CHAPTER 1

1.1 INTRODUCTION

The IoT-based Smart Factory System with Sensor Integration and Cloud Connectivity project uses the power of Internet of Things technologies. By combining various sensors and connecting them to the cloud connectivity, we create a smooth and seamless system, this system revolutionizes factory operations by enabling automation, optimization, and real-time monitoring.

These sensors encompass temperature and humidity sensors, flame sensors, infrared (IR) sensors for product counting, and load cells for precise weight measurement. The data collected from these sensors serves as a valuable resource, offering deep insights into the factory environment and facilitating informed decision-making for enhanced operational efficiency.

A key highlight of our project lies in the seamless integration of cloud connectivity. By establishing a secure connection between the smart factory system and the cloud, we enable the seamless transmission and storage of sensor data in cloud-based databases.

The IoT-based Smart Factory System provides a many number of advantages to manufacturers. It automates critical processes such as activating ventilation systems based on temperature and humidity readings, or triggering water pumps in response to flame detection. The system also offers real-time display of weight measurements, allowing precise monitoring of product quantities.

By implementing our IoT-based Smart Factory System, manufacturers gain unprecedented levels of efficiency, and productivity. The integration of sensors, cloud connectivity, and data analytics provides invaluable insight. We have used the following hardware/software components in our project:

i. Node MCU ESP8266 with integrated Wi-Fi and BLE connectivity which is used in a wide range of applications.
ii. Arduino Uno
iii. LCD display
iv. Load cell sensor
v. HX711 amplifier

vi. IR sensor
vii. Flame sensor
viii. DHT11 sensor
ix. BO motor
x. Node MCU base connector
xi. Fan
xii. Motor pump
xiii. 2channel 5v Relay
xiv. Arduino IDE
xv. Thinger.io Cloud platform

1.2 SCOPE OF THE CAPSTONE PROJECT

- **Sensor Integration:** The project involves integrating various sensors like IR, flame, DHT11, and load cell with the HX711 module to collect real-time data.

- **Wireless Communication and Cloud Integration:** The system will utilize the ESP8266 board's Wi-Fi capabilities to establish communication with the Thinger.io platform for seamless data transmission and control, enabling remote monitoring and control functionalities.

- **Remote Monitoring and Notifications:** The project will provide remote access to monitor sensor readings, system status, and actuator control through the Thinger.io platform. It will also support notifications and alerts for abnormal conditions.

- **User Interface and Data Visualization:** The system will feature an LCD screen to display relevant information, including real-time weight measurements

- **Power Management and Scalability:** The project will implement power-saving mechanisms for energy optimization and consider scalability for future expansion, accommodating additional sensors or devices as needed.

- **Documentation and Support**: The project will include comprehensive documentation, such as system design, implementation, and setup instructions, along with user manuals. It will provide support for system configuration and troubleshooting to ensure smooth operation and maintenance of the smart factory system.

CHAPTER 2

2.1 CAPSTONE PROJECT PLANNING

Capstone project planning is an important step in project management where we carefully plan and organize our project. It helps us to understand the purpose of the project and make decisions about how to achieve our goals. Through planning, we can determine what needs to be done, when it should be done, and who will be responsible for each task. It ensures that everyone is on the same page and working towards a common objective. Overall, capstone project planning sets the stage for a successful project by providing a roadmap for its execution.

2.1.1 Work Breakdown Structure (WBS) for Smart Factory system

Work Breakdown Structure shows the project tasks level from beginning to end of the project, and objectives of smart factory system. It provides a roadmap to build a project with successful with co-ordination.

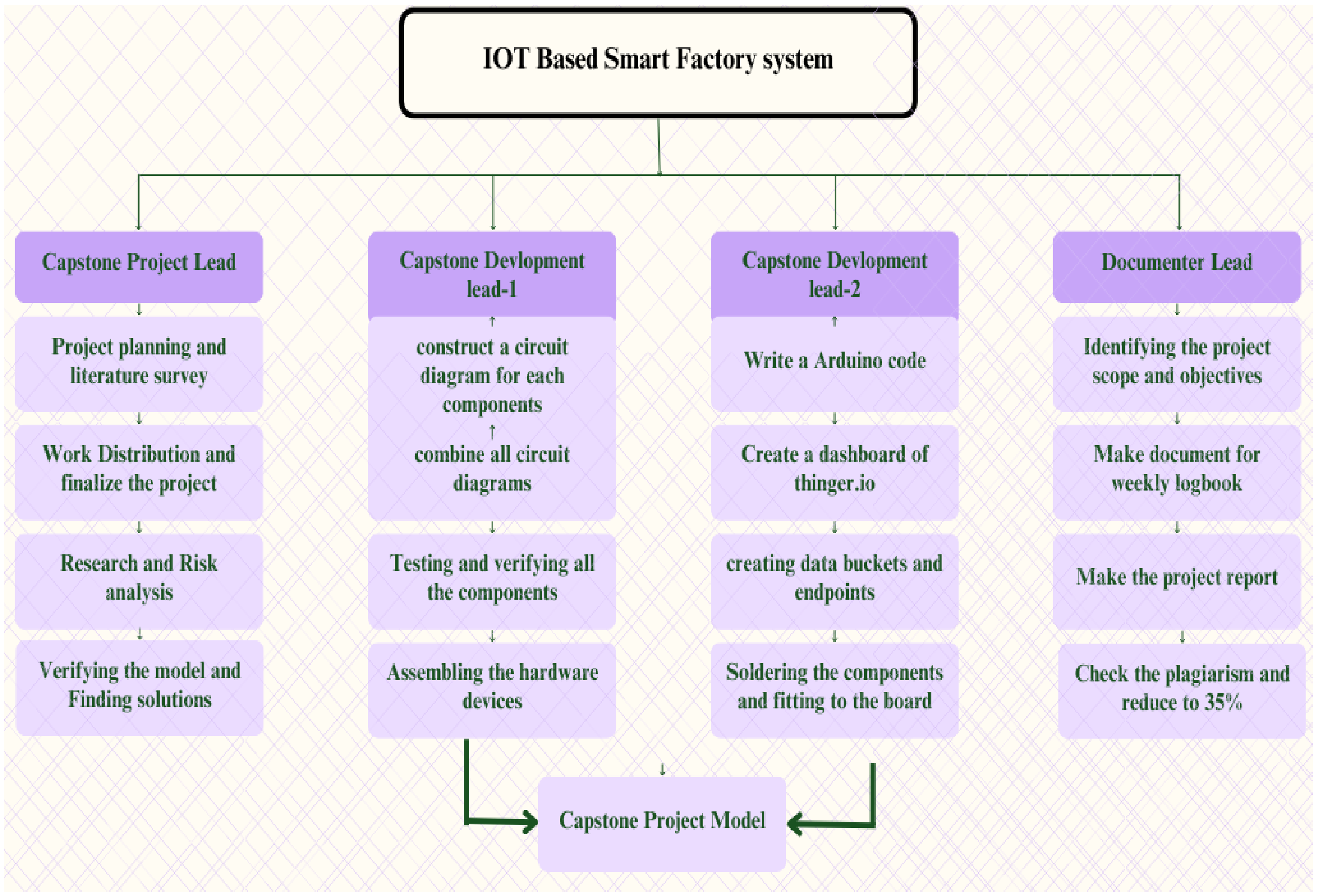

IOT Based Smart Factory system
Capstone Project Lead
Project planning and literature survey
Work Distribution and finalize the project
Research and Risk analysis
Verifying the model and Finding solutions
Capstone Devlopment lead-1
construct a circuit diagram for each components
combine all circuit diagrams
Testing and verifying all the components
Assembling the hardware devices
Capstone Devlopment lead-2
Write a Arduino code
Create a dashboard of thinger.io
creating data buckets and endpoints
Soldering the components and fitting to the board
Documenter Lead
Identifying the project scope and objectives
Make document for weekly logbook
Make the project report
Check the plagiarism and reduce to 35%
Capstone Project Model

Figure 2.1 Flow chart of Work Breakdown structure (WBS)

2.1.2 Timeline Development – Schedule for Smart Factory system

Week-1: Discussing about our capstone project with the reference of marketing scope.
Week-2: Prepared a literature survey based on discussed with the group members.
Week-3: Planning & designing of our project “IOT Based Smart Factory with Sensor Integration and Cloud Connectivity" & collecting related document of our project
Week-4: To start basic requirement of project as per requirement & configuration.
Week-5: To buy the electronic interfacing modules like controller, sensors, and fabrication materials
Week-6: To start a programing for in our project we are using "ARDUINO IDE” software.
Week-7: As per programming to create as separate or individual components circuit diagram with reference of coding.
Week-8: Searched for perfect cloud platform and finalize the Thinger.io Cloud.
Week-9: To complete coding part then finish up necessity of a fabrication works.
Week-10: To connect the hardware components with the controller as per coding instruction then finalize our project.
Week-11: Integration of components as per required for capstone project.

Week-12: To start necessity document preparation for individual components and make a Report on chapter 1-3.
Week-13: Prepared a project banner for introduction and fitted to the wall.
Week-14: To check the report as per DTE plagiarism to check Plagiarism and reduce up to 35%.
Week-15: Finalize our Project designing, testing, and Plagiarism.
Week-16: Planning for PPT and demonstration.

