**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CH.NO** | **TITLE OF TABLE** | **PAGE NO** |

**CHAPTER 1: INRODUCTION 2**

**CHAPTER 2:** **WORKING PROCESS 4**

**CHAPTER 3: CONCLUSION 18**

**ABSTRACT:**

From last few years, the chicken production in the world has been increasing gradually because of standardized farming management and good manufacturing practices.A chicken needs 14-16 hours of light to produce healthy eggs, so we have designed our project in such a way that chicken should produce healthy hens which will increase the productivity and improve the income of agriculture sector.

According to world’s agricultural produce survey, chicken is the most favourite produce, since it is a nutrient rich food providing high protein, low fat and low cholesterol, and lower energy than other kinds of poultries. From last few days around the world, there has been an increased level of awareness regarding the safety of food products like chickens and there has been a high demand for good quality chicken food.This work aims to provide details on how to build an IoT Enabled Smart Poultry Farm using low cost commodity hardware and open source software. A comprehensive system was built usingNodeMCU, DHTsensor, Ultrasonicsensor, MQ2sensor (Gassensor), Servomotor.The system has been thoroughly investigated for various physical parameters associated with effective poultry management which includes temperature and to on or off the motor. It was found thatthe system not only monitors these parameters, but also regulates these parameters effectively. The framework wasobserved to be very useful for farmers as they could easily access and control the system remotely using their handheldmobile devices. The system reduces human intervention, savestime, optimizes resource utilization and increases poultryproduction.

**INTRODUCTION:**

INDIA is considered as an agricultural wealthycountry in terms of food and environmental resources. Nevertheless, such prosperity was gradually regressed directlycontributing to a low agricultural productivity and farmerincomes. The farmers additionally lacked insights in agricultural marketing strategies and high-quality productionplanning. According to world’s agricultural produce, chicken is themost favoured produce, since it is a nutrient-rich food providinghigh protein, low fat and cholesterol, and lower energy anotherkind of poultries. Also, it is quite easy to look after andpropagate its species. For 5 years, the chicken productionhas been increasing on an average of 4.63% yearly because ofstandardized farming management and good manufacturingpractices, leading to more chicken consumption and anincreased export number of both domestic and internationaldestinations. On the other hand, a lacking of labour inchicken production processes has affected fresh chickenexport, which is found to be the principal problem. Another significant obstacle can be wrong knowledge sharingand folk wisdom in chicken farming which effects efficiency.

This study aims to set up a new model by using a moderntechnology applied to chicken farming known as a "SmartFarm" or "Intelligent Farm", which is expected to clear upthe IOT Based smart poultry farming using commodity hardware and software farming. Smart Farm could perceiveany changed information derived from a semi-automaticmicroprocessor, alarming all notification to a connected personalcomputer. The farm monitoring could be conducted viaapplication programs on smart phones for convenience use,time saving, and reduced labour dependency, cost.

**PROBLEM STATEMENT:**

Illumination is one of the key parameters in deciding the breading date of hens.

Higher illumination causes early breading of hens with smaller size eggs.

Controlling the illumination w.r.t. the breading date and external weather will

lead to higher productivity.

**WORKING PROCESS:**

The main aim of the project is to improve the health of chicken,improve productivity and to make traditional poultry more smarter .Here we are designing poultry in such a way that we provide good illuminating light for breeding of hens,environmental conditions can be altered.

Firstly the, DHT 11 sensor is used to detect the temperature in the farm. If the temperature is greater than 35 in the farm then the owner will get a message that the Temperature is HIGH.

But if the temperature is less than 25 the person should get a message that temperature is LOW.

Later by using two servomotors which will rotate 0 to 180 and 180 to 0 degrees for closing and opening it reaches again its original position.

When LDR senses daylight roof of poultry farm will be opened, meanwhile the illuminating light inside the poultry farm will be turned off automatically.

When it turns dark outside,doors(roof) will be closed simultaneously illuminating light inside the poultry will be turned on .

The timer will calculate the time of the daylight, and according to the data present the remaining time will be given by the artificial light.

