Project Report

Student Name: Anurup Jyoti Krishna Panda **UID: 23MCC20010**
Branch: MCA(CC & DevOps) **Section/Group : 23MCD2(A)**
Semester: 4 **Date of Performance: 18.04.2025**
Subject Name: Internet Of Things **Subject Code: 23CAH-702**

Aim: AUTOMATED MEDICINE DISPENSER USING NODEMCU

Objectives:

1. Timed Reminders for Medication via Telegram
2. Automatic Tablet Dispensing
3. Sensor-based User Interaction

Components Required: (Times New Roman-14)

Sno	Name of Component	Qty.
1.	NodeMCU (ESP8266)	1
2.	Ultrasonic Sensor	1
3.	Servo Motor	1
4.	LED	3
5.	Buzzer	3
6.	Breadboard	1

Details of Components:

1. NodeMCU (ESP8266):
NodeMCU is an open-source IoT platform that includes firmware and a development board with Wi-Fi capability. It is based on the ESP8266 microcontroller, which allows for wireless communication with platforms like Telegram. It serves as the brain of the project, managing all inputs, outputs, and cloud interactions.

2. Ultrasonic Sensor (HC-SR04):
This sensor is used to detect the presence of a hand under the dispenser using sound waves. It measures the distance by emitting ultrasonic waves and calculating the time taken for the echo to return. If the detected distance is within a set threshold, it triggers the tablet dispensing mechanism.

3. Servo Motor (SG90):
A servo motor is used to physically dispense the tablet. Controlled through PWM signals from the NodeMCU, it rotates to a specific angle (like 90° or 180°) to drop the tablet and then returns to its original position for the next cycle.

4. LEDs (Red, Green, Yellow):
LEDs are used for visual indicators. The Red LED indicates system power ON status, the Green LED signals a medicine alert, and the Yellow LED can optionally be used for status/reset indications.

5. Buzzer:
Buzzers are used to provide audible alerts. When it's time for the user to take their medicine, the buzzer is activated along with the LED to draw attention.

6. Breadboard:
The breadboard is used to make non-permanent connections between electronic components during prototyping. It allows easy adjustments and wiring of all sensors and modules without soldering.

Block Diagram of Designed Model:

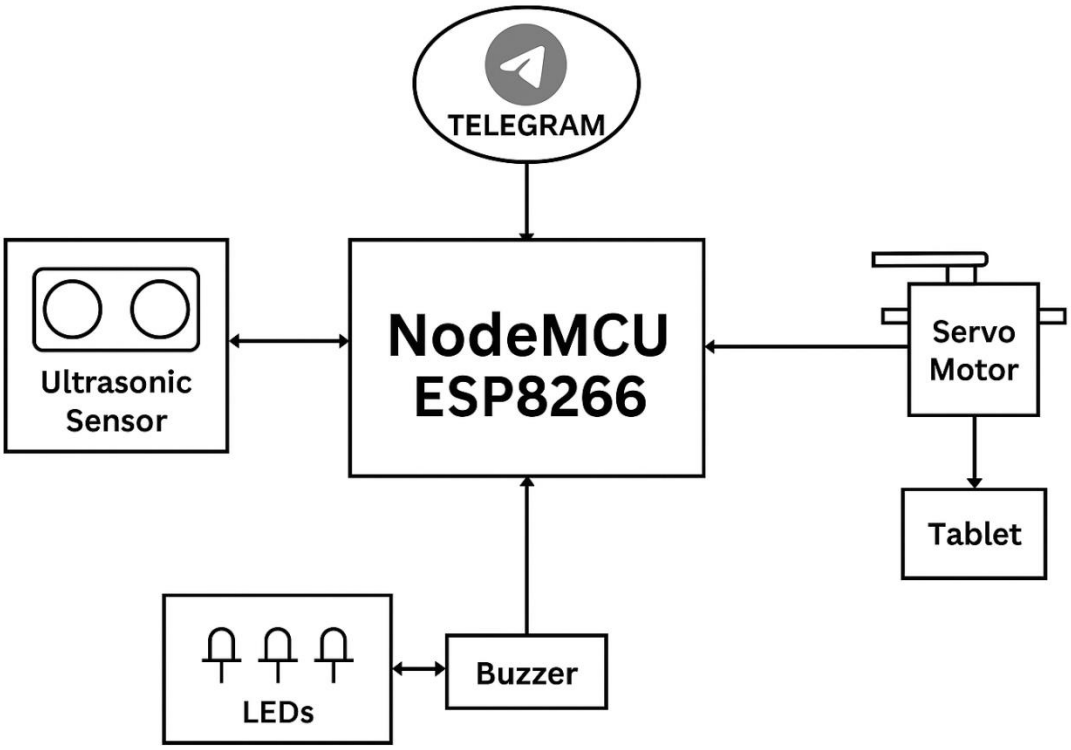


Figure 1 : Block Diagram

Working of Designed Model:

The designed model functions as an intelligent and automated medicine reminder and dispenser system using IoT technologies. Below is the step-by-step working of the model:

1. **System Initialization**
   When powered ON, the red LED glows to indicate the system is active. The NodeMCU connects to the predefined Wi-Fi network and initializes all components, including the ultrasonic sensor, servo motor, and Telegram Bot.
2. **Scheduled Alerts**
   The system is programmed to send medicine reminders at fixed intervals. At the set time, the buzzer sounds, the green LED turns ON, and a Telegram message is sent to the user: *"□ It's time to take your medicine!"*
3. **User Detection**
   The ultrasonic sensor continuously checks for the user's hand beneath the dispenser.

When the hand is detected within a defined distance (e.g., 5 cm), the buzzer and green LED turn OFF.

4. **Tablet Dispensing**
   Once the hand is detected, the servo motor rotates to dispense a tablet. For each dose, the servo moves to a specific angle to release the tablet and then returns to its original position.
5. **Acknowledgment Message**
   After the tablet is dispensed, a confirmation message is sent to the user via Telegram:
   "✅Thank you for taking your medicine."
6. **Missed Dose Handling**
   If the user does not respond within 2 minutes, the system considers the dose as missed. A message is then sent to the user:
   "⚠ You missed your medicine dose!"
   The alert LED and buzzer are turned OFF, and the system resets for the next dose.
7. **Repeat Cycle**
   The entire process is cyclic and continues to function as per the programmed schedule, allowing the user to receive medicine doses on time with proper notifications and automation.

Code :

```
#include <Servo.h>
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>

// WiFi credentials
const char* ssid = "OnePlus Nord 3 5G";
const char* password = "111111115";

// Telegram credentials
#define BOT_TOKEN "8150130430:AAEDsr6k_rhfwn2xY28-
B6UUrL1UZob2XCQ"
#define CHAT_ID "1673085106"

WiFiClientSecure client;
UniversalTelegramBot bot(BOT_TOKEN, client);

// Pin definitions
#define TRIG_PIN D4
```

