

بسمه تعالی



آزمایشگاه سخت افزار

گزارش نهایی پروژه

گروه ۵

عنوان: قهوه ساز هوشمند

نگارندگان:

ساعی سعادت - ۹۷۱۱۰۲۶۳

امین مراضی - ۹۷۱۰۶۲۷۳

محمدصادق سلیمی - ۹۷۱۰۱۸۲۹

استاد گرامی:

جناب آقای دکتر اجلالی - جناب آقای دکتر فصحتی

تابستان ۱۴۰۱

فهرست مطالب

2	فهرست مطالب
4	مقدمه
5	دیتاشیت محصول
6	معماری سیستم
7	طراحی و پیاده‌سازی سخت‌افزار
7	برد آردوینو NodeMCU
7	توضیح کلی روند قهوه‌سازی - Staging
9	متغیرها و ثوابت
10	ایجاد نقطه اتصال قهوه‌ساز
10	اتصال به وای‌فای خانه
11	اتصال به سرور MQTT
12	تابع callback
12	توابع getMetrics و sendMetrics
13	تابع loop
14	مخزن قهوه
15	موتور Servo
16	موتور آرمیچر
17	ترانزیستور
18	مخزن آب
18	شیر برقی
20	برق دستگاه
21	بسیه‌بندی

24	طراحی و پیاده‌سازی سرور
24	مقدمه
24	توضیحاتی در ارتباط به کد سرور
24	سیگنال‌ها
25	طراحی و پیاده‌سازی اپلیکیشن
26	تصاویری از محیط نرم‌افزار
31	قیمت
32	جمع‌بندی

مقدمه

هدف از این پژوهه، ارائه یک قهوهساز هوشمند می‌باشد که در آن، فرآیند انتقال آب و قهوه به مخازن مربوطه در دستگاه، روشن و خاموش کردن دستگاه، تنظیم میزان غلظت قهوه و تعیین تعداد فنجان تولید شده، توسط خود کاربر و از طریق اپلیکیشن متصل به دستگاه مدیریت شود.

مزیت رقابتی اصلی این محصول که ما را به فکر ساخت و طراحی این محصول انداخت، عدم وجود قهوهساز هوشمند با توانایی انتقال قهوه از یک مخزن خارجی به مخزن درون قهوهساز بود؛ در واقع یکی از بهترین قهوهسازهای هوشمند موجود در بازار، قهوهساز شرکت شیاومی می‌باشد که تصویر آن را در زیر مشاهده می‌کنید:



تصویر ۱: قهوهساز هوشمند شیاومی به همراه یک کپسول قهوه که باید درون دستگاه جاساز شود.

راهکار آن برای حل مشکل قهوه، استفاده از کپسول‌های حاوی قهوه به تعداد مشخصی است که دستگاه صرفا درب این کپسول‌ها را باز کرده و از آن‌ها استفاده می‌کند و همانطور که مشخص است، تهیه و تعویض این کپسول‌ها کار آسانی نیست. لذا ما بر آن شدیم که از یک مخزن خارجی قهوه استفاده کنیم و از طریق مکانیزمی (که دارای چالش‌های کمی هم نبود ولی با موفقیت بر تمام آن‌ها فائق آمدیم) که در ادامه به توضیح دقیق آن خواهیم پرداخت، قهوه را به مخزن مربوطه در قهوهساز انتقال دهیم. از طرفی این محصول توسط یک نرم افزار کنترل می‌شود که این امکان را برای کاربر فراهم می‌آورد تا از هر جایی در خانه و در هر زمانی که دلش خواست، دستور به تهیه قهوه دهد.

همچنین یکی از ایده‌های خوبی که درباره این محصول داشتیم و محدودیت زمانی و چالش‌های زیاد موجود در مسیر به ما امکان پیاده‌سازی کامل آن را نداد، قابلیت برنامه‌ریزی زمانی برای تهیه قهوه بود که به

واسطه‌ی آن کاربر می‌تواند ساعت و روز مدنظر خود برای تهیه قهوه را به اپلیکیشن متصل به دستگاه اعلام کند و همچنین روزهای تکراری برای اجرای همین برنامه‌ی زمانی را نیز تعیین کند.

دیتاشیت محصول

ولتاژ ورودی به دستگاه و مورد استفاده در شیر برقی	۲۲۰ ولت
ولتاژ مورد استفاده در سایر اجزای مدار	۵ ولت
قطر شلنگ مورد استفاده برای انتقال آب	۸ میلی‌متر
زمان لازم برای انتقال آب به ازای یک و دو فنجان	۱۸ ثانیه - ۲۰ ثانیه
زمان لازم برای انتقال قهوه به ازای غلظت ۱ - ۲ - ۳	۵ ثانیه - ۸ ثانیه - ۱۱ ثانیه
زاویه موتور Servo به ازای حالت بسته و باز	۱۴۰ درجه - ۶۰ درجه

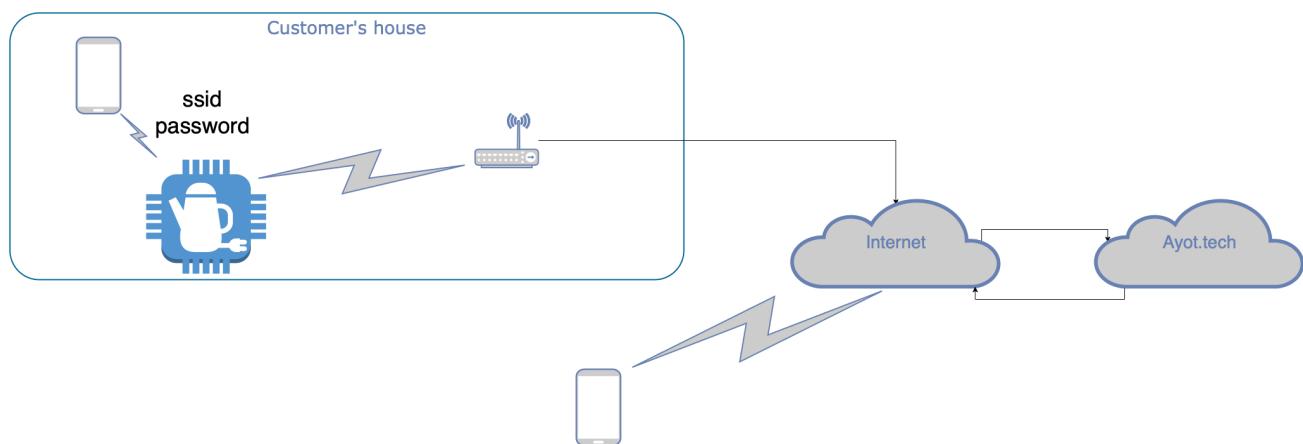
- در این محصول دو مخزن پلاستیکی بکار رفته که می‌توانند با مدل‌های دیگری جایگزین شده و ابعاد متفاوتی داشته باشند. به همین دلیل ابعاد مخازن آب و قهوه در دیتاشیت آورده نشده است.
- وزن محصول بسته به مخازن مورد استفاده و میزان قهوه و آب بکار رفته در آن متغیر است؛ به همین دلیل از پرداختن به وزن محصول در دیتاشیت اجتناب کردیم.

معماری سیستم

سیستم طراحی شده توسط ما از سه قسمت اصلی تشکیل شده است. قسمت سخت افزاری مشتمل از NodeMCU، مخازن آب و قهوه، شیر برقی، موتور Servo، شلنگ آب، موتور آرمیچر، قهوه ساز و یک سری قطعات الکتریکی مورد نیاز برای بستن مدار که در صورت نیاز دقیق‌تر آنها را بررسی خواهیم کرد.