**COMPONENTS USED:**

A comprehensive system was built using:

1.Hardware

2.Software

The hardware components are:

* NodeMCU
* DHT sensor
* Servomotor
* MQ2 sensor
* OLED (to display)

**1.NodeMCU:**

NodeMCU is an open source [IoT](https://en.wikipedia.org/wiki/Internet_of_Things" \o "Internet of Things) platform. The term "NodeMCU" by default refers to the firmware rather than the development kits. NodeMCU provides access to the [GPIO](https://en.wikipedia.org/wiki/General-purpose_input/output" \o "General-purpose input/output) (General Purpose Input/Output).It includes [firmware](https://en.wikipedia.org/wiki/Firmware" \o "Firmware) which runs on the [ESP8266](https://en.wikipedia.org/wiki/ESP8266" \o "ESP8266) [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi" \o "Wi-Fi) [SoC](https://en.wikipedia.org/wiki/System_on_a_chip" \o "System on a chip) from [Espressif Systems](https://en.wikipedia.org/w/index.php?title=Espressif_Systems&action=edit&redlink=1" \o "Espressif Systems (page does not exist)), and hardware which is based on the ESP-12 module.

An open source firmware and development kit that helps you to prototype your IoT product within a few Lua script lines.



**Features:**

Open-source, Interactive, Programmable, Low cost, Simple, Smart, WI-FI enabled.

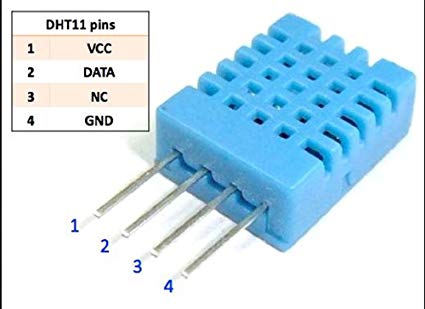
**2.DHT Sensor:**

This DHT11 Temperature and Humidity Sensor features a calibrated digital signal output with the temperature and humidity sensor capability. It is integrated with a high-performance 8-bit microcontroller. Its technology ensures the high reliability and excellent long-term stability.  This sensor includes a resistive element and a sensor for wet NTC temperature measuring devices. It has excellent quality, fast response, anti-interference ability and high performance.

Each DHT11 sensors features extremely accurate calibration of humidity calibration chamber. The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, we should call these calibration coefficients. The single-wire serial interface system is integrated to become quick and easy. Small size, low power, signal transmission distance up to 20 meters, enabling a variety of applications and even the most demanding ones. The product is 4-pin single row pin package. Convenient connection, special packages can be provided according to users need.

**Specification:**

* Supply Voltage: +5 V
* Temperature range :0-50 °C error of ± 2 °C
* Humidity :20-90% RH ± 5% RH error
* Interface: Digital

****

|  |  |  |
| --- | --- | --- |
|  |  |  |

**3.Servomotor:**

A servomotor is a [rotary actuator](https://en.wikipedia.org/wiki/Rotary_actuator" \o "Rotary actuator) or [linear actuator](https://en.wikipedia.org/wiki/Linear_actuator" \o "Linear actuator) that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.Servomotors are not a specific class of motor although the term servomotoris often used to refer to a motor suitable for use in a [closed-loop control](https://en.wikipedia.org/wiki/Closed-loop_control" \o "Closed-loop control) system.

Servomotors are used in applications such as [robotics](https://en.wikipedia.org/wiki/Robotics" \o "Robotics), [CNC machinery](https://en.wikipedia.org/wiki/CNC_machine" \o "CNC machine) or manufacturing. A servomotor is a [closed-loop](https://en.wikipedia.org/wiki/Closed-loop_controller" \o "Closed-loop controller) [servomechanism](https://en.wikipedia.org/wiki/Servomechanism" \o "Servomechanism) that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.The motor is paired with some type of [encoder](https://en.wikipedia.org/wiki/Encoder" \o "Encoder) to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an [error signal](https://en.wikipedia.org/wiki/Error_signal" \o "Error signal) is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.



