دانشگاه علم و صنعت ایران
دانشکده مهندسی کامپیوتر

پایان‌نامه پروژه کارشناسی
مهندسی کامپیوتر

عنوان:

شبکه‌های حسگر بی‌سیم برای کشاورزی و مدیریت آبیاری

نگارش:

سید محسن اسلام‌پناه

استاد راهنما:

دکتر امیرمهدی حسینی منزه

شهریورماه ۱۴۰۲


بسم الله الرحمن الرحیم

به نام خدا

دانشگاه علم و صنعت ایران
دانشکده مهندسی کامپیوتر

پایان‌نامه پروژه کارشناسی

عنوان: شبکه‌های حسگر بی‌سیم برای کشاورزی و مدیریت آبیاری

کمیته ممتحنین

استاد راهنما: دکتر امیرمهدی حسینی‌منزّه امضاء:

استاد داور: امضاء:

تاریخ:

سپاس و تشکر

از اســـاتید گران‌قدر، به‌ویژه جناب آقای دکتر امیرمهدی حســـینی منزه و مهندس علی جوادی که در طول مدت تحقیق، مرا از رهنمودها و تجربه‌های با ارزش خویش بهره‌مند ساختند، صمیمانه سپاسگزارم. همچنین بر خود لازم می‌دانم تا از حمایت‌ها و محبت‌های بی‌دریغ خانواده، دوستان و همسر عزیزم صمیمانه تشکر و قدردانی کنم.

و تمام آن را تقدیم می‌کنم به ناجی عالم عجّل‌الله تعالی فرجه و ارواحنا فداه به امید آن که قدمی کوچک در مسـیر رسـیدن به آن نجات و اصلاح حقیقی باشد.

چکیده

در این پروژه قصد داریم با هوشمند کردن آبیاری زمین‌های کشاورزی کیفیت محصولات را افزایش داده، مصرف آب را به حداقل رسانده و نیاز به نیروی انسانی را کاهش دهیم؛ برای این کار از یک شبکه حسگر بی‌سیم Mesh استفاده می‌کنیم که دارای چندین گره است. انرژی هر کدام از گره‌های آن به‌وسیله برداشتگر انرژی تأمین می‌شود و هر گره از حسگرهای رطوبت خاک، نور، دما، رطوبت هوا و.. بهره می‌برد که اطلاعات را جمع‌آوری و به گره مرکزی ارسال می‌کنند. گره مرکزی علاوه بر شبکه Mesh به شبکه دیگری برای ارتباط با اینترنت متصل است و اطلاعات دریافت شده از سایر گره‌ها را با استفاده از پروتکل MQTT به سرور ارسال می‌کند. از طرف دیگر کاربر به سایتی دسترسی دارد که توسط آن بر وضعیت زمین کشاورزی و مقادیر حسگرها نظارت می‌کند؛ با تحلیل این اطلاعات می‌توان شرایط زمین را برای کشت انواع درختان و گیاهان بررسی کرد یا با استفاده از برنامه‌های مدیریت آبیاری که در حال حاضر رایج هستند، به مدل مناسب زمین و محصول موردنظر دست‌یافت. یکی دیگر از وظایف سایت ذکر شده، مدیریت آبیاری با کمک یک سرور پردازشگر است. برای مدیریت آبیاری بخشی در سایت وجود دارد که کاربر در آن بخش دو انتخاب دارد. انتخاب اول زمان‌بندی می‌باشد. در آن کاربر یک بازه زمانی را در روز انتخاب می‌کند تا آبیاری در آن ساعات انجام شود، همچنین امکان تعیین تعداد روزهای میان دو آبیاری وجود دارد و برای هر شیر آب می‌تواند مدل متفاوتی تعریف کند. انتخاب دوم تعیین شرط است که کاربر باید به ازای هر حسگر، یک بازه مشخص کند که اگر مقادیر فعلی حسگرها در آن بازه قرار گرفت، آبیاری شروع شود که در این بخش نیز می تواند برای هر شیر مدل جداگانه تعریف کند.

واژه‌های کلیدی: شبکه حسگر بی‌سیم، اینترنت‌اشیاء، آبیاری، سرور، ESP32، پنل مدیریت و کشاورزی

فهرست مطالب

فهرست تصاویر

۱. فصل اول: مقدمه

مقدمه

با توجه به یک سری مسائل مطرح شده در سال های اخیر مانند بحران جهانی آب، خشکسالی و کمبود منابع آب شیرین، محققان به دنبال راه حلی برای این مشکلات با منطقی کردن مصرف آب در بخش کشاورزی به عنوان یکی از پرمصرف ترین بخش های آب هستند[۱]. در دهه‌های اخیر، موضوع مدیریت منابع آب به یکی از مسائل حیاتی و بحرانی در ایران نیز تبدیل شده است. با افزایش جمعیت، تغییرات اقلیمی، و تبدیل‌شدن آب به منبع محدود، نیاز به روش‌های نوین و هوشمند در زمینه آبیاری از اهمیت چندبرابری برخوردار است. در این راستا، به‌منظور بهبود بهره‌وری، کاهش هدررفت آب، و مدیریت بهینه منابع آب، ترکیبی از تکنولوژی‌های نوین، از جمله اینترنت‌اشیاء[۱] و شبکه‌های حسگر بی‌سیم، به‌عنوان ابزارهایی کارآمد موردتوجه قرار گرفته‌اند. در صنعت کشاورزی، شبکه‌های حسگر بی‌سیم می‌توانند ابزار مهمی برای ارتقای رشد اقتصادی باشند[۲].

آبیاری هوشمند به‌عنوان یکی از اجزای اصلی کشاورزی هوشمند و مدیریت هوشمند منابع آب مطرح می‌شود. این سیستم‌ها از تجهیزات مبتنی بر حسگرها[۲] استفاده می‌کنند تا داده‌های مرتبط با شرایط آبیاری را

[۱] Internet Of Things

[۲] Sensor

به‌صورت پیوسته و برخط جمع‌آوری و انتقال دهند. این داده‌ها شامل اطلاعاتی نظیر رطوبت خاک، رطوبت هوا، دما، میزان نور و.. هستند. با بهره‌گیری از این داده‌ها سیستم‌های آبیاری هوشمند می‌توانند به طور دقیق‌تر، بهینه‌تر، و به‌موقع تر تصمیمات مربوط به آبیاری را اتخاذ کنند.

از طرف دیگر، اینترنت‌اشیاء به ما این امکان را می‌دهد که از راه دور دستگاه‌های مرتبط با آبیاری را مدیریت کنیم و دسترسی به داده‌های حسگرها و آبیاری را داشته باشیم. این دسترسی به داده‌ها به کشاورزان و مدیران منابع آبیاری این امکان را می‌دهد که سیستم‌های آبیاری خود را بهبود بخشند و کمترین ضایعات منابع آبی را داشته باشند. با توجه به این که ایران مشکل کمبود آب روبرو است، و این مشکل به دلیل خشکسالی‌ها و کاهش تدریجی منابع آب تازه در حال تشدید نیز هست. سیستم‌های آبیاری هوشمند با کنترل دقیق و بهینه مصرف آب می‌توانند به حفظ منابع آبی کمک کنند.

حسن دیگر سیستم‌های آبیاری هوشمند این است که از فناوری‌های جدید برای ارتقاء بهره‌وری و کاهش هدررفت آب استفاده می‌کنند. این افزایش بهره‌وری کشاورزی می‌تواند در تولید محصولات کشاورزی بیشتر با هزینه‌های کمتر نتیجه دهد. در این سیستم‌ها، فقط زمانی آبیای انجام می‌شود که گیاه نیاز به آب داشته باشد و جلوی آبیاری بیش از حد به گیاه و همچنین تشنه ماندن آن گرفته می‌شود، در نتیجه محصول بهتری خواهیم داشت.

با استفاده از این سیستم‌ها، زحمت کشاورزها و به طور خاص باغ دارها نیز کمتر می‌شود. خیلی ازکار ها به صورت خودکار انجام خواهد شد و در بسیاری از موارد دیگر نیازی به حضور فیزیکی نیست که کاربرد خیلی زیادی برای افرادی دارد که در حاشیه شهرها باغ‌هایی به منظور تفریح تهیه می‌کنند و به جلوگیری از هدررفت زمان و هزینه آن‌ها کمک خواهد کرد.

با افزایش محبوبیت اینترنت‌اشیاء، ایده‌های پیرامون فناوری هوشمند کشاورزی افزایش یافته است[۳]. نمونه‌ها و مقاله‌هایی از پیاده‌سازی آبیاری هوشمند بر مبنای اینترنت‌اشیاء[٤]، شبکه‌های حسگر بی‌سیم در کشاورزی[٥] به صورت جداگانه وجود دارد. نوآوری این پروژه در این است که ترکیبی از تکنولوژی‌ها و

ایده‌ها می‌باشد و هم‌زمان از اینترنت‌اشیاء، شبکه حسگر بی‌سیم، برداشتگر انرژی، Esp-Mesh و چند مبحث دیگر بهره برده است.

این پایان‌نامه شامل شش فصل می‌باشد که در فصل اول آن کلیاتی از قبیل مقدمه، انگیزه و هدف از انتخاب پروژه و چالش‌ها و ساختار کلی پایان‌نامه را گفته ایم. در فصل دوم به معرفی ابزارها و تعریف مفاهیم آن خواهیم پرداخت. در فصل سوم راه‌کارهای پیشنهادی برای حل چالش‌ها را ارائه داده و ساختار پروژه را به‌طور فنی معرفی می‌کنیم. فصل چهارم شامل جزئیات فنی سخت افزار ما و نحوه برنامه نویسی آن می‌باشد. فصل پنجم طراحی سایت و ساخت سرور را توضیح خواهد داد و در نهایت فصل ششم با نتیجه گیری و صحبت‌هایی پیرامون برنامه‌های آینده خاتمه خواهد یافت.

۲. فصل دوم: مفاهیم و ابزارها

در این فصل قصد داریم که به ادبیات یکسانی برسیم و تمام مفاهیم را تعریف کرده و تمام اجزا را معرفی کنیم. در این پایان‌نامه بنابراین است که مطالب تا جای ممکن به ساده‌ترین زبان گفته شود، تا برای افرادی که به این زمینه علاقه‌مندند اما دانش فنی ندارند نیز قابل استفاده و قابل فهم باشد. به همه پیشنهاد می‌شود که قبل از سایر فصل‌ها حتماً این فصل را مطالعه کنند.

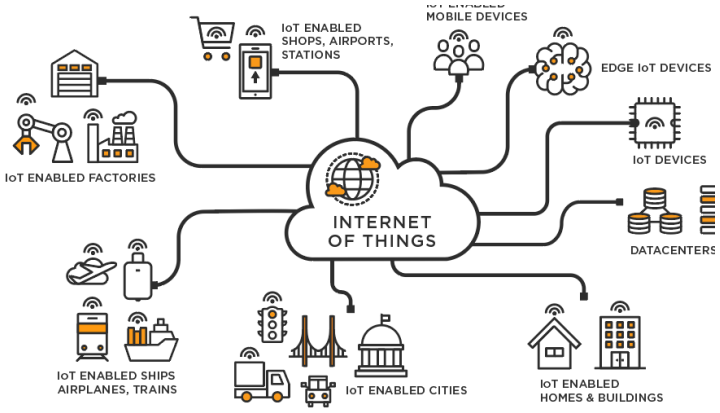
۲ -۱ اینترنت‌اشیاء[۱]

اینترنت‌اشیاء، معنی این دارد که اشیاء معمولی که تا قبل از این به اینترنت متصل نبودند، حالا به اینترنت متصل می‌شوند. این اشیاء می‌توانند چیزهای مختلفی باشند، مثلاً یخچال شما، ماشین، لامپ‌ها یا حتی لباس‌ها. وقتی این اشیاء به اینترنت متصل می‌شوند، می‌توانند داده‌هایی از محیط خود جمع‌آوری کنند و این داده‌ها را به سرورها یا سیستم‌های دیگری ارسال کنند.

مثلاً اگر یخچال شما به اینترنت متصل باشد، می‌تواند دمای داخل آن را نظارت کند و شما را به وضعیت فعلی آن آگاه کند. در کل، اینترنت‌اشیاء به اشیاء اجازه می‌دهد که با هم و با ما تعامل کنند و اطلاعات مهمی

[۱] Internet of Things (IoT)

را جمع‌آوری و به اشـتراک بگذارند تا زندگی ما را راحت‌تر و بهتر کنند. برای درک بهتر **تصـویر ۱** را مشـاهده کنید.



تصویر ۱: اینترنت‌اشیاء

۲ـ۲ ساختار شبکه Mesh

شـبکه Mesh[۱] معنای این دارد که دسـتگاه‌ها یا اشـیاء مختلف می‌توانند با یکدیگر مسـتقیماً ارتباط برقرار کنند، تقریباً مثل یک جمعی که دسـت به دسـت یک چیز را به یکدیگر می‌دهند. هر دسـتگاه در این شـبکه می‌تواند به عنوان یک گره[۲] عمل کند و داده‌ها را از یک گره به گره دیگر منتقل کند.

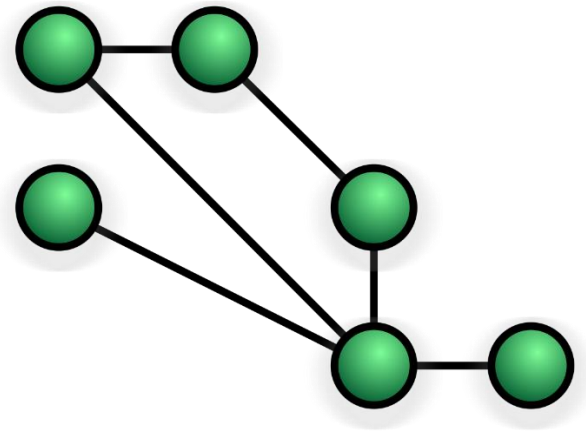
این شبکه بسیار انعطاف‌پذیر است. اگر یک گره در مسیر ارسال داده‌ها مشکل داشت یا قطع شد، داده‌ها می‌توانند از مسیر دیگری انتقال پیدا کنند. این باعث می‌شود که شبکه Mesh بسیار پایدار و قابل اعتماد باشد.