سرور که برای دریافت فرمان‌های کاربر از سمت اپلیکیشن موبایل و انتقال آنها به قسمت سخت افزاری محصول استفاده شده است.

اپلیکیشن موبایل که برای ارتباط هرچه بهتر کاربر با دستگاه بصورت ریموت (در بستر اینترنت و بدون نیاز به حضور فیزیکی در محل دستگاه) و ست کردن یکسری از موارد دلخواه در رابطه با قهوه خود، مانند: غلظت قهوه و تعداد فنجان‌های آب مورد نیاز، بکار می‌رود. معما ری سطح بالای سیستم در شکل زیر قابل مشاهده است.



تصویر ۲: شمای سطح بالا از معما ری سیستم

همان طور که در تصویر نشان داده شده، در ابتدا تلفن هوشمند کاربر به وای‌فای قهوه ساز متصل می‌شود تا مشخصات مودم خانه را به آن ارسال کند. سپس دستگاه به مودم وصل شده و از طریق آن با سرورهای آیوت ارتباط می‌گیرد. از این به بعد تلفن کاربر از هر جایی، به شرطی که به اینترنت جهانی متصل باشد می‌تواند وضعیت دستگاه خود را مانیتور و یا آن را کنترل کند.

طراحی و پیاده‌سازی سخت‌افزار

اصلی‌ترین قسمت این پروژه، طراحی و پیاده‌سازی قسمت‌های سخت‌افزاری آن است. در زیر لیستی از قطعات سخت‌افزاری مورد استفاده آمده است و پس از آن ابتدا توضیحاتی در مورد روند قهوه‌سازی از روی کدهای سخت‌افزار ارائه می‌دهیم و سپس توضیحاتی در مورد هر یک از قطعات اصلی و نحوه کارکرد و راهاندازی آن ذکر شده است.

- برد آردوینو NodeMCU
- شیر برقی Solenoid
- رله Relay
- موتور Servo
- ترانزیستور TIP42
- موتور آرمیچر Armicger

برد آردوینو NodeMCU

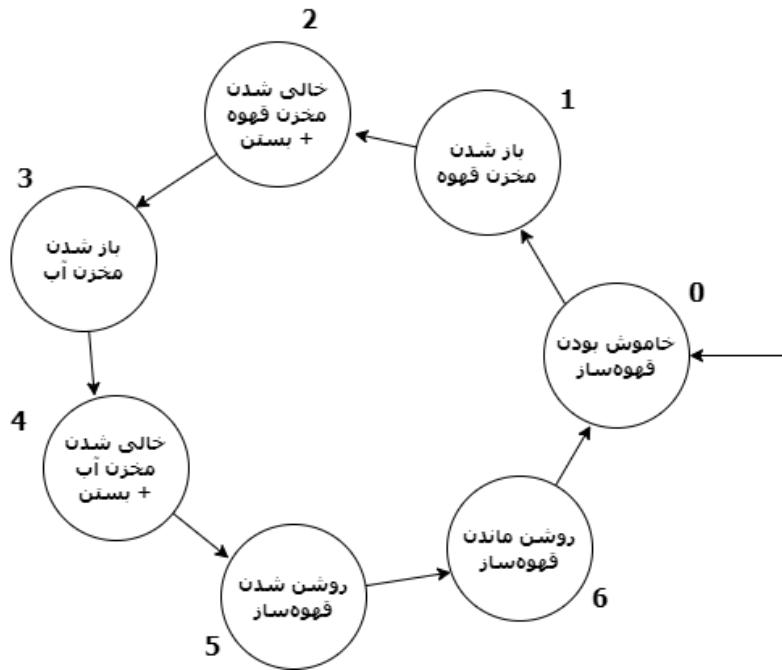
در این قسمت به تشریح قطعه‌ی NodeMCU، نحوه‌ی کارکرد آن و توضیحات کدی که روی آن زده شده است، می‌پردازیم.

در این پروژه ما از میکروکنترلر NodeMCU CH340 ESP8266 استفاده کردیم زیرا هم مازول وای‌فای را دارد و قابلیت‌های یک میکروکنترلر را دارد.

توضیح کلی روند قهوه‌سازی - Staging

در کد این پروژه ما یک FSM داریم که نشان می‌دهد ما در کدام بخش از فرایند قهوه‌سازی قرار داریم. در هر بار اجرای تابع loop در آردوینو، بررسی می‌کنیم که در کدام Stage قرار داریم و تابع مربوطه برای بررسی آن Stage را صدا می‌کنیم.

تصویر زیر یک Finite State Machine از نحوه عملکرد قهوه‌ساز را نشان می‌دهد.



تصویر ۳: استیج های FSM

برای پیاده سازی این M در تابع loop یک switch case قرار دادیم که تابع مربوط به آن بخش را اجرا کند. (هر stage یک شماره دارد که آن را در FSM نیز مشخص کردہ ایم).

```

switch (stage) {
    case 1:
    case 2: // draining coffee
        drainCoffee();
        break;

    case 3:
    case 4: // draining water
        drainWater();
        break;

    case 5: // start brewing
        setCoffeePower(true);
        startTime = millis();
        stage = 6;
        break;

    case 6: // brewing
        if (millis() - startTime >= coffeeTurnOnTime) {
            setCoffeePower(false);
            stage = 0;
        }
        break;
}

```

بلوک مربوط به

متغیرها و ثوابت

ابتدا چند تا از متغیرهای سراسری را توضیح می‌دهیم. ثوابت openDegree و closeDegree مربوط به زوایای مخزن قهوه هستند که جلوتر آن را توضیح می‌دهیم.

پس از آن ۴ پین کنترلی داریم:

- پین D0: کنترل روشن کردن برق آرمیچر (همزن قهوه‌ساز)
- پین D1: کنترل باز و بسته کردن شیر برقی مخزن آب
- پین D2: پین کنترلی زاویه موتور Servo
- پین D3: پین کنترلی روشن کردن قهوه‌ساز

```
int closeDegree = 140;
int openDegree = 60;
int armaturePin = D0;
int waterPin = D1;
int servoDataPin = D2;
int coffeePowerPin = D3;
```

پین‌های کنترلی و زوایای مذکور

متغیرهای coffeeLevel و waterLevel نشان می‌دهد که ما چه میزان قهوه و آب برای فرایند قهوه‌سازی می‌خواهیم که کاربر آن‌ها را از طریق اپلیکیشن تغییر می‌دهد.

متغیرهای coffeeInitTime و waterInitTime نشان می‌دهند که برای قهوه و آب سطح ۱ چه مقدار زمان باید برای باز بودن مخزن آن‌ها صرف شود.

برای قهوه و آب سطح بالاتر نیز به ازای هر افزایش سطح مقداری به زمان init اضافه می‌شود که آن‌ها را در متغیرهای waterLevelUpUnit و coffeeLevelUpUnit مشخص کرده‌ایم.