**4.OLED:**

The OLED display module is one of the most attractive display available for a microcontroller. It has a good view angle and pixel density which makes it reliable for displaying small level graphics. Interfacing this IC with MCU can either be done using IIC or using SPI hence helps to save some pins as well. OLED is organic light emitting diode that emits light in response to an electric current. OLED display works with no backlight so it can display deep black levels. It is small in size and light in weight than Liquid Crystal Displays128x64 OLED display is simple dot matrix graphic display. It has 128 columns and 64 rows which make it display of total 128x64 = 8192 pixels. By just turning on/off these pixel’s led we can display graphical image of any shape on it.

OLED display is used for displaying text, images and various patterns. It is also suitable for mobile phone sub-display, MP3 player, calculators etc. OLED display has 256 steps for +brightness control. OLED display also available with different resolution like 128x32, 128x64.

**Pin Configuration:**

* Ground (Gnd) - Connected to the ground of the circuit
* Vcc (Vdd,5V) - Can be powered by either 3.3V or 5V
* SCL (D0, CLK) - It is a clock signal. This pin transmits clocks to slave, SCL. Data will be sent to other devices on clock tick event. Only master device has control over this SCL line
* SDA (D1, MOSI) - SDA is used to transmit data between master and slave. The data and acknowledgement are sent through SDA.



The software components are:

* Arduino IDE
* Blynk App
* Nodered

**1.Arduino IDE:**

The [Arduino](https://en.wikipedia.org/wiki/Arduino" \o "Arduino) integrated development environment ([IDE](https://en.wikipedia.org/wiki/Integrated_development_environment" \o "Integrated development environment)) is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform" \o "Cross-platform) application (for [Windows](https://en.wikipedia.org/wiki/Windows" \o "Windows), [macOS](https://en.wikipedia.org/wiki/MacOS" \o "MacOS), [Linux](https://en.wikipedia.org/wiki/Linux" \o "Linux)) that is written in the programming language [Java](https://en.wikipedia.org/wiki/Java_(programming_language)" \o "Java (programming language)). It is used to write and upload programs to Arduino compatible boards, but also, with the help of 3rd party cores, other vendor development boards.

The source code for the IDE is released under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License" \o "GNU General Public License), version 2. The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)" \o "C (programming language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B" \o "C++) using special rules of code structuring. The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library" \o "Software library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)" \o "Wiring (development platform)) project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive" \o "Cyclic executive) program with the [GNU toolchain](https://en.wikipedia.org/wiki/GNU_toolchain" \o "GNU toolchain), also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. 

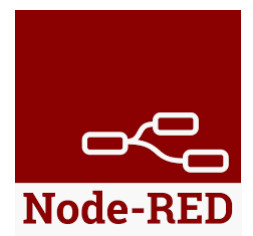
**2.Blynk app:**

Blynk - The most popular IoT platform to connect your devices to the cloud, design apps to control them, and manage your deployed products at scale.

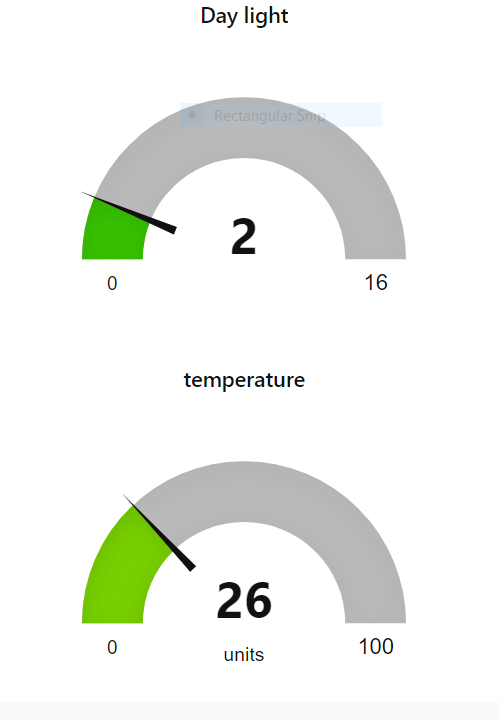
Blynk can control any electronics, GPIOs, relays, anything

