به نام خدا



**گزارش کار مرحله اول پروژه کارشناسی**

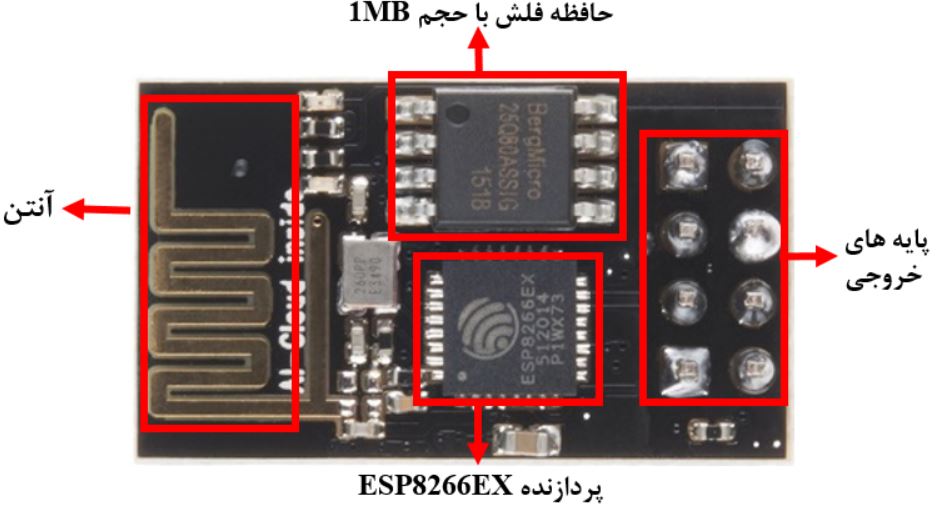
**سید حمید حسینی نژاد**

**صدرا صمدی**

**قسمت اول : معرفی تجهیزات**

**ماژول WiFi**

ESP8266 میکرو کنترلری ساخت شرکت Espressif Systems که به دلیل داشتن قابلیت انجام فعالیت های مربوط به WiFi بیشتر به عنوان ماژول WiFi شناخته شده است. لیکن به خاطر میکروکنترلر بودن آن، می‌توان آن را برای پردازش اطلاعات استفاده کرد.



این ماژول انواع مختلفی دارد که از ESP8266-01 تا ESP8266-12 نام گذاری شده اند. تمام این مدل ها از پردازنده ای یکسان استفاده می‌کنند و تفاوت آن ها در تعداد پایه های در دسترس و شکل ظاهری آن‌هاست.



در این پروژه از مدل ESP8266-01 که دسترسی به آن راحت تر است استفاده می‌کنیم.

این ماژول می‌تواند به سه حالت عمل کند:

