VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELGAUM

A MINI-PROJECT REPORT ON

SMART MEDICINAL BOX

Submitted by:

Mohammed bathish ks

Muhammad Ifaaz Beig B

Mohammed Affan Ahamed

Mohammed Fouzan

Prem kumar

Pavan.R

Under the Guidance of:

Prof.RIDHWAN ABDULLA M S

(Asst. Professor, Department of ECE)

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

P. A. COLLEGE OF ENGINEERING

Nadupadavu, Mangaluru – 574153, D. K. KARNATAKA 2024 – 2025

TABLE OF CONTENTS

INTRODUCTION:

In today's fast-paced world, many individuals—especially the elderly and patients with chronic illnesses—struggle to remember their daily medication schedules. Missing a dose can lead to serious health complications. To address this issue, this project presents a **Smart Medicine Reminder Box** based on **IoT (Internet of Things)** technology.

The system is designed to remind users to take their medicine at a specific time using a buzzer and monitor whether the medicine has been taken using an IR sensor. A Real-Time Clock (RTC) module is used to keep accurate track of the time. The project is powered by an ESP8266 microcontroller, which is connected to the internet via Wi-Fi.

Furthermore, the system communicates with the **ThingSpeak** platform, allowing users or caretakers to:

- **Set or change** the medicine reminder time remotely.
- **Track** whether the medicine has been taken or missed.

This project aims to improve medication adherence and support remote health monitoring, making it a useful tool for patients and healthcare providers alike.

- **Uses of the Smart Medicine Reminder System**

1. **Medication Reminder**
   It alerts users with a buzzer at a specific time (e.g., 10:30 AM) to remind them to take their medication on time, reducing the chances of missed doses.
2. **Elderly and Patient Care**
   Especially helpful for elderly people or patients with memory-related issues who may forget to take their medicine regularly.
3. **Remote Health Monitoring**
   Using ThingSpeak integration, caregivers and doctors can monitor whether medicine has been taken or not—even from a remote location.
4. **Dynamic Time Updates via IoT**
   The reminder time can be updated remotely using the ThingSpeak website, eliminating the need for manual changes in the hardware or code.
5. **Low-Cost and Easy to Use**
   Designed using basic components like ESP8266, IR sensor, RTC, buzzer, and LCD—making it an affordable solution for homes and clinics.
6. **Scalable Design**
   The system can be extended to support multiple medicine compartments and different schedules (e.g., morning, afternoon, night).

2.Hardware and software requirements

1.1 Hardware components:

(a) ESP8266 (NodeMCU)

Fig. 1.1.1 ESP8266 (NodeMCU)

The ESP8266 is a low-cost Wi-Fi-enabled microcontroller chip used in embedded systems. It allows devices to connect to the internet and communicate wirelessly. It features digital input/output pins, an analog input, UART, SPI, and I2C interfaces. The chip is ideal for Internet of Things (IoT) applications due to its built-in Wi-Fi capabilities.
→ In this project, the ESP8266 is used as the brain of the system. It controls all components, handles time checking, sends/receives data from ThingSpeak, and triggers the buzzer at the correct time.

(b) RTC Module (DS3107)

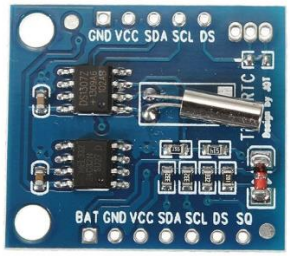


Fig. 1.1.2 RTC Module (DS3107)

he DS3231 is a real-time clock (RTC) module that keeps accurate time and date using an integrated crystal oscillator and temperature sensor. It continues timekeeping even when the main power is disconnected, using its onboard coin cell battery.
→ In this project, it is used to keep track of the current time so that the buzzer can be triggered exactly at the medicine reminder time (e.g., 10:30 AM).