معمولاً این نوع از شـبکه برای ارتباطات بی‌سـیم و اتصـال دسـتگاه‌های هوش مصـنوعی، حسـگرها، دسـتگاه‌های خانگی هوشـمند و حتی خودروهای خودران اسـتفاده می‌شـود. به عبارت دیگر، شـبکه Mesh به اشـیاء این امکان را می‌دهد تا با یکدیگر در ارتباط باشـند و اطلاعات را بین آنها به اشـتراک بگذارند، حتی اگر فاصله زیادی بین آنها وجود داشته باشد. در **تصویر ۲** نمونه ای از یک شبکه mesh را می‌توان دید.

---

[۱] Network

[۲] Node



تصویر ۲: شبکه Mesh جزئی

۲ -۳ حسگرها

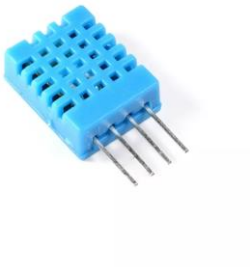
همان طور که از نام این پروژه مشخص است، حسگرها بخش مهمی از آن می‌باشند و اطلاعات مهمی را به ما می‌دهند، در این بخش با حسگرهایی که در این پروژه استفاده شده‌اند آشنا می‌شویم و تصاویر آن‌ها را می‌بینیم اما کار به این‌ها محدود نمی‌شود و شما می‌توانید مطابق نیازتان از حسگرهای دیگری هم استفاده کنید.

۲ -۳ -۱ حسگر DHT-11[۱]

حسگر DHT-11 که در **تصویر ۳** می توان آن را دید، یک نوع حسگر دما و رطوبت دیجیتال است که معمولاً در پروژه‌ها و سیستم‌های الکترونیکی مورد استفاده قرار می‌گیرد. این حسگر به شکل یک ماژول کوچک عرضه می‌شود و دارای دو حسگر اصلی برای اندازه‌گیری دما و رطوبت است. در زیر توضیحات بیشتری در مورد این حسگر آمده است:

[۱] Digital Temperature And Humidity Sensor

- اندازه‌گیری دما: DHT-11 قادر به اندازه‌گیری دما با دقت خوبی است. محدوده اندازه‌گیری دما از حدود ۰ درجه سانتی‌گراد تا ۵۰ درجه سانتی‌گراد است. این اندازه‌گیری‌ها به وسیله یک ترمیستور در حسگر انجام می‌شود.
- اندازه‌گیری رطوبت: DHT-11 همچنین قادر به اندازه‌گیری رطوبت محیط است. محدوده اندازه‌گیری رطوبت از حداقل ۲۰% تا حداکثر ۸۰% است.



تصویر ۳: حسگر DHT-11

۲ -۳ -۲ حسگر LDR[۱]

حسگر LDR **(تصویر ٤)** به عنوان "حسگر مقاومت نوری" یا "حسگر فشار نوری" نیز شناخته می شود. این حسگر یک نوع حسگر مقاومتی است که به تغییرات شدت نوری محیط حساسیت دارد. در واقعیت، مقاومت این حسگر به اندازه شدت نوری کاهش یا افزایش می‌یابد. وقتی نور بیشتری بر روی LDR تابیده می‌شود، مقاومت آن کمتر می‌شود و در مقابل، در شرایط کم نوری، مقاومت آن بیشتر می‌شود.

[۱] Light Dependent Resistor

تصویر ٤: حسگر LDR

۲ -۳ -۳ حسگر رطوبت خاک

حســـگر رطوبت خاک دو نوع مقاومتی و خازنی دارد که یک مقدار آنالوگ به ریزکنترل‌گر می‌دهند که می‌توان آن را به یک عدد بین ۰ تا ۱۰۰ نگاشـــت کرد و همان طور که از اســـم آن پیداســـت، رطوبت خاک را بدست می‌آورد. مدل‌های گوناگونی دارد که ما محصولی که در **تصویر ٥** دیده می‌شود، استفاده کرده‌ایم.

تصویر ٥: حسگر رطوبت خاک مقاومتی

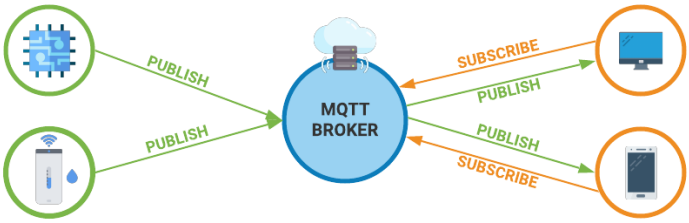
۲ -٤ پروتکل‌های شبکه

پروژه ما از دو شبکه تشکیل شده است، یک شبکه محلی برای ارتباط گره‌های دارای حسگر گره مرکزی برای ارسال اطلاعات دریافتی از حسگرها و دیگری اینترنت برای گره مرکزی جهت دریافت دستورات کاربر و ارسال اطلاعات حسگرها به سرور که در این بخش پروتکل‌های مورد استفاده در هر کدام را معرفی کرده‌ایم.

۲ -٤ -۱ پروتکل MQTT[۱]

MQTT یک پروتکل ارتباطی است که برای انتقال داده‌ها و پیام‌ها در اینترنت و به خصوص بین دستگاه‌ها در شبکه‌های اینترنت‌اشیاء استفاده می‌شود[٦]. ویژگی اصلی این پروتکل سبک، سریع و ساده بودن آن است که آن را برای استفاده در شبکه‌های اینترنت‌اشیاء به یکی از بهترین انتخاب‌ها تبدیل می‌کند. در **تصویر ٦** می‌توان نمای کلی آن را دید.

طرز کار این پروتکل به این صورت است که یک واسط یا دلال[۲] وجود دارد که شامل موضوع[۳]های مختلفی می‌شود (می‌توانیم هر موضوع دلخواهی را در آن داشته باشیم) و دستگاه‌های کاربر[٤] به آن دلال متصل می‌شوند و می‌توانند به موضوع‌های خاصی گوش فرا دهند یا به اصطلاح تخصصی آن اشتراک[٥] بگیرند تا پیام‌های آن را دریافت کنند یا برای موضوع دلخواهشان پیام منتشر[٦] کنند.



تصویر ٦: MQTT

۲ -٤ -۲ پروتکل ESP-Mesh

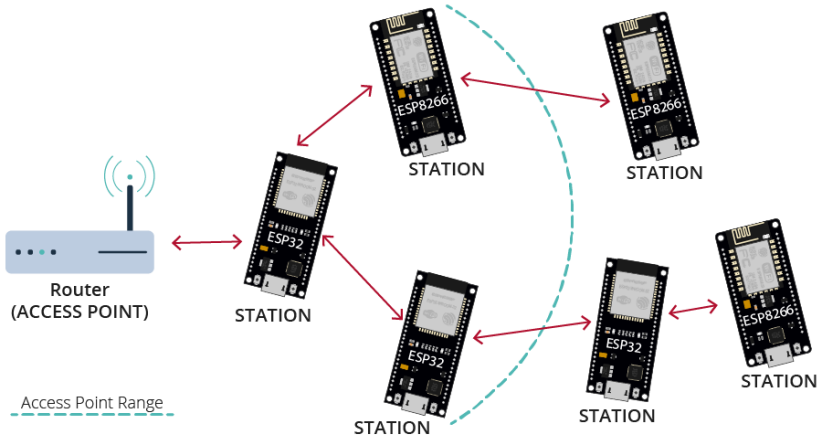
ESP-Mesh یک پروتکل شبکه است که بر روی زیر ساخت Wi-Fi ساخته شده است[۷]. ESP-Mesh اجازه می‌دهد تا گره‌های مختلف که در یک منطقه فیزیکی بزرگ (هم در داخل و هم در خارج از منزل) پخش

---

۱ Message Queuing Telemetry Transport
۲ Broker
۳ Topic
٤ Client
٥ Subscribe
٦ Publish

شـــده‌اند، تحت یک WLAN[۱] واحد (شـــبکه محلی بی ســـیم) به هم متصـــل شـــوند. ESP-MESH خود سـازماندهی و خودترمیمی اسـت به این معنی که شـبکه می‌تواند به طور مسـتقل سـاخته و نگهداری شـود. این تکنولوژی به دســتگاه‌ها این امکان را می‌دهد تا مســتقیماً با یکدیگر صــحبت کنند و اطلاعات را به اشــتراک بگذارند، بدون نیاز به یک نقطه مرکزی یا اینترنت پرســرعت. در **تصــویر ۷** تعدادی ESP را می‌بینیم که از این شبکه استفاده می‌کنند.

در یک شبکه، هر دستگاه به عنوان یک گره عمل می‌کند و می‌تواند اطلاعات را به دیگر دستگاه‌ها منتقل کند یا از آن‌ها دریافت کند. این اطلاعات از طریق دستگاه‌های دیگر در شبکه انتقال پیدا می‌کنند تا به مقصد برسند. به این ترتیب، شبکه به صورت توزیع شده عمل می‌کند. ESP-MESH به طور خاص برای دستگاه‌های مبتنی بر تراشه‌های ESP8266 و ESP32 طراحی شده است.



تصویر ۷: شبکه ESP-Mesh

[۱] Wireless local-area network

۲ -۵ ریزکنترل‌گرها

ریزکنترل‌گرها مغز متفکر، مدیریت کننده و بخش اصلی پروژه ما هستند. حسگرها به آن متصل می‌شوند، وظیفه ایجاد شبکه، ارسال اطلاعات حسگرها، ارتباط با سرور، تغییر وضعیت شیرهای آب و.. بر عهده آن‌ها می‌باشد و قابل برنامه نویسی‌اند و باید امکانات آن‌ها به نحوی باشد که تمام نیازهای ما را برطرف کند.

۲ -۵ -۱ ریزکنترل‌گر ESP32

ESP32 یک تراشه[۱] ریزکنترل‌گر و ماژول[۲] ارتباط بی‌سیم است که توسط شرکت Espressif Systems طراحی و توسعه داده شده است. این تراشه به عنوان یک پلتفرم[۳] سخت‌افزاری کامل عمل می‌کند و قابلیت کنترل دستگاه‌ها، ارتباط با دستگاه‌های دیگر و اتصال به شبکه‌های بی سیم را فراهم می سازد[۸]. ویژگی‌های این تراشه عبارت‌اند از:

- دارای پردازشگر دو هسته‌ای که به صورت مستقل از یکدیگر عمل می‌کنند.
- دارای ماژول وای‌فای داخلی که امکان اتصال به شبکه‌های بی‌سیم Wi-Fi را فراهم می‌کند.
- امکان اتصال به دستگاه‌های بلوتوث[۴]
- پین‌های مختلف برای اتصال دستگاه و حسگرها به عنوان خروجی یا ورود.
- و..

۲ -۵ -۲ ریزکنترل‌گر ESP8266

ESP8266 نسخه قدیمی‌تر ESP32 می‌باشد که امکانات خیلی کمتری را ارائه می‌دهد (برای مثال بلوتوث ندارد) با این حال در بسیاری از پروژه‌های دانشگاهی که از ESP32 استفاده می شود، ESP8266 نیز تمام نیاز

---

۱ Chip
۲ Module
۳ Platform
۴ Bluetooth

را بر طرف ســـاخته و امکانات لازم را با هزینه‌ی کمتر در اختیار ما می‌گذارد. در **تصـــویر ۷** می‌توانید بردهای توسعه ESP32 و ESP8266 را مشاهده کنید.

۲ -٦ تامین انرژی

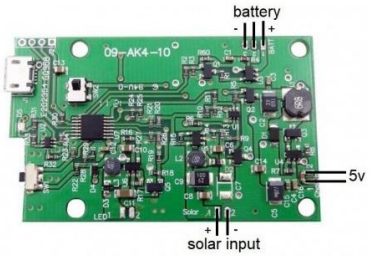
در این پروژه فرض شده است که گره‌های دارای حسگر دسترسی به برق مستقیم ندارند زیرا برای استفاده از برق مســتقیم برای هر گره در زمین‌های کشــاورزی نیاز به ســیم کشــی زیادی دارد که هزینه آن ســر به فلک می‌زند. در این قســمت ابزارهای مورد نیاز تأمین انرژی گره‌ها را معرفی می‌کنیم. (گره مرکزی به برق مســتقیم متصل می‌شود.)

۲ -٦ -۱ باتری لیتیوم پلیمر

باتری‌های لیتیوم پلیمر یا Li-Po به علت وزن سبک، حجم کمتر و کارایی بالا، در بسیاری از دستگاه‌های قابل حمل از جمله تلفن‌های همراه، لپ‌تاپ‌ها، دســتبندهای هوشــمند و ریزکنترل‌گرها اســتفاده می‌شــوند. باتری‌های Li-Po دارای توان بالا و نرخ تخلیه بهتری نسبت به بسیاری از باتری‌های دیگر دارند. این به معنای این است که آنها قادر به تأمین انرژی بیشتری در زمان‌های کوتاه مدت می‌باشند. در این پروژه، باتری با ماژول شارژر به سلول خورشیدی متصل می‌شود و مانند خازن عمل خواهد کرد.

