بسم الله الرحمن الرحيم

دانشگاه دامغان

دانشگاه دامغان
دانشکده فنی و مهندسی

پایان نامه برای دریافت درجه کارشناسی
مهندسی برق

ایستگاه هواشناسی دیجیتال

دانشجویان

مهدی جمشیدی
محمدابراهیم ابراهیم‌طوسی

استاد راهنما

دکتر بهزاد بقراطی

۱۳۹۹

بسمه تعالی

«فرم تأیید پایان نامه توسط استاد راهنما و مسئول پروژه های کارشناسی»

دانشکده فنی مهندسی

معاون محترم آموزشی دانشکده فنی مهندسی دانشگاه دامغان

با سلام و احترام،

بدینوسیله اعلام می دارم نسخه پایان نامه کارشناسی آقای / خانم ..... به شماره دانشجویی ..... مورد تأیید اینجانب می باشد و نسخه مذکور نیاز به هیچگونه اصلاحات نگارشی و متنی ندارد.

نام و نام خانوادگی استاد راهنما **مسئول پروژه های کارشناسی**

امضاء و تاریخ: **امضاء و تاریخ:**

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب ..... به شـــماره دانشـــجویی ..... دانشـــجوی رشـــته ..... مقطع تحصـــیلی..... تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه/رســـاله حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری‌شده از آثار دیگران را با ذکر کامل مشـخصـات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصـــوص احقاق حقوق مکتســـب و تشـــخیص و تعیین تخلف و مجازات را از خویش ســـلب می‌نمایم. در ضـمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی‌صلاح (اعم از اداری و قضـایی) به عهده‌ی اینجانب خواهد بود و دانشـگاه هیچ‌گونه مسـؤولیتی در این خصـوص نخواهد داشت.

نام و نام خانوادگی:

امضا و تاریخ:

مجوز بهره‌برداری از پایان‌نامه

بهره‌برداری از این پـایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توســـط اســـتاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

☐ بهره‌برداری از این پایان‌نامه/ رساله برای همگان بلامانع است.

☐ بهره‌برداری از این پایان‌نامه/ رساله با اخذ مجوز از استاد راهنما، بلامانع است.

☐ بهره‌برداری از این پایان‌نامه/ رساله تا تاریخ ..... ممنوع است.

نام استاد راهنما:

تاریخ:

امضا:

چکیده

محور اصلی پروژه، طراحی و پیاده‌سازی ایستگاه هواشناسی سینوپتیک (Synoptic) متشکل از یک دستگاه که در فاصله چند کیلومتری از مرکز اصلی قرار داده می‌شود تا پارامترهای جوی نظیر دمای هوا، فشار، رطوبت، شدت نور، سرعت و جهت باد را اندازه‌گیری کرده و داده‌ها را از طریق امواج رادیویی به دستگاه دیگر که در پایگاه قرار دارد جهت ثبت و نمایش روی رایانه‌های پایگاه مخابره کند، می‌باشد. در این پروژه از میکروکنترلر STM32 جهت برنامه‌ریزی و ایجاد ارتباط میان بخش‌های مختلف و همچنین از گیرنده و فرستنده‌های آلتراسونیک جهت محاسبه سرعت و جهت باد استفاده‌شده است. به‌طورکلی با استفاده از سنسورهای دیجیتال در این پروژه علاوه بر کم‌تر شدن حجم دستگاه‌ها مصرف انرژی آن نیز نسبت به ایستگاه‌های سنتی که تجهیزاتی اغلب مکانیکی داشتند کمتر شده است. درنهایت نیز استفاده از ماژول لورا (LoRa) به‌عنوان بهینه‌ترین و کم‌هزینه‌ترین روش ارتباط راه گشای انتخاب روش ارسال داده‌های رادیویی خواهدبود.

واژه‌های کلیدی: STM32، LoRa، Ultrasonic، Python

فهرست مطالب

۱ مقدمه ۱

۱.۱ مقدمه ۲

۲ آشنایی با میکروکنترلرهای ARM و نرم‌افزار STM32CubeIDE ۴

۱.۲ مقدمه ۵

۲.۲ معماری ARM ۶

۳.۲ میکروکنترلرهای شرکت ST ۶

۱.۳.۲ مقدمه ۶

۲.۳.۲ نحوه نام‌گذاری ۷

۳.۳.۲ انواع کتابخانه‌های مورد استفاده در میکروکنترلرهای ST ۹

۴.۳.۲ نرم‌افزار STM32CubeIDE ۱۰

۴.۲ میکروکنترلر ۱۱

۳ آشنایی با LoRa ۱۲

۱.۳ مقدمه ۱۳

۲.۳ جنبش جهانی شبکه اشیاء (TTN:The Things Network) ۱۴

۳.۳ مقایسه ماژول LoRa و NB-IOT ۱۶

۴.۳ معماری LoRa ۱۸

۵.۳ انواع پیاده‌سازی ۲۴

۱.۵.۳ پیاده‌سازی LoRa ۲۴
۲.۵.۳ LoRaWAN ۲۵
۳.۵.۳ انواع کلاس در شبکه LoRaWAN ۲۸
۶.۳ امنیت ۲۹
۷.۳ ماژول RN2483 . . . . . ۲۹

۴ زبان برنامه‌نویسی پایتون ۳۱
۱.۴ مقدمه ۳۲
۲.۴ تاریخچه ۳۴
۳.۴ ویژگی‌ها ۳۴
۴.۴ مزایا ۳۵
۵.۴ معایب ۳۶

۵ آشنایی با Git و GitHub ۳۷
۱.۵ مقدمه ۳۸
۲.۵ آموزش کار با گیت ۳۹
۱.۲.۵ نیازهای بنیادی ۳۹
۲.۲.۵ راه‌اندازی یک مخزن (Repository) ۴۰
۳.۲.۵ کلون یا کپی کردن یک مخزن ۴۰
۴.۲.۵ تنظیمات ۴۱
۵.۲.۵ اعمال اولین تغییر ۴۱
۶.۲.۵ مشاهده‌ی تغییرات ۴۲
۷.۲.۵ نکات مهم درباره‌ی استیجینگ یا ایندکس ۴۳
۳.۵ آموزش کار با گیت‌هاب ۴۴
۱.۳.۵ ساخت مخزن یا Repository ۴۴
۲.۳.۵ ساخت شاخه یا Branch ۴۵

۳.۳.۵ ایجاد تغییرات و اعمال آن‌ها ۴۵
۴.۳.۵ ایجاد درخواست اعمال تغییرات یا pull request ۴۵
۵.۳.۵ پذیرش درخواست تلفیق تغییرات اعمال‌شده ۴۶
۴.۵ آموزش انتقال پروژه از گیت به گیت‌هاب ۴۶

۶ پروتکل‌های ارتباطی ۴۸
۱.۶ مقدمه ۴۹
۲.۶ پروتکل SPI ۵۰
۳.۶ پروتکل I^2C ۵۱
۴.۶ پروتکل USB ۵۲

۷ سنسورها ۵۵
۱.۷ مقدمه ۵۶
۲.۷ فشارسنج ۵۶
۳.۷ شدت نور ۵۷
۴.۷ قطب‌نما ۵۸
۵.۷ دما و رطوبت هوا ۵۹
۶.۷ شدت و جهت باد ۶۰

۸ طراحی بردمدارچاپی ۶۵
۱.۸ مقدمه ۶۶
۲.۸ ملاحظات طراحی PCB ۶۷
۱.۲.۸ خطوط انتقال ۶۹
۲.۲.۸ اندازه گیری بازتابش سیگنالها ۶۹
۳.۲.۸ ساختار برد های مدار چاپی ۷۱
۴.۲.۸ صفحه مرجع ۷۲

۵.۲.۸ برد های چند لایه ۷۲
۶.۲.۸ روند طراحی بردهای فرکانس بالا ۷۳
۷.۲.۸ انتخاب المانها و طراحی شماتیک ۷۳
۸.۲.۸ چینش المانها و مسیر کشی آنها ۷۳
۹.۲.۸ شبیه سازی پس از چینش المان‌ها ۷۴
۱۰.۲.۸ نکات طراحی ۷۴
۱۱.۲.۸ حذف پد بدون استفاده ۷۴
۱۲.۲.۸ حداقل فاصله درساخت ۷۵
۱۳.۲.۸ فاصله قطعات ار یکدیگر ۷۶
۱۴.۲.۸ مالحظات عمومی در ساختار ۷۶
۱۵.۲.۸ مسخصات مس ۷۷
۱۶.۲.۸ ضخامت دی الکتریک ۷۷
۱۷.۲.۸ اندازه PCB ۷۸
۱۸.۲.۸ تعداد لایه مس در مدارچاپی ۷۸
۱۹.۲.۸ مقدار ratio aspect ۷۸
۲۰.۲.۸ پهنا و فاصله ی track ۷۹
۲۱.۲.۸ قوانین پیشگیری از مشکل مدار -باز روی مسیرهای حساس ۸۱
۲۲.۲.۸ پوشش نهایی روی برد و قطعات ۸۲
۲۳.۲.۸ مسیرهای نیازمند به کنترل امپدانس ۸۲
۲۴.۲.۸ مدارچاپی دیجیتال ۸۳
۲۵.۲.۸ پوشش نهایی مدارچاپی ۸۶
۲۶.۲.۸ برخی اصطالحات فنی ۸۶
۲۷.۲.۸ نحوه ی طراحی بخش سنسور ۸۷
۲۸.۲.۸ نقشه ی شماتیک بخش سنسور ۸۷
۳.۸ نحوه ی طراحی بخش ایستگاه ۹۱

۱.۳.۸ نقشه ی شماتیک بخش ایستگاه ۹۱

۹ تشریح پروژه ۹۴

۱.۹ مقدمه ۹۵

۲.۹ سنسور ۹۶

۱.۲.۹ نحوه خواندن دما و فشار از سنسور BMP180 . . . . . ۹۸

۲.۲.۹ نحوه خواندن شدت نور از سنسور MAX44009 . . . . . ۱۰۱

۳.۲.۹ نحوه دریافت جهت جغرافیایی از سنسور QMC5883L ۱۰۳

۴.۲.۹ نحوه خواندن رطوبت از سنسور AHT10 . . . . . ۱۰۴

۵.۲.۹ نحوه محاسبه سرعت و جهت باد ۱۰۴

۶.۲.۹ نحوه ارسال اطلاعات از طریق لورا ۱۰۵

۷.۲.۹ پیکربندی I2C ۱۰۷

۸.۲.۹ پیکربندی SPI ۱۰۸

۹.۲.۹ حالت توقف میکروکنترلر ۱۰۹

۳.۹ ایستگاه ۱۰۹

۱.۳.۹ نحوه دریافت اطلاعات از لورا ۱۱۰

۲.۳.۹ نحوه ارسال اطلاعات از طریق USB ۱۱۲

۳.۳.۹ حالت خواب میکروکنترلر ۱۱۴

۴.۹ برنامه دسکتاپ ۱۱۴

۱.۴.۹ نحوه طراحی رابط کاربری ۱۱۶

۲.۴.۹ ساختار دیتابیس ۱۱۷

۳.۴.۹ نحوه دریافت اطلاعات از طریق USB ۱۱۸

۱۰ نتایج ۱۲۱

۱.۱۰ مقدمه ۱۲۲

۲.۱۰ تست برنامه دسکتاپ ۱۲۳

۳.۱۰ تست برد ماژول لورا ۱۲۴
۴.۱۰ تست سنسور سنجش سرعت و جهت باد ۱۲۴

مراجع **۱۲۵**

پیوست **۱۲۸**

فصل ۱

مقدمه

مقدمه

ایستگاه هواشناسی، مرکزی مجهز به تجهیزات و ابزارهایی برای اندازه‌گیری‌های جوی است که به ارائه اطلاعات برای پیش‌بینی و مطالعه آب‌وهوا می‌پردازد. اندازه‌گیری انجام‌شده معمولاً شامل دما، فشار هوا، رطوبت، سرعت باد، جهت باد و مقدار بارش است. مشاهدات دستی حداقل یک بار در روز انجام می‌شود، درحالی‌که اندازه‌گیری خودکار حداقل یک بار در ساعت انجام می‌پذیرد.

ایستگاه‌های هواشناسی معمولی مجهز به ابزارهای زیر هستند [۱]:

- رطوبت‌سنج برای اندازه‌گیری رطوبت
- فشارسنج برای اندازه‌گیری فشار جو

- دماسنج برای اندازه‌گیری دمای هوا
- پیرانومتر برای اندازه‌گیری تشعشعات خورشیدی
- باران‌سنج برای اندازه‌گیری میزان بارش باران در طی یک دوره زمانی مشخص
- تجهیزاتی نظیر بادسنج، پرچم باد یا جوراب باد برای اندازه‌گیری سرعت و جهت باد

ایستگاه‌های پیشرفته‌تر همچنین ممکن است شاخص فرابنفش، رطوبت برگ، رطوبت خاک، دمای خاک، دمای آب در حوضچه‌ها، دریاچه‌ها، نهرها یا رودخانه‌ها و گاهی داده‌های دیگر را اندازه‌گیری کنند. به‌جز دستگاه‌هایی که نیازمند تماس مستقیم با عناصر مورداندازه‌گیری هستند (نظیر بادسنج)، دیگر سنسورها و دستگاه‌ها باید در محفظه‌ای به‌دوراز تابش مستقیم خورشید و وزش باد قرار بگیرند.

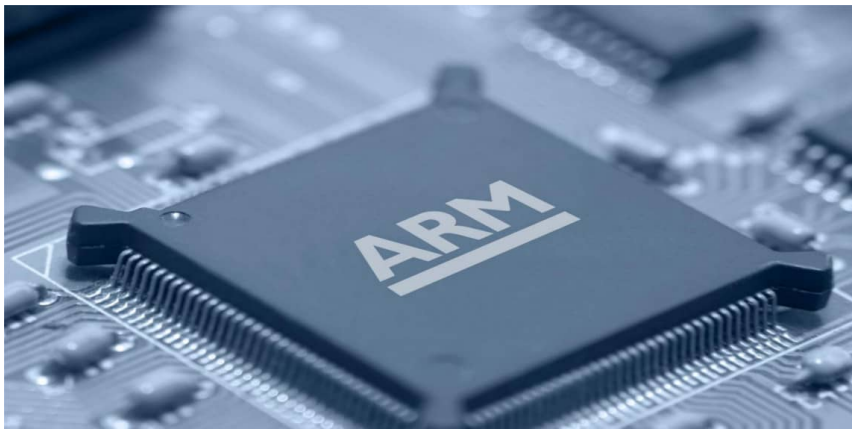
ایستگاه‌های هواشناسی سینوپتیک (Synoptic) ۲۴ ساعته به‌صورت خودکار هر سه ساعت به سه ساعت پارامترهای جوی را پس از اندازه‌گیری و جمع‌آوری از طریق شبکه‌های مخابراتی منتقل می‌کنند. به‌طور مشابه ایستگاه‌هایی با نام متار (Metar) این کار را هر یک ساعت انجام می‌دهند. وظیفه این ایستگاه‌ها جمع‌آوری اطلاعات جوی از محدوده‌هایی وسیع و مخابره به ایستگاه‌های اصلی به‌منظور اطلاع از وضعیت حال و گذشته و پیش‌بینی شرایط آب و هوایی مناطق در آینده است.

هدف این پروژه پیاده‌سازی نوعی ایستگاه هواشناسی سینوپتیک است که با تجهیزات ارزان و کم‌مصرف دیجیتالی پارامترهای جوی لازم را جمع‌آوری و به‌صورت بی‌سیم به ایستگاهی جهت ثبت و نمایش مخابره می‌کند. در این پروژه از میکروکنترلر ARM (Microcontroller) سری STM32F10X به‌عنوان هسته اصلی پردازش در هر دو سمت سنسور و ایستگاه و از ماژول لورا (LoRa) با چیپ SX1278 به‌منظور برقراری ارتباط بی‌سیم استفاده شده است.

فصل ۲

آشنایی با میکروکنترلرهای ARM و نرم‌افزار STM32CubeIDE

۱.۲ مقدمه



ARM یک شرکت بریتانیایی و چندملیتی است که در زمینه نیمه‌رساناها و طراحی نرم‌افزارهای رایانه‌ای فعالیت می‌کند. نحوه تجارت این شرکت به این صورت است که گواهی‌نامه محصولات خود را به عنوان IP Core به فروش می‌رساند و شرکت‌های دیگر از جمله NXP، Microelectronics ST، Qualcomm، Texas Xilinx،

Nvidia، Atmel، Apple این لایسنس را خریداری کرده و محصولات خود را بر اساس آن تولید می‌کنند.

۲.۲ معماری ARM

معماری آرم[1]، نوعی از معماری ۳۲ بیتی و ۶۴ بیتی، بر طبق طراحی RISC CPU و ساختار پردازنده‌های رایانه‌ای است که به‌وسیله شرکت انگلیسی آرم هولدینگز[2] طراحی شده‌است و به دلیل قیمت ارزان، سرعت بسیار زیاد، سخت‌افزارهای جانبی متعدد از جمله: CAN، Ethernet، UART، SPI، USB، DAC، ADC، SDRam و غیره، حافظه داخلی زیاد و توان مصرفی پایین (به ازای هر MHz، جریانی از ۰/۲ میلی‌آمپر تا ۱ میلی آمپر مصرف می کنند). این پردازنده ها؛ اکثر سیستم های نهفته (مثل میکروکنترلرها، موبایل و تبلت و کلا سیستم هایی با حجم کوچک و امکانات بالا) از این پردازنده استفاده می کنند. معماری آرم از دهه ۱۹۸۰ میلادی تا به امروز در حال توسعه و گسترش است. ARM مخفف Advanced RISC Machine است و از آنجایی که این معماری براساس طراحی RISC بنا شده‌است، برای هسته اصلی پردازشگر تنها به حدود ۳۵ هزار ترانزیستور نیاز است و این باعث می‌شود که پردازنده بسیار کم‌مصرف شود، کم‌تر داغ کند و نیازی به خنک‌کننده یا فن نداشته باشد بر خلاف معماری x۸۶ به‌کار رفته در پردازنده‌های شرکت‌های اینتل و ای‌ام‌دی که نیازمند میلیون‌ها ترانزیستور هستند و همین مسئله باعث افزایش توان مصرفی و داغ شدن آنان می‌شود. شرکت آرم هولدینگز در سال ۲۰۱۴ معماری آرم با قابلیت پشتیبانی از دستورالعمل‌های ۶۴ بیتی در پردازنده‌های کورتکس-ای۵۳ و کورتکس-ای۵۷ توسط این شرکت تولید و عرضه شد. از میان شرکت‌هایی که تولیدکننده میکروکنترلرهای ۳۲بیتی هستند؛ میکروکترلرهای کمپانی ST بیشترین محبوبیت را در صنعت دارد که قیمت پایین و در حین حال امکانات بالا و منابع اموزشی کامل از مزایای آن هستند.

۳.۲ میکروکنترلرهای شرکت ST

۱.۳.۲ مقدمه

اس‌تی‌ام‌الکترونیکس، شرکت فرانسوی-ایتالیایی و چندملیتی تولیدکننده تجهیزات الکترونیکی و نیمه‌هادی‌ها می‌باشد، که دفتر مرکزی آن در شهر ژنو، سوئیس قرار دارد. این شرکت بر پایه میزان درآمد، به عنوان بزرگترین سازنده تراشههای نیم‌رساناها در اروپا محسوب می‌شود. در حالی که دفتر مرکزی شرکت اس‌تی‌میکروالکترونیکز و ستاد مدیریتی آن،

[1] ARM

[2] Arm Holdings

در ژنو قرار گرفته‌است، ولی بخش عملیاتی این شرکت، در شهر آمستردام، هلند مستقر می‌باشد. دفتر مرکزی شعبه ایالات متحده این شرکت، در شهر کاپل، تگزاس قرار دارد. ستاد مرکزی شعبه آسیا اس‌تی‌میکروالکترونیکز در سنگاپور و ساختمان مرکزی شعبه ژاپن و کره جنوبی آن نیز، در توکیو مستقر می‌باشند. دفتر مرکزی شعبه چین این شرکت، در شهر شانگهای قرار گرفته‌است. شرکت اس‌تی‌مایکروالکترونیکس، در فهرست اولیه بازار بورس یورونکست ذکر شده‌است و به عنوان جزئی از شاخص فرانسوی کاک ۴۰ محسوب می‌شود. این شرکت همچنین در فهرست ثانویه از بازار بورس نیویورک و نیز بازار بورس ایتالیا، ذکر شده‌است.

۲.۳.۲ نحوه نام‌گذاری

نام‌گذاری این میکروکنترلرها همواره با عبارت ST شروع می‌شود که معرف شرکت سازنده خود می‌باشد. بعد از ترکیب ST حرف M می‌آید که نشانگر این است که محصول حاضر یک میکروکنترلر می‌باشد. بعد از STM یکی از عبارات 32، 8 یا $8A$ را خواهیم دید که به ترتیب معرف میکروهای ۳۲بیتی، ۸بیتی و ۸بیتی Automotive می‌باشد. تا این مرحله خانواده کلی محصول مورد نظر معرفی گردید و سپس نوبت به تشریح ویژگی‌های میکروکنترلر می‌رشد. کاراکتر بعدی در نام‌گذاری فقط شامل یک حرف انگلیسی می‌باشد و یکی از حروف F L، P، S، T، یا W است. این حروف نوع محصول را مشخص می‌کنند. به طور مختصر باید گفت که حرف F مربوط به محصولات Foundation، حرف L مربوط به Ultra-low power، حرف P محصولات Pre-programmed، حروف S مختص Standard، عبارت T برای Touch sensing و در نهایت W برای معرفی محصولات Wireless به کار می‌روند. سپس شاهد یک عدد سه رقمی خواهیم بود. این عدد (مخصوصا رقم سوم) با حرف قبلی در ارتباط است. این عدد ویژگی‌های خاص

MCU – Typical designations and part number suffixes

STM32 F 103 R E T 6

Family

STM32	32-bit MCUs
STM8	8-bit MCUs
STM8A	8-bit Automotive MCUs

Product type

F	Foundation
L	Ultra-low power
P	Pre-programmed
S	Standard
T	Touch sensing
W	Wireless

Specific features (3 digits)

Depends on product series
None exhaustive list.

STM32x...	
051	Entry-level
407	High-performance and DSP with FPU
152	Ultra-low-power
STM8x... / STM8Ax...	
103	Mainstream access line
52	Automotive CAN
31	Automotive low-end

Pin count (pins)

Y	16
F	20
E	24
G	28
K or 6	32
L	34
T	36
D	38
H	40
J	42
S	44
C or 8	48
N	56
U	63
R or 9	64
M or A	80
O	90
V	100
W	128
Q	132
Z	144
I	176
B	208
X	336
P	420

Code size (Kbytes)

0	1
1	2
2	4
3	8
4	16
5	24
6	32
7	48
8	64
9	72
A	96 or 128*
B	128
Z	192
C	256
D	384
E	512
F	768
G	1024
H	1536
I	2048

Note:
* For STM8A only

Packaging

B	Plastic DIP*
D	Ceramic DIP*
G	Ceramic QFP
H	UFBGA or TFBGA
I	UFBGA
M	Plastic SO
P	TSSOP
Q	Plastic QFP
T	Plastic TQFP
U	UQFN
Y	CSP

*Dual in-line package

Temperature range

6 and A	-40 to +85 °C
7 and B	-40 to +105 °C
3 and C	-40 to +125 °C
D	-40 to +150 °C

شکل ۱.۲: جدول نحوه نام‌گذاری میکروکنترلرهای شرکت ST.

هر خانواده میکروکنترلر رو نشان می‌دهد، از جمله معماری، طبقه‌بندی، حداکثر فرکانس، حداکثر حافظه SRAM و حداکثر حافظه Flash. به عنوان مثال عدد ۱۰۳ در میکروکنترلرهای STM32F103 نشانگر این می‌باشد که معماری این میکرو Cortex-M3 در طبقه‌بندی Mainstream با حداکثر فرکانس ۷۲ مگاهرتز، حداکثر حافظه فلش ۱۰۲۴ کیلوبایتی و حداکثر حافظه اِس-رم ۹۶ کیلوبایتی می‌باشد. لازم به ذکر است این اعداد منحصر بفرد می‌باشد و تقریبا الگوی مشخصی ندارد. در این عدد رقم سوم (رقم صدگان) از اهمیت بیشتری برخوردار هستش و اطلاعات اصلی را همین رقم بیان می‌کند. بعد از این عدد دوباره شاهد یک حرف انگلیسی خواهیم بود. این حرف تعداد پین‌های میکروکنترلر مورد نظر را مشخص می‌کند. بعد از آن، یک عدد یا حرف می‌بینیم که بیانگر مقدار حافظه Flash میکروکنترلر می‌باشد. توجه داشته باشید که عدد ۳ رقمی قبل مقدار حداکثری حافظه Flash خانواده را مشخص می‌کرد ولی عدد یا حرف حاضر دقیقا مقدار این حافظه را در این میکروکنترلر برای ما بازگو می‌کند. بعد از این قسمت دوباره شاهد یک حرف انگلیسی خواهیم بود که برای ما مشخص می‌کند میکروکنترلر مورد بحث از چه نوع پکیجی برخوردار است و در نهایت باز شاهد یک عدد یا حرف انگلیسی خواهیم بود که مشخص کننده رنج دمای کاری میکروکنترلر مربوطه می‌باشد. به این صورت که A یا ۶ محدوده ۴۰- تا ۸۰ درجه سانتی‌گراد، B یا ۷ محدوده ۴۰- تا ۱۰۵ درجه، C یا ۳ محدوده ۴۰- تا ۱۲۵ درجه و D محدوده ۴۰- تا ۱۵۰ درجه را مشخص می‌کند. مطالب بیان شده به صورت کلی در شکل ۱.۲ قابل مشاهده است.

۳.۳.۲ انواع کتابخانه‌های مورد استفاده در میکروکنترلرهای ST

CMSIS

یک لایه نرم‌افزاری برای سخت‌افزار پردازنده‌های کورتکس M هست که توسط بنیاد ARM، تعریف‌شده و فارغ از شرکت سازنده میکروکنترلر است. به عبارتی CMSIS ارائه‌شده برای شرکتی چون ST مشابه همان ایCMSIS است که برای شرکت فیلیپس یا شرکت‌های دیگر ارائه شده. این یکسان بودن موجب می‌شود تا اصطلاحاً Portability برنامه نوشته شده، بالا رود. درواقع شما به کمک CMSIS یک‌بار برای میکرویی کد می‌زنید و با کمترین تغییرات ممکن، می‌توانید آن را بر روی میکرویی از شرکتی دیگر انتقال دهید. این لایه بیشتر از آنکه حاوی توابعی برای انجام کارهای مختلف با پریفرال‌های میکرو باشد، دارای تعریف‌های۱ مختلف از رجیسترهای میکروکنترلر است. درنتیجه شما باید بیشتر دست‌به‌کار شوید و خودتان برای راه‌اندازی و استفاده از امکانات میکروکنترلر کد بزنید. اگر بخواهم به‌صورت نه‌چندان دقیقی حرف‌های بالا را خلاصه کنم، باید بگویم CMSIS تنها یکسری اسم است که بر روی آدرس رجیسترهای میکرو گذاشته‌شده تا ما بتوانیم راحت‌تر با این رجیسترها کارکنیم.