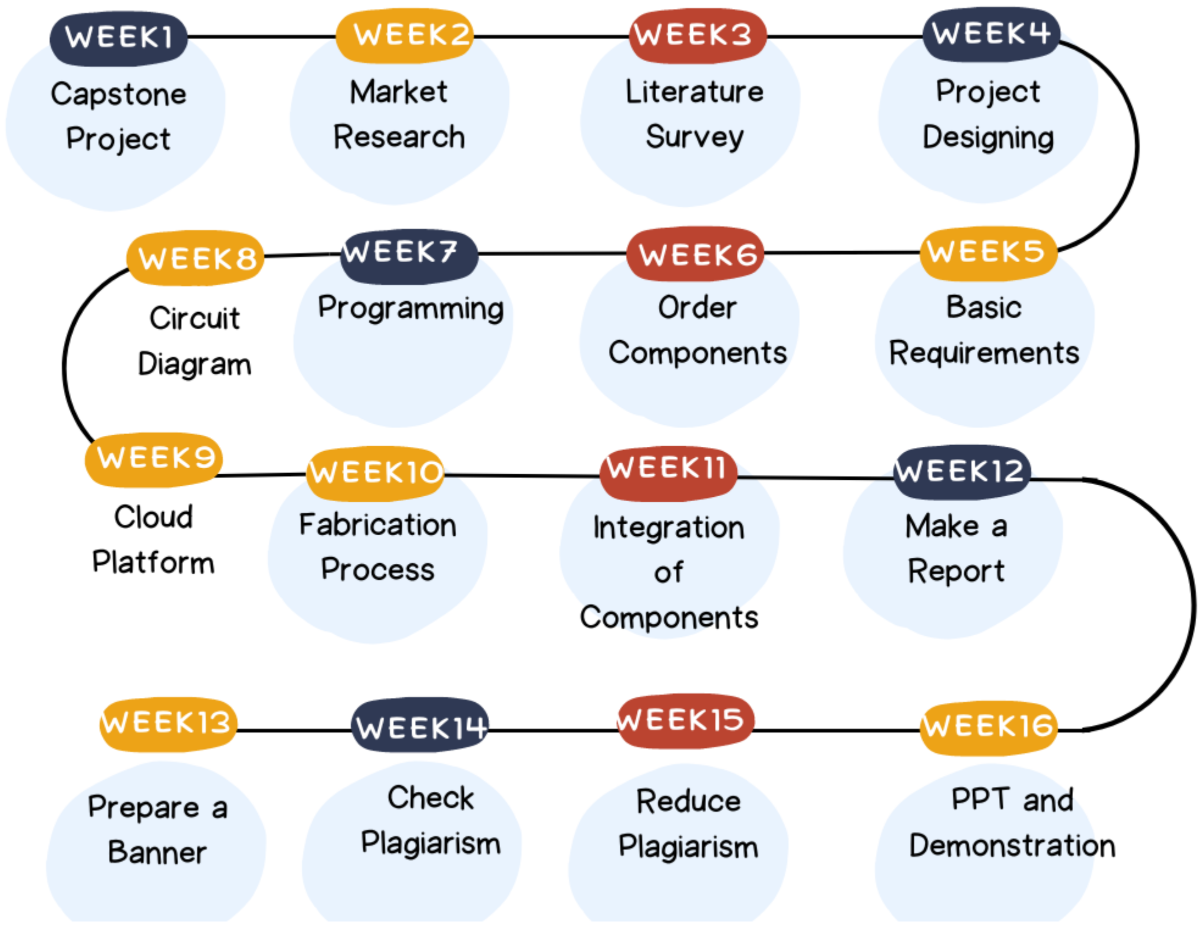


Figure 2.2 Flow chart of Time line schedule

2.1.3 Cost Breakdown Structure (CBS) for Smart Factory system

In our project we have used various components that are listed below with their cost.

Sl. No	Components	Quantity	Cost
1	NodeMCU	1	500
2	Load Cell	1	700
3	Flame Sensor	1	200
4	IR Sensor	1	200
5	DHT11	1	200
6	Mini Fan	1	200
7	Buzzer	1	100
8	Water Pump	1	200
9	DC Motor	2	300
10	jumpers	1set	250
11	Smoke Sensor	1	200
12	Wooden Board	1	200
13	Arduino Uno	1	750
Total			4000

Table 2.1 – Cost Breakdown Structure (CBS) Table

2.1.4 Capstone project risk analysis for Smart Factory system

Risk analysis is a crucial process in project management where potential risks and uncertainties are identified, assessed, and managed. It helps in understanding the potential challenges and their impact on project objectives. By conducting risk analysis, proactive measures can be taken to mitigate risks and ensure project success.

- **Technical risks:** Incompatibility between the ESP8266 and Arduino Uno boards, leading to integration challenges.
- **Safety risks:** Insufficient protection measures for electrical components, leading to potential hazards.
- **Employee resistance:** The implementation of a smart not supportive for employees who are not familiar with new technologies.
- **High upfront costs:** Implementing a smart factory can be expensive, investment in advanced technologies and infrastructure.
- **Performance risks:** insufficient testing to software problems or inaccurate sensor readings.
- **Time risks:** Delays in the obtaining of required hardware components, causing a delay in project timeline.

2.2 REQUIREMENTS SPECIFICATION

2.2.1 Functional

- Automate turning on of ventilation through HVAC system according to the DHT11.
- Real-time display of the weight of products on the 16*2 LCD display.
- Ability to control the speed of the conveyor belt via ESP8266.
- Automatically turning on of water pump through flame sensor when flame detected.
- Storing the data of all sensors in the data buckets for easy retrieval and analysis.
- Maximum number of products per day was counted through IR sensor via conveyor belt.

2.2.2 Non-Functional (Quality Attributes)

- The system should be capable of handling an increasing workload or expanding requirements
- The UI and overall system design should be user-friendly
- The system should very Strong security measures to protect sensitive data and prevent unauthorized access.
- The system should be stable and operate continuously without any failures.
- The system should demonstrate high performance, with low latency and fast response times.
- The system should be implemented in a way to facilitates for easy to maintenance and to update or replace the components without disrupting the overall system.

2.2.3 User Input

- In this system user input is not needed.
- Only is to monitor the Factory Environment.
- Only you can control the speed of work through web dashboard.

2.2.4 Technical Constrains

- The Microcontrollers used in this, such as NodeMCU and Arduino, having processing capabilities and memory limitations.
- The range and stability of WiFi network could impose constraints on the system functionality.
- The components require a stable power supply to operate effectively.
- The accuracy and reliability of the sensors can impact the overall system.
- The system involves data transmission and remote access and malicious activities is crucial
- The project should consider environmental factors, such temperature and dust that may impact the performance of the hardware components.

2.3 Software and Hardware Details

2.3.1 ESP8266

NodeMCU is an open-source Lua based firmware and development board specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

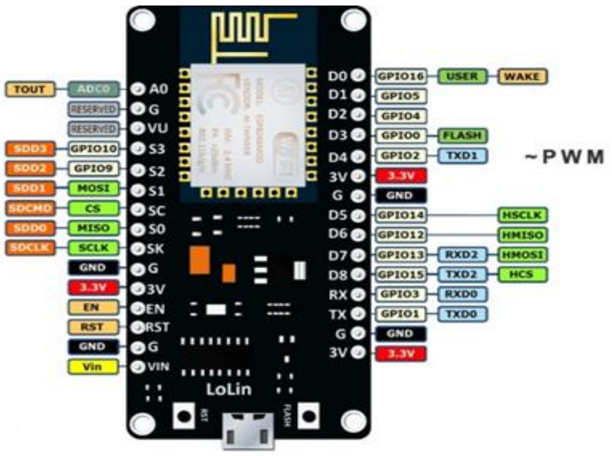


Figure 2.3 Pinout Diagram of Nodemcu ESP8266

CPU	L106 32-bit RISC microprocessor
GPIO	16 Pins
UART	4
SPI	4
I^2C	1
Input Voltage	7-12 Volts
Output Voltage	3.3 volts
Wi-Fi	802.11 b/g/n
Bluetooth	BLE (shares the radio with Wi-Fi)
Clock Speed	80 MHz

Table 2.2 Nodemcu ESP8266 Specification Table

In our project we have used the ESP8266 Development Board to get the data from all sensors and sends to the actuators and analyze it and send the data to Cloud and check it and get back the corresponding output from the server and collected in different data buckets.

2.3.2 Arduino Uno

Arduino Uno is a popular microcontroller development board based on 8-bit ATmega328P microcontroller. Along with ATmega328P MCU IC, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller.

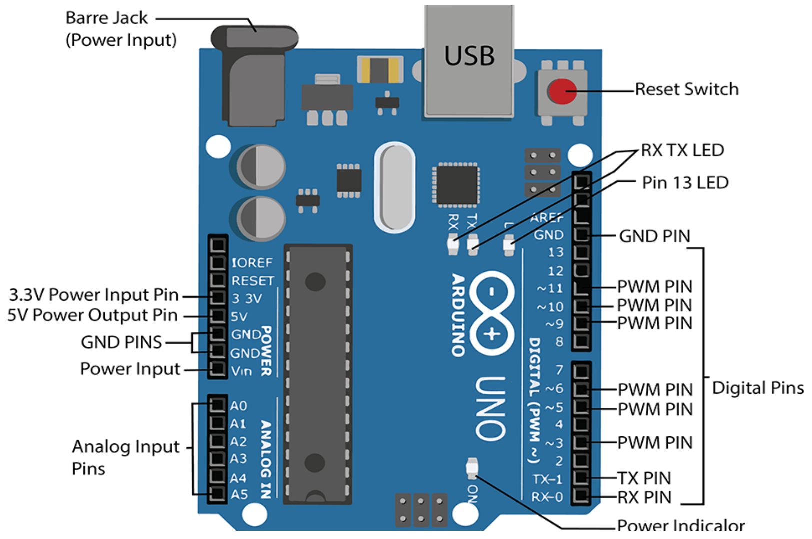


Figure 2.4 Pinout Diagram of Arduino Uno

CPU	ATmega328P – 8 bit AVR family microcontroller
Analog pins	6 Pins
Digital pins	14 Pins
Serial pins	Rx, Tx
PWM	5 Pins
Input Voltage	6 to 12 volts
Output Voltage	5 volts
SPI	4 Pins
Flash memory	ATmega328P – 8 bit AVR family microcontroller
Clock Speed	16 MHz

Table 2.3 Arduino Uno Specification Table

In our project we have used the Arduino Uno Development Board to get the Weight Measure data from Load cell and analyze it and send the data to HX711 amplifier and check it and get back the corresponding output from the Arduino and LCD display.

