ARDUINO RADAR SCANNER

2025

فصل اول
معرفی میکروکنترلر آردوینو

1.1) مقدمه

با پیشرفت فناوری‌های دیجیتال و توسعه سخت‌افزارهای هوشمند، نیاز به بسترهایی که بتوانند ارتباط میان نرم‌افزار و سخت‌افزار را ساده‌تر کنند بیش از پیش احساس می‌شود. در این میان، آردوینو (Arduino) به‌عنوان یکی از محبوب‌ترین و پرکاربردترین پلتفرم‌های متن‌باز در زمینه طراحی سیستم‌های الکترونیکی و میکروکنترلری، جایگاه ویژه‌ای یافته است. این پلتفرم به دلیل سادگی در یادگیری، هزینه پایین، و پشتیبانی از طیف گسترده‌ای از سنسورها و ماژول‌ها، توانسته است هم در آموزش و هم در پروژه‌های صنعتی و تحقیقاتی مورد استفاده قرار گیرد.

1-2) تاریخچه آردوینو

آردوینو برای اولین بار در سال 2005 توسط گروهی از پژوهشگران در مؤسسه طراحی تعامل Ivrea در ایتالیا معرفی شد. هدف اولیه آن‌ها ایجاد یک ابزار آموزشی ارزان و ساده برای دانشجویانی بود که پیش‌زمینه تخصصی در الکترونیک نداشتند اما می‌خواستند ایده‌های خود را به صورت عملی پیاده‌سازی کنند. نام «Arduino» از یک بار محلی در شهر Ivrea گرفته شد، جایی که بنیان‌گذاران این پروژه گرد هم می‌آمدند. متن‌باز بودن سخت‌افزار و نرم‌افزار آردوینو باعث شد جامعه بزرگی از توسعه‌دهندگان و علاقه‌مندان به سرعت شکل بگیرد و این پلتفرم به یکی از پرطرفدارترین ابزارهای آموزشی و صنعتی تبدیل شود.

1-3) معماری کلی برد آردوینو

هر برد آردوینو شامل اجزای اصلی زیر است:

- میکروکنترلر (Microcontroller): قلب اصلی برد که پردازش داده‌ها و اجرای برنامه‌ها را بر عهده دارد (مانند ATmega328 در Arduino Uno).

- پین‌های ورودی و خروجی دیجیتال (Digital I/O Pins): برای اتصال به سنسورها، کلیدها و عملگرها مانند LED و موتور.

- پین‌های آنالوگ (Analog Input Pins): برای خواندن داده‌های آنالوگ مانند دما یا شدت نور.
- مبدل USB به سریال: برای ارتباط بین برد و رایانه.
- منبع تغذیه: امکان تغذیه برد از طریق کابل USB یا منبع خارجی (مانند باتری یا آداپتور).
- حافظه‌ها: شامل حافظه فلش برای ذخیره برنامه، SRAM برای داده‌های موقت و EEPROM برای داده‌های پایدار.

1–4) محیط توسعه نرم‌افزاری (Arduino IDE)

برای برنامه‌نویسی آردوینو از Arduino IDE استفاده می‌شود که یک محیط ساده و رایگان است. این محیط از زبان برنامه‌نویسی مبتنی بر C/C++ پشتیبانی می‌کند و با کمک کتابخانه‌های آماده، برنامه‌نویسی سنسورها و ماژول‌ها بسیار آسان می‌شود.
ساختار کدنویسی در آردوینو معمولاً شامل دو بخش اصلی است:

1. setup(): بخشی که یک بار در ابتدای اجرای برنامه اجرا می‌شود (برای مقداردهی اولیه).
2. loop(): بخشی که به صورت مداوم اجرا می‌شود (برای اجرای مکرر دستورات).

1–5) ویژگی ها و مزایا

- سادگی و در دسترس بودن: مناسب برای مبتدیان و متخصصان.
- متن‌باز بودن: امکان طراحی و تغییر سخت‌افزار و نرم‌افزار.
- هزینه پایین: نسبت به بسیاری از بردهای توسعه مشابه، ارزان‌تر است.
- جامعه کاربری فعال: وجود هزاران پروژه آماده، کتابخانه و مستندات آموزشی.
- قابلیت گسترش: امکان اتصال شیلدها (Shields) برای افزودن قابلیت‌های جدید مانند وای‌فای، بلوتوث و GPS.

1–6) انواع بردهای آردواینو

برخی از رایج‌ترین مدل‌های آردوینو عبارت‌اند از:

- **Arduino Uno:**

بردهای آردوینو در مدل‌های مختلفی عرضه شده‌اند که هرکدام برای نوع خاصی از پروژه‌ها طراحی شده‌اند. پرکاربردترین مدل آردوینو، Uno است که از میکروکنترلر ATmega328P استفاده می‌کند. این برد با ولتاژ کاری ۵ ولت عمل کرده و دارای ۱۴ پین دیجیتال است که ۶ تای آن‌ها قابلیت PWM دارند. همچنین ۶ ورودی آنالوگ دارد و از طریق پورت USB-B یا منبع تغذیه خارجی بین ۷ تا ۱۲ ولت تأمین انرژی می‌شود. حافظه فلش آن ۳۲ کیلوبایت، SRAM دو کیلوبایت و EEPROM یک کیلوبایت است. به دلیل سادگی، قیمت مناسب و منابع آموزشی بسیار زیاد، این برد بهترین گزینه برای شروع یادگیری و اجرای پروژه‌های پایه محسوب می‌شود.

- **Arduino Mega:**

مدل دیگر آردوینو، Mega 2560 است که نسبت به Uno بسیار قدرتمندتر و بزرگ‌تر است. این برد دارای میکروکنترلر ATmega2560 بوده و با ولتاژ کاری ۵ ولت کار می‌کند. ۵۴ پین دیجیتال در اختیار کاربر قرار می‌دهد که ۱۵ عدد از آن‌ها قابلیت PWM دارند. علاوه بر این، ۱۶ ورودی آنالوگ دارد و از چهار رابط UART پشتیبانی می‌کند. حافظه فلش آن ۲۵۶ کیلوبایت، SRAM هشت کیلوبایت و EEPROM چهار کیلوبایت است. این ویژگی‌ها باعث شده Mega انتخاب مناسبی برای پروژه‌هایی باشد که به تعداد زیادی ورودی و خروجی یا حافظه بیشتر برای ذخیره برنامه نیاز دارند، مانند رباتیک پیشرفته و سیستم‌های کنترل صنعتی.

- **Arduino Nano:**

Nano یکی دیگر از بردهای محبوب آردوینو است که ابعاد کوچکی دارد و برای پروژه‌های قابل‌حمل یا دستگاه‌های پوشیدنی بسیار کاربردی است. این برد همان میکروکنترلر ATmega328 را دارد اما در اندازه‌ای بسیار کوچک‌تر طراحی شده است. Nano با ولتاژ ۵ ولت کار می‌کند، ۱۴ پین دیجیتال (۶ PWM) و ۸ ورودی آنالوگ دارد. حافظه فلش آن ۳۲ کیلوبایت، SRAM دو کیلوبایت و EEPROM یک کیلوبایت است. این برد از طریق پورت Mini-USB متصل می‌شود و به‌راحتی روی بردبورد نصب می‌گردد.