کتابخانه‌های SPL و HAL

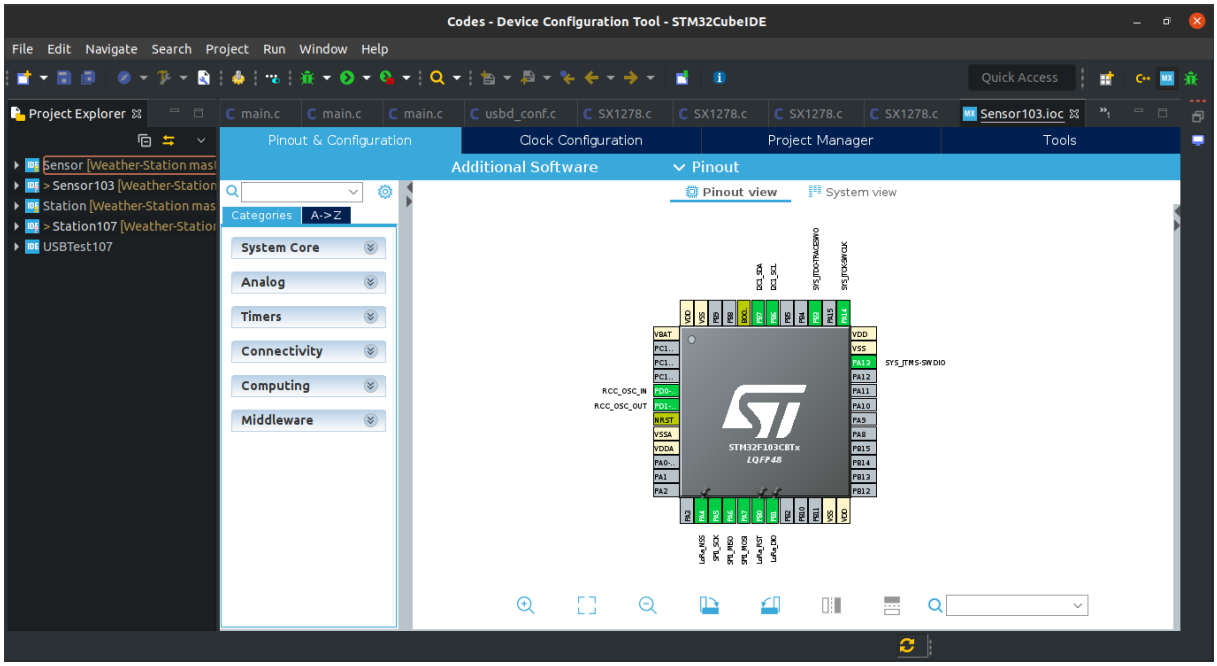
ازلحاظ کلی SPL و HAL شبیه هم‌اند. هر دو دارای توابع زیادی برای کار با قسمت‌های مختلف میکروکنترلر دارند و برخلاف CMSIS صرفاً حاوی تعریف رجیسترها نیستند. بلکه از تعاریف ارائه‌شده در CMSIS، در این دو استفاده‌شده است. SPL قدیمی‌تر از کتابخانه HAL است و هر دو توسط شرکت ST توسعه داده‌شده‌اند. اما ظاهراً شرکت ST علاقه‌ای به پشتیبانی و ادامه کار بر روی SPL ندارد و درنتیجه تمرکز خود را بر روی توسعه هرچه بیشتر و بهتر HAL گذاشته است. شما هم ممکن است قبلاً با کتابخانه SPL کارکرده باشید و آن را بیشتر از HAL بپسندید. با این وجود بهتر است از رویه شرکت پیروی کنیم و هر چه زودتر خود را به کار با توابع HAL عادت دهیم. چراکه علاوه بر اینکه دیگر از کتابخانه SPL پشتیبانی نمی‌شود، برای میکروهای سری M7 و H7 هم این کتابخانه وجود ندارد. بعد از انتشار کتابخانه HAL بسیاری از افراد از این گلایه داشتند که این کتابخانه پردازش را سنگین می‌کند و درون توابع خود به چیزهایی می‌پردازد که برایشان لزومی ندارد. به همین دلیل ST تصمیم گرفت در کنار این کتابخانه، کتابخانه‌ای سبک‌تر به اسم LL[3] ارائه دهد که سطح پایین‌تر از HAL باشد. این‌گونه دست افراد در برنامه‌نویسی بازتر است. هرچند که خودشان بایست نکات گفته‌شده در دیتاشیت و رفرنس‌منوال را رعایت کنند. در پروژه‌های ایجادشده توسط CubeMX ما می‌توانیم صرفاً با اضافه کردن خطی برای اینکلود این کتابخانه، در کنار کتابخانه HAL از توابع

[3] Low Layer

LL در قسمت‌هایی که می‌خواهیم، استفاده کنیم. اگر می‌خواهید در لایه بالاتر کد بزنید و البته کد نوشته‌شده را راحت‌تر برای میکرو دیگری از شرکت ST نیز استفاده کنید، بهتر است سراغ HAL بروید. اما اگر سرعت از اهمیت بیشتری برایتان برخوردار است و از شلوغی توابع HAL فراری هستید، بهتر است با LL برنامه خود را توسعه دهید.

۴.۳.۲ نرم‌افزار STM32CubeIDE

اخیرا شرکت ST Microelectronics به‌دلیل ارائه نرم‌افزار STM32CubeIDE محبوبیت زیادی در میان دیگر شرکت‌ها پیدا کرده است. این نرم‌افزار کار با میکروکنترلرهای این شرکت را برای توسعه‌دهندگان بسیار ساده تر کرده است. این نرم افزار IDE بسیار قدرتمندی برای میکروکنترلرهای این شرکت است و به راحتی می‌توان امکاناتی که در پروژه به آن نیاز داریم را تنها با چند کلیک فعال کنیم. پس از انتخاب مورد نظر کد را به صورت خودکار بر اساس تنظیمات انجام شده تولید می‌کند. تصویری از محیط تنظیم امکانات این برنامه در شکل ۲.۲ قابل مشاهده است.



شکل ۲.۲: تصویر محیط نرم‌افزار STM32CubeIDE.

نکته‌ای که در اینجا حائز اهمیت است، این است تعدادی از خطوط برنامه توسط نرم‌افزار STM32CubeIDE تولید شده است که نباید آن‌ها را حذف کنیم. دلیل وجود این کامنت‌ها این است که برنامه‌نویس کدهای خود را در قسمت‌های مشخص‌شده بنویسد و اگر پس از تکمیل پروژه نیاز به فعال کردن واحدی در میکروکنترولر داشت، بتواند دوباره به محیط تنظیمات وارد شد و پس از فعال‌کردن واحد موردنظر، تغییرات مورد نظر در کد اصلی بدون هیچ خطایی اعمال شود. در انتها نیز پس از اتمام کدنویسی، باید آن را توسط پروگرامر به میکروکنترلر انتقال دهیم. پروگرامر های مختلفی برای میکروکنترلرهای ARM موجود است که پروگرمرهای JLINK و STLINK از معروف‌ترین آن‌ها است

که بسیاری از میکروکنترلرها را پشتیبانی می کنند. در شکل ۳.۲ تصاویری از پروگرامرهای استفاده شده در پروژه را مشاهده می‌کنید.

(آ) پروگرمر STLINK (ب) پروگرمر JLINK

شکل ۳.۲: تصاویر پروگرمرهای استفاده شده در این پروژه.

۴.۲ میکروکنترلر

هسته اصلی پردازش در هر دو سمت ایستگاه و سنسور میکروکنترلر STM32f103CBT6 انتخاب شده است که با توجه به موجود بودن در بازار ایران و دارا بودن ۲ عدد I^2C[۴]، ۲ عدد SPI[۵]، اینترفیس[۶] USB[۷] و ۳ عدد تایمر ۱۶ بیتی نیاز به حداقل ۱ عدد I^2C (در سمت سنسور)، ۱ عدد SPI (در هر دو سمت)، اینترفیس USB (در سمت ایستگاه) و ۱ عدد تایمر (در سمت سنسور) را برآورده می‌کند. همچنین حالت Sleep و واحد RTC موجود در این میکروکنترلرها به کاهش مصرف انرژی در وقفه‌های سه ساعته کمک می‌کند؛ به‌طوری‌که استفاده از سیستم باتری و پنل خورشیدی را ممکن می‌سازد.

[4] Inter-Integrated Circuit

[5] Serial Peripheral Interface

[6] Interface

[7] Universal Serial Bus

فصل ۳

آشنایی با LoRa

۱.۳ مقدمه

اینترنت اشیاء[1] واژه‌ای که این روزها برسر زبان بسیاری از افراد است. بدون شک این تکنولوژی آینده فناوری اطلاعات را در دست خواهد گرفت. جهت پیاده سازی این تکنولوژی زیرساخت‌های خوبی لازم است چرا که داده ها در این تکنولوژی بسیار زیاد است و همچنین دیگر تکنولوژی هایی چون Wi-Fi و Li-Fi جهت ارتباطات جواب گوی این حجم انتقالات نیست و باید به فکر تکنولوژی های جایگزین بود.

در حوزه اینترنت اشیاء، فاکتورهای بسیاری از جمله هزینه Nodeها، هزینه شبکه، طول عمر باتری، نرخ اطلاعات، تاخیر، مسافت پوشش‌دهی و محل استقرار اهمیت پیدا می‌کند. هیچ فناو ری به‌تنهایی قادر نیست به‌طور همزمان نسبت به تمامی بخش‌ها پاسخ‌گو باشد اما در این میان، از فناوری LoRa به‌دلیل داشتن بیشترین ویژگی مثبت، می‌توان استفاده کرد.

[1] IOT: Internet of things

شبکه LoRaWAN یک پروتکل توان پایین با برد وسیع[۲] است که به‌خصوص برای دستگاه های بیسیم در اینترنت اشیا طراحی شده است و در سطح شبکه های منطقه ای، ملی یا جهانی میتواند عمل کند. معماری شبکه LoRaWAN به صورت توپولوژی ستاره یا استار میباشد که در آن گیتوی‌ها پیام ها را بین دستگاه های پایانی و سرور مرکزی شبکه انتقال می دهند. گیتوی‌ها مانند یک پل نامرئی عمل می کنند و از طریق اتصالات استاندارد آی پی به سرور در شبکه متصل می شوند، به سادگی بسته‌های RF را به بسته‌های IP تبدیل می کنند و بلعکس. همان‌طور که در شکل ۱ مشاهده می‌کنید، تعداد زیادی Node می‌توانند به گیتوی متصل شوند که معمولا این Nodeها به سنسورهایی مثل سنسور دود، رطوبت، دما و ... متصل هستند و اطلاعات این سنسورها را به گیتوی مخابره می‌کنند.

به‌طورکلی در این پروژه مصرف پایین انرژی به دلیل استفاده از سیستم باطری و پنل خورشیدی بسیار اهمیت دارد. استفاده از شبکه تلفن همراه[۳] به‌عنوان راه‌حلی ابتدایی برای ارتباط بی‌سیم، علاوه بر نداشتن صرفه اقتصادی مصرف انرژی زیادی را به سیستم تحمیل می‌کند. همچنین تضمینی برای وجود پوشش شبکه تلفن همراه در مناطقی که قرار است داده‌های جوی از آن جمع‌آوری شود وجود ندارد. ازاین‌رو بهترین رویکرد استفاده از گیرنده و فرستنده‌های رادیویی در باندهای فرکانسی بدون نیاز به مجوز (ISM) است. از میان گزینه‌های موجود ماژول‌های لورا[۴] به لطف مدولاسیون CSS[5] که از آن بهره می‌برند دارای مصرف توان پایین، ناحیه پوشش وصیع و نفوذپذیری مناسبی هستند که کاملاً با نیازهای ما سازگار است. در ادامه به بررسی و مقایسه این تکنولوژی با سایر گزینه‌های موجود پرداخته می‌شود.

۲.۳ جنبش جهانی شبکه اشیاء (TTN:The Things Network)

با هدف ایجاد یک شبکه جهانی باز و عمومی (شهر هوشمند) ، غیر متمرکز و مبتنی بر همکاری جمعی، مدتی است که با سرعت بالا در سراسر جهان در حال فراگیر شدن است. به این صورت که تعدادی دروازه در نقاط مختلف کشور نصب می‌شود و به توسعه‌دهندگان صنعت اینترنت اشیا اجازه داده می‌شود که محصولات خود را بطور رایگان به این درگاه ها متصل کنند و پس از ایجاد حساب در سایت The Things Network[۶] می‌توانند از هر نقطه‌ای محصولات خود را کنترل کنند. هر دروازه LoRa، توانایی پشتیبانی هزاران Node را دارد. این درگاه‌ها می‌توانند تا شعاع ۱۵ کیلومتری را پوشش دهند. در شکل ۱.۳، نقاط پوشش‌دهی LoRaWAN را در سطح جهانی مشاهده می‌کنید.

شبکه اشیا ابتدا در شهر آمستردام هلند آغاز به کار کرد و در مدت چهار هفته، کل این شهر تحت پوشش شبکه

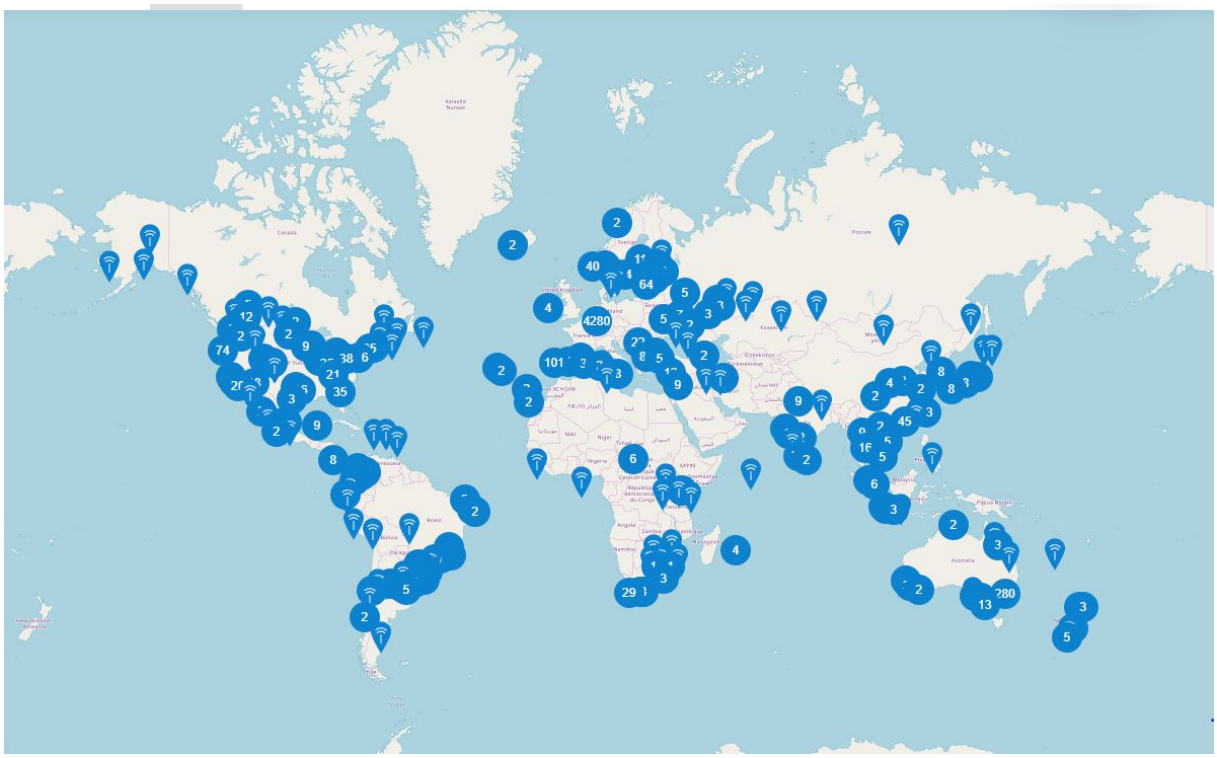
[2] LPWA

[3] The Global System for Mobile Communications (GSM)

[4] LoRa (Long Range)

[5] Chirp Spread Spectrum

[6] www.thethingsnetwork.org



شکل ۱.۳: نقاط پوشش‌دهی LoRaWAN در سطح جهانی.

اینترنت اشیا قرار گرفت. کشور ایران نیز پیش‌تر درخواست خود را برای پیوستن به این شبکه اعلام کرده و در سال ۱۳۹۵ موفق به کسب موافقت برای راه‌اندازی شبکه اشیا در تهران شد. اولین درگاه، در همان سال در منطقه نارمک تهران نصب شد. این نخستین درگاه شبکه اشیاء در خاورمیانه است و ایران را در کنار کشورهای دارای این فناوری قرار می‌دهد[۲].

مخاطب اصلی این شبکه را اپراتورهای ارتباطی، استارتاپ‌ها و کسب و کارهای مرتبط با فناوری اطلاعات است. این شبکه در واقع شبکه‌ای غیرانتفاعی است که به‌جز هزینه‌های پایین سرمایه‌گذاری اولیه، اجرایی و نگهداری، هزینه دیگری در بر ندارد و هدف از توسعه آن درآمدزایی مستقیم نیست. چنین شبکه‌ای می‌تواند بستری شود تا شرکت‌ها و افراد خلاق با ارائه خدمات نوآورانه خود، از آن‌ها کسب ارزش کنند.

تفاوت ارائه خدمات ارائه خدمات بر روی این شبکه با سرویس‌هایی که از طریق شبکه‌های ارتباطی موبایل ارائه می‌شود، توان مصرفی بسیار پایین و در مقابل پوشش گسترده آن است. از سوی دیگر، به هیچ عنوان هزینه این اتصال قابل قیاس با اتصال اشیاء از طریف سیم‌کارت موبایل نیست. برای مثال این شبکه می‌تواند در حوزه خدمات شهری برای هوشمندسازی پارکینگ‌ها، کنتورهای هوشمند، اتصال حسگرهای انبوه برای اندازه‌گیری آلودگی، رطوبت و... و نهایتا کاهش ترافیک و میزان آلودگی کاربرد داشته باشد. به عبارتی کاربرد اصلی این شبکه را توسعه دهندگان تعریف می‌کنند و بسته به نیاز و خلاقیت خود می‌توانند به کاربردهایی بیندیشند که تا کنون ارائه آن‌ها ممکن نبوده و یا هزینه

زیادی در پی داشته است. با راه‌اندازی این زیرساخت، شرکت‌ها دیگر مجبور نیستند که به صورت جزیره‌ای شبکه ایجاد کنند و به جای آن می‌توانند بر روی خدمات مناسب تمرکز کرده و از آن‌ها ارزش‌آفرینی کنند.

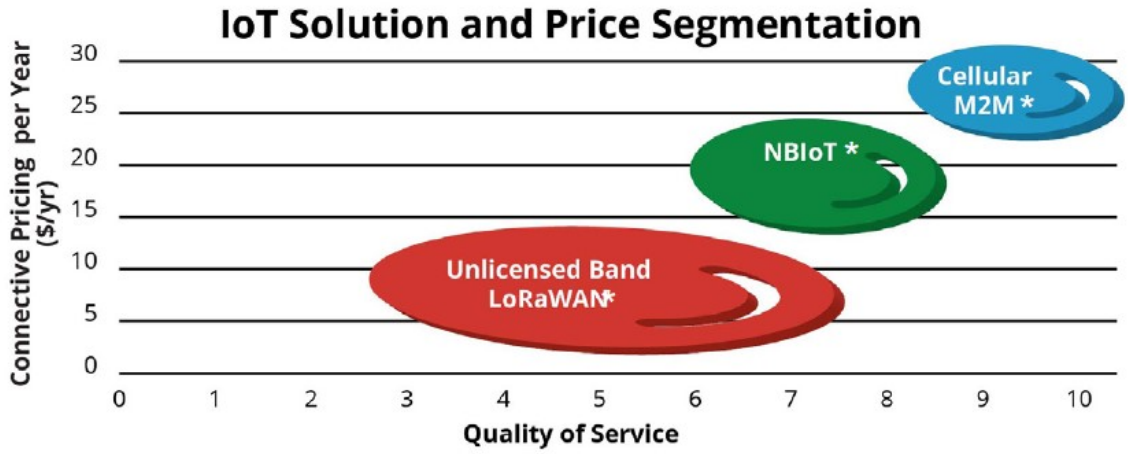
این شبکه قرار نیست رقیب شبکه‌های کنونی ثابت و سیار باشد، در حقیقت شبکه اشیاء آمده است تا جای خالی خود را پر کند و به عنوان شبکه‌ای مکمل این دو عمل کند. این امکان را فراهم می‌کند تا کاربردهایی که پیش‌تر به دلیل هزینه‌های بالای اتصال، امکان بروز و ظهور نمی‌یافتند، اکنون عملیاتی شوند. ایجاد چنین شبکه‌ای می‌تواند بستری شود تا شرکت‌ها و افراد خلاق با ارائه خدمات نوآورانه خود، از آن‌ها کسب ارزش کنند. این ماژول، رقابت تنگاتنگی با ماژول NB-IOT دارد که توسعه‌دهندگان با توجه به نیازهای خود، یکی از این دو را انتخاب می‌کنند. در ادامه، این دو ماژول را باهم مقایسه خواهیم کرد.

۳.۳ مقایسه ماژول LoRa و NB-IOT

در حوزه اینترنت اشیا فاکتورهای بسیاری از جمله هزینه Nodeها، هزینه شبکه، طول عمر باتری، نرخ انتقال داده، تاخیر، تحرک، رنج، پوشش دهی و مدل استقرار اهمیت پیدا می کند. هیچ تکنولوژی به تنهایی قادر نیست به طور همزمان نسبت به تمامی بخش‌ها پاسخگو باشد. فناوری NB-IOT و LoRa هر کدام دارای ویژگی‌های فنی و تجاری مخصوص به خود می‌باشند که در زیر به بیان آنها خواهیم پرداخت:

۱. طیف، QOS و هزینه: LoRa در طیف فرکانسی بدون لایسنس و زیر ۱ گیگاهرتز مورد استفاده قرار می‌گیرد، لذا برای کاربران آن، هزینه‌ای در این خصوص نخواهد داشت، در حالی که شبکه‌های NB-IOT و Cellular از باندهای لایسنس‌دار زیر ۱ گیگاهرتز استفاده می‌کنند. علت این امر آن است که باندهای فرکانسی زیر گیگاهرتز یعنی بین ۵۰۰ مگاهرتز و ۱ گیگاهرتز، برای ارتباطات در فواصل زیاد، سایز شبکه و بهره‌وری از آنتن‌ها مناسب می‌باشد. LoRaWAN از یک طیف بدون لایسنس رایگان و یک پروتکل آسنکرون استفاده می‌کند از این رو از نظر طول عمر باتری و هزینه بهینه می‌باشد. از طرفی شبکه LoRa و LoRaWAN نمیتوانند همانند شبکه سلولار، QoS مشابه ارائه دهند. با وجود اینکه شبکه سلولار و NB-IOT از دیدگاه QoS بهینه هستند ولی از دیدگاه طول عمر باتری قابل مقایسه با تکنولوژی LoRa نیستند. در شکل ۲.۳، هزینه نگهداری سالیانه راهکارهای مختلف در حوزه اینترنت اشیاء را مشاهده می‌کنید.

۲. طول عمر باتری و تاخیر در Downlink: طول عمر باتری از دو جنبه مهم قابل بررسی است:
مصرف جریان تجهیزات (پیک و میانگین) و نقش پروتکلLoRaWAN. یک پروتکل آسنکرون مبتنی بر ALOHA می باشد، به این معنی که تجهیزات مادامی که درخواستی مطرح نشود می تواند استراحت کنند. در حالی‌که در پروتکل

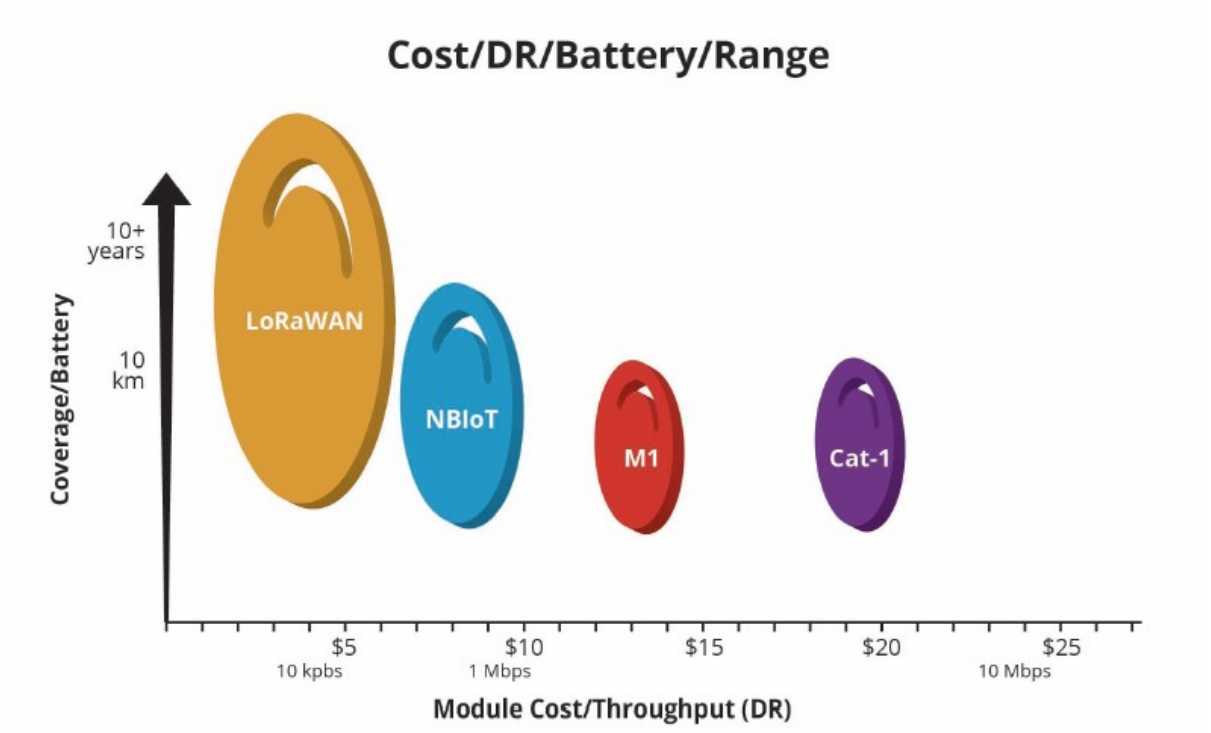


شکل ۲.۳: دسته‌بندی هزینه نگهداری سالیانه راهکارهای مختلف در حوزه اینترنت‌اشیاء.