۲ -٦ -۲ ماژول شارژر Module 09-AK-10

باتری‌های لیتیوم پلیمر را نمی‌توان مســتقیم به ســلول خورشــیدی متصــل کرد، زیرا احتمال شــارژ و تخلیه بیش از حد وجود دارد. برای جلوگیری از این اتفاق باید از ماژول‌های شــارژر اســتفاده کرد که a09-AK-10 به علت داشتن خروجی ثابت ۵ ولت و ورودی پنل خورشیدی انتخاب مناسبی می‌باشد. در **تصویر ۸** نمونه‌ای از را می‌شود مشاهده کرد.



تصویر ۸: ماژول شارژر

۲ -۶ -۳ پنل خورشیدی CCLamp CL-638WP

پنل خورشیدی CCLamp CL-638WP که در **تصویر ۹** آن را مشاهده می‌کنیم، با توان ۳/۸ وات[۱]، ولتاژ بیشینه‌ی ۶ ولت و جریان بیشینه‌ی ۰/۶۳ آمپر[۲] را تولید خواهد کرد و خروجی‌های متنوعی برای اتصال به دستگاه‌های مختلف و قیمت به صرفه‌ای دارد که از مزایا آن محسوب می‌شود و از آن در این پروژه استفاده کرده‌ایم.

تصویر ۹: CCLamp CL-638WP

[۱] Watt

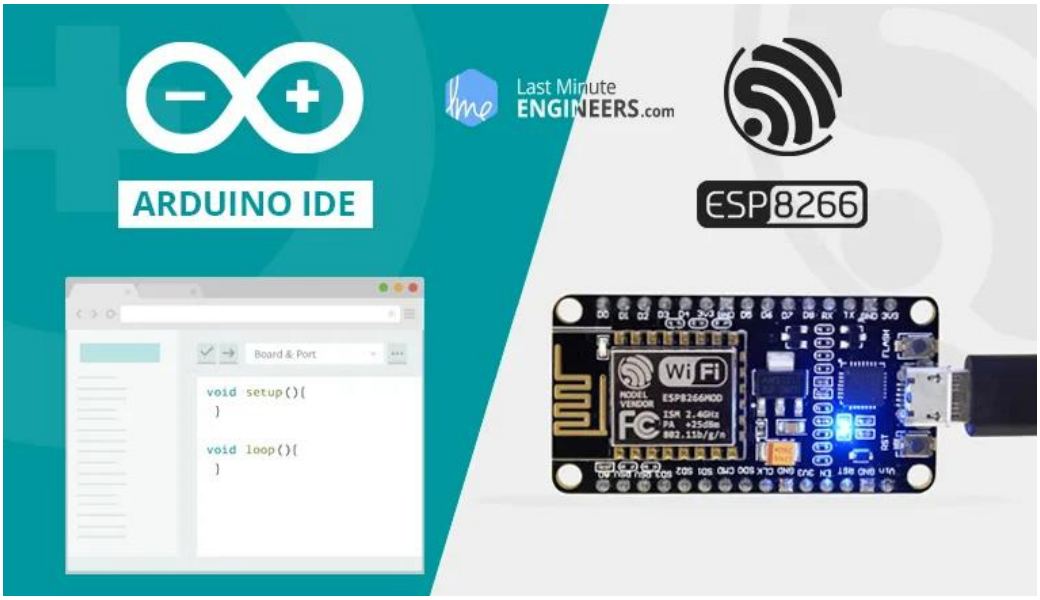
[۲] Amper

۲ -۷ نرم افزار و فریم‌ورک[۱]

در طول این پروژه از نرم افزار و فریم‌ورک‌های مختلفی برای برنامه‌نویسی ریزکنترل‌گرها، طراحی سایت و سرور استفاده کرده‌ایم که در این قسمت معرفی خواهیم کرد. برای برنامه‌نویسی ریزکنترل‌گرها از کتابخانه‌هایی بهره برده‌ایم اما برای پرهیز از تخصصی شدن این فصل، در بخش‌های مختلف فصل چهارم آن‌ها را معرفی می‌کنیم.

۲ -۷ -۱ ویرایشگر Arduino IDE

Arduino IDE (**تصویر ۱۰**) یک نرم‌افزار متن‌باز[۲] رایگان است که از آن برای برنامه‌نویسی روی ریزکنترل‌گرها استفاده می‌شود. Arduino IDE کاملاً از قواعد C++ پیروی می‌کند و کتابخانه‌های بسیاری برای آن نوشته شده است که کار برنامه‌نویس‌ها را راحت‌تر می‌کنند و ما از آن برای نوشتن کدهای برنامه و بارگذاری بر روی ESP32ها کرده‌ایم.



تصویر ۱۰: Arduino IDE

[۱] Framework

[۲] Open source

۲-۷-۲ MQTT Broker

Brokerها سرور و برنامه‌های واسطی هستند که تمام پیام‌های MQTT برای آن‌ها فرستاده می شود و آن جا قرار می‌گیرد و بخش مهم و جدا نشدنی MQTT هستند. برای این پروژه باید از یک Broker بهره ببریم که برای این منظور، Broker های عمومی[۱] رایگان وجود دارد که از بین آن ها، ioitiran.ir را به علت کم نقص بودنش انتخاب کرده ایم.

۲-۷-۳ طراحی سایت

ما نیاز داریم که برای ایجاد پنل مدیریت، یک سایت طراحی کنیم که امکانات مدنظر ما را داشته باشد. تأکیدی بر استفاده از تکنولوژی خاصی نداریم و برای هرکس با توجه به مهارت‌هایش می‌تواند متفاوت باشد، بنده با موارد زیر را انتخاب کرده‌ام:

- HTML/CSS: که بخش جدانشدنی طراحی هر سایتی است.
- JavaScript: برای ایجاد فضای تعاملی، ارتباط با سرور، ارسال پیام‌ها و...
- BootStrap: یک کتابخانه مبتنی بر HTML/CSS/JS برای ایجاد سایت زیباتر، با قابلیت تنظیم خودکار اندازه[۲] سایت بر اساس دستگاه کاربر است.
- Paho.MQTT: از این کتابخانه برای برقراری ارتباط با دلال MQTT و دریافت مقدار حسگرها و منتشر کردن مدل آبیاری تعیین شدن توسط کاربر در موضوع[۳] مورد نظر استفاده می‌کنیم.
- Xampp: برای ساخت یک سرور آزمایشی درون سیستم کاربر برای راه اندازی سایت.

---

[۱] Public

[۲] Responsive

[۳] Topic

۲ -۷ -٤ سرور پردازشگر

ما به یک سرور نیاز داریم که دستورها و مدل‌های آبیاری را از پنل مدیریت از طریق MQTT دریافت کند و مقدار حسگرها را هم از گره مرکزی، با MQTT دریافت کند و با توجه به مدل تعیین شده برای آبیاری، وضعیت شیرهای آب را به گره مرکزی اطلاع دهد و گره مرکزی در صورت نیاز وضعیت شیرها را تغییر دهد. برای ساخت این Server از Node.js استفاده کرده‌ایم.

۲ -۸ سایر تجهیزات

برای ساخت این پروژه آزمایشی علاوه بر مواردی که ذکر شد، از ابزارهای جزئی دیگری نیز بهره بردیم که نیازی به بخش جداگانه ندارند و فقط به نام بردن آن‌ها و توضیحی مختصر اکتفا می‌کنیم که در زیر شما می‌توانید تعدادی را که از بقیه بااهمیت‌تر هستند را مشاهده کنید:

- **Breadboard**: که تمام تجهیزات از قبیل ESP32 ، حسگرها، سیم‌های اتصال و... روی آن قرار می‌گیرند.
- **سیم‌ها اتصال**[۱]: برای اتصال بخش‌های مختلف پروژه.
- **کابل‌های MicroUSB**: برای تأمین برق ESP32 ها و همچنین برنامه نویسی آن‌ها.
- **مقاومت**[۲]: برای راه‌اندازی حسگرها.
- **LED:** جایگزین شیرهای آب، برای آزمودن پروژه.
- **پاوربانک:** پنل خورشیدی به عنوان نمونه فقط برای یکی از ESPها استفاده شده است و برای بقیه از پاوربانک بهره برده‌ایم.

[۱] Jumper wire

[۲] Resistor

۳. فصل سوم: چالش‌ها و راهکارها

در این فصـــل قصـــد داریم چالش‌ها را گفته و راهکارهای ارائه شـــده برای حل آن‌ها را مطرح و علت انتخاب سخت‌افزارها و روش‌های مختلف را بررسی کنیم. در انتهای فصل به درک درست‌تری از ساختار فنی پروژه می‌رسیم و آماده‌ی رفتن به سراغ فصل چهار و یک شیرجه‌ی عمیق درون قسمت فنی پروژه شویم.

۳ - ۱ چالش‌ها

بخش مهمی از هر پروژه و پژوهش، چالش‌های آن می‌باشد که جذابیت ویژه‌ای دارند و پس از رسیدن به راهکار، ثبت کردن آن‌ها می‌تواند به بقیه کمک کند تا در صـــورت مواجه شـــدن با موارد مشـــابه، از تجربه‌های پیشـــین بهره بگیرند، در این پروژه ما نیز با چالش‌هایی مواجه شـــدیم که ســـه تا از مهم‌ترین‌های آن را در ادامه بررسی می‌کنیم.

۳ - ۱ - ۱ انتخاب پروتکل مناسب برای ایجاد شبکه بی سیم

برای برقراری ارتباط میان حســـگرها باید شـــبکه‌ای انتخاب شـــود که بیشـــترین برد[۱] ممکن را دهد، انرژی کمتری مصـــرف کند، پیچیدگی کمتر، پیاده ســـازی راحت‌تر و با امکانات در اختیار ما هم‌خوانی داشـــته باشـــد.

[۱] Range

برای این کار بین چند گزینه باید انتخاب می‌کردیم که در فصل‌های آینده، آن‌ها را مقایسه کرده و کامل توضیح داده‌ایم.

۳ -۱ -۲ ارتباط هم‌زمان گره مرکزی با اینترنت و سایر گره‌ها

گره مرکزی ما باید هم‌زمان عضو دو شبکه باشد: شبکه داخلی با سایر گره‌های دارای حسگر، و شبکه‌ای که به اینترنت متصل است؛ تا بتواند دستورات لازم را از سرور دریافت کند و اطلاعات دریافتی از حسگرها را برای آن ارسال کند. زمانی اهمیت این چالش بیشتر درک خواهد شد که بدانیم هر دوی این شبکه از Wi-Fi[۱] استفاده خواهد کرد.

۳ -۱ -۳ ولتاژ[۲] متغیر پنل‌های خورشیدی[۳]

بعد از اتصال ماژول‌های شارژر و باتری‌ها به سلول‌های خورشیدی، ولتاژ خروجی اکثر آن‌ها برابر خواهد شد با ولتاژ پنل خورشیدی، در صورتی که انتظار می‌رفت برابر با ولتاژ دو سر باتری باشد و به علت متغیر بودن ولتاژ پنل خورشید، ریزکنترل‌گر[۴] ما به خوبی کار نمی‌کند به خصوص اگر ولتاژ نامی پنل حدود ٦ ولت یا کمتر باشد.

۳ -۲ انتخاب‌ها و راهکارها

حال که از تعدادی از چالش‌ها و جزئیات آن‌ها مطلع شدید، در این قسمت برای هر کدام از آن‌ها چند راهکار پیشنهاد داده و راهکارها را با هم مقایسه کرده و از بین آن‌ها بهترین‌ها را انتخاب می‌کنیم و در ابتدا به اصلی‌ترین انتخاب، یعنی انتخاب ریزکنترل‌گر می‌پردازیم و سپس سراغ چالش‌های مطرح شده می‌رویم.

---

۱ Wireless Fidelity

۲ Voltage

۳ Solar panels

٤ Microcontroller

۳ -۲ -۱ انتخاب ریزکنترل‌گر

ریزکنترل‌گرها مغز متفکر این پروژه و عنصر اصلی آن می‌باشند پس می‌توان نتیجه گرفت که انتخاب آن‌ها اساسی‌ترین انتخاب و تصمیم گیری در طول این پروژه خواهد بود. ریزکنترل‌گرها انواع مختلفی دارند مانند Arduino Uno، Raspberry Pi، STM32، ATmega Series، Intel Edison، Particle Photo، ESP32، ESP8266 و... که ما در هر پروژه با توجه بر نیازهای آن پروژه و منافع حداکثری خودمان، ESP32 را انتخاب می‌کنیم. ESP32 یک تراشه ریزکنترل‌گر و ماژول ارتباط بی‌سیم است که توسط شرکت Espressif Systems طراحی و توسعه داده شده است و به دلایل زیر گزینه مناسبی برای این پروژه است:

- **پردازشگر[۱] دو هسته‌ای:** ESP32 دارای دو هسته پردازشگر است که به ما امکان می‌دهد کارهای مختلف را به صورت موازی انجام دهیم.
- **قیمت مناسب ESP32 :**به عنوان یک میکروکنترلر با ویژگی‌های پیشرفته، قیمت مناسبی دارد که این امکان را به برنامه‌نویسان و سازندگان می‌دهد تا پروژه‌های خود را با هزینه کمتری انجام دهند.
- **مصرف انرژی**: ESP32 مصرف انرژی پایینی دارد و حالت‌هایی را پشتیبانی می‌کند که مصرف انرژی آن را بیشتر هم کاهش می‌دهد (مثلاً برای دستگاه‌هایی از سلول‌های خورشیدی تغذیه می‌شوند مناسب است).
- **سهولت برنامه‌نویسی**: هم می‌توان از Arduino IDE و هم MicroPython استفاده کرد.
- **اکوسیستم**: ESP32 اکوسیستم بسیار گسترده و پویا دارد و شامل مجموعه‌ای از منابع، ابزار و سیستم‌ها می‌شود که توسعه‌دهندگان به کمک آن‌ها می‌توانند پروژه‌های مختلفی را بر روی بردهای ESP32 ایجاد کنند.