1. در نقش Access Point



1. در نقش Station



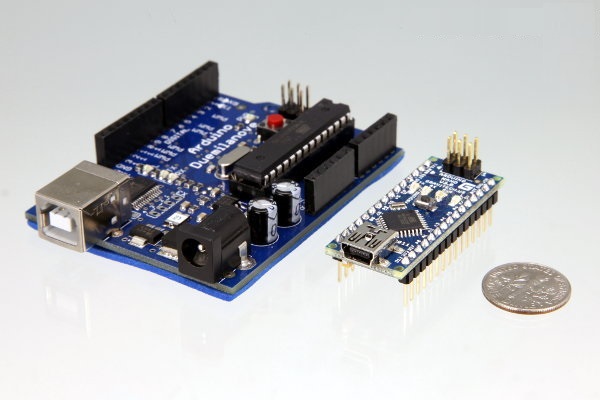
1. حالت ترکیبی Station و Access Point

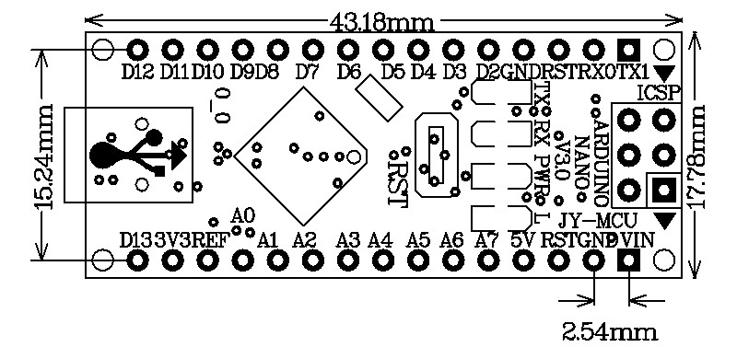


**آردوینو nano**

این برد با میکروکنترلر ATMEGA328P دقیقا مشابه آردوینو UNO کارمی کند . از لحاظ کارایی و تعداد پین ها هیچ تفاوتی با آردوینو UNO ندارد. مزیت اصلی این برد ابعاد بسیار کوچک و قیمت مناسب آن است که به راحتی می توانید از آن به طور مستقیم در پروژه های خود مورد استفاده قرار دهید.

از آردوینو نانو در بسیاری از پروژه ها که محدویت فضا دارند من جمله ربات ها می توان استفاده کرد .





مشخصات

میکروکنترلر: ATmega328

ولتاژ کاری: 5v

ولتاژ ورودی پیشنهادی: 7-12v

محدوده ولتاژ ورودی: 6- 20v

تعداد پینهای I/O دیجیتال: 14

تعداد پینهای آنالوگ ورودی: 8

جریان ورودی برای هر پین: 40mA

حافظه فلش: 32KB

حافظه SRAM: 2KB

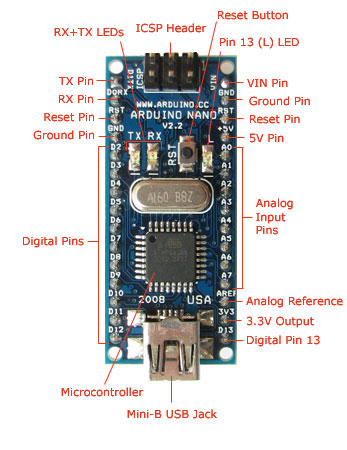
حافظه EEPROM:1KB

فرکانس: 16MHZ

ابعاد: 0.73” \* 1.7”

آردوینو نانو به راحتی با استفاده از کابل MINI USB در محیط نرم افزار آردوینو پروگرام می شود.

ترتیب پایه های آن به صورت زیر است:



