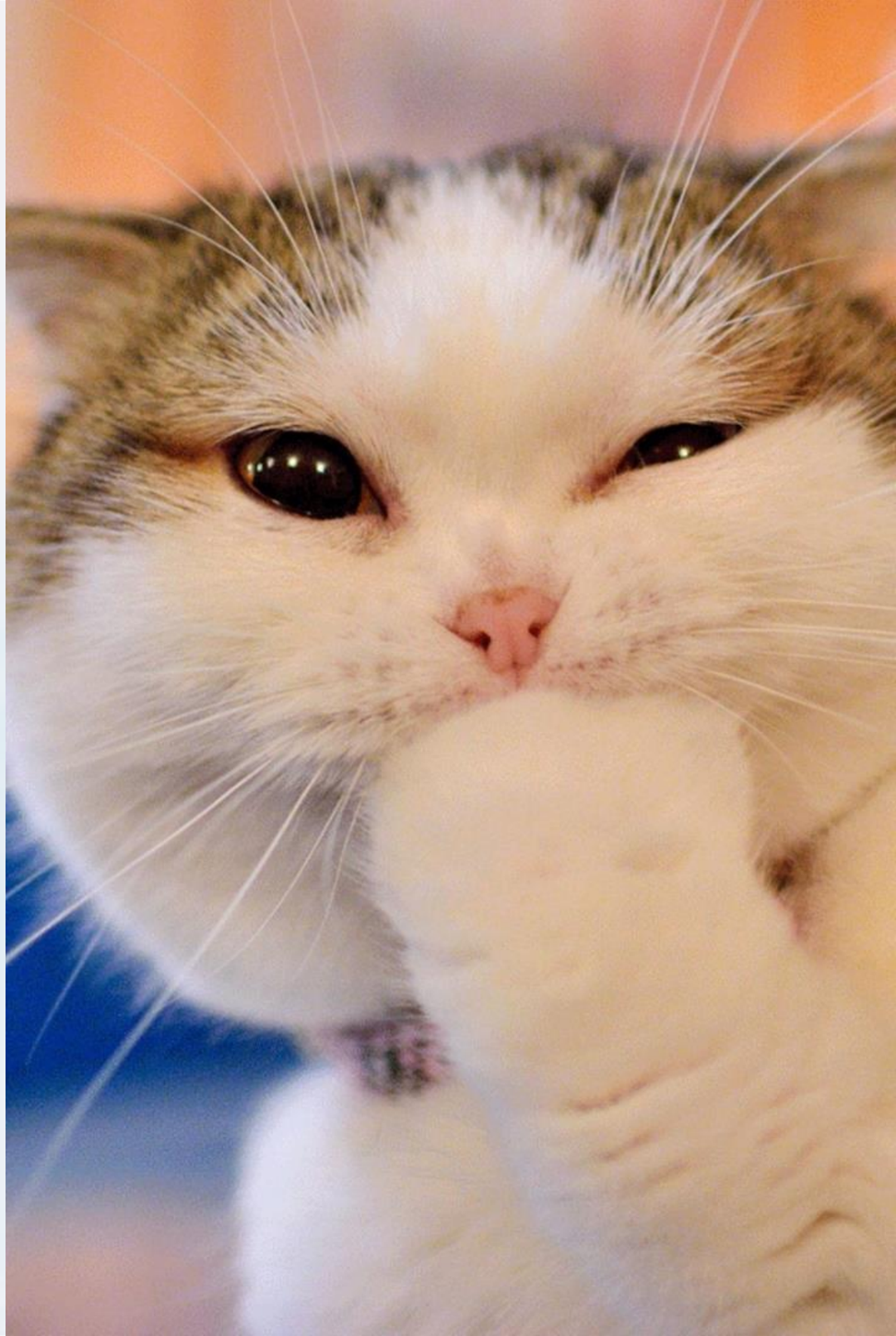
Google Assistant Controlled IoT Pet Feeder

This project uses ESP8266 to build a smart pet feeder that can be controlled by Google Assistant.



Components Required

ESP8266 Module

This is the core component of our project. ESP8266 is a powerful Wi-Fi module that enables enables remote control of devices using wireless wireless communication.