2.3.3 Relay

The dual-channel relay module is more or less the same as a single-channel relay module, but with some extra features like optical isolation. The dual-channel relay module can be used to switch mains powered loads from the pins of a microcontroller.

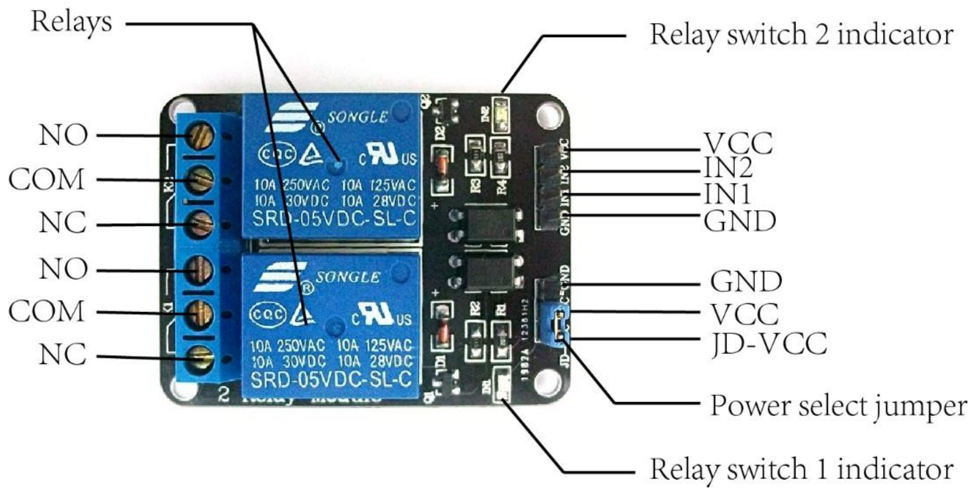


Figure 2.5 Pinout Diagram of 2channel 5v Relay

Current when relay is active	~70mA (single), ~140mA (both)
GND	Input ground reference
IN1	Input to activate the first relay
IN2	Input to activate the second relay
VCC	Input for directly powering the relay coils
Supply Voltage	3.75V to 6V
Trigger current	5mA
maximum current	10A
maximum contact voltage	250VAC, 30VDC
Clock Speed	240MHz

Table 2.4 2channel 5v Relay Specification Table

In our project we have used the 2channel 5v Relay used for fan and water pump for external supply when esp8266 triggered these two will turn on through relay module.

2.3.4 LCD Display

16x2 LCD modules are very commonly used in most embedded projects, the reason being its cheap price, availability, programmer friendly and available educational resources.

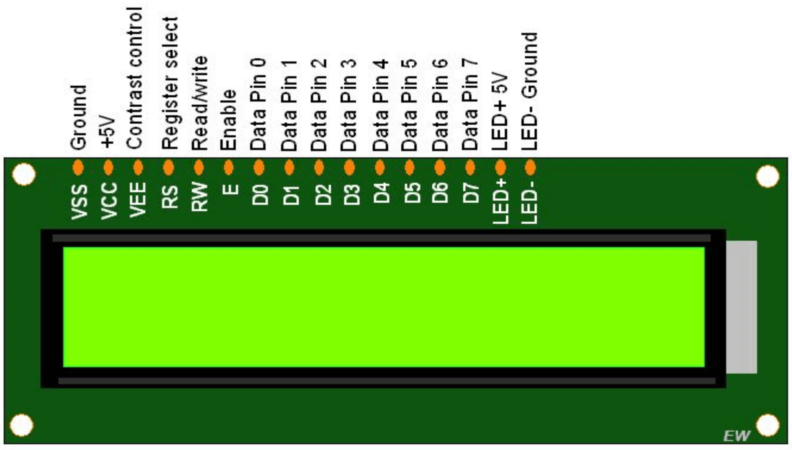


Figure 2.6 Pinout Diagram of LCD display

Vss (Ground)	Ground pin connected to system ground
Data pins	8 Pins
Vdd	Powers the LCD with +5V (4.7V – 5.3V)
VE (Contrast V)	Decides the contrast level of display. Grounded to get maximum contrast.
Register Select	Connected to Microcontroller to shift between command/data register
Input Voltage	4.7V to 5.3V
Current consumption	1mA
Read/Write	Used to read or write data. Normally grounded to write data to LCD
Enable	Connected to Microcontroller Pin and toggled between 1 and 0 for data acknowledgement
LED Positive	Backlight LED pin positive terminal

Table 2.5 16*2 LCD Display Specification Table

In our project we have used the 16*2 LCD Display to get the weight measured data from load cell through HX711 amplifier and Arduino Uno microcontroller.

2.3.5 ESP8266 Base connector

ESP8266 Base connector is a used for Nodemcu Esp8266 for external supply.

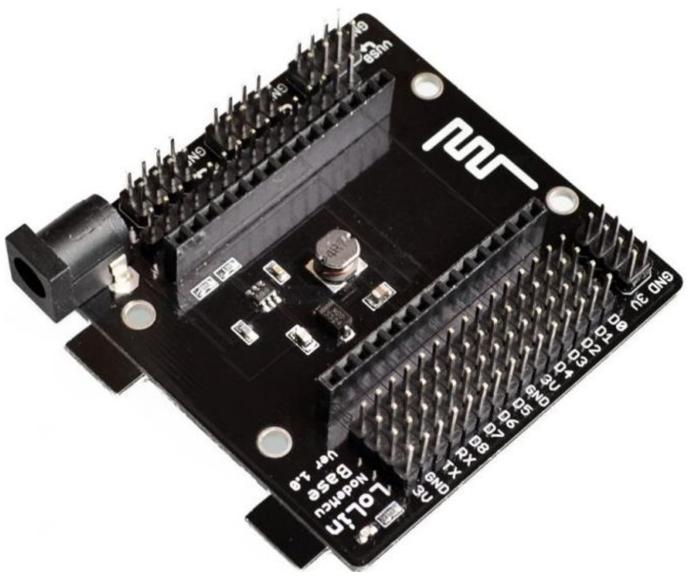


Figure 2.7 Pinout Diagram of ESP8266 Base connector

PCB Thickness	1.6mm
Pin Pitch:	2.54mm (0.1in)
UART	3
SPI	4
I^2C	2
Input Voltage	6V – 24V DC
Output Voltage	5 volts
NodeMCU Pin Distance between sides	~28.0mm
Board Dimensions	60.0mm x 60.0mm x 15.0mm (L x W x H)
Weight:	22.5 g

Table 2.6 ESP8266 Base connector Specification Table

Pins for Extra connections for microcontroller ESP8266, it will powered by adaptor.

2.3.6 IR Sensor

The IR sensor module consists mainly of the IR Transmitter and Receiver, Op-amp, Variable Resistor (Trimmer pot), output LED along with few resistors

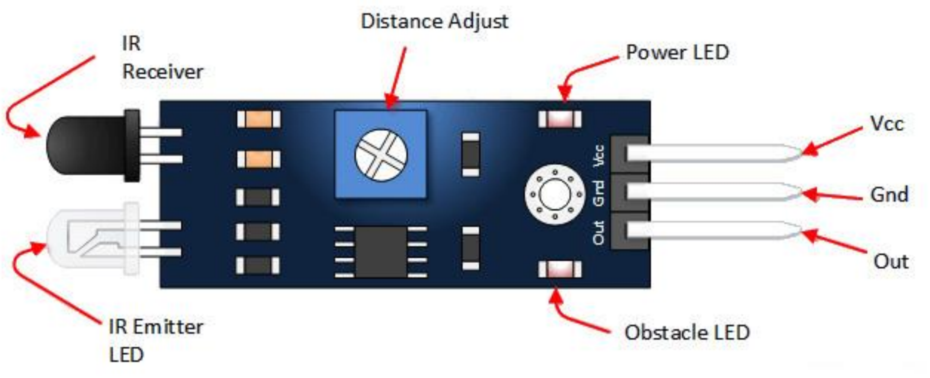


Figure 2.8 Pinout Diagram of IR sensor

VCC	Power Supply Input
GND	Power Supply Ground
OUT	Active High Output
Range	Up to 20cm
supply current	20mA
Input Voltage	3.3 to 5 volts

Table 2.7 IR sensor Specification Table

In our project we have used the IR Sensor for product counting in was installed in the conveyor belt, it counted data was send to cloud thinger.io and stored in the different data buckets.

2.3.7 DHT11 Sensor

The DHT11 is a commonly used Temperature and humidity sensor that comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.

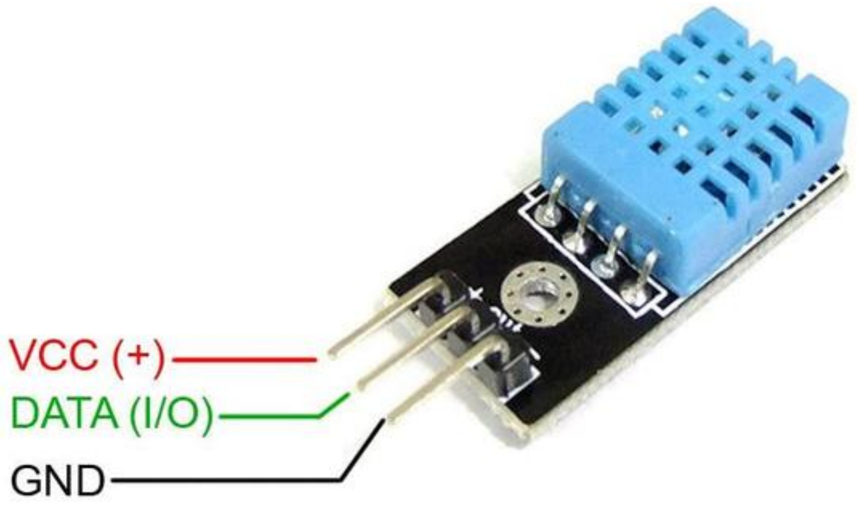


Figure 2.9 Pinout Diagram of DHT11 sensor

Vcc	Power supply
Data	Outputs both Temperature and Humidity through serial Data
Ground	Connected to the ground of the circuit
Operating Voltage	3.5V to 5.5V
Operating current	0.3mA (measuring) 60uA (standby)
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%

Table 2.8 DHT11 Sensor Specification Table

In our project we have used the DHT11 Temperature and Humidity sensor Board to get the data of factory environment and analyze it and send the data to Cloud and check it and get back the corresponding output from the server and it to the Fan or HVAC system and make other corresponding actions if any available.