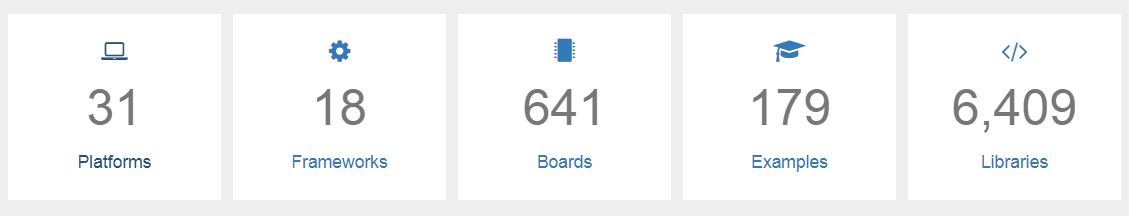
**Plarformio**

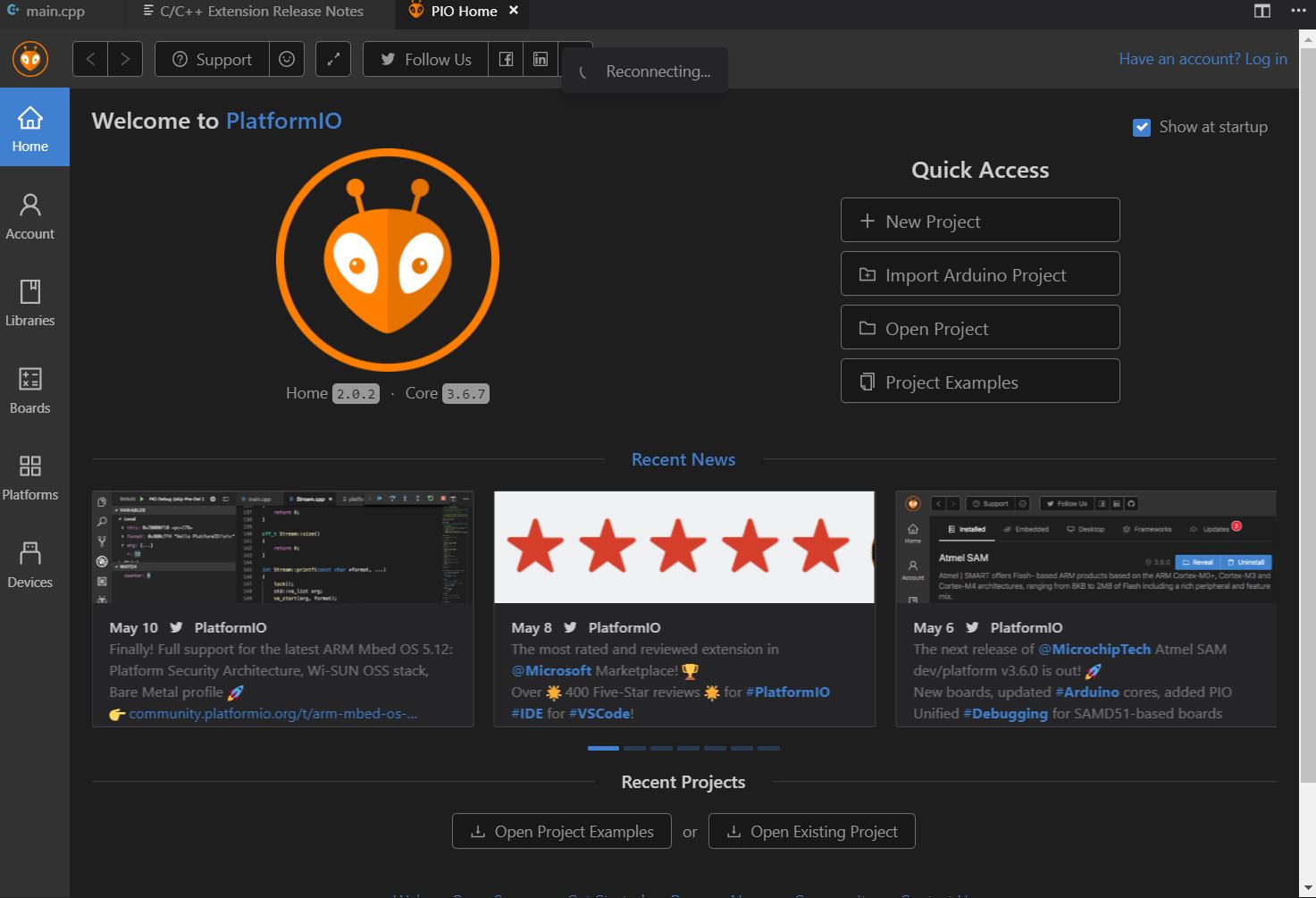
در کنار IDE شرکت آردوینو، می‌توان از محیط های دیگری برای برنامه نویسی آن استفاده کرد.

محیط مورد استفاده در این پروژه، Plarformio است که برای توسعه اینترنت اشیاء مناسب است.

این محیط با فراهم کردن امکانات بهتر برای دیباگ و توسعه و نگه داری کدهای بیشتر، محیط مناسب تری از arduino ide می‌باشد و به صورت افزونه برای اضافه کردن در vscode ، atom و .. وجود دارد.

در این محیط به راحتی می‌توان بورد ، فریم ورک کتابخانه مورد نظر را پیدا و استفاده کرد.





**قسمت دوم: کدها**

**1.قسمت سرور**

سرور این پروژه با استفاده از node js انجام شده. و برای ارتباط با ماژول و همچنین client سرور سوکت ایجاد شده

const http = require('http');

const express = require('express');

const socket\_io = require('socket.io');

const { BehaviorSubject } = require('rxjs');

const port = parseInt(process.env.PORT) || 3303;

const app = express();

const server = http.createServer(app);

const io = socket\_io(server);

const state = new BehaviorSubject('off');

io.on('connection', client => {

console.log('client connected');

let subscription = state.subscribe(s => {

client.emit('state', s);

console.log(`state emitted: ${s}`);