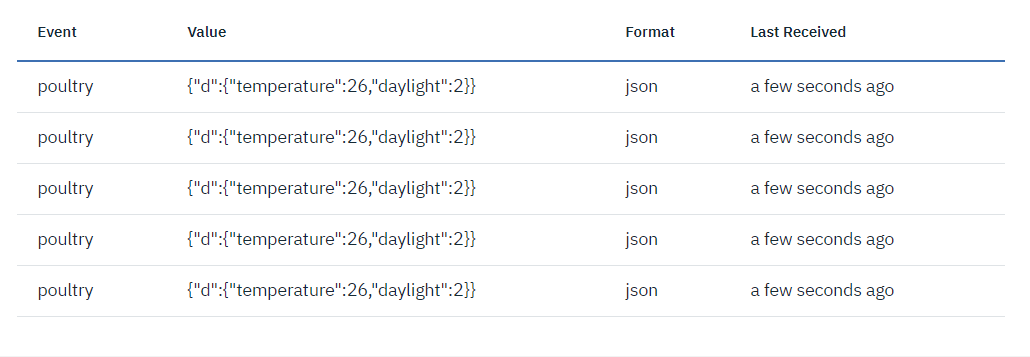
1. **NODERED:**

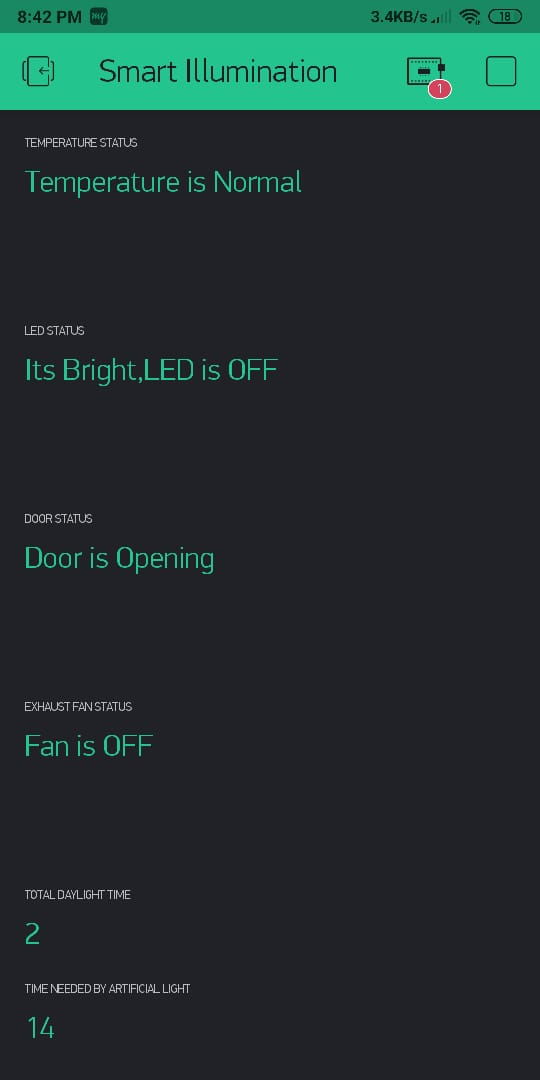
Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions



**OUTPUT SCREENSHOTS:-**







**CODE:**

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <Wire.h>

#include <Time.h>

#include <TimeLib.h>

#include <Servo.h>

#include <BlynkSimpleEsp8266.h>

#define BLYNK\_PRINT Serial

#include <SimpleTimer.h>

SimpleTimer timmer;

#include "DHT.h"

#define ldrPin A0

#define ledPin D3

#include <Adafruit\_SSD1306.h>

#include <Adafruit\_GFX.h>

#define SSD1306\_LCDHEIGHT 64

#define OLED\_ADDR 0x3C

Adafruit\_SSD1306 display(-1);

#if (SSD1306\_LCDHEIGHT != 64)

#error("Height incorrect, please fix Adafruit\_SSD1306.h!");