(c) Buzzer

Fig. 1.1.3 Buzzer

A buzzer is an audio signaling device that emits a sound when voltage is applied. It is commonly used to alert users of certain events or conditions through sound.
→ In this project, the buzzer acts as a medicine reminder alarm. It rings at the set time (e.g., 10:30 AM) to alert the user to take their medicine. (d) Jumper wires

(d) Jumper Wires

Fig. 1.1.4 jumper wires

Jumper wires are simple insulated wires used to make quick and temporary connections between components in a circuit.
→ In this project, they are used to connect the ESP8266, sensors, and other modules on the breadboard.

(e) Transistor(BC547)

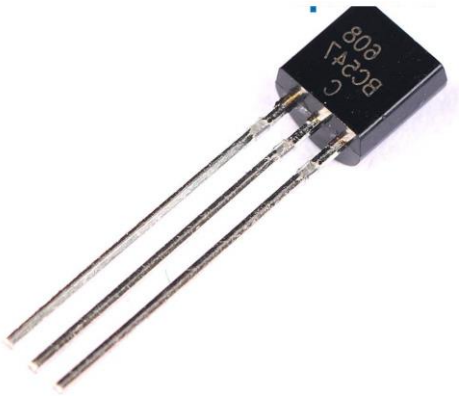


Fig. 1.1.5 Transistor(BC547)

A transistor is a semiconductor device that can act as a switch or amplifier. When used as a switch, it allows low-voltage control of higher-current devices.
→ In this project, the transistor is used to control the buzzer. Since the buzzer may require more current than the ESP8266 can provide, the transistor acts as a switch to safely operate the buzzer..

(f) IR Sensor

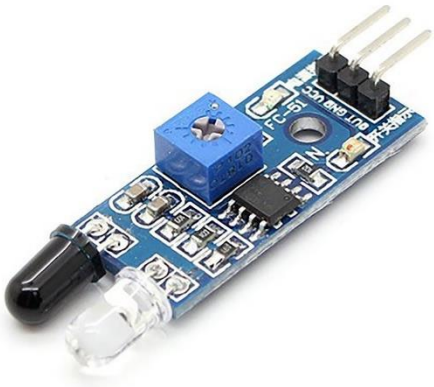


Fig 1.1.6 IR Sensor

An Infrared (IR) sensor detects objects by emitting infrared light and sensing its reflection. If an object is in front of the sensor, the reflected IR light is detected, indicating presence or movement.
→ In this project, the IR sensor is used to detect if the medicine has been picked up or not. It helps determine whether the user has taken the medicine when the buzzer goes off.

(g) Resistor

Fig 1.1.7 Resistor

A resistor is a passive electrical component that limits the flow of electrical current in a circuit. It protects sensitive components from receiving too much current.
→ In this project, resistors are used to limit current to the IR sensor, buzzer, and transistor to prevent damage and ensure stable operation

(h)Breadboard

Fig 1.1.8 Breadboard

A breadboard is a solderless prototyping tool used to build and test electronic circuits quickly and efficiently. It allows components to be connected without the need for soldering, making circuit design and testing flexible and easy to modify.
→ In this project, the breadboard is used to set up the entire circuit temporarily, helping us test and make changes before finalizing the hardware

ccc

Software requirements:

(a)Arduino IDE software:

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated

Development Environment. The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '. Ino'

