CLOUD BASED LEDGER

A project report submitted in partial fulfillment of the requirements for the 6th semester of the degree of

BACHELOR OF TECHNOLOGY
in ELECTRONICS ENGINEERING

Submitted by

Raghav Gupta
(21001017047)
Rajnish Singh
(21001017049)
Uday Sharma
(21001017065)

Under the Supervision of

DR. ROHIT TRIPATHI

DR. SUNIL JADAV

Department of Electronics Engineering
Faculty of Engineering and Technology
J.C. Bose University of Science and Technology, YMCA,
Faridabad, Haryana-121006

1. INTRODUCTION

Our innovative project aims to revolutionize shopkeeping operations with the introduction of an IoT calculator equipped with cloud integration capabilities. In today's fast-paced retail landscape, efficiency, accuracy, and accessibility are paramount for businesses to thrive. Our solution addresses these needs by combining traditional calculator functionalities with advanced cloud-based transaction logging, empowering shopkeepers to streamline their operations and make data-driven decisions with ease.

The Need for Innovation: Shopkeepers face a multitude of challenges in their day-to-day operations, from managing inventory and processing transactions to maintaining accurate financial records. Traditional calculators, while essential, lack the capacity to seamlessly integrate with modern business technologies, often resulting in manual data entry, human errors, and limited accessibility to transaction records. Recognizing these pain points, our project seeks to bridge the gap between conventional tools and contemporary business requirements by introducing a calculator solution that leverages cloud technology.

There have been many phone applications introduced to the market to solve this issue, but due to a lack of robust hardware implementation the apps were not convenient to use. The phone apps were cumbersome to use with the overlapping of the functions of a phone themselves.This device aims to merge the functions of a calculator and a digital ledger using IoT. The device is capable of storing daily transanctions of small shopkeepers onto a Google Spreadsheet stored in Google Cloud using Https function of the ESP8266 library. The device aims at making the logging of transanctions of shopkeepers easy and not dependent on other smart device and built it in a handy calculator itself. The device works both as a simple 4 function calculator and also logs the transanctions when instructed.

The device is a low powered device which can be operated via a battery and uses an open-source microcontroller(ESP8266) for low power working. The Google Cloud is used for storing the data in a Google Spreadsheet via the Google App Script API.

2. OVERVIEW OF PRINCIPLES INVOLVED

1. **Embedded programming using ESP8266 :** The ESP 8266 microcontoller is used to program the interface and functionality for the Keypad and the LCD display. The ESP8266 is programmed using Arduino IDE with ESP8266 library installed.

2. **HTTP protocol to send data to cloud from device :** ESP8266 has inbuilt WiFi capabilities which allows it to act as a client or a server so it can send or receive data via the HTTP protocol.

3. **Membrane Keypad :** This device is used to input data to a microcontoller. It is programmable and can be used to input various instructions. We are using a 4X4 keypad which used a grid system to tell the microcontoller which key is pressed.

4. **I2C LCD display :** The I2C module is used to show data to a LCD display via the I2C protocol. This reduces the number of pins to be connected to interface the LCD screen.

5. **Google App Script :** The Google app script is used to deploy a simple backend app written in Javascript which is an API to fetch data from the NODEMCU to the Google Spreadsheet in the Google Cloud.

TABLE OF CONTENTS

1. Introduction	1
2. Overview of principles Involved	2
3. Block Diagram	3
4. Flow Chart	4
5. Circuit Diagram	5
6. Hardware Required	6
7. Code for NODEMCU	9
8. Code for Google App Scripts	13
9. Cost of Production	14
10. Advantages	15
11. Disadvantages	16
12. Future Prospect	17