متغیر coffeeTurnOnTime نیز نشان می‌دهد که قهوه‌ساز برای چه مدت زمانی می‌باشد روشن باشد.

```
// constants:
int coffeeLevel = 1;
int waterLevel = 1;
int coffeeLevelUpUnit = 3000;
int waterLevelUpUnit = 10000;
int coffeeInitTime = 5000;
int waterInitTime = 18000;
int coffeeTurnOnTime = 120000;
```

ثوابت به کاررفته در کد

ایجاد نقطه اتصال قهقهه ساز

برد ابتدا یک Access point ایجاد می کند که ssid و رمز وای فای خانگی را به آن منتقل کنیم. در تابع setupAP این کار انجام می شود که کد آن را در تصویر زیر مشاهده می کنید.

```
void setupAP() {  
    Serial.printf("AP name %s\n", ACCESS_POINT_SSID);  
    Serial.print("Set config ");  
    Serial.println(WiFi.softAPConfig(local_ip, gateway, subnet) ? "Successful" : "Failed!");  
    Serial.print("Setup AP ");  
    Serial.println(WiFi.softAP(ACCESS_POINT_SSID, ACCESS_POINT_PASS) ? "Successful" : "Failed!");  
    IPAddress IP = WiFi.softAPIP();  
    Serial.print("AP IP address");  
    Serial.println(IP);  
    Serial.println("Setup wifi done");  
    Serial.println(getCharArrayFromString(WiFi.macAddress()));  
    Serial.println();  
    // access_token = device_id = "A4:CF:12:F0:00:B3";  
    access_token = device_id = getCharArrayFromString(WiFi.macAddress());  
}
```

بلوک setupAP

پس از آن این نقطه اتصال باید به یک سرور تبدیل شود که درخواست ها را دریافت می کند گوشی ما با دستور /connect مشخصات ssid و رمز وای فای خانه را به نقطه اتصال می فرستد تا بتواند به اینترنت جهانی و سرور وصل شود.

```
void setupServer() {  
    server.on("/ping", HTTP_GET, [] (AsyncWebServerRequest * request) {  
        request->send(200, "text/plain", "PONG");  
    });  
  
    server.on("/connect", HTTP_POST, [] (AsyncWebServerRequest * request) {  
        if (request->hasArg("ssid") && request->hasArg("pass") && request->arg("":  
            ssid = getCharArrayFromString(request->arg("ssid"));  
            pass = getCharArrayFromString(request->arg("pass"));  
            char* data = (char*) malloc(1024);  
            data[0] = '\0';  
    });  
}
```

بلوک setupServer

اتصال به وای فای خانه

پس از آن که ssid و رمز وای فای خانه به دست NodeMCU رسید با صدا کردن تابع connectToWifi اینترنت جهانی متصل می شود. کد این تابع را در تصویر زیر مشاهده می کنید.

```

bool connectToWifi() {
    Serial.printf("connecting to %s %s\n", ssid, pass);
    // if (!WiFi.config(local_ip, gateway, subnet)){Serial.println("STA Failed to configure");}
    WiFi.begin(ssid, pass);
    int i = 0;
    while (WiFi.status() != WL_CONNECTED) { // Wait for the Wi-Fi to connect
        delay(500);
        //   Serial.print(++i); Serial.print(' ');
        if (i == 20) {
            Serial.println("Failed to connect");
            WiFi.disconnect();
            has_ssid_pass = has_ssid_pass_really;
            return false;
        }
    }
}

```

بلوک اتصال به وای‌فای

اتصال به سرور MQTT

پس از آن که به اینترنت جهانی متصل شدیم، نیاز است تا به سرور MQTT متصل شویم. این فرایند در تابع `connectToMqtt` پیاده‌سازی شده است.

در این تابع یک `callback` مشخص می‌شود که در آن تابع دستورها و تغییراتی که از سرور/کاربر به دستگاه می‌رسد را اعمال کنیم.

کد این تابع را در تصویر زیر مشاهده می‌کنید.

```

bool connectToMqtt() {
    // Serial.printf("Connecting to mqtt %s:%d\n", MQTT_HOST, MQTT_PORT);
    client.setServer(MQTT_HOST, MQTT_PORT);
    client.setCallback(callback);
    if (!client.connect(device_id, device_id, access_token)) {
        Serial.println("Failed to connect to mqtt!");
        delay(1000);
        return false;
    }
    sub_topic[0] = '\0';
    strcat(sub_topic, SUB_TOPIC);
    strcat(sub_topic, device_id);
    pub_topic[0] = '\0';
    strcat(pub_topic, PUB_TOPIC);
    strcat(pub_topic, device_id);
    client.subscribe(sub_topic);
    // Serial.println("Connected to mqtt");
    return true;
}

```

بلوک اتصال به سرور MQTT

تابع callback

در این تابع دستورات و تغییراتی که از سمت سرور/کاربر به ما می‌رسد را اعمال می‌کنیم. مثلاً اگر در stage شماره صفر قرار داشته باشیم و سیگنال power مقدار ۱ را اخذ کند، فرایند قهوه‌سازی را شروع می‌کنیم و وارد stage ۱ می‌شویم.

اگر سطح آب و قهوه نیز تغییر کند این جا تغییر را اعمال می‌کنیم.

```
void callback(char *topic, byte *payload, unsigned int length) {
    Serial.printf("Call back %s %s \n", topic, payload);
    DynamicJsonDocument data(length + 10);

    if (data["signals"].containsKey("power")) {
        int power = data["signals"]["power"]["value"];
        if (stage == 0 && power == 1) {
            stage = 1;
        } else if (power == 0) {
            stage = 0;
        }
    } else if (data["signals"].containsKey("coffee")) {
        coffeeLevel = data["signals"]["coffee"]["value"];
    } else if (data["signals"].containsKey("water")) {
        waterLevel = data["signals"]["water"]["value"];
    }
    return;
}
```

تابع callback

توابع getMetrics و sendMetrics

در تابع getMetrics اطلاعات مربوط به وضعیت فعلی دستگاه (سخت‌افزار) را گرفته و آنها را آپدیت می‌کنیم. با تابع sendMetrics وضعیت فعلی را به سرور می‌فرستیم که سرور بتواند مطمئن شود که تغییراتش اعمال شده است یا حتی قابلیت monitoring در آینده سمت سرور ایجاد شود که وضعیت دستگاه (شامل دما، شرایط محیطی و ...) قابل مشاهده باشد.

```

void sendMetrics() {
    DynamicJsonDocument doc(2048);
    doc["id"] = device_id;
    doc["metrics"] = getMetrics();
    char Buf[2048];
    serializeJson(doc, Buf);
    client.publish(PUB_TOPIC, Buf);
    Serial.println("Sent metrics: coffee: ");
    delay(100);
    // Serial.print(coffeeLevel);
    // Serial.print(", water: ");
    // Serial.print(waterLevel);
    // Serial.print(" and power: ");
    // Serial.println(stage > 0);
}

```

بلوک ارسال وضعیت

تابع loop

تابع loop دو بخش اصلی دارد که بخش اول آن تلاش برای اتصال به وای‌فای و سرور MQTT را نشان می‌دهد؛ همچنین در این بخش مشخص کردہ‌ایم که هر ۳ ثانیه یک بار وضعیت دستگاه را به سرور اعلام کند.

تصویر زیر بخش اول کد این تابع را نشان می‌دهد. بخش دوم نیز همان staging است که بالاتر توضیح داده شد.

```

if (has_ssid_pass) {
    connected_to_wifi = connectToWifi();
    has_ssid_pass = false;
}
if (connected_to_wifi && !connected_to_mqtt) {
    Serial.println("Connecting to mqtt");
    connected_to_mqtt = connectToMqtt();
}
if (!client.connected() && connected_to_mqtt) {
    Serial.println("mqtt disconnected");
    connected_to_mqtt = false;
}

if (millis() - lastMetricSent > 3000) {

    if (connected_to_mqtt)
        sendMetrics();
    lastMetricSent = millis();
}

```

بخش اول تابع loop

مخزن قهوه

در این قسمت به ارائه قطعات سخت‌افزاری مربوط به قسمت مخزن قهوه می‌پردازیم که شامل سه قطعه اصلی می‌باشد؛ موتور Servo، موتور آرمیچر و ترازنیستور که در واقع لازمه‌ی قرارگیری آرمیچر در مدار دستگاه می‌باشد. در ادامه به توضیح هریک از این قطعات می‌پردازیم اما در اینجا قصد داریم یک چالش مهم که در رابطه با انتقال قهوه به مخزن قهوه‌ساز وجود داشت و مجبور به دست و پنجه نرم کردن با آن شویم را شرح دهیم.

