**DESKTOP MOOD BOT**

**Final code**

#include <Arduino.h>

#include <WiFi.h>

#include <WiFiClientSecure.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

// = Wi‑Fi (fill) =

const char\* WIFI\_SSID = "wifi name";

const char\* WIFI\_PASSWORD = "PASSWORD";

// = Perplexity API (fill key) =

const char\* PPLX\_ENDPOINT = "https://api.perplexity.ai/chat/completions";

const char\* PPLX\_MODEL = "sonar"; // try "sonar-chat" if 400 persists

const char\* PPLX\_API\_KEY = "API KEY";

// = OLED (SSD1306 128x64 I2C) =

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_I2C\_ADDR 0x3C

#define I2C\_SDA\_PIN 4

#define I2C\_SCL\_PIN 5

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

// = IR pins (active-LOW) =

const int IR\_LEFT\_PIN = 34; // LEFT = Happy

const int IR\_RIGHT\_PIN = 35; // RIGHT = Tired

// Timings (UI + motion)

const unsigned long DEBOUNCE\_MS = 80;

const unsigned long COOLDOWN\_MS = 1200;

const unsigned long SHOW\_MS = 9000;

const unsigned long HAPPY\_SPIN\_MS = 900; // ~360° spin

const unsigned long FWD\_MS = 1000; // forward duration

const unsigned long STOP\_MS = 1000; // stop pause

unsigned long lastTrig=0;

// ===================== BIG EYES + IDLE ANIMATION =====================

unsigned long idleTimer=0;

unsigned long blinkTimer=0;

bool blinkActive=false;

uint8\_t blinkPhase=0;

struct EyeGeom {

int w = 40;

int h = 36;

int r = 8;

int gap = 10;

int leftX, leftY, rightX, rightY;

void computeCentered() {

int totalW = (w\*2) + gap;

leftX = (SCREEN\_WIDTH - totalW)/2;

rightX = leftX + w + gap;

leftY = (SCREEN\_HEIGHT - h)/2;

rightY = leftY;

}

} eyes;

void drawBigEye(int x, int y, int w, int h, int r, float blinkFrac, int offsetX, int offsetY){

int hh = h;

if (blinkFrac > 0) {

int closePix = (int)(h \* blinkFrac);

hh = max(1, h - closePix);

y += closePix/2;

}

x += offsetX; y += offsetY;

if (x < -w || y < -hh || x > SCREEN\_WIDTH || y > SCREEN\_HEIGHT) return;

display.fillRoundRect(x, y, w, hh, r, SSD1306\_WHITE);

}

void drawTiredLids(int x, int y, int w, int h, int r, int lidH){

lidH = constrain(lidH, 0, h/2);

if (lidH <= 0) return;

display.fillTriangle(x, y-1, x+w, y-1, x, y+lidH-1, SSD1306\_BLACK);

display.fillTriangle(x, y-1, x+w, y-1, x+w, y+lidH-1, SSD1306\_BLACK);

}

void drawHappyBottomLid(int x, int y, int w, int h, int r, int lift){

lift = constrain(lift, 0, h);

if (lift <= 0) return;

display.fillRoundRect(x-1, (y + h) - lift + 1, w+2, h, r, SSD1306\_BLACK);

}

void drawFaceMood(uint8\_t mood, float blinkFrac, int gazeX, int gazeY){

display.clearDisplay();

drawBigEye(eyes.leftX, eyes.leftY, eyes.w, eyes.h, eyes.r, blinkFrac, -gazeX, gazeY);

drawBigEye(eyes.rightX, eyes.rightY, eyes.w, eyes.h, eyes.r, blinkFrac, gazeX, gazeY);

if (mood == 1) { // tired

int lidH = eyes.h/3;

drawTiredLids(eyes.leftX, eyes.leftY, eyes.w, eyes.h, eyes.r, lidH);

drawTiredLids(eyes.rightX, eyes.rightY, eyes.w, eyes.h, eyes.r, lidH);