- **Arduino Due:**

یکی از بردهای قدرتمندتر آردوینو Due است. این برد بر پایه میکروکنترلر ARM Cortex-M3 طراحی شده و پردازنده‌ای ۳۲ بیتی با سرعت ۸۴ مگاهرتز دارد. برخلاف مدل‌های قبلی، ولتاژ کاری آن ۳.۳ ولت است. Due دارای ۵۴ پین دیجیتال است که ۱۲ تای آن‌ها قابلیت PWM دارند و همچنین ۱۲ ورودی آنالوگ دارد. حافظه فلش آن ۵۱۲ کیلوبایت و SRAM آن ۹۶ کیلوبایت است. این برد به دلیل قدرت پردازشی بالا، برای پروژه‌های پیچیده مانند پردازش تصویر، تحلیل داده‌های سنگین یا پروژه‌های نیازمند سرعت بالا استفاده می‌شود.

- **Arduino Leonardo:**

Leonardo برد دیگری از خانواده آردوینو است که با میکروکنترلر ATmega32u4 ساخته شده است. این برد دارای ۲۰ پین دیجیتال است که ۷ تای آن PWM هستند و ۱۲ ورودی آنالوگ دارد. حافظه فلش آن ۳۲ کیلوبایت، SRAM حدود ۲.۵ کیلوبایت و EEPROM یک کیلوبایت است. تفاوت اصلی Leonardo با Uno و Nano در این است که می‌تواند مانند یک دستگاه ورودی (مانند صفحه‌کلید یا ماوس) عمل کند و این ویژگی آن را برای پروژه‌های مرتبط با تعامل انسان و رایانه بسیار جذاب می‌سازد.

- **Arduino Micro:**

مدل Micro نیز همانند Leonardo از میکروکنترلر ATmega32u4 بهره می‌برد اما اندازه بسیار کوچک‌تری دارد. این برد دارای ۲۰ پین دیجیتال و ۱۲ ورودی آنالوگ است. حافظه فلش آن ۳۲ کیلوبایت، SRAM حدود ۲.۵ کیلوبایت و EEPROM یک کیلوبایت است. به دلیل ابعاد جمع‌وجور و قابلیت شبیه‌سازی ماوس و کیبورد، انتخابی عالی برای پروژه‌های پوشیدنی و دستگاه‌های کوچک محسوب می‌شود.

- **Arduino LilyPad:**

برد خاص دیگری به نام LilyPad طراحی شده است که بیشتر در پروژه‌های پوشیدنی و لباس‌های هوشمند استفاده می‌شود. این برد شکلی دایره‌ای و باریک دارد تا بتوان آن را به‌سادگی روی پارچه دوخت. از میکروکنترلر ATmega328 استفاده می‌کند و با ولتاژ بین ۲.۷ تا ۵.۵ ولت کار می‌کند. ۱۴ پین دیجیتال دارد که ۶ تای آن PWM هستند و ۶ ورودی آنالوگ نیز در اختیار کاربر قرار می‌دهد. این برد به دلیل طراحی خاص خود، گزینه‌ای مناسب برای لباس‌ها و زیورآلات هوشمند است.

- **Arduino Pro Mini :**

در نهایت، Pro Mini یکی از ساده‌ترین و کوچک‌ترین بردهای آردوینو است. این برد نیز بر پایه ATmega328 ساخته شده و در دو نسخه ۳.۳ ولت و ۵ ولت عرضه می‌شود. دارای ۱۴ پین دیجیتال، ۸ ورودی آنالوگ و ۳۲ کیلوبایت حافظه فلش است. برخلاف مدل‌های دیگر، پورت USB داخلی ندارد و برای پروگرام کردن آن نیاز به مبدل FTDI وجود دارد. مزیت اصلی Pro Mini مصرف انرژی پایین، ابعاد کوچک و قیمت مناسب آن است که آن را برای پروژه‌های کم‌هزینه و مبتنی بر باتری بسیار کاربردی می‌کند.

1-7) کاربردهای آردوینو

- **اینترنت اشیا (IoT):**

آردوینو به دلیل ساختار ساده، انعطاف‌پذیری بالا و پشتیبانی گسترده از سنسورها و ماژول‌ها، کاربردهای بسیار متنوعی در حوزه‌های آموزشی، پژوهشی، صنعتی و حتی زندگی روزمره دارد. یکی از مهم‌ترین زمینه‌های استفاده از آردوینو، اینترنت اشیا است. با استفاده از ماژول‌های ارتباطی مانند Wi-Fi، بلوتوث و GSM می‌توان برد آردوینو را به شبکه متصل کرد و دستگاه‌هایی ساخت که قادر به تبادل داده با یکدیگر یا با سرورهای ابری باشند. نمونه بارز آن خانه‌های هوشمند است که در آن وسایلی نظیر چراغ‌ها، سیستم گرمایش و سرمایش، قفل‌های درب و لوازم خانگی از طریق آردوینو و اینترنت کنترل می‌شوند.

- **رباتیک:**

کاربرد مهم دیگر آردوینو در رباتیک است. این برد می‌تواند به‌عنوان مغز یک ربات عمل کند و وظایفی مانند کنترل موتورهای سروو و DC، پردازش داده‌های سنسورهای فاصله‌سنج، ژیروسکوپ و دوربین، و تصمیم‌گیری در حرکت را بر عهده داشته باشد. پروژه‌های رباتیک از ربات‌های ساده تعقیب‌کننده خط گرفته تا ربات‌های پیچیده چندمفصلی و پهپادها، همگی می‌توانند با آردوینو طراحی و کنترل شوند.

- **پایش محیطی:**

آردوینو در پایش محیطی کاربرد فراوانی دارد. با اتصال سنسورهایی مانند دماسنج، رطوبت‌سنج، سنسور کیفیت هوا، فشارسنج و حسگرهای نوری، می‌توان سیستم‌هایی ساخت که شرایط محیطی را اندازه‌گیری کرده و نتایج را ذخیره یا به‌صورت آنلاین ارسال کنند. این نوع پروژه‌ها در حوزه کشاورزی هوشمند برای پایش دما و رطوبت خاک، در ایستگاه‌های هواشناسی کوچک و در پروژه‌های پایش آلودگی هوا بسیار پرکاربرد هستند.

- **کنترل صنعتی:**

یکی دیگر از حوزه‌های پرکاربرد آردوینو، کنترل صنعتی است. در بسیاری از صنایع سبک و متوسط، نیاز به سیستم‌هایی وجود دارد که فرآیندهای ساده مانند روشن و خاموش شدن موتور، باز و بسته شدن شیرهای برقی یا نظارت بر دما و فشار را کنترل کنند. آردوینو به دلیل هزینه کم و سادگی در برنامه‌ریزی، گزینه‌ای مناسب برای چنین کاربردهایی است. هرچند در صنایع بسیار بزرگ و حساس معمولاً از PLCها استفاده می‌شود، اما برای پروژه‌های کوچک‌تر آردوینو جایگزینی مناسب و مقرون‌به‌صرفه محسوب می‌شود.

- **پروژه‌های آموزشی:**

بسیاری از دانشگاه‌ها و مؤسسات آموزشی از این برد به‌عنوان ابزاری برای آموزش مفاهیم پایه الکترونیک و برنامه‌نویسی استفاده می‌کنند. دانشجویان می‌توانند به‌سادگی مفاهیمی مانند ورودی و خروجی دیجیتال، سیگنال‌های PWM، خواندن داده‌های آنالوگ و ارتباط سریال را با کمک آردوینو درک کنند. علاوه بر دانشگاه‌ها، مدارس و کارگاه‌های آموزشی نیز از آردوینو برای تقویت خلاقیت دانش‌آموزان و آموزش علوم STEM بهره می‌برند.