2.3.8 Flame Sensor

A sensor which is most sensitive to a normal light is known as a flame sensor. That's why this sensor module is used in flame alarms. This sensor detects flame otherwise wavelength within the range of 760 nm – 1100 nm from the light source.

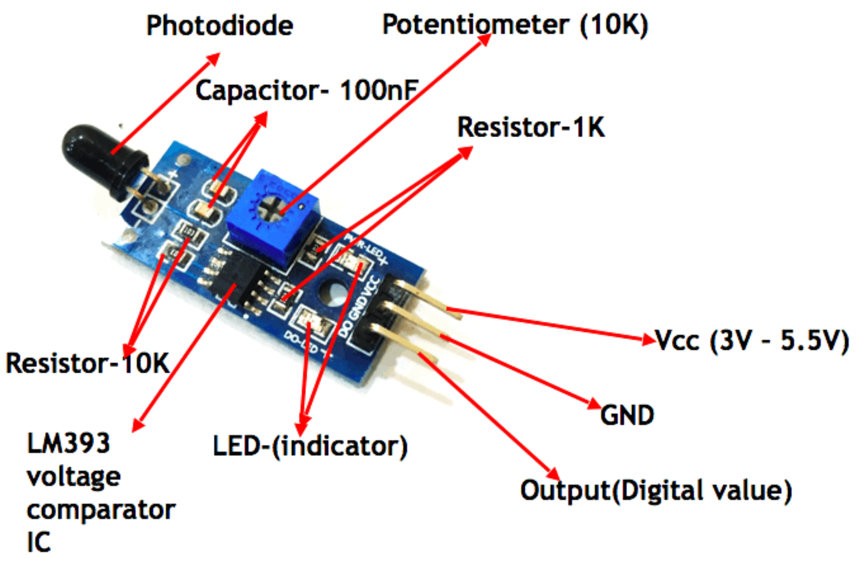


Figure 2.10 Pinout Diagram of Flame sensor

Pin1	VCC pin
Pin2	GND
Pin3	AOUT
Pin4	DOUT
A0	This is an analog output pin (MCU.IO)
Input Voltage	3.3V to 5.3V
D0	This is a digital output pin (MCU.IO)
Range	Upto 100cm
Wavelength	760 nm to 1100 nm

Table 2.9 Flame sensor Specification Table

In our project we have used the Flame sensor Board to get the data of fire detected in the factory environment and send the data to Cloud and check it and get back the corresponding output from the server and display it on the OLED Display and make other corresponding actions if any available.

2.3.9 BO motor

It is a BO Series 1 100RPM DC Motor Plastic Gear Motor. The BO series straight motor gives good torque and rpm at lower operating voltages, which is the biggest advantage of these motors.

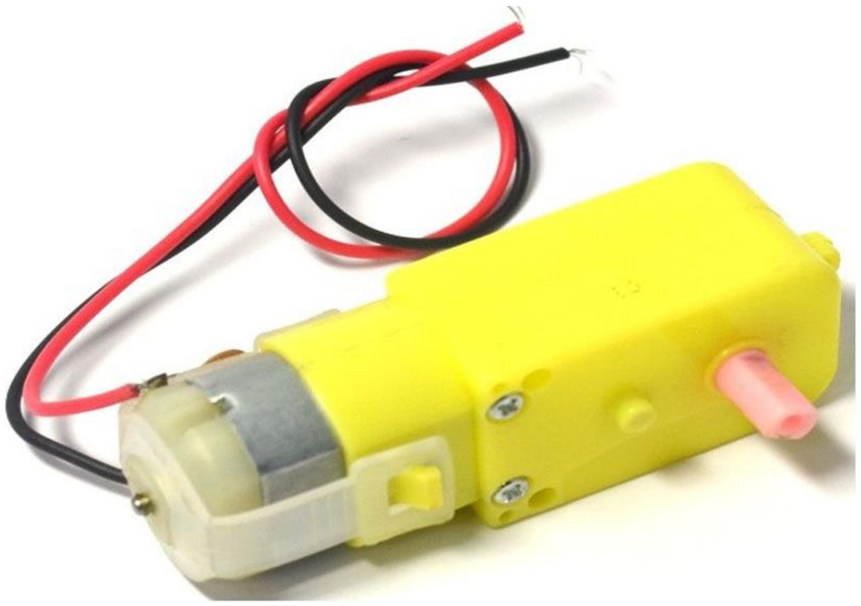


Figure 2.11 Pinout Diagram of BO motor

MOTOR	Permanent Magnet DC Gear Motor BO(Battery Operated)
VOLTAGE(V)	3-9v
CURRENT	0.01A(no load);0.07A(at max. eff.)
LOCKED-ROTOR CURRENT	>=0.15A
SPEED(RPM)	60RPM+-10%(no load)
TORQUE	0.5Kg-cm
ROTATION	CW/CCW
MOUNTING TYPE	Horizontal or Vertical

Table 2.10 BO motor Specification Table

In our project we have used the 2 Bo motors are used parallel the conveyor belt, and we can adjust the speed of conveyor belt through the motors.

2.3.10 Mini 9v water pump

A 9V water pump is a small device that uses a 9-volt battery to pump water. It has a motor and rotating blades that push the water out. It's handy for things like aquariums or fountains.

Figure 2.12 Pinout Diagram of Water pump

Diameter	7mm
Max Flow Rate	100L/H
Temperature	-20~50°C
Voltage	9V DC
Maximum Rated Current	0.18A - Power: 0.36W

Table 2.11 Water Pump Specification Table

In our project we have used the 9v water pump for safety of factory environment, when flame sensor detects the will triggered and the data was sent to the thinger cloud.

2.3.11 Buzzer

An audio signaling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

Figure 2.13 Pinout Diagram of Buzzer

frequency range	3,300Hz
Operating Temperature ranges	– 20° C to +60°C
voltage ranges	3V to 24V DC
sound pressure level	85dBA or 10cm
supply current	below 15mA

Table 2.12 Buzzer Specification Table

In our project we have used the as actuator for flame sensor, when flame detects the buzzer will vibrates for every one second.

2.3.12 Arduino IDE software

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its hardware products are licensed under a CC BY-SA license, while the software is licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially from the official website or through authorized distributors.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (for prototyping) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs. The microcontrollers can be programmed using the C and C++ programming languages, using a standard API which is also known as the Arduino Programming Language, inspired by the Processing language and used with a modified version of the Processing IDE.

2.3.12 Thinger.io Cloud platform

Thinger.io is a cloud IoT Platform that provides every needed tool to prototype, scale and manage connected products in a very simple way. Our goal is to democratize the use of IoT making it accessible to the whole world, and streamlining the development of big IoT projects.

Free IoT platform: Thinger.io provides a lifetime freemium account with only few limitations to start learning and prototyping when your product becomes ready to scale, you can deploy a Premium Server with full capacities within minutes.

Simple but Powerful: Just a couple code lines to connect a device and start retrieving data or controlling its functionalities with our web-based Console, able to connect and manage thousands of devices in a simple way.

Hardware agnostic: Any device from any manufacturer can be easily integrated with Thinger.io's infrastructure.

Extremely scalable & efficient infrastructure: thanks to our unique communication paradigm, in which the IoT server subscribes device resources to retrieve data only when it is necessary, a single Thinger.io instance is able to manage thousands of IoT devices with low computational load, bandwidth and latencies.

Open-Source: most of the platform modules, libraries and APP source code are available in our Github repository to be downloaded and modified with MIT license.

2.4 DESIGN SPECIFICATION

ESP8266 Microcontroller: The ESP8266 serves as the main control unit for the system. It is responsible for handling all the communication with the sensors and actuators and managing data flow between different components.

- **Arduino Uno:** The Arduino controls the Weighing scale machine and displays real-time information about the weight of products on LCD screen.
- **DHT`11:** The temperature and humidity sensor monitors and provides surrounding environment, and it was connected to the cloud through microcontroller.
- **Flame Sensor:** The flame sensor monitors and provides information about the safety of the factory, and it was connected to the cloud through microcontroller.
- **IR Sensor:** The IR sensor and provides information about the number of Products manufactured, and it was connected to the cloud through microcontroller.

- **Fan:** The fan is controlled by temperature and humidity sensor based on the surrounding temperature it operates.
- **Water Pump:** The pump is controlled by Flame sensor it acts as a fire extinguisher.