});

client.on('update', s => {

console.log(`updating state: ${s}`);

state.next(s);

});

client.on('disconnect', () => {

subscription.unsubscribe();

console.log('client disconnected');

});

});

server.listen(port, () => console.log(`server started on port ${port}`));

**2.قسمت پنل کنترل کاربری**

با استفاده از کاتلین و socketio.client دو کار انجام می‌گیرد:

1. نظارت (watch) بر وضعیت (state) ازطریق سرور
2. تغییر(update) وضعیت به روشن و خاموش

class MainActivity : AppCompatActivity() {  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 val textView = findViewById<TextView>(R.id.*textView*)  
 val toggleButton = findViewById<ToggleButton>(R.id.*toggleButton*)  
 textView.setText(R.string.*connecting*)  
 toggleButton.*isEnabled* = false  
 val socket = IO.socket("https://iot-switch-server.herokuapp.com")  
 socket.on(Socket.*EVENT\_CONNECT*) **{** runOnUiThread **{** toggleButton.*isEnabled* = true **}** textView.setText(R.string.*connected*)  
 **}** socket.on(Socket.*EVENT\_DISCONNECT*) **{** runOnUiThread **{** toggleButton.*isEnabled* = false **}** textView.setText(R.string.*disconnected*)  
 **}** socket.on("state") **{** args **->** val state = args[0]  
 runOnUiThread **{** toggleButton.*isEnabled* = true  
 if (state == "on")  
 toggleButton.*isChecked* = true  
 else if (state == "off")  
 toggleButton.*isChecked* = false  
 **}  
 }** toggleButton.setOnCheckedChangeListener **{** \_, isChecked **->** toggleButton.*isEnabled* = false  
 if (isChecked)  
 socket.emit("update", "on")  
 else  
 socket.emit("update", "off")  
 **}** socket.connect()  
 }  
  
}

**3.برنامه برد آردوینو**

همانطور که گفته شد، ماژول wifi به صورت جداگانه قابلیت برنامه ریزی را داراست، اما به دلیل اینکه تعداد پورت های ورودی و خروجی آن محدود است ما از برد آردوینو استفاده می کنیم تا فقط دستورات را از ماژول توسط نوع ارتباطی UART گرفته و کنترل خروجی را به برد آردوینو واگذار کنیم.

همچنین برای پروگرم کردن ماژول ما نیاز به یک مبدل UART به USB داریم که برد آردوینو این قابلیت را هم برای ما فراهم می کند.

برای این منظور دو حالت برای آردوینو در نظر گرفته ایم که توسط یه سوییچ کنترل می شود.

حالت اول برد را به یک پل ارتباطی تبدیل می کند که یک سمت آن به پورت USB کامپیوتر و سمت دیگر از طریق UARTبه ماژول متصل میشود. به این صورت ماژول توسط کامپیوتر شناخته و پروگرم می شود.

حالت دوم که پس از اتمام پروگرم کردن استفاده می شود، فقط از طریق UART دستورات را از ماژول wifi گرفته و خروجی مورد نظر را اعمال می کند.

**4.برنامه ماژول WiFi**

برنامه نویسی این بخش به عنوان مهم ترین قسمت پروژه شناخته می شود. زیرا چالش اصلی ما متصل کردن ماژول به یک نقطه دسترسی WiFi و انتقال داده از طریق آن بود. اما به لطف پروژه PlatformIO و کتابخانه های مناسبی که در آن وجود دارد این قسمت با موفقیت به پایان رسید.

برای کار کردن با ماژول اولین مرحله اتصال آن به یک نقطه دسترسی است. بدین منظور ما از کتابخانه WifiManager استفاده کردیم که در هنگام راه اندازی، ماژول را به حالت نقطه دسترسی برده و روی یک IP خاص یک صفحه web برای نمایش نقاط دسترسی موجود و اتصال به یک از آن ها را فراهم می کند.

پس از آن، ماژول از طریق سوکت و کتابخانه SocketIoClient به سرور متصل می شود و سپس منتظر دریافت دستورات از سرور می ماند تا به محض رسیدن، آن ها را از طریق UART به برد آردوینو منتقل کرده تا به خروجی برسند.