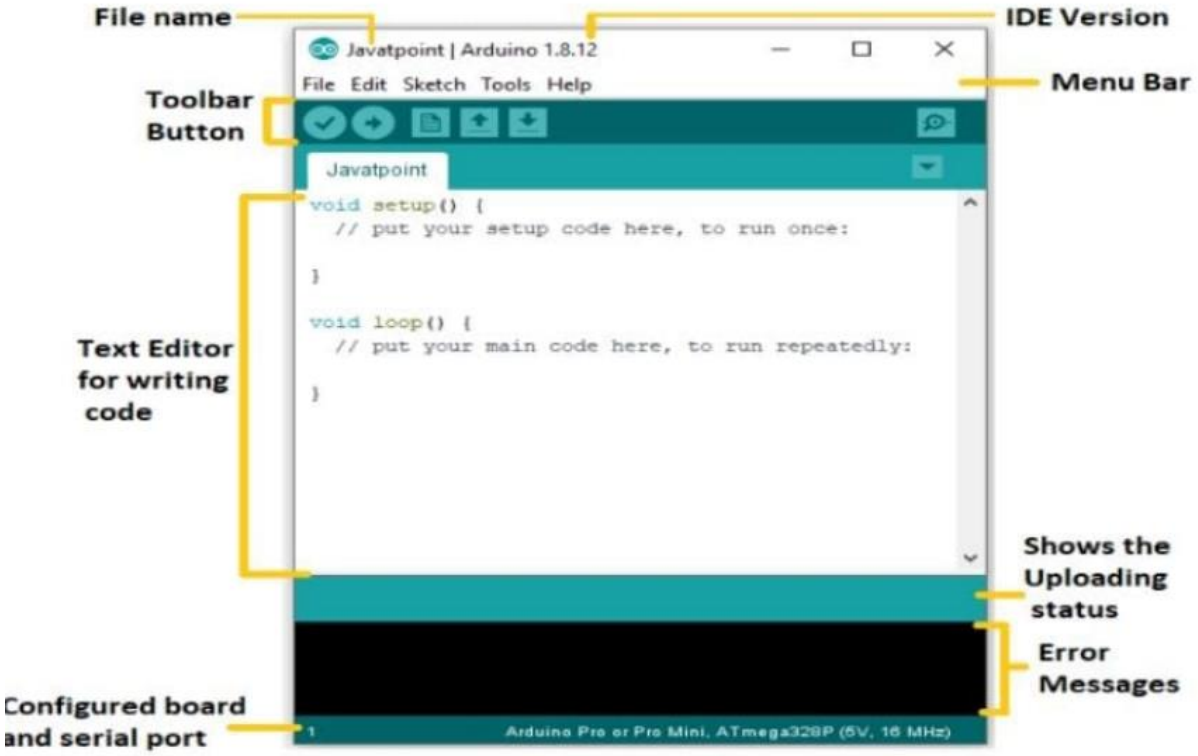


Fig. 1.2.1 Arduino software tool

(b)ThingSpeak Web Platform

To enable cloud-based data monitoring and remote analysis, the project utilizes [ThingSpeak](https://thingspeak.com), a cloud-based IoT analytics platform. The ESP8266 microcontroller sends and receives data through ThingSpeak using designated **API keys** (read and write), allowing real-time communication between the medicine reminder device and the cloud.

ThingSpeak provides the following functionalities in the project:

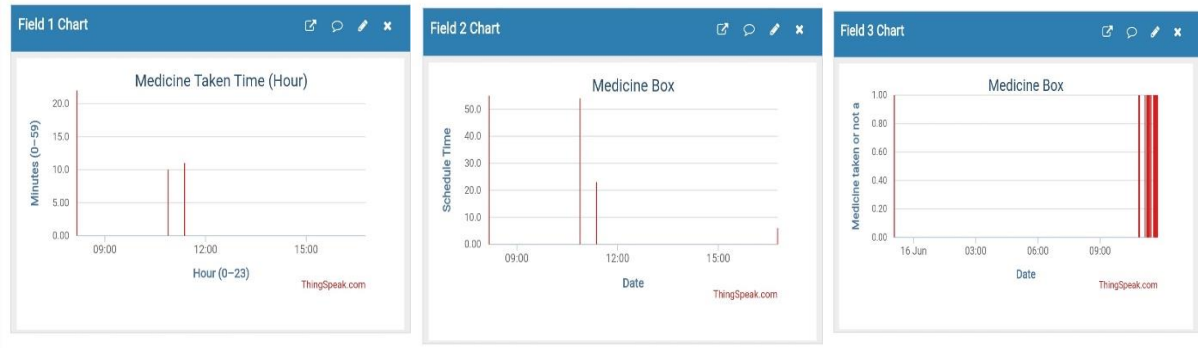


Fig. 1.2.2 ThingSpeak Web Platform

Field 1: Medicine Taken Time (Hour)

Records the hour and minute when medicine was taken, based on IR sensor detection. It helps track if there was any **delay** compared to the scheduled time.