3. BLOCK DIAGRAM

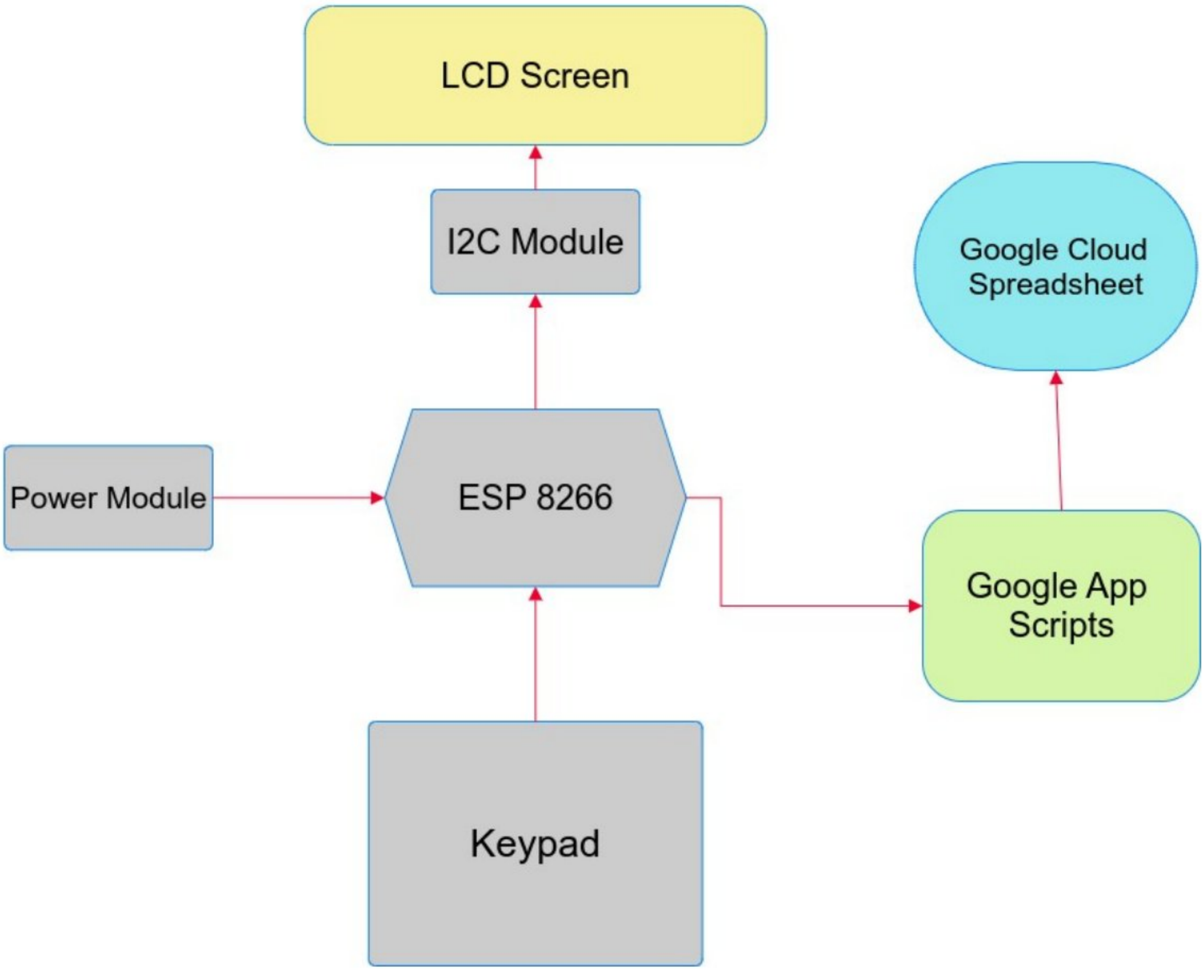

Fig. 1: Block Diagram of system

4. PROCESS FLOW CHART

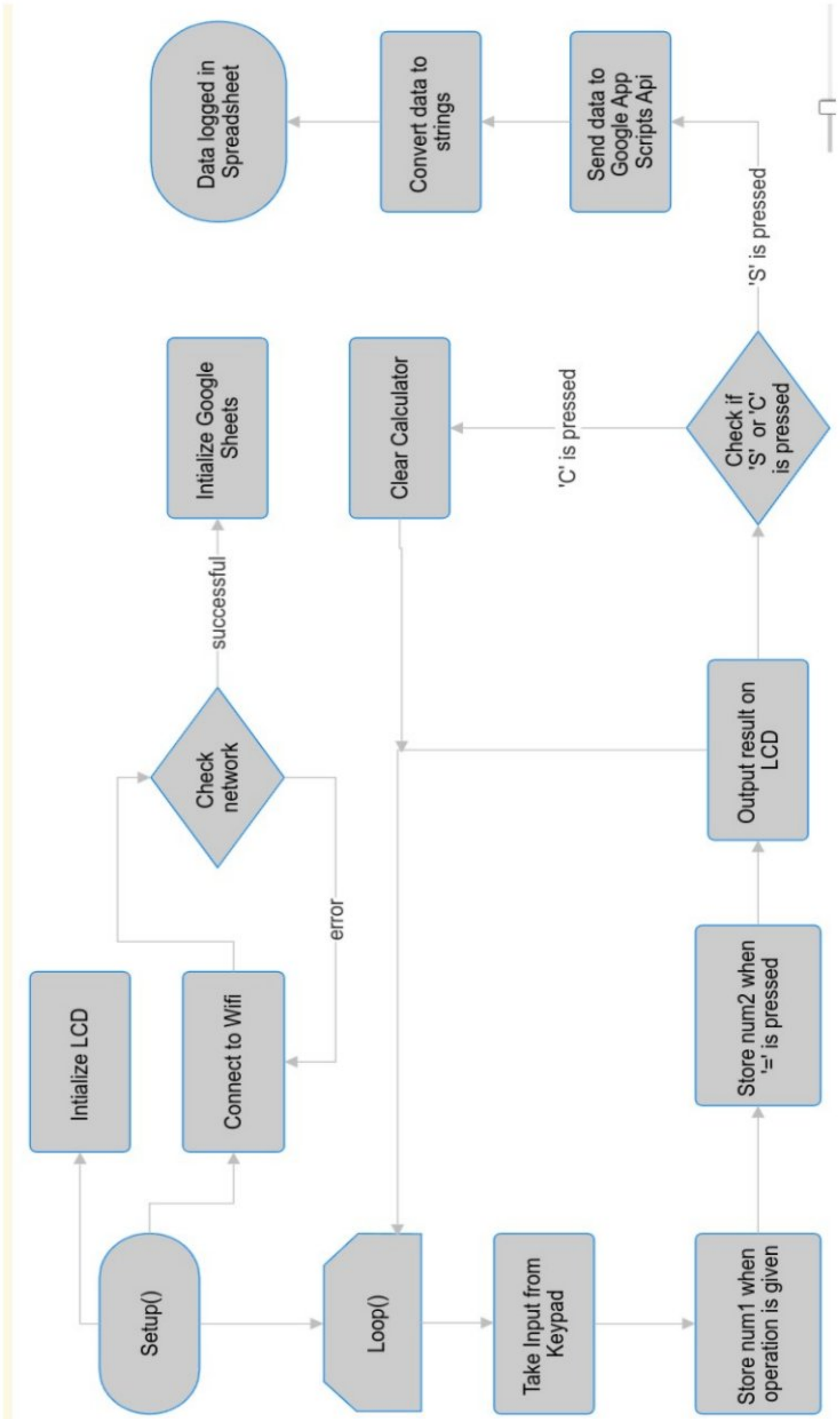

Fig. 2: Process Flow Chart

5. WIRING DIAGRAM

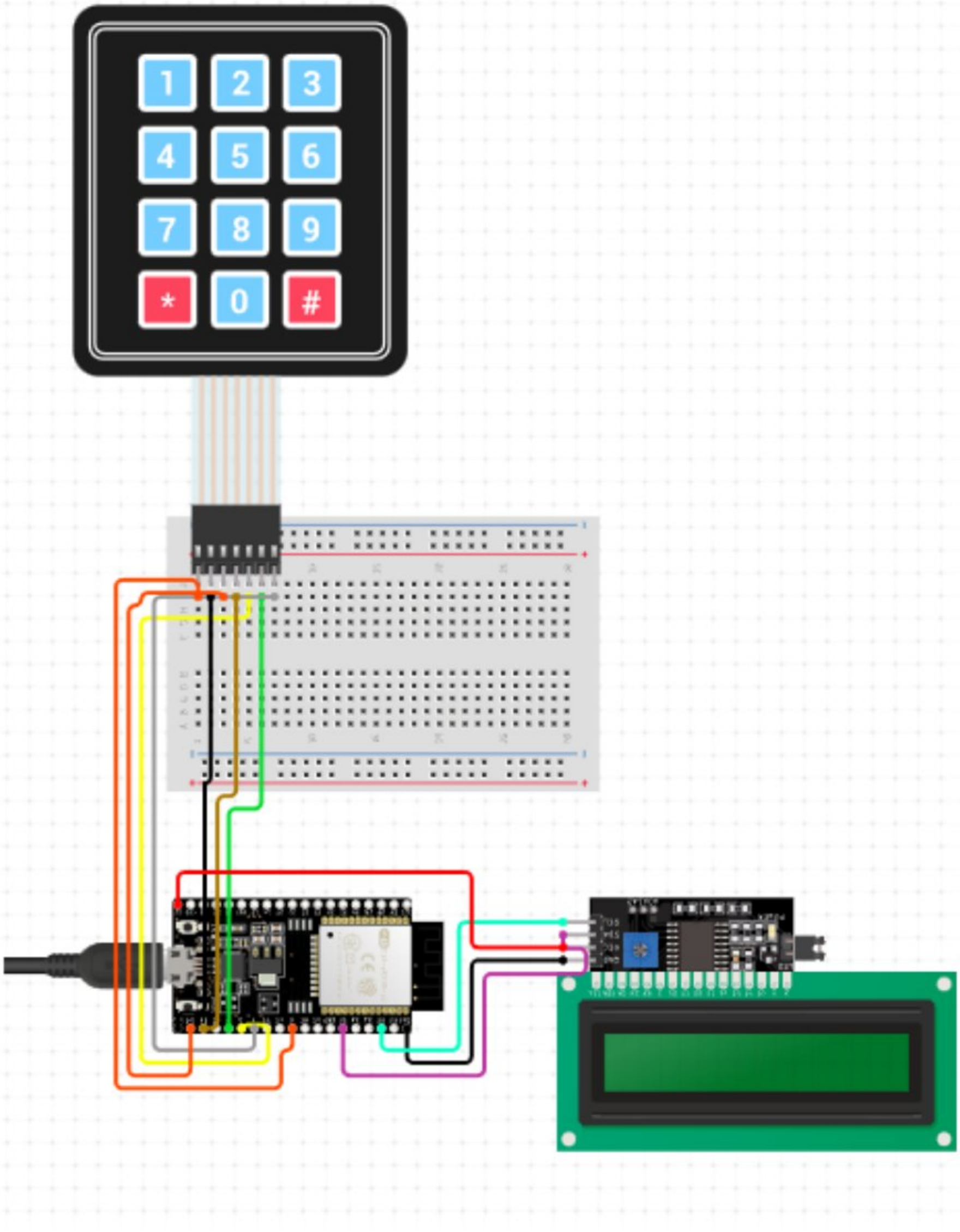

Fig.3: Wiring Diagram

6. HARDWARE REQUIREMENTS

NODEMCU ESP8266:-

The NodeMCU ESP8266 is a compact development board combining the ESP8266 Wi-Fi module with a microcontroller unit. It supports Lua scripting and Arduino IDE programming, making it versatile for Internet of Things (IoT) and embedded projects. Equipped with GPIO pins, a USB-to-serial interface, and built-in Wi-Fi connectivity, it simplifies prototyping of smart devices and remote control applications. The ESP 8266 microcontoller is used to program the interface and functionality for the Keypad and the LCD display. The ESP8266 is programmed using Arduino IDE with ESP8266 library installed. ESP8266 has inbuilt WiFi capabilities which allows it to act as a client or a server so it can send or receive data via the HTTP protocol.