همانطور که احتمالاً می‌دانید، پودر قهوه به مانند نمک و یا شکر بلوری نیست و بر روی هم نمی‌لغزند و فراتر از این نکته، پودر قهوه یک خاصیت چسبندگی نسبی دارد که باعث می‌شود پودر قهوه مسیر عبور خود از سوراخ مخزن را سد کرده و در فرآیند مدنظر ما اختلال ایجاد کند. برای فائق آمدن به این مشکل، نهایتاً تصمیم به استفاده از موتور آرمیچر گرفتیم تا به هنگام باز شدن درب خروجی مخزن، این موتور شروع به کار کند و با نی‌هایی قهوه موجود در دستگاه را هم بزند و بدین صورت از سد شدن مسیر خروج جلوگیری به عمل آید. در تابع drainCoffee فرایندهای مربوط به انتقال قهوه انجام می‌شوند که بلوک کد مربوطه را در زیر مشاهده می‌کنید:

```
void drainCoffee() {
    //////////////////////////////##///////////////////////////////
    int coffeeOpenTime = coffeeInitTime + (coffeeLevel - 1) * coffeeLevelUpUnit;

    if (stage == 1) {
        setCoffeeTank(true);           // Open the coffee tank
        delay(100);                  // Small wait
        setArmature(true);           // Turn on armature
        drainCoffeeStartTime = millis();
        stage = 2;
    } else if (stage == 2) {
        if (millis() - drainCoffeeStartTime >= coffeeOpenTime) {
            stage = 3;
            setArmature(false);       // Turn off armature
            delay(100);              // Small wait
            setCoffeeTank(false);     // Close the coffee tank
        } else {
            delay(100);
        }
    }
}
```

بلوک کد مربوط به انتقال قهوه به قهوه‌ساز

همانطور که در تابع بالا می‌بینید دو تابع دیگر نیز در میان آن فراخوانده شده‌اند که در تصویر زیر به یکی از آنها می‌پردازیم و دیگری را در قسمت مربوطه نمایش می‌دهیم:

```

void setCoffeeTank(bool isOpen) {
    // 1 for open and 0 for close
    if (isOpen) {
        myservo.write(openDegree);
    } else {
        myservo.write(closeDegree);
    }
}

```

تابع مربوط به باز و بسته کردن درب مخزن قهوه

Servo موتور

مотор Servo یک موتور الکترومکانیکی است که شکل آن را در تصویر زیر مشاهده می‌کنید:



تصویر ۴: موتور Servo

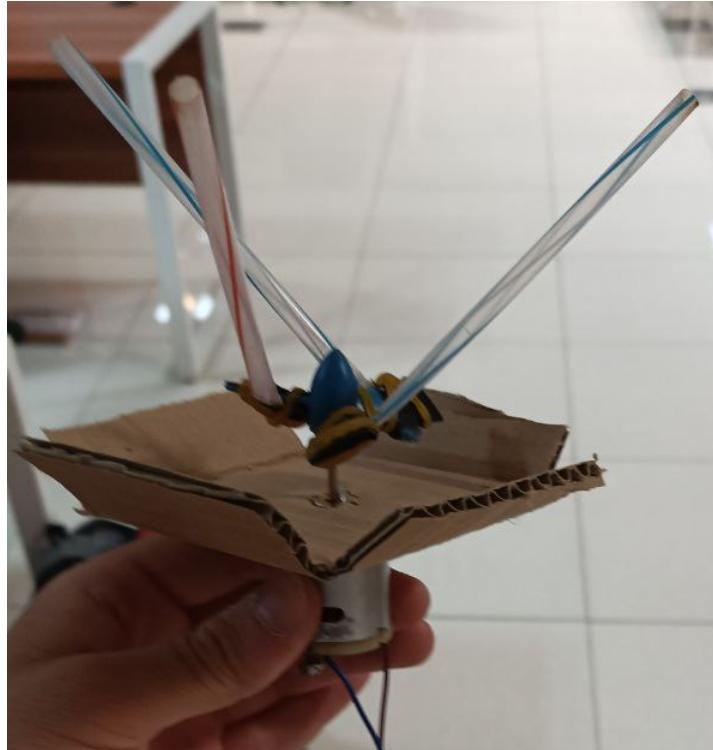
همانطور که در تصویر بالا می‌بینید، این دستگاه ۳ ورودی مختلف دارد، سیم قرمز مربوط به VCC، سیم قهوه‌ای/سیاه مربوط به Gnd و سیم زرد در واقع پین دیتای دستگاه می‌باشد که زاویه‌ی قرارگیری پرهی این موتور را تعیین می‌کند. بنا به یک سری آزمایش و تست، زاویه‌های مدنظر ما برای بسته بودن، ۱۴۰ درجه و برای باز بودن درب خروجی مخزن، ۶۰ درجه تعیین و در کد، ست شد.

این موتور Servo به برد NodeMCU متصل است و پین D2 را به خود اختصاص داده است و از طریق این پین، زوایای مختص به باز و بسته بودن را به صورت پالس، به دستگاه می‌دهیم.

موتور آرمیچر

همانطور که در چند قسمت قبل تر و در رابطه با چالش موجود برای انتقال قهوه بیان شد، ایده‌ی ما برای حل چالش ذکر شده، استفاده از یک موتور آرمیچر بود که به محض باز شدن درب خروجی مخزن قهوه، شروع به چرخیدن کند و تعداد نی را درون مخزن هم بزند تا از سد شدن راه خروج قهوه جلوگیری کند.

تصویر آرمیچر را در پایین مشاهده می‌کنید:



تصویر ۵: آرمیچر به همراه پره و نی‌های متصل به آن برای همزدن قهوه

آرمیچر ما در اصل با ۱۲ ولت کار می‌کند ولی برای اجتناب از استفاده کردن از مبدل تبدیل ۵ به ۱۲ ولت، از همان ولتاژ ۵ موجود در مدار برای راه انداختن آرمیچر نیز استفاده می‌کنیم.

این موتور آرمیچر به برد NodeMCU متصل است و پین D0 را به خود اختصاص داده است؛ دقت کنید که برای اینکه فقط در مواقعی که ما می‌خواهیم آرمیچر شروع به کار کند از یک ترانزیستور بعنوان کلید استفاده می‌کنیم که در بخش بعد دقیق‌تر آن را توضیح می‌دهیم.

یک نکته دیگر که در مورد آرمیچر باید بیان شود این است که این قطعه LOW Active می‌باشد و با LOW بودن ولتاژ D0، شروع به چرخیدن می‌کند و با HIGH بودن آن از چرخش می‌ایستد.

