threshold!");
    sendStatusFeedback("[ALERT] -- Temperature below Lower Threshold", 1);
    if (!manualOverride) {
      thermostat(true, tempLowThreshold + 2);
    }
  }
  if (temperature > tempHighThreshold) {
    Serial.println("[ALERT] -- Temperature above Higher Threshold!");
    sendStatusFeedback("Temperature above Higher Threshold", 1);
    if (!manualOverride) {
      waterPump(true);
    }
  }
  if(temperature > tempSafe -5 && temperature < tempSafe+5){
    Serial.println("[INFO] -- Temperature in safe levels!");
    sendStatusFeedback("Temperature in safe levels!", 1);
  }
  delay(500);
  if (humidity < humLowThreshold ) {
    Serial.println("[ALERT] -- Humidity below Lower threshold!");
    sendStatusFeedback("Humidity below Lower threshold!", 1);
    if (!manualOverride) {
```

```
    waterPump(true);
  }
}
if (humidity > humHighThreshold ) {
  Serial.println("[ALERT] -- Humidity above Higher threshold!");
  sendStatusFeedback("Humidity above Higher threshold!", 1);
  if (!manualOverride) {
    thermostat(true, tempLowThreshold + 4);
  }
}
if(humidity > humSafe -5 && humidity < humSafe+5){
  Serial.println("[INFO] -- humidity in safe levels!");
  sendStatusFeedback("Humidity in safe levels!", 1);
}
delay(500);
if (ldrOne < ldrOneLowThreshold) {
  Serial.println("[ALERT] -- LDR ONE below Lower threshold!");
  sendStatusFeedback("LDR ONE below Lower threshold!", 1);
  if (!manualOverride) {
    windows(true);
  }
}
if (ldrOne > ldrOneHighThreshold) {
  Serial.println("[ALERT] -- LDR ONE above higher threshold!");
  sendStatusFeedback("LDR ONE above higher threshold!", 1);
  if (!manualOverride) {
    windows(false);
  }
}
```

```
delay(500);
if (ldrTwo < ldrTwoLowThreshold) {
  Serial.println("[ALERT] -- LDR TWO below Lower threshold!");
  sendStatusFeedback("LDR TWO below Lower threshold!", 1);
  if (!manualOverride) {
    windows(true);
  }
}
if (ldrTwo > ldrTwoHighThreshold) {
  Serial.println("[ALERT] -- LDR TWO above higher threshold!");
  sendStatusFeedback("LDR TWO above higher threshold!", 1);
  if (!manualOverride) {
    windows(false);
  }
}
delay(500);
if (soilMoisture < soilMoistureLowThreshold ) {
  soilSafeState = false;
  Serial.println("[ALERT] -- Soil Moisture  below lower threshold!");
  sendStatusFeedback("Soil Moisture below lower threshold!", 1);
  if (!manualOverride) {
    waterPump(true);
  }
}
if (soilMoisture > soilMoistureHighThreshold ) {
  soilSafeState = false;
  Serial.println("[ALERT] -- Soil Moisture  above higher threshold!");
  sendStatusFeedback("Soil Moisture above higher threshold!", 1);
  if (!manualOverride) {
      waterPump(false);
    }
  }
  delay(500);
   if(soilMoisture > soilSafe -5 && soilMoisture < soilSafe+5){
    soilSafeState = true;
    Serial.println("[INFO] -- Soil Moisture  in safe levels!");
    sendStatusFeedback("soil Moisture in safe levels!", 1);
    delay(500);
    if(soilSafeState && (!manualOverride)) waterPump(false);
  }
  if (pH < pHLowThreshold ) {
    Serial.println("[ALERT] -- pH below Lower threshold!");
    sendStatusFeedback("pH below Lower threshold!", 1);
    sendStatusFeedback(" ", 2);
  }
  if (pH > pHHighThreshold ) {
    Serial.println("[ALERT] -- pH above Higher threshold!");
    sendStatusFeedback("pH above Higher threshold!", 1);
    sendStatusFeedback(" ", 2);
  }
  if(pH > pHSafe -5 && pH < pHSafe+5){
    Serial.println("[INFO] -- PHin safe levels!");
    sendStatusFeedback("pH in safe levels!", 1);
    sendStatusFeedback(" ", 2);
  }
  delay(1000);
}
```