[۱] Proccesor

- **جامعه:** ESP32 جامعه‌ای بزرگ و پرانرژی از توسعه‌دهندگان و علاقه‌مندان دارد. این جامعه فعال در انجمن‌ها، شبکه‌های اجتماعی و دیگر منابع آنلاین حضور دارد و به تبادل اطلاعات و حل مشکلات کمک می‌کند.

۳ -۲ -۲ حالت ایستگاه[۱] و نقطه‌دسترسی[۲]

هنگامی که یک دستگاه از طریق Wi-Fi به دستگاه دیگری متصل می‌شود، در حالت ایستگاه و آن دستگاهی که دیگران به آن متصل می‌شوند در حالت نقطه دسترسی قرار دارد. ESP32 می‌تواند هم‌زمان در حالت ایستگاه و نقطه دسترسی باشد. این ویژگی در آینده به کمک ما می‌آید و چالش ما را برطرف خواهد کرد.

۳ -۲ -۳ انتخاب پروتکل شبکه حسگر بی‌سیم

این پروژه از دو شبکه تشکیل شده است که یکی از آن‌ها شبکه‌ی داخلی برای اتصال گره‌های دارای حسگر به گره مرکزی است. که برای این شبکه چند گزینه اصلی وجود دارد که در این قسمت سه تا از آن‌ها را مطرح کرده و مزایا و معایبشان را بررسی می‌کنیم. یکی از آن‌ها را که برای شرایط پروژه مناسب‌تر است را باید انتخاب کنیم.

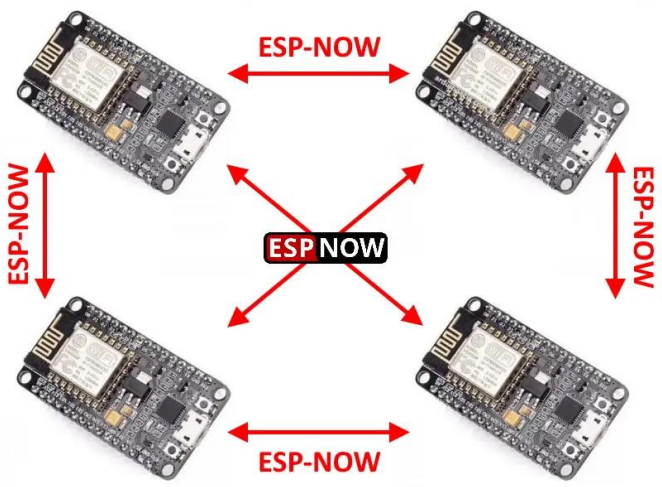
۳ -۲ -۳ -۱ پروتکل ESP-Now

پروتکل ESP-Now یک پروتکل ارتباط بی سیم است که به منظور استفاده در اینترنت‌اشیاء و برای تراشه‌های خانواده‌ی ESP ساخته شده و از بستر Wi-Fi بهره می‌برد. نحوه کار ESP-Now بسیار ساده است و نیازی به گره مرکزی ندارد و تمام دستگاه‌ها مستقیم به یک دیگر متصل می‌شوند؛ ESP-Now از مسیریابی پشتیبانی نمی‌کند و پیام‌ها به طور مستقیم به گره مقصد ارسال می‌شوند که این ویژگی‌های آن، ما را با

[۱] Station

[۲] Access point

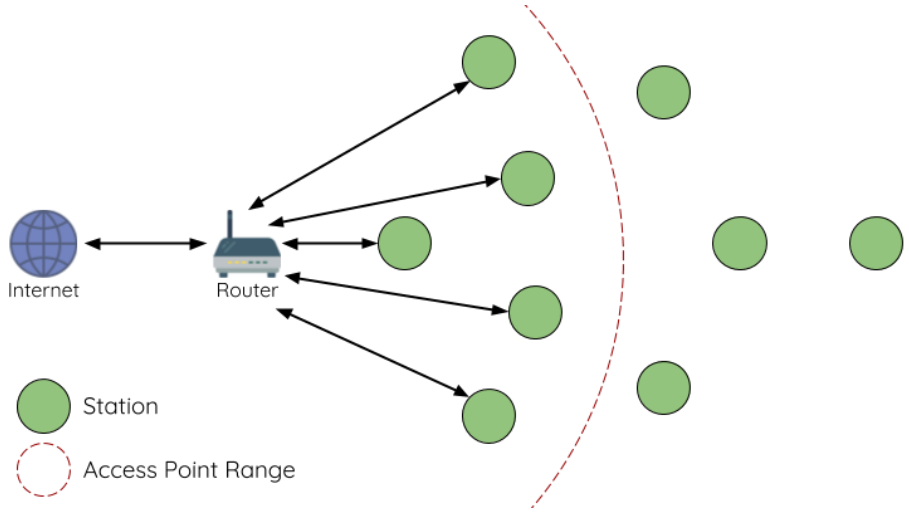
محدودیت‌هایی از قبیل تعداد دستگاه‌های کمتر (زیرا اگر زیاد شوند، تعداد اتصالات به صورت تصاعدی رشد کرده و شبکه خیلی پیچیده‌تر و مصرف انرژی زیاد خواهد شد.) و همچنین پشتیبانی از فاصله فیزیکی کمتر به علت ارتباط مستقیم گره‌ها مواجه خواهد کرد و می‌توان نتیجه گرفت که ESP-Now انتخاب خوبی برای پروژه ما نیست. **تصویر ۱۱** نحوه اتصال ESPها به یک دیگر را نشان داده است.



تصویر ۱۱: پروتکل ESP-Now

۳ -۲ -۳ -۲ شبکه Wi-Fi سنتی (ستاره)

که در این شبکه یک دستگاه در حالت نقطه دسترسی باید باشد و همه‌ی دستگاه‌ها در حالت ایستگاه به آن متصل شوند و پیام‌ها را برای آن ارسال کنند و در این معماری فقط گره‌هایی که در برد دستگاه نقطه دسترسی قرار دارند درون شبکه قرار می‌گیرند (همان‌طور که **تصویر ۱۲** نشان می‌دهد) که باز با محدودیت فاصله فیزیکی که اکثر مواقع در زمین‌های کشاورزی مطرح است، مواجه می‌شویم.



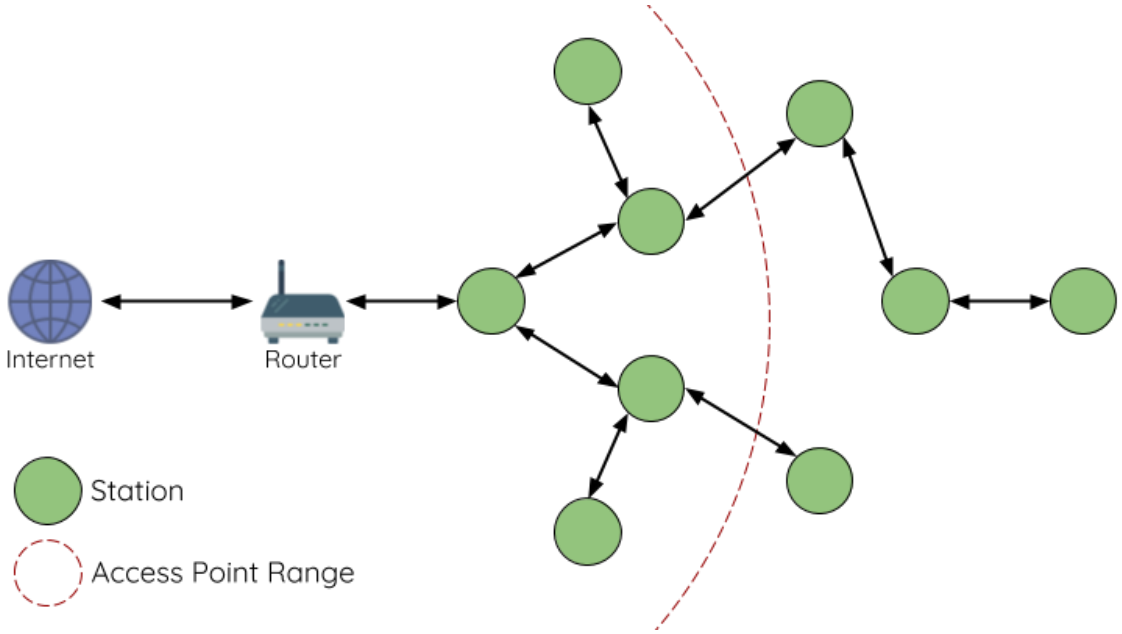
تصویر ۱۲: شبکه سنتی Wi-Fi

۳ -۲ -۳ -۳ پروتکل ESP-Mesh

ESP-Mesh یک پروتکل شبکه است که بر روی زیرساخت Wi-Fi بنا شده است. ESP-Mesh اجازه می‌دهد تا گره‌های مختلف که در یک منطقه فیزیکی بزرگ (هم در داخل و هم در خارج از منزل) پخش شده‌اند، تحت یک WLAN[۱] واحد (شبکه محلی بی سیم) به هم متصل شوند. ESP-MESH خود سازماندهی و خودترمیمی است به این معنی که شبکه می‌تواند به طور مستقل ساخته و نگهداری شود.

در این شبکه همان طور که در **تصویر ۱۳** می‌بینیم، محدودیت فاصله فیزیکی نخواهیم داشت زیرا هر گره می‌تواند به نزدیک‌ترین گره متصل شود و پیامش را از طریق آن انتقال دهد و دیگر نیازی به ارتباط مستقیم تمام گره‌ها با هم یا ارتباط با نقطه دسترسی مرکزی وجود ندارد؛ به واسطه همین ویژگی، محدودیت تعداد دستگاه‌های شبکه نی برداشته خواهد شد و به نظر می‌رسد که این پروتکل دقیقاً همانی است که ما می‌خواستیم و انتظارهایمان را برآورده خواهد کرد.

[۱] Wireless local-area network



تصویر ۱۳: پروتکل ESP-Mesh

۳ -۲ -٤ راهکار ارتباط همزمان گره مرکزی با اینترنت و سایر گره‌ها

با انتخاب پروتکل ESP-Mesh با این چالش روبه‌رو می‌شـــویم که گره مرکزی هم‌زمان باید در دو شـــبکه قرار گیرد، شبکه درونی که با سایر ESPها می‌سازد و شبکه اینترنت که برای این کار باید به روتر[۱] متصل شود تا بتواند اطلاعات حسگرها را برای سرور ارسال کند. برای حل این چالش چند راهکار خواهیم داشت.

۳ -۲ -٤ -۱ ارتباط سریال[۲] دو ESP

می‌توانیم از دو ESP اسـتفاده کنیم که یکی از آن‌ها درون شـبکه داخلی اسـت و یک هم با روتر در ارتباط می‌باشـد و این دو با کابل سـریال به یک دیگر متصـل می‌شـوند و پیام‌ها را تبادل می‌کنند. در حال حاضـر در مواردی دیده شده که از این روش استفاده کرده‌اند و سیستم آن‌ها به خوبی کار می‌کند اما به نظر نمی‌رسد این راهکار اصولی باشد و هزینه‌ی یک ESP اضافه‌تر را هم بر ما تحمیل می‌کند.

---

۱ Router

۲ Serial

۳ -۲ -٤ -۲ استفاده همزمان از حالت ایستگاه و نقطه‌دسترسی

ESP32 می‌تواند هم‌زمان در حالت ایستگاه و نقطه دسترسی باشد در نتیجه می‌توانیم به این ایده برسیم که ESP مرکزی هر دو حالتش هم‌زمان فعال باشد. در حالت نقطه دسترسی به سایر ESPها متصل شود و پیام‌ها را از آن‌ها دریافت کند و با حالت ایستگاه به روتر متصل شود که بتواند از طریق اینترنت با سرور ارتباط برقرار کند. این روش چالش‌های برنامه‌نویسی دارد که با استفاده از کتابخانه‌های آماده‌ای مانند PainlessMesh بخش زیادی از آن‌ها برطرف شده و راحت‌تر خواهد شد ولی همچنان با کمبود مثال‌ها و منابع مواجهیم.

۳ -۲ -۵ ولتاژ متغیر پنل خورشیدی

پس از اتصال ماژول شارژر به پنل خورشیدی انتظار می‌رود که ولتاژ خروجی ماژول با ولتاژ خروجی برابر باشد اما در اکثر مواقع این اتفاق نمی‌افتد و ولتاژ خروجی برابر با ولتاژ پنل خورشیدی می‌شود. برای حل این مشکل می‌توان از پنل خورشیدی با ولتاژ بیشتر (مانند ولتاژ بیشینه ۹ ولت) استفاده کرد که حتی در نورهای کم هم نیاز ما را برطرف کند اما خطر خراب شدن ESP و ماژول شارژ را در حالت بیشینه پنل خورشیدی افزایش خواهد یافت و به نظر نمی‌رسد که راهکار درستی باشد. راهکار دیگر این است که یک افزاینده ولتاژ به ماژول شارژر اضافه کنیم که ولتاژ ثابت پنج ولت بدهد. البته ماژول شارژر a09-AK-10 خودش این کار را برای ما انجام خواهد داد.

٤. فصل چهارم: پیاده سازی سخت افزار

پس از شناخت تجهیزات و چالش‌ها و راه‌حل آن‌ها در فصل‌های گذشته، در این فصل قصد داریم که وارد جزئیات فنی راه اندازی اینترنت‌اشیاء و قسمت سخت افزار آن بشویم. برنامه‌های نوشته شده برای گره مرکزی و راه اندازی بخش‌های مختلف آن و همچنین گره‌های دارای حسگر را در ادامه مشاهده می‌کنیم.