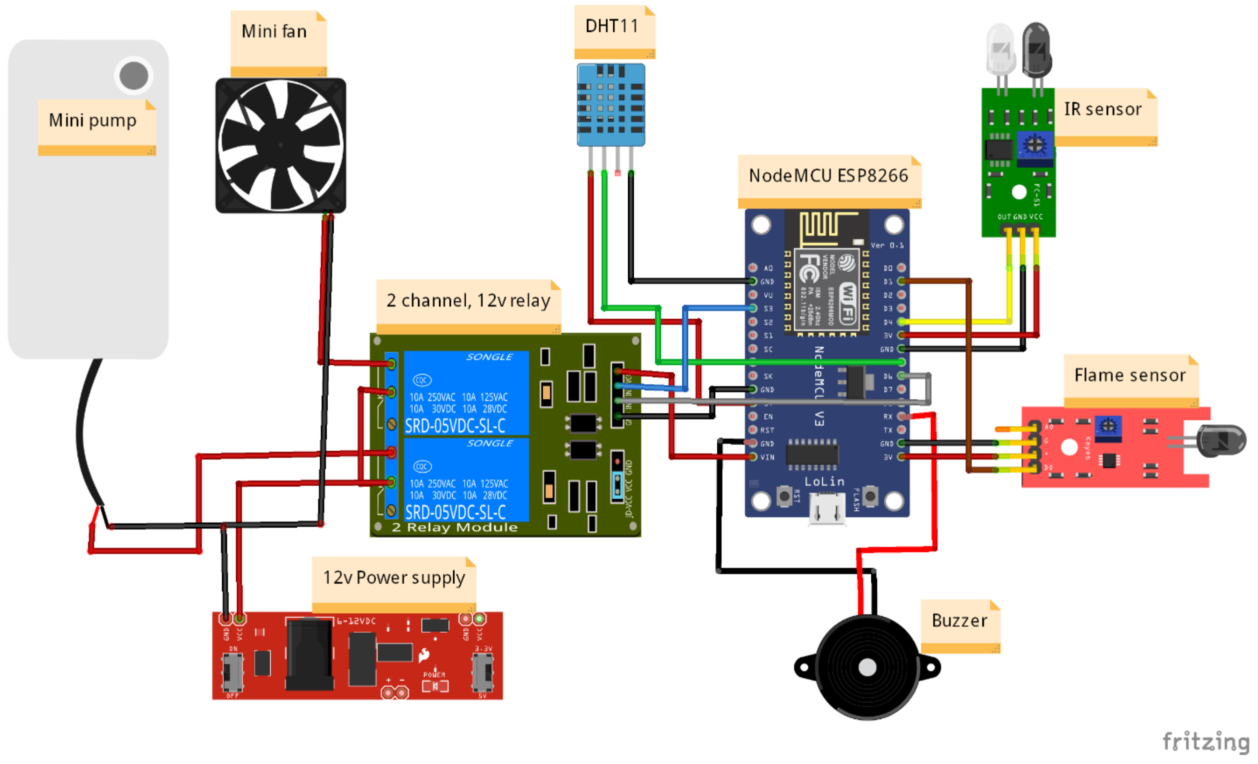


Figure 2.14 Circuit Diagram of Smart Factory

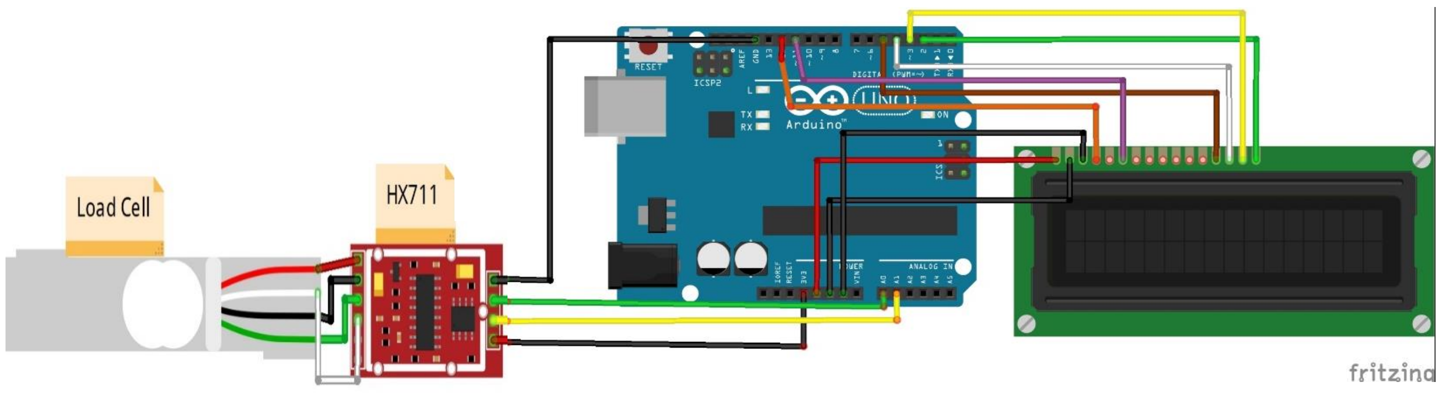


Figure 2.15 Circuit Diagram of Weighing Scale machine

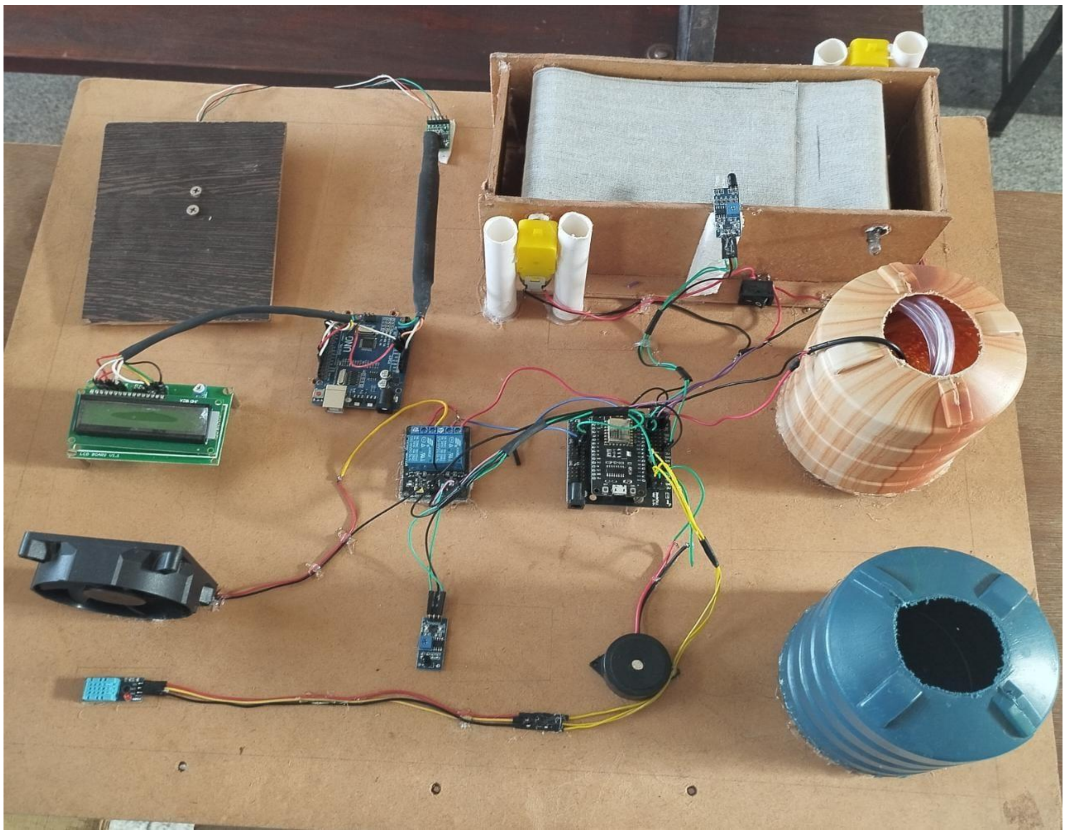


Figure 2.16 Smart Factory System Project Model

CHAPTER 3

3.1 APPROACH & METHODOLOGY

1. **Requirement Analysis:** We thoroughly understanding the specific needs and objectives of the project for Smart Factory system. This helps us understand the needful functions and determine the integration of sensors and cloud connectivity. The components that are used for our project as per requirement analysis they are ESP8266, Arduino Uno, Load cell, LCD display, Base connector , sensors, actuators, and etc.

2. **Technology Research:** We will more research to explore available technologies, sensor types, cloud platforms, and protocols. This research enables us to select the most suitable components and tools for the project. In our project Smart Factory system we implemented Thinger.io Cloud platform and Arduino IDE software platform, etc.

3. **System Design:** Based on the requirements analysis and technology research, we design the Smart Factory system in detailed. This involves the placement and configuration of sensors, Microcontrollers, Actuators, defining communication protocols, and outlining the integration with Thinger cloud services.

4. **Hardware Setup:** After all these three steps next to set the hardware components, such as Esp8266, Arduino, Conveyor belt, Load cell, HVAC system and communication modules. This step up includes proper wiring and compatibility between components, and testing to ensure proper functioning.

5. **Software Development:** Develop the necessary software components, including firmware for microcontrollers used for programming the Arduino and Esp8266, user interface software, and data handling algorithms. The software facilitates seamless communication between sensors, actuators, and the cloud platform.

6. **Testing and Optimization:** To test the software and hardware components, we conduct testing and optimization process to verify the reliability, accuracy, and performance of the

project Smart Factory system. This includes verifying the performance and functionality of sensor readings, efficiency of testing control mechanisms, and evaluating overall system functionality and they work as expected and meet the requirements of the project.

7. **Documentation:** We prepare comprehensive documentation, including system manuals, user guides, working principles, working procedures, references, used system versions, when the last update done, time period for maintenance and technical specifications. This documentation will serve as a reference for future maintenance, or system expansion.

8. **Deployment:** Once fully tested and documented, we deploy the system in the factory environment. We provide ongoing support and maintenance to address any issues and make necessary improvements.