در کد، تابع زیر مربوط به راه اندازی آرمیچر می‌باشد که یکی دیگر از توابع فراخوانده شده در خلال تابع drainCoffee می‌باشد و ساختار آن را در تصویر زیر مشاهده می‌کنید:

```

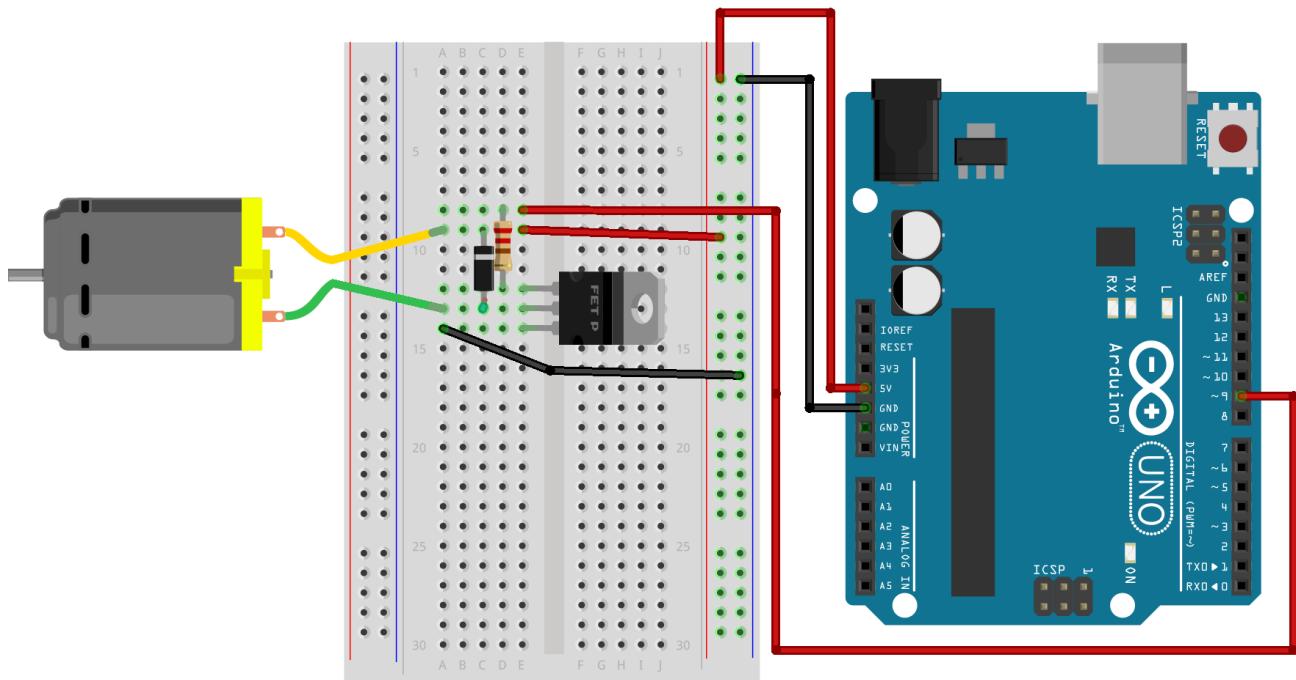
void setArmature(bool isOn) {
    // 1 for on and 0 for off
    if (isOn) {
        digitalWrite(armaturePin, LOW);
    } else {
        digitalWrite(armaturePin, HIGH);
    }
}

```

بلوک مربوط به روشن شدن موتور آرمیچر

ترانزیستور

از ترانزیستور به عنوان یک کلید برای راه اندازی موتور آرمیچر در موقعی که ما به آن نیاز داریم استفاده می‌کنیم؛ مدار مربوطه برای اتصال آرمیچر به ترانزیستور و قرار دادن آن‌ها در مدار را در ادامه می‌بینیم:



تصویر ۶: اتصال آرمیچر به مدار توسط ترانزیستور

همانطور که در تصویر بالا مشاهده می‌کنید پایه‌های ترانزیستور از بالا به پایین به ترتیب زیر در مدار قرار گرفته‌اند:

- پایه‌ی بالا توسط یک مقاومت به پین دیتای آرمیچر متصل شده است که در برد ما پین D0 است.
- پایه وسط به یک سر موتور آرمیچر متصل است.
- پایه پایین به Gnd متصل است.

همچنین سر دیگر آرمیچر نیز به Vcc متصل می‌شود.

بدین ترتیب هرگاه پین D0 دارای ولتاژ LOW باشد، آرمیچر شروع به چرخش خواهد کرد و با HIGH شدن آن، از چرخش می‌پسند.

مخزن آب

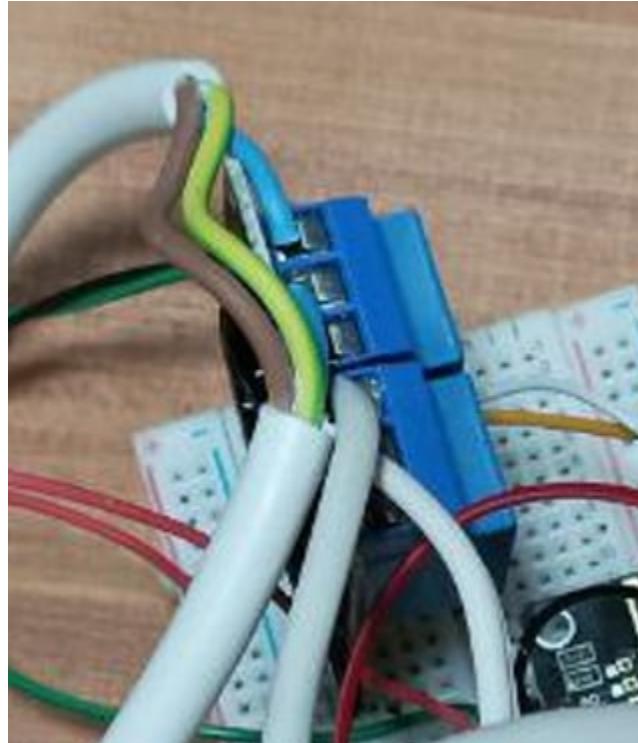
شیر برقی

در این قسمت به توضیح قطعات سخت‌افزاری بکار رفته در بخش مخزن آب می‌پردازیم که تنها شامل شیر برقی می‌باشد؛ به کمک شیر برقی Solenoid، قطع و وصل کردن جریان آب به درون دستگاه قهوه‌ساز را مدیریت می‌کنیم.



تصویر ۷: شیر برقی به همراه مخزن آب

این شیر برقی به یک رله متصل است که آن رله به برد NodeMCU متصل است و پین D1 را به خود اختصاص داده است. از نکاتی که نیاز است در مورد این قطعه بیان شود، LOW Active بودن آن است که بدان معناست که این جریان آب با LOW بودن ولتاژ پین D1 در NodeMCU برقرار می‌شود و با HIGH بودن ولتاژ آن پین، قطع می‌شود.



تصویر ۸: رله دوقلو برای برقرسانی به قهوهساز و شیر برقی

در استیج سوم از کد کارهای مربوط به این قطعه انجام می‌شود و در شروع آن هم مدت زمان مورد نیاز برای باز بودن شیر در متغیر WaterOpenTime ست می‌شود که برای حالت یک فنجان 18 ثانیه و برای حالت دو فنجان 28 ثانیه می‌باشد. دقت کنید که این زمان‌ها به کمک تست نسخه‌ی دموی محصول حاصل شده و برای این کار یک فنجان آب را سعی کردیم به اندازه کافی پر کنیم و زمان آن را اندازه گرفتیم و بدین صورت تخمین مدنظر خود را از این کار بدست آوردیم.