Servo Motor

This motor will enable us to control the amount amount of food dispensed by the feeder.

16x2 LCD Module

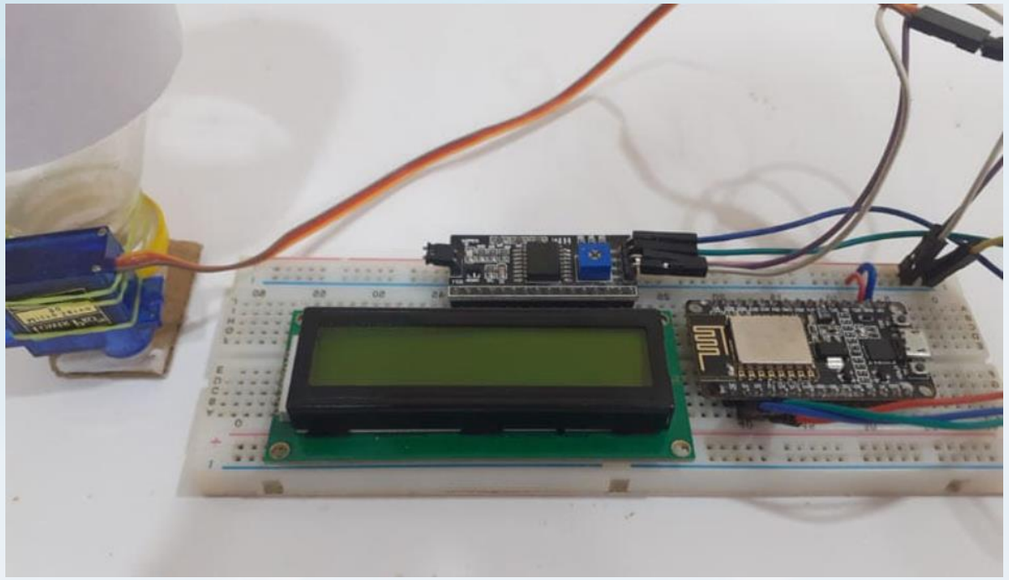
Using a 16x2 LCD module can enhance the user user experience by providing real-time information such as feeding schedules, status status updates, error messages etc.

LCD I2C Module

Using an LCD I2C module provides you with a with a visual interface to display important information, status updates, and settings related related to the pet feeder. This enhances the user user experience and allows for real-time interaction with the device

THEORY

The experiment involves creating an IoT-based pet feeder controlled by Google Assistant using an ESP8266 microcontroller. A servo motor is used to open and close the feeding trap door. The NTPClient library fetches accurate time from an NTP server, and a LiquidCrystal I2C display shows the current time and feeding schedule. The Adafruit MQTT protocol facilitates communication between the feeder and Google Assistant, allowing commands like "ON" and scheduled feeding times to be received. The microcontroller responds to commands, displaying relevant information on the LCD and dispensing food at scheduled times. The setup provides real-time feedback and ensures error handling for reliable operation.The Vcc & GND pin of Servo motor and LCD I2C module are connected with Vin & GND pin of NodeMCU. While SCL and SDA pins of I2C module is connected with D1 and D2 pin of NodeMCU .



ADAFRUIT MQTT BROKER

The Adafruit MQTT server is used as the broker. The Adafruit MQTT server acts as a message broker, facilitating communication between the pet feeder (ESP8266 microcontroller) and external devices, such as Google Assistant. It enables the exchange of messages based on specific topics.

The pet feeder subscribes to specific topics to receive commands, and the server relays those commands to the commands to the microcontroller. This communication mechanism enables remote control and interaction with interaction with the pet feeder through voice commands issued to Google Assistant.

Connecting the ESP8266 to Wi-Fi

1

Connect to Wi-Fi Network

First, we need to connect the ESP8266 to ESP8266 to our Wi-Fi network. We'll need need to know our network name and password.

2

Load the Program to ESP8266

We'll download and use Arduino IDE to run the code in the ESP8266.

3

Test the Connectivity

After the code is loaded, we'll test the the connectivity of the device with a few few simple commands.

Setting Up Google Assistant

Configure actions on on Google Console Console

We'll create an Actions on Google project and setup Google Assistant to interact with with ESP8266.