سنکرون مبتنی بر شبکهCellular ، تجهیزات می‌بایست به صورت منظم با شبکه هماهنگ شوند. به عنوان مثال به طور میانگین تلفن‌های سلولار می‌بایست هر ۵/۱ ثانیه با شبکه سینک شود حتی چنانچه استفاده نشوند. در مقایسه در NB-IOT نیز اگرچه همزمانی اغلب کمتر اتفاق می‌افتد اما همچنان منظم می‌باشد که سبب می‌شود انرژی بیشتری از باتری استفاده نماید. در حالی که مدولاسیونی که در شبکه سلولار مورد استفاده قرار می‌گیرد از دیدگاه استفاده از طیف موثرترین است اما از دیدگاه تجهیزات انتهایی چندان کارآمد نیست. در مدولاسیون سلولار FDMA به منظور ایجاد مدولاسیون به یک فرستنده خطی نیاز داریم و یک فرستنده خطی نیز در مقایسه با مدولاسیون غیر خطی به میزان بیشتر از پیک جریان نیاز دارد. این میزان جریان بالاتر از پیک، نیاز به باتری بیشتر و گرانتری دارد تا آن را پشتیبانی نماید. خاصیت سنکرون یک شبکه سلولار فواید بسیاری در کاربردهایی که نیاز به تاخیر کوتاه در downlink دارند ایجاد می نماید.

NB-IOT همچنین می‌تواند برای کاربردهایی که نیاز به throughput داده بالاتری دارند نرخ داده بالاتری را ارائه نماید. LoRaWAN با پشتیبانی از کلاس B که به گونه ای طراحی شده است که تاخیر downlink را با بیداری تجهیزات در فواصل قابل برنامه ریزی کاهش دهد. برای کاربردهایی با نیازمندی‌هایی همچون ارتباطات بسیار مکرر و تاخیر خیلی کوتاه یا حجم اطلاعات بالا، NB-IOT بهترین گزینه خواهد بود. اگرچه در کاربردهایی که نیاز به باتری با طول عمر بسیار بالا و هزینه بهبود یافته می باشد، LoRa بهترین گزینه است. در شکل ۳.۳، مقایسه‌ای بین رنج پوشش‌دهی، قیمت ماژول و طول عمر باتری را مشاهده می‌کنید.

۳. پوشش شبکه: یکی از فواید NB-IOT این است که زیر ساختهای موجود به منظور سرویس دهی، قابل ارتقا می‌باشد، اگرچه این ارتقا محدود به ایستگاه‌های پایه 4G/LTE می‌باشد و گران است. البته این استراتژی برای محیط‌های شهری پرتراکم که پوشش شبکه 4G/LTE دارند مناسب می‌باشد، برای نواحی روستایی و یا غیرشهری که



شکل ۳.۳: مقایسه‌ای بین رنج پوشش‌دهی، قیمت ماژول و طول عمر باتری.

چنین پوشش‌دهی وجود ندارد ایده آل نیست. ویژگی های NB-IOT در ماه ژوئن سال ۲۰۱۶ منتشر شد و پیش‌بینی شد که در نیمه اول سال ۲۰۱۷ در دسترس باشد. یکی از ویژگی‌های LoRaWAN قابلیت کار اجزای آن در مدل‌های اینترپرایز و خصوصی به خوبی مدلهای شبکه عمومی می‌باشد. در حالی که NB-IOT تنها محدود به مدل عمومی ایستگاه‌های پایه سلولار می باشد.

۴. هزینه تجهیزات و شبکه: از دیدگاه تجهیزات انتهایی، پروتکل LoRaWAN در مقایسه با NB-IOT ساده‌تر است و به آسانی با هزینه پایین قابل پیاده سازی است. همچنین مدولاسیون و پروتکل NB-IOT پیچیده‌تر است. بنابراین هزینه راهکار را افزایش می‌دهد.

۴.۳ معماری LoRa

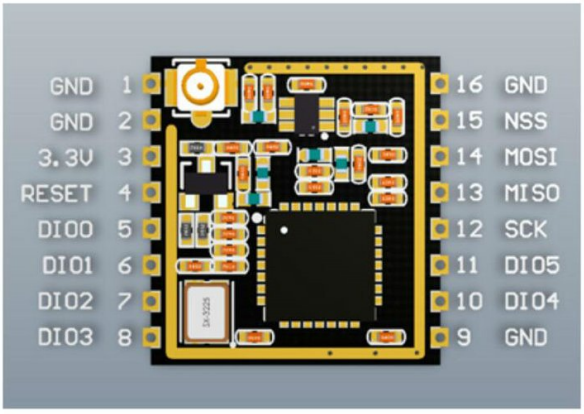
LoRa مخفف Long Range Ratio است و یک فناوری مدولاسیون طیف گسترده[۷] است که از فناوری CSS[۸] برگرفته شده است. CSS در سال ۱۹۴۰ برای رادارهای نظامی طراحی شد. مهم‌ترین ویژگی‌های LoRa برد بالا، هزینه پیاده‌سازی پایین، مصرف توان پایین و سرعت انتقال داده پایین می‌باشدکه کاربردهای متنوعی در حوزه اینترنت

[7] Spread Spectrum

[8] Chirp Spread Spectrum

اشیاء دارد. در LoRa، گسترش طیف با تولید یک سیگنال CHIRP که بطور مداوم در فرکانس‌های مختلف جابجا می‌شود. این فناوری، در ابتدا در شهر گرنوبل فرانسه توسط شرکت SEMTECH در سال ۲۰۱۲ معرفی شد. شرکت SIMTECH لایسنس محصولات خود را به‌عنوان IP CORE به فروش می‌رساند و شرکت‌های دیگر از جمله MICROCHIP، NXP و AI-THINKER ماژول خود را بر اساس طراحی‌های شرکت SIMTECH می‌سازند.

در بازار ایران تنها محصولات شرکت AI-THINKER در دو مدل RA-01 و RA-02 عرضه می‌شود که هر دو بر اساس معماری SX1278 شرکت SEMTECH ساخته شده‌اند. هسته پردازنده این ماژول از محصولات شرکت STM است که ما برای این پروژه از مدل RA-02 استفاده کردیم که علاوه پروتکل LoRa، پروتکل‌های FSK و OOK را نیز پشتیبانی می‌کند. در شکل ۴.۳ نمای داخلی و خارجی این ماژول را مشاهده می‌کنید.



(ب) نمای داخلی ماژول RA-02

(آ) نمای خارجی ماژول RA-02

شکل ۴.۳: نمای داخلی و خارجی ماژول RA-02.

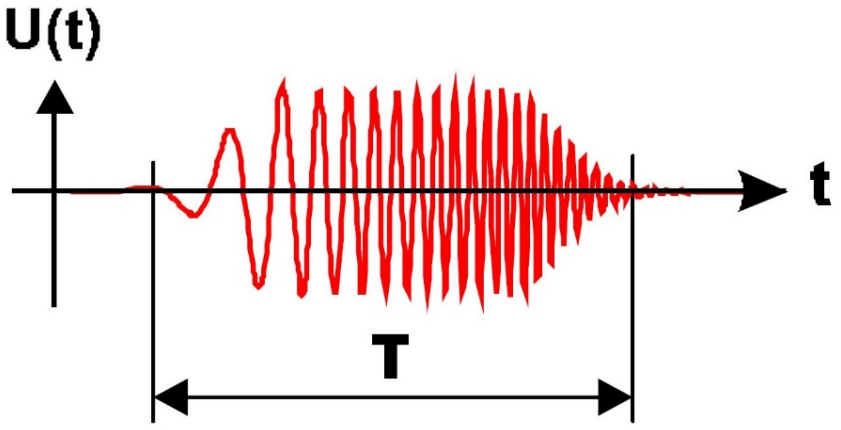
ماژول‌های LoRa در بازه‌های فرکانسی مختلف تولید می‌شود. ماژول خریداری شده حتما می‌بایست با قوانین سازمان تنظیم مقررات و ارتباطات رادیویی کشور مورد نظر، سازگاری داشته باشد. (در آسیا، فرکانس ،۴۳۳MHZ در اروپا ۸۶۸MHZ و در آمریکای شمالی ۹۱۵MHZ محدوده مجاز است)

در ژانویه سال ۲۰۱۸، نسل جدیدی از LoRa رونمایی شد که از ویژگی‌های آن، افزایش توان انتقال، کاهش توان مصرفی و کاهش ابعاد آن نسبت به مدل‌های قدیمی است. البته هنوز محصولی از آن به بازار عرضه نشده است.

نرخ انتقال داده: حداکثر نرخ داده در LoRa مقدار ۳۰kb/s می‌باشد که نسبتا مقدار کمی است.

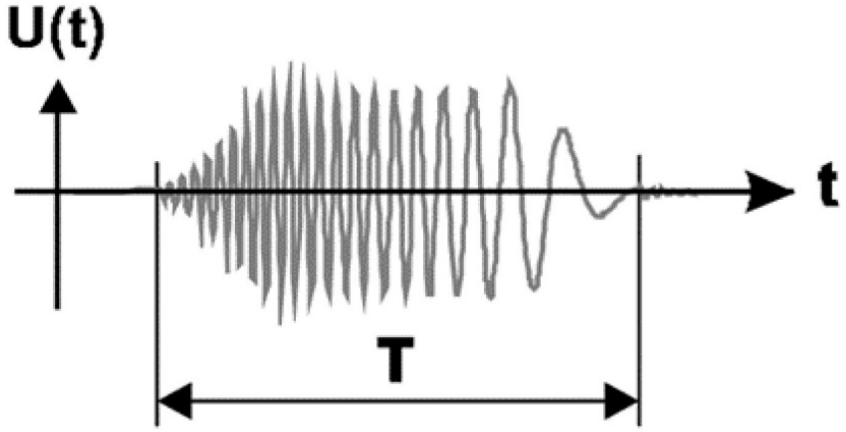
CSS (Chirp Spread Spectrum): یک Chirp، یک سیگنال سینوسی از افزایش یا کاهش فرکانس است

که به ماهیت سیگنال‌های Chirp برای پخش طیف پهنای‌باند سیگنال ارسالی وابسته است. یک Chirp از افزایش فرکانس را که up-chirp می‌نامند، در شکل ۵.۳ مشاهده می‌کنید.



شکل ۵.۳: up-chirp در واحد زمان.

یک Chirp از کاهش فرکانس را که down-chirp می‌نامند، در شکل ۶.۳ مشاهده می‌کنید.

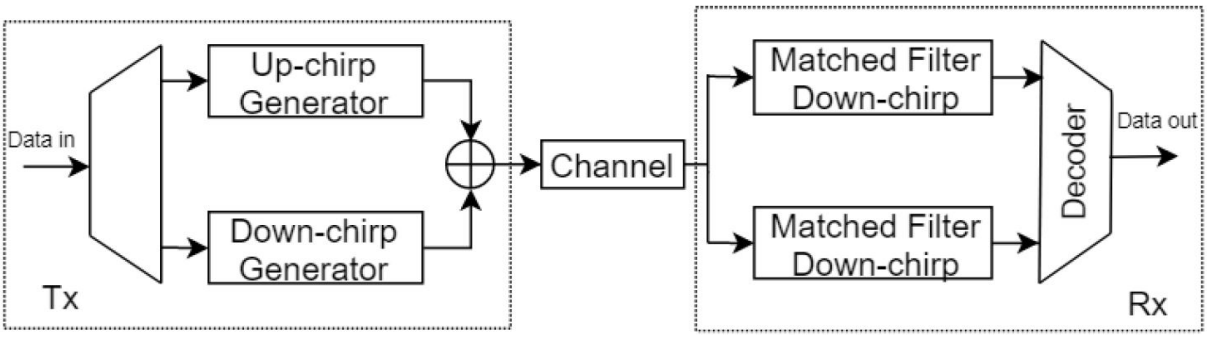


شکل ۶.۳: down-chirp در واحد زمان.

up-chirp دارای مقدار مثبت و down-chirp دارای مقدار منفی است. تغییر فرکانس می‌تواند خطی یا نمایی باشد. پهنای باند سیگنال Chirp، تفاوت بین فرکانس شروع و فرکانس پایانی است. ۱. فشرده‌سازی پالس: فشرده‌سازی پالس، فرآیندی است که طی آن پالس طولانی مدت با قدرت پیک پایین به یک پالس کوتاه‌مدت با پیک بالا تبدیل می‌شود. پالس‌های مستقیم با استفاده از یک فیلتر همگام[9] و توسط هم‌بستگی[10]، فشرده‌سازی می‌شوند. شکل ۷.۳ ساختمان سیستم CSS را نشان می‌دهد داده‌ها با استفاده از up-chirp و down-chirp در سمت فرستنده مدوله می‌شود و با استفاده از همبستگی با فیلتر همگرا برای فشرده‌سازی پالس، به‌دست می‌آیند و

[9] matched filter
[10] correlation

به‌عنوان پالس تیز انرژی بالا که می‌تواند به راحتی رمزگشایی شود.



شکل ۷.۳: نمودار بلوک CSS.

یکی از مهم‌ترین ویژگی‌های CSS، مقیاس‌پذیری نرخ سرعت داده است. برای گسترش سیگنال هر دو در زمان و فرکانس به‌طور مستقل، می‌توان از طیف گسترده‌ای از Chirp استفاده کرد.

۲. پخش فرکانسی: همانطور که در بخش قبلی توضیح داده شده است، گسترش پهنای‌باند سیگنال می‌تواند در کاهش اثر نویز کانال کمک کند و سیگنال را در برابر تداخل ایمن سازد. با استفاده از پالس‌های Chirp پهنای‌باند بسیار بالاتر، پهنای باند سیگنال می‌تواند به اندازه مورد نیاز افزایش یابد.

۳. پخش زمانی: از آنجایی که پهنای‌باند سیگنال‌های Chirp تنها به شروع و پایان فرکانس Chirp بستگی دارد، سرعت داده از مدولاسیون Chirp می‌تواند به طور مستقل از پهنای باند بالا و پایین شود. بنابراین این امکان وجود دارد که آزادانه پهنای‌باند و سرعت داده انتخاب شود.

ساختمان بسته[11] LoRa: ساختمان بسته LoRa را در شکل ۸.۳ مشاهده می‌کنید.

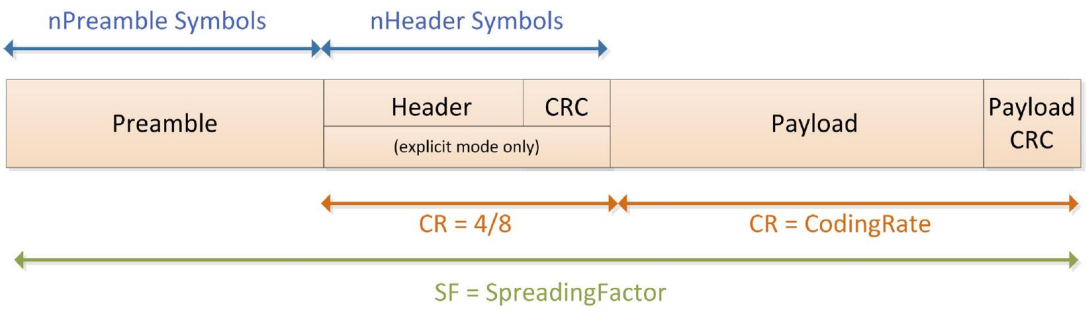


Figure 6. LoRa™ Packet Structure

شکل ۸.۳: ساختمان بسته LoRa.

به‌طورکلی در LoRa دو فرم برای ارسال داده داریم و می‌توان آن را با بیت ImplicitHeaderModeOn که در رجیستر RegModemConfig1 وجود دارد، تنظیم کرد:

[11] Packet

۱. Explicit: شامل یک سرصفحه[۱۲] که حاوی اطلاعات در مورد تعداد بایت‌ها، نرخ کدگذاری و وجود یا عدم وجود [۱۳] ۱۶ بیتی در بسته. در این مدل، حضور CRC در انتهای بارگیری تنها در سمت فرستنده از طریق بیت RxPayloadCrcOn در رجیستر RegModemConfig1 ثبت می‌شود. در سمت گیرنده، این بیت استفاده نمی‌شود و فقط بارگیری دریافت می‌شو. کاربر می‌بایست بیت CrcOnPayload در رجیستر RegHopChannel را کنترل کند.در صورت '۱' بودن باید بیت IrqFlag را از رجیستر PayloadCrcError تا مطمئن شود که CRC معتبر است. '۰' بودن بیت CrcOnPayload، به این معنی است که CRC در بارگیری حضور ندارد. در نتیجه IrqFlag حتی اگربارگیری خطا داشته باشد، فعال نخواهد شد. به جدول ۱.۳ توجه کنید.

جدول ۱.۳: وضعیت CRC در حالت Explicit .

Explicit Header	Transmitter	Receiver	CRC Status
	0	0	CRS is not checked
Value of the bit	0	1	CRS is not checked
RxPayloadCrcOn	1	0	CRS is checked
	1	1	CRS is checked

۲. Implicit: در جایی که مقدار بارگیری[۱۴]، نرخ کدگذاری و حضور CRC ثابت یا شناخته شده است بهتر است سر صفحه را از بسته حذف کنیم و این مقادیر را دستی تنظیم کنیم. در این مدل لازم‌است تا بیت RxPayloadCrcOn در رجیستر RegModemConfig1 را در هر دو سمت فرستنده و گیرنده تنظیم کنیم. توجه داشته باشید که اگر مقدار پارامتر SF برابر ۶ باشد، ماژول فقط در مدل Implicit می‌تواند کار کند.

پیشوند[۱۵]: این قسمت برای همگام‌سازی گیرنده با داده ورودی کاربرد دارد. بطور پیش‌فرض، بسته با دنباله‌ای به طول پیوند ۱۲نمادی[۱۶] پیکربندی می‌شود.این مقدار با تنظیم رجیستر PreambleLenght قابل برنامه‌ریزی است و میتواند بین ۶ تا ۶۵۵۳۵ مقدار بگیرد. این رجیستر باید در سمت فرستنده و گیرنده باید یکسان پیکربندی شود. در جایی که طول پیوند فرستنده متغیر است، در گیرنده باید بیشترین طول پیوند که ممکن‌است رخ دهد، تنظیم شود.

نرخ کدگذاری: جهت بهبود پایداری اتصال، از این ویژگی برای اصلاح خطاهای پیش‌رو استفاده می‌شود. اضافه کردن این ویژگی به کد، موجب افزایش تعداد بیت‌های ارسالی می‌شود. در جدول ۱ نرخ OVERHEAD به ازاء نرخ

[12] Header

[۱۳] نام واحدی است مشخص‌کردن خطای داده ارسالی که گیرنده می‌تواند به کمک آن سرصفحه‌های نامعتبر را دریافت نکند.

[14] Payload

[15] Preamble

[16] Symbol

کدگذاریهای مختلف نشان داده شده است.

عامل گسترش[۱۷]: پارامتری است که مقدار آن می‌تواند بین ۶ تا ۱۲ باشد. هر چه مقدار آن بیشتر باشد، نرخ دیتا پایین‌تر، حساسیت، برد و توان مصرفی آن بیشتر می‌شود و هرچه مقدار آن پایین‌تر باشد، نرخ دیتا بالاتر، حساسیت، برد و توان مصرفی آن کمتر می‌شود. همچنین با فرکانس و پهنای باند مشخص، اگر عامل‌گسترش فرستنده با گیرنده یکسان نباشد، گیرنده نمی‌تواند پیام را دریافت کند. در جدول ۲.۳ SFهای مختلف و پارامتر SNR آن‌ها را نمایش می‌دهد. در جدول ۳.۳ نرخ داده و SFهای مختلف را نمایش می‌دهد.

جدول ۲.۳: مقدار SNR به ازاء عامل گسترش‌های مختلف.

Spreading Factor (RegModulationCfG)	Spreading Factor (Chips / symbol)	LoRa Demodulator SNR (dB)
6	64	-5
7	128	-7،5
8	256	-10
9	512	-12،5
10	1024	-15
11	2048	-17،5
12	4096	-20

جدول ۳.۳: نرخ داده‌های LoRa.

Spreading Factor	Bit rate (bits / s)
7	5469
8	3125
9	1758
10	977
11	537
12	293

Time-On-Air: مدت زمانی که طول می‌کشد تا یک بسته از ماژول فرستنده به ماژول گیرنده برسد و مقدار آن

[17] Spreading Factor (SF)

از رابطه زیر بدست می‌آید:

$$T_s = \frac{2^{SF}}{BW} \tag{۱.۳}$$

پارامترهای مختلفی از جمله نرخ کدگذاری، پهنای باند، SF، جهت رسیدن به برد و سرعت داده مورد استفاده قرار می‌گیرند. در جدول تصویر ؟؟fig :TOA مقدار TOA را با ترکیب مقادیر مختلف این پارامترها و نیز سایز بارگیری‌های مختلف نشان داده شده است.

Range

Throughput

			time on air in second for payload size of					
BW	CR	SF	5 bytes	55 bytes	105 bytes	155 Bytes	205 Bytes	255 Bytes
125	4/5	12	0.95846	2.59686	4.23526	5.87366	7.51206	9.15046
250	4/5	12	0.47923	1.21651	1.87187	2.52723	3.26451	3.91987
125	4/5	10	0.28058	0.69018	1.09978	1.50938	1.91898	2.32858
500	4/5	12	0.23962	0.60826	0.93594	1.26362	1.63226	1.95994
250	4/5	10	0.14029	0.34509	0.54989	0.75469	0.95949	1.16429
500	4/5	11	0.11981	0.30413	0.50893	0.69325	0.87757	1.06189
250	4/5	9	0.07014	0.18278	0.29542	0.40806	0.5207	0.63334
500	4/5	9	0.03507	0.09139	0.14771	0.20403	0.26035	0.31667
500	4/5	8	0.01754	0.05082	0.08154	0.11482	0.14554	0.17882
500	4/5	7	0.00877	0.02797	0.04589	0.06381	0.08301	0.10093

شکل ۹.۳: مقدار TOA در برای ترکیب‌های مختلف تنظیمات LoRa.

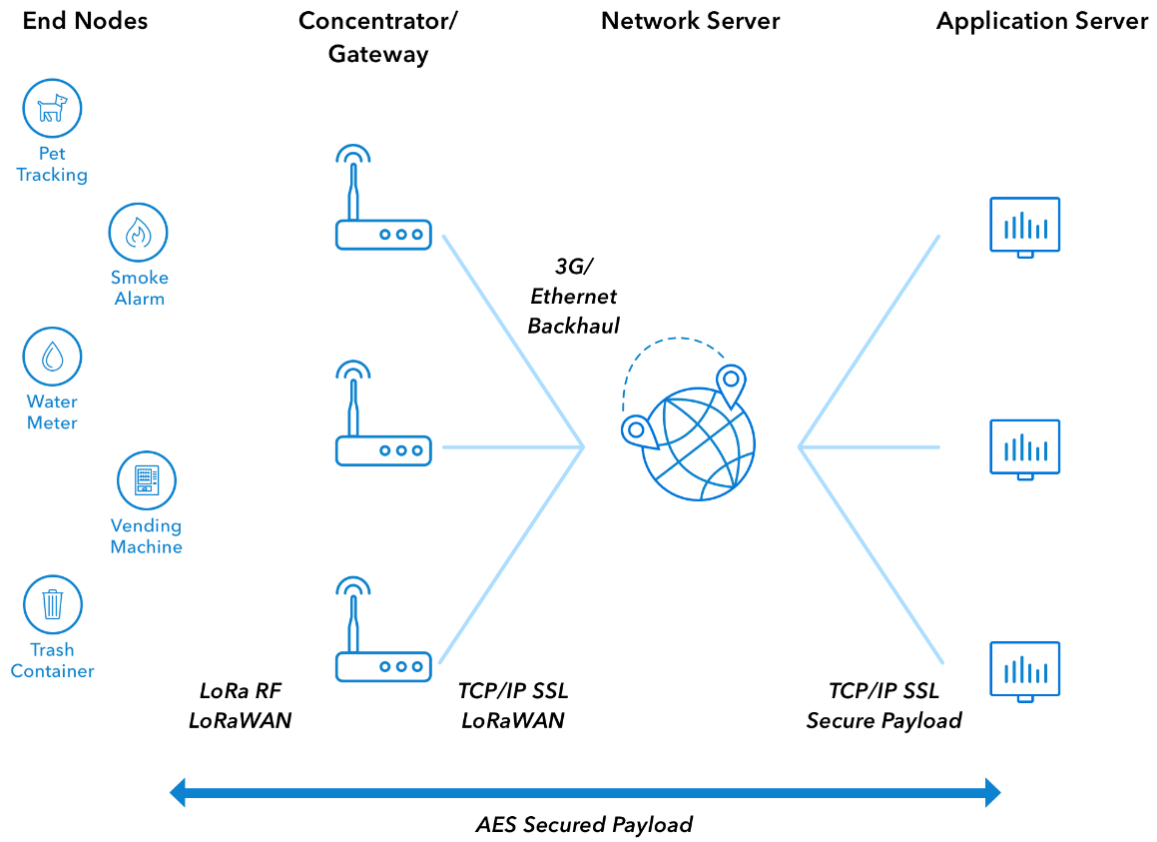
نحوه پیکربندی رجیسترهای درون ماژول LoRa: این کار می‌بایست از طریق پروتکل SPI انجام شود. نوشتن رجیسترها تنها در Sleep Mode امکان‌پذیر است. (مد کاری ماژول درون رجیستر RegOpMode پیکربندی می‌شود.)

۵.۳ انواع پیاده‌سازی

۱.۵.۳ پیاده‌سازی LoRa

در این روش تعدادی ماژول LoRa به صورت خصوصی با هم ارتباط دارند. پروژه مربوطه نیز بر اساس همین روش پیاده‌سازی شده است.