٤ -١ گره مرکزی

گره مرکزی وظیفه دارد اطلاعات حسگرها را از سایر گره‌ها دریافت کند و به سرور اصلی انتقال دهد و همچنین دستورات مشخص شده توسط کاربر و مدل آبیاری را از سرور اصلی دریافت کرده و وضعیت شیرهای آب را در صورت نیاز، تغییر دهد. برای انجام این وظایف، گره مرکزی بخش‌های متفاوتی دارد که در ادامه آن‌ها را بررسی خواهیم کرد.

٤ -١ -١ راه‌اندازی ESP-Mesh و اتصال Wi-Fi

اولین مرحله راه‌اندازی ESP-Mesh می‌باشد که برای انجام این کار از کتابخانه painlessMesh استفاده کرده‌ایم که کار ما را بسیار راحت‌تر خواهد کرد. قبل از همکاری اطمینان حاصل می‌کنیم که Wi-Fi نقطه دسترسی اینترنت و Wi-Fi شبکه Esp-Mesh از کانال‌های یکسانی استفاده می‌کنند. پیشنهاد می‌شود که هر دو از کانال شماره یک استفاده کنند (کانال پیش‌فرض painlessMesh شماره یک می‌باشد اما کانال تنظیم شده Wi-Fi موبایل را باید به صورت دستی روی یک قرار داد).

ابتدا کتابخانه را به پروژه اضافه می‌کنیم و متغیرهای SSID[۱] شبکه Mesh و کلمه عبور[۲] آن و همچنین SSID نقطه دسترسی اینترنت و کلمه عبور آن را مقداردهی می‌کنیم[۹] و یک شیء[۳] از کلاس[۴] painlessMesh با نام mesh ایجاد می‌کنیم. کدهای این بخش را می‌توان در **تصویر ١٤** مشاهده کرد.

```
#include "painlessMesh.h"

#define   MESH_PREFIX     "NewMesh"
#define   MESH_PASSWORD   "Asdfghjkl12"
#define   MESH_PORT       5555

#define   STATION_SSID     "Mohajer"
#define   STATION_PASSWORD "12345678"

#define HOSTNAME "MQTT_Bridge"

painlessMesh  mesh;
```

تصویر ١٤ : مقداردهی اولیه painlessMesh

[۱] Service Set Identifier
[۲] Password
[۳] Object
[٤] Class

در ادامه تابع callback دریافت پیام را به این صورت تعریف می‌کنیم که به هنگام دریافت پیام از سایر گره‌ها (گره‌ها مقادیر حسگرها و نام خود را ارسال می‌کنند.) پیام را توسط سریال[۱] چاپ (برای اشکال‌زدایی[۲] کردن) و با MQTT به سرور ارسال می‌کند(در قسمت بعدی MQTT را توضیح می‌دهیم.). کدهای این بخش را می‌توان در **تصویر ۱۵** مشاهده کرد.

```
void receivedCallback( const uint32_t &from, const String &msg ) {
  Serial.printf("bridge: Received from %u msg=%s\n", from, msg.c_str());
  String topic = "imhere/nodes";
  mqttClient.publish(topic.c_str(), msg.c_str());
}
```

تصویر ۱۵: تابع callback دریافت پیام painlessMesh

پس از تعریف تابع callback دریافت پیام، در قسمت setup برنامه، متغیرهای تعریف شده در ابتدای برنامه و تابع callback را به شیء mesh می‌شناسانیم و راه‌اندازی ساخت شبکه ESP-Mesh و اتصال به اینترنت (توسط `mesh.stationManual()`) را انجام می‌دهیم و سپس به mesh می‌گوییم که این گره، مرکزی است و بعد به کل شبکه اعلام می‌کنیم، شبکه دارای گره مرکزی است. (به این علت که در پروتکل ESP-Mesh می‌توان بدون گره مرکزی هم شبکه را ایجاد کرد، پس باید حتماً به شبکه گفته شود که گره مرکزی وجود دارد.) برای اتمام کار، باید در loop برنامه، مقدار `mesh.update()` را قرار دهیم که شبکه به‌روز نگه داشته شود. کدهای این بخش را می‌توان در **تصویر ۱۶** مشاهده کرد.

---

۱ Serial

۲ Debug

```
void setup() {
  mesh.setDebugMsgTypes( ERROR | STARTUP | CONNECTION );
  mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT, WIFI_AP_STA);
  mesh.onReceive(&receivedCallback);

  //For Internet Connection
  mesh.stationManual(STATION_SSID, STATION_PASSWORD);
  mesh.setHostname(HOSTNAME);


  mesh.setRoot(true);
  // This node and all other nodes should ideally know the mesh contains a root
  mesh.setContainsRoot(true);

}
```

تصویر ۱۶ : بخش painlessMesh setup **گره مرکزی**

٤ -۱ -۲ راه‌اندازی MQTT

برای راه‌اندازی MQTT نیاز است درک درستی از مفاهیم آن مانند topic، publish، subscribe و broker داشت که در فصل دوم کامل توضیحشان دادیم. برای سهولت از کتابخانه‌ی PubSubClient استفاده می‌کنیم و ابتدا کتابخانه را به پروژه اضافه کرده و سپس یک شیء از آن به نام mqttClient می‌سازیم و آدرس دلال MQTT و پورت[۱] مدنظر و تابع callback را به آن می‌دهیم. کد های این بخش را می‌توان در **تصویر ۱۷** مشاهده کرد.

```
#include "PubSubClient.h"

void mqttCallback(char* topic, byte* payload, unsigned int length);

const char *mqtt_broker = "test.mosquitto.org";
WiFiClient wifiClient;
PubSubClient mqttClient(mqtt_broker, 1883, mqttCallback, wifiClient);
```

تصویر ۱۷ : راه اندازی MQTT

[۱] Port

حال نوبت به تعریف تابع callback دریافت پیام MQTT می‌رسد. هر وقت که در topicهای subscribe شده پیامی گذاشته شود، این تابع فراخوانی خواهد شد که در آن ابتدا فضایی در حافظه را به پیام دریافت شده اختصاص می‌دهیم و پیام را داخل آن فضا قرار داده و انتهای پیام بایت null('\0') می‌گذاریم و آن را به String تبدیل می‌کنیم تا کار کردن با آن راحت‌تر شود؛ سپس topic آن را بررسی می‌کنیم، اگر برابر با "state" باشد به معنای این است که باید وضعیت شیرهای آب را تغییر دهیم و این کار را می‌کنیم. کدهای این بخش را می‌توانید در **تصویر ۱۸** مشاهده کنید.

```
void mqttCallback(char* topic, uint8_t* payload, unsigned int length) {
  char* cleanPayload = (char*)malloc(length+1);
  memcpy(cleanPayload, payload, length);
  cleanPayload[length] = '\0';
  String msg = String(cleanPayload);
  free(cleanPayload);

  String targetStr = String(topic).substring(strlen("imhere/"));
  Serial.println(targetStr);
  else if(targetStr == "state")
  {
    deserializeJson(doc, msg);
    digitalWrite(F1, doc["1"]);
    digitalWrite(F2, doc["2"]);
    digitalWrite(F3, doc["3"]);
  }

}
```

تصویر ۱۸ : تابع callback دریافت پیام MQTT

همان‌طور که در **تصویر ۱۹** دیده می‌شود در قسمت setup پروژه تابع Connect را برای شیء mqttClient فراخوانی می‌کنیم تا اتصال برقرار شود و بعد topic مدنظرمان را subscribe می‌کنیم و در آخر درون بخش loop پروژه `mqttClient.loop()` را قرار می‌دهیم و اگر احتمال می‌دهیم که ممکن است اتصال قطع شود، می‌توانیم یک تابع برای اتصال مجدد نیز تعریف کنیم که فرایند اتصال را تکرار کند و هر چند ثانیه یک بار

اتصـــال را چک کند و در صـــورت قطع بودن فراخوانی شـــود که در برنامه ما هر ٦٠ ثانیه یک بار این اتفاق می‌افتد.

```
void setup() {
  if (mqttClient.connect("painlessMeshClient")) {
     mqttClient.publish("imhere","Ready!");
     mqttClient.subscribe("imhere/#");
    }
}
void loop() {
  mqttClient.loop();
  if ( ( millis() >= nexttime ) && ( initialized ) )
  {
    nexttime=millis()+CHECKCONNDELTA*1000;
    if (!mqttClient.connected())
      {reconnect();}
  }
}
```

تصویر ۱۹: بخش loop و setup برای mqtt

٤ -۱ -۳ ساخت WebServer و نمایش نقشه اتصال‌ها

برای مشـــاهـده نحوه اتصـــال گره‌هـا بـه یـک دیگر و گره مرکزی، بـا اســـتفـاده از کتـابخـانـه ESPAsyncWebServer به راحتی یک webserver می‌ســـازیم. ابتدا یک شـــیء به نام server از کلاس AsyncWebServer می‌سازیم و مقدار پورت آن را برابر ۸۰ می‌گذاریم، سپس تابع scanprocessor را تعریف می‌کنیم که در صورت فراخوانی، نحوه اتصالات را از شیء mesh می‌گیرد و خروجی می‌دهد.

```
#include <ESPAsyncWebServer.h>

AsyncWebServer server(80);

String scanprocessor(const String& var)
{
  if(var == "SCAN");
    return mesh.subConnectionJson(false) ;
  return String();
}
```

تصویر ۲۰: شروع کار با WebServer

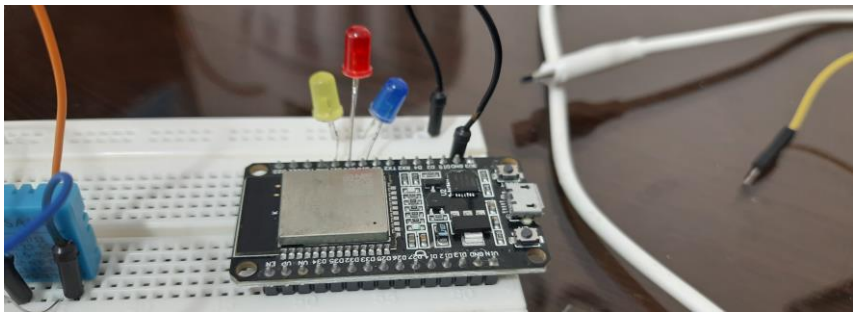
در بخش setup می‌توانیم به سرور بگوییم که اگر کاربر به میسر[۱] خا صی درخوا ست داد، چه چیزی به او نمایش داده شود که در این برنامه می‌خواهیم بگوییم اگر به مسیر map/ درخواست داد؛ تابع scanprocessor را فراخوانی کرده و نتایج آن را با استفاده از یک کد از قبل آماده شده[۲] به صورت گرافیکی[۳] نمایش دهد.کدهای این قسمت در **تصویر ۲۱** قابل مشاهده است.

```
void setup() {
  server.on("/map", HTTP_GET, [](AsyncWebServerRequest *request)
    {
    request->send_P(200, "text/html", "<html><head><script type='text/javascript' src='https://cdnjs.cloudfla
    });
  server.begin();
}
```

تصویر ۲۱: راه اندازی WebServer

٤ -۱ -٤ کنترل شیر آب

به جای شیر آب، از LED استفاده می‌کنیم و مانند **تصویر ۲۲** عمل کرده و آن‌ها را به پین‌های ESP32 متصل کرده و داخل برنامه مانند **تصویر ۲۳** نوع پین‌ها را به عنوان خروجی[۴] تعیین می‌کنیم، در **تصویر ۱۹** دیدیم که هر وقت دستور از سمت سرور ارسال شود؛ در صورت نیاز وضعیت شیرها تغییر می‌کند.



تصویر ۲۲: نحوه اتصال LEDها به گره مرکزی

۱ Path
۲ Template
۳ Graphical
٤ Output

```
#define F1 18
#define F2 19
#define F3 21

void setup() {
  pinMode(F1, OUTPUT);
  pinMode(F2, OUTPUT);
  pinMode(F3, OUTPUT);
}
```

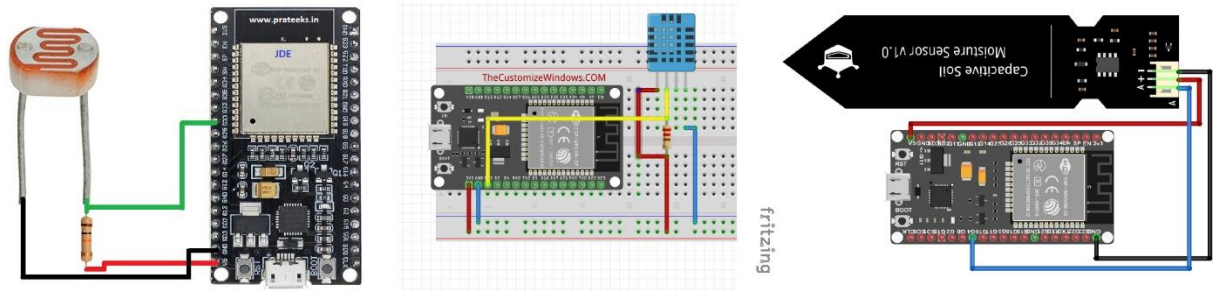
تصویر ۲۳: تعریف پین های LEDها

٤ -۲ سایر گره‌ها

وظیفه اصلی سایر گره‌ها دریافت اطلاعات از حسگرها و ارسال آن‌ها از طریق ESP-Mesh می‌باشد که ابتدا به نحوه وصل کردن حسگرها به گره مرکزی می‌پردازیم و بعد برنامه‌ی خواندن اطلاعات آن‌ها را می‌نویسیم. در آخر هم آن‌ها را به شبکه ESP-Mesh متصل می‌کنیم که اطلاعات را ارسال کنند.