فصل دوم
انواع ماژول ها در آردواینو

2-1) مقدمه

یکی از ویژگی‌های مهم پلتفرم آردوینو، قابلیت توسعه و گسترش آن با استفاده از ماژول‌ها است. ماژول در واقع یک مدار الکترونیکی آماده و از پیش طراحی شده است که برای انجام یک وظیفه مشخص به کار می‌رود. این وظایف می‌توانند شامل اندازه‌گیری داده‌های محیطی، ارتباط بی‌سیم، ذخیره‌سازی اطلاعات، نمایش داده‌ها یا حتی کنترل قطعات دیگر باشند. استفاده از ماژول‌ها باعث می‌شود کاربر بدون نیاز به طراحی و ساخت مدارهای پیچیده از پایه، بتواند پروژه‌های متنوع و پیشرفته‌ای را با سرعت بیشتری پیاده‌سازی کند.
ماژول‌ها معمولاً دارای ورودی و خروجی‌های مشخصی هستند که امکان اتصال مستقیم آن‌ها به برد آردوینو را فراهم می‌سازند. این ارتباط می‌تواند به صورت دیجیتال، آنالوگ، سریال (UART)، SPI، I2C یا حتی بی‌سیم برقرار شود. به همین دلیل ماژول‌ها انعطاف‌پذیری بالایی در توسعه پروژه‌های مختلف ایجاد می‌کنند و همین امر سبب شده است که جامعه بزرگی از توسعه‌دهندگان و طراحان، مجموعه گسترده‌ای از ماژول‌ها را برای آردوینو تولید و عرضه کنند. از جمله ماژول‌های پرکاربرد که در پروژه‌های آردوینویی به وفور استفاده می‌شوند می‌توان به موارد زیر اشاره کرد:

- ماژول سنسور دما و رطوبت (مانند DHT11 یا DHT22): برای اندازه‌گیری پارامترهای محیطی.
- ماژول RFID: جهت شناسایی و احراز هویت از طریق کارت‌ها و تگ‌های RFID.
- ماژول بلوتوث (مانند HC-05 یا HC-06): برای برقراری ارتباط بی‌سیم با دستگاه‌های دیگر.
- ماژول Wi-Fi (مانند ESP8266 یا ESP32): برای اتصال پروژه به اینترنت و ایجاد قابلیت‌های IoT.
- ماژول GPS: جهت تعیین موقعیت جغرافیایی.
- ماژول رله: برای کنترل وسایل برقی و مدارهای با ولتاژ بالا.
- ماژول نمایشگر (LCD یا OLED): برای نمایش اطلاعات خروجی به کاربر.

در این پروژه نیز برای دستیابی به اهداف مورد نظر، از چند ماژول استفاده شده است. هرچند تمام ماژول‌ها در عملکرد کلی سیستم نقش داشته‌اند، اما دو بخش اصلی بیشترین اهمیت را در طراحی و اجرای پروژه داشته‌اند. این بخش‌ها شامل **سروو موتور** و **ماژول فاصله‌سنج** هستند. سروو موتور وظیفه حرکت مکانیکی دقیق و کنترل‌شده را بر عهده دارد و نقش کلیدی در اجرای عملی فرامین ایفا می‌کند، در حالی که ماژول فاصله‌سنج مسئول اندازه‌گیری فاصله اجسام و دریافت داده‌های محیطی است. ترکیب این دو ماژول امکان ایجاد یک سیستم هوشمند و پویا را فراهم کرده است که می‌تواند به تغییرات محیطی واکنش نشان دهد. در بخش‌های بعدی این فصل، توضیحات کامل‌تری درباره ساختار، ویژگی‌ها، نحوه عملکرد و مزایا و محدودیت‌های این دو ماژول ارائه خواهد شد تا نقش آن‌ها در پروژه به‌طور شفاف مشخص شود.

2-2) سروو موتور (Servo Motor)

سروو موتور یک نوع موتور الکتریکی کنترل‌شده است که برای ایجاد حرکت زاویه‌ای دقیق به کار می‌رود. برخلاف موتورهای DC معمولی که به‌صورت پیوسته می‌چرخند، سروو موتور قادر است محور خود را در یک زاویه مشخص و با دقت بالا تنظیم کند. این ویژگی باعث شده است که سروو موتورها در بسیاری از پروژه‌های رباتیک، مکاترونیک و سیستم‌های کنترل مورد استفاده قرار گیرند.
یکی از نمونه‌های پرکاربرد در پروژه‌های آردوینویی، ماژول Servo SG90 است. این سروو موتور کوچک و سبک‌وزن، توانایی چرخش در محدوده‌ی تقریبی 0 تا 180 درجه را دارد و به دلیل قیمت مناسب، مصرف توان پایین و سهولت در استفاده، محبوبیت زیادی میان علاقه‌مندان به الکترونیک و رباتیک پیدا کرده است.
ساختار داخلی یک سروو موتور از سه بخش اصلی تشکیل شده است:

1. موتور DC کوچک که منبع اصلی حرکت است.
2. گیربکس (Gearbox) که وظیفه کاهش سرعت و افزایش گشتاور موتور را بر عهده دارد.
3. مدار کنترلی و پتانسیومتر داخلی که زاویه‌ی شفت موتور را اندازه‌گیری کرده و آن را با سیگنال ورودی مقایسه می‌کند. این مکانیزم فیدبک باعث می‌شود موتور تنها تا زاویه‌ی مورد نظر حرکت کرده و سپس متوقف شود.

کنترل سروو موتور توسط سیگنال PWM (Pulse Width Modulation) انجام می‌شود. در این روش، عرض پالس دریافتی تعیین‌کننده زاویه‌ی چرخش شفت موتور خواهد بود. به طور معمول، پالس‌هایی با دوره زمانی 20 میلی‌ثانیه به موتور ارسال می‌شوند؛ اگر عرض پالس حدود 1 میلی‌ثانیه باشد، موتور در زاویه صفر درجه قرار می‌گیرد، و اگر عرض پالس به 2 میلی‌ثانیه برسد، موتور به زاویه 180 درجه حرکت می‌کند. مقادیر بین این دو مقدار، زاویه‌های میانی را پوشش می‌دهند.

مزایای سروو موتور SG90:

- دقت بالا در تنظیم موقعیت زاویه‌ای.
- اندازه کوچک و وزن سبک که آن را برای پروژه‌های کوچک مانند رادار یا بازوهای رباتیک مناسب می‌کند.
- مصرف انرژی کم و قابلیت راه‌اندازی مستقیم با برد آردوینو.
- سهولت در برنامه‌نویسی به‌دلیل پشتیبانی کتابخانه Servo در محیط Arduino IDE.

البته سروو موتور SG90 محدودیت‌هایی هم دارد. از جمله اینکه تنها قادر است در بازه‌ی محدود زاویه‌ای (0 تا 180 درجه) حرکت کند و توان مکانیکی بالایی برای بارهای سنگین ندارد. با این حال، برای پروژه‌هایی مانند رادار که نیازمند حرکت زاویه‌ای سبک و دقیق هستند، گزینه‌ای ایده‌آل محسوب می‌شود.

