پروژه پایانی (تلسکوپ هوشمند)

سیستمهای نهفته و بیدرنگ

استاد درس

دکتر حسینی منزه

دانشجویان

امیرحسین احمدی - ۹۷۵۲۲۲۹۲

محمد صدرا خاموشی – ۹۷۵۲۱۲۶۱

امید میرزاجانی - ۹۷۵۲۲۰۶۷

بهار ۱۴۰۱

فهرست مطالب

شرح پروژه ۳
ماژولهای استفاده شده ۴
اپلیکیشن موبایل ۵
سرور (MQTT) ۶
تلسکوپ ۷

شرح پروژه

در این پروژه تلاش بر این بوده است تا با استفاده از اینترنت اشیا، بتوان سیستم نهفته ای ساخت تا به کمک آن بتوانیم ستاره های موجود در آسمان را رصد کنیم. این سیستم به این صورت عمل میکند که با دریافت مختصات مورد نیاز برای نمایش یک ستاره موتور ها را در جهت هم راستا با آن ستاره میچرخاند.

ماژولهای استفاده شده

برای هسته اصلی این سیستم از یک میکروکنترلر ESP32 مدل WROOM استفاده شده است که در داخل خودش ماژول WIFI مورد نیاز برای ارتباط با سرور را نیز دارد. همچنین برای چرخیدن تلسکوپ در زوایای مختلف از دو موتور Servo مدل MG90S استفاده کردیم تا بتوانیم تمام زوایای فضایی را رصد کنیم. برای سر هم بندی این قطعات از برد بورد استفاده کردیم.

در مجموع سیستم از سه بخش کلی تشکیل شده:

۱. اپلیکیشن موبایل: که وظیفه نمایش ستاره ها و محاسبه زوایای چرخش مناسب هر کدام را دارد. همچنین وظیفه دارد در صورت درخواست کاربر، اطلاعات چرخش را به سرور جهت استفاده تلسکوپ ارسال کند.
۲. تلسکوپ: که تشکیل شده از میکروکنترلر و موتور هاست و وظیفه خواندن زوایا از سرور و نمایش آن به وسیله موتور ها را دارد.
۳. سرور: که وظیفه ایجاد ارتباط بین دو بخش اول را دارد تا بتوانند اطلاعات را با یک دیگر به اشتراک بگذارند.

در ادامه توضیحات مربوط به نحوه کار و کد هر بخش را مشاهده میکنید.

اپلیکیشن موبایل

برای این سیستم یک اپلیکیشن موبایل با سیستم عامل اندروید پیاده سازی کردیم. عملکرد کلی اپلیکیشن به صورت زیر میباشد:

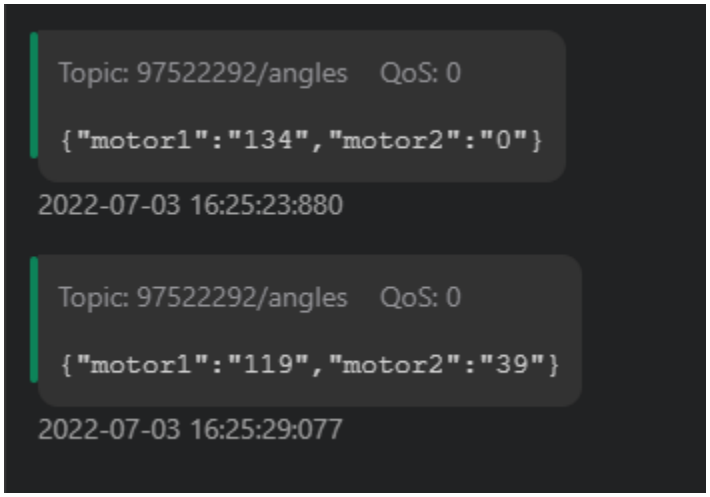
۱. مشخصات چندین ستاره مختلف را درون یک فایل csv در درون اپلیکیشن نگه میدارد. در فایل csv برای هر ستاره دو مشخصه RA و DEC وجود دارد که مختصات آن ستاره نسبت به سیاره زمین را مستقل از مکان و زمان نشان میدهد.
۲. هرگاه کاربر روی هر کدام از این ستاره ها ضربه بزند، اپلیکیشن با گرفتن مختصات کاربر و زمان ارسال درخواست و RA و DEC ستاره مورد نظر دو زاویه مربوط به دو موتور را با استفاده از برخی فرمول های فیزیکی محاسبه کرده و در درون دو باکس موجود در پایین اپلیکیشن نمایش میدهد.
۳. حال کاربر میتواند دو زاویه محاسبه شده در قسمت قبل را به دلخواه تغییر داده و زوایای مورد نظر خودش را ارسال کند یا بدون تغییر در دو عدد بدست آمده دکمه ارسال را زده تا دو زاویه به سرور ارسال گردند. زاویه اول مربوط به موتور پایینی است که عددی بین صفر تا ۱۸۰ درجه است که به صورت پادساعت گرد از غرب به شرق میچرخد. (موتور پایینی باید پشت به شمال باشد.) زاویه دوم نیز بین صفر تا ۱۸۰ است که زاویه صفر جهت روبه رو و زاویه ۱۸۰ پشت موتور را نمایش میدهد.
۴. زوایا به صورت یک Json و تحت پروتکل MQTT به سرور ارسال میگردد که در ادامه توسط تلسکوپ استفاده شوند.

سرور (MQTT)

برای انتقال داده ها در سیستم های IOT به دلیل کم بودن حجم RAM و محدودیت های سخت افزاری دیگری که در این سیستم ها وجود دارند، به جای استفاده از پروتکل های سنگینی مانند http میتوان از MQTT بهره برد. MQTT در واقع یک Message Broker است که از مفهوم Producer/Consumer استفاده میکند. (در MQTT نام آن ها به ترتیب به Receiver/Publisher تغییر پیدا کرده است)

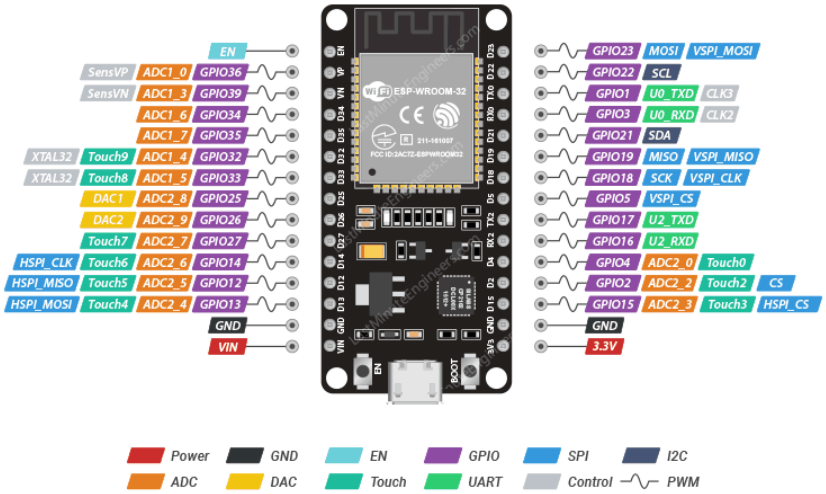
شیوه استفاده از MQTT به این صورت است: ابتدا Publisher (در اینجا اپلیکیشن موبایل) داده مورد نظرش (زوایای چرخش) را روی یک بستر مشخص که به آن ها Topic قرار میدهد. Receiver (میکروکنترلر) با اتصال به سرور و Subscribe کردن Topic مورد نظرش میتواند پیام های Publish شده روی آن را بخواند و درون برنامه خود استفاده کند.