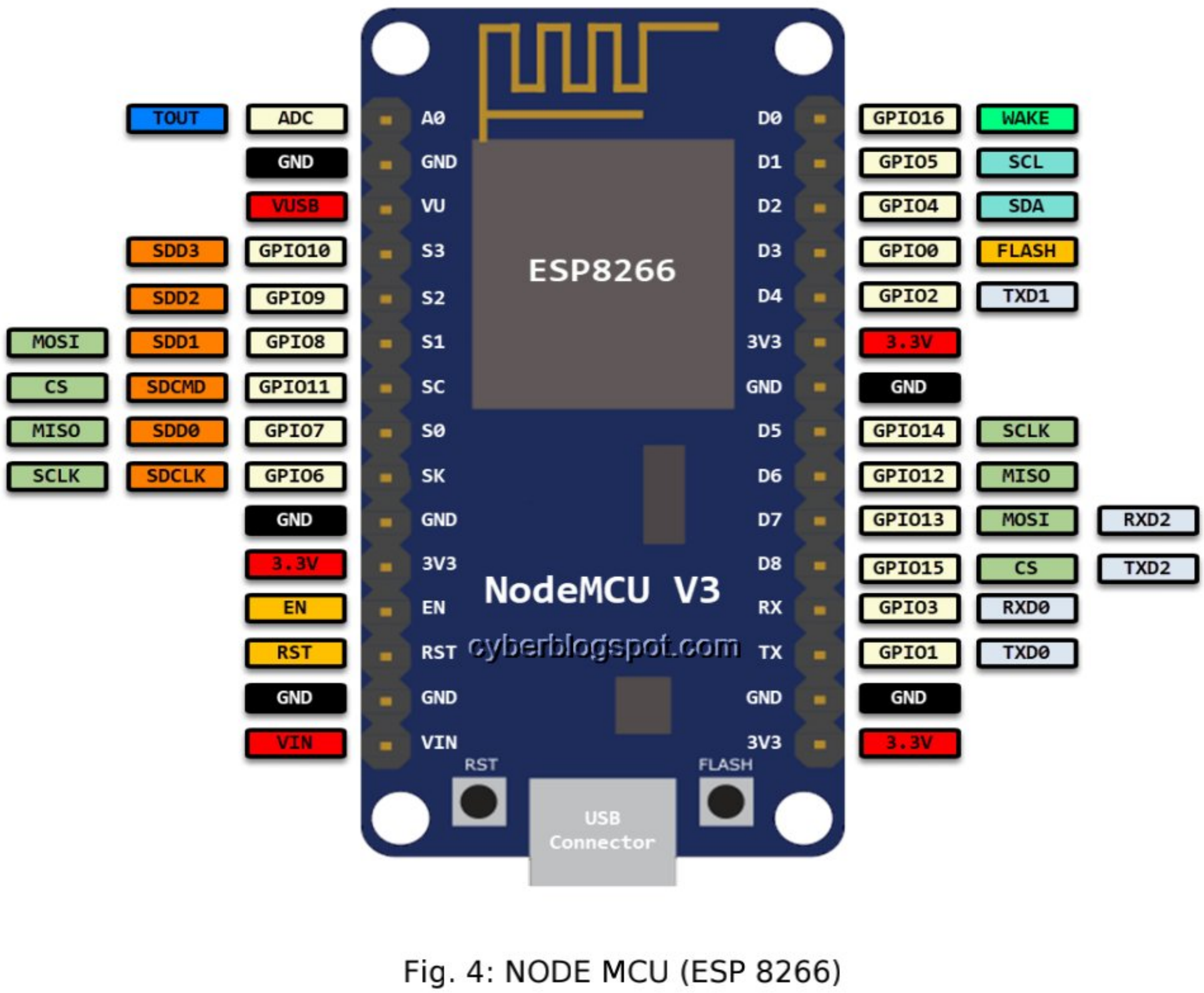

Fig. 4: NODE MCU (ESP 8266)

I2C MODULE:-

An I2C module, also known as an I2C interface or I2C controller, facilitates communication between microcontrollers and peripheral devices using the Inter-Integrated Circuit (I2C) protocol. It typically consists of hardware components and associated software libraries that enable serial communication over short distances with minimal wiring. I2C modules commonly feature multiple I2C channels, allowing connection to various sensors, displays, EEPROMs, and other I2C-compatible devices. By providing a standardized interface and protocol, I2C modules simplify the integration of peripherals into embedded systems, making them popular for IoT, robotics, and other electronic projects.