٤ -۲ -۱ راه‌اندازی حسگرها

در این قسمت ابتدا باید سه حسگر DHT-11، LDR و رطوبت خاک را به ESP32 متصل کنیم و بعد کدهای مربوط به آن‌ها را بنویسیم. برای اتصال آن‌ها، مانند **تصویر ۲٤** عمل می‌کنیم (پین دریافت اطلاعات را می‌توانیم متناسب با برنامه‌ای که نوشته‌ایم به پین دیگری از ریزکنترل‌گر متصل کنیم).



تصویر ۲٤: نحوه اتصال حسگرها

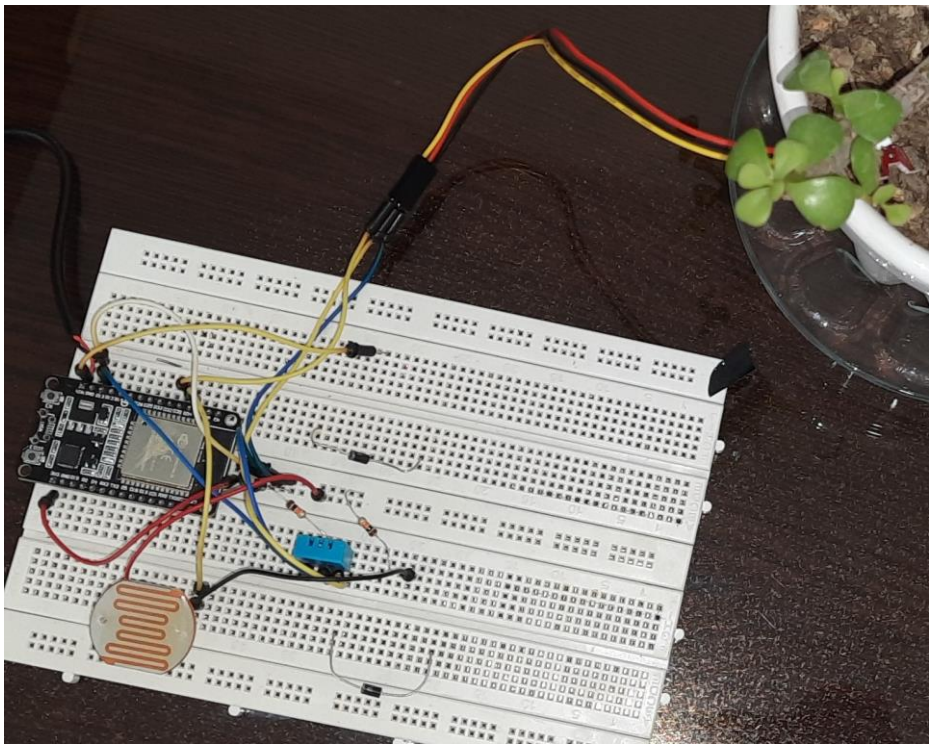
برای DHT-11 از کتابخانه DHT اســتفاده می‌کنیم. کتابخانه را به پروژه اضــافه کرده و یک شــیء از آن می‌سازیم، برای بقیه حسگرها نیازی به کتابخانه خاصی نداریم، حسگر رطوبت خاک را نیاز است که کالیبره[1] کنیم، ابتدا بیرون از آب و خاک، در فضای خشک قرار می‌دهیم و مقدار آن را توسط ESP32 مشاهده می‌کنیم و برابر با کف بازه می‌شود (معمولاً صفر است)؛ سپس درون آب گذاشته و مقدار را به دست می‌آوریم و به عنوان سقف بازه قرارش می‌دهیم و بر اساس آن بازه، درصد رطوبت را محاسبه می‌کنیم. در آخر هم مقادیر حسگرها را با استفاده از تابع construnct_json که تعریف کرده‌ایم می‌خوانیم و داخل JSON[2] قرار می‌دهیم. در **تصویر ۲۵** می‌توانیم این فرایند را ببینیم. **تصویر ۲۶** یکی از گره‌ها می‌باشد که حسگرها به آن متصل‌اند.

```
#include "DHT.h"
DynamicJsonDocument myVar(1024);
#define DHTTYPE DHT11
#define DHTPIN 13  // DHT11 data pin
DHT dht(DHTPIN, DHTTYPE);
float Temperature;
float Humidity;
int sensorVal;
const int LDRPIN = 34;
int _moisture, sensor_analog;
const int MoisturePIN = 35;
void setup() {
  pinMode(DHTPIN, INPUT);
  dht.begin();
  pinMode(LDRPIN, INPUT);
  pinMode(MoisturePIN, INPUT);
}
String construnct_json(){
  Temperature = dht.readTemperature();
  Humidity = dht.readHumidity();
  sensorVal = analogRead(LDRPIN);
  sensor_analog = analogRead(MoisturePIN);
  sensor_analog = sensor_analog > 2600 ? 2600 : sensor_analog;
  _moisture = sensor_analog * 100 / 2600;
  myVar["DHT11"]["Temperature"] = Temperature;
  myVar["DHT11"]["Humidity"] = Humidity;
  myVar["LDR"] = sensorVal;
  myVar["Moisture"] = _moisture;
  String json_string;
  serializeJson(myVar, json_string);
  Serial.println(json_string);
  return json_string;
}
```

تصویر ۲۵: راه‌اندازی حسگرها

[1] Calibrated

[2] JavaScript Object Notation



تصویر ۲٦ : یکی از گره های متصل به حسگرها

٤ -۲ -۲ راه‌اندازی ESP-Mesh

راه‌اندازی پروتکل ESP-Mesh مانند گره مرکزی می‌باشد؛ در چند مورد تفاوت دارند که آن‌ها را خواهیم گفت. ابتدا باید SSID و کلیدواژه را یکسان با گره مرکزی قرار دهیم و به شیء mesh اطلاع دهیم که شبکه گره مرکزی دارد. دیگر نیازی نیست به Wi-Fi متصل شویم و بخش `mesh.stationManual()` حذف خواهد شد.

در این گره‌ها باید یک "وظیفه" زمان‌بندی[۱] ایجاد و به شیء mesh اضافه‌اش کنیم. کار این "وظیفه" زمان‌بندی شده این است که هر ٤٥ ثانیه یک بار تابعی را فراخوانی کند که در آن تابع، construct_json را فراخوانی می‌کنیم تا اطلاعات را از حسگرها بخواند و JSON ساخته شده توسط آن را برای گره مرکزی ارسال می‌کنیم. در **تصویر ۲۷** برنامه این بخش را می‌بینیم.

[۱] ScheduleTask

```
#include <painlessMesh.h>
#include <ArduinoJSON.h>

#define MESH_PREFIX "NewMesh"
#define MESH_PASSWORD "Asdfghjkl12"
#define MESH_PORT 5555

Scheduler userScheduler;
painlessMesh mesh;

void sendMessage();
Task taskSendMessage(TASK_SECOND * 45, TASK_FOREVER, &sendMessage);

void sendMessage() {
  String msg = construnct_json();
  mesh.sendBroadcast(msg);
  taskSendMessage.setInterval(random(TASK_SECOND * 45, TASK_SECOND * 55));
}

void setup() {
  mesh.setDebugMsgTypes(ERROR | STARTUP);
  mesh.init(MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT);
  userScheduler.addTask(taskSendMessage);
  taskSendMessage.enable();
  mesh.setContainsRoot(true);
}

void loop() {
  mesh.update();

}
```

تصویر ۲۷ : برنامه ESP-Mesh گره‌های دارای حسگر

۵. فصل پنجم: راه اندازی سایت مدیریت و سرور

این پروژه نیاز به یک سایت برای مدیریت و نظارت بر زمین کشاورزی و یک سرور برای پردازش دستورات کاربر داریم که در این فصل، بخش‌هایی از آن‌ها را فقط معرفی کرده و کاربردشان را توضیح می‌دهیم و وارد جزئیات فنی پیاده سازی آن‌ها نمی‌شویم، زیرا مربوط به مبحث طراحی سایت می‌شوند و پرداختن به آن‌ها ما را از هدف اصلی این پایان‌نامه، یعنی کاربرد اینترنت اشیا در کشاورزی، دور خواهد کرد. بخشی از پیاده‌سازی سایت و سرور که درباره اتصال به دلال MQTT و دریافت و ارسال اطلاعات است را عمیق‌تر بررسی کرده و توضیح فنی خواهیم داد.

۵ ـ ۱ سایت

سایت شامل تب[۱] های مختلفی دارد که شامل یک صفحه برای نظارت[۲] بر شیر آب و حسگرها، یک صفحه برای کنترل توسط زمان‌بندی و صفحه‌ای برای کنترل با تعیین شرط روی متغیرها و علاوه بر این تب‌ها،

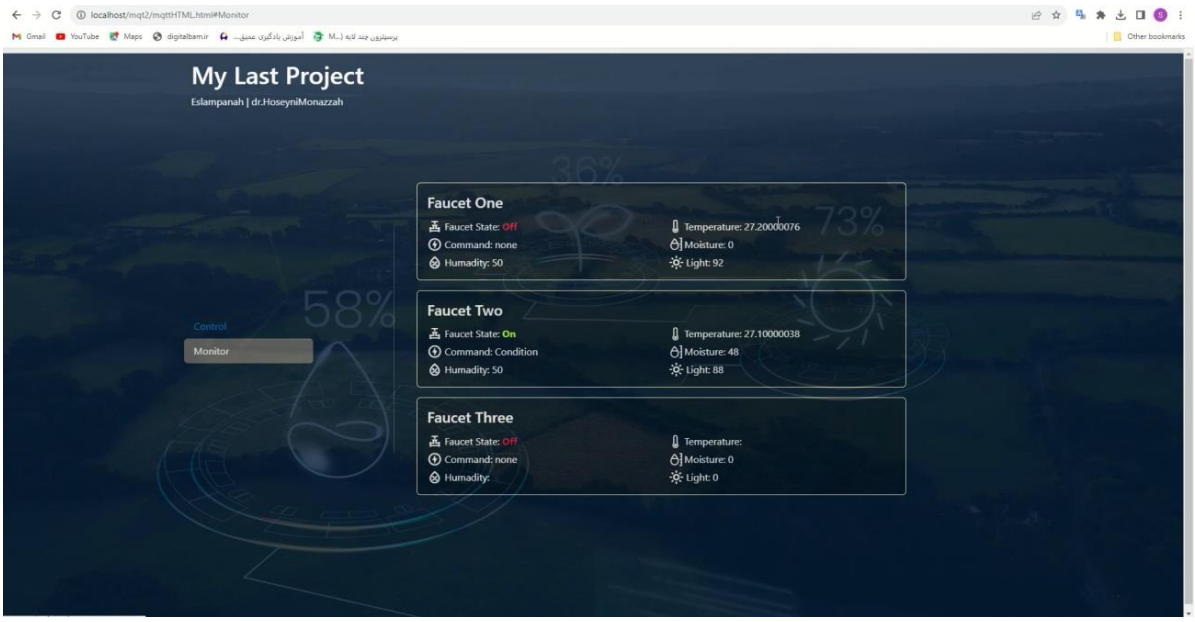
[۱] Tab

[۲] Monitor

کدهای JavaScript برای اتصـــال به MQTT و تبادل اطلاعات دارد که به همه‌ی آن‌ها می‌پردازیم، برای ران اندازی سایت نیاز است که نرم‌افزار Xampp را نصب کرده و فایل‌های سایت را درون پوشه مربوطه قرار دهیم.

۵ -۱ -۱ صفحه نظارت

وقتی روی تب Monitor کلیک می‌کنیم صفحه نظارت باز می‌شود که سه بخش دارد که هر کدام مربوط به یک شیر آب می‌شود. اطلاعات هر بخش شامل شماره‌ی شیر آب، وضعیت روشن یا خاموش بودن آن، دمای هوا، رطوبت هوا، دمای خاک، رطوبت خاک و آخرین دستور تعیین شده برای آن می‌باشد. می‌توانید در **تصویر ۲۸** این صفحه را مشاهده کنید.