Field 2: Scheduled Time (Minutes)

Holds the scheduled time for medicine intake, which the ESP8266 reads from ThingSpeak. This enables **custom time updates** from the cloud.

Field 3: Medicine Status

Indicates whether the medicine was taken (`1`) or not (`0`). This allows caregivers to monitor **compliance** remotely.

API Key Use:

1.Write Key: Sends data (Fields 1 & 3) to ThingSpeak.

2.Read Key: Reads the schedule (Field 2) from ThingSpeak.

These keys are generated in the channel's **API Keys** section and used in the microcontroller code securely.

Visual graphs on ThingSpeak help monitor user activity, improve health adherence, and track real-time medicine intake.

4. SOFTWARE DESCRIPTION:

Source Code:

```
// IoT Medicine Reminder Final Code

// Purpose: Set a medicine time from ThingSpeak and buzz at that time if medicine is not taken.


#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <Wire.h>

#include "RTClib.h"

const char* ssid = "ENTER_YOUR_WIFI_NAME";      // Your Wi-Fi/Hotspot Name

const char* password = "ENTER_YOUR_WIFI_PASSWORD";  // Your Wi-Fi Password

const String readAPIKey = "ETL01GN5PQC8J82E";     // ThingSpeak Read API Key

const String writeAPIKey = "A2M5V70I6M6K2ZJ8";    // ThingSpeak Write API Key

const int channelID = 2974161;                  // ThingSpeak Channel ID

const int irSensorPin = D6;   // Pin for IR sensor

const int buzzerPin = D5;     // Pin to control transistor connected to buzzer

RTC_DS1307 rtc;

WiFiClient client;

void setup() {

  Serial.begin(115200);

  pinMode(irSensorPin, INPUT);

  pinMode(buzzerPin, OUTPUT);
```

```
digitalWrite(buzzerPin, LOW);
Wire.begin();
if (!rtc.begin()) {
  Serial.println("❌ Couldn't find RTC module");
  while (1);
}
if (!rtc.isrunning()) {
  Serial.println("✅ RTC is NOT running, setting time now...");
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
}
// Connect to Wi-Fi
WiFi.begin(ssid, password);
Serial.print("Connecting to WiFi");
int retry = 0;
while (WiFi.status() != WL_CONNECTED && retry < 20) {
  delay(500);
  Serial.print(".");
  retry++;
}
if (WiFi.status() == WL_CONNECTED) {
  Serial.println("\n✅ WiFi connected!");
} else {
  Serial.println("\n❌ Failed to connect to WiFi!");
}
```

```
}
void loop() {
  DateTime now = rtc.now();
  int currentHour = now.hour();
  int currentMinute = now.minute();
  Serial.print("\n🕒 Current Time: ");
  Serial.print(currentHour);
  Serial.print(":");
  Serial.println(currentMinute);
  int scheduledHour = fetchFromThingSpeak(1);
  int scheduledMinute = fetchFromThingSpeak(2);
  Serial.print(" 📥 Scheduled Hour from ThingSpeak: ");
  Serial.println(scheduledHour);
  Serial.print(" 📥 Scheduled Minute from ThingSpeak: ");
  Serial.println(scheduledMinute);
  if (scheduledHour == currentHour && scheduledMinute == currentMinute) {
    int irValue = digitalRead(irSensorPin);
    Serial.print(" 📊 IR Sensor State: ");
    Serial.println(irValue);
    if (irValue == LOW) {
      Serial.println(" ✅ Medicine taken. Sending to ThingSpeak.");
      sendMedicineStatusToThingSpeak(1); // 1 = taken
    } else {
      Serial.println(" ⏰ Buzzing! Medicine not taken.");
```

```
    ringBuzzer(30); // 30 seconds
    sendMedicineStatusToThingSpeak(0); // 0 = not taken
   }
   delay(60000); // Wait a minute to prevent repeated triggering
  }
  delay(5000); // Check every 5 seconds
}
int fetchFromThingSpeak(int fieldNumber) {
  HTTPClient http;
  String url = "http://api.thingspeak.com/channels/" + String(channelID) +
"/fields/" + String(fieldNumber) + "/last.txt?api_key=" + readAPIKey;
  http.begin(client, url);
  int httpCode = http.GET();
  if (httpCode == 200) {
   String payload = http.getString();
   http.end();
   return payload.toInt();
  } else {
   Serial.print("❌ Failed to fetch field ");
   Serial.print(fieldNumber);
   Serial.print(" from ThingSpeak. HTTP Code: ");
   Serial.println(httpCode);
   http.end();
   return -1;
  }
```

```
}
void sendMedicineStatusToThingSpeak(int status) {
  HTTPClient http;
  String writeURL = "http://api.thingspeak.com/update?api_key=" +
writeAPIKey + "&field3=" + String(status);
  http.begin(client, writeURL);
  int httpCode = http.GET();
  if (httpCode > 0) {
    Serial.print(" 📤 ThingSpeak Response: ");
    Serial.println(httpCode);
  } else {
    Serial.println("❌ Failed to update ThingSpeak");
  }
  http.end();
}
void ringBuzzer(int durationSeconds) {
  for (int i = 0; i < durationSeconds * 5; i++) {
    digitalWrite(buzzerPin, HIGH);
    delay(100);
    digitalWrite(buzzerPin, LOW);
    delay(100);
  }
}
```