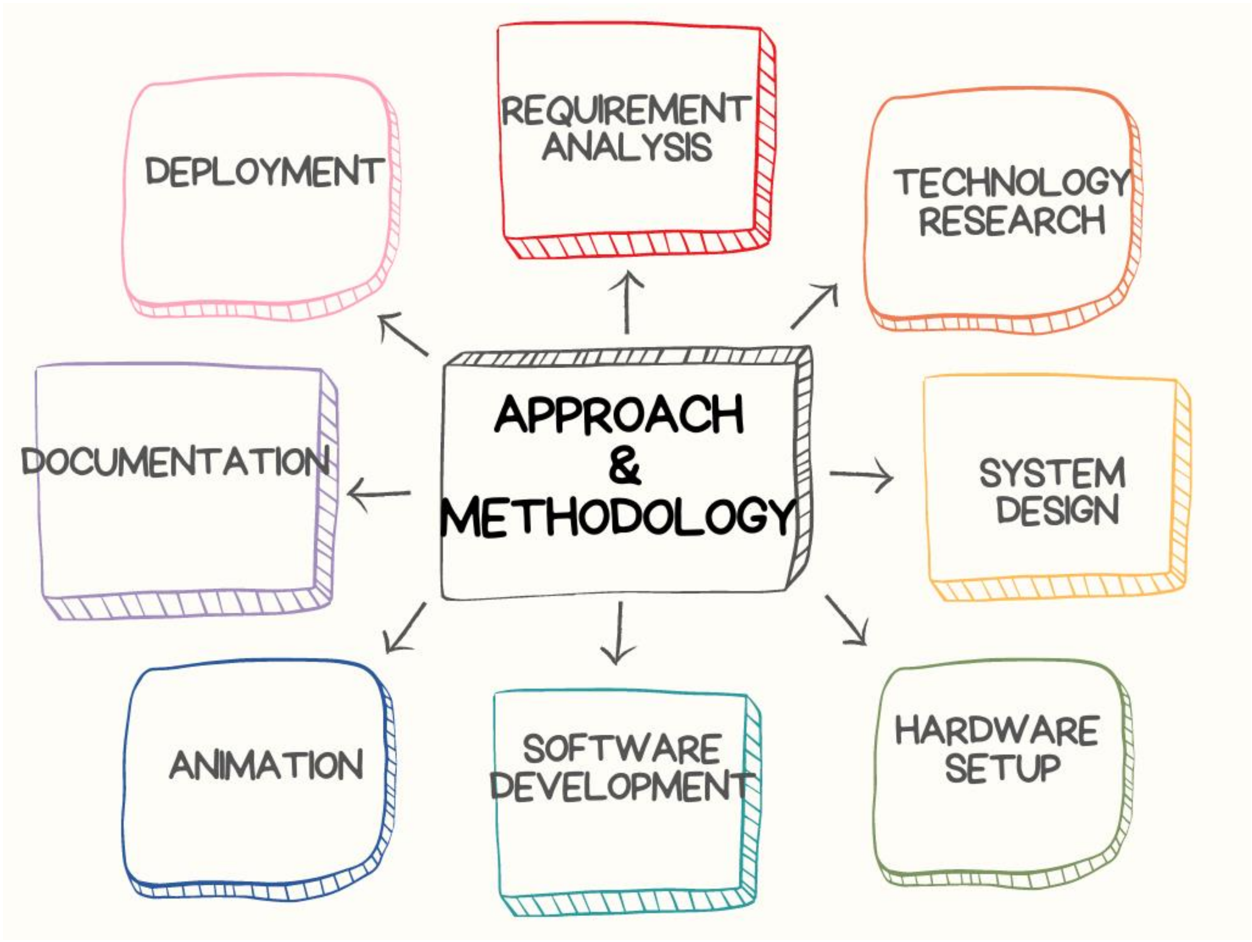


Figure 3.1 Block diagram of Approach and Methodology

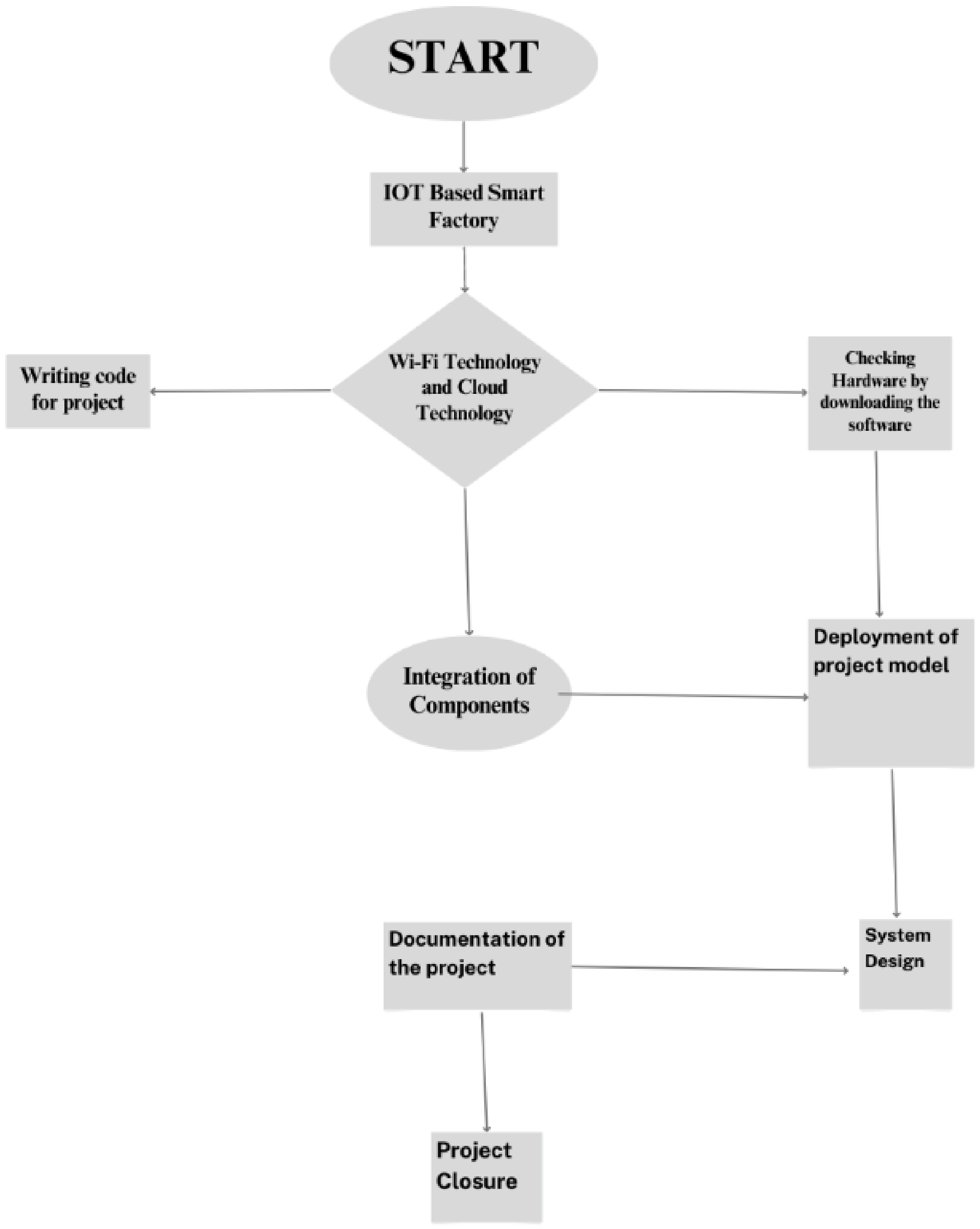

START
IOT Based Smart Factory
Wi-Fi Technology and Cloud Technology
Writing code for project
Checking Hardware by downloading the software
Integration of Components
Deployment of project model
System Design
Documentation of the project
Project Closure

Figure 3.2 Flow chart of Approach and Methodology

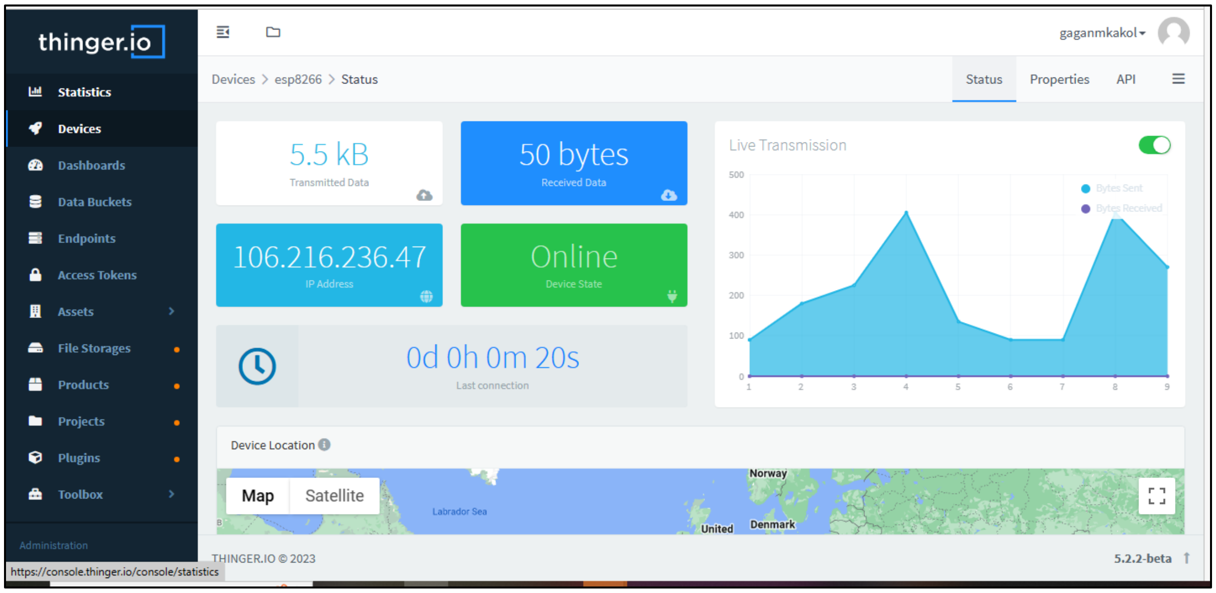
CHAPTER 4

4.1 TEST AND VALIDATION

Test and validation is crucial and important part of a project or system it helps as to find out any errors in the system. And solve them before deployment of the project.

4.1.1 Test and Validation for ESP8266

- Connect the ESP8266 to the power supply and check it boots up successfully.
- Write a test code to establish a Wi-Fi connection and send/receive data.
- The ESP8266 module successfully establishes a Wi-Fi connection to the specified network.
- Connect the GPIO pins with LED, buttons or any devices and check the pins correct working.
- Features tested: Wi-Fi connectivity, Data transmission and reception.
- No interference or significant issues observed during testing.
- Check that ESP8266 is compatible with required of your project like libraries, programming and cloud that you use for the project. Make that all the things mentioned above works perfectly in ESP8266.