} else if (mood == 2) { // happy

int lift = eyes.h/3;

drawHappyBottomLid(eyes.leftX, eyes.leftY, eyes.w, eyes.h, eyes.r, lift);

drawHappyBottomLid(eyes.rightX, eyes.rightY, eyes.w, eyes.h, eyes.r, lift);

}

display.display();

}

void updateIdleAnimation(){

static int breath=0, bdir=1;

if (millis() - idleTimer >= 40) {

idleTimer = millis();

breath += bdir;

if (breath > 3) bdir = -1;

if (breath < -3) bdir = 1;

}

static int gx=0, gdir=1; static unsigned long gazeT=0;

if (millis() - gazeT >= 100) {

gazeT = millis();

gx += gdir;

if (gx > 4) gdir = -1;

if (gx < -4) gdir = 1;

}

if (!blinkActive && millis() - blinkTimer > 3000UL + (unsigned long)random(0, 4000)) {

blinkActive = true; blinkPhase = 0; blinkTimer = millis();

}

float blinkFrac = 0.0f;

if (blinkActive) {

if (blinkPhase <= 8) blinkFrac = blinkPhase / 8.0f;

else blinkFrac = max(0.0f, 1.0f - (blinkPhase-8)/8.0f);

if (millis() - blinkTimer >= 20) {

blinkTimer = millis();

blinkPhase++;

if (blinkPhase > 16) { blinkActive = false; blinkFrac = 0.0f; }

}

}

drawFaceMood(0, blinkFrac, gx, breath);

}

// = Small status text =

void showSmall(const String& a, const String& b=""){

display.clearDisplay();

display.setTextColor(SSD1306\_WHITE);

display.setTextSize(1);

display.setCursor(0,0);

display.println(a);

if (b.length()) display.println(b);

display.display();

}

// = Quote rendering =

bool renderWrapped(uint8\_t sz,int topY,const String& text){

display.clearDisplay();

display.setTextColor(SSD1306\_WHITE);

display.setTextSize(sz);

int lineH = 8\*sz + (sz==1?2:4);

int maxC = (sz==2?10:20);

int y=topY, start=0;

while(start < (int)text.length() && y <= SCREEN\_HEIGHT - lineH){

int len = min(maxC, (int)text.length() - start);

int br=-1; for(int i=len;i>0;--i){ if(text.charAt(start+i-1)==' '){ br=i-1; break; } }

if(br<0) br=len;

String line = text.substring(start,start+br);

int16\_t bx,by; uint16\_t bw,bh; display.getTextBounds(line,0,0,&bx,&by,&bw,&bh);

int x=(SCREEN\_WIDTH-bw)/2; if(x<0)x=0; display.setCursor(x,y); display.print(line);

y+=lineH; start+=br; while(start<(int)text.length() && text.charAt(start)==' ') start++;

}

display.display();

return start >= (int)text.length();

}

void showQuoteReadable(const String& t){

if(!t.length()){ showSmall("No text"); return; }

display.clearDisplay();

if(!renderWrapped(2,4,t)) renderWrapped(1,0,t);

}

// = IR helper =

bool triggeredLow(int pin){

if(digitalRead(pin)==LOW){ delay(DEBOUNCE\_MS); return digitalRead(pin)==LOW; }

return false;

}

// = Variety prompts =

const char\* happyPrompts[] = {

"Fresh upbeat micro-quote, simple words, 6-9 words.",

"Short cheerful line, plain words, 6-9 words.",

"Quick uplifting boost, simple words, 6-9 words."

};

const char\* tiredPrompts[] = {

"Supportive micro-quote for tired mood, simple words, 6-9 words.",

"Gentle encouragement for fatigue, plain words, 6-9 words.",

"Kind, calm nudge for rest, simple words, 6-9 words."