Implement the webhook code

We'll implement the Webhook code to act as a mediator between ESP8266 and Google Cloud Speech API.

Test the Google Assistant integration

We'll test the functionality by by syncing and interacting with with our Google Assistant project from a web interface.

Controlling the Pet Feeder with Google Google Assistant

Via Smartphone

We can use Google Assistant on our smartphone to smartphone to ask the pet feeder to dispense food. food.

Via Voice Controller

For highly customised or non-standard use-cases, we cases, we can build a voice controller, and connect it connect it to the feeder for easy control anytime. anytime.

Code Walkthrough

1

Set Up the Web Server

We'll use the ESP8266 module to build a build a local web server to control the the feeder. Our server will listen to the the request and trigger specific actions actions from there on.

2

Implement Webhooks

The Webhook on Google Cloud Platform Platform enables communication between between our Google Assistant and ESP8266 module.You can change specific specific settings and/or add commands commands through this intermediary. intermediary.

3

Configure Google Assistant Backend

We'll use Dialogflow to set up our conversational interface with our AI assistant, after which we will configure the configure the backend to integrate with with the ESP8266 module.

PROGRAMMING CODE

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <Servo.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 19800, 60000);

Servo servo;
LiquidCrystal_I2C lcd(0x27, 16, 2);

#define WIFI_SSID "Galaxy-M20"
#define WIFI_PASS "ac312124"
#define MQTT_SERV "io.adafruit.com"
#define MQTT_PORT 1883
#define MQTT_NAME "aschoudhary"
#define MQTT_PASS "1ac95cb8580b4271bbb6d9f75d0668f1"
int SERVO_PIN = D3;
int CLOSE_ANGLE = 0;
int OPEN_ANGLE = 60;
int hh, mm, ss;
int feed_hour = 0, feed_minute = 0;
```

```
WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, MQTT_SERV, MQTT_PORT, MQTT_NAME, MQTT_PASS);
Adafruit_MQTT_Subscribe onoff = Adafruit_MQTT_Subscribe(&mqtt, MQTT_NAME
"/f/onoff");
boolean feed = true;

void setup() {
  Serial.begin(9600);
  timeClient.begin();
  Wire.begin(D2, D1);
  lcd.begin();
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) delay(500);
  mqtt.subscribe(&onoff);
  servo.attach(SERVO_PIN);
  servo.write(CLOSE_ANGLE);
}

void loop() {
  MQTT_connect();
  timeClient.update();
  hh = timeClient.getHours();
  mm = timeClient.getMinutes();
  ss = timeClient.getSeconds();
  lcd.setCursor(0, 0);
  lcd.print("Time:");
  lcd.print(hh > 12 ? hh - 12 : hh);
  lcd.print(":");
  lcd.print(mm);
  lcd.print(":");
  lcd.print(ss);
  lcd.println(hh > 12 ? " PM " : " AM ");
  lcd.setCursor(0, 1);
```

```
lcd.print("Feed Time:");
lcd.print(feed_hour);
lcd.print(':');
lcd.print(feed_minute);

Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(5000))) {
  if (subscription == &onoff) {
    if (!strcmp((char*)onoff.lastread, "ON")) {
      open_door();
      delay(1000);
      close_door();
    }
    if (!strcmp((char*)onoff.lastread, "Morning")) {
      feed_hour = 10;
      feed_minute = 30;
    }
    if (!strcmp((char*)onoff.lastread, "Afternoon")) {
      feed_hour = 1;
      feed_minute = 30;
    }
    if (!strcmp((char*)onoff.lastread, "Evening")) {
      feed_hour = 6;
      feed_minute = 30;
    }
  }
}
if (hh == feed_hour && mm == feed_minute && feed) {
  open_door();
  delay(1000);
  close_door();
  feed = false;
}
```

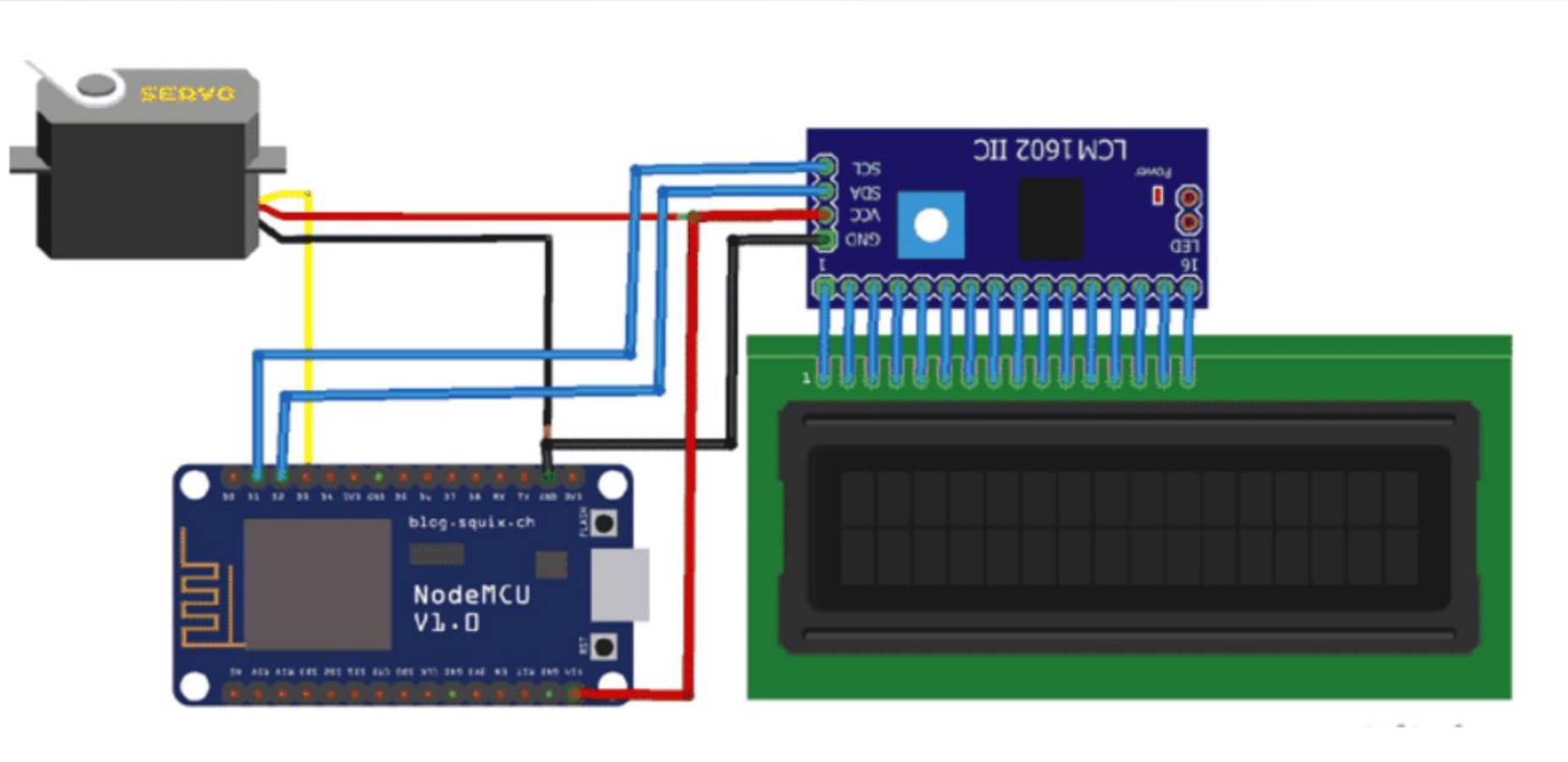
```
}

void MQTT_connect() {
  int8_t ret;
  if (mqtt.connected()) return;
  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) {
    mqtt.disconnect();
    delay(5000);
    if (--retries == 0) while (1);
  }
}

void open_door() {
  servo.write(OPEN_ANGLE);
}

void close_door() {
  servo.write(CLOSE_ANGLE);
}
```

CIRCUIT DIAGRAM



Conclusion and Additional Project Ideas

1

Conclusion

The Google Assistant controlled IoT pet feeder feeder project is a wonderful way to become become familiar with IoT devices and Google Google Assistant.

2

Additional Projects

You can try extending the project by building a building a more complex web interface, adding adding more devices or even integrating it with it with your pet's weight sensor.