3. METHODOLOGY

1 Hardware Description:

The below block diagram represents the working of the *Medicine Reminder Box.When the system is powered on, the ESP8266 microcontroller connects to the internet and fetches the scheduled medicine time from the *ThingSpeak* platform.

Based on the time received:

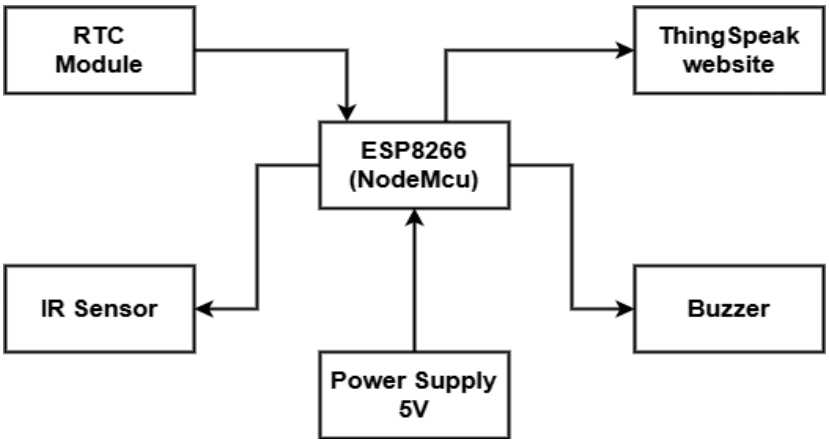


Fig. 2.1 Block diagram

1.If the current time matches the scheduled time, the buzzer is triggered to alert the user.

2.An IR sensor checks whether the medicine has been picked up from the box.

3.All events — including the scheduled time, actual pickup time, and whether the medicine was taken — are sent back to ThingSpeak for monitoring and record-keeping.