۲.۵.۳ LoRaWAN

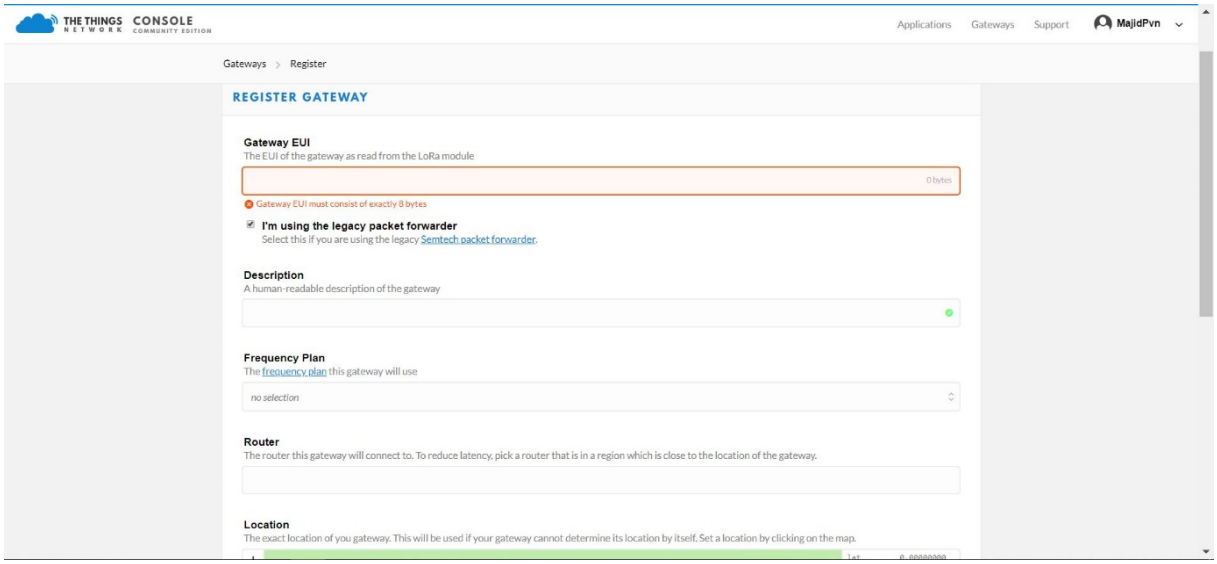


شکل ۱۰.۳: نمونه ای از شبکه LoRaWAN.

همانطور که در شکل ۱۰.۳ مشاهده می‌کنید، معماری شبکه LoRaWAN به صورت توپولوژی ستاره یا استار میباشد که در آن دروازه‌ها پیام‌ها را بین دستگاه‌های پایانی و سرور مرکزی شبکه انتقال می‌دهند. دروازه‌ها مانند یک پل نامرئی عمل می‌کنند و از طریق اتصالات استاندارد آی‌پی به سرور در شبکه متصل می‌شوند، به سادگی بسته‌های RF را به بسته‌های IP تبدیل می‌کنند و بالعکس. ارتباطات بی‌سیم از مزایای برد زیاد لایه‌فیزیکی LoRa بهره می‌برد و اجازه می‌دهد یک لینک تک کانون بین دستگاه پایانی و یا یک یا چند دروازه باشد. تمام حالت ها قادر به برقراری ارتباط دو طرفه هستند و پشتیبانی از گروه‌های آدرس‌دهی چندرسانه‌ای برای استفاده کارآمد از طیف در حین انجام وظایف مانند ارتقاء نرم‌افزار ها مانند Firmware (FOTA) یا سایر پیام‌های اپگرید وجود دارد. این شیوه پیاده سازی در اصطلاح، LoRaWAN نامیده می‌شود. همچنین در این نوع پیاده‌سازی می‌توان از ماژول LoRa به‌عنوان موقعیت‌یاب (بدون استفاده از GPS) استفاده کرد. به این صورت که تاخیر دریافت پیام توسط دروازه‌های اطراف ماژول سنجیده می‌شود و بر اساس الگوریتم خاص، موقعیت جغرافیایی ماژول به‌دست می‌آید.

اتحادیه LoRa[1]، در سال ۲۰۱۵ برای پشتیبانی از پروتکل LoRaWAN تشکیل شده است. این اتحادیه، بیش از ۵۰۰ عضو دارد که مهم ترین آنها، IBM، ACTILITY، MICROCHIP، ORANGE، CISCO، KPN، SWISSCOM، SIMTECH، BOUYGUES، TELECOM، SINGTEL، PROXIMUS می‌باشند.

گیتوی‌ها هم می‌تواند خودش سرور باشد و هم می‌تواند بعنوان Packet Forwarder عمل کند و دیتا را به سروری مانند TTN[2] که دارای پلتفرمی user friendly است، ارسال کند. برای این‌کار، ابتدا در سایت ثبت نام می‌کنیم و پس از ورود به اکانت خود، گیتوی مورد نظر را رجیستر می‌کنیم. از بخش CONSOLE -> GATEWAYS -> register gateway صفحه‌ای همانند شکل ۱۱.۳ ظاهر می‌شود.

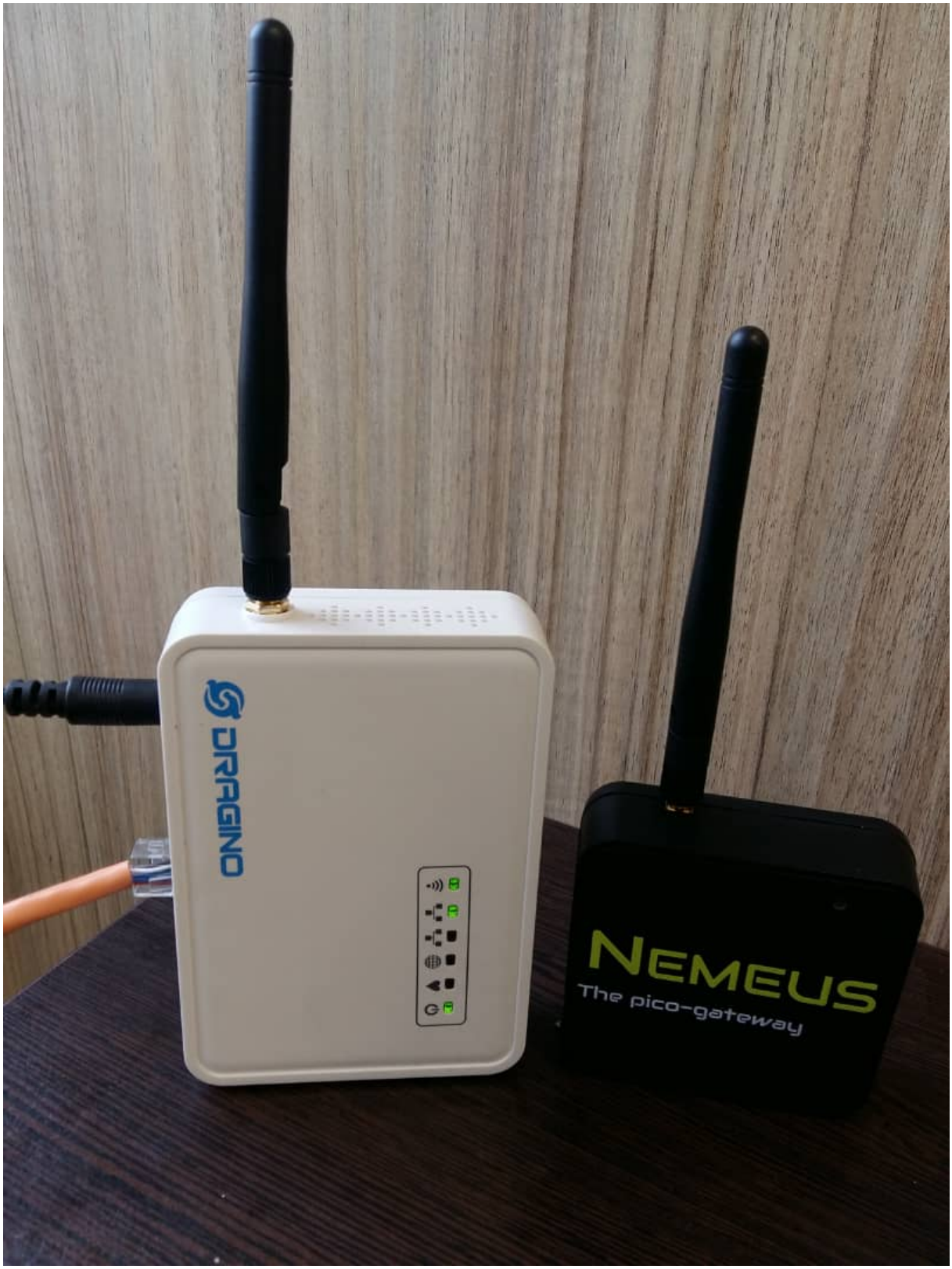


شکل ۱۱.۳: فرم رجیستر گیتوی در سایت TTN.

در اینجا باید تیک legacy packet forwarder را بزنیم و یک ایدی منحصر به فرد دلخواه وارد کنیم. قسمت بعدی مربوط به توضیحات است که می‌توان توضیحی از وضعیت، مدل یا هر اطلاعاتی که در مورد ها مورد نیاز است را وارد می‌کنیم. (این فیلد میتواند خالی باشد.). نمونه‌هایی از گیتوی‌ها را در شکل ۱۲.۳ مشاهده می‌کنید.

گیتوی سمت راستی یک گیتوی معمولی و نسبتا ارزان‌تری است اما گیتوی سمت چپی امکانات بیشتری دارد که در ادامه توضیح مختصری درباره آن خواهیم داد.

گیتوی DRAGINO-LG01: شرکت DRAGINO، یک شرکت فعال در کشور چین است که بر روی اینترنت اشیاء متمرکز است. این مدل می‌تواند بعنوان مودم اینترنت عمل کند. یعنی کابل شبکه به آن متصل شود و اینترنت را از طریق WIFI در محیط پخش کند. همچنین می‌توان به‌عنوان گیتوی LoRaWAN از آن استفاده کرد. بخشی از مدار داخلی آن از یک ماژول RFM95 شرکت MICROCHIP و یک میکروکنترلر ATMEGA328 تشکیل



شکل ۱۲.۳: دو نمونه از گیتوی‌ها با پشتیبانی از LoRaWAN.

شده است که کاربر می‌تواند برنامه موردنظر خود را در محیط آردینو[۱۸] بنویسد و درون میکروکنترلر آپلود کند. روش آپلود

[18] Arduino

فایل هم به دو صورت است هم می‌توان از پلتفرم داخلی آن فایل HEX را آپلود کرد و هم می‌توان از محیط آردوینو این کار را انجام دهد. لازم به ذکر است که این مدل فقط از کلاس‌های A و C پشتیبانی می‌کند.

۳.۵.۳ انواع کلاس در شبکه LoRaWAN

۱. کلاس : A کمترین توان، دستگاه‌های پایانی دو طرفه: کلاس پیش‌فرض باید توسط تمام Node[۱۹]ها پشتیبانی شود. ارتباط کلاس A همیشه توسط دستگاه پایانی آغاز می شود و کاملا ناهمگام است. هر گونه انتقال uplink[۲۰] می‌تواند در هر زمانی ارسال شود و همچنین توسط دو downlink[۲۱] کوتاه دنبال می‌شود.

Node ها این قابلیت را دارا خواند بود که توسط برنامه مشخص شده خود وارد حالت خواب بشوند. در حالت خواب با کم شدن قدرت مصرف دستگاه به حداقل میرسد و به همین علت هست که یک باتری برای دستگاه یک یا چند سال کار میکند. همچنین در این حالت دیگر نیازی به شبکه برای بیداری دوره ای دستگاه وجود ندارد. همین علت است که کلاس A کمترین مصرف را در این کلاس بندی پروتکل دارد. در این کلاس هر موقع که ارتباطی برقرار شود اجازه برقراری آن ارتباط به آن داده میشود. از آنجا که ارتباط downlink همیشه باید به یک انتقال uplink با برنامه ای که توسط برنامه نهایی دستگاه تعریف می شود پیروی می کند،اگر ارتباط downlink بعدی بیاید باید در سرور شبکه نگهداری شود تا کار پیام قبلی تمام شود.

۲. کلاس B - دستگاه های پایانی دو طرفه با تاخیر قطعی Downlink: علاوه بر کلاس A ، دستگاه ها در کلاس B با استفاده از موج‌های دوره‌ای، همگام با شبکه می‌شوند و در زمان‌های برنامه‌ریزی شده بازه پینگ های downlink را باز می کنند. با استفاده از این مورد شبکه توانایی ارسال ارتباطات downlink را با تاخیر قطعی بدست می آورد، اما همین امر باعث افزایش مصرف انرژی دستگاه پایانی میشود. زمان تاخیر تا ۱۲۸ ثانیه قابل برنامه ریزی است تا با برنامه های مختلف متفاوت باشد، و مصرف انرژی اضافی به اندازه کافی کم است که هنوز هم برای استفاده از باتری برای برنامه ها قابل اعتماد باشد. ۳. کلاس C - کمترین زمان تاخیر، دستگاههای پایانی دو طرفه: علاوه بر ساختار کلاس A که از uplink به دنبال دو مسیر downlink میباشد، کلاس C باعث کاهش زمان تأخیر در downlink می شود. چطور؟!!! با نگه داشتن Node در تمام زمان هایی که دستگاه چیزی انتقال نمی دهد (نیمه دو طرفه). بر اساس این، سرور شبکه می تواند در هر زمان بر اساس فرضیه گیرنده دستگاه پایانی باز شود، بنابراین هیچ تأخیری نمی تواند یک انتقال downlink را شروع کند. کلاس C مناسب برای برنامه های کاربردی است که نیاز به قدرت و مصرف مداوم دارند. برای دستگاه هایی که از باتری استفاده می‌کنند، حالت تعویض موقت بین کلاس A و

[۱۹] همان End-Device است که دیتا را از نقاط مختلف به گیتوی ارسال می‌کند.

[۲۰] انتقال دیتا از NODE به اپلیکیشن سرور

[۲۱] انتقال پیام از اپلیکیشن سرور به NODE

کلاس C امکان پذیر است و برای انجام وظایف متناوب مانند به روز رسانی سیستم‌عامل از طریق راه دور (OTA[۲۲]) مناسب است.

۶.۳ امنیت

شبکه LoRaWAN با تعدادی کلیدواژه امنیتی ارتباط برقرار می‌کند که طول تمامی آن‌ها ۱۲۸بیت می‌باشند.

Network Session Key (NwkSKey): برای ارتباط میان Node و سرور می‌باشد. این کلیدواژه، برای اعتبارسنجی پیام‌ها کاربرد دارد.

Application Session Key (AppSKey): این کلیدواژه جهت کدگذاری و دیکود کردن دیتای ارسالی (دریافتی) می‌باشد. به این صورت که کلیدواژه ای دلخواه خودمان در سمت Node وارد می‌کنیم و با این کلیدواژه، دیتای ارسالی کدگذاری می‌شود و در سمت سرور تنها با وارد کردن همان کلیدواژه قادر به دیکود کردن آن می‌باشیم. این بدان معنی است که هیچ‌کسی غیر از خودتان قادر به دسترسی به دیتای ارسالی نخواهد داشت.

هر دو کلیدواژه‌های بالا برای هر دستگاهی منحصر به فرد است. اگر بخواهید این کلیدواژه‌ها را از راه دور و به‌طور دینامیکی تغییر دهید، باید از مد OTAA استفاده کنید و اگر بخواهیم در هنگام ساخت دستگاه این اطلاعات را وارد کنیم و از راه دور امکان به تغییر آن وجود نداشته باشد، باید از مد ABP استفاده کنیم.

۷.۳ ماژول RN2483

این ماژول متعلق به شرکت MICROCHIP می‌باشد. همانطور که در شکل ۱۳.۳ مشاهده می‌کنید، مدار داخلی آن از یک چیپ SX1276 شرکت SEMTECH و یک میکروکنترلر PIC18LF تشکیل شده است. و دارای دو خروجی آنتن با فرکانس‌های ۴۳۳ مگاهرتز و ۸۶۸ مگاهرتز می‌باشد که بسته به نیاز کاربر، می‌توان از یکی از آن‌ها استفاده کند. که این دو از طریق پروتکل SPI باهم در ارتباط هستند و تعدادی کامند برای میکروکنترلر اش تعریف شده است که کاربر می‌تواند از طریق آن کامندها توسط پروتکل UART با باد ریت ۵۷۶۰۰، با SX1276 ارتباط برقرار کند و بطور دلخواه پیکربندی اش کند. این ماژول تنها از کلاس A پشتیبانی می‌کند.

[۲۲] OTA=Over The Air به طور خلاصه به معنی بروزرسانی دستگاه ها از طریق اینترنت یا موارد دیگر از طریق فواصل دور است.

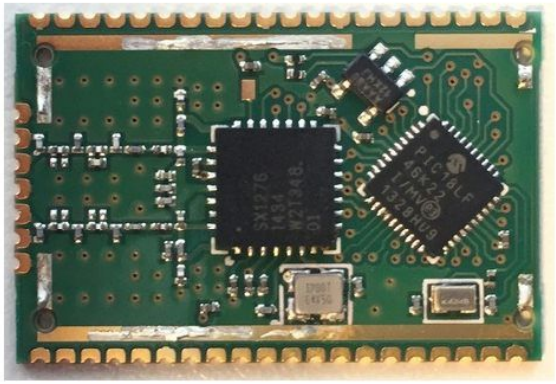
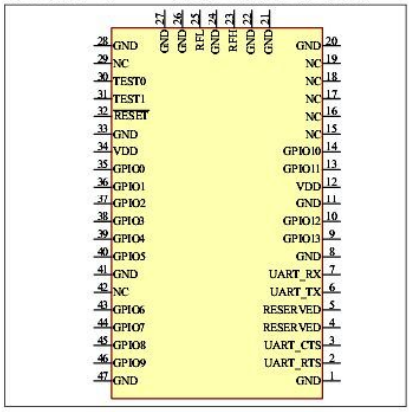


FIGURE 1-2: RN2483 PIN DIAGRAM



شکل ۱۳.۳: نمای بیرونی، مدار داخلی و پایه‌های خروجی ماژول RN2483.

فصل ۴

زبان برنامه‌نویسی پایتون

۱.۴ مقدمه

شکل ۱.۴: لوگوی زبان برنامه‌نویسی پایتون.

امروزه تعداد زبان های برنامه نویسی بسیار زیاد است و هر کدام کاربردهای مختلفی دارند. هر کدام از این زبان‌ها مزایا و معایب خودشان را دارند. یکی از زبان های برنامه نویسی مطرح بین برنامه نویسان پایتون[1] است که روز به روز به میزان محبوبیت آن اضافه می شود. از این زبان برنامه نویسی برای انجام کارهایی زیادی از جمله برنامه نویسی هوش

[1] Python

مصنوعی، توسعه وب، ساخت اپلیکیشن های موبایل و دسکتاپ استفاده می شود.

پایتون یک زبان برنامه نویسی سطح بالا تفسیر شده برای برنامه نویسی عمومی است. این زبان دارای یک فلسفه طراحی است که بر خواندن کد، به خصوص با استفاده از فضای خالی مهم استوار است. Python دارای یک سیستم نوع پویا و مدیریت حافظه خودکار است و پارادایم های چندگانه برنامه نویسی را پشتیبانی می کند. مفسر پایتون برای بسیاری از سیستم عامل ها در دسترس است.

به بیان فنی، پایتون یک زبان برنامه‌نویسی شی‌گرا[۲] و سطح بالا [۳] با معناشناسی[۴] پویای یکپارچه شده برای وب و ساخت و توسعه نرم‌افزارهای کاربردی[۵] است. این زبان برنامه‌نویسی در زمینه توسعه سریع نرم‌افزارهای کاربردی[۶] دارای جذابیت بالایی محسوب می‌شود زیرا دارای انواع پویا[۷] و انقیاددهنده پویا[۸] است.

پایتون، زبان برنامه‌نویسی نسبتا ساده‌ای محسوب می‌شود که یادگیری آن به دلیل دارا بودن نحو یکتایی که بر قابلیت خوانایی تمرکز و تاکید دارد آسان است. خواندن و ترجمه کدهای نوشته شده به زبان برنامه‌نویسی پایتون نسبت به دیگر زبان‌ها برای توسعه‌دهندگان[۹] ساده‌تر محسوب می‌شود. این موضوع به نوبه خود هزینه‌های نگهداری و توسعه برنامه‌های نوشته شده به این زبان را کاهش می‌دهد زیرا امکان همکاری تیم‌ها بدون مواجهه با موانع زبانی و وجود تجربیات کاری متفاوت در میان اعضای تیم را به دست می‌دهد.

علاوه بر این، پایتون از ماژول‌ها[۱۰] و بسته‌ها[۱۱] استفاده می‌کند، بدین معنا که برنامه‌های این زبان قابل طراحی به سبک ماژولار[۱۲] هستند و کدهای نوشته شده در یک پروژه در پروژه‌های گوناگون دیگر نیز قابل استفاده مجدد محسوب می‌شوند. هنگامی که کاربری ماژول یا بسته مورد نیاز خود را توسعه داد، خودش یا دیگر علاقمندان (در صورتی که کد در اختیار عموم قرار بگیرد) می‌توانند آن را برای استفاده در دیگر پروژه‌ها گسترش دهند. ایمپورت[۱۳] و اکسپورت[۱۴] کردن این ماژول‌ها نیز کار آسانی است.

[2] Object-Oriented
[3] High-Level
[4] Semantic
[5] Application software
[6] Rapid Application Development
[7] Dynamic Types
[8] Dynamic Binding
[9] Developers
[10] modules
[11] packages
[12] modular
[13] Import
[14] Export

۲.۴ تاریخچه

زبان برنامه‌نویسی پایتون در سال ۱۹۹۱ میلادی توسط یک برنامه‌نویس هلندی به‌نام خیدو فان روسوم (Guido van Rossum) ایجاد شد، این زبان با قابلیت‌های فراوان و شگفت‌انگیزی که دارد تحولی در دنیای برنامه‌نویسی به‌وجود آورده است، از توسعه‌ی برنامه‌های تحت وب گرفته تا ایجاد بازی‌های رایانه‌ای!

در سال‌های اخیر، پایتون مورد توجه بسیاری از برنامه‌نویسان بوده و روز به روز بر تعداد آن‌ها افزوده می‌شود، هرچند هنوز در ایران هنوز جایگاه خود را پیدا نکرده است، ولی می‌توان آینده‌ی روشنی برای آن تصور کرد، زیرا کاربردهای فراوانی داشته و در برنامه‌نویسی بسیاری از وبسایت‌های معتبر مورد استفاده قرار گرفته است.

پایتون یک زبان اسکریپتی است که کدهای آن در پلتفرم های لینوکس، ویندوز، مکینتاش، سیستم عامل‌های موبایل و حتی پلی‌استیشن قابل اجراست و به‌دلیل قابلیت‌های فراوانی که دارد، به یکی از زبان‌های مورد علاقه‌ی برنامه‌نویسان وب تبدیل شده و شرکت‌های بزرگی مثل گوگل، یاهو، اینستاگرام، ناسا، یوتیوب و... در سطح بالایی در حال استفاده از آن هستند.

۳.۴ ویژگی‌ها

اولین و مهمترین ویژگی این زبان «سادگی و آسانی» آن می باشد. این ویژگی آموزش پایتون را به یکی از بهترین گزینه ها برای انتخاب کسانی تبدیل کرده است که قصد دارند برای اولین بار به آموزش برنامه نویسی بپردازند. پایتون را خیلی سریع می توانید یاد بگیرید و به سرعت کدنویسی را با آن شروع کنید. اما اشتباه نکنید، سادگی پایتون به معنای ضعیف بودن آن نیست، بلکه python یک زبان برنامه نویسی مفسری، چندمنظوره و سطح بالاست که از شی گرایی و برنامه نویسی ساختار یافته به طور کامل پشتیبانی می کند.

از این زبان برنامه نویسی به طور گسترده در دنیا استفاده می‌شود و برای آن فرقی نمی کند که هدف شما از استفاده آن ایجاد وب اپلیکیشن و برنامه نویسی دسکتاپ است و یا حتی برنامه نویسی هوش مصنوعی و یادگیری ماشینی، این زبان به بهترین نحو از عهده تمام آن ها بر خواهد آمد و به جرات می‌توان ادعا کرد که در دیگر زمینه های برنامه نویسی شما را تنها نخواهد گذاشت. برای اینکه بدانید که مهمترین ویژگی های پایتون چیست که آن را به چنین زبان قدرتمندی تبدیل کرده است، باید با ساختار آن آشنا شوید.

پایتون زبانی قدرتمند و منعطف است که ساختاری بسیار منظم و کدنویسی در آن بسیار راحت است. این زبان برنامه نویسی تمرکز زیادی روی خوانایی کدها دارد، به طوری که هر کسی می‌تواند به راحتی کدهای این زبان را بخواند

و با آن ارتباط برقرار کند. همانطور که اشاره شد، پایتون از انواع روش‌های برنامه نویسی مثل پروسه‌ای، تابعی و البته شیء گرایی پشتیبانی می‌کند. همین مورد باعث شده که در پروژه‌های بزرگ و حساس روی پایتون حساب باز کنند.

مفسری بودن این زبان یعنی کدهای پایتون خط به خط توسط مفسر ترجمه می‌شوند. مفسر پایتون برای سیستم عامل های مختلف وجود دارد. وظیفه این مفسر تبدیل کدهای پایتون به زبان ماشین و سپس اجرای آن ها است. بسیاری از توزیع‌های لینوکس به طور پیشفرض پایتون و مفسر آن را دارند و شما بدون نیاز به انجام هیچ اقدامی می‌توانید کدنویسی را در این سیستم عامل شروع کنید. همچنین فایل نصبی نسخه ویندوز و سیستم عامل مک هم از سایت پایتون قابل دانلود است.

هر نرم افزار یا ابزار جدید که معرفی می‌شود، برای انتشار از یک پروانه یا مجوز (License) استفاده می‌کند. این مجوزها ماهیت برنامه را مشخص می‌کنند و به بقیه نشان می‌دهند که چطور باید از نرم افزار استفاده کرد. مثلا نرم افزارهای آزاد ۴ سطح آزادی را به برنامه نویس تضمین می‌دهند و لایسنس GPL این چهار سطح را در خود جا داده است. ۴ سطح آزادی که نرم افزارهای آزاد باید از آن پیروی کنند عبارتند از:

کاربران بتوانند نرم افزار را به هر منظوری استفاده کنند. کابران بتوانند نرم افزار را تغییر دهند و به سورس کد برنامه دسترسی داشته باشند. کاربران بتوانند یک نسخه از نرم افزار را بازنشر کنند. دیگر کاربران بتوانند نسخه‌های بازنشر شده را ویرایش کرده و دوباره انتشار دهند. پایتون تحت پروانه اختصاصی خودش به نام PSFL یا Python License Foundation Software منتشر می‌شود. این مجوز کاملا مشابه مجوز GPL است، اما در بند چهارم سخت‌گیری کمتری برای بازنشر نرم افزارها در نظر گرفته است. به همین دلیل توسعه دهندگان به راحتی برای این زبان کتابخانه های زیادی طراحی کرده اند و به لطف آن‌ها پایتون کتابخانه‌های مشهوری دارد که در توسعه و پیشرفت آن نقش مهمی ایفا کردند. این کتابخانه‌ها بودند که پایتون را محبوب‌تر کرده و پای این زبان را به حوزه‌های مختلفی مثل طراحی سایت، طراحی اپلیکیشن، هوش مصنوعی و داده کاوی باز کردند.