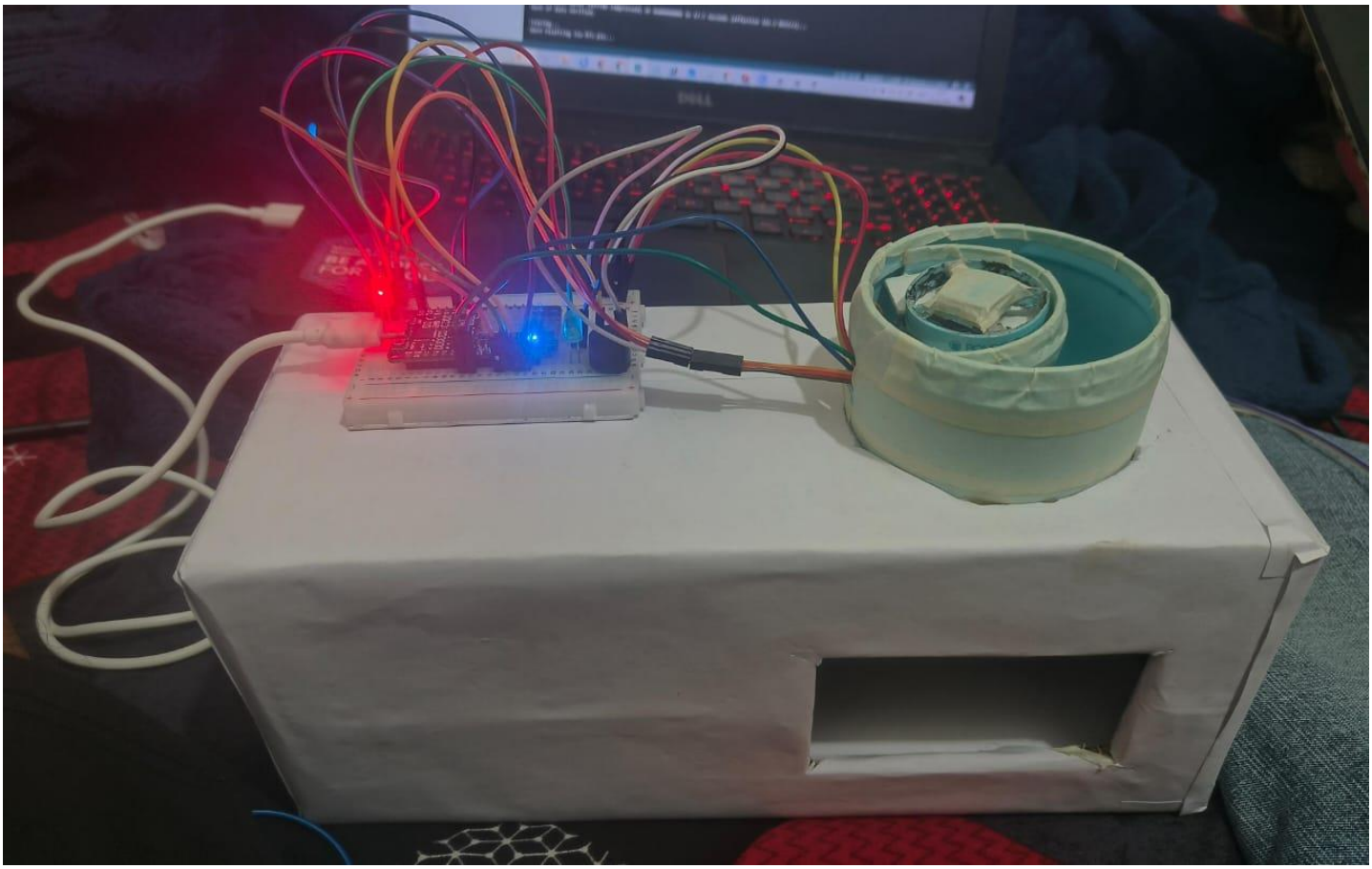
```
#define ECHO_PIN D8
#define SERVO_PIN D7
#define BUZZER_PIN D2
#define ALERT_LED_PIN D3
#define POWER_LED_PIN D1

#define DISTANCE_THRESHOLD 5    // Object detection distance in
cm
#define ALERT_INTERVAL 60000    // Time between alerts
#define MISSED_TIME 30000       // Time to wait before missed
alert

Servo myServo;

unsigned long lastAlertTime = 0;
unsigned long alertSentTime = 0;

bool missedMessageSent = false;

enum DoseStage {
  NONE,
  FIRST_ALERT_SENT,
  FIRST_DOSE_TAKEN,
  SECOND_ALERT_SENT
};

DoseStage doseStage = NONE;

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

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(ALERT_LED_PIN, OUTPUT);
  pinMode(POWER_LED_PIN, OUTPUT);

  digitalWrite(POWER_LED_PIN, HIGH);
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(ALERT_LED_PIN, LOW);
```

```
  myServo.attach(SERVO_PIN, 500, 2400);
  myServo.write(0);

  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" Connected!");

  client.setInsecure(); // Skip SSL cert validation
}

void loop() {
  unsigned long currentMillis = millis();

  // Handle alert timings
  if (currentMillis - lastAlertTime >= ALERT_INTERVAL) {
    lastAlertTime = currentMillis;
    alertSentTime = currentMillis;
    missedMessageSent = false;

    if (doseStage == NONE) {
      bot.sendMessage(CHAT_ID, "□ It's time to take your 1st
medicine!", "");
      digitalWrite(ALERT_LED_PIN, HIGH);
      digitalWrite(BUZZER_PIN, HIGH);
      doseStage = FIRST_ALERT_SENT;
    } else if (doseStage == FIRST_DOSE_TAKEN) {
      bot.sendMessage(CHAT_ID, "□ It's time to take your 2nd
medicine!", "");
      digitalWrite(ALERT_LED_PIN, HIGH);
      digitalWrite(BUZZER_PIN, HIGH);
      doseStage = SECOND_ALERT_SENT;
    }
  }

  // Distance sensing logic
```

```
  Serial.print("Dose Stage: ");
  Serial.println(doseStage);
  int distance = getDistance();
  Serial.print("Distance: ");
  Serial.println(distance);
  if (distance > 0 && distance <= DISTANCE_THRESHOLD) {
    digitalWrite(BUZZER_PIN, LOW);
    digitalWrite(ALERT_LED_PIN, LOW);

    if (doseStage == FIRST_ALERT_SENT) {
      myServo.write(90);