#endif

int timer;

Servo servo1;

Servo servo2;//defining the name usage as servo itself

int lopper;

int fanled=D8;

int bal;

#define DHTPIN D0

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

//String command;

char auth[] = "d02da247a04b4f1eb9aae9366e3d7412";

const char\* ssid = "kevin";

const char\* password = "12345678i";

char pass[] = "12345678i";

#define ORG "gawt0d"

#define DEVICE\_TYPE "praneetha"

#define DEVICE\_ID "1843"

#define TOKEN "123456789"

String command;

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char pubtopic[] = "iot-2/evt/poultry/fmt/json";

char topic[] = "iot-2/cmd/poultry/fmt/String";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

WiFiClient wifiClient;

void callback(char\* topic, byte\* payload, unsigned int payloadLength);

PubSubClient client(server, 1883, callback, wifiClient);

float h;

void setup()

{

wifiConnect();

mqttConnect();

Blynk.begin(auth, ssid, pass);

dht.begin();

timer=0;

Serial.begin(9600); //sets serial port for communication

pinMode(ldrPin, INPUT);

pinMode(ledPin, OUTPUT);

servo1.attach(D5);

servo2.attach(D6);

int templdr = analogRead(ldrPin);

lopper=1;

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Smart Illumination");

Serial.println("Smart Illumination");

display.display();

delay(3000);

}

void loop()

{

Blynk.run();

timmer.run();

int gassensor = digitalRead(D7);

//float h = gassensor;

//Serial.print("Pin D7: ");

//Serial.println(gassensor);

int ldrStatus = analogRead(ldrPin);

int i,k;

float t = dht.readTemperature();

//if(lopper==1)

//{

if(t>35)

{

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Temperature is HIGH");

display.display();

char q[]=("Temperature is High");

Blynk.virtualWrite(V7, q);

delay(100);

Serial.println("Temperature is HIGH");

//digitalWrite(temp\_led, HIGH); //Red

delay(200);

// char i[]=("Temperature is High");

//Blynk.virtualWrite(V5, i);

delay(100);

}

else if(t<25)

{

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Temperature is LOW");

display.display();

char q[]=("Temperature is LOW");

Blynk.virtualWrite(V7, q);

delay(100);

Serial.println("Temperature is LOW");

// char h[]=("Temperature is Low");

// Blynk.virtualWrite(V5, h);

delay(100);

}

else

{

// digitalWrite(temp\_led, LOW);

// char j[]=("Temperature is Normal");

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Temperature is Normal");

display.display();

char q[]=("Temperature is Normal");

Blynk.virtualWrite(V7, q);

delay(100);

Serial.println("Temperature is Normal");

//Blynk.virtualWrite(V5, j);

delay(1000);

}

if (ldrStatus <= 1000)

{ //its bright

digitalWrite(ledPin, LOW);

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Its Bright,LED is OFF : ");

display.display();

char w[]=("Its Bright,LED is OFF");

Blynk.virtualWrite(V5, w);

delay(1000);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Door is Opening");

char e[]=("Door is Opening");

Blynk.virtualWrite(V4, e);

Serial.println("Its Bright,Turn off the LED : ");

delay(100);

Serial.println("Door is Opening");

Serial.println(ldrStatus);

delay(100);

servo1.write(180);

servo2.write(0);

if (gassensor==0)

{

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

digitalWrite(fanled,HIGH);

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Fan is ON");

display.display();

Serial.println("Fan is ON");

char r[]=("Fan is ON");

Blynk.virtualWrite(V6, r);

//Shows the value of LDR sensor

}

else

{

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

digitalWrite(fanled,LOW);

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Fan is OFF");

display.display();

Serial.println("Fan is OFF");

Serial.println(gassensor);

char r[]=("Fan is OFF");

Blynk.virtualWrite(V6, r);

lopper=1;

}

delay(10000);

timer=timer+1;

h=timer;

//Serial.println(ldrStatus); //Shows the value of LDR sensor

}

else if(ldrStatus > 1000)

{ //its dark

Serial.println("Finished");