2-3) ماژول فاصله‌سنج (Ultrasonic Sensor HC-SR04)

ماژول HC-SR04 Ultrasonic Sensor یکی از پرکاربردترین ماژول‌های سنجش فاصله در پروژه‌های آردوینویی است. این ماژول بر پایه‌ی امواج فراصوت (Ultrasound) کار می‌کند و می‌تواند فاصله‌ی اجسام تا سنسور را با دقت قابل قبول و سرعت بالا اندازه‌گیری کند. ویژگی اصلی این ماژول، سادگی استفاده، قیمت مناسب و پشتیبانی گسترده در میان جامعه‌ی کاربری آردوینو است.

2-3-1) ساختار و نحوه عملکرد

ماژول HC-SR04 از دو بخش اصلی تشکیل شده است:

1. **فرستنده (Trigger):** یک مبدل فراصوت که امواج صوتی با فرکانس حدود 40 کیلوهرتز را ارسال می‌کند.
2. **گیرنده (Echo):** یک مبدل دیگر که وظیفه دریافت بازتاب امواج از سطح جسم مقابل را بر عهده دارد.

عملکرد ماژول به این صورت است که ابتدا پین Trigger با یک پالس کوتاه (معمولاً 10 میکروثانیه) فعال می‌شود و امواج فراصوت ارسال می‌شوند. این امواج پس از برخورد به مانع بازتاب پیدا کرده و توسط گیرنده (Echo) دریافت می‌شوند. سپس ماژول مدت زمان بین ارسال و دریافت امواج را محاسبه کرده و بر اساس آن فاصله جسم تا سنسور تعیین می‌شود. فرمول کلی محاسبه فاصله به صورت زیر است:

$$\frac{Time \times Speed\,of\,Sound}{2} = Distance$$

در این فرمول، Time مدت زمان رفت و برگشت موج و Speed of Sound سرعت صوت در هوا (تقریباً 340 متر بر ثانیه در دمای اتاق) است.عملیات تقسیم بر 2 به این دلیل است که موج یک بار مسیر رفت و یک بار مسیر برگشت را طی می‌کند.

2-3-2)ویژگی‌ها و مشخصات فنی

- ولتاژ کاری: 5 ولت (مستقیم قابل اتصال به آردوینو).
- جریان مصرفی: حدود 15 میلی‌آمپر.
- برد اندازه‌گیری: از 2 سانتی‌متر تا حدود 4 متر.
- زاویه دید (Measuring Angle): حدود 15 درجه.
- دقت: تقریباً 3 میلی‌متر.

این ویژگی‌ها باعث می‌شوند HC-SR04 برای پروژه‌های مختلفی مانند ربات‌های اجتناب از مانع و سیستم‌های پارک خودرو انتخاب مناسبی باشد.

2-3-3) مزایا

- قیمت پایین و در دسترس بودن.
- سادگی اتصال و برنامه‌نویسی.
- دقت و سرعت اندازه‌گیری مناسب برای پروژه‌های آموزشی و نیمه‌حرفه‌ای.
- امکان یکپارچه‌سازی با سروو موتور برای ساخت سیستم‌های اسکن محیطی (Radar-like).

2-3-4)محدودیت‌ها

- حساسیت بالا به شرایط محیطی مانند دما، رطوبت و سطح اجسام (سطوح نرم مثل پارچه ممکن است امواج را جذب کنند و بازتاب ضعیف باشد).
- محدودیت زاویه دید، که باعث می‌شود تنها اجسامی در محدوده جلوی سنسور شناسایی شوند.
- عدم توانایی در اندازه‌گیری اجسام پشت موانع (چون فقط از بازتاب مستقیم استفاده می‌کند).

با وجود این محدودیت‌ها، ماژول HC-SR04 در کنار سروو موتور SG90 ترکیب فوق‌العاده‌ای برای ساخت یک سیستم رادار ساده اما کارآمد ایجاد می‌کند. سروو موتور امکان حرکت سنسور در زوایای مختلف را فراهم می‌سازد، و سنسور HC-SR04 داده‌های فاصله را جمع‌آوری می‌کند. در نهایت، این داده‌ها توسط آردوینو پردازش شده و می‌توانند به صورت گرافیکی در نرم‌افزارهایی مثل Processing نمایش داده شوند.

2-4) جمع بندی

ماژول‌ها به دلیل آماده بودن، سادگی در اتصال و تنوع گسترده، امکان پیاده‌سازی سریع و دقیق ایده‌های خلاقانه را فراهم می‌سازند. در این فصل پس از معرفی انواع مختلف ماژول‌های پرکاربرد، تمرکز اصلی بر روی دو ماژول کلیدی پروژه حاضر، یعنی سروو موتور SG90 و ماژول فاصله‌سنج HC-SR04 قرار گرفت.

سروو موتور SG90 با قابلیت حرکت زاویه‌ای کنترل‌شده و دقیق، بخش مکانیکی سیستم را تشکیل داده و امکان چرخش سنسور در زوایای مختلف را فراهم می‌کند. از سوی دیگر، ماژول HC-SR04 به‌عنوان قلب اندازه‌گیری پروژه، وظیفه تشخیص فاصله اجسام را بر عهده دارد. ترکیب این دو ماژول در کنار برد آردوینو، باعث شکل‌گیری یک سیستم رادار ساده اما کارآمد می‌شود که می‌تواند محیط اطراف خود را اسکن کرده و داده‌های فاصله را جمع‌آوری کند.

در نهایت، می‌توان گفت که نقش سروو موتور و سنسور فاصله‌سنج در پروژه به‌طور مکمل یکدیگر را کامل می‌کنند؛ یکی مسئول ایجاد حرکت و دیگری مسئول برداشت داده است. همین هماهنگی، پایه‌گذار عملکرد اصلی پروژه بوده و امکان توسعه‌های پیشرفته‌تر مانند پردازش داده‌ها و نمایش گرافیکی نتایج را در مراحل بعدی فراهم می‌سازد.

فصل سوم
پیاده سازی پروژه

3-1) پیاده سازی سخت افزاری

پس از بررسی تئوری ماژول‌ها و نقش هر یک در پروژه، در این فصل به پیاده‌سازی فیزیکی سیستم رادار با استفاده از آردوینو، سروو موتور و ماژول فاصله‌سنج HC-SR04 پرداخته می‌شود. این مرحله شامل اتصال سخت‌افزاری قطعات به برد آردوینو و آماده‌سازی مدار برای اجرای برنامه است.

در ابتدا، ماژول HC-SR04 به‌عنوان سنسور فاصله‌سنج به برد آردوینو متصل شد. این ماژول دارای چهار پایه اصلی است: VCC، GND، Trig و Echo. پایه VCC به ولتاژ 5 ولت آردوینو و پایه GND به زمین (GND) متصل گردید. پایه Trig که وظیفه ارسال پالس فراصوت را بر عهده دارد، به پایه دیجیتال شماره 10 آردوینو متصل شد. پایه Echo نیز که سیگنال بازتاب یافته را دریافت می‌کند، به پایه دیجیتال شماره 11 آردوینو وصل شد. این انتخاب به دلیل در دسترس بودن پایه‌ها و سادگی در مدیریت سیگنال‌های ورودی و خروجی در برنامه انجام گرفت.