This system ensures real-time monitoring, scheduling flexibility, and helps in tracking medicine intake habits effectively.

The circuit consists of the following components:

1.ESP8266 NodeMCU: Acts as the central controller and Wi-Fi module.

2.RTC Module (Real-Time Clock): Maintains accurate time even during power resets.

3.IR Sensor: Detects if the medicine has been taken from the box.

4.Buzzer: Sounds the alert at the scheduled medicine time.

5.OLED or LCD Display (optional): Displays messages (not used in final version).

6.Transistor & Resistor: Controls buzzer activation.

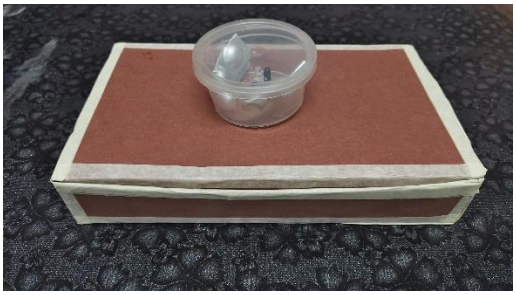
7.ThingSpeak Cloud Platform: Used to read medicine time and write intake status.

All connections are made carefully to GPIO pins of the ESP8266 and powered through a 5V USB power source. The IR sensor is aligned with the medicine compartment to detect motion, and the data is uploaded to ThingSpeak using API keys.

4. RESULT & DISCUSSION

1. Snapshot of the Project:

The circuit was assembled as per the circuit and block diagrams using an ESP8266 microcontroller. The required components such as IR sensor, RTC module and buzzer were connected correctly.
The program was uploaded to the ESP8266 using Arduino IDE, and the system was powered by a 5V USB power source for safe and reliable operation.
The device is then enclosed in a neatly designed brown and white box, housing all the electronic components securely to ensure portability and usability.



2.Result:

The Medicine Reminder Box project successfully reminds users to take their medication on time using a buzzer triggered by real-time data. The ESP8266 fetches scheduled time from the ThingSpeak cloud platform and activates the buzzer accordingly.

An IR sensor detects whether the medicine has been taken, and this data is recorded and visualized on the ThingSpeak dashboard. The three fields used track:

1.Actual time medicine is taken,

2.Scheduled medicine time,

3.And whether medicine was taken (1) or missed (0).

The system enables real-time monitoring and data logging, which can be accessed remotely. This project improves medicine intake adherence, especially helpful for elderly users or those under long-term medication. The cloud-based integration provides flexibility to adjust schedules without reprogramming the device.

5. CONCLUSION

The Smart Medicine Reminder System is an IoT-based solution developed to help users take their medicines on time. By integrating an ESP8266 microcontroller with an RTC module, IR sensor, buzzer, and optional LCD display, the system alerts users at a set time using an alarm. The IR sensor checks if the medicine was taken, and the data is sent to the ThingSpeak platform for monitoring.

One of the standout features of the project is that the medicine reminder time can be updated directly from the ThingSpeak website, allowing for flexible schedule management without needing to modify the code. This makes the system suitable for both home use and elderly care settings.

The Smart Medicine Reminder System is an IoT-based solution developed to help users take their medicines on time. By integrating an ESP8266 microcontroller with an RTC module, IR sensor, buzzer, and optional LCD display, the system alerts users at a set time using an alarm. The IR sensor checks if the medicine was taken, and the data is sent to the ThingSpeak platform for monitoring.

This project allows users to remotely manage medicine reminder times through the ThingSpeak website, offering ease of customization without altering the microcontroller's code. This cloud integration makes it possible to monitor medicine intake history and adjust schedules as needed, which is especially useful for caregivers or users with multiple medications.

Overall, the project is cost-effective, reliable, and easy to use, offering a smart solution to improve medication adherence and reduce missed doses. It also demonstrates how IoT can be applied in real-world healthcare challenges.