در کد نیزتابع drainWater مربوط به انتقال آب به مخزن مربوطه در قهوهساز می‌باشد که ساختار آن را در تصویر زیر می‌بینید:

```

void drainWater() {
    int waterOpenTime = waterInitTime + (waterLevel - 1) * waterLevelUpUnit;
    if (stage == 3) {
        setWaterTank(true);           // Open water tank
        drainWaterStartTime = millis();
        stage = 4;
    } else if (stage == 4) {
        if (millis() - drainWaterStartTime >= waterOpenTime) {
            setWaterTank(false);     // Close water tank
            stage = 5;
        }
    }
}

```

بلوک کد مربوط به انتقال آب به قهوهساز

برق دستگاه

سیم برق دستگاه قهوهساز، متشکل از ۳ سیم فاز، نول و ارث میباشد. مطابق تصویر، بر سر راه سیم فاز، یک رله قرار دادیم (قسمت دیگر رله دوتایی که در شیر برقی نیز به کار رفته بود) و سیم کنترل این رله، پین D3 میباشد که مطابق دیگر قطعات low active است و در صورتی که LOW شود، رله وصل شده و دستگاه برق میگیرد. روشن شدن دستگاه استیج آخر میباشد و از همه طولانیتر است. هنگامی که به استیج ۵ میرسیم دستگاه روشن میشود و پس از ۱۲۰ ثانیه (که این زمان باید وابسته به مقدار آب تعیین شود ولی برای سریعتر پیش رفتن تست ۱۲۰ ثانیه گذاشته شده) D3 مقدار HIGH میگیرد و دستگاه خاموش میشود. در اینجا سیگنال power دستگاه برابر شده و اپلیکیشن مطلع میشود که فرایند پایان یافته.

```

case 5: // start brewing
    setCoffeePower(true);
    startTime = millis();
    stage = 6;
    break;

case 6: // brewing
    if (millis() - startTime >= coffeeTurnOnTime) {
        setCoffeePower(false);
        stage = 0;
    }
    break;
}

void setCoffeePower(bool isOn) {
    // 1 for on and 0 for off
    if (isOn) {
        digitalWrite(coffeePowerPin, LOW);
    } else {
        digitalWrite(coffeePowerPin, HIGH);
    }
}

```

کد مربوط به برق قهوهساز

بسته‌بندی

بسته‌بندی این محصول یکی از چالش‌های این پروژه بود چرا که باید مخازن قهوه و آب به اندازه‌ی کافی بالاتر از خود قهوه‌ساز قرار می‌گرفتند. برای همین از ۲ استند فلزی گلدان با ارتفاع ۳۰ سانتی‌متر استفاده کردیم. در این حالت ارتفاع مخازن مناسب بودند اما باعث شد اندازه‌ی محصول نهایی چندین برابر بزرگ‌تر از خود قهوه‌ساز بشود. برای متصل شدن مخازن به استندها، از کش استفاده کردیم. همانطور که در تصویر مشاهده می‌کنید، زیر مخزن قهوه یک سوراخ به قطر ۱ سانتی‌متر وجود دارد که توسط موتور سرورو باز و بسته می‌شود. وسیله‌ای که توسط موتور سرورو جابجا می‌شود یک ورق آلومینیومی است که برای چفت و بست بهتر سوراخ مخزن قهوه، یک تکه ابر بر روی آن ورق چسباندیم. هنگامی که باز می‌شود (به کمک آرمیچر) قهوه داخل یک قیف پلاستیکی ریخته شده که روی سوراخ جلویی ایجاد شده روی قهوه‌ساز نصب شده ریخته می‌شود و در نتیجه داخل مخزن قهوه‌ی داخلی قهوه ساز ریخته می‌شود.

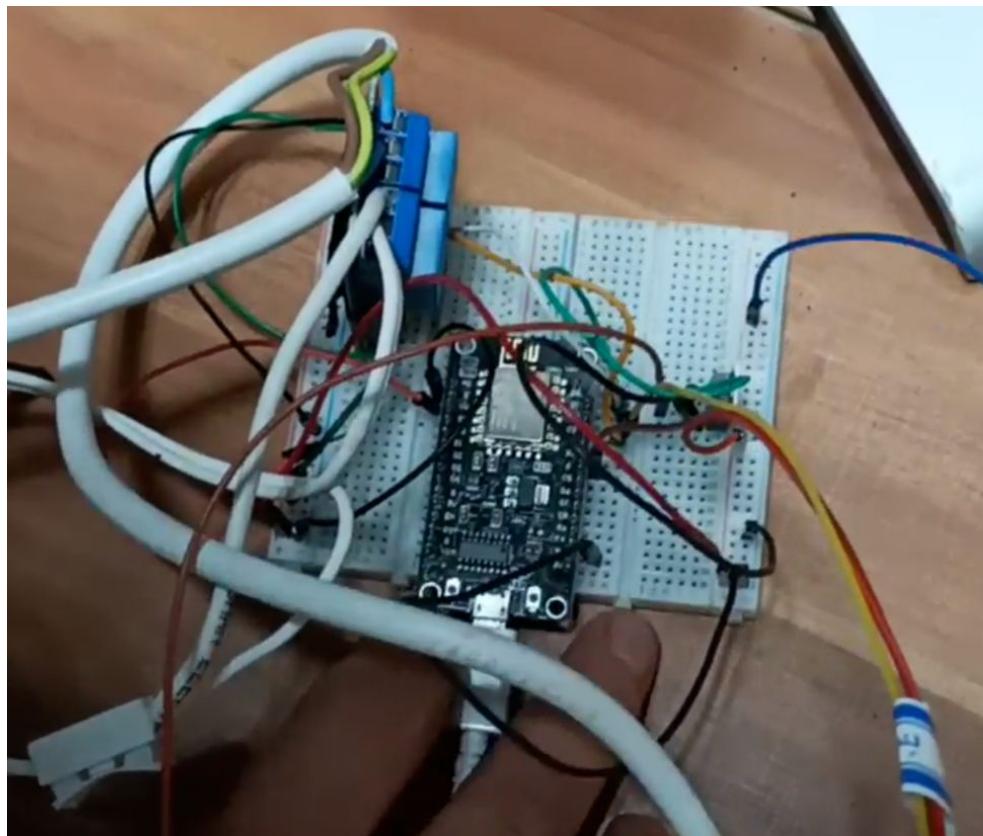
به صورت مشابه مخزن آب نیز روی یک استند مشابه با کش نصب شده و زیر آن مخزن، شلنگ ۸ میلی‌متری نصب شده که وارد شیر سلونوئید می‌شود و از طرف دیگر شیر، شلنگ ۸ میلی‌متری دیگری وارد سوراخ دیگر قهوه‌ساز شده که به مخزن آب داخلی آن راه دارد.



تصویر ۹: مخزن آب و قهوه بر روی استند فلزی

شیر برقی و قهقهه‌ساز ۲ دوشاخه‌ی برق جداگانه دارند و هر دو از رله‌های دوقلوی نصب شده روی برد بورد عبور می‌کنند.

در ادامه یک تصویر از فرم نهایی مدار و یک تصویر از بسته‌بندی نهایی محصول بصورت فیزیکی را قرار می‌دهیم؛ لازم به ذکر است که این بسته‌بندی توسط خود استاد برای این پروژه به تایید رسید هرچند قابلیت بهتر شدن نیز دارد.



تصویر ۱۰: فرم نهایی مدار



تصویر ۱۱: بسته‌بندی نهایی محصول

طراحی و پیاده‌سازی سرور

مقدمه

سرور آیوت مدتی قبل از آغاز این پروژه در دست ساخت بوده و برای این پروژه صرفا تغییرات مربوط به قهوه‌ساز و وضعیت برخط بودن دستگاه به قابلیت‌های آن اضافه شده است. معماری سیستم آن به صورت مایکروسرویس بوده و بر روی سرورهای ابر آروان قرار دارد.