در مرحله بعد، سروو موتور SG90 به آردوینو متصل شد. این سروو دارای سه پایه اصلی است: VCC، GND و Signal. پایه VCC آن به خروجی 5 ولت آردوینو و پایه GND به زمین مشترک سیستم وصل گردید. پایه Signal که سیگنال PWM کنترل‌کننده زاویه موتور را دریافت می‌کند، به پایه دیجیتال شماره 12 آردوینو متصل شد. دلیل این انتخاب، سهولت در استفاده از کتابخانه Servo و امکان مدیریت جداگانه حرکت موتور در کنار پردازش داده‌های سنسور بود.

پس از اتصال هر دو ماژول به برد آردوینو، تمامی اتصالات به دقت بررسی شدند تا از درستی مدار اطمینان حاصل شود. همچنین به دلیل اینکه سروو موتور هنگام حرکت ممکن است **جریان بیشتری** نیاز داشته باشد، توجه به تامین توان کافی از طریق منبع تغذیه آردوینو و جلوگیری از نویز الکتریکی اهمیت ویژه‌ای داشت.
در پایان، با اطمینان از صحت اتصالات و اشتراک زمین (GND) بین ماژول‌ها و آردوینو، مدار آماده‌ی بارگذاری کد و اجرای پروژه گردید. بدین ترتیب بستر سخت‌افزاری برای پیاده‌سازی رادار فراهم شد و مراحل بعدی شامل بارگذاری برنامه و آزمایش عملکرد سیستم خواهد بود.

3-2) پیاده سازی نرم افزاری

3-2-1) پیاده سازی در نرم افزار Arduino

برای کنترل سروو موتور و ماژول فاصله‌سنج HC-SR04، نیاز است که یک برنامه مشخص روی برد آردوینو بارگذاری شود. این برنامه در محیط Arduino IDE (Integrated Development Environment) نوشته و اجرا می‌شود. محیط آردوینو IDE بستری ساده و کاربرپسند است که به کاربر اجازه می‌دهد کدهای خود را به زبان C++ نوشته و سپس آن‌ها را روی برد آردوینو بارگذاری کند. این محیط شامل یک ویرایشگر متن برای نوشتن کد، دکمه‌های کامپایل و آپلود، و یک مانیتور سریال برای مشاهده داده‌های ارسالی از برد است.
مزیت اصلی استفاده از Arduino IDE این است که بسیاری از کتابخانه‌های پرکاربرد مانند Servo یا کتابخانه‌های مربوط به ماژول‌های مختلف از پیش در دسترس قرار دارند و تنها با یک دستور ساده قابل استفاده هستند. همچنین IDE آردوینو امکان ارتباط مستقیم با برد از طریق کابل USB را فراهم می‌کند و این امر فرآیند بارگذاری و آزمایش پروژه را ساده‌تر می‌سازد. برنامه پیاده سازی شده در میکروکنترلر به شرح زیر می باشد:

```
1   #include <Servo.h>
2   Servo myServo;
```

در ابتدای برنامه کتابخانه Servo اضافه شده است. این کتابخانه ابزارهای لازم برای کنترل سروو موتور از طریق سیگنال PWM را فراهم می‌کند. سپس شیء myServo از نوع Servo تعریف می‌شود تا بتوانیم از آن برای کنترل حرکت موتور استفاده کنیم.

```
4   const int trigPin = 10;
5   const int echoPin = 11;
6   const int servoPin = 12;
```

در این بخش پایه‌های متصل به ماژول HC-SR04 و سروو موتور مشخص می‌شوند. پایه Trig سنسور به شماره 10، پایه Echo به شماره 11 و پایه سیگنال سروو موتور به شماره 12 آردوینو متصل شده‌اند.

```
8   void setup()
9   {
10    Serial.begin(9600);
11    myServo.attach(servoPin);
12    pinMode(trigPin, OUTPUT);
13    pinMode(echoPin, INPUT);
14  }
```

تابع setup یک بار در ابتدای اجرای برنامه اجرا می‌شود. در این بخش:

- با دستور Serial.begin(9600) ارتباط سریال با نرخ انتقال 9600 بیت بر ثانیه برقرار می‌شود تا داده‌ها در مانیتور سریال قابل مشاهده باشند.
- با دستور myServo.attach(servoPin) سروو موتور به پایه شماره 12 متصل و آماده دریافت فرمان می‌شود.
- سپس وضعیت پایه‌های Trig و Echo مشخص می‌شود؛ پایه Trig به عنوان خروجی و پایه Echo به عنوان ورودی تعریف می‌شود.

```
long readDistance()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  long distance = duration * 0.034 / 2;
  return distance;
}
```

ین تابع وظیفه خواندن فاصله از ماژول HC-SR04 را بر عهده دارد.

1) ابتدا پایه Trig برای چند میکروثانیه خاموش و سپس برای 10 میکروثانیه روشن می‌شود تا پالس فراصوت ارسال گردد.
2) سپس با دستور pulseIn مدت زمان بازگشت پالس از پایه Echo اندازه‌گیری می‌شود.
3) در نهایت با استفاده از فرمول استاندارد (زمان × سرعت صوت ÷ 2)، فاصله جسم تا سنسور محاسبه شده و برگردانده می‌شود.

```
void loop()
{
  for (int angle = 0; angle <= 180; angle += 1)
  {
    myServo.write(angle);
    delay(100);
    long dist = readDistance();

    Serial.print("Angle : ");
    Serial.print(angle);
    Serial.print(",");
    Serial.print("Distance : ");
    Serial.println(dist);

  }
```

در بخش اول تابع loop (که به صورت پیوسته تکرار می‌شود)، سروو موتور از زاویه صفر تا 180 درجه حرکت می‌کند. در هر زاویه:

1) دستور myServo.write(angle) موتور را به زاویه مورد نظر می‌برد.
2) با delay(100) زمان کوتاهی داده می‌شود تا سروو به موقعیت خود برسد.
3) سپس تابع readDistance اجرا شده و فاصله جسم محاسبه می‌شود.
4) داده‌های زاویه و فاصله به صورت سریال چاپ می‌شوند تا در مانیتور سریال یا نرم‌افزارهایی مثل Processing قابل نمایش باشند.

```
  for (int angle = 180; angle >= 0; angle -= 1)
  {
    myServo.write(angle);
    delay(100);
    long dist = readDistance();

    Serial.print("Angle : ");
    Serial.print(angle);
    Serial.print(",");
    Serial.print("Distance : ");
    Serial.println(dist);

  }
}
```

در این قسمت، حرکت معکوس اجرا می‌شود. سروو موتور از زاویه 180 درجه به سمت صفر حرکت می‌کند و همان فرآیند خواندن فاصله و ارسال داده تکرار می‌گردد. این حرکت رفت و برگشتی باعث می‌شود سنسور کل محدوده 180 درجه را اسکن کرده و داده‌های فاصله در هر زاویه ثبت شوند.

در مجموع، این کد ترکیبی از حرکت مکانیکی (توسط سروو موتور) و برداشت داده (توسط ماژول HC-SR04) را پیاده‌سازی می‌کند. خروجی این برنامه مجموعه‌ای از داده‌های زاویه و فاصله است که می‌تواند برای ایجاد یک نمایش گرافیکی شبیه رادار مورد استفاده قرار گیرد.