```
void setup()
{
  pinMode(WATERPUMP, OUTPUT);
  pinMode(LDR1, INPUT);
  pinMode(WINDOW_MOTOR, OUTPUT);
  // Initialize the Serial Monitor with a baud rate
  Serial.begin(115200);

  // Attach the pin to the servo object
  windowMotor.attach(WINDOW_MOTOR);

  // Initialize the DHT Sensor
  dht.begin();
  // if analog input pin 11 is unconnected, random analog
  // noise will cause the call to randomSeed() to generate
  // different seed numbers each time the sketch runs.
  // randomSeed() will then shuffle the random function.
  // https://www.arduino.cc/reference/en/language/functions/random-numbers/random/
  randomSeed(analogRead(11));
  // The below timers are for reconnecting to the WiFi router and MQTT broker
  mqttReconnectTimer = xTimerCreate("mqttTimer", pdMS_TO_TICKS(2000),
pdFALSE, (void*)0, reinterpret_cast<TimerCallbackFunction_t>(connectToMqtt));
  wifiReconnectTimer = xTimerCreate("wifiTimer", pdMS_TO_TICKS(2000),
pdFALSE, (void*)0, reinterpret_cast<TimerCallbackFunction_t>(connectToWifi));
  // https://techtutorialsx.com/2019/08/11/esp32-arduino-getting-started-with-wifi-
events/
  WiFi.onEvent(WiFiEvent);
  // http://marvinroger.viewdocs.io/async-mqtt-client/2.-API-reference/
  mqttClient.onConnect(onMqttConnect);
  mqttClient.onDisconnect(onMqttDisconnect);
  mqttClient.onSubscribe(onMqttSubscribe);
  mqttClient.onPublish(onMqttPublish);
  mqttClient.setServer(MQTT_HOST, MQTT_PORT);
  mqttClient.onMessage(onMqttMessage);
  // If your broker requires authentication.
  mqttClient.setCredentials(MQTT_USERNAME, MQTT_PASS);
  connectToWifi();
  sendStatusFeedback("ALL OK!", 2);

}

void loop() {

  unsigned long currentMillis = millis();
  // Every X number of seconds (interval = 10 seconds)
  // it publishes a new MQTT message
  if (currentMillis - previousMillis >= 2000) {
    //Save the last time a new message was sent.
    previousMillis = currentMillis;
    getSensorData(false);
    Serial.printf("[ACTION] -- Publishing temperature\n");
    Serial.printf("[INFO] -- Temperature:%d\n", temperature);
    mqttClient.publish(MQTT_PUB_TEMP, 1, true, String(temperature).c_str());
    delay(500);

    Serial.printf("[ACTION] -- Publishing humidity\n");
    Serial.printf("[INFO] -- Humidity:%d\n", humidity);
    mqttClient.publish(MQTT_PUB_HUM, 1, true, String(humidity).c_str());
    delay(500);

    Serial.printf("[ACTION] -- Publishing soil moisture\n");
    Serial.printf("[INFO] -- Soil Moisture:%d\n", soilMoisture);
    mqttClient.publish(MQTT_PUB_SOIL_MOISTURE, 1, true,
String(soilMoisture).c_str());
    delay(500);

    Serial.printf("[ACTION] -- Publishing ldr one\n");
    Serial.printf("[INFO] -- LDR ONE:%d\n", ldrOne);
    mqttClient.publish(MQTT_PUB_LDR1, 1, true, String(ldrOne).c_str());
    delay(500);

    Serial.printf("[ACTION] -- Publishing ldr two\n");
    Serial.printf("[INFO] -- LDR TWO:%d\n", ldrTwo);
    mqttClient.publish(MQTT_PUB_LDR2, 1, true, String(ldrTwo).c_str());
    delay(500);

    Serial.printf("[ACTION] -- Publishing pH\n");
    Serial.printf("[INFO] -- pH:%d\n", pH);
    mqttClient.publish(MQTT_PUB_PH_LEVEL, 1, true, String(pH).c_str());
    delay(500);
  }
  delay(3000);
  setSafeValues();
  thresholdChecks();
}
```