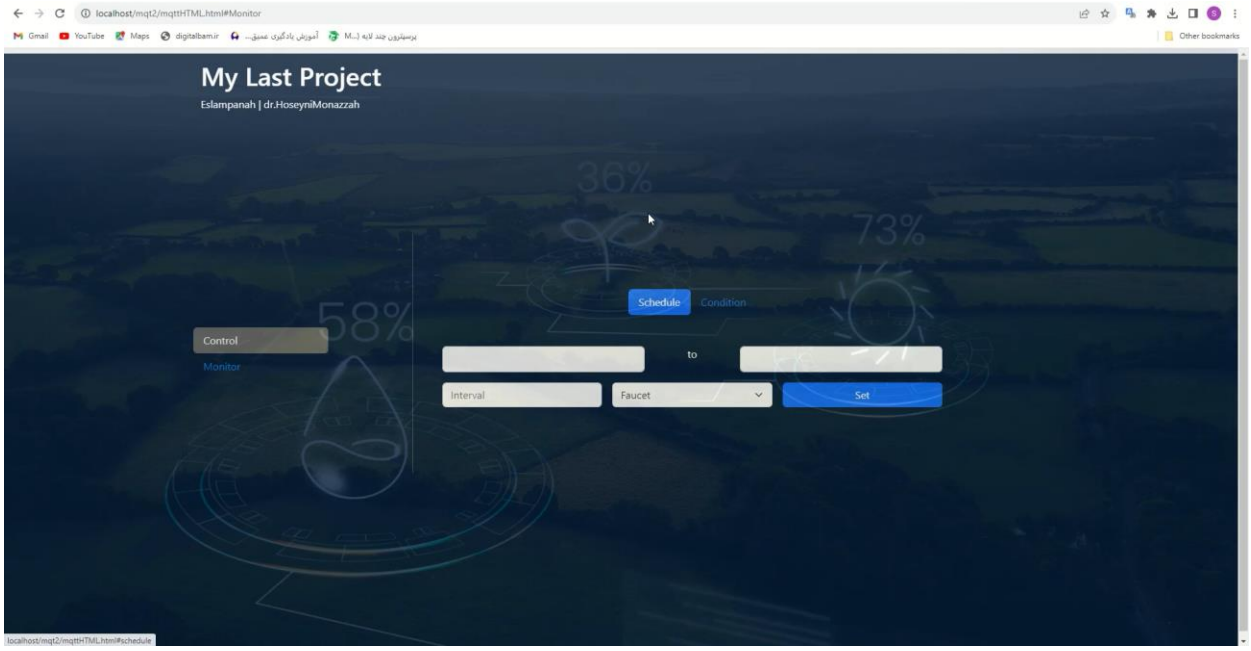
تصویر ۲۸ : صفحه‌ی نظارت (Monitor)

۵ -۱ -۲ صفحه زمان‌بندی

هنگامی که روی تب Control کلیک می‌کنیم، دو انتخاب داریم که یکی از آن‌ها زمان‌بندی می‌باشـــد، در صـــفحه‌ی زمان‌بندی دو جعبه متن[۱] خالی می‌بینیم که با کلیک بر هر کدام از آن‌ها پنجره‌ای برای وارد کردن

[۱] Text Box

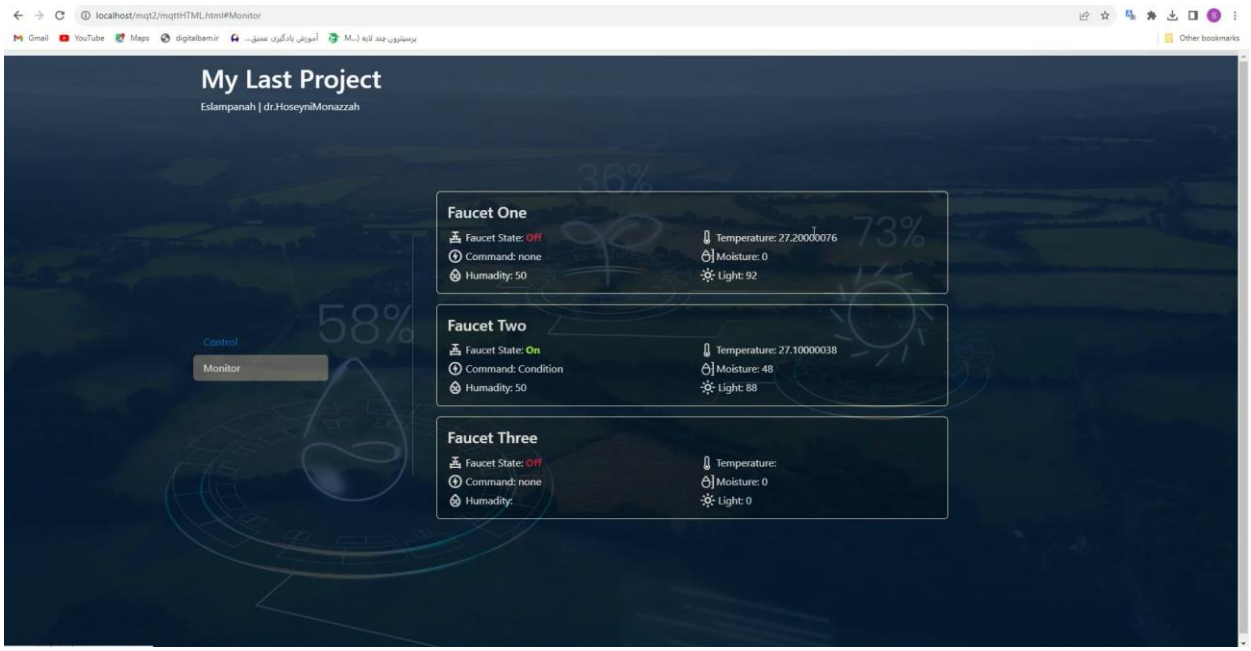
ساعت باز می‌شود و پس از انتخاب بازه، می‌توانیم درون جعبه‌ی Interval تعداد روزهای فاصله میان دو بار آبیاری را مشخص کرده و توسط جعبه‌ی Faucet شیر آب مدنظر را تعیین کنیم. در **تصویر ۲۹** می‌توانیم این صفحه را مشاهده کنیم.



تصویر ۲۹: صفحه‌ی زمان‌بندی

۵ -۱ -۳ صفحه تعیین شرط

بعد از کلیک بر روی تب Control انتخاب دوم ما تعیین شرط است که پس از انتخاب آن، صفحه‌ای باز می‌شود که در آن چهار محل برای تعیین بازه می‌بینیم که به ترتیب مربوط به تعیین بازه برای رطوبت هوا، دما، رطوبت خاک و نور می‌باشد و در زیر آن‌ها بخشی برای انتخاب شیر آب وجود دارد. بعد ضربه بر روی کلید، شرایط تنظیم می‌شود و اگر مقادیر حسگرهای شیر آب مدنظر در بازه تعیین شده قرار گیرند، آبیاری شروع می‌شود. در **تصویر ۳۰** می‌توانید این صفحه را ببینید.



تصویر ۳۰: صفحه‌ی تعیین شرط

۵ -۱ -٤ اتصال با MQTT

سایت ما برای دریافت اطلاعات حسگرها، همچنین ارسال دستورات لازم نیاز است که دلال MQTT متصل شود و ما برای این کار از Paho.MQTT استفاده کرده‌ایم. ابتدا یک شیء آن می‌سازیم و Broker و پورت مدنظر را به آن می‌دهیم، سپس تابعی تعریف می‌کنیم که به هنگام اتصال فراخوانی شود تا به Topicهای مورد نیاز ما Subscribe کند.در **تصویر ۳۱** کدهای این بخش را می‌توانید ببینید.

برای دریافت پیام‌های MQTT تابعی تعریف می‌کنیم و آن را به شیء ساخته شده از Paho.MQTT می شنا سانیم. زمانی که پیامی به topic مدنظر ما بیاید این تابع فراخوانی می شود و بررسی می‌کند، اگر از جانب سرور باشد، وضعیت شیرها را در صفحه نظارت تغییر می‌دهد؛ اگر هم از جانب حسگرها باشد، اطلاعات مربوط به سنسورها را در صفحه نظارت تغییر خواهد داد، **تصویر ۳۲** شامل کدهای این قسمت می‌باشد.

```
window.onload = function () {
  // Create a client instance
  client = new Paho.MQTT.Client("test.mosquitto.org", Number(8081), "/mqtt");
  client.startTrace();
  // set callback handlers
  client.onConnectionLost = onConnectionLost;
  client.onMessageArrived = onMessageArrived;

  // connect the client
  client.connect({
    onSuccess: onConnect
    /*,useSSL: true*/
  });
  console.log("attempting to connect...")
  document.getElementById("formSch").addEventListener('submit', submitForm);



}
function onConnect() {
  // Once a connection has been made, make a subscription and send a message.
  console.log("onConnect");
  client.subscribe("imhere/#");
  message = new Paho.MQTT.Message("Hello");
  message.destinationName = "imhere";
  client.send(message)
  client.send("imhere", "Hello from a better publish call!", 1, false)

  //   topicMessage = new Paho.MQTT.Message("This is a message where the topic is set by setTopic");
  //   topicMessage.topic = "/imhere";
  //   client.send(topicMessage)
}
```

تصویر ۳۱: اتصال سایت به MQTT

```
// called when a message arrives
function onMessageArrived(message) {
  console.log("onMessageArrived:" + message.payloadString);
  switch (message.destinationName) {
    case topic3:
      const obj = JSON.parse(message.payloadString)
      let i = obj["Node Name"]
      document.getElementById("t"+i).innerHTML = obj["DHT11"]["Temperature"];
      document.getElementById("h"+i).innerHTML = obj["DHT11"]["Humidity"];
      document.getElementById("m"+i).innerHTML = obj["Moisture"];
      document.getElementById("l"+i).innerHTML = obj["LDR"];
      break

    case topic4:
      const obj2 = JSON.parse(message.payloadString)
      document.getElementById("fs1").innerHTML = obj2["1"]?"On":"Off"
      document.getElementById("fs2").innerHTML = obj2["2"]?"On":"Off"
      document.getElementById("fs3").innerHTML = obj2["3"]?"On":"Off"

      document.getElementById("fs1").style.color = obj2["1"]?"greenyellow":"crimson"
      document.getElementById("fs2").style.color = obj2["2"]?"greenyellow":"crimson"
      document.getElementById("fs3").style.color = obj2["3"]?"greenyellow":"crimson"

      break;

    default:
      break;
  }
}
```

تصویر ۳۲: دریافت پیام MQTT توسط سایت

۵ ـ ۱ ـ ۵ ارسال اطلاعات

پس این که کاربر زمان‌بندی یا شـــرایط خود را وارد کرد و کلید را فشـــرد، ســـایت باید اطلاعات را روی Topic مربوطه قرار دهد، برای این کار دو تابع جداگانه با نام‌های submitForm و submitCon ایجاد می‌کنیم که اولی برای زمان‌بندی و دومی برای تعیین شـــرط اســـت، با فراخوانی هر کدام، اطلاعات وارد شـــده توســـط کاربر را درون JSON گذاشته و ارسال می‌کند. در **تصویر ۳۳** کدهای این دو تابع را می‌توانید مشاهده کنید.

```
function submitForm(event) {
  try {
    event.preventDefault();
    from = document.forms["formSch"]["from"].value
    to = document.forms["formSch"]["to"].value
    interval = document.forms["formSch"]["interval"].value
    faucet = document.forms["formSch"]["faucet"].value
    command = "sch"
    const obj = { from: from, to: to, interval: interval, faucet: faucet }
    const myJSON = JSON.stringify(obj);
    client.send("imhere/schedule", myJSON, 1, false)
    console.log(myJSON)
    const div = document.getElementById("alertDiv")
    div.innerHTML = `
    <div class="alert alert-success alert-dismissible fade show" role="alert" id="alertSch" >
    Well done!
    <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
     </div>`;
    document.getElementById("c" + faucet).innerHTML = "Schedule"
    //document.getElementById("alertSch").classList.add("show")
  } catch (error) {
    document.getElementById("ErrorH").innerHTML = error;
  }

}
function submitCon(event) {
  hMin = document.getElementsByClassName("range1")[0].value;
  hMax = document.getElementsByClassName("range11")[0].value;
  tMin = document.getElementsByClassName("range2")[0].value;
  tMax = document.getElementsByClassName("range21")[0].value;
  mMin = document.getElementsByClassName("range3")[0].value;
  mMax = document.getElementsByClassName("range31")[0].value;
  lMin = document.getElementsByClassName("range4")[0].value;
  lMax = document.getElementsByClassName("range41")[0].value;
  faucet = document.getElementsByName("faucet2")[0].value
  const obj = { hMin: hMin, hMax: hMax, tMin: tMin, tMax: tMax, mMin: mMin, mMax: mMax, lMin: lMin, lMax: lMax, faucet: faucet }
  const myJSON = JSON.stringify(obj);
  client.send("imhere/condition", myJSON, 1, false)
  console.log(myJSON)
  const div = document.getElementById("alertDiv2")
  div.innerHTML = `
  <div class="alert alert-success alert-dismissible fade show" role="alert" id="alertSch" >
  Well done!
  <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
   </div>`;
  document.getElementById("c" + faucet).innerHTML = "Condition"
  //document.getElementById("alertSch").classList.add("show")
}
```

تصویر ۳۳: ارسال اطلاعات فرم ها توسط سایت

۵ ـ۲ سرور

ما به یک سرور نیاز داریم که اطلاعات را از گره مرکزی و دستورات را از سایت دریافت کند و بررسی کند. هر وقت که شرایط تعیین شده توسط کاربر برقرار شد، به گره مرکزی دستور دهد تا وضعیت شیرهای آب متناسب با آن شرط‌ها را تغییر دهد. ما این سرور را با Node.js نوشته‌ایم و در ادامه بخش‌های مهم آن را بررسی می‌کنیم.