۴.۴ مزایا

حضور ماژول های شخص ثالث Python (PyPI) شامل چندین ماژول شخص ثالث است که باعث می شود Python بتواند با بسیاری از زبان ها و سیستم عامل های دیگر ارتباط برقرار کند.

کتابخانه های پشتیبانی گسترده: پایتون کتابخانه استاندارد بزرگی را ارائه می دهد که شامل موضوعات مختلف مانند پروتکل اینترنت ، عملیات رشته ، ابزارها و سرویس های وب و رابط های سیستم عامل است. بسیاری از کارهای برنامه نویسی پر کاربرد قبلاً در کتابخانه استاندارد نگاشته شده اند که باعث می شود طول کد به طور قابل توجهی کاهش

داده شود.

منبع باز: زبان پایتون تحت مجوز OSI تأیید شده است که استفاده و توزیع آن را آزاد می کند ، از جمله برای اهداف تجاری. علاوه بر این ، توسعه آن توسط جامعه ای انجام می شود که از طریق میزبانی کنفرانس ها، برای کد آن همکاری می کنند و ماژول های بی شماری را برای توسعه آن فراهم می کنند.

یادگیری سریع و آسان: ، پایگاه گسترده کاربران و توسعه دهندگان فعال باعث شده است تا یک بانک منابع اینترنتی غنی برای ترغیب توسعه و ادامه پذیرش زبان ایجاد شود.

ساختار داده های کاربر پسند: : پایتون دارای ساختار داخلی داده ها و فرهنگ نامه ها است که می تواند برای ساخت سریع داده های زمان اجرا سریع استفاده شود.

بهره وری و سرعت: پایتون دارای طراحی شی گرا تمیز است ، قابلیت های کنترل پیشرفته یک فرایند را فراهم می کند ، و توانایی های ادغام و پردازش متن دارد ، که همه اینها به افزایش سرعت و بهره وری آن کمک می کند. پایتون گزینه ای مناسب برای ساخت برنامه های پیچیده دارای چند پروتکل تحت شبکه محسوب می شود.

۵.۴ معایب

سرعت: پایتون کندتر از C یا C ++ است. پایتون یک زبان سطح بالا است ، برخلاف C یا C ++ به سخت افزار نزدیک نیست.

توسعه موبایل: پایتون یک زبان خیلی خوب برای توسعه موبایل نیست. این یک زبان ضعیف برای محاسبات موبایل است. به همین دلیل است که برنامه های اندکی در تلفن های همراه مانند Carbonnelle در آن ساخته شده اند.

مصرف حافظه: پایتون برای کارهای فشرده حافظه گزینه مناسبی نیست. به دلیل انعطاف پذیری انواع داده ها ، مصرف میزان حافظه پایتون نیز زیاد است.

دسترسی به پایگاه داده: پایتون با دسترسی به بانک اطلاعات محدودیت هایی دارد. در مقایسه با فن آوری های رایج مانند JDBC و ODBC ، لایه دسترسی به پایگاه داده Python کمی توسعه نیافته و بدوی است.

خطاهای زمان اجرا: برنامه نویسان پایتون در زمینه طراحی زبان چندین موضوع را ذکر کردند. از آنجا که این زبان به صورت پویا تایپ می شود ، به آزمایش بیشتری نیاز دارد و دارای خطاهایی است که فقط در زمان اجرا نشان می دهد.

فصل ۵

آشنایی با Git و GitHub

۱.۵ مقدمه

برای فهم گیت‌هاب[1]، ابتدا باید با گیت[2] آشنا شویم. گیت یک سیستم کنترل نسخه‌ی متن باز است که توسط خالق لینوکس، لینوس تروالدز، ساخته شد. گیت مانند سایر سیستم‌های کنترل نسخه از قبیل ساب‌ورژن، CVS و مرکوریال است؛ اما اساسا سیستم کنترل نسخه چیست؟ وقتی توسعه‌دهندگان چیز جدیدی مانند یک اپلیکیشن خلق می‌کنند، مدام تغییراتی در کدهای آن می‌دهند تا اولین نسخه‌ی رسمی و غیر بتا آماده‌ی انتشار شود. این روند در به‌روزرسانی برای نسخه‌های بعدی نیز ادامه دارد.

سیستم‌های کنترل نسخه تغییرات و بازنگری‌های توسعه‌دهندگان را در یک مخزن مرکزی ذخیره می‌کنند. با این کار همکاری بین توسعه‌دهندگان آسان می‌شود؛ به این شکل که هر توسعه‌دهنده می‌تواند نسخه‌ی جدید را دانلود کند، تغییرات را در آن اعمال و سپس آپلود کند. کلیه‌ی توسعه‌دهندگان قادر به مشاهده‌ی تغییرات جدید، دانلود آن‌ها و

[1] Github

[2] Git

مشارکت در پروژه خواهند بود.

در دنیای رایانه، هاب به تجهیزات سخت‌افزاری گفته می‌شود که از آن اشتراک‌گذاری شبکه با گجت‌های مختلف استفاده می‌شود و در نتیجه هاب به نوعی شبکه را گسترش می‌دهد. هاب در گیت‌هاب نیز چنین مفهومی دارد. توسعه‌دهندگان پروژه‌های خود را در گیت‌هاب ذخیره می‌کنند و از این طریق به شبکه‌ی عظیم توسعه‌دهندگان دنیا وصل می‌شوند. در گیت‌هاب این امکان وجود دارد که پروژه‌ای را از مخزن توسعه‌دهنده به مخزن خود کپی و در آن تغییرات اعمال کنید و سپس درخواست اعمال تغییرات را به صاحب پروژه بفرستید تا در پروژه‌ی اصلی اعمال کند. امکان پرسش و پاسخ نیز در این شبکه‌ی گیت فراهم است.

حالا که با مفاهیم گیت و گیت‌هاب آشنا شدیم به نحوه‌ی کار با این سرویس‌ها می‌پردازیم.

۲.۵ آموزش کار با گیت

۱.۲.۵ نیازهای بنیادی

قبل از هرچیز لازم است مواردی را نصب کنید. برای این‌کار نسخه‌ی متناسب با سیستم‌عامل خود را از اینجا دانلود و نصب کنید. اگر از لینوکس استفاده می‌کنید، از طریق پکیج منیجر نیز می‌توانید اقدام کنید.

در مرحله‌ی بعد، از آن‌جایی که در روند آموزش، یک مخزن شامل یک کد و یک README خواهیم ساخت، یک دایرکتوری برای آن در نظر بگیرید.

پس از آن، به عملیات معمولی نظیر init، کلون، کامیت و diff می‌پردازیم. البته، عملیات پیشرفته‌تری نیز وجود دارد که در مراحل اولیه نیازی به آن‌ها نخواهید داشت.

۲.۲.۵ راه‌اندازی یک مخزن (Repository)

قبل از شروع کار با گیت، باید یک مخزن پروژه راه‌اندازی کنید تا به کمک گیت آن را مدیریت کنید. ترمینال را باز کنید و در دایرکتوری پروژه‌ی خود دستور git init را وارد کنید.

با این کار یک دایرکتوری مخفی با نام .git در دایرکتوری پروژه‌ی شما ساخته خواهد شد. این دایرکتوری همان مسیری است که گیت دیتابیس و تنظیمات خود را در آن ذخیره می‌کند تا تغییرات پروژه‌ی شما را دنبال کند.

۳.۲.۵ کلون یا کپی کردن یک مخزن

راه دیگری برای دسترسی به مخزن وجود دارد که به کلونینگ مشهور است. درست مثل بررسی مخزن در سایر سیستم‌ها، اجرای کد git clone <repository URL> یک کپی کامل از مخزن مورد نظر به سیستم شما منتقل خواهد کرد. سپس، می‌توانید تغییرات دلخواه را در آن اعمال کنید. روند اعمال تغییرات ساخت تغییرات، اعمال موقت آن‌ها بدون وارد کردن در مخزن اصلی (staging)، اعمال تغییر در مخزن یا کامیت (commit) را شامل می‌شود.

جدید فایل افزودن در این مرحله می‌توان از زبان‌های برنامه‌نویسی مختلف مانند پایتون، روبی، Go یا هر زبان دیگری استفاده کرد. ما در این آموزش از زبان Python که معمول‌تر است استفاده می‌کنیم. فایلی به نام main.py را در دایرکتوری خود ایجاد و کد زیر را در آن وارد کنید.

```
print(”Hello World”)
```

بعد از ذخیره‌ی فایل، دستور git status را از ترمینال اجرا کنید. این دستور وضعیت حاضر مخزن کار شما را نشان می‌دهد. نتیجه‌ی به نمایش درآمده باید مشابه تصویر زیر باشد که در آن main.py به‌عنوان یک فایل untracked یا بررسی‌نشده فهرست شده است.

حالا طرز کار با چند فایل بدون اعمال تغییرات در مخزن را بررسی می‌کنیم. برای این‌کار یک فایل دوم به نام REDME.md بسازید. در این فایل جزئیاتی مثل نام پروژه، نام و نشانی ایمیل خود را وارد کنید. دستور لاتین git status را مجددا اجرا کنید. خواهید دید که این‌بار دو فایل به‌عنوان بررسی‌نشده فهرست شده‌اند.

حالا می‌خواهیم main.py را به‌اصطلاح استیج (stage) کنیم. فایلی که استیج می‌شود؛ یعنی تغییرات آن انجام گرفته اما هنوز در مخزن اصلی اعمال نشده است. برای این‌کار دستور main.py add git را وارد کنید. حالا، دستور وضعیت گیت (git status) را مجددا اجرا کنید. خواهید دید که main.py به‌عنوان فایلی جدید در بخش تغییرات در انتظار اعمال (changes to be commited) فهرست شده است و REDME.md در همان بخش Untracked files قرار دارد.

۴.۲.۵ تنظیمات

در این مرحله همه‌چیز آماده‌ی اعمال تغییرات یا commit است؛ اما قبل از این‌کار باید با تنظیمات ویرایشگر که گیت هنگام نوشتن پیام‌های کامیت مورد استفاده قرار می‌دهد آشنا شوید.

اگر از لینوکس استفاده می‌کنید گیت به‌طور پیش‌فرض، از برنامه‌هایی مانند pico، vi، vim یا emacs استفاده خواهد کرد. اگر با این برنامه‌ها آشنایی ندارید، ممکن است بخواهید آن‌ها را با نرم‌افزاری مثل Notepad، TextEdit یا Gedit عوض کنید. برای این‌کار دستور زیر را از ترمینال اجرا کنید:

```
git config –global core.editor <your app's name
```

در قسمت آخر کد به جای your app's name نام نرم‌افزار مورد نظر خود را وارد کنید.

تنظیمات دیگری مانند تغییر نام و ایمیل و چگونگی نمایش پیام کامیت نیز قابل انجام است. ما در این آموزش از vim به‌عنوان ادیتور استفاده می‌کنیم؛ اما شما می‌توانید انتخاب متفاوت خود را داشته باشید.

۵.۲.۵ اعمال اولین تغییر

کامیت در گیت شباهت بسیار زیادی با کامیت در سایر سیستم‌های کنترل نسخه مانند ساب‌ورژن دارد. روند کار به این شکل است که کار را آغاز می‌کنید و پیامی جهت توضیح اینکه دلیل تغییر انجام گرفته چیست وارد می‌کنید و فایل تغییر می‌یابد. پس دستور git commit را اجرا کنید. با این کار ویرایشگر به‌صورت خودکار باز می‌شود و الگوی زیر را نمایش می‌دهد.

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
#
# Initial commit
#
# Changes to be committed:
# new file: main.py
#
# Untracked files:
# REDME.md
#
```

با بررسی مداوم وضعیت تغییرات اعمال‌شده توسط دستور git status از شرایط مخزن خود آگاهی پیدا خواهید کرد. با این‌کار همواره خواهید دانست چه تغییری را اعمال کرده و چه تغییری را هنوز اعمال نکرده‌اید. یک پیام کامیت

خوب باید شامل دو بخش باشد؛ اول این‌که کوتاه و در حد ۷۲ کاراکتر باشد و به‌طور خلاصه تغییر اعمال‌شده را اعلام کند. دیگر این‌که دارای توضیحی بلندتر باشد که به‌طور مجزا در سطری دیگر جزئیات تغییر اعمال‌شده را توضیح دهد. البته مورد دوم اختیاری است و الزامی برای نوشتن آن وجود ندارد.

ما در این مرحله نیاز به نوشتن توضیح پیچیده‌ای نداریم؛ چرا که تنها یک فایل را به مخزن اضافه کرده‌ایم؛ اما چنان‌چه تغییری که اعمال می‌کنید دارای الگوریتم‌های پیچیده‌ای باشد، لازم است توضیحاتی در این بخش برای مطالعه‌ی سایر توسعه‌دهندگان بنویسید و آن‌ها را از چرایی اعمال این تغییر آگاه سازید. بنابراین، پیام ساده‌ی زیر را در ویرایشگر وارد و ذخیره کنید و خارج شوید.

“Adding the core script file to the repository”

حالا که تغییرات اعمال شدند. وضعیت گیت را مجددا بررسی کنید. خواهید دید که REDME.md همچنان در قسمت untracked قرار دارد.

۶.۲.۵ مشاهده‌ی تغییرات

در این مرحله از آن‌جایی که چند فایل در قسمت کنترل نسخه داریم و با دستورهای پایه آشنا شده‌ایم، به بررسی تغییرات می‌پردازیم. برای بررسی تغییرات اعمال‌شده در یک فایل از دستور git diff استفاده می‌کنیم. این دستور مشابه دستور diff Linux دو فایل را با هم مقایسه می‌کند و تغییرات فایل جدیدتر را نمایش می‌دهد.

در اینجا برای مشاهده‌ی تغییرات فایل REDME.md دستور git diff REDME.md را اجرا می‌کنیم. با این‌کار تغییرات جدیدترین نسخه نسبت به اولین نسخه به نمایش درمی‌آید.

به خاطر داشته باشید که به‌طور پیش‌فرض، دستور git diff تغییرات را نسبت به فایل اولیه نشان می‌دهد، نه فایل استیج‌شده. اگر می‌خواهید تغییرات استیج‌شده را مشاهده کنید، دستور git diff –cached REDME.md را اجرا کنید. این دستور چیزی شبیه کد زیر را به نمایش درخواهد آورد.