توضیحاتی در ارتباط به کد سرور

از مهمترین سرویس‌های آیوت می‌توان به mqtt broker (با نرم‌افزار اوپن سورس Emxq) برای برقراری ارتباط با سخت‌افزار، و api-gateway برای برقراری ارتباط با تلفن همراه کاربر اشاره کرد. تمامی api‌های ارتباط بین دستگاه با سرور با پروتکل MQTT و ارتباط تلفن همراه با سرور با پروتکل https برقرار می‌شود. استک سرور: python (fast-api), postgresql, Emxq, Docker, Docker-compose, redis, kafka

سیگنال‌ها

دستگاه‌ها، اپلیکیشن و سرور از طریق ارسال سیگنال با یکدیگر ارتباط می‌گیرند و به هم دستور و اعلام وضعیت ارسال می‌کنند. ساختار سیگنال‌ها به این شکل می‌باشد:

- نام سیگنال: <- > مانند power
 - نوع سیگنال: int <- > سیگنال‌ها ۳ نوع دارند، float و string. نوع این سیگنال‌ها با مقادیر ۱/۲/۳ مشخص می‌شود.
 - مقدار سیگنال: any <- > وابسته به نوع سیگنال، مقدار مربوطه را در خود نگه می‌دارد.
- در این پروژه، قهوه‌ساز ۳ سیگنال دارد:

Signal name	Signal type	Signal value
Power	Int	0 = Off, 1 = On
Coffee	Int	1 = low, 2 = medium, 3 = high
Water	Int	1 = 1 cups, 2 = 2 cups

طراحی و پیاده‌سازی اپلیکیشن

برای کنترل کردن دستگاه، اپلیکیشن iOS آیوت نوشته شده است.
از قابلیت‌های نرم‌افزار می‌توان به

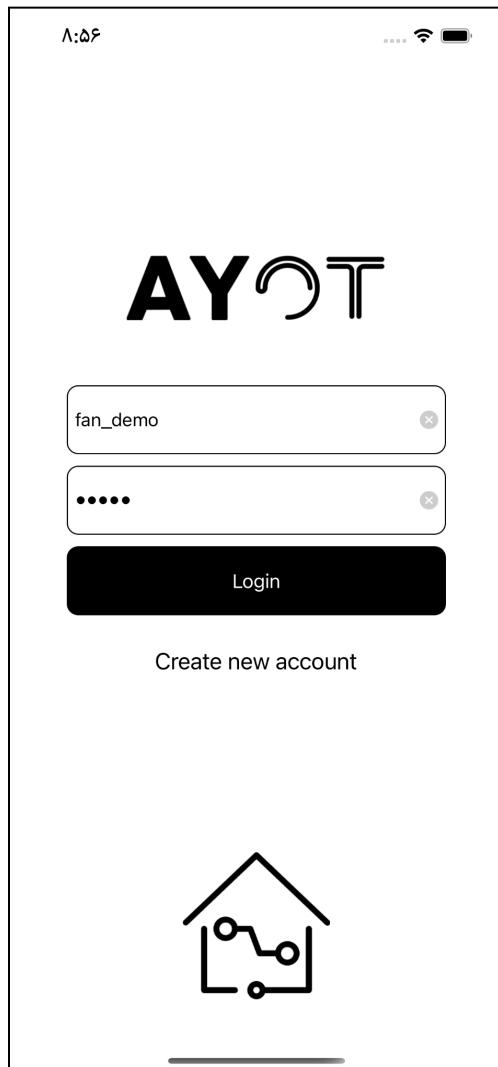
- لاگین کردن و ایجاد حساب کاربری
- مشاهده‌ی دستگاه‌های رجیستر شده برای اکانت
- مشاهده‌ی وضعیت آنلاین/آفلاین بودن دستگاه‌ها
- قابلیت کنترل و مانیتور کردن ۲ نوع دستگاه (پنکه و قهوه‌ساز)
- قابلیت تعیین میزان قهوه و آب برای قهوه‌ساز
- قابلیت زمان‌دهی برای ساخت قهوه
- قابلیت شروع و متوقف کردن فرایند ساخت قهوه

اشاره کرد.

استک برنامه: Swift - iOS 13+, UIKit, SwiftHero (for animations), Alamofire
در حال حاضر با توجه به تحریم‌های اپ‌استور در برابر ایران، اپلیکیشن آیوت در اپ‌استور قابل دسترسی نمی‌باشد.

تصاویری از محیط نرم افزار

صفحه‌ی ورود کاربر:



تصویر ۱۲: صفحه‌ی ورود کاربر در اپلیکیشن

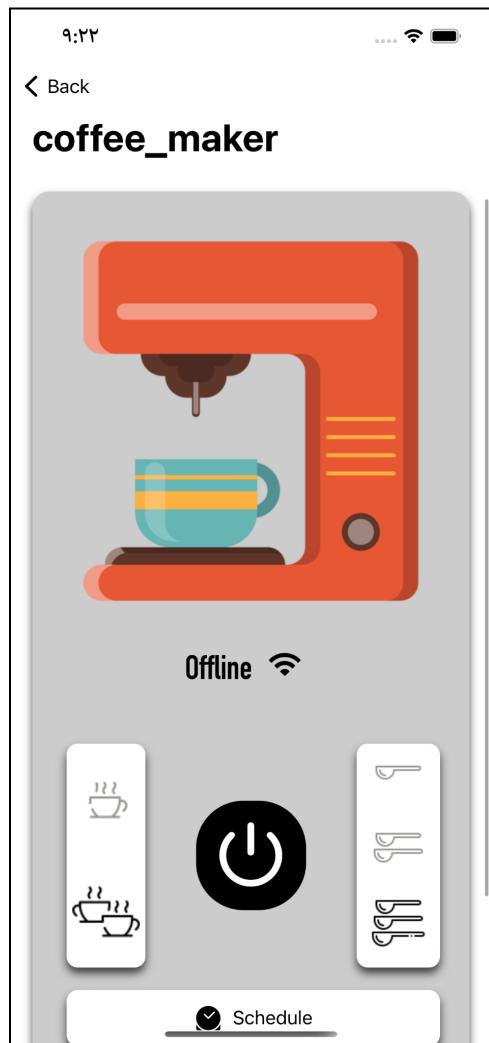
لیست دستگاه‌ها:



تصویر ۱۳: لیست دستگاه‌های کاربر در اپلیکیشن

در صفحه‌ی اول پس از ورود، اپلیکیشن لیست دستگاه‌های رجیستر شده‌ی کاربر را از سرور می‌گیرد و نشان می‌دهد. کنار نام هر دستگاه indicator ای وجود دارد که نشان می‌دهد دستگاه بربط می‌باشد یا خیر. دستگاه مورد نظر ما در این پروژه coffee_maker می‌باشد. هر دستگاه با استفاده از mac address چیپ وای‌فای (در این پروژه چیپ esp8266 در nodemcu) شناخته می‌شود.