در سیستم ما، اپلیکیشن داده ها را به صورت یک Json که از دو پارامتر motor1 و motor2 تشکیل شده بر روی Topic، 97522292/angles ارسال کرده و میکروکنترلر با Subscribe کردن این Topic فایل Json ارسال شده را میخواند و تلسکوپ را می چرخاند. نمونه ای از پیام های ارسالی به روی MQTT را در زیر مشاهده میکنید.

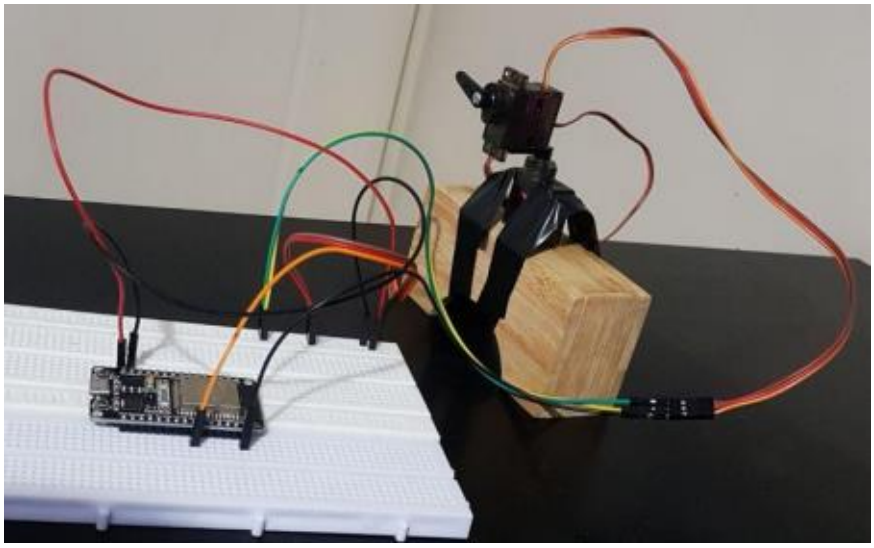


تلسکوپ

همانطور که پیش تر گفته شد، تلسکوپ از دو بخش میکروکنترلر و موتور ها جهت نمایش زوایا تشکیل شده است. میکروکنترلر مورد استفاده در این پروژه ESP32 WROOM است. برای استفاده از این میکروکنترلر، ابتدا آن را به روی برد بورد نصب کرده تا بتوانیم اتصالات سیمی با موتور ها را به راحتی با سیم جامپر انجام دهیم. سرو موتور های مورد استفاده مدل MG90S هستند. این موتور ها ۳ سیم خروجی دارند که سیم های مربوط به GND، VCC و DATA هستند. سیم DATA موتور اول را به پین ۱۸ و موتور دوم را به پین ۲۳ میکروکنترلر وصل کردیم. دو سیم اول را نیز به ترتیب به GND و VIN متصل کردیم.



نحوه قرار گیری موتور ها روی هم را نیز در تصویر زیر میتوانید مشاهده کنید. توجه داشته باشید که پشت موتور اول باید به سمت شمال باشد تا زوایای تلسکوپ به صورت درست محاسبه شوند.



حال کد میکروکنترلر را خط به خط بررسی میکنیم. برای استفاده از MQTT در میکروکنترلر، نیاز است که آن را به اینترنت متصل کنیم، برای اینکار باید اسم و رمز یک WIFI را به میکروکنترلر داده تا بتواند به آن متصل شود. همچنین اطلاعات مربوط به MQTT مانند سرور مورد استفاده، Topic ها و ... را نیز همان ابتدا مقدار دهی میکنیم.

```
const char* ssid = "WIFI";
const char* password = "PASSWORD";

const char* mqtt_server = "45.149.77.235";
const int mqtt_port = 1883;
const char* mqtt_username = "97522292";
const char* mqtt_password = "kYLB2MaR";
const char* mqtt_client_name = "ESP32Client";
const char* mqtt_out_topic = "97522292/connected";
const char* mqtt_in_topic = "97522292/angles";
```