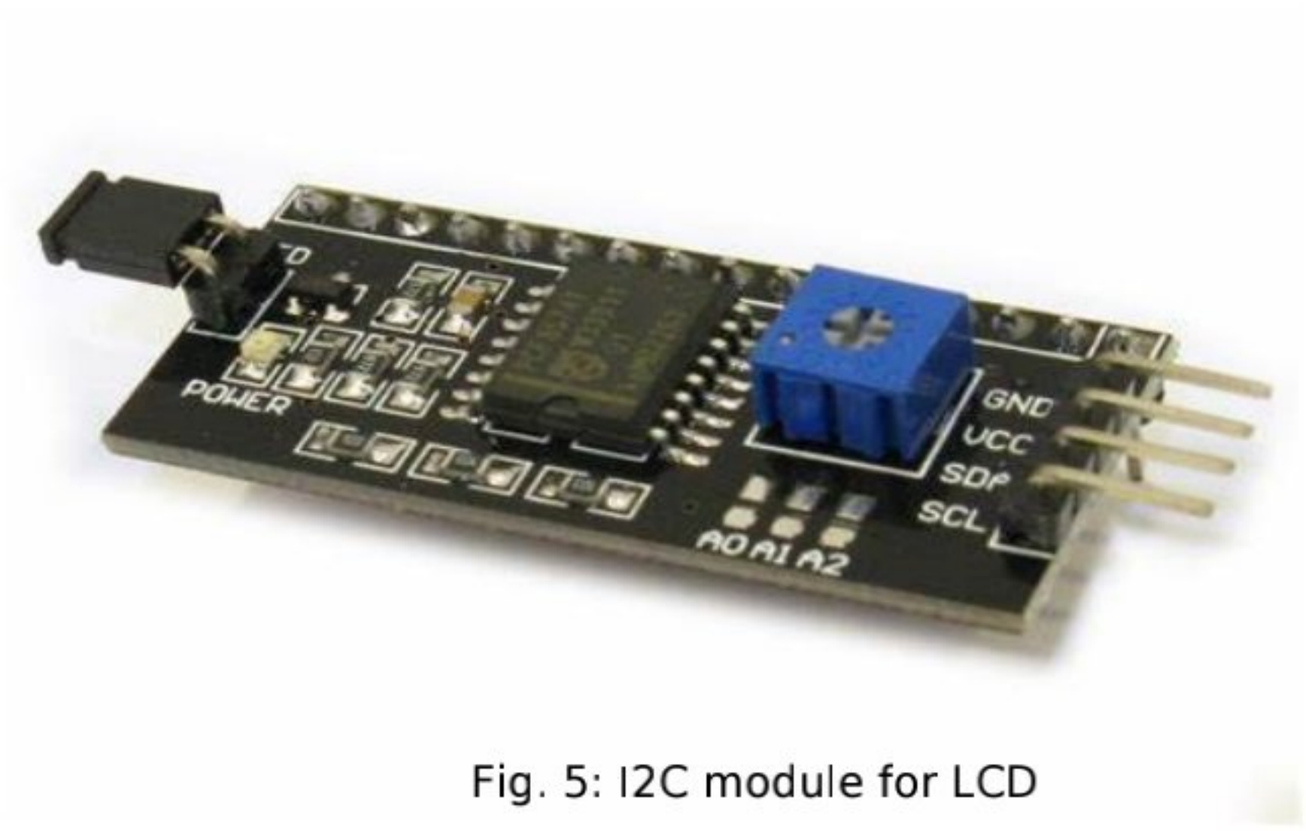

Fig. 5: I2C module for LCD

LCD DISPLAY 16*2:-

A 16x2 LCD display is a liquid crystal display module with a grid of 16 characters in each of its 2 rows, allowing for the display of alphanumeric characters and symbols. These displays typically feature an integrated controller such as the Hitachi HD44780 or equivalent, which simplifies interfacing with microcontrollers. Users can control the display by sending commands and data over a parallel or serial interface, commonly using protocols like I2C or SPI. 16x2 LCD displays are commonly used in various electronic projects, including DIY gadgets, instrumentation, and embedded systems, due to their simplicity, affordability, and ease of use.

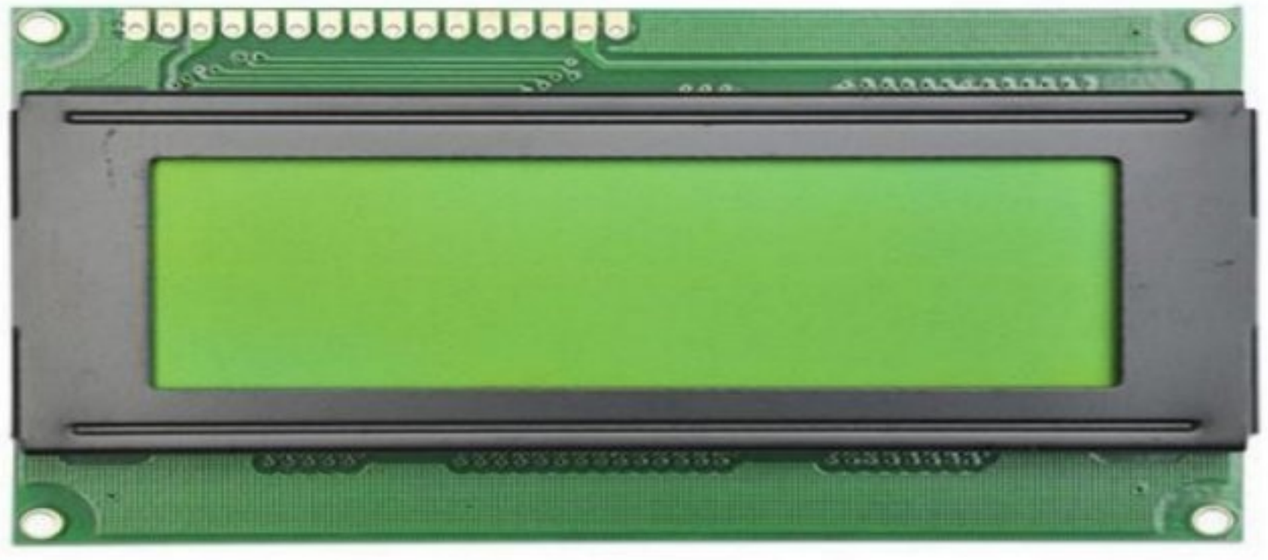

Fig. 6: 16X2 LCD

KEYPAD MODULE:-

A keypad module is an input device used for accepting user input in the form of alphanumeric characters, numbers, or symbols. It typically consists of a grid of buttons arranged in rows and columns, with each button corresponding to a specific character or function. The keypad module is interfaced with a microcontroller or other electronic devices, allowing users to enter data or commands