// elapsed=finished-start;

// Serial.print(elapsed);

//digitalWrite(ledPin, HIGH); //led light illumination

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Its Dark,LED is on : ");

display.display();

char w[]=("Its Bright,LED is Off");

Blynk.virtualWrite(V5, w);

delay(1000);

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Door is closing");

char e[]=("Door is closing");

Blynk.virtualWrite(V4, e);

Serial.println("Its Dark,LED is ON: ");

delay(100);

Serial.println("Door is Closing");

delay(100);

servo1.write(0);

servo2.write(180);

Serial.print("Total Daylight Time : ");

Serial.print(timer);

int z=timer;

Blynk.virtualWrite(V2, z);

Serial.println("Hours");

Serial.print("ldrStatus: ");

Serial.println(ldrStatus);

if(lopper==1)

{

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

digitalWrite(fanled,HIGH);

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Exhaust fan is on");

display.display();

Serial.println("Exhaust fan is on");

if(timer<16)

{

k=timer;

bal=(16-k);

int p=bal;

Blynk.virtualWrite(V3,p);

for(i=0;i<bal;i++)

{

digitalWrite(ledPin, HIGH);

delay(1000);

}

digitalWrite(ledPin,LOW);

}

}

lopper=0;

if (gassensor==0)

{

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

digitalWrite(fanled,HIGH);

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Fan is ON");

display.display();

Serial.println("Fan is ON");

char r[]=("Fan is ON");

Blynk.virtualWrite(V6, r);

//Shows the value of LDR sensor

}

else

{

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

digitalWrite(fanled,LOW);

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.print("Fan is OFF");

display.display();

Serial.println("Fan is OFF");

Serial.println(gassensor);

char r[]=("Fan is OFF");

Blynk.virtualWrite(V6, r);

}

}

if(!client.loop()) {

mqttConnect();

}

PublishData(t,h);

delay(100);

}

void wifiConnect() {

Serial.print("Connecting to "); Serial.print(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.print("nWiFi connected, IP address: "); Serial.println(WiFi.localIP());

}

void mqttConnect() {

if (!client.connected()) {

Serial.print("Reconnecting MQTT client to "); Serial.println(server);

while (!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

initManagedDevice();

Serial.println();

}

}

void initManagedDevice() {

if (client.subscribe(topic)) {

Serial.println("subscribe to cmd OK");

} else {

Serial.println("subscribe to cmd FAILED");

}

}

void callback(char\* topic, byte\* payload, unsigned int payloadLength) {

Serial.print("callback invoked for topic: "); Serial.println(topic);

for (int i = 0; i < payloadLength; i++) {

//Serial.println((char)payload[i]);

command += (char)payload[i];

}

command ="";

}

void PublishData(float t,int h){

if (!!!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {

Serial.print(".");

delay(500);

}

Serial.println();

}

String payload = "{\"d\":{\"temperature\":";

payload += t;

payload+="," "\"daylight\":";

payload += h;

payload += "}}";

Serial.print("Sending payload: ");

Serial.println(payload);

if (client.publish(pubtopic, (char\*) payload.c\_str())) {

Serial.println("Publish ok");

} else {

Serial.println("Publish failed");

}

}

**ADVANTAGES:**

**Healthier and Good breed of chicken will be produced which will improve the income of agiculture sector**

**The correct amount of light produced by illuminating light for breeding will improve the productivity.**

**CONCLUSION:**

The traditional way of chicken farming is replacing with the smart and intelligent chicken farming using embedded system based innovative application. It helps the farmers real time controlled and monitoring environmental aware context parameters such as temperature, humidity, air quality, light intensity and controlling filter fan, ventilation window. This smart system can effectively control the farm from any location and reduces cost time and man power. This will improve productivity and quality of chickens in poultry farming. In the future advanced IOT based technologies should be use for monitoring and controlling health related parameters of chicken to improve quality and productivity of chicken farming, which will Result into profits for farmers and quality food for human being.

TEAM MEMBERS: KEVIN JOE RONIN

PRANEETHA

PAVITHRA HOSALLIKAR