      bot.sendMessage(CHAT_ID, "✅ Thank you for taking your 1st
medicine.", "");
      doseStage = FIRST_DOSE_TAKEN;
    } else if (doseStage == SECOND_ALERT_SENT) {
      myServo.write(180);
      bot.sendMessage(CHAT_ID, "✅ Thank you for taking your 2nd
medicine.", "");
      delay(1000);
      myServo.write(0);
      doseStage = NONE; // Reset for next cycle
    }

    delay(1000);
  }

  // Missed dose logic
  if (!missedMessageSent && (currentMillis - alertSentTime >=
MISSED_TIME)) {
    if (doseStage == FIRST_ALERT_SENT) {
      bot.sendMessage(CHAT_ID, "⚠ You missed your 1st medicine
dose!", "");
      digitalWrite(BUZZER_PIN, LOW);
      digitalWrite(ALERT_LED_PIN, LOW);
      doseStage = NONE;
    } else if (doseStage == SECOND_ALERT_SENT) {
      bot.sendMessage(CHAT_ID, "⚠ You missed your 2nd medicine
dose!", "");
      digitalWrite(BUZZER_PIN, LOW);
      digitalWrite(ALERT_LED_PIN, LOW);
```

```
      doseStage = NONE;
    }
    missedMessageSent = true;
  }

  delay(500);
}

// Distance measurement
int getDistance() {
  long duration;
  int distance;

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.034 / 2;
  return distance;
}
```

Pictures of Prototype:



Output of Deigned Model/Prototype :

Figure 1 : Output on BLYNK Clou

Learning outcomes (What I have learnt):

1. I understood the working and integration of NodeMCU (ESP8266) with various electronic components.
2. I learned how to use an ultrasonic sensor for proximity detection in real-time applications.
3. I explored how servo motors can be used for mechanical automation like tablet dispensing.
4. I gained hands-on experience in connecting LEDs and buzzers for alert systems.
5. I learned how to build a Telegram bot and integrate it with an IoT device for sending notifications.
6. I understood how to use breadboards for circuit prototyping.
7. I improved my programming skills by writing and modifying Arduino code.
8. I experienced how to create a block diagram and explain system workflow effectively.
9. I learned the importance of user interaction and alert mechanisms in healthcare-based IoT projects.
10. I developed skills in testing, debugging, and troubleshooting hardware-software interfacing.