3-2-2) پیاده سازی در نرم افزار Processing

Processing یک محیط و زبان برنامه‌نویسی ساده و مبتنی بر جاوا است که برای ایجاد گرافیک‌های تعاملی و دیداری (visualization) طراحی شده است. Processing IDE محیطی سبک و سریع برای نوشتن اسکچ‌های گرافیکی دو/سه‌بعدی، دریافت ورودی‌ها، و برقرار کردن ارتباط با سخت‌افزار دارد. کتابخانه‌ی processing.serial امکان خواندن و نوشتن سریال را به‌سادگی فراهم می‌کند، بنابراین ترکیب Processing + Arduino گزینه‌ای رایج برای نمونه‌سازیِ سریع نمایش داده‌های حسگرهاست. ممکن است که این سوال پیش بیاید که چرا در این پروژه چرا انتخاب ما Processing بوده است.

1) **نمایش بلادرنگ:** Processing برای رسم سریع و بلادرنگ (real-time) عالی است زیرا می‌توان اسکن زاویه‌ای و نقاط فاصله را به شکل لحظه‌ای بررسی کرد.
2)**کار با سریال ساده است:** Serial.list()، new Serial(...) و bufferUntil('\n') کار اتصال و دریافت خطوط از آردوینو را راحت می‌کنند.
3) **تبدیل آسان مختصات قطبی → کارتزین:** با توابع cos() و sin() به راحتی می‌توان داده‌های زاویه/فاصله را به مختصات صفحه تبدیل کرد و رادار نیم‌دایره رسم کنیم.
4) **قابل شخصی‌سازی** برای جلوه‌های بصری (پالس، هاله، متن‌ها و پنل هشدار)
5) **مناسب برای هنرمندان و مهندسین**؛ می توان به سرعت پروتوتایپ زد و نیاز به پیاده‌سازی گرافیک پیچیده در پلتفرم دیگری نداریم.

توضیحات کلی بخش Processing به شرح زیر می باشد.

1) کتابخانه ها و متغییر های اصلی :

```
import processing.serial.*;
import java.util.*;

Serial port;
int angle = 0;
int distance = 0;

final int BAUD = 9600;
final int radarRadius = 350;
final int maxDistanceCm = 200;

float smoothAngle = 0;
float smoothDistance = 0;
final float ALPHA = 0.2;

int[] prevScan = new int[181];
int[] currScan = new int[181];

ArrayList<String> warnings = new ArrayList<String>();

```

- .* → جهت ارتباط سریال با آردوینو.
- java.util.* → برای استفاده از لیست‌ها (ArrayList).
- angle , distance → داده‌هایی که از آردوینو دریافت می شود (زاویه و فاصله).
- radarRadius → شعاع رادار روی صفحه نمایش.
- maxDistanceCm → بیشترین فاصله‌ای که نمایش داده می‌شود (۲۰۰ سانتی‌متر).
- smoothAngle , smoothDistance → برای نرم کردن حرکت خط رادار گرافیکی .
- ALPHA → ضریب فیلتر نمایی (smooth).
- prevScan و currScan → ذخیره مقادیر اسکن قبلی و فعلی برای مقایسه تغییرات.
- warnings → لیستی جهت نگه‌داری و دخیره تغییرات هشدارها.

2) تابع Setup()

```
21 void setup() {
22   size(900, 700);
23   smooth(8);
24   println("Available ports: " + join(Serial.list(), " | "));
25 
26   String chosen = pickPort();
27   if (chosen == null) {
28     println(" Connect Arduino Port / There is no Serial port !!!");
29     noLoop();
30     return;
31   }
32 
33   port = new Serial(this, chosen, BAUD);
34   port.bufferUntil('\n');
35 
36   for (int i = 0; i <= 180; i++) {
37     prevScan[i] = -1;
38     currScan[i] = -1;
39   }
40 }
41 
```

- size(900,700) → اندازه پنجره.
- smooth(8) → نرم کردن رسم‌ها.
- Serial.list() → لیست پورت‌های سریال متصل به محیط گرافیکی.
- pickPort() → یک پورت مناسب را انتخاب می‌کند.
- port.bufferUntil('\n') → خواندن کامل داده ها در صورت اضافه شدن خط جدید.
- prevScan و currScan ابتدا مقادیر را برابر با 1- قرار میدهیم به این معنا که هنوز داده ای نداریم.

3) Port Selection

```
String pickPort() {
  String[] ps = Serial.list();
  if (ps.length == 0) return null;
  for (String p : ps) {
    if (p.contains("ACM") || p.contains("USB")) return p;
  }
  return ps[0];
```

در این مرحله تابع PickPort تمامی پورت های ورودی را بررسی می کند. اگر اسم پورت شامل "ACM" یا "USB" باشد، آن را انتخاب میکند. در غیر این صورت اولین پورت موجود را بر می گرداند.

4) رسم صفحه با تابع draw ()

4-1) پس زمینه و متن ها

```
void draw() {
  background(0);

  fill(0, 255, 0);
  textSize(18);
  text("Angle: " + angle + "°", 20, 30);
  text("Distance: " + distance + " cm", 20, 60);
```

به وسیله دستورات بالا مشخص میکنیم که رنگ پس زمینه مشکی باشد و متن و محتوای زاویه و فاصله در صفحه در مختصاتی مشخص قرار بگیرد.

4-2) نرم‌سازی زاویه و فاصله

```
  smoothAngle = ALPHA * angle + (1 - ALPHA) * smoothAngle;
  smoothDistance = ALPHA * distance + (1 - ALPHA) * smoothDistance;
```

زاویه و فاصله با استفاده از فرمول فیلتر نمایی نرم شده تا حرکت‌ها روان‌تر دیده بشوند.

4-3) رسم محدوده رادار

```
  stroke(0, 180, 0, 180);
  noFill();
  for (int r = 50; r <= radarRadius; r += 50) {
    arc(0, 0, r*2, r*2, PI, TWO_PI);
  }
  for (int a = 0; a <= 180; a += 15) {
    float x = cos(radians(a)) * radarRadius;
    float y = -sin(radians(a)) * radarRadius;
    line(0, 0, x, y);
  }
```

نیم دایره ای سبزرنگ با خطوط شعاعی رسم می‌شود.

Scanner Line (4-4

```
pushMatrix();
translate(width/2, height - 20);

stroke(0, 180, 0, 180);
noFill();
for (int r = 50; r <= radarRadius; r += 50) {
  arc(0, 0, r*2, r*2, PI, TWO_PI);
}
for (int a = 0; a <= 180; a += 15) {
  float x = cos(radians(a)) * radarRadius;
  float y = -sin(radians(a)) * radarRadius;
  line(0, 0, x, y);
}
```

در این مرحله مختصات مد نظر را برای خط سبز رنگ متحرک رادار تعریف می کنیم.