```
void waterPump(int state) {
  if (state) {
    Serial.println("[ACTION] -- MOTOR ON");
    digitalWrite(WATERPUMP, HIGH);
    sendStatusFeedback("MOTOR ON", 2);
    delay(2000);
  }
  else {
    Serial.println("[ACTION] -- MOTOR OFF");
    digitalWrite(WATERPUMP, LOW);
    sendStatusFeedback("MOTOR OFF", 2);
    delay(2000);

  }
}


void windows(int state) {
  if (state) {
    Serial.println("[ACTION] -- OPENING WINDOWS");
    delay(2000);
    windowMotor.write(180);
    sendStatusFeedback("OPENING WINDOWS", 2);
    delay(2000);

  }
  else {
    Serial.println("[ACTION] -- CLOSING WINDOWS");
    delay(2000);
    windowMotor.write(0);
    sendStatusFeedback("CLOSING WINDOWS", 2);
    delay(2000);

  }

}


void thermostat(int state, int temp) {
  if (state) {
    Serial.println("[ACTION] -- TURNING ON THERMOSTAT");
    Serial.printf("Setting temperature to:%d", temp);
    sendStatusFeedback("Setting thermostat to " + String(temp) + "°C", 2);
  }
  else {
    Serial.println("[ACTION] -- TURNING OF THERMOSTAT");
    sendStatusFeedback("Thermostat off", 2);
  }
            }
```

3.6 ADVANTAGES