5.5 kB
Transmitted Data

50 bytes
Received Data

106.216.236.47
IP Address

Online
Device State

0d 0h 0m 20s
Last connection

4.1 ESP8266 is connected to cloud via Wi-Fi

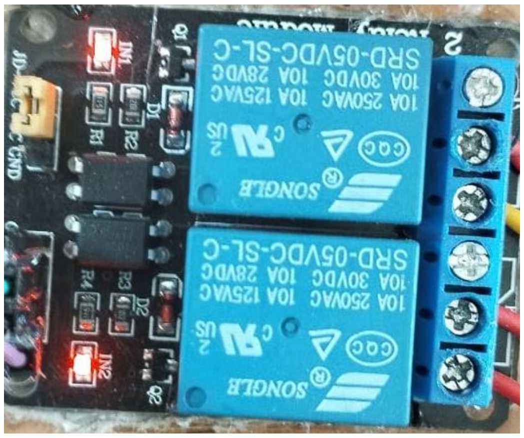
4.1.2 Test and Validation for Arduino Uno

- Connect the Arduino Uno to the computer and power source.
- Write a test code to perform basic operations, such as blinking an LED.
- Upload the test code to the Arduino Uno and verify the expected behavior.
- Test various input/output pins and verify their functionality.
- Evaluate the stability and reliability of the Arduino Uno during extended operation.
- All input/output pins function properly.
- No interference or significant issues observed during testing.

4.2.3 Test and Validation for Relay Module

- Apply the required supply voltage to the relay coil. Verify the switches that contacts form open to closed or closed to open when the coil is energized or de-energized
- Connect relay to suitable load example; Light or bulb, and also test the relay's ability to handle specified current and voltage rating of load without exceeding maximum capacity
- Measure the time taken by the relay to energize or de-energize the coil. Observe the time with the manufacturer's specification to ensure it works within the acceptable limits.
- Endurance testing of relay: repeatedly energize and de-energize the relay for extended period under operating condition. Make sure that the relay can handle more number of cycles without degradation or failure.
- Environmental testing of relay: test the relay in various condition like temperature, humidity. Make sure that relay can easily operate in these temperature condition and work perfectly.
- Apply minimum voltage that is needed to the relay in which relay works perfectly. Also test the relay by applying over voltage and low voltage make sure that it can handle both.

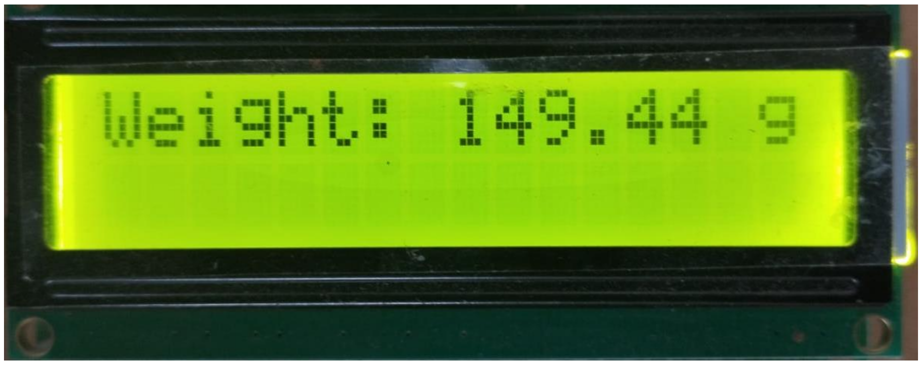
- Verify the relay with safety measures and conditions. Also check it for overvoltage, isolation protection against the electrical hazards.



4.2 Proper working of Relay Module

4.1.4 Test and Validation for Loadcell

- Connect the load cell to the appropriate interface circuitry and Arduino Uno.
- Write a test code to read and display weight measurements from the load cell.
- Calibrate the load cell to ensure accurate weight readings.
- Place known weights on the load cell and compare the readings with the expected values.
- Repeat the test with different weights to evaluate consistency and reliability.
- Features Tested: Weight measurement accuracy, Calibration functionality, Consistency and reliability of readings
- Consistent and reliable performance observed during testing.



```
Weight: 149.44 g
```

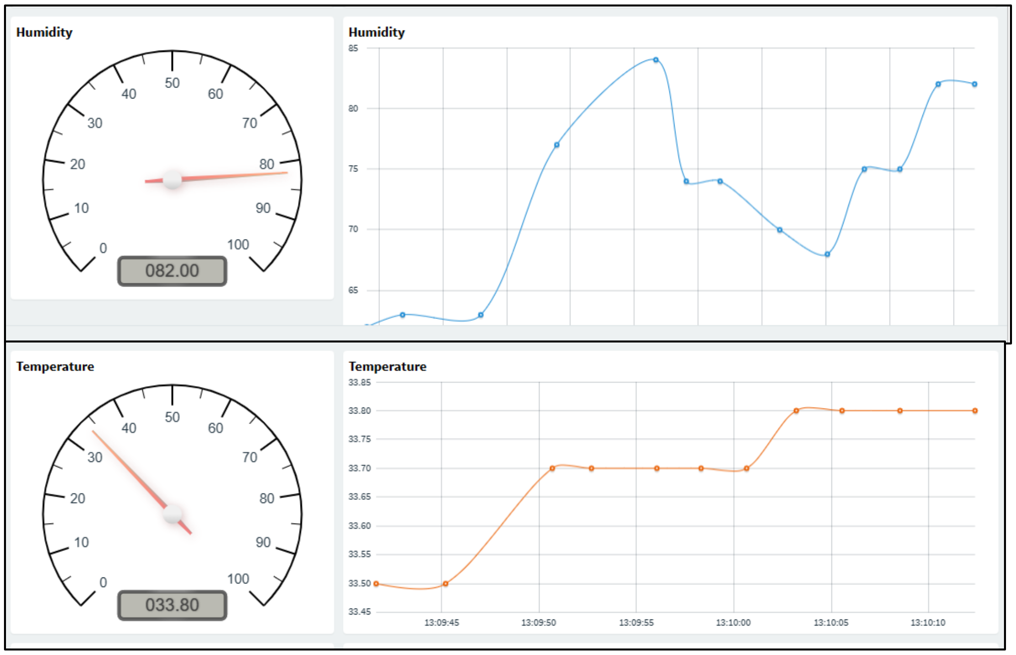
4.3 Proper working of Load cell sensor Module

4.1.5 Test and Validation for LCD Display

- Connect the LCD display to the Arduino Uno.
- Write a test code to display text and/or numerical values on the LCD.
- Verify the readability of the displayed information under various lighting conditions.
- Test different display functionalities, such as scrolling and clearing the display.
- Evaluate the stability and reliability of the LCD display during continuous operation.
- Features Tested: Display readability, Display functionalities
- No interference or significant issues observed during testing.

4.1.6 Test and Validation for DHT11 Sensor

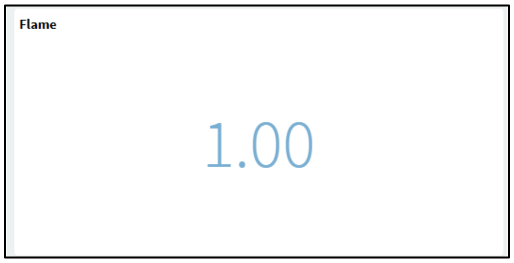
- Connect the DHT11 sensor to the ESP8266.
- Write a test code to read and display the temperature and humidity values.
- Place the sensor in a controlled environment with known temperature and humidity.
- Compare the readings from the DHT11 sensor with the expected values.
- Repeat the test multiple times to ensure consistent and accurate readings.
- Features Tested: Temperature sensing, Humidity sensing, Readings accuracy and consistency.
- The readings are consistent across multiple test runs.
- No interference or significant issues observed during testing.



4.4 Temperature & Humidity readings on DHT11 sensor

4.1.7 Test and Validation for Flame Sensor

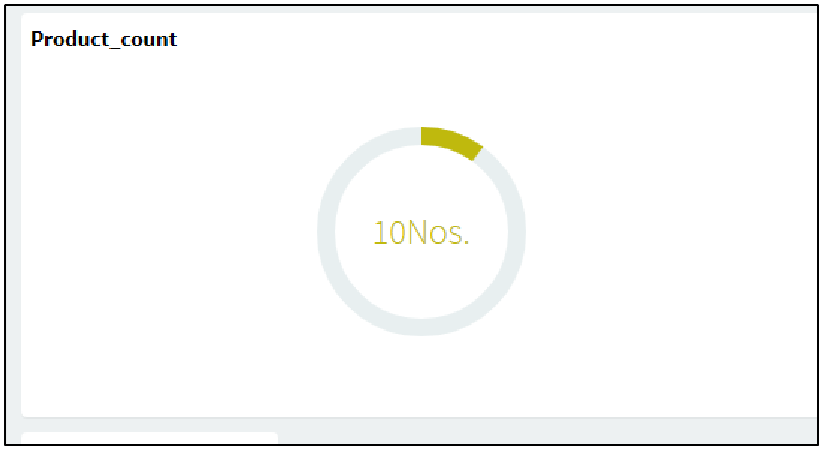
- Connect the flame sensor to the appropriate pins of ESP8266.
- Write a test code to detect and respond to flame presence.
- Present a flame source to the sensor and observe the response.
- Repeat the test with varying flame intensities and distances to evaluate detection accuracy and sensitivity.
- Ensure the sensor responds correctly by activating a connected device (e.g., water pump) upon flame detection.
- Features Tested: Flame detection accuracy, Sensitivity to flame intensity and distance, Triggering of connected device upon flame detection.
- Proper triggering of connected devices upon flame detection observed.