4-5) نمایش نقاط (اشیاء شناسایی‌شده)

```
84   for (int a = 0; a <= 180; a++) {
85     int d = currScan[a];
86     if (d > 0 && d <= maxDistanceCm) {
87       float r = map(d, 0, maxDistanceCm, 0, radarRadius);
88       float x = cos(radians(a)) * r;
89       float y = -sin(radians(a)) * r;
90
91       if (d < 20) {
92         drawPulsingBlip(x, y, color(255, 0, 0));
93
94         fill(255);
95         textSize(14);
96         text(a + "° , " + d + " cm", x + 12, y - 12);
97       } else {
98         fill(0, 255, 0);
99         noStroke();
100        ellipse(x, y, 5, 5);
101      }
102    }
103  }
```

در این بخش، کد وظیفه دارد داده‌های فاصله که از حسگر آلتراسونیک دریافت می‌شود را پردازش کرده و بر اساس زاویه فعلی اسکنر، موقعیت جسم شناسایی‌شده را روی صفحه رادار نمایش دهد. ابتدا برای هر زاویه از ۰ تا ۱۸۰ درجه، فاصله متناظر در آرایه‌ی currScan ذخیره شده است. سپس برنامه بررسی می‌کند که آیا این مقدار معتبر است یا خیر (یعنی بزرگ‌تر از صفر و کوچک‌تر یا مساوی با حداکثر فاصله تعریف‌شده).
در صورت معتبر بودن داده، ابتدا مقدار فاصله از واحد سانتی‌متر به مختصات گرافیکی روی صفحه تبدیل می‌شود. این تبدیل از طریق تابع map() انجام می‌گیرد، به این شکل که فاصله‌ی واقعی در بازه‌ی [0 , maxDistanceCm] به یک شعاع در بازه‌ی [0 , radarRadius] تبدیل می‌شود. سپس با استفاده از توابع مثلثاتی cos و sin مختصات دکارتی نقطه‌ی مورد نظر محاسبه می‌شود. این مختصات همان نقطه‌ای است که جسم روی صفحه‌ی رادار باید نمایش داده شود.

برای نمایش نقطه‌ها دو حالت در نظر گرفته شده است:
1) اگر جسمی در فاصله‌ی کمتر از ۲۰ سانتی‌متر شناسایی شود، این نقطه به رنگ قرمز و به صورت چشمک‌زن (pulsing) نمایش داده می‌شود. علاوه بر این، مقدار زاویه و فاصله نیز به صورت متن کنار نقطه درج می‌شود تا کاربر اطلاعات دقیق‌تری از محل جسم داشته باشد. این نمایش ویژه به عنوان هشدار نزدیکی جسم عمل می‌کند.
2) اگر جسم در فاصله‌ی بیش از ۲۰ سانتی‌متر قرار داشته باشد، تنها یک نقطه‌ی سبزرنگ کوچک روی رادار رسم می‌شود که موقعیت جسم را نشان می‌دهد، اما بدون هشدار متنی یا افکت چشمک‌زن.

این بخش از کد در واقع هسته‌ی نمایشی رادار است، زیرا با ترکیب داده‌های زاویه و فاصله، نقشه‌ای از محیط اطراف حسگر ایجاد می‌شود و اجسام شناسایی‌شده به شکل بصری برای کاربر قابل مشاهده می‌گردند. به بیان دیگر، این قسمت همان چیزی است که رادار را از یک ابزار محاسباتی ساده به یک سامانه‌ی دیداری تعاملی تبدیل می‌کند.

4-6) بخش هشدارها

```
108   fill(50, 0, 0, 180);
109   rect(width - 310, 10, 300, 220, 10);
110   fill(255, 0, 0);
111   textSize(16);
112   text("⚠ ALERT CHANGES:", width - 300, 35);
113
114   int yOffset = 60;
115   for (int i = max(0, warnings.size()-10); i < warnings.size(); i++)
116     text(warnings.get(i), width - 300, yOffset);
117     yOffset += 20;
118   }
119 }
```

در این بخش از کد، یک پنل هشدار در سمت راست صفحه رادار رسم می‌شود تا تغییرات قابل توجه در فاصله اجسام نسبت به اسکن قبلی به کاربر اطلاع داده شود. ابتدا با دستورهای rect() و fill() یک مستطیل نیمه‌شفاف قرمز رنگ ایجاد می‌شود که به عنوان پس‌زمینه‌ی لیست هشدارها عمل می‌کند. سپس عنوانی با نام "⚠ ALERT CHANGES:" بالای این بخش نمایش داده می‌شود تا کاربر متوجه شود که اطلاعات مربوط به تغییرات ناگهانی در این قسمت درج می‌شوند.
مقدار هشدارها از لیست warnings خوانده می‌شود؛ این لیست در هنگام دریافت داده‌های سریال (در تابع serialEvent) پر می‌شود. مکانیزم آن به این صورت است که اگر فاصله‌ی جسمی در یک زاویه مشخص نسبت به اسکن قبلی بیش از حد آستانه‌ی مشخص‌شده (۵ سانتی‌متر) تغییر کرده باشد، یک پیام هشدار ساخته و به لیست اضافه می‌شود. پیام هشدار شامل اطلاعاتی مانند زاویه، مقدار فاصله جدید و فاصله قبلی است.
برای نمایش این لیست، تنها ۱۰ هشدار آخر نشان داده می‌شود تا از شلوغ شدن صفحه جلوگیری شود. متن هر هشدار با دستور text() روی مستطیل نمایش داده می‌شود و به ترتیب از بالا به پایین در موقعیت‌های مختلف قرار می‌گیرد.
به طور خلاصه، این بخش همان گزارش متنی لحظه‌ای رادار است که علاوه بر نمایش گرافیکی نقطه‌ها، تغییرات سریع محیط را با جزئیات نشان می‌دهد. این قابلیت باعث می‌شود رادار علاوه بر بُعد تصویری، یک بخش تحلیلی نیز داشته باشد و کاربر بتواند تغییرات محیطی را به شکل متنی دنبال کند.

5) دریافت داده از آردوینو (تابع serialEvent)

```
void serialEvent(Serial p) {
  String line = p.readStringUntil('\n');
  if (line == null) return;
  line = trim(line);
  if (line.length() == 0) return;

  line = line.replace("Angle :", "").replace("Distance :", "");

  String[] toks = splitTokens(line, ",");
  if (toks.length >= 2) {
    try {
      angle = constrain(Integer.parseInt(trim(toks[0])), 0, 180);
      distance = max(0, Integer.parseInt(trim(toks[1])));

      currScan[angle] = distance;


      if (prevScan[angle] != -1 && abs(prevScan[angle] - distance) > 5) {
        String warn = "Angle " + angle + "° → " + distance + " cm (was " + prevScan[angle] + ")";
        warnings.add(warn);
      }


      if (angle == 180) {
        prevScan = currScan.clone();
      }
      if (angle == 0) {
        prevScan = currScan.clone();
      }

    } catch (Exception e) {
      println("Parse failed → " + line);
    }
  }
}
```

تابع serialEvent مسئول دریافت و پردازش داده‌هایی است که از طریق پورت سریال توسط آردوینو ارسال می‌شوند. این داده‌ها شامل دو مقدار اصلی، یعنی زاویه (Angle) و فاصله (Distance) هستند که به‌صورت یک رشته‌ی متنی در هر خط منتقل می‌گردند. روند کار این تابع به شرح زیر است:

1. **خواندن داده از پورت سریال :** با استفاده از دستور readStringUntil('\n') داده‌های ورودی تا رسیدن به کاراکتر پایان خط خوانده می‌شوند. در صورتی که رشته تهی باشد یا ورودی معتبری دریافت نشود، تابع بدون انجام تغییر خاتمه می‌یابد.

2. **پاک‌سازی رشته ورودی :** داده‌ی متنی دریافتی معمولاً شامل کلمات اضافی مانند "Angle : " و "Distance : " است. این بخش از کد با استفاده از تابع replace این مقادیر را حذف می‌کند تا تنها داده‌ی خام (اعداد زاویه و فاصله) باقی بماند.

3. **جداسازی مقادیر زاویه و فاصله:** رشته‌ی اصلاح‌شده با استفاده از splitTokens به دو بخش تقسیم می‌شود. اگر حداقل دو مقدار موجود باشد، هرکدام از آن‌ها به‌عنوان زاویه و فاصله در نظر گرفته می‌شوند.