```
diff –git a/REDME.md b/REDME.md new file mode 100644 index 0000000..27c0a86
— /dev/null +++ b/REDME.md @@ -0,0 +1,5 @@ +# Simple Git Project + +##
Authors + +Matthew Setter <matthew@maltblue.com>
```

در کد نمایش داده‌شده به پنج خط آخر دقت کنید. قبل از هر سطر یک علامت + وجود دارد. این علامت نشانگر افزودن چیزی به فایل است. در اینجا ما فقط اضافه کرده‌ایم؛ اما اگر چیزی حذف کرده بودیم علامت منفی (-) نمایش داده می‌شد.

۷.۲.۵ نکات مهم درباره‌ی استیجینگ یا ایندکس

اگر مبتدی هستید، یکی از گیج‌کننده‌ترین قسمت‌ها برای شما محیط استیجینگ و رابطه‌ی آن با کامیت خواهد بود.

کامیت در واقع ثبت سوابق تغییرات فایل‌ها نسبت به آخرین تغییر اعمال‌شده است. یعنی شما تغییری در مخزن خود اعمال می‌کنید و به کیت می‌گویید آن فایل‌ها را در یک کامیت ثبت سابقه کند.

کامیت‌ها ماهیت پروژه‌ی شما را در مراحل مختلف حفظ می‌کنند تا هر زمان که نیاز شد بتوانید به مرحله‌ی قبل برگردید.

اما چطور به گیت می‌گویید کدام فایل‌ها را در کامیت قرار دهد؟ اینجا است که استیجینگ یا ایندکس ایفای نقش می‌کند. برای اضافه کردن یک فایل در کامیت، ابتدا باید آن را به محیط استیجینگ اضافه کنید. برای انجام این کار می‌توانید از دستور git add <filename> استفاده کنید. پس از اینکه که فایل‌های مورد نظر خود را با این دستور به محیط استیجینگ اضافه کردید، می‌توانید با دستور git commit آن‌ها را به‌عنوان کامیت در مخزن اعمال کنید.

شاخه‌بندی یا برنچینگ تا این‌جا با نحوه‌ی شروع کار و اعمال تغییرات و بررسی آن‌ها آشنا شدیم. حالا با مفهوم پیشرفته‌تری به نام شاخه‌بندی آشنا می‌شویم. وقتی به‌طور تیمی روی یک نرم‌افزار کار می‌کنیم. آزمون و خطاهای هر برنامه‌نویس روی شاخه‌ی اصلی کدهای یک برنامه ممکن است دردسرساز شود. گیت این اجازه را به شما می‌دهد که شاخه‌ی اختصاصی خود را داشته باشید. در این حالت، وقتی روی ساخت یک ویژگی کار می‌کنید، آزمایش‌های شما صدمه‌ای به شاخه‌ی اصلی نمی‌زند و می‌توانید وقتی به نتیجه رسیدید، مجموعه‌ی تغییرات اعمال‌شده را با شاخه‌ی اصلی تلفیق یا merge کنید.

تا این بخش از آموزش در حال کار روی شاخه‌ی اصلی یا مستر برنچ بوده‌ایم. شاخه‌ی اصلی در واقع همان شاخه‌ای است که به‌صورت پیش‌فرض راه‌اندازی گیت با آن آغاز می‌شود. اکنون قصد داریم یک شاخه به نام development (به معنی توسعه) راه‌اندازی کنیم. از ترمینال دستور git checkout -b develop را اجرا کنید تا شاخه‌ای به نام لاتین develop ساخته شود. اجرای این دستور علاوه بر ساخت شاخه‌ی مذکور بررسی آن را نیز اعمال می‌کند. این شاخه در ابتدا یک کپی از شاخه‌ی اصلی است. یعنی اگر دستور git status را اجرا کنید همان دو تغییر اعمال‌شده در REDME.md را مشاهده خواهید کرد. حالا فرض کنید می‌خواهیم همین دو کامیت را در شاخه‌ی اصلی تلفیق کنیم.

برای انجام این‌کار ابتدا باید مشخص کنید که قصد تلفیق تغییرات موجود در کدام شاخه را دارید.

پس، دستور git checkout master را اجرا کنید. حالا باید تغییرات را از شاخه‌ای که در حال کار روی آن بوده‌اید در این شاخه تلفیق کنید. برای این‌کار دستور git merge develop را اجرا کنید.

وقتی کار تمام شد، پیامی مبنی بر تغییر فایل‌ها و خلاصه گزارشی از آن تغییرات نشان داده خواهد شد.

۳.۵ آموزش کار با گیت‌هاب

پس از آشنایی مقدماتی با گیت وقت آن رسیده است که به گیت‌هاب بپردازیم. همان‌طور که گفته شد، گیت‌هاب چیزی فراتر از یک مخزن پروژه است. این سرویس یکی از نسخه‌های گرافیکی گیت به شمار می‌رود. در واقع، اگر کار با گیت را بلد باشید به‌ندرت در کار با گیت‌هاب به مشکل برخواهید خورد. مسئله‌ی دیگر این است که ممکن است نخواهید پروژه‌ی خود را با دیگران به اشتراک بگذارید. در این حالت استفاده از گیت و ذخیره‌ی تغییرات به‌صورت لوکال بهترین گزینه است؛ همچنین در استفاده از گیت‌هاب شما ملزم به داشتن اتصال اینترنت هستید؛ ولی در کار با گیت چنین الزامی وجود ندارد. با تمام این اوصاف، ممکن است بخواهید صفر تا صد پروژه‌ی خود را در گیت‌هاب انجام دهید؛ چرا که کار با گیت مرارت‌های خاص خود را دارد. برای مثال باید کل منبع را دانلود کنید، سپس وایرایش‌های خود را در قالب یک پچ تهیه کنید و به طریقی مثل ایمیل به سازنده‌ی اصلی بدهید و او این پچ را که از سازنده‌ی آن اطلاع دقیقی ندارد بررسی و در صورت امکان استفاده کند. خواهید دید که در گیت‌هاب به‌عنوان یک شبکه‌ی برخط و متن باز هیچ‌یک از این مرارت‌ها وجود نخواهد داشت؛ اما قبل از شروع کار با گیت‌هاب نیاز به آشنایی با مفاهیم خاصی است که در زیر به آن‌ها می‌پردازیم.

۱.۳.۵ ساخت مخزن یا Repository

مخزن یا Repository که به اختصار به آن repo نیز گفته می‌شود می‌تواند شامل فولدر، فایل، تصویر، ویدیو و هر آنچه پروژه‌ی شما به آن نیاز دارد باشد. گیت‌هاب در ابتدای ساخت پروژه امکان افزودن README و سایر موارد مانند لایسنس را در اختیار می‌گذارد.

مخزن اول شما با نام hello-world می‌تواند مکانی برای ذخیره کردن ایده‌ها، منابع یا حتی اشتراک‌گذاری و بحث در مورد چیزهای مختلف باشد.

برای ساخت یک مخزن جدید در گوشه‌ی بالا سمت راست و نزدیک به آواتار یا نماد کاربری شما، روی + و سپس New repository کلیک کنید. توضیح کوتاهی بنویسید. در صورت تمایل به اضافه کردن README گزینه‌ی Initialize this repository with a README را انتخاب کنید. رو Creat repository کلیک کنید.

۲.۳.۵ ساخت شاخه یا Branch

برنچینگ روشی برای کار همزمان روی نسخه‌های مختلف یک مخزن است. به‌طور پیش‌فرض مخزن شما یک شاخه به نام master دارد که شاخه‌ی اصلی به‌حساب می‌آید. از شاخه‌ها برای آزمون و خطا و ویرایش کدها قبل از اعمال تغییرات در شاخه‌ی اصلی استفاده می‌شود. وقتی یک شاخه بر پایه‌ی شاخه‌ی اصلی می‌سازید، یک کپی از آن با آخرین تغییراتی که تا آن لحظه اعمال‌شده در شاخه‌ی جدید ایجاد می‌شود. اگر زمانی که شما روی شاخه‌ی خود کار می‌کنید کسی کامیتی به شاخه‌ی اصلی اضافه کند می‌توانید آن را در شاخه‌ی خود اعمال کنید. برای ساخت یک شاخه‌ی جدید به مخزن جدیدی که با نام hello-world ساخته‌اید بروید. روی فهرست بازشونده‌ی موجود در بالای فهرست فایل‌ها که روی آن نام شاخه نوشته شده است کلیک کنید. در تکست‌باکس بازشده نام شاخه‌ی جدید، مثلا readme-edits را وارد کنید. روی دکمه‌ی Create branch کلیک کنید یا دکمه‌ی Enter را در کیبورد خود بزنید. حالا دو شاخه دارید؛ یکی master و دیگری readme-edits که کاملا شبیه به هم هستند؛ البته تا زمانی که تغییری در هیچ‌یک اعمال نکرده‌ایم.

۳.۳.۵ ایجاد تغییرات و اعمال آن‌ها

حالا که شاخه‌ی جدیدی با محتویات یکسان با شاخه‌ی اصلی داریم، بدون ترس از خرابکاری ویرایش‌های خود را آغاز می‌کنیم. در گیت‌هاب به تغییرات اعمال‌شده کامیت (commit) می‌گویند. هر کامیت یک پیام کامیت نیز به همراه دارد که توضیح کوتاهی در رابطه با دلیل اعمال آن تغییر است. پیام‌های کامیت به سایر مشارکت‌کنندگان در پروژه این امکان را می‌دهد که متوجه شوند شما چه تغییری را به چه دلیل اعمال کرده‌اید. برای اعمال یک تغییر روی فایل REDME.md کلیک کنید. روی آیکون مداد در گوشه‌ی بالا سمت راست کلیک کنید تا بتوانید آن را ویرایش کنید. در ویرایشگر، کمی درباره‌ی خود بنویسید. یک پیام کامیت برای توصیف تغییرات خود بنویسید. روی دکمه‌ی Commit changes کلیک کنید.

تغییرات ایجادشده در فایل README تنها در شاخه‌ی readme-edits اعمال شده‌اند. حالا این شاخه دارای تغییراتی نسبت به شاخه‌ی master است.

۴.۳.۵ ایجاد درخواست اعمال تغییرات یا pull request

حالا برای اعمال این تغییرات در شاخه‌ی اصلی باید درخواست آن را ایجاد کنید. pull request هسته‌ی تمام همکاری‌ها در گیت‌هاب را تشکیل می‌دهد. با ارسال چنین درخواستی، شما از صاحب شاخه می‌خواهید که تغییرات

انجام‌شده توسط شما را بررسی و در صورت صلاحدید به شاخه‌ی اصلی اضافه کند. درخواست‌های اعمال تغییرات تغییرات بین دو شاخه را نمایش می‌دهند. تغییرات، حذف و اضافه‌ها در رنگ‌های سبز و قرمز نشان داده می‌شوند. با استفاده از سیستم منشن در گیت‌هاب می‌توانید در سیستم پیام pull request خود از سایر افراد یا تیم‌ها درخواست کنید که درباره‌ی ویرایش‌های شما نظر بدهند. شما می‌توانید درخواست اعمال تغییرات را برای خود نیز ارسال کنید. یعنی وقتی تغییری را به‌طور کامل در شاخه‌ی فرعی انجام دادید، می‌توانید برای تلفیق آن در شاخه‌ی اصلی از pull request استفاده کنید.

۵.۳.۵ پذیرش درخواست تلفیق تغییرات اعمال‌شده

در این گام پایانی، نحوه‌ی تلفیق تغییرات شاخه‌ی فرعی در شاخه‌ی master را بررسی می‌کنیم. روی دکمه‌ی سبزرنگ Merge pull request کلیک کنید تا تغییرات شما در شاخه‌ی اصلی اعمال شوند.آموزش گیت‌هاب روی Confirm merge کلیک کنید. حالا که تغییرات را اعمال کرده‌اید، می‌توانید با استفاده از دکمه‌ی Delete branch شاخه‌ی فرعی را حذف کنید.

گ‌ی‌ت‌ه‌اب در رای‌ج اص‌طل‌اح‌ات از ت‌ع‌دادی کد : (Code) حالت نمایشی که به‌صورت پیش‌فرض در آن قرار دارید و فایل‌های پروژه به شما نمایش داده می‌شوند.

مسائل : (Issues) چنان‌چه شما یا هم‌تیمی‌های شما بخواهند مشکلی را در نرم‌افزار گزارش کنند، یا درخواست افزودن قابلیت یا مسائلی این‌چنینی را مطرح کنند، از این گزینه استفاده می‌کنند.

ویکی : (Wiki) امکانی است برای ثبت جزئی‌تر پروژه نسبت به آن‌چه در REDME.md می‌آید.

ضربان : (Pulse) خلاصه‌ای از آمار پروژه شامل مسائل مطرح‌شده، حل‌شده و حل‌نشده که نشانگر میزان فعال بودن پروژه است.

نمودارها : (Graphs) پیشرفت پروژه در طول زمان شامل روزهای پرکار و زمان‌هایی که پروژه رها شده و بی‌تغییر مانده است نشان می‌دهد.

۴.۵ آموزش انتقال پروژه از گیت به گیت‌هاب

حالا قصد داریم پروژه‌ی کوچکی را که در گیت روی آن کار کرده بودیم در گیت‌هاب بارگذاری کنیم. برای این‌کار ابتدا نیاز به ساخت یک حساب کاربری در گیت‌هاب دارید. توجه داشته باشید در روند ساخت حساب کاربری در گیت‌هاب

پس از وارد کردن نام کاربری، ایمیل و پسورد، دو تعرفه پیش روی شما قرار می‌گیرد. در گزینه‌ی اول استفاده از گیت‌هاب رایگان خواهد بود اما نمی‌توانید پروژه‌ی محرمانه بسازید. طبعا گزینه‌ی دوم پولی و با امکان ساخت پروژه‌ی محرمانه یا خصوصی است.

پس از ورود به حساب کاربری خود برای بارگذاری پروژه روی علامت مثبت موجود در بالا گوشه‌ی راست کلیک کنید و در فهرست بازشده New repository را برای راه‌اندازی مخزن جدید انتخاب کنید. در این مرحله فرم مربوط به ساخت پروژه‌ی جدید ظاهر خواهد شد.

در قسمت Repository name یک نام برای مخزن خود وارد کنید. این نام می‌تواند first-project به معنی اولین پروژه باشد. می‌توانید توضیحی نیز در خصوص آن ذکر کنید. مثلا my first Github project به معنی اولین پروژه‌ی من در گیت‌هاب. حالت پروژه با توجه به نوع کاربری شما، public خواهد بود. نهایتا تیک inilialize this repository with a README را بزنید و دو قسمت دیگر را به همان شکل روی رهاNone کنید. حالا روی Create repository کلیک کنید.

در این مرحله به صفحه‌ی راه‌اندازی سریع هدایت خواهید شد. این صفحه امکان بارگذاری پروژه‌ی گیت را به ما خواهد داد. برای انجام این‌کار اولین سطر زیر or push an existing repository from the command line... را کپی کنید و آن را در ترمینال یعنی همان‌جایی که تا کنون روی آن کار می‌کردیم پیست کنید. این‌کار به ما خروجی نخواهد داد. حالا همین کار را برای سطر دوم انجام دهید. در این مرحله تغییرات ما به گیت‌هاب منتقل می‌شود و خروجی مشابه آن‌چه در تصویر زیر می‌بینید خواهد بود. حالا در مرورگر خود صفحه‌ی گیت‌هاب را ریفرش کنید. فایل‌های REDME.md و main.py به‌عنوان اولین فایل‌های پروژه نمایش داده می‌شوند و محتویات REDME.md در پایین صفحه نمایان می‌شود.

فصل ۶

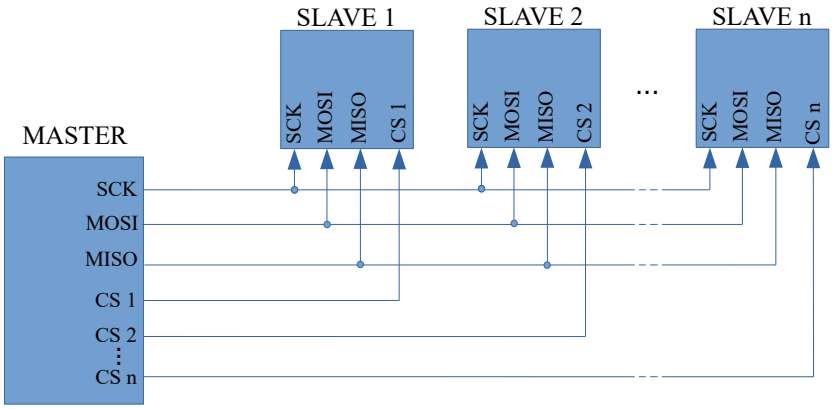
پروتکل‌های ارتباطی

۱.۶ مقدمه

قطعات الکترونیک نیز همانند انسان‌ها برای ارتباط با یکدیگر باید از یک زبان واحد قابل‌فهم برای هر دو طرف استفاده کنند که به این زبان‌ها پروتکل‌های ارتباطی گفته می‌شود. در این پروژه نیز همانند اکثر پروژه‌های الکترونیکی از تعدادی از این پروتکل‌ها برای برقراری ارتباط بین قطعات الکترونیکی مختلف (سنسورها، فرستنده-گیرنده‌های رادیویی و میکروکنترلر) استفاده شده است. در زیر اطلاعات کلی هر یک از پروتکل‌های ارتباطی استفاده‌شده در این پروژه آمده است.

۲.۶ پروتکل SPI

این پروتکل یک رابط ارتباط سریالی سنکرون[1] (همزمان) چهار سیم است که برای ارتباط بین تنها یک Master و چندین Slave می‌تواند مورداستفاده قرار گیرد. خطوط ارتباطی بین Master و Slave شامل خط سیگنال کلاک (SCLK[2])، خط ارسال داده از Master به Slave (MOSI[3])، خط ارسال داده از Slave به Master (MISO[4]) و خط انتخاب Slave (SS[5] یا CS[6]). در حالت Full Duplex به ازای هر Slave معمولاً نیاز به یک پین انتخاب‌گر (SS یا CS) مجزا در سمت Master خواهد بود. نحوه اتصال دستگاه‌های Slaveبه Master در شکل ۱.۶ آمده است.



شکل ۱.۶: نحوه اتصال Slaveها به Master در حالت Full Duplex

سیگنال کلاک همواره توسط Master تولید و به ازای هر سیگنکال کلاک یک بیت داده منتقل می‌شود. ازاین‌رو ارسال داده از Slave به Master به‌طور تصادفی و در هرلحظه ممکن نیست و تنها Slave می‌تواند در لحظاتی که Master درخواست می‌کند و سیگنال کلاک را تولید می‌کند داده را برروی خط MISO ارسال کند. البته به دلیل وجود خطوط مجزای ارسال و دریافت داده، ارسال و دریافت همزمان داده نیز ممکن است. همچنین در این پروتکل دستگاه‌های Slave به‌تنهایی نمی‌توانند با یکدیگر ارتباط برقرار کنند.

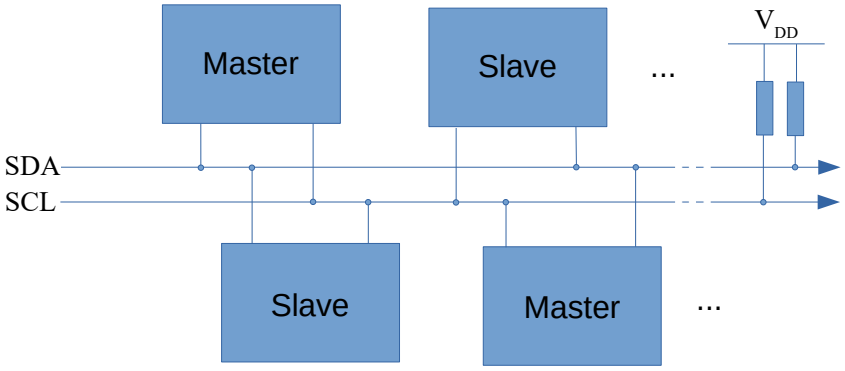
به جهت ارسال داده، Master از طریق قراردادن خط SS (یا CS) در حالت low انتخاب می‌کند که با کدام Slave می‌خواهد ارتباط برقرار کند. وضعیت خط را تا پایان فرآیند ارسال و دریافت داده در همین حالت (low)

[1] synchronous
[2] Serial Clock
[3] Master Out Slave In
[4] Master In Slave Out
[5] Slave Select
[6] Chip Select

نگهمیدارد. در همین حال سیگنال کلاک را روی خط SCLK متناسب با طول داده ارسالی قرار می‌دهد (ماکسیموم فرکانس کلاک بستگی به ماکسیموم مقدار قابل‌شناسایی توسط Slave دارد) و داده ارسالی را نیز متناسب با کلاک روی خط MOSI قرار می‌دهد (قرار دادن داده روی لبه بالارونده یا پایین‌رونده و همچنین ارسال از بیت MSB یا LSB بستگی به Slave دارد)؛ اگر قرار باشد Slave دیتایی در پاسخ به دیتای دریافتی ارسال کند (این امر باید از قبل مشخص باشد چراکه همان‌طور که گفته شد مسئول تولید کلاک فقط Master است) باید Master کلاک‌هایی متناسب با دیتای مورد انتظار برای دریافت را روی خط SCLK تولید کند و دیتای مربوطه را از روی خط MISO بخواند.

۳.۶ پروتکل I^2C

پروتکل I^2C رابط سریالی سنکرون دو سیم است که علاوه بر پشتیبانی از چندین Slave قابلیت پشتیبانی از چند Master را نیز دارا می‌باشد (البته Masterها قادر به برقراری ارتباط با یکدیگر نیستند و از خطوط بأس به‌طور همزمان نمی‌توانند استفاده کنند). در این پروتکل برای ارتباط از تنها دو خط سیگنال کلاک (SCL[۷]) و سیگنال دیتا (SDA[۸]) استفاده می‌شود. نحوه اتصال دستگاه‌ها در این پروتکل در شکل ۲.۶ آمده است.



شکل ۲.۶: نحوه اتصال در پروتکل I^2C

در این پروتکل نیز همانند پروتکل SPI سیگنال کلاک توسط Master کنترل می‌شود. اعلام شروع ارسال داده با بالا نگهداشتن خط SCL و پایین کشیدن خط SDA انجام می‌شود. اگر دو Master به‌طور همزمان قصد تبادل داده با Slaveها را داشته باشند هرکدام که زودتر SDA را پایین بکشید اجازه استفاده از خط را دارد و Master دیگر باید تا پایان تبادلات Master اول صبر کند.

[7] Serial Clock Line
[8] Serial Data Line

برای انتقال داده پس از اعلام شروع ارسال توسط Master، ۷ بیت آدرس Slave که قرار است با آن ارتباط برقرار شود ارسال می‌شود و پس‌ازآن یک بیت read/write ارسال می‌شود تا Slave از قصد Master برای خواندن (سطح منطقی ۱) یا نوشتن (سطح منطقی ۰) مطلع شود (امکان ارتباط با Slave با آدرس‌های ۱۰ بیتی نیز وجود دارد که نحوه ارسال آن کمی متفاوت است ولی فرآیندهای دیگر در دریافت و ارسال داده‌ها یکسان است). در تمام طول ارسال و دریافت داده سیگنال کلاک نیز به‌طور منظم روی خط SCL تولید می‌شود. بعد از ارسال آدرس توسط Master، Slaveها آدرس ارسالی را با آدرس خود مقایسه کرده و درصورتی‌که با آن مطابقت داشت بیت تصدیق (ACK) را با پایین کشیدن SDA تا قبل از کلاک نهم ارسال می‌کنند. اگر بیت تصدیق در این زمان ارسال نشود و SDA در سطح high بماند ارسال داده متوقف می‌شود چراکه عدم دریافت بیت تصدیق نشان‌دهنده عدم وجود Slave موردنظر روی خط و یا عدم توانایی Slave در رمزگشایی داده ارسالی است.

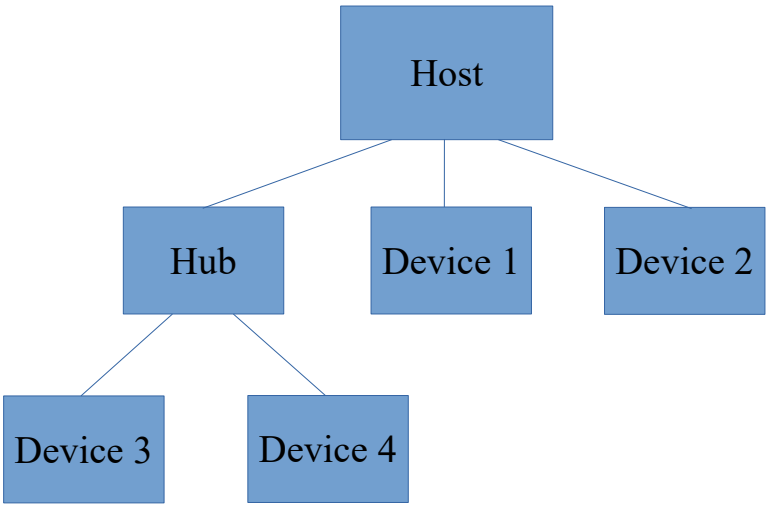
پس از ارسال آدرس و دریافت تصدیق از سمت Slave با توجه به بیت read/write ارسالی، داده توسط Slave (درحالت خواندن) و یا Master (در حالت نوشتن) روی خط SDA قرار داده می‌شود. بعد از ارسال هر ۸ بیت داده نیز لازم است بیت تصدیق دریافت داده‌ها (ACK) از طرف دریافت‌کننده داده‌ها ارسال شود. بعد از ارسال یا دریافت تمام داده‌ها باید وضعیت توقف اعلام شود تا دیگر Masterها بتوانند پس‌ازآن از خط استفاده کنند. اعلام وضعیت توقف با یک تغییر وضعیت SDA از سطح منطقی ۰ به ۱ و پس‌ازآن یک تغییر وضعیت از سطح منطقی ۰ به ۱ روی خط SCL انجام می‌شود.

۴.۶ پروتکل USB

پروتکل USB یک پروتکل ارتباطی سریال آسنکرون[9] (ناهمزمان) دو سیم است که ارتباطی بین چندین دستگاه جانبی یا Device و دستگاه اصلی یا Host را فراهم می‌کند. USB توسط یک گروه متشکل از شرکت‌های فعال در زمینه رایانه و الکترونیک ایجاد شد که هدف ساخت پروتکل سریال همه‌منظوره برای اتصال لوازم جانبی به رایانه را داشتند. به دلیل همه‌منظوره بودن USB استفاده از این پروتکل دارای جزئیات بسیار زیادی است و برخلاف دو پروتکل قبلی راه‌اندازی این پروتکل ممکن است زمان‌بر باشد. در این پروژه از این پروتکل برای ارتباط دستگاه سمت ایستگاه با رایانه و همچنین تأمین تغذیه آن استفاده می‌شود. نحوه اتصال دستگاه‌ها به Host در شکل ۳.۶ آمده است.

به‌طورکلی برای ارتباط دستگاه‌های جانبی با دستگاه اصلی، دستگاه جانبی نیازمند دارا بودن Device Class مشخص و شناخته‌شده برای دستگاه اصلی است. پیاده‌سازی دیوایس کلاس‌های اختصاصی به خاطر پیچیدگی‌هایی

[9] Asynchronous



شکل ۶.۳: نحوه اتصال دستگاه‌های جانبی به دستگاه اصلی در پروتکل USB

که این پروتکل دارد امری بسیار زمان‌بر خواهد بود به همین دلیل معمولاً استفاده از دیوایس کلاس‌های آماده (و لایبرری‌های متناظر با آن) برای ساخت دستگاه‌های جدید مناسب‌ترین روش ممکن است (این کار همچنین امر دریافت گواهی‌های سخت‌افزاری و نرم‌افزاری USB را نیز تسهیل می‌کند).

در این پروتکل ۴ نوع انتقال داده وجود دارد. انتقال Control که تنظیمات و مشخصات دستگاه و دستورات را منتقل می‌کند. انتقال Isochronous که برای انتقال داده‌هایی که زمان‌بندی در آنها اهمیت دارد نظیر صدای میکروفون و تصویر وب‌کم مورداستفاده قرار می‌گیرد. انتقال Bulk که برای ارسال داده‌های حجیم‌تر به‌صورت یکجا مورداستفاده قرار می‌گیرد مانند انتقال تصویر برای پرینت و یا اسکن و یا انتقال داده‌های ذخیره‌شده روی یک فلش مموری. انتقال Interrupt که داده‌هایی با حجم بسیارکم و اهمیت زیاد را جابجا می‌کند که معمولاً توسط موس، کیبرد و جواستیک (Joystick) و عموماً در کلاس HID[10] مورداستفاده قرار می‌گیرد.

در این پروتکل لازم است دستگاه اصلی از تمامی جزئیات کارکرد و مشخصات دستگاه جانبی مطلع شود. مشخصات کلی دستگاه مثل ورژن USB مورداستفاده، شناسه ونام سازنده، شناسه ونام دستگاه و مشخصات کلاس دستگاه در بخشی با عنوان Device Descriptor با فرمتی خاص و مشخص به اطلاع رایانه می‌رسد. بخش دیگری با عنوان Configuration Descriptors بیان‌کننده نحوه تغذیه و ماکسیموم توان موردنیاز دستگاه متصل به USB است. بخش Interface Descriptors نیز بیان‌کننده ویژگی‌های کلاس دستگاه است و شامل چندین Endpoint Descriptors می‌شود که برای انجام آن ویژگی‌ها موردنیاز هستند. برای هر Endpoint در سمت دستگاه جانبی ازنظر سخت‌افزاری معمولاً یک رجیستر یا مموری تعریف می‌شود که دیتا با توجه به مشخصات هر Endpoint از آن خوانده می‌شود یا در آن نوشته می‌شود هر Endpoint فقط مسئول یکی از اعمال خوانده شدن یا نوشته شدن است