4.5 Flame detected on dashboard via Flame sensor

4.1.8 Test and Validation for IR Sensor

- Connect the IR sensor to the ESP8266.
- Write a test code to detect the presence of objects using the IR sensor.
- Place objects of different sizes and materials within the sensor's range and observe the detection.
- Evaluate the accuracy of object detection by comparing the sensor's output with the actual presence of objects.
- Test the sensor's response time and sensitivity to different objects.
- Features Tested: Object detection accuracy, Response time and sensitivity
- No interference or significant issues observed during testing.



4.6 Working of IR sensor counting the products

4.1.9 Test and Validation for Fan and Motor

- Connect the fan and motor to the appropriate pins of the ESP8266.
- Write a test code to control the activation and deactivation of the fan and motor.
- Verify that the fan and motor turn on/off as expected based on sensor inputs or programmed conditions.
- Test different speed settings for the fan to ensure proper control.
- Evaluate the stability and reliability of the fan and motor operation during extended periods of use.
- Features Tested: Activation and deactivation control, Speed control of the fan
- No interference or significant issues observed during testing.

4.1 Test and Validation of Arduino Code

```
#include <ThingerESP8266.h>

#include <DHT.h>


#define USERNAME "Enter your username"

#define DEVICE_ID "Enter your Device ID"

#define DEVICE_CREDENTIAL "Enter your Credential"


const int IRPIN = 2;     // IR sensor pin

#define FLAME_PIN 5  // Flame sensor pin

#define DHTPIN 14     // DHT11 sensor pin

#define Buzzer 3     // Buzzer pin

#define FANPIN 10     //Fan pin

#define MOTOR_PIN 12  // Motor pin

#define DHTTYPE DHT11      // DHT11 sensor type

DHT dht11(DHTPIN, DHTTYPE);  // initialize DHT11 sensor object


int count = 0;           // product count variable

int lastState = LOW;     // last IR sensor state

int currentState = LOW;  // current IR sensor state


#define SSID "Enter Wi-Fi Name"

#define SSID_PASSWORD "Enter Wi-fi Password"
```

```
ThingerESP8266 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

float temperature,humidity;

void setup() {
   Serial.begin(9600);
  pinMode(IRPIN, INPUT);
  pinMode(FLAME_PIN, INPUT);
  pinMode(Buzzer, OUTPUT);
  pinMode(FANPIN, OUTPUT);
  pinMode(MOTOR_PIN, OUTPUT);

  thing.add_wifi(SSID, SSID_PASSWORD);  // add wifi credentials
  thing["count"] >> [](pson& out) { out = count; };  // read product count

 thing["dht11"] >> [](pson& out) {
    out["temperature"] = temperature;
    out["humidity"] = humidity;
    };

 thing["flame"] >> [](pson& out) {
    out = digitalRead(FLAME_PIN);
  };
  thing["fan"] << digitalPin(FANPIN);

  thing["motor"] << digitalPin(MOTOR_PIN);  // read motor status

  dht11.begin();

}

void loop() {
  thing.handle();  // handle ThingSpeak communication
  temperature = dht11.readTemperature();  // read temperature
  humidity = dht11.readHumidity();
  int flame_value = digitalRead(FLAME_PIN);
  thing.stream(thing["dht11"]);

  lastState = currentState;
  currentState = digitalRead(IRPIN);

  if (lastState == LOW && currentState == HIGH)
  {
    count++;
    Serial.print("Product count: ");
    Serial.println(count);
  }
  if (temperature > 35 || humidity >70) {
    digitalWrite(FANPIN, 0);             // turn on fan
     pson data;
      data["temperature"] = temperature;
      data["humidity"] = humidity;
      thing.call_endpoint("HVAC_Notification", data);
  }else{
     digitalWrite(FANPIN, 1);
  }
  if (flame_value == 0) {
    digitalWrite(MOTOR_PIN, 0);  // turn on motor if flame is detected
     pson data;
      data["Flame"] = flame_value;
    thing.call_endpoint("Flame_Notification", data);
    digitalWrite(Buzzer, 1);
    delay(100);
    digitalWrite(Buzzer, 0);
    delay(100);
  } else {
    digitalWrite(MOTOR_PIN, 1);  // turn off motor if flame is not detected
  }
}
```

CHAPTER 5

5.1 BUSINESS ASPECTS

The smart factory system developed using ESP8266, Arduino Uno, sensors and actuators brings together various technologies to automate factory system and access control. The integration of these components into a seamless solution offers convenience, efficiency, and enhanced security compared to tradition factory.

Market Potential:

The IoT-based Smart Factory System with Sensor Integration and Cloud Connectivity project targets the manufacturing industry, which is witnessing a growing demand for automation and smart solutions.

Cost Savings:

By automating various tasks and processes, the project can lead to substantial cost savings for manufacturing companies.

Improved Quality and Compliance:

The implementation of sensor integration allows for real-time monitoring of critical parameters, ensuring quality control and compliance with industry standards.

Scalability and Flexibility:

The project's architecture is designed to be scalable and adaptable to the specific needs of different manufacturing environments.

Data-driven Decision Making:

The project's cloud connectivity enables the collection, storage, and analysis of vast amounts of data generated by the sensors.

Enhanced Customer Satisfaction:

By leveraging IoT technologies, the project empowers manufacturers to deliver products faster, with improved quality and customization options.

Regulatory Compliance:

The project takes into consideration the regulatory requirements applicable to the manufacturing industry, ensuring compliance with safety, environmental, and quality standards.

Business Growth and Sustainability:

By embracing smart factory solutions, companies can position themselves as innovative and forward-thinking.

Return on Investment (ROI):

While the initial investment in implementing the IoT-based Smart Factory System may be significant, the project's potential for cost savings, efficiency improvements, quality enhancement, and competitive advantage can yield a substantial return on investment over time.

5.2 FINANCIAL CONSIDERATIONS

The budget for the project would depend on various factors such as the scale of implementation, hardware and software costs, development resources, and any additional expenses related to research, testing, and documentation. But the budget of the components (ESP8266, Arduino Uno, LCD, IR, DHT11, Flame sensors, Relay, etc.) development tools and software licenses, server or hosting costs and any other miscellaneous expenses are mentioned in the **Table 2.1**

- Initial investment required for hardware, software, and infrastructure setup.
- Ongoing operational costs for maintenance, software updates, and connectivity.
- Potential cost savings through improved efficiency and resource utilization.
- Return on Investment (ROI) based on increased output and reduced expenses.
- Financial viability assessed through payback period and profitability projections.
- Funding options such as internal budget, external financing, or grants.
- Cost-benefit analysis to compare benefits against investment costs.
- Risk management plan to mitigate financial risks.
- Long-term sustainability and scalability of the project.
- Monitoring and evaluation of financial performance and benefits.

5.3 CONCLUSION AND RECOMMENDATION

In summary, the IoT-based Smart Factory System with Sensor Integration and Cloud Connectivity project offers great advantages to the manufacturing industry. By using Internet of Things technologies, the project aims to improve efficiency, product quality, and resource management in factories.

By seamlessly integrating sensors and connecting to the cloud, the project enables real-time monitoring of important factors like temperature, humidity, weight, and flame detection. This helps make better decisions, predict maintenance needs, and optimize processes for increased productivity and reduced downtime.

The project also provides valuable insights and analytics through cloud connectivity, allowing for informed decision-making and continuous process improvement. Storing and analyzing sensor data in the cloud offers a flexible and scalable solution for long-term data management.

- The project represents a significant advancement in the manufacturing industry, leveraging the power of Internet of Things technologies to enhance operational efficiency and optimize resource utilization.
- By seamlessly integrating sensors and establishing cloud connectivity, the project enables real-time monitoring of crucial parameters such as temperature, humidity, weight, and flame detection, leading to proactive decision-making and process optimization.

Recommendations:

- Regular Maintenance: Ensure regular maintenance and calibration of sensors and hardware components to ensure accurate data collection and reliable system performance.
- Software Updates: Continuously update and optimize the software system to incorporate new features, enhance security, and address any identified bugs or vulnerabilities.
- Data Security: Implement robust data security measures to protect sensitive information collected by the sensors, including encryption, access controls, and regular security audits.
- Data Analysis and Insights: Regularly analyze the collected data to identify patterns, trends, and areas for process improvement. Use data analytics tools and techniques to gain actionable insights for better decision-making.
- Training and Support: Provide comprehensive training and support to the staff members responsible for operating and maintaining the system. Ensure they have the necessary knowledge and skills to effectively utilize the system and troubleshoot any issues.
- Integration with ERP Systems: Explore integration possibilities with existing enterprise resource planning (ERP) systems to streamline operations, optimize resource allocation, and enable seamless data exchange between different systems.

REFERENCE

Journals

- B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee and B. Yin, "Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges," in IEEE Access, vol. 6, pp. 6505-6519, 2018, doi: Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges | IEEE Journals & Magazine | IEEE Xplore
- P. A. Okeme, A. D. Skakun and A. R. Muzalevskii, "Transformation of Factory to Smart Factory," 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), St. Petersburg, Moscow, Russia, 2021, pp. 1499-1503, doi: Transformation of Factory to Smart Factory | IEEE Conference Publication | IEEE Xplore

Online Reference

- ESPRESSIF ESP8266 - Thinger.io Documentation
- GitHub - thinger-io/Arduino-Library: Arduino Library for connecting devices to thinger.io #IoT.
- Arduino Uno Pinout, Specifications, Pin Configuration & Programming (components101.com)
- NodeMCU ESP8266 Pinout, Specifications, Features & Datasheet (components101.com)
-
- DHT11 Sensor Pinout, Features, Equivalents & Datasheet (components101.com)

- IR Sensor Module Pinout, Features & Datasheet (components101.com)
- Flame Sensor : Working, Types, and Its Applications (elprocus.com)
- Buy Dual Shaft BO Series DC Motor Straight - 100 RPM Online at the Best Price (robu.in)
- Buzzer : Working, Types, Circuit, Advantages & Disadvantages (elprocus.com)
- Arduino - Wikipedia
- OVERVIEW - Thinger.io Documentation