4.اعتبارسنجی و ذخیره‌سازی داده‌ها: زاویه با استفاده از دستور constrain در بازه‌ی معتبر ۰ تا ۱۸۰ درجه محدود می‌شود. فاصله نیز با تابع max بررسی شده تا از منفی بودن مقدار جلوگیری گردد. مقادیر معتبر سپس در آرایه‌ی currScan ذخیره می‌شوند که بیانگر وضعیت فعلی اسکن رادار است.

5.تشخیص تغییرات ناگهانی: برای مقایسه‌ی مقادیر جدید با اسکن قبلی از آرایه‌ی prevScan استفاده می‌شود. در صورتی که اختلاف فاصله‌ی جدید با مقدار قبلی بیش از ۵ سانتی‌متر باشد، یک هشدار متنی (warning) تولید و به لیست هشدارها اضافه می‌گردد. این قابلیت امکان شناسایی تغییرات غیرعادی یا ورود ناگهانی اجسام به محدوده‌ی دید رادار را فراهم می‌سازد.

6.به‌روزرسانی اسکن قبلی: در پایان هر چرخه‌ی اسکن (هنگامی که زاویه به مقدار ۰ یا ۱۸۰ می‌رسد)، مقادیر currScan به prevScan منتقل می‌شوند تا در مرحله‌ی بعدی امکان مقایسه‌ی داده‌های جدید با داده‌های قبلی وجود داشته باشد.

7.مدیریت خطا: در صورتی که داده‌های ورودی دچار مشکل باشند یا امکان تجزیه‌ی آن‌ها وجود نداشته باشد، پیام خطای "Parse failed" در کنسول چاپ می‌شود.

به‌طور خلاصه، این تابع نه تنها وظیفه‌ی دریافت و ثبت داده‌ها را بر عهده دارد، بلکه با افزودن مکانیزم تشخیص تغییرات غیرمنتظره و تولید هشدار، سیستم رادار را به ابزاری هوشمندتر برای ردیابی اجسام تبدیل می‌کند.

6)نقطه چشمک‌زن (تابع drawPulsingBlip)

```
157 void drawPulsingBlip(float x, float y, color c) {
158   noStroke();
159   float pulse = 8 + sin(millis()/100.0) * 4;
160   fill(c, 200);
161   ellipse(x, y, pulse, pulse);
162   fill(c, 80);
163   ellipse(x, y, pulse*2, pulse*2);
164 }
```

تابع drawPulsingBlip مسئول رسم نقطه‌ی چشمک‌زن است و به‌صورت زیر عمل می‌کند:

- noStrok() → حذف خطوط دور نقطه تا ظاهر آن به‌صورت نرم و دایره‌ای باشد.
- pulse = 8 + sin(millis()/100.0) * 4 → محاسبه‌ی اندازه‌ی متغیر برای افکت چشمک‌زن. تابع sin باعث می‌شود اندازه‌ی نقطه به‌صورت نوسانی تغییر کند و حالت چشمک‌زن ایجاد شود.
- fill(c, 200) → رنگ اصلی نقطه با شفافیت نسبی ۲۰۰ تنظیم می‌شود.
- ellipse(x, y, pulse, pulse) → رسم نقطه‌ی مرکزی با شعاع محاسبه‌شده.
- fill(c, 80) و ellipse(x, y, pulse*2, pulse*2) → رسم یک دایره‌ی کمرنگ بزرگ‌تر اطراف نقطه برای ایجاد افکت درخشندگی و تشدید توجه بصری.

این تابع باعث می‌شود اجسام نزدیک که نیاز به هشدار فوری دارند، به‌صورت چشمک‌زن و برجسته روی رادار نمایش داده شوند. ترکیب نقطه‌ی اصلی و حلقه‌ی کمرنگ، جلوه‌ای بصری ایجاد می‌کند که تشخیص آن توسط کاربر ساده و سریع باشد. به عبارتی، drawPulsingBlip نقش هشدار بصری و فوری را در سیستم رادار ایفا می‌کند.

فصل چهارم
جمع بندی و نتیجه گیری

در این پروژه، یک سیستم رادار مبتنی بر آردوینو و سنسور اولتراسونیک همراه با سروو موتور طراحی و پیاده‌سازی شد. هدف اصلی پروژه، ایجاد محیطی تعاملی برای اسکن و نمایش موقعیت اجسام در اطراف رادار بود.

در بخش سخت‌افزاری، استفاده از آردوینو Uno به‌عنوان برد اصلی پروژه امکان کنترل دقیق سروو موتور و دریافت داده‌های سنسور فاصله را فراهم کرد. سنسور اولتراسونیک HC-SR04 برای اندازه‌گیری فاصله با دقت مناسب و سروو موتور برای حرکت اسکن زاویه‌ای انتخاب شدند. اتصال دقیق پایه‌ها و برنامه‌نویسی آردوینو باعث شد داده‌های واقعی محیط به صورت پیوسته جمع‌آوری و ارسال شوند.

در بخش نرم‌افزاری، محیط Processing برای دریافت داده‌های سریال، پردازش و نمایش گرافیکی انتخاب شد. داده‌های زاویه و فاصله به شکل نقاط روی صفحه رادار نمایش داده شدند و با افزودن قابلیت‌هایی مانند نقاط چشمک‌زن برای اجسام نزدیک و لیست هشدار برای تغییرات ناگهانی، سیستم نه تنها توانست محیط را شناسایی کند، بلکه سطح ایمنی و قابلیت تصمیم‌گیری کاربر را نیز افزایش داد.

با اجرای موفق پروژه، نتایج زیر حاصل شد:

1. ترکیب موفق سخت‌افزار و نرم‌افزار: داده‌های سنسور به‌طور بلادرنگ دریافت، پردازش و بصری‌سازی شدند.
2. ایجاد نمای گرافیکی دقیق: نمایش زاویه و فاصله اجسام در محدوده اسکن رادار امکان تحلیل محیط را به کاربر داد.
3. هشدارهای بصری و متنی: نقاط چشمک‌زن و لیست هشدار تغییرات ناگهانی، ابزارهایی مفید برای تشخیص سریع اجسام نزدیک و تغییرات محیطی ارائه کردند.
4. پایه‌ای برای توسعه‌های آینده: پروژه می‌تواند با افزودن سنسورهای دقیق‌تر، فیلترهای نرم‌افزاری برای کاهش نویز و قابلیت ذخیره‌سازی یا ارسال بی‌سیم داده‌ها ارتقا یابد.

نتیجه‌گیری نهایی:

این پروژه نشان داد که ترکیب آردوینو، سنسورهای ماژولار و محیط Processing می‌تواند یک سیستم رادار آموزشی و کاربردی بسازد. علاوه بر ایجاد تجربه عملی در کار با سخت‌افزار و نرم‌افزار، پروژه مهارت‌های طراحی، برنامه‌نویسی، پردازش داده و تجسم گرافیکی را نیز توسعه داد. بنابراین، سیستم طراحی‌شده نمونه‌ای موفق از یک رادار ساده اما هوشمند و قابل توسعه محسوب می‌شود که می‌تواند پایه‌ای برای پروژه‌های پیشرفته‌تر در حوزه رباتیک، امنیت و اینترنت اشیاء باشد.

پایان