};

// = Robust extractor with guards =

String extractFirstText(JsonVariantConst root){

if (!root.containsKey("choices")) return String();

JsonVariantConst choices = root["choices"];

if (!choices.is<JsonArrayConst>()) return String();

JsonArrayConst arr = choices.as<JsonArrayConst>();

if (arr.size() == 0) return String();

JsonVariantConst c0 = arr[0];

if (c0["message"]["content"].is<const char\*>()) {

return String(c0["message"]["content"].as<const char\*>());

}

if (c0["text"].is<const char\*>()) {

return String(c0["text"].as<const char\*>());

}

if (c0["message"]["content"].is<JsonArrayConst>()) {

JsonArrayConst blocks = c0["message"]["content"].as<JsonArrayConst>();

for (JsonVariantConst b : blocks) {

if (b["text"].is<const char\*>()) return String(b["text"].as<const char\*>());

if (b["type"].is<const char\*>() &&

String(b["type"].as<const char\*>())=="text" &&

b["data"]["text"].is<const char\*>()) {

return String(b["data"]["text"].as<const char\*>());

}

}

}

return String();

}

// = Networking: online quotes only =

String fetchQuoteOnline(const String& topic, int& httpCodeOut){

httpCodeOut=-1;

if(WiFi.status()!=WL\_CONNECTED) return String();

WiFiClientSecure client;

client.setInsecure();

HTTPClient http;

http.setReuse(false);

http.setTimeout(15000);

if(!http.begin(client, PPLX\_ENDPOINT)) return String();

http.addHeader("Content-Type","application/json");

http.addHeader("Authorization", String("Bearer ")+PPLX\_API\_KEY);

String nonce=String((uint32\_t)random(0xFFFFFFFF),HEX);

const int HN = sizeof(happyPrompts)/sizeof(happyPrompts[0]);

const int TN = sizeof(tiredPrompts)/sizeof(tiredPrompts);

const char\* prompt=(topic=="happy")

? happyPrompts[random(0,HN)]

: tiredPrompts[random(0,TN)];

StaticJsonDocument<640> req;

req["model"]=PPLX\_MODEL;

req["temperature"]=0.9;

req["max\_tokens"]=24;

JsonArray msgs=req.createNestedArray("messages");

JsonObject m1=msgs.createNestedObject(); m1["role"]="system";

m1["content"]="Reply ONLY with one very short line (<=10 words). No author, no quotes.";

JsonObject m2=msgs.createNestedObject(); m2["role"]="user";

m2["content"]=String(prompt)+" nonce="+nonce;

String payload; serializeJson(req,payload);

delay(5);

int code=http.POST(payload);

httpCodeOut=code;

String out;

if(code==200){

String body=http.getString();

StaticJsonDocument<3072> doc;

DeserializationError err = deserializeJson(doc, body);

if (!err) {

out = extractFirstText(doc.as<JsonVariantConst>());

}

}

http.end();

out.trim();

if(out=="null"||out=="(null)") out="";

if(out.startsWith("\"") && out.endsWith("\"") && out.length()>=2)

out=out.substring(1,out.length()-1);

while(out.endsWith(".")||out.endsWith("!")||out.endsWith("?"))

out.remove(out.length()-1);

out.trim();

return out;

}

// ===================== MOTOR CONTROL (20kHz, 8-bit, dead-time) =====================

// Motor A (left): IN1=27, IN2=26, ENA=14

// Motor B (right): IN3=25, IN4=33, ENB=32

const int IN1 = 27, IN2 = 26, ENA = 14; // Left (A)

const int IN3 = 25, IN4 = 33, ENB = 32; // Right (B)

const int CH\_A = 0, CH\_B = 1;

const int FREQ = 20000; // 20kHz (quiet)

const int RES = 8; // 8-bit duty (0..255)

// Speed tunables (0..255). Adjust to taste.

uint8\_t DUTY\_FWD = 200; // forward speed

uint8\_t DUTY\_REV = 200; // reverse speed

uint8\_t DUTY\_SPIN = 210; // spin speed

void motorA\_stop(){

digitalWrite(IN1, LOW);

digitalWrite(IN2, LOW);

ledcWrite(CH\_A, 0);

}

void motorB\_stop(){

digitalWrite(IN3, LOW);

digitalWrite(IN4, LOW);

ledcWrite(CH\_B, 0);