۵ ـ۲ ـ۱ اتصال به MQTT

برای استفاده از پروتکل MQTT از کتابخانه mqtt استفاده می‌کنیم و یک شیء از آن ایجاد کرده و مقادیر اولیه (آدرس Broker، پورت، پروتکل و..) را به تابع connect از آن شیء می‌دهیم و پس از اتصال، Topicهای مورد نیاز را Subscribe می‌کنیم. کدهای مربوط به آن را در **تصویر ۳۴** مشاهده کنید.

```
const protocol = 'ws'
const host = 'test.mosquitto.org'
const port = '8080'
const path = '/mqtt'
const clientId = `mqtt_${Math.random().toString(16).slice(3)}`
const connectUrl = `ws://${host}:${port}${path}`
const client = mqtt.connect(connectUrl, {
  clientId,
  connectTimeout: 4000,
  reconnectPeriod: 1000,
})
const topic1 = 'imhere/condition'
const topic2 = 'imhere/schedule'
const topic3 = 'imhere/nodes'
const topic4 = 'imhere/state'
client.on('connect', () => {
  console.log('Connected')

  client.subscribe([topic1], () => {
    console.log(`Subscribe to topic '${topic1}'`)
  })
  client.subscribe([topic2], () => {
    console.log(`Subscribe to topic '${topic2}'`)
  })
  client.subscribe([topic3], () => {
    console.log(`Subscribe to topic '${topic3}'`)
  })
})
```

تصویر ۳۴: اتصال Node.js به MQTT

۵ -۲ -۲ دریافت دستور

ابتدا بررسی می‌کنیم که پیام دریافت شده، از کدام Topic می‌باشد؛ اگر از "/schedule" یا "/condition" باشد. یعنی پیام دریافتی از طرف سایت ارسال شده است و مدل آبیاری جدیدی را برای یکی از شیرها تعریف کرده است، در نتیجه مقادیر شروط آن‌ها (بازه زمانی و یا بازه‌های که برای حسگرها تعیین شده است) را دریافت کرده و درون متغیرهای مربوط به هر شیر قرار می‌دهد. کدهای این بخش را در **تصویر ۳۵** مشاهده کنید.

```
client.on('message', (topic, payload) => {
  console.log('Received Message:', topic, payload.toString());
  const obj = JSON.parse(payload);
  switch (topic) {
    case topic1: //condition
      faucets[obj["faucet"]]["command"] = "condition";
      faucets[obj["faucet"]]["hMin"] = parseInt(obj["hMin"]);
      faucets[obj["faucet"]]["hMax"] = parseInt(obj["hMax"]);
      faucets[obj["faucet"]]["tMin"] = parseInt(obj["tMin"]);
      faucets[obj["faucet"]]["tMax"] = parseInt(obj["tMax"]);
      faucets[obj["faucet"]]["mMin"] = parseInt(obj["mMin"]);
      faucets[obj["faucet"]]["mMax"] = parseInt(obj["mMax"]);
      faucets[obj["faucet"]]["lMin"] = parseInt(obj["lMin"]);
      faucets[obj["faucet"]]["lMax"] = parseInt(obj["lMax"]);
      console.log(faucets);
      break;

    case topic2: //schedule
      faucets[obj["faucet"]]["command"] = "schedule";
      faucets[obj["faucet"]]["from"] = obj["from"];
      faucets[obj["faucet"]]["to"] = obj["to"];
      faucets[obj["faucet"]]["interval"] = parseInt(obj["interval"]) + 1;
      faucets[obj["faucet"]]["day_counter"] = 0;
      console.log(faucets);
      break;
```

تصویر ۳۵: دریافت دستور توسط سرور

۵ ـ۲ ـ۳ بررسی شروط

در قسمت قبل، اگر پیام دریافتی از موضوع "nodes"/ باشد، به معنای این است که گره مرکزی اطلاعات حسگرها را ارسال کرده است در نتیجه سرور تابعی را فراخوانی می‌کند تا بررسی کند که آیا برای گره‌ای که اطلاعاتش دریافت شده، مدل آبیاری تعیین شده است؟ اگر تعیین نشده باشد که اتفاقی رخ نمی‌دهد، اگر زمان‌بندی تعیین شده است، زمان فعلی را دریافت کرده و با بازه‌ی تعیین شده توسط کاربر مقایسه می‌کند و اگر شرط تعیین شده باشد، مقادیر دریافت شده حسگرها را با شرط تعیین شده مقایسه می‌کند و در نهایت در صورت نیاز، پیام تغییر وضعیت شیرهای آب را ارسال می‌کند تا به دست گره مرکزی و سایت مدیریت برسد. در **تصویر ۳۶** کدهای این بخش را می‌توانید ببینید.

```
function timeCheck(obj) {
  let date_ob = new Date();
  let hm = date_ob.getHours() * 100 + date_ob.getMinutes();
  let publishState = false;
  for (let i of Object.keys(faucets)) {
    if (faucets[i]["command"] == "schedule") {
      if (parseInt(faucets[i]["from"]) <= hm && parseInt(faucets[i]["to"]) > hm) {
        if (faucets[i]["state"] == false && faucets[i]["day_counter"] % faucets[i]["interval"] == 0) {
          faucets[i]["state"] = true;
          faucets[i]["day_counter"] = faucets[i]["day_counter"] + 1;
          publishState = true;
        }
      } else {
        if (faucets[i]["state"] == true) {
          faucets[i]["state"] = false;
          publishState = true;
        }
      }
    }
    if (faucets[i]["command"] == "condition") {
      if (obj["Node Name"] == i) {
        if (faucets[i]["hMin"] < obj["DHT11"]["Humidity"] && faucets[i]["hMax"] >= obj["DHT11"]["Humidity"] &&
          faucets[i]["tMin"] < obj["DHT11"]["Temperature"] && faucets[i]["tMax"] >= obj["DHT11"]["Temperature"] &&
          faucets[i]["mMin"] < obj["Moisture"] && faucets[i]["mMax"] >= obj["Moisture"] &&
          faucets[i]["lMin"] < obj["LDR"] && faucets[i]["lMax"] >= obj["LDR"]) {
          if (faucets[i]["state"] == false) {
            faucets[i]["state"] = true;
            publishState = true;
          }
        } else {
          if (faucets[i]["state"] == true) {
            faucets[i]["state"] = false;
            publishState = true;
          }
        }
      }
    }
  }
  if (publishState) {
    const obj = { "1": faucets["1"]["state"], "2": faucets["2"]["state"], "3": faucets["3"]["state"] };
    const myJSON = JSON.stringify(obj);
    console.log(myJSON);
    pub(topic4, myJSON);
  }
}
```

تصویر ۳۶: بررسی شروط در سرور

٦. فصل ششم: نتیجه‌گیری و کارهای آینده

با توجه به منابع آبی محدود ایران و جهان که در سال‌های اخیر نسبت به قبل کاهش چشم گیری داشته‌اند، نیاز به ایجاد سیستم‌های آبیاری هوشمند برای کاهش هدررفت آب و افزایش بهره‌وری، بیش از قبل حس می‌شود و با توجه به این که در سال‌های اخیر کاربردهای اینترنت‌اشیاء رشد چشمگیری در حوزه‌های مختلف داشته است[۱۰]، در این پروژه سعی بر این داشتیم که روی یک سیستم آبیاری مبتنی بر اینترنت‌اشیاء و شبکه‌های حسگر بی‌سیم کار کنیم و یک نمونه آزمایشگاهی از آن بسازیم؛ طوری که گره‌های آن از برداشتگرهای انرژی استفاده کنند و بیشترین برد مسافتی ممکن را داشته باشند تا بتوانند مساحت بیشتری را پوشش دهند و در اجرا این پروژه مطابق انتظارمان پیش رفتیم و با چالش‌هایی مواجه شدیم که همه‌ی آن‌ها بر طرف شدند.

در آینده باید به سمتی حرکت کنیم که بتوانیم به یک طرح با قابلیت اجرایی واقعی و بازده عینی کشاورزی برسیم. برای انتخاب موضوع این پروژه با تعدادی از اساتید دانشکده کشاورزی دانشگاه تهران مشورت شده است؛ با این حال نیاز است که برای صنعتی شدن آن با تعداد افراد آگاه بیشتری در زمینه کشاورزی مانند کشاورزان، اساتید، محققین، مسئولین ارگان‌های مرتبط تبادل نظر شود تا نیازهای حقیقی، ظرفیت‌های موجود و شرایط فعلی به خوبی شناخته شوند تا به یک طرح کاربردی نزدیک شویم. پس از آن با توجه به این که مسئولین مختلف استان‌ها و مسئولین کشوری بارها اعلام نیاز کرده‌اند، باید با افراد دغدغه‌مند و دارای

اختیارات قانونی ارتباط گرفته شـود و طرح پخته شـده را به آن‌ها ارائه داد تا شـاید بتوانند به اجرایی شـدن آن کمکی کنند. در صورت نا امیدی از این اقدامات دولتی، باید سراغ بخش خصوصی برای سرمایه گذاری رفت و اگر از بخش خصـوصـی هم به نتیجه‌ای نرسـیدیم، در مرحله بعدی باید به سـراغ صـاحبان باغ‌هایی رفت که علاقه‌مند هستند باغ‌های حاشیه شهرشان را از منزل خودشان کنترل کنند و برای آبیاری، مسافت زیادی را تا باغ نپیمایند و نمونه‌های کوچکی از طرح را برای باغ‌های آن‌ها اجرا کرد و پس از مدتی یک شـرکت کوچک راه اندازی کنیم. اگر باغ‌دارها هم تمایلی نشـان ندادند، وقت آن رسـیده که این طرح را فراموش کرده و به سـراغ پروژه دیگری برویم.

Bibliography

[1] Hamami, L., *Application of wireless sensor networks in the field of irrigation.* Computers and Electronics in Agriculture, 2020.

[2] G. Oussama, A.R., F. Tarek, Ahmed S. Alanazi, M. Abid *Fast and Intelligent Irrigation System Based on WSN.* Comput Intell Neurosci, 22.

[3] Gilroy P. Pereira, M.Z.C., Fawwad Daroge, *IoT-Enabled Smart Drip Irrigation System Using ESP32.* IoT, 2023.

[4] Khaled Obaideen, B.A.A.Y., Maryam Nooman AlMallahi, Yong Chai Tan, Montaser Mahmoud, Hadi Jaber, Mohamad Ramadan, *An overview of smart irrigation systems using IoT.* Energy Nexus, 2022.

[5] Haider Mahmood Jawad, R.N., Sadik Kamel Gharghan, Aqeel Mahmood Jawad, Mahamod Ismail, *Energy-Efficient Wireless Sensor Networks for Precision Agriculture.* Sensors, 2017.

[6] R A Atmoko, R.R., M K Hasin, *IoT real time data acquisition using MQTT protocol.* Journal of Physics: Conference Series, 2017.

[7] Anwar Ulla Khan, M.E.K., Mashhood Hasan, Waleed Zakri, Waleed Alhazmi, Tarikul Islam, *An Efficient Wireless Sensor Network Based on the ESP-MESH Protocol for Indoor and Outdoor Air Quality Monitoring.* Sustainability, 2022.

[8] Darko Hercog, T.L., Mitja Truntič, Oto Težak, *Design and Implementation of ESP32-Based IoT Devices.* Sensors, 2023.

[9] davidefa. *ESP32 Wireless Mesh (Made Easy with PainlessMesh).* 2020; Available from: https://www.hackster.io/davidefa/esp32-wireless-mesh-made-easy-with-painlessmesh-part-3-982af1.

[10] W. Marcia, T.M., C. Krishna, K. Sharad, R. Gabriele, R. Heriberto, R. Alexandra, *Worldwide internet of things forecast, 2022–2026.* IDC research, 2022.

واژه‌نامه

الف

ایستگاه..... Station

اینترنت‌اشیاء..... Internet Of Things

اشتراک در ..... Subscribe

اشکال‌زدایی..... Debug

الگو..... Template

انتشار..... Publish

ب

بازه..... Range

برگه..... Tab

پ

پردازنده..... Proccesor

ت

تراشه..... Chip

ج

جعبه متن ..... Text Box

ح

حسگر..... Sensor

خ

خروجی..... Output

د

درگاه ..... Port

دلال..... Broker

ر

ریزکنترل‌گر ..... Microcontroller

س

سکو..... Platform

سیم جامپر(جهنده)..... Jumper wire

ش

شبکه..... Network

شناسه مجموعه خدمات . Service Set Identifier

شیء..... Object

ص

صفحه خورشیدی..... Solar panels

ع

عمومی..... Public

ک

کاربر..... Client

کالیبره.....Calibrated

گ

گذرواژه.....Password

گره..... Node

م

متن‌باز..... Open source

مسیر ..... Path

مقاومت ..... Resistor

مقاومت وابسته به نور. Light Dependent Resistor

موضوع ..... Topic

ن

نظارت ..... Monitor

نقطه دسترسی.....Access point

و

وات ..... Watt

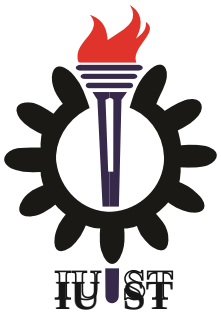
ولتاژ.....Voltage

وظیفه زمان‌بندی شده.....ScheduleTask

Abstract

In this project, we intend to increase the quality of crops, minimize water consumption and reduce the need for manpower by making the irrigation of agricultural lands smarter. For this purpose, we use a Mesh wireless sensor network that has several nodes. The energy of each of its nodes is supplied by the energy harvester, and each node uses sensors for soil moisture, light, temperature, air humidity, etc., which collect information and send it to the central node. In addition to the Mesh network, the central node is connected to another network to communicate with the Internet and sends the information received from other nodes to the server using the MQTT protocol. On the other hand, the user has access to a site through which he monitors the condition of the agricultural land and the values of the sensors; By analyzing this information, it is possible to check the conditions of the land for the cultivation of various types of trees and plants, or by using the irrigation management programs that are currently common, the appropriate model of the land and the desired product can be obtained. Another task of the mentioned site is irrigation management with the help of a processor server. For irrigation management, there is a section on the site where the user has two choices. The first choice is timing. In that, the user selects a time period in the day so that irrigation is done during those hours, it is also possible to determine the number of days between two irrigations, and he can define a different model for each faucet. The second choice is to set the condition that the user must specify an interval for each sensor, that if the current values of the sensors are in that interval, irrigation will start, which can also define a separate model for each valve in this section.

Keywords: Wireless sensor network, Internet of Things, Irrigation, Server, ESP32, Management panel, Agriculture



Iran University of
Science and Technology

Iran University of Science and Technologhy

Department of Computer Engineering

A Thesis Submitted in Partial Fulfillment of the Requirement for the Degree of Bachelor of Science in Computer Engineering

Wireless sensor networks for agriculture and irrigation management

By:

Seyed Mohsen Eslampanah

Supervisor:

Dr. Amir Mahdi Hosseini Monazzah

September 2023