Fig. 7: 4X4 Keypad

7. PROGRAM FOR ESP 8266

```
#include "TRIGGER_WIFI.h"
#include "TRIGGER_GOOGLESHEETS.h"
#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

char column_name_in_sheets[ ][6] = {"val1","val2","val3"}; /*1. The Total no of column depends on how many value you have created in Script of Sheets;2. It has to be in order as per the rows decided in google sheets*/
String Sheets_GAS_ID = "AKfycbyxlhfIIF8hp5hh0IHMjM50mYGeO-IdoR0DWkqzGvJ56lRztggfAAah0AFYreib5Qi2"; /*This is the Sheets GAS ID, you need to look for your sheets id*/
int No_of_Parameters = 3;


const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
{'1', '2', '3', '+'},
{'4', '5', '6', '-'},
{'7', '8', '9', 'C'},
{'*', '0', '=', 'D'}
};
byte rowPins[ROWS] = {0, 2, 14, 12};
byte colPins[COLS] = {13, 15, 3, 1};

// Create keypad object
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

// Define LCD display settings
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Calculator variables
String input = ""; // String to store user input
float num1 = 0; // First operand
float num2 = 0; // Second operand
//float result=0;
char operatorSymbol = ' '; // Operator symbol
bool resultShown = false; // Flag to indicate if result is shown
```

```
void setup() {
// Initialize serial communication
Serial.begin(9600);

lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
lcd.print("Auto Ledger");
delay(1000);
lcd.clear();

// Wait for serial connection to be established
while (!Serial);

// Connect to Wi-Fi
WIFI_Connect("SamsungA10", "raghu007"); // Provide your Wi-Fi SSID and password
// Initialize Google Sheets communication
Google_Sheets_Init(column_name_in_sheets, Sheets_GAS_ID, No_of_Parameters); // Set
column names, GAS ID, and number of parameters
// Initialize LCD display
}

void loop() {

float a = 0, b = 0, c = 0;
char key = keypad.getKey();

// If a key is pressed
if (key != NO_KEY) {
// Check if the result is shown, if so, clear the display
if (resultShown) {
lcd.clear();
resultShown = false;
}

// Check if the key is a digit or a decimal
point if (isdigit(key) || key == '.') {
// Append the key to the input string
input += key;
lcd.print(key);
}
```

```
// Check if the key is an operator (+, -, *,
/) else if (key == '+' || key == '-' ) {
// Store the input string as the first operand
num1 = input.toFloat();
// Store the operator symbol
operatorSymbol = key;
// Reset the input string
input = "";
// Print the operator symbol on the LCD display
lcd.setCursor(15,  0);
lcd.print(operatorSymbol);
}
// Check if the key is the equals sign (=)
else if (key == '=') {
// Store the input string as the second operand
num2 = input.toFloat();
// Reset the input string
input = "";
// Perform the calculation based on the operator symbol
float result;
switch (operatorSymbol) {
case '+':
a=result = num1 + num2;Data_to_Sheets(No_of_Parameters, a, b, c);
break;
case '-':
a=result = num1 - num2;Data_to_Sheets(No_of_Parameters, a, b, c);
break;
// case '*':
// result = num1 * num2;
// break;
// case '/':
// result = num1 / num2;
// break;
default:
// Display error message if an invalid operator is used
lcd.setCursor(0, 1);
lcd.print("Error");
delay(1000);
lcd.clear();
break;
}
// Print the result on the LCD display
lcd.setCursor(0,  1);
```

```
lcd.print("Result: ");
lcd.print(result); resultShown = true;
}
// Check if the key is the 'C' key for clear
else if(key=='C'){
Data_to_Sheets(No_of_Parameters, a, b, c);
input = "";
num1 = 0;
num2 = 0;
operatorSymbol = ' ';
resultShown = false;
lcd.clear();

}
}
}
```