برای چرخاندن موتور ها باید بدانیم که هر موتور Servo یک بازه Pulse width مخصوص به خود دارد که باعث میشود بتوانیم آن را با زاویه درست بچرخانیم. به همین منظور حداقل و حداکثر Pulse width مربوط به مدل MG90S را مقدار دهی میکنیم و پین و زاویه اولیه مربوط به موتور را نیز تعیین میکنیم.

```
Servo servo1, servo2;
int servo1_pin = 18;
int servo2_pin = 23;
int min_us = 400, max_us = 2400; // Min/Max pulse width (microseconds) for MG90S servo

int angle1 = 0;
int angle2 = 0;
```

در تابع setup ابتدا Port سریال را برای مشاهده log های چاپ شده begin میکنی. سپس تنظیمات اول مربوط به هر موتور را انجام داده و پین های آن ها را ست میکنیم. سپس تلاش میکنیم تا به WIFI تعریف شده در بالا متصل شویم و client های مربوط به MQTT را روی سرور و پورت تعریف شده ست کرده و تابع callback را برای خواندن داده های ارسال شده روی سرور ست میکنیم.

```
void setup() {
    Serial.begin(115200);

    // Allow allocation of all timers
    ESP32PWM::allocateTimer(0);
    ESP32PWM::allocateTimer(1);
    ESP32PWM::allocateTimer(2);
    ESP32PWM::allocateTimer(3);

    servo1.setPeriodHertz(50);    // standard 50 hz servo
    servo2.setPeriodHertz(50);

    servo1.attach(servo1_pin, min_us, max_us);
    servo2.attach(servo2_pin, min_us, max_us);


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

    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);
}
```

تابع callback به این صورت عمل میکند که در صورت subscribe کردن یک Topic و به ازای ارسال هر پیام روی آن Topic یک بار فراخوانی شده و عملیات های لازم را انجام میدهد. در ابتدای callback بعد از آمدن هر پیام، محتوای آن و Topic مربوطه را روی ترمینال چاپ میکند.

```
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");

  char json[length];
  for (int i = 0; i < length; ++i) {
      json[i] = (char)payload[i];
      Serial.print(json[i]);
  }
  Serial.println();
```

سپس متن پیام که String است را به صورت Json خوانده و زوایای دو موتور را آپدیت میکند.

```
if (doc.containsKey("motor1") && doc.containsKey("motor2")) {
    angle1 = (int)doc["motor1"];
    angle2 = (int)doc["motor2"];

    Serial.print("New angles: ");
    Serial.print(angle1);
    Serial.print(" ");
    Serial.println(angle2);
} else {
    Serial.println("The json does not contain \'motor1\' or \'motor2\'");
    return;
}
```

در تابع loop در هر مرحله در صورت متصل نبودن به سرور MQTT، تابع reconnect را صدا میزند و در صورت متصل بودن، Client loop را اجرا میکند. در نهایت نیز دو زوایای دو موتور را به آن ها ارسال کرده تا تلسکوپ در زاویه درست قرار گیرد.

```
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    servo1.write(angle1);
    delay(300);
    servo2.write(angle2);
}
```

در تابع reconnect سعی میکند تا اتصال به MQTT برقرار شود، در صورت متصل شدن روی connected یک پیام میفرستد و angles را subscribe میکند تا بتواند پیام های آن را بخواند. در صورت متصل نشدن نیز هر ۵ ثانیه این کار را تکرار میکند تا بالاخره به سرور متصل شویم.

```
void reconnect() {
    while (!client.connected()) {
        Serial.println("Attempting MQTT connection...");
        if (client.connect(mqtt_client_name, mqtt_username, mqtt_password)) {
            Serial.println("connected");
            client.publish(mqtt_out_topic, "esp32 is up.");

            client.subscribe(mqtt_in_topic);
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            // Wait 5 seconds before retrying
            delay(5000);
        }
    }
}
```