[10] Human interface device

و برای هر دو عمل خواندن و نوشتن به دو Endpoint نیاز است. ماکسیموم می‌توان ۱۶ Endpoint تعریف کرد، Endpoint صفر رزرو و برای مشخصات کنترلی استفاده می‌شود دیگر Endpointها را می‌توان با مشخات موردنیاز تعریف کرد.

به‌طور مثال در این پروژه از کلاس CDC[11] استفاده‌شده است (که در کارت‌های شبکه و مودم‌ها نیز مورداستفاده قرارمی‌گیرد). برای تبادل داده با رایانه در این کلاس از ارسال نوع Bulk استفاده می‌شود. پس اینترفیسی از نوع CDC باید یک Endpoint با دیتاتایپ Bulk به‌عنوان ورودی و یک Endpoint دیگر با دیتاتایپ Bulk به‌عنوان خروجی داشته باشد. آدرس Endpoint ورودی 0x01 و Endpoint خروجی 0x81 تنظیم‌شده‌اند. همچنین چون قصد استفاده از USB در حالت Full Speed راداریم ماکسیموم سایز بسته ارسالی نیز ۶۴ بایت تنظیم شده است.

[11] Communications Device Class

فصل ۷

سنسورها

۱.۷ مقدمه

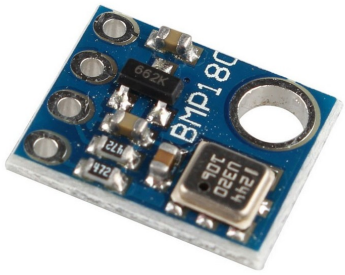
به‌منظور استخراج داده‌های جوی نظیر دمای هوا، رطوبت، فشار، شدت نور و... از سنسورهای دیجیتال استفاده می‌شود. این سنسورها علاوه بر کم‌مصرف بودن دارای دقت بالا و تأخیر پایینی در اندازه‌گیری پارامترهای موردنظر هستند. با دارا بودن این ویژگی‌ها این سنسورها کاملاً با سیستم تغذیه باتری و سلول خورشیدی سازگار هستند.

۲.۷ فشارسنج

تغییرات فشار جوی یکی از عناصر مهم در پیش‌بینی وضعیت آب‌وهوا است. فشارسنج‌های جیوه‌ای از اواخر قرن ۱۶ جهت پیش‌بینی وضعیت آب‌وهوا مورداستفاده قرار می‌گرفت. به‌طورکلی تغییر فشار هوا رو به بالا نشان‌دهنده آسمان آفتابی، گرم و صاف و تغییر فشار هوا رو به پایین نشان‌دهنده بارش باران، طوفان و آسمانی مملو از ابرهای باران‌زا است.

علاوه بر این فشار هوا یکی از عوامل مؤثر در سنجش سرعت باد نیز به شمار می‌آید.

سنسور استفاده‌شده برای این منظور ماژول سنسور BMP180 است که با مصرف جریان تنها در حد چند میکرو آمپر دقتی معادل با ۰٫۳ هکتوپاسکال[1] دارد و توانایی اندازه‌گیری فشار هوا در بازه ۳۰۰ تا ۱۱۰۰ هکتوپاسکال را دارا است. نحوه ارتباط با این ماژول از طریق رابط I^2C است. تصویر رو و پشت این ماژول در شکل ۱.۷ آمده است.



(آ) تصویر روی ماژول BMP180.

(ب) تصویر پشت ماژول BMP180.

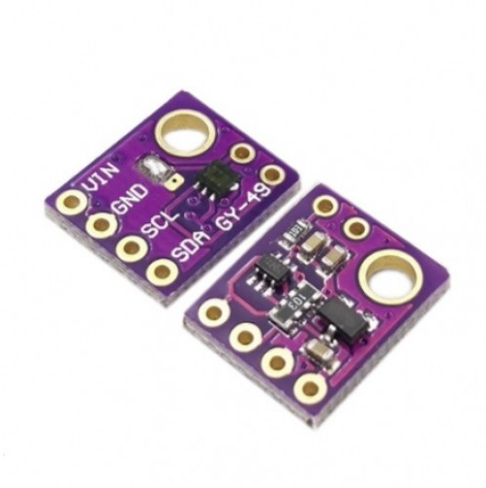
شکل ۱.۷: تصویر پشت و روی ماژول BMP180.

۳.۷ شدت نور

شدت نور خورشید یکی از پارامترهای اصلی سنجش وضعیت آب‌وهوا است. از این سنسور جهت نشخیص ابری یا آفتابی بودن هوا می‌توان استفاده کرد. همچنین به دلیل اهمیت موضوع سلامت پوست، معمولاً در کنار این سنسور از سنسور سنجش شدت UV به‌منظور اطلاع‌رسانی شدت UV نیز استفاده می‌شود.

جهت سنجش شدت نور از ماژول سنسور MAX44009 استفاده‌شده است که با ۰٫۶۵ میکرو آمپر مصرف جریان در هنگام کارکرد شدت نور در بازه‌ی ۰٫۰۴۵ لوکس تا ۱۸۸ هزار لوکس را اندازه‌گیری می‌کند. همچنین اینترفیس ارتباطی این ماژول رابط سریالی I^2C می‌باشد. تصویر رو و پشت این ماژول در شکل ۲.۷ آمده است.

[1] Hectopascal (hPa)

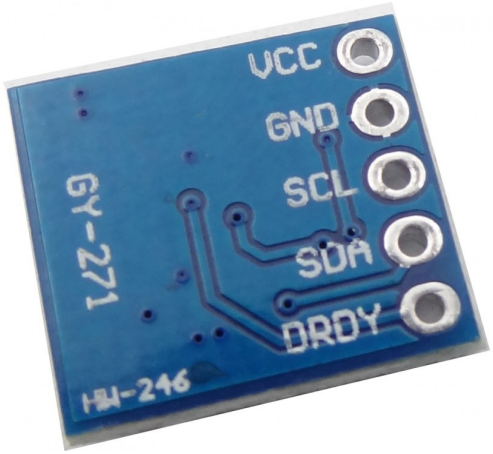


شکل ۲.۷: تصویر پشت و روی ماژول MAX44009.

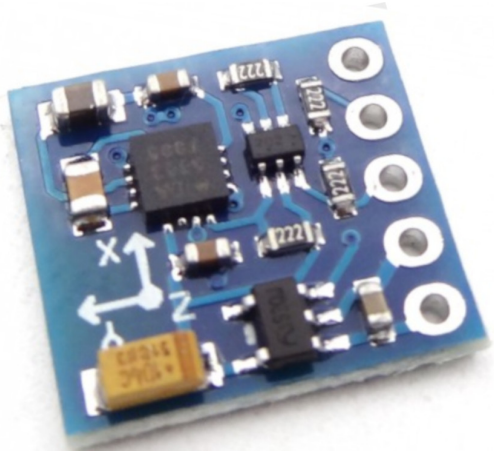
۴.۷ قطب‌نما

ازآنجایی‌که این دستگاه در سمت سنسور معمولاً در ارتفاع ۱۰ متری زمین روی یک میله نصب می‌شود، قرار دادن دستگاه در جهت جغرافیایی خاص، به جهت سنجش جهت باد، ممکن است دشوار باشد (یا حتی خیلی دقیق نباشد). باوجود سنسور قطب‌نما در این دستگاه (برخلاف برخی دستگاه‌های مشابه) دیگر نیازی به نصب دستگاه در جهت جغرافیایی خاص نخواهد بود.

سنسور استفاده‌شده برای این منظور سنسور QMC5883L است. که با ماکسیموم ۱۰۰ میکرو آمپر جریان مصرفی می‌توان به‌دقت یک تا دو درجه در جهت‌یابی رسید. همچنین طریقه ارتباط این سنسور با میکروکنترلر از طریق رابط I^2C است. تصویر رو و پشت این ماژول در شکل ۳.۷ آمده است.



(آ) تصویر پشت ماژول QMC5883L.



(ب) تصویر روی ماژول QMC5883L.

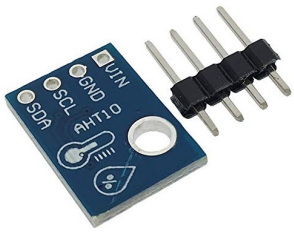
شکل ۳.۷: تصویر پشت و روی ماژول QMC5883L.

۵.۷ دما و رطوبت هوا

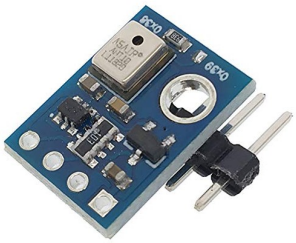
درکتار وضعیت آسمان (ابری، آفتابی، بارانی و... بودن) معمولاً به‌طور مستقیم پارامتر دمای هوا نیز در اطلاع‌رسانی وضعیت و پیش‌بینی آب‌وهوا نقش ایفا می‌کند. در کنار این موارد رطوبت هوا نیز، علاوه بر نقشی که در پیش‌بینی دارد، معمولاً به‌طور مستقیم به سمع و نظر مخاطبین می‌رسد. علاوه این رطوبت نسبی هوا در کنار دمای هوا نقش مهمی در رسیدن به آسایش حرارتی در بدن انسان (و موجودات) دارد. به‌طوری کلی در دمای هوای بالاتر به رطوبت نسبی کمتری نسبت به دمای هوای پایین‌تر برای رسیدن به سطح آسایش حرارتی نیاز است [۳].

ماژول سنسور AHT10 دما و رطوبت نسبی هوا را با دقت ۰٫۰۱ درجه سلسیوس[۲] و ۰٫۰۲۴ درصد با تنها ۳٫۳ میکرو وات مصرف توان اندازه‌گیری می‌کند. همچنین این ماژول در رطوبت ۰ تا ۱۰۰ درصد و دمای ۴۰- تا ۱۰۰ درجه‌ی سلسیوس قابل‌استفاده است. خروجی این ماژول نیز سیگنال دیجیتال ارتباطی I^2C است. تصویر رو و پشت این ماژول در شکل ۴.۷ آمده است.

[۲] Celsius



(آ) تصویر پشت ماژول AHT10.



(ب) تصویر روی ماژول AHT10.

شکل ۴.۷: تصویر پشت و روی ماژول AHT10.

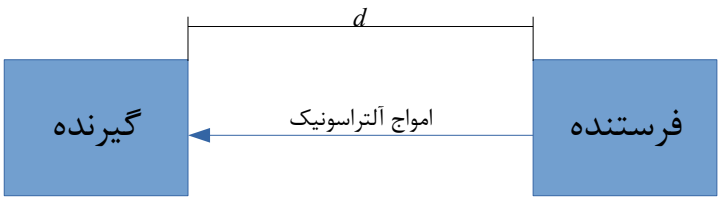
۶.۷ شدت و جهت باد

روش‌های مختلفی برای سنجش شدت باد وجود دارد، عمدتاً در این کاربرد دو روش سنجش مکانیکی و آلتراسونیک [۳] مورداستفاده قرار می‌گیرد. در سنجش شدت و جهت باد در روش مکانیکی از دو ابزار که به‌صورت مستقل کار می‌کنند (در برخی موارد این دو ابزار در قالب یک دستگاه در کنار هم قرار می‌گیرند) استفاده می‌شود، به طوری که یک ابزار برای سنجش شدت و ابزاری دیگر برای تعیین جهت، مورداستفاده قرار می‌گیرد. هر دو دستگاه دارای قطعات متحرک‌اند و یکی با داشتن پره‌هایی شبیه به دم هلی‌کوپتر با وزرش باد در جهت وزش قرار می‌گیرد و دیگری دارای پره‌هایی است که با وزش باد پره‌ها همانند پره‌های توربین به حرکت درمی‌آید که با توجه به‌سرعت چرخش پره‌ها سرعت باد قابل‌اندازه‌گیری است. در اندازه‌گیری‌های این دستگاه‌ها محدودیت‌هایی وجود دارد و اغلب این نوع ابزارها در وزش باد ملایم عملکرد صحیحی از خود نشان نمی‌دهند. همچنین در اندازه‌گیری زاویه وزش ممکن است محدود به اندازه گیری زوایای خاصی باشند.

در روش اندازه‌گیری شدت و جهت باد با آلتراسونیک اندازه‌گیری‌ها می‌تواند در قالب تنها یک دستگاه، بدون قطعات متحرک و با دقتی بالاتر انجام پذیرد. در این روش ۲ فرستنده و گیرنده آلتراسونیک همان‌طور که در شکل ۵.۷ نشان داده‌شده است روبروی یکدیگر در فاصله مشخص d قرار داده می‌شوند.

فرستنده امواج صوتی با فرکانس ۴۰ کیلوهرتز (که برای گوش انسان قابل شنیدن نیست) تولید می‌کند. فاصله

[3] Ultrasonic



شکل ۵.۷: نحوه قرارگیری فرستنده و گیرنده آلتراسونیک.

زمانی بین ارسال امواج صوتی از فرستنده و دریافت این امواج در گیرنده اندازه‌گیری می‌شود. با توجه به رابطه ۱.۷، سرعت v با داشتن فاصله d و زمان تأخیر بین ارسال موج صوتی در فرستنده و دریافت آن در گیرنده t، قابل‌اندازه‌گیری است.

$$v = \frac{d}{t} \tag{۱.۷}$$

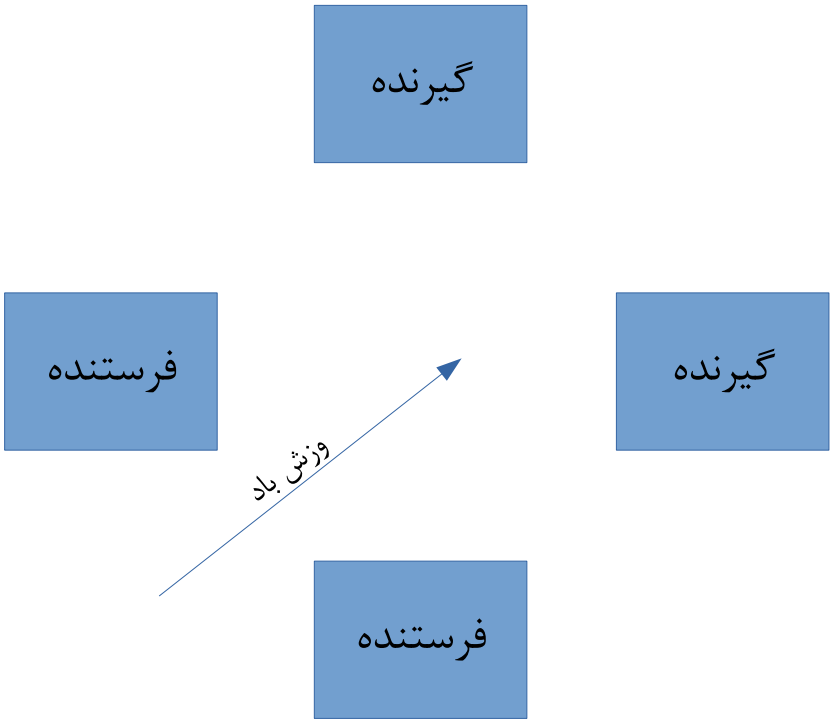
سرعت v به‌دست آمده از این رابطه مطابق رابطه ۲.۷ متشکل از سرعت صوت v_s و سرعت باد v_{wx} است [۴].

$$v = v_s + v_{wx} \tag{۲.۷}$$

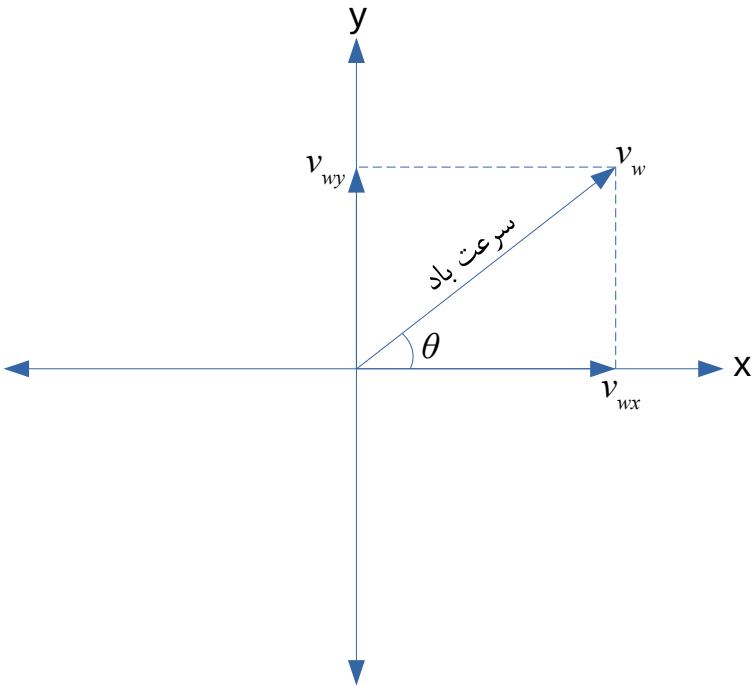
درصورتی‌که وزش باد در جهت موافق حرکت امواج صوتی باشد زمان تأخیر در دریافت امواج نسبت به حالی که باد نوزد کمتر شده و درنتیجه سرعت v نسبت به حالتی که باد نوزد افزایش می‌یابد (یعنی علامت v_{wx} مثبت بوده و $v = v_s + v_{wx}$). درصورتی‌که وزش باد در خلاف جهت حرکت امواج صوتی باشد زمان تأخیر در دریافت امواج نسبت به حالتی که باد نوزد بیشتر شده و درنتیجه سرعت v کاهش می‌یابد (یعنی علامت v_{wx} منفی بوده و $v = v_s - v_{wx}$). با داشتن سرعت صوت v_s، فاصله d و محاسبه زمان t می‌توان با توجه به معادلات ۱.۷ و ۲.۷ سرعت باد v_{wx} و جهت باد (علامت سرعت v_{wx}) روی یک محور مطابق رابطه ۳.۷ به‌دست آورد.

$$v_{wx} = \frac{d}{t} - v_s \tag{۳.۷}$$

به‌منظور سنجش شدت و جهت باد در دو بعد می‌توان دو فرستنده و گیرنده دیگر بر روی محوری عمود بر محور متصل‌کننده فرستنده و گیرنده فعلی قرارداد و با محاسبه دو بردار سرعت، بردار سرعت برآیند را به‌دست آورد.



شکل ۶.۷: نحوه قرارگیری فرستنده و گیرنده آلتراسونیک دو محوره.



شکل ۷.۷: بردارهای متناظر با نحوه قرارگیری فرستنده و گیرنده آلتراسونیک دو محوره.

اگر نحوه قرارگیری فرستنده و گیرنده‌ها مطابق شکل ۶.۷ باشد در این صورت صفحه مختصات متناظر با آن

جدول ۱.۷: ضرایب محاسبه سرعت صوت (فرمول ۵.۷) [۵].

ضرایب	
a_0	331.5024
a_1	0.603055
a_2	-0.000528
a_3	51.471935
a_4	0.1495874
a_5	-0.000782
a_6	-1.82×10^{-7}
a_7	3.73×10^{-8}
a_8	-2.93×10^{-10}
a_9	-85.20931
a_{10}	-0.228525
a_{11}	5.91×10^{-5}
a_{12}	-2.835149
a_{13}	-2.15×10^{-13}
a_{14}	29.179762
a_{15}	0.000486

مطابق شکل ۷.۷ خواهد بود. اندازه v_w و زاویه θ بردار سرعت باد از طریق روابط ۴.۷ به‌دست می‌آیند.

$$\begin{aligned} v_w &= \sqrt{v_{wx}^2 + v_{wy}^2} \\ \theta &= \tan^{-1}\left(\frac{v_{wy}}{v_{wx}}\right) \end{aligned} \tag{۴.۷}$$

سرعت صوت v_s به‌عنوان تابعی از دما، فشار و کسر مولی رطوبت و کربن دی‌اکسید، با استفاده از رابطه ۵.۷

قابل‌محاسبه است [۵]. ثوابت a_1 تا a_{15} در جدول ۱.۷ آمده‌اند.

$$\begin{aligned} v_s(\tau, p, x_w, x_c) =& a_0 + a_1\tau + a_2\tau^2 + \left(a_3 + a_4\tau + a_5\tau^2\right) x_w \\ &+ \left(a_6 + a_7\tau + a_8\tau^2\right) p + \left(a_9 + a_{10}\tau + a_{11}\tau^2\right) x_c \\ &+ a_{12}x_w^2 + a_{13}p^2 + a_{14}x_c^2 + a_{15}x_w p x_c \end{aligned} \tag{۵.۷}$$

که τ دمای هوا (برحسب درجه سلسیوس)، p فشار هوا (برحسب پاسکال)، x_w کسر مولی بخارآب در هوا و x_c کسر مولی کربن دی‌اکسید در هوا است. x_c را ثابت و برابر 400×10^{-6} در نظر می‌گیریم. کسر مولی بخارآب در هوا x_w از رابطه ۶.۷ به‌دست می‌آید [۶].

$$x_w = \frac{hfp_{sv}}{100p} \tag{۶.۷}$$

که h درصد رطوبت هوا، p_{sv} فشار اشباع بخارآب در هوا و f ضریب تقویت است و از طریق روابط ۷.۷ و ۸.۷ محاسبه می‌شوند [۷].

$$f = 1.00062 + 3.14 \times 10^{-8}p + 5.6 \times 10^{-7}\tau^2 \tag{۷.۷}$$

$$\begin{aligned} p_{sv} =& \exp\left(1.2811805 \times 10^{-5}T^2 - 1.9509874 \times 10^{-2}T \right. \\ &\left. +34.04926034 - 6.3536311 \times 10^3/T\right) \end{aligned} \tag{۸.۷}$$

که در این روابط T دمای محیط برحسب کلوین[۴] است، یعنی:

$$T = \tau + 273.15 \tag{۹.۷}$$

[4] Kelvin

فصل ۸

طراحی بردمدارچاپی

۱.۸ مقدمه

بردمدارچاپی[1] بردی است که با کانکتور و ترک‌هایی که روی آن قرار دارد قطعات را به‌طور الکتریکی به یکدیگر متصل می‌کند. بردمدارچاپی به‌طورکلی متشکل از هسته‌ای محکم معمولاً از جنس فیبرشیشه FR4 و لایه‌های مسی نازک که در یک یا دو طرف آن قرار دارند می‌شود. ترک‌ها و پدها روی لایه‌های مسی و پس از حل کردن مس سایر بخش‌ها در اسید به وجود می‌آیند. برای محافظت از تماس ناخواسته ترک‌های مسی با قطعات و ذرات رسانا و همچنین جدا نمودن محل‌های لحیم‌کاری از سایر بخش‌ها، بر روی لایه مسی لایه‌ی Solder mask قرار می‌گیرد که معمولاً به رنگ سبز است.

یکی از موارد مهم در مرحله طراحی‌بردمدارچاپی توجه به تأثیرات نویز در مدار و تلاش به کاهش اثرات آن است. نویزها می‌تواند به دو صورت الکتریکی (در اثر خاصیت خازنی بین دو هادی) و مغناطیسی (در اثر خاصیت سلفی هادی)

[1] Printed Circuit Board (PCB)

به وجود بیایند. منابع اصلی نویز سیگنال‌های پرودیک با فرکانس بالا نظیر زوج سیم‌های تفاضلی (D+ و D- موجود در USB) یا سیگنال‌های ساعت[۲] هستند.

با عبور جریان از یک هادی میدان مغناطیسی در اطراف آن به وجود می‌آید و در صورت وجود هادی‌ای دیگر در نزدیکی آن میدان تولیدشده روی آن تأثیر گذاشته و باعث ایجاد جریان در هادی دوم می‌شود. همچنین وجود اختلاف‌پتانسیل بین دو هادی مجاور سبب ایجاد میدان الکتریکی شده و تغییرات پتانسیل یک هادی روی دیگری همانند صفحات یک خازن اثر می‌گذارد. کاهش طول هادی‌ها یکی از عوامل مؤثر در کاهش این نوع نویزها است. استفاده از خازن‌های کوپلاژ برای میکروکنترلر [۸]، استفاده از فیلترهای RC و LC، استفاده از Polygon (یا Pour) و استفاده از انواع شیلد و محافظ‌های زمین شده، از روش‌ها کاهش انواع نویز هستند که در طراحی بردمدار چاپی باید به آن‌ها توجه کرد. بردهای این پروژه نیز مطابق با این نکات طراحی‌شده‌اند.

۲.۸ ملاحظات طراحی PCB

در گذشته فرکانس سیستمها از حد چند مگاهرتز تجاوز نمیکرد و برد های مدار چاپی یک یا دو لایه تنها به عنوان بستری جهت قرار دادن المانها و برقراری ارتباط بین آنها مورد استفاده قرار می گرفت. همچنین نحوه اتصال و قرار گیری المانها و مسیر ها تاثیری در کارایی سیستم نداشت.

عوامل تاثیر گذار در بازنگری نحوه طراحی سیستم های جدید

۱- کاهش ابعاد المانهای مدار و فاصله ی بین آنها

۲- کاهش عرض مسیر سیگنالها و فاصله بین آنها

۳- کاهش ولتاژ تغذیه سیستم و افزایش جریان آن

۴- کاهش حاشیه نویز

۵- افزایش فرکانس و نرخ انتقال اصلاعات در سیستم های دیجیتال

۶- المان های سوئیچینگ با جریان بالا

۷- تراشته هایی با تعداد پایه های وروی - خروجی زیاد

اغلب طراحان دیجیتال، سیستم را در حوزه و زمان و با توجه به پارامتر هایی از قبیل زمان صعود سیگنال یا تاخیر انتشار توصیف میکند در حالیکه جهت تحلیل و بررسی بسیاری از پدیده ها در سیستم های آنالوگ از حوزه فرکانس استفاده می شود.

[2] Clock

کاهش ابعاد سیستم های طراحان برد های الکترونیکی را بر آن واداشته است که المانها و مسیر سیگنالها را در کمترین فاصله ممکن از یکدیگر قرار دهند که همین امر موجب شده است اثر متقابل میدان های الکتریکی و مغناطیسی تولید شده توسط اجزای مختلف مدار بیش از قبل مورد توجه قرار گیرد. اتصال بین المانهای مختلف سیستم از مبدا تا مقصد توسط سیم، کابل یا مسیرهای موجود به روی برد برقرار میشود. در فرکانس های پایین می توان این مسیرها رو به صورت ایده آل در نظر گرفت.

اما همانگونه که در شکل مشاهده میشود با افزایش فرکانس، هاصیت سلفی و خازنی و مقاومتی مسیر اثر خود را نشان می دهد. لازم است طراحی سیستم مورد توجه قرار گیرد زیرا امپدانس ایجاد شده توسط این المانها به تضعیف سیگنال در فرکانس های بالا منتهی می شود. به این ترتیب با افزایش فرکانس، اثرات سلفی و خازنی المانهای مدار که در فرکانسهای پایین از آنها صرفنظر میشود، قابل چشم پوشی نمی باشد و مسیر های اتصال دهنده المانها به روی برد سیستم، به صورت مجموعه ای از سلفها و خازنها مدلسازی میشود که این مدل، خط انتقال نامیده میشود.

نکته: باید توجه داشت که بر خلاف تصور رایج تنها فرکانس سیگنالها عامل تعیین کننده نمی باشد بلکه زمان نزول و صعود سیگنال نیز معیار تصمیم گیری است. به عنوان مثال طراحی سیستمی با فرکانس کاری ۱۰۰MHZ و زمان صعود و نزول ۲ns ساده تر از طراحی و نزول ۱ns می باشد.

در سیستمهای مرکب که از دو بخش دیجیتال و آنالوگ تشکیل شده اند مشکلات دیگری نیز وجود دارد. به عنوان مثال در مورد مبدل های آنالوگ به دیجیتال پیشرفته، انرژی سیگنال در سمت دیجیتال ممکن است ۱۳۰ db بیشتر از انرژی سیگنال در سمت آنالوگ باشد، به این ترتیب نویز موجود در قسمت دیجیتال، ورودی آنالوگ را به شدت تحت تاثیر قرار می دهد و باید از روش های مناسب به منظور کاهش این اثرات استفاده نمود.

از سوی دیگر با پیشرفت تکنولوژی ساخت ادوات نیمه هادی، تعداد پایه های خارجی تراشه ها و حجم گیت های داخلی آنها ، اقزایش چشم گیری داشته است.

همچنین با اقزایش سرعت پردازش سیستم از چند مگاهرتز به چند گیگاهرتز، مصرف جریان دینامیک تراشه ها چندین برابر شده است. با توجه به آنکه نویز ایجاد شده در سیستم مطابق با رابطه ،L*di/dt با سرعت و مقدار تغییرات جریان متناسب است عوامل فوق به افزایش نویز تولید شده منتهی میشود.

با کاهش ولتاژ تغذیه سیتم از ۵v به مقادیر ۲v.۱، ۸v.۱، ۵v.۲، حساسیت تراشه ها به نویز سیستم نیز بیش از پیش افزایش یافته است که عدم توجه به موارد فوق در هنگام طراحی، به عملکرد نامطلوب سیستم منتهی خواهد شد.

با توجه به مطالب عنوان شده شناخت پدیده های ایجاد شده در فرکانس های بالا از قبیل اثر سیکنال ها به روی یکدیگر تغییر ولتاژ سیگنال مرجع و بررسی راه حل های از قبیل کنترل امپدانس، مسیرکشی صحیح سیگنالهای آرایش صحیح لایه ها در برد های مدار چاپی و طراحی صحیح سیتم تغذیه از جمله مواردی هستند که به منظور حذف یا

کاهش اثر پدیده های فوق در فرکانس های بالا مورد توجه می باشد.

۱.۲.۸ خطوط انتقال

با افزایش فرکانس لازم است اثرات خطوط انتقال سیگنال در طراحی سیستم مورد توجه قرار گیرد به این ترتیب در صورتی که طول مسیر سیگنال به روی برد مدار چاپی تا حدی افزایش یابد تا زمان لازم جهت رفت و برگشت سیگنال در طول مسیر قابل مقایسه با زمان صعود یا نزول آن باشد، لازم است محاسبات خطوط انتقال در طراحی مورد توجه قرار گیرد.

عدم تطابق بین انپدانس نشخصه خط انتقال با امپدانس خروجی در درایور یا امپدانس ورودی بار به بازتابش سیگنال منتهی می شود. این باز تابش ها به اعوجاج سیگنال و بروز فرا جهش یا فرو جهش در شکل موج آنها می انجامد که در سیستم های آسنکرون ممکن است تغییرات ناخواسته سیگنال و عملکرد ناصحیح سیستم را در پی داشته باشد.

نتیجه‌گیری

کلیه مسیرها توسط باری برابر با امپدانس مشخصه خط پایان دهی شوند.

مسیر سیگنالهای فرکانس بالا با توجه به اصول خط انتقال طراحی شود.

تغییر امپدانس مسیر می تواند بر اثر هر یک از عوامل زیر رخ دهد:

۱- تغییر عرض مسیر سیگنال

۲- قرار ادن via در میسیر سیگنال و تغییر لایه آن

۳- عبور از کانکتور

۴- انشعابات ایجاد شده در مسیر سیگنال

۲.۲.۸ اندازه گیری بازتابش سیگنالها

به منظور اندازه گیری باز تابش سیگنالها نرم افزارهای متعددی ارئه شده است که میتوان با شبیه سازی خط انتقال و مدلسازی آن، سیگنال مناسب را به خط انتقال اعمال کرد و نحوه بازتابش سیگنال در نقاط مختلف را بررسی نمود. علاوه بر نرم افزارهای موجود از وسیله ای به نام TDR استفاده می شود.

بر اسای مطالب گفته شده علت اصلی بازتابش سیگنالها عدم تطبیق امپدانس مشخصه خط انتقال با امپدانس

فرستنده و گیرنده می باشد به این ترتیب با رفع این مشکل میتوان اثر پدیده های ناشی از بازتابش سیگنالها را تا حد زیادی کاهش داد. در ادامه روشهای موجود جهت تطبیق امپدانس خط انتقال با مشخصات فرستنده و گیرنده را مورد بررسی قرار میدهیم.

پایان دهی موازی

در این روش مقاومتی برابر با امپدانس مشخصه خط به انتهای مسیر متصل می شود. به این ترتیب کلیه انرژی سیگنال توسط مقاومت جذب می شود و هیچگونه بازتابش سیگنال رخ نمی دهد. مقاومت می تواند به زمین یا تغذیه متصل شود و لازم است در کمترین فاصله ممکن از پایه تراشه قرار گیرد و از ایجاد انشعابات ناخواسته اجتناب شود. از مزایای این روش میتوان به موارد زیر اشاره کرد:

۱- مقدار مقاومت به اسانی قابل تعیین است.

۲- با افزودن یک المان قابل پیاده سازی است.

عیب این روش در آنست که همواره یک جریان DC ثابت از مقاومت به سمت زمین برقرار می باشد و با افزایش تعداد این مقاومتها، مصرف توان سیستم به مقدار قابل توجهی افزایش می یابد.

پایان دهی تونن

در این روش از دو مقاومت که یکی به زمین و دیگری به منبع تغذیه متصل می شود، استفاده می گردد. این مقاومتها می توانند به صورت pull-diwn،pull-up عمل نمایند و سطح dc باس را تنظیم نمایند. مقدار آنها به نحوی تعیین می شود که معادل تونن آنها با امپدانس مشخصه خط انتقال برابر باشد.