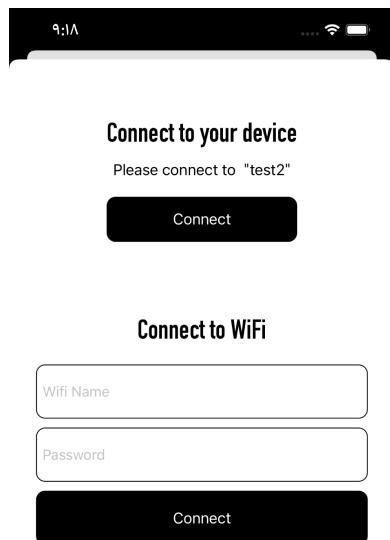
صفحه‌ی کنترل دستگاه:



تصویر ۱۴: صفحه کنترل دستگاه اپلیکیشن

در این صفحه، indicator ای وجود دارد که نشان می‌دهد دستگاه به اینترنت متصل هست یا خیر. در صورتی که نباشد، با فشردن دکمه با علامت وای‌فای کنار indicator، صفحه‌ی اتصال به وای‌فای نمایش داده می‌شود. زیر آن دکمه‌ی روشن و خاموش شدن دستگاه، در کنار دو نوار تعیین میزان آب و قهوه وجود دارد که هر کدام، سیگنال‌های مربوطه را به سرور (و از سرور به دستگاه) می‌فرستد.

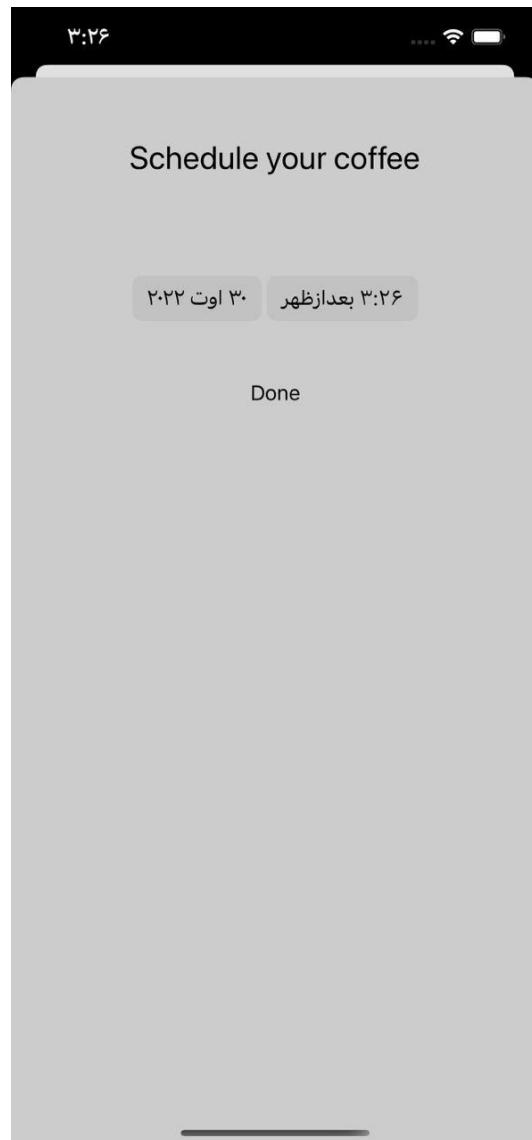
با زدن روی علامت wifi، صفحه‌ی متصل کردن دستگاه به اینترنت نشان داده می‌شود.



تصویر ۱۵: صفحه‌ی اتصال به وای‌فای

در ابتدا با فشردن دکمه‌ی connect اول، کاربر به صفحه‌ی تنظیمات موبایل راهنمایی می‌شود و در آنجا باید به access point قهقهه‌ساز متصل شود. نام این access point نیز در سرور وجود دارد و به صورت dynamic به کاربر در این صفحه نشان داده می‌شود (در این تصویر: test2) سپس کاربر به اپلیکیشن بازمی‌گردد و مشخصات وای‌فای (ssid و password) را نوشته و connect را نوشته و دوم را فشار می‌دهد. اپلیکیشن به connect/192.168.1.1 (که این مسیر نیز از سرور پرسیده می‌شود) یک درخواست ارسال می‌کند که nodeMCU ما آن‌ها را گرفته و به اینترنت متصل می‌شود. پس از آن قهقهه‌ساز access point خود را invisible می‌کند و موبایل از آن قطع می‌شود و مجدداً به اینترنت متصل می‌گردد. در این حالت باید indicator در صفحه‌ی اصلی، وضعیت online را برای دستگاه نشان دهد.

و در نهایت با زدن بر روی دکمه‌ی Schedule در تصویر ۱۷، به صفحه‌ی Scheduling دستگاه منتقل می‌شویم که در تصویر ۱۹ آن را مشاهده می‌کنید؛ در این قسمت کاربر می‌تواند زمان شروع فرایند قهوه‌سازی را به روز و ساعت مشخص کند و مثلا برنامه بدهد که ساعت ۷ صبح فرایند آغاز شود تا پس از بیدار شدن، قهوه‌اش آماده باشد.



تصویر ۱۶: صفحه Scheduling اپلیکیشن

قیمت

یکی از مسائل مهم در طراحی محصول قیمت آن است. البته با توجه به این که این محصول به صورت نمونه اولیه طراحی شده است، طبیعتاً قیمت تمام شده آن از محصولی که بخواهد تولید عمده بشود بالاتر خواهد بود. در جدول زیر قیمت تقریبی قطعات بکار رفته در پروژه و مجموع آن برای محصول نهایی را مشاهده می‌کنید.

ردیف	قطعه	قیمت(هزار تومان)
۱	برد آردوینو NodeMCU	۱۰۳
۲	موتور Servo	۵۶
۳	موتور آرمیچر	۵۰
۴	رله ۲تاپی	۵۰
۵	ترانزیستور	۱۰
۶	شیر برقی Solenoid	۲۴۰
۷	شنلگ	۲۰
۸	مخزن قهوه	۲۵
۹	مخزن آب	۳۲
۱۰	قهوهساز	۴۵۵.۷
۱۱	قیف	۱۰
۱۲	پره + نی + کش + ابر	۲۱
۱۳	Breadboard + سیم	۸۰
۱۴	دو استند گلدان	۸۰
-	مجموع	۱۲۳۲.۷

جمع‌بندی

در این پروژه به پیاده‌سازی قهوه‌ساز هوشمند پرداختیم. در این محصول، کاربر با استفاده از اپلیکیشن توسعه داده شده به قهوه‌ساز متصل شده و در صفحه اصلی این اپلیکیشن، غلظت قهوه و میزان آب مدنظر خود را انتخاب کرده و دکمه استارت را می‌فشارد. بدین ترتیب فرایند قهوه‌سازی به صورت فیزیکی استارت می‌خورد. ابتدا قهوه به میزان لازم از مخزن مربوط به داخل قهوه‌ساز منتقل می‌شود سپس آب به میزان دلخواه کاربر به قهوه‌ساز منتقل می‌شود و نهایتاً قهوه‌ساز روشن می‌شود و شروع به کار می‌کند.

تصویر کلی محصول ما علاوه بر بخش سخت‌افزاری، دارای یک سرور و یک نرم‌افزار نیز می‌باشد که ارتباط با محصول سخت‌افزاری را برای کاربر بسیار آسان می‌کند.

آنچه که این محصول را از قهوه‌سازهای هوشمند دیگر متمایز می‌کند، انتقال پودر قهوه از مخزن خارجی به داخل قهوه‌ساز می‌باشد که این کار را به کمک همزدن قهوه به هنگام باز شدن درب خروجی مخزن قهوه توسط یک موتور آرمیچر انجام می‌دهیم.

در نهایت شایان توجه است که ویدیوی توضیحات و همچنین دموی محصول را می‌توانید در [آپرات](#) و [یوتیوب](#) مشاهده بفرمایید.