- Increased quality of production.
- Operation cost is low.
- Reduced environmental footprint.
- Accurate farm and field evaluation.

3.7 DISADVANTAGES

- It requires continuous internet connection.
- People with lack of knowledge cannot handle the system.

3.8 APPLICATIONS

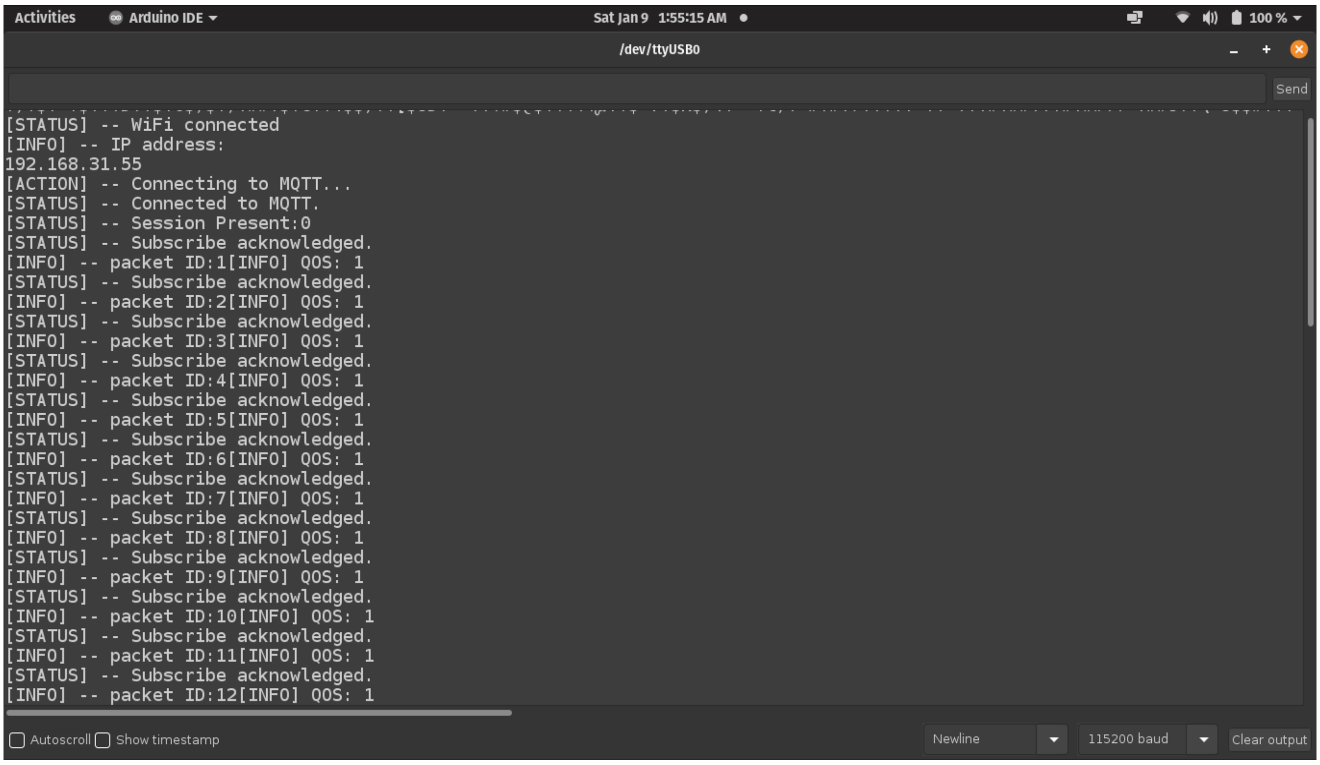
CHAPTER 4
OPERATING PROCEDURE

- Initiate the MQTT server and then connect the esp32 to a power supply.
- Initiate the app to connect it to the MQTT broker
- The ESP32 will read the values from the sensors.
- The sensor data is pushed to the MQTT server and this will serve to the app which had subscribed to the topics.
- The esp32 will monitor the sensor values according to the thresholds set in the app
- Suitable action is taken according to the situation
- Notifications are sent to alert the user.
- Manual Override mode can be selected in order to override system functions that are coded for particular situation. Here the user can specify the action he/s

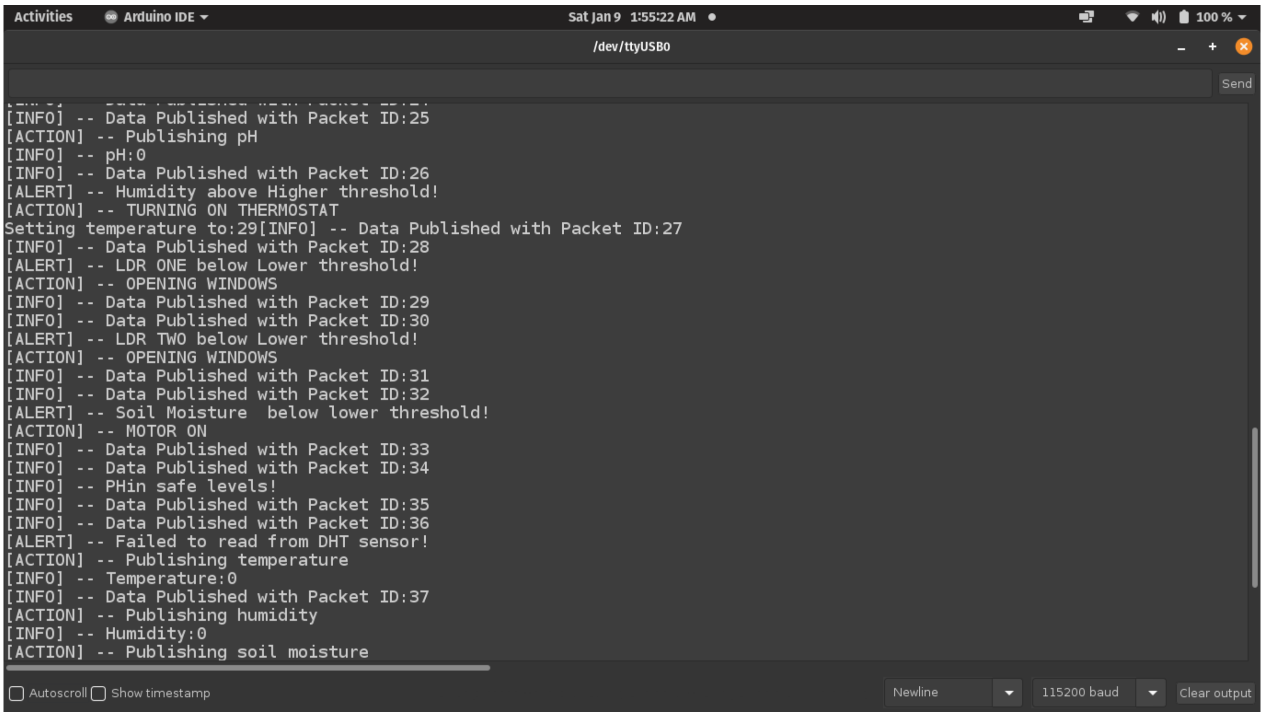
CHAPTER 5

RESULT

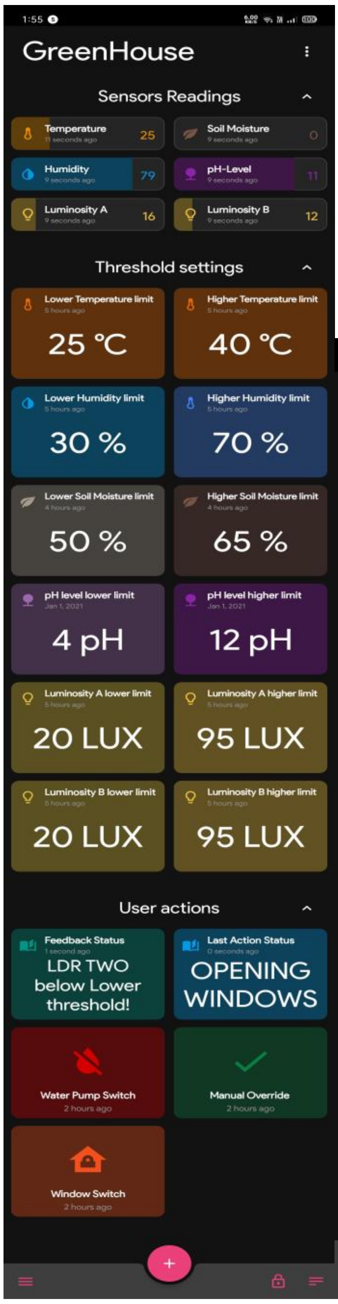
Some Screenshots from the terminal and app:



```
[STATUS] -- WiFi connected
[INFO] -- IP address:
192.168.31.55
[ACTION] -- Connecting to MQTT...
[STATUS] -- Connected to MQTT.
[STATUS] -- Session Present:0
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:1[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:2[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:3[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:4[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:5[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:6[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:7[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:8[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:9[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:10[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:11[INFO] QOS: 1
[STATUS] -- Subscribe acknowledged.
[INFO] -- packet ID:12[INFO] QOS: 1
```



```
[INFO] -- Data Published with Packet ID:25
[ACTION] -- Publishing pH
[INFO] -- pH:0
[INFO] -- Data Published with Packet ID:26
[ALERT] -- Humidity above Higher threshold!
[ACTION] -- TURNING ON THERMOSTAT
Setting temperature to:29[INFO] -- Data Published with Packet ID:27
[INFO] -- Data Published with Packet ID:28
[ALERT] -- LDR ONE below Lower threshold!
[ACTION] -- OPENING WINDOWS
[INFO] -- Data Published with Packet ID:29
[INFO] -- Data Published with Packet ID:30
[ALERT] -- LDR TWO below Lower threshold!
[ACTION] -- OPENING WINDOWS
[INFO] -- Data Published with Packet ID:31
[INFO] -- Data Published with Packet ID:32
[ALERT] -- Soil Moisture  below lower threshold!
[ACTION] -- MOTOR ON
[INFO] -- Data Published with Packet ID:33
[INFO] -- Data Published with Packet ID:34
[INFO] -- PHin safe levels!
[INFO] -- Data Published with Packet ID:35
[INFO] -- Data Published with Packet ID:36
[ALERT] -- Failed to read from DHT sensor!
[ACTION] -- Publishing temperature
[INFO] -- Temperature:0
[INFO] -- Data Published with Packet ID:37
[ACTION] -- Publishing humidity
[INFO] -- Humidity:0
[ACTION] -- Publishing soil moisture
```


1:55

GreenHouse

Sensors Readings

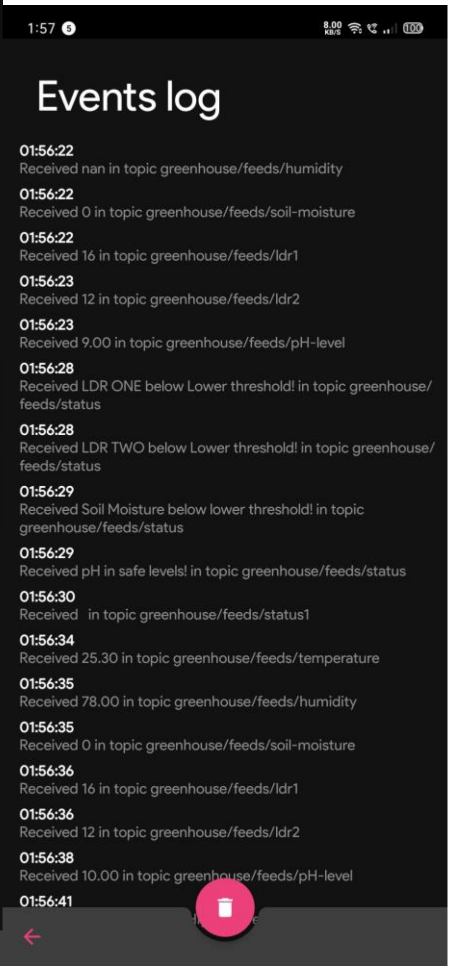
Temperature (11 seconds ago) 25	Soil Moisture (9 seconds ago) 0
Humidity (9 seconds ago) 79	pH-Level (9 seconds ago) 11