}

void motorA\_forward(uint8\_t d){

ledcWrite(CH\_A, 0);

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

delayMicroseconds(200); // dead-time

ledcWrite(CH\_A, d);

}

void motorA\_reverse(uint8\_t d){

ledcWrite(CH\_A, 0);

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

delayMicroseconds(200);

ledcWrite(CH\_A, d);

}

void motorB\_forward(uint8\_t d){

ledcWrite(CH\_B, 0);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

delayMicroseconds(200);

ledcWrite(CH\_B, d);

}

void motorB\_reverse(uint8\_t d){

ledcWrite(CH\_B, 0);

digitalWrite(IN3, LOW);

digitalWrite(IN4, HIGH);

delayMicroseconds(200);

ledcWrite(CH\_B, d);

}

void motorsSetup(){

pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);

pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);

ledcSetup(CH\_A, FREQ, RES);

ledcSetup(CH\_B, FREQ, RES);

ledcAttachPin(ENA, CH\_A);

ledcAttachPin(ENB, CH\_B);

motorA\_stop();

motorB\_stop();

}

inline void motorsStop(){ motorA\_stop(); motorB\_stop(); }

inline void motorsForward(){ motorA\_forward(DUTY\_FWD); motorB\_forward(DUTY\_FWD); }

inline void motorsBackward(){ motorA\_reverse(DUTY\_REV); motorB\_reverse(DUTY\_REV); }

inline void motorsSpinRight(){ motorA\_forward(DUTY\_SPIN); motorB\_reverse(DUTY\_SPIN); }

inline void motorsSpinLeft(){ motorA\_reverse(DUTY\_SPIN); motorB\_forward(DUTY\_SPIN); }

// ===================== END MOTOR CONTROL =====================

// = Setup =

void setup(){

randomSeed(analogRead(34)^millis());

pinMode(IR\_LEFT\_PIN, INPUT\_PULLUP);

pinMode(IR\_RIGHT\_PIN, INPUT\_PULLUP);

Wire.begin(I2C\_SDA\_PIN, I2C\_SCL\_PIN);

Wire.setClock(400000);

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_I2C\_ADDR);

eyes.computeCentered();

updateIdleAnimation();

// Motors

motorsSetup();

WiFi.mode(WIFI\_STA);

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

unsigned long t0=millis();

while(WiFi.status()!=WL\_CONNECTED && millis()-t0<15000) {

delay(200);

}

}

// = Loop =

void loop(){

updateIdleAnimation();

if(millis()-lastTrig<COOLDOWN\_MS){ delay(20); return; }

// LEFT => Happy: show face -> spin -> quote

if(triggeredLow(IR\_LEFT\_PIN)){

lastTrig=millis();

unsigned long tFace = millis();

while (millis() - tFace < 600) {

drawFaceMood(2, 0.0f, 2, -1);

delay(16);

}

motorsSpinRight();

delay(HAPPY\_SPIN\_MS);

motorsStop();

delay(150);

int code=0;

String q=fetchQuoteOnline("happy", code);

if(q.length()){

display.clearDisplay();

showQuoteReadable(q);

delay(SHOW\_MS);

} else {

showSmall(String("HTTP ")+code, "No quote");

delay(1200);

}

updateIdleAnimation();

return;

}

// RIGHT => Tired: show face -> forward, stop, forward -> quote

if(triggeredLow(IR\_RIGHT\_PIN)){

lastTrig=millis();

unsigned long tFace = millis();

while (millis() - tFace < 600) {

drawFaceMood(1, 0.0f, -2, 1);

delay(16);

}

motorsForward(); delay(FWD\_MS);

motorsStop(); delay(STOP\_MS);

motorsForward(); delay(FWD\_MS);

motorsStop(); delay(150);

int code=0;

String q=fetchQuoteOnline("tired", code);

if(q.length()){

display.clearDisplay();

showQuoteReadable(q);

delay(SHOW\_MS);

} else {

showSmall(String("HTTP ")+code, "No quote");

delay(1200);

}

updateIdleAnimation();

return;

}

delay(15);

}