8. PROGRAM FOR GOOGLE APP SCRIPT

```
function doGet(e) {
Logger.log( JSON.stringify(e) ); // view parameters
var result = 'Ok'; // assume success
if (e.parameter == 'undefined') {
result = 'No Parameters';
}
else {
var sheet_id = '1PEOzxklHrF4NrGu4KwRfTEI2OIcqUlfO2U7DDFqP81Q'; // Spreadsheet ID
var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet(); // get Active sheet
var newRow = sheet.getLastRow() + 1;
var rowData = [];
d=new Date();
rowData[0] = d; // Timestamp in column A
rowData[1] = d.toLocaleTimeString(); // Timestamp in column A
for (var param in e.parameter) {
Logger.log('In for loop, param=' + param);
var value = stripQuotes(e.parameter[param]);
Logger.log(param + ':' +
e.parameter[param]); switch (param) {
case 'val1': //Parameter 1, It has to be updated in Column in Sheets in the code, orderwise
rowData[2] = value; //Value in column A
result = 'Written on column
A'; break;
case 'val2': //Parameter 2, It has to be updated in Column in Sheets in the code, orderwise
rowData[3] = value; //Value in column B
result += ' Written on column
B'; break;
case 'val3': //Parameter 3, It has to be updated in Column in Sheets in the code, orderwise
rowData[4] = value; //Value in column C
result += ' Written on column
C'; break;
default:
result = "unsupported parameter";
}
}
Logger.log(JSON.stringify(rowData));
// Write new row below
var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
newRange.setValues([rowData]);
}
// Return result of operation
return ContentService.createTextOutput(result);
}
function stripQuotes( value ) {
return value.replace(/^["']|['"]$/g, "");
}
```

9. COST OF PRODUCTION

COMPONENTS	QUANTITY	COST (INR)
ESP 8266	1	400 INR
LCD Display	1	150 INR
I2C module	1	100 INR
Keypad 4X4	1	150 INR
Misc.		200 INR
Total		1000 INR

Table 1: Cost of Production

10. ADVANTAGES

1. **Streamlined Operations:** Integration of transaction logging with calculator functions reduces the need for separate devices or manual record-keeping, streamlining shopkeeping operations.

2. Real-time Data Access: Cloud-based transaction logging enables shopkeepers to access transaction records in real-time from any location with internet connectivity, facilitating remote monitoring and management.

3. Accuracy and Error Reduction: Automated transaction logging minimizes human errors associated with manual data entry, ensuring greater accuracy in financial records and calculations.

4. Enhanced Security: Robust encryption protocols employed in cloud storage ensure the security and confidentiality of transaction data, mitigating the risk of unauthorized access or data breaches.

5. Data Analysis and Insights: Access to transaction data analytics enables shopkeepers to gain valuable insights into sales trends, inventory management, and customer behavior, empowering data-driven decision-making and strategic planning.

11. DISADVANTAGES

1.Dependency on Internet Connectivity: Cloud integration requires a stable internet connection for accessing transaction records and performing calculations, posing a challenge in areas with unreliable internet infrastructure or during network outages.

2. Cost Considerations: Implementation and subscription costs associated with cloud storage services may pose a financial barrier for small-scale businesses or shopkeepers operating on tight budgets.

3. Learning Curve: Adoption of new technology may require training for shopkeepers unfamiliar with cloud-based systems, potentially leading to initial resistance or slower uptake among users.

4. Data Privacy Concerns: Storing transaction data on the cloud raises concerns regarding data privacy and compliance with regulatory requirements, necessitating robust data protection measures and adherence to data protection laws.

5. System Downtime and Maintenance: Dependence on cloud service providers introduces the risk of system downtime or maintenance disruptions, potentially impacting shopkeeping operations and accessibility to transaction records.

12. FUTURE PROSPECTS

1. **Integration with Point-of-Sale (POS) Systems:** Future iterations of the cloud-integrated shopkeeper calculator may offer seamless integration with POS systems, enabling comprehensive management of sales, inventory, and financial data within a unified platform.

2. **Machine Learning and Predictive Analytics:** Integration of machine learning algorithms with transaction data analytics could enable predictive analytics capabilities, allowing shopkeepers to forecast sales trends, anticipate customer demand, and optimize inventory management strategies.

3. **Mobile Payment Integration:** Integration with mobile payment platforms could facilitate secure and convenient payment processing directly through the calculator device, enhancing the customer checkout experience and reducing reliance on traditional payment terminals.

4. **Blockchain Technology for Enhanced Security:** Adoption of blockchain technology for transaction logging could further enhance data security and integrity, providing immutable records of financial transactions while ensuring transparency and accountability in shopkeeping operations.

5. **API Integration for Third-Party Applications:** Provision of application programming interfaces (APIs) could allow seamless integration with third-party applications such as accounting software, inventory management systems, or customer relationship management (CRM) platforms, enabling shopkeepers to leverage a wider range of tools and functionalities within their business ecosystem.