Luminosity A (9 seconds ago) 16	Luminosity B (9 seconds ago) 12

Threshold settings

Lower Temperature limit (5 hours ago) 25 °C	Higher Temperature limit (5 hours ago) 40 °C
Lower Humidity limit (5 hours ago) 30 %	Higher Humidity limit (5 hours ago) 70 %
Lower Soil Moisture limit (4 hours ago) 50 %	Higher Soil Moisture limit (4 hours ago) 65 %
pH level lower limit (Jan 1, 2021) 4 pH	pH level higher limit (Jan 1, 2021) 12 pH
Luminosity A lower limit (5 hours ago) 20 LUX	Luminosity A higher limit (5 hours ago) 95 LUX
Luminosity B lower limit (5 hours ago) 20 LUX	Luminosity B higher limit (5 hours ago) 95 LUX

User actions

Feedback Status (1 second ago) LDR TWO below Lower threshold!	Last Action Status (0 seconds ago) OPENING WINDOWS
Water Pump Switch (2 hours ago)	Manual Override (2 hours ago)
Window Switch (2 hours ago)	


1:57

Events log

```
01:56:22
Received nan in topic greenhouse/feeds/humidity
01:56:22
Received 0 in topic greenhouse/feeds/soil-moisture
01:56:22
Received 16 in topic greenhouse/feeds/ldr1
01:56:23
Received 12 in topic greenhouse/feeds/ldr2
01:56:23
Received 9.00 in topic greenhouse/feeds/pH-level
01:56:28
Received LDR ONE below Lower threshold! in topic greenhouse/
feeds/status
01:56:28
Received LDR TWO below Lower threshold! in topic greenhouse/
feeds/status
01:56:29
Received Soil Moisture below lower threshold! in topic
greenhouse/feeds/status
01:56:29
Received pH in safe levels! in topic greenhouse/feeds/status
01:56:30
Received  in topic greenhouse/feeds/status1
01:56:34
Received 25.30 in topic greenhouse/feeds/temperature
01:56:35
Received 78.00 in topic greenhouse/feeds/humidity
01:56:35
Received 0 in topic greenhouse/feeds/soil-moisture
01:56:36
Received 16 in topic greenhouse/feeds/ldr1
01:56:36
Received 12 in topic greenhouse/feeds/ldr2
01:56:38
Received 10.00 in topic greenhouse/feeds/pH-level
01:56:41
```

CHAPTER 6
CONCLUSION

- Efficiency-The crops can be grown in more efficient manner.
- Yield- A positive growth in the yield rate can be observed by using this.
- Condition monitoring-The factors effecting the crops growth can be constantly.

CHAPTER 7

FUTURE SCOPE

- Image Recognition for detecting crop diseases
- Using Higher Accuracy Sensors for detailed output and precise measurement