پایان دهی AC

مشکل دو روش قبل آنست که جریان ثابتی توسط مقاومتها مصرف می شود جهت رفع این مشکل می توان خازنی را با مقاومت سری نمود به این ترتیب جریان DC مسیر توسز خازن مسدود می شود و هیچگونه تلفاتی در مقاومت رخ نمی دهد. اما مشکل این روش آنست که المان دیگری به مدار افزوده می شود همچنین با توجه به مقدار ثابت زمانی RC شارژ و دشارژ خازن باعث بروز اعوجاج در شکل موج سیگنال خواهد شد.

پایان دهی سری

در روشهای قبل تطبیق امپدانس گیرنده و خط انتقال جهت حذف یا کاهش باز تابش سیگنالها مورد استفاده قرار میگرفت. اما در روش سری، تطبیق امپدانس در بخش فرستنده صورت میگیرد. مزیت این روش در آنست که جریان DC از مقاومت عبور نمی کند و تلفات جریان DC نخواهیم داشت. استفاده از این روش در باس داده و آدرس حافظه های پر سرعت و توزیع سیگنال کلاک بسیار رایج می باشد.

پایان دهی دیود

این روش تاثیری در وقوع بازتابش سیگنال نداشته، تنها به منظور محدود کردن دامنه بازتابش بین ،Vcc GND مورد استفاده قرار می گیرد. دیودها می توانند در هر نقطه از مسیر قرار گیرند. به طور معمول دیودهای شاتکی و فرکانس بالا در این پیکربندی مورد استفاده قرار میگیرند.

۳.۲.۸ ساختار برد های مدار چاپی

TRACK به منظور ایجاد اتصالات لازم بین المانهای مختلف مدار مورد استفاده قرار میگیرد. در فرکانسهاد کم همانند سیم با مقاومت بسیار کم عمل میکند اما با افزایش فرکانس اثر سلفی آن اهمین می یابد. با کاهش طول و افزایش عرض مسیر از خاصیت سلفی آن کاسته می شود.

باید توجه داشت که هر مسیر سیگنال همانند یک مقاومت عمل میکند که با افزایش عرض، مقاومت آن کاهش یافته، حرارت کمتری در آن تلف میشود. مقاومت هر مسیر علاوه بر عرض به ضخامت مس قرار گرفته بر روی آن نیز وابسته است که این ضخامت توسط واحد اونس بر فوت مربع بیان میشود.

PAD به منظور اتصال قطعات به برد مورد استفاده قرار میگیرد و با توجه به نوع المانها به دو دسته زیر تقسیم می شود: THT[3] و SMD[4] به طور کلی در کابرد های فرکانس بالا بهتر است از قطعات SMD استفاده شود زیرا طول بلند پایه ها در قطعات THT به ایجاد اندوکتانس ناخواسته و اثرات نامطلوب به آن منتهی خواهد شد.

Via جهت اتصال مسیر سیگنال در لایه های مختلف به یکدیگر مورد استفاده قرار می گیرد و با توجه به نوع کاربرد به انواع زیر تقسیم میشود:

Through hold: رایجترین نوع می باشد و سوراخ ایجاد شده، کل مقطع برد را در بر میگیرد.

Blind via: از لایه بالا یا پایین برد به یکی از لایه های میانی متصل می شود.

[3] through hole technology

[4] surface mount technology

Buried via: به منظور اتصال لایه های داخلی مورد استفاده قرار می گیرد.

Via مزیت استفاده از انواع blind و Buried در آنست که به دلیل سوراخ نشدن کلیه لایه ها، فضای بیشتر جهت مسیر کشی سیگنالها در دسترس می باشد اما هزینه پیاده سازی آنها بیشتر است. هر Via دارای خاصیت سلفی و خازنی می باشد که باعث می شود سیگنال عبوری از آنرا تحت تاثیر قرار دهد. ساختار داخلی Via باعث می شود که به صورت سلف عمل نماید و با افزایش فرکانس اثر سلفی آن افزایش می یابد.

نکته: با توجه به آنکه رابطه قطر Via با خاصیت سلفی آن به صورت لگاریتمی می باشد و افزایش قطر، تاثیر زیادی در کاهش مقدار سلفی نداره، در مواردی که لازم است مقدار اندوکتانس مسیر کاهش یابد به عنوان مثال در اتصال خازنهای تغذیه بهتر است از چند Via به صورت موازی استفاده شود.

۴.۲.۸ صفحه مرجع

در بردهای چند لایه، یک یا چند عدد از لایه ها به ولتاژ زمین یا تغذیه متصل شده، به عنوان مرجعی جهت انجام محاسبات امپدانس مورد استفاده قرار میگیرند. یکی از مزایای بسیار مهم استفاده از صفحات مرجع، کاهش امپدانس مسیر برگشت سیگنالها می باشد که به کاهش نویز تولید شده منتهی می شود.

۵.۲.۸ برد های چند لایه

در مواردی که پیچیدگی سیم کشی اندک است و فرکانس کار آن هم بالا نیست می توان از بردهای یک یا دو لایه جهت نگهداری المانها و برقراری ارتباط بین آنها استفاده نمود. در صورت استفاده از بردهای دو لایه بهتر است لایه فوقانی جهت مسیر کشی سیگنالها مورد استفاده قرار گیرد و لایه زیرین بصورت صفحه ای یکپارچه، به سیگنال زمین اختصاص یابد.

مزایای بردهای چند لایه

با توجع به آنکه سیگنال زمین بیشترین حجم مسیر را شامل میشود. اختصاص یک صفحع یکپارچه به آن، مسیر یابی سایر سیگنالها را تسهیل میکند. استحکام مکانیکی برد افزایش میابد. با کاهش امپدانس زمین به کاهش نویز منتهی میشود. به عنوان یک شیلد عمل کرده، باعث کاهش تاثیر نویز خارجی به روی سیستم و کاهش تشعشعات ساطع شده از آن میشود.

۶.۲.۸ روند طراحی بردهای فرکانس بالا

به طور کلی به منظور طراحی بردهای فرکانس بالا می توان از دو روش زیر استفاده نمود:

۱- طراحی تنها با توجه به مشخصات المانها و تجربه شخص طراح انجام میشود.

۲- از ابزار های مناسب جهت شبیه سازی و اطمینان از صحت عملکرد برد طراحی شده قبل از و منتاژ آن استفاده شود.

۷.۲.۸ انتخاب المانها و طراحی شماتیک

در این بخش المانهای لازم جهت پیاده سازی سیستم از قبل تراشه ها، کانکتورها، مقاومت و خازنها انتخاب شده، اتصالات لازم بین آنها در محیط شماتیک برقرار می شود.

نرم افزار شبیه ساز از این اطلاعات به منظور مدلسازی عملکرد ورودی، خروجی تراشه ها، بخشهای درایور و گیرنده و سایر المانهای مدار استفاده می کند. این مدلها غالبا توسط تولید کننده تراشه و به دو شکل زیر ارائه میشوند: ۱- Spice model ۲- IBIS model باید توجه داشت که سرعت شبیه سازی با استفاده از مدل IBIS، در مقایسه با مدل Spice بیشتر است.

در این بخش میتوان موارد زیر را با توجه به نقش شماتیک سیستم آنالیز نمود: بررسی میزان تاثیر متقابل سیگنالها

تحلیل عدم تطبیق امپدانس میسرها و بازتابش امواج

نحوه پایان دهی خطوط انتقال

تعیین حداکثر طول مسیرها با توجه به تاخیر انتشار

چینش بهینه المانهای برد

۸.۲.۸ چینش المانها و مسیر کشی آنها

در این مرحله مکان فیزیکی تراشه ها و سایر المانهای برد با توجه به قیود طراحی تعیین می شوند و مسیر کشی سیگنالها با رعایت نکات لازم به صورت دستی یا خودکار صورت می پذیرد.

۹.۲.۸ شبیه سازی پس از چینش المان‌ها

در این بخش قبل از ساخت برد، شبیه سازی لازم جهت بررسی کیفیت سیگنالها و اثر متقابل آنها به روی یکدیگر انجام میشود. نرم افزار شبیه ساز با توجه به آرایش برد و مدل المانها، مسیرهای با ریسک بالا را که منشا مشکلات بالقوه سیستم هستند، مشخص مینماید و آنالیزهای لازم را به روی آنها انجام میدهد. به این ترتیب می توان از بروز مشکلاتی که یافتن آنها با تست فیزیکی برد بسیار پیچیده یا غیر ممکن خواد بود، جلوگیر نمود.

۱۰.۲.۸ نکات طراحی

جدا کردن قسمتهای مختلف مدار المانها و تراشه های مختلف موجود به روی برد، داری فرکانس کاری و نویز پذیر متفاوتی هستند. نخستین گام در یک طراحی اصولی، چینش صحیح المانهای مدار میباشد که توجه به این مساله باعث حدف بسیاری از مشکلات بالقوه موجود خواهد شد.

۱۱.۲.۸ حذف پد بدون استفاده

الف- پدهای بی استفاده باید در همه ی لایه ها باقی باشند به جز مواردی که در بخش ب مشخص می شوند.

- وجود پدهای بی استفاده خطر رانش دیواره سوراخ و نیز ترک خوردگی در نواحی رزین را کاهش می دهد.

ب- پدهای بی استفاده در موارد زیر می توانند حذف بشوند:

۱ .وقتی که وجود این پدها از کارآیی الکتریکال بکاهد می توانند حذف بشوند.

۲ .برای پیشگیری فشار باال در زمان laminate کردن که باعث کچلی رزین در prepreg می‌شود این پدها را سازنده می تواند حذف نماید.

ج- وقتی پدهای بی استفاده مطابق با توضیحات بالا حذف شدند:

۱ .پدهای بی استفاده در همه ی لایه های plane باقی بمانند.

۲ .پدهای بی استفاده در متن‌لاتینlaminate منعطف باقی بماند،

۳ .نباید حذف این پدها در بیش از نیمی از لایه ها انجام شود.

۴ .پدهای بی استفاده نباید در بیش از دو لایه مجاور حذف بشوند.

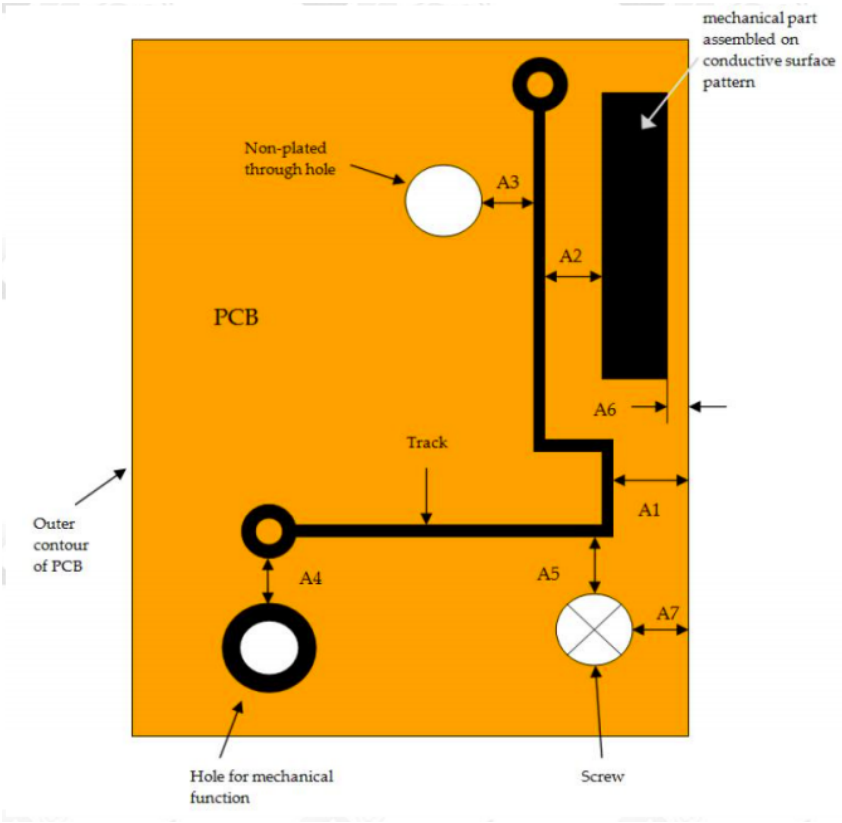
۵ .پدهای بی استفاده فقط در یک طرف یک laminate می تواند حذف بشود.

۱۲.۲.۸ حداقل فاصله درساخت

حداقل فاصله اجزای مختلف مدار در طراحی برد مدار چاپی در جدول ۲.۸ و تصویر ۱.۸ آمده است.

جدول ۱.۸: حداقل فاصله با اجزای مختلف در طراحی برد مدار چاپی.

	عنوان	مجاز	پیشنهادی
A1	Track – pcb edge	0.7	
A2	Track – mechanical part	1.0	2.0
A3	Track – nPTH	0.4	
A4	Via – Mechanical function	0.1	
A5	Track – Screw	0.5	
A6	Copper – pcb edge	0.25	
A7	Edge of screw hole – pcb edge	1	1.5



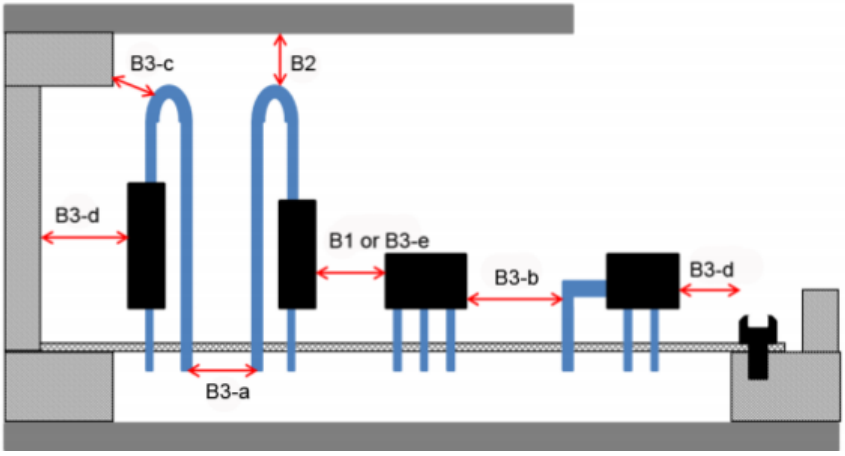
شکل ۱.۸: حداقل فاصله با اجزای مختلف در طراحی برد مدار چاپی.

۱۳.۲.۸ فاصله قطعات ار یکدیگر

حداقل فاصله قطعات از یکدیگر در طراحی برد مدار چاپی در جدول ؟؟ و تصویر ۲.۸ آمده است.

جدول ۲.۸: حداقل فاصله قطعات از یکدیگر در طراحی برد مدار چاپی.

	عنوان	مجاز	پیشنهادی
A1	Track – pcb edge	0.7	
A2	Track – mechanical part	1.0	2.0
A3	Track – nPTH	0.4	
A4	Via – Mechanical function	0.1	
A5	Track – Screw	0.5	
A6	Copper – pcb edge	0.25	
A7	Edge of screw hole – pcb edge	1	1.5



شکل ۲.۸: حداقل فاصله قطعات از یکدیگر در طراحی برد مدار چاپی.

۱۴.۲.۸ مالحظات عمومی در ساختار

الف- ساختار لایه ها باید متقارن باشد.

- این کار از تاب برداشتن و پیچ خوردن برد پیشگیری می کند.

۱۵.۲.۸ مسخصات مس

الف- ضخامت ورق مسی در دو طرف دی الکتریک باید برابر باشد، البته به جز لایه های بیرونی

- برابر بودن ضخامت مس ها زمانی الزامی است که ETCHING یک laminate در یک مرحله با هم انجام شود، که این موضوع مربوط به لایه های میانی است. استثنای در مورد لایه های بیرونی به این دلیل است که هر طرف آن در مرحله ای متفاوت اسیدکاری می شود و نیز به دلیل اینکه لایه های بیرونی پس از آبکاری، ضخامتشان بیشتر می شود.

- البته در ساختاری که تعداد prepreg ها از core ها بیشتر باشد، اسید کاری دو لایه ی بیرونی با هم و در یک مرحله انجام می شود.

ب- در صورتی که عدم تقارن در ضخامت مس ها الزامی بود، ۷۰/۱۷ خطرناکتر از ۳۵/۱۷ یا ۷۰/۳۵ است.

ج- لایه های میانی و بیرونی باید ضخامتی برابر با ۱۷ میکرومتر یا ۳۵ میکرومتر یا ۷۰ میکرومتر داشته باشند.

د- توزیع مس در لایه ها باید به صورت همگون باشد.

- این امر، فشار در مرحله ی laminating را اطمینان می بخشد، چرا که در این مرحله، امکان پر شدن جاهای خالی، شکستگی و فشرده شدن شیشه وجود دارد.

هـ- توازن مس باید با مس های بدون اتصال انجام شود.

- البته در این مورد مطالب بسیار کمی موجود است و این که اشکال داشتن صفحه زمین در لایه های بیرونی یا مزیت داشتن مس های بدون اتصال نسبت به صفحه زمین چیست جایی عنوان نشده است.

- اهمیت این کار در موارد استفاده از مس های ضخیمی مثل ۷۰ میکرومتر بیشتر است.

و- سیگنال ها، نباید در لایه های مربوط به صفحات زمین و تغذیه مسیرکشی شوند.

ز- کیفیت مس باید باال باشد و از نوع HTE انتخاب شود.

- نوع HTE می تواند مطابق با نوع E در مستند IPC-4562 باشد.

۱۶.۲.۸ ضخامت دی الکتریک

الف- حداقل دو ورق از prepreg باید برای جدا کردن دو لایه استفاده شود.

ب- حداقل از دو ورق شیشه در laminate های تقویت شده با شیشه باید استفاده شود.

ج- laminate های با ضخامت دی الکتریک ۱۰۰ میکرومتر در طراحی، فقط برای موارد زیر صحیح است:

۱ .ولتاژ کمتر از ۳۰ ولت

۲ .ضخامت ورق مسی در دو طرف ۱۷ میکرومتر یا ۳۵ میکرومتر باشد.

۳ .سازنده تضمین کند ضخامت laminate بیش از ۳/۸ میل خواهد شد.

د- درطراحی ضخامت دی الکتریک ۱۲۵ میکرومتر برای ولتاژ باالی ۳۰ ولت نباید استفاده شود، مگر اینکه:

۱ .ضخامت laminate درساخت مطابق با جداول ارائه شده باشد.

۲ .سازنده تضمین کند که ضخامت laminate اسیدکاری شده، بیش از ۴/۸ میل می شود.

۱۷.۲.۸ اندازه PCB

بیشترین ابعاد مدارچاپی بستگی به ابعاد پنلی که سازنده تایید کرده دارد. اندازه یک پنل بزرگ نوعی ۴۵x۶۰ سانتی‌متر می باشد. البته همه ی سطح پنل برای طراحی مدارچاپی در دسترس نیست، چرا که مواردی به صورت اجباری در پنل قرار دارند که فضای طراحی را محدودتر می کنند. ضخامت مدار چاپی باید به طور کامل با لایه پوشش محافظ mask(solder)اندازه گیری شود.

نکته۱ - تلرانس ساخت را سازنده مقرر می کند.

نکته۲ - تلرانس ساخت مطابق با استاندارد ST-Q-ECSS-70-11 باید ۱۰ درصد باشد.

نکته۳ -تلرانس مربوط به مس مربوط به copper clad laminate با ضخامت ۱۰۰ میکرومتر حدود ۱۸± درصد می باشد. اما تلرانس کلی مدار چاپی نوعا پایین تر است.

۱۸.۲.۸ تعداد لایه مس در مدارچاپی

الف- تعداد لایه های مس برای پلی-آمید نباید بیش از ۲۶ باشد.

ب- تعداد لایه های مس برای اپوکسی نیز نباید بیش از ۲۰ لایه باشد.

۱۹.۲.۸ مقدار ratio aspect

مقدار aspect ratio نباید بیش از ۷ باشد.

نکته- نیاز به مالحظات بیشتری در مورد قطر ویا با توجه به اینکه ویاها آبکاری می شوند نیست و رعایت همان عدد ۷ درطراحی در مورد قطر ویا کفایت می کند.

۲۰.۲.۸ پهنا و فاصله ی track

تلرانس ساخت برای پهنا و فاصله

الف- تلرانس نسبی ساخت پهنا و فاصله لایه های میانی و بیرونی باید طبق جدول تصویر ۳.۸ باشد.

ب- حداکثر تلرانس مطلق ساخت پهنا و فاصله باید مطابق جدول ۳.۸ باشد.

نکته۱ -به عنوان مثال، در لایه های میانی، پهنای مسیر ۴ میلی‌متر روی مس ۳۵ میکرومتر تلرانس نسبی اش ۲۰± درصد است، یعنی ۸۰ میکرومتر که این عدد با توجه به حداکثر تلرانس مطلق به ۵۰ میکرومتر محدود می گردد.

نکته۲ -تلرانس در اندازه های طراحی اعمال می شود. به عنوان مثال، یک فاصله ۲ میلی‌متر درطراحی با تلرانس ۲۰± درصد نتیجه اش درساخت می شود فاصله‌ای بین ۱۶ میلی‌متر تا ۲۴ میلی‌متر.

Thickness category [μm]	Basic Cu [μm]	Plated Cu [μm]	Total Cu thickness [μm]	Relative tolerance on track width and spacing	Maximum absolute tolerance on track width and spacing
Th≤17	≤ 17	0	≤ 17	± 20 % FOR 11.5.5a.5: -10% to +20%	± 30 μm
17<Th≤60	≤ 17 35 35	1x25 0 1x25	≤ 42 35 60	± 20 %	± 50 μm
60<Th≤70	≤ 17 70	2x25 0	≤ 67 70		± 50 μm
70<Th≤95	35 70	2x25 1x25	85 95		± 70 μm
Note: Explanation of "FOR" is given in Table 13-1.					

شکل ۳.۸: تلرانس ساخت برای پهنا و فاصله.

توضیح جدول تصویر ۳.۸: تلرانس پهنا و فاصله مسیر در لایه های میانی و بیرونی

- ستون اول، ضخامت مس
- ستون دوم، ضخامت مس پایه
- ستون سوم، ضخامت آبکاری مس

ستون چهارم، ضخامت مس نهایی

- ستون پنجم، تلرانس نسبی پهنا و فاصله track
- ستون ششم،تلرانس مطلق پهنا و فاصله

لایه های بیرونی

الف- پهنا و فاصله ترک درساخت برای لایه های بیرونی باید مطابق با جدول تصویر ۴.۸ باشد.

ب- پهنا و فاصله ترک درطراحی برای لایه های بیرونی مدارچاپی سخت باید مطابق با مقادیر مشخص شده در جدول ۴.۸ باشد.

ج- نباید در لایه های بیرونی track مسیرکشی شود.

د- در صورتی که در لایه های بیرونی track مسیرکشی شود، نباید از زیر قطعات عبور کند.

۵ هـ- ضخامت فویل مسی برای .باشد ۱۷ میکرومتر باید

pitch fine و- پهنا و فاصله ترک برای fine pitch ممکن است به صورت زیر باشد:

۱ .ترک های fine pitch برای مسیرکشی فوت پرینت قطعه fine pitch استفاده شود.

۲ .طول ترک fine pitch از محل پد لحیم کاری بیش از ۲۰ میلی‌متر نباشد. (شاید برای کاهش همشنوایی!)

۳ .اختلاف ولتاژ در fine pitch نباید بیش از ۳۰V باشد.

ز- در خصوص پهنا و فاصله پدها نیز باید مطالب گفته شده در این بخش رعایت گردد.

Basic Cu [μm]	Plated Cu [μm]	Thickness category [μm]	Pitch	As-manufactured	
				width [μm]	spacing [μm]
17	1x 25	17<Th≤60	fine/normal	120/160	120/160
17	2x 25	60<Th≤70	fine/normal	120/160	120/160
35	1x25	17<Th≤60	normal	160	160
35	2x25	70<Th≤95	normal	240	240
70	1x25	70<Th≤95	normal	240	240

شکل ۴.۸: فاصله ی ترک ها در لایه های بیرونی.

این جدول به شرح حداقل پهنا و فاصله لایه های بیرونی درساخت می پردازد. به عنوان مثال وقتی مس پایه ۱۷ میکرومتر انتخاب شود و ۲۵ میکرومتر هم آبکاری شود، حداقل پهنا و فاصله درساخت برای fine- .است ۱۶.۱۲/۰.۰ میلی‌متر با برابر pitch/normal از آنجایی که ردیف اول در این جدول مربوط به کمترین حد ممکن است، به طور خالصه کمترین فاصله نزدیک شدن track ها به هم و همینطور پهنای track ها در همه حاالت نمی تواند کمتر از ۱۲ میلی‌متر.۰ باشد که در محاسبه امپدانس مسیرها باید به این مورد توجه شود و در مسیرکشی نیز رعایت گردد.

فاصله ی معمول ترک ها در لایه های میانی

الف- پهنا و فاصله ترک ها درساخت باید مطابق با جدول شکل ۵.۸ باشد.

ب- پهنا و فاصله ترک لایه های میانی مدارچاپی rigid درطراحی باید مطابق با مقادیر جدول ۷-۱۳ باشد.

ج- ضخامت مس لایه های میانی نیز مطابق با جدول ۳-۷ تعیین می شود. category(thickness)

Thickness category [μm]	Pitch	As-manufactured	
		width [μm]	spacing [μm]
Th≤17	fine/normal	80/104	96/104
17<Th≤60	normal	120	120
60<Th≤70	normal	160	160
70<Th≤95	normal	240	240

شکل ۵.۸: فاصله ی ترک ها در لایه های بیرونی.

همانطور که در جدول شکل ۵.۸ می بینیم، برای لایه های میانی، با توجه به تلرانس درساخت، پهنا و فاصله ی مسیر نسبت به لایه های بیرونی کمتر است، خصوصا وقتی مس ۱۷ میکرون را داریم، پهنا به ۱۰۴ میکرومتر/۸۰ و فاصله نیز به ۱۰۴ میکرومتر/۹۶ می رسد. بنابراین برای لایه های میانی در محاسبه امپدانس و مسیرکشی می توانیم ترک هایی با پهنا و فاصله ۱ میلی‌متر.۰ داشته باشیم.

۲۱.۲.۸ قوانین پیشگیری از مشکل مدار -باز روی مسیرهای حساس

مشکل مدار-باز یک مسیر در PCB می تواند منجر به از دست رفتن عملکرد، یا رفتار تصادفی، یا خطا بشود که انتشار خطاهای دیگر را در پی دارد. به طور مثال، مشکل مدار-باز یک اتصال در حلقه فیدبک مبدل DC/DC می تواند عملکرد DC/DC را دچار خطا کند و این خطا، باعث انتشار خطا در مداراتی که از این راه تغذیه می شوند، گردد.

مسیرکشی

حساسیت سیگنالها باید مطابق نیازمندیهای ۲a.۱۳.۹ مشخص شوند.

- سیگنالهای بحرانی بهتر است فقط در یک لایه مسیرکشی شوند.

- برای سیگنالهای حساس الزم است دقت بیشتری شود تا استفاده از via ها کاهش یابد.
- برخی نرم افزارهای مسیرکشی خودکار می توانند باعث تعدد ویاهای غیر ضروی گردند.
- برای یک مسیر حساس، از یک via اضافه ثانویه بهتر است برای اتصال بهره بگیریم.
- این کار انجام می گردد تا از بروز خطای مدار-باز در یک اتصال تک بین track ، پد via و لوله via جلوگیری کند.

۲۲.۲.۸ پوشش نهایی روی برد و قطعات

- رساناها در لایه های بیرونی باید با پوشش نهایی، پوشانده شوند.

- رساناهای با فاصله ی fine pitch در لایه های بیرونی باید با پوشش نهایی، پوشانده شوند.

- رساناهای بدون پوشش روی PCB نباید برای ولتاژهای بیش از ۳۰v به کار بروند (این برای پیشگیری از دشارژ های سرگردان در زمانهای سوییچ می باشد).

- رساناهای بدون پوشش ممکن است در ولتاژهای باالی ۳۰v برای پدهای کانکتور به کار رود و این به دلیل عدم امکان نفوذ پوشش در میان پینهای کانکتور است.

- برای فوت پرینت قطعات و کانکتورهایی که نمی توانند پوشش نهایی شوند، فاصله جداسازی باید مطابق با نیازمندیهای مندرج در بند ۱۳.۳ برای X،Y بیرونی بدون پوشش نهایی در نظر گرفته شود. البته این مورد فقط مختص مواردی است که پوشش نهایی وجود دارد اما نمی تواند به ناحیه پدها و آنها را به خوبی پوشش دهد.

- انواع پوشش دربرگیرنده PCB باید شامل پوشش نهایی باشد.

ولتاژ باالی ۳۰V در PCB طراحی شده بدون پوشش نهایی، با روشهای دیگر نظیر ۱۰ potting پوشانده می شوند. کانکتورها نیز می توانند با روش coated یا potted پوشانده شوند.

۲۳.۲.۸ مسیرهای نیازمند به کنترل امپدانس

Microstrip یک track است که به یک صفحه مس تزویج شده است.

Stripline یک track بین دو صفحه مس می باشد. Stripline شیلدینگ بهتری را نسبت به microstrip فراهم می کند، اما نیازمند لایه های بیشتر plane است.

Mictostrip تفاضلی، دو ترک موازی در یک لایه می باشد که به یک صفحه مس تزویج شده اند.

Stripline تفاضلی، دو ترک موازی است که یا به صورت لبه یا به صورت رویهم تزویج دارند و به دو صفحه مس تزویج شده اند.

قواعد عمومی

مسیرهای کنترل امپدانس باید به عنوان موارد بازبینی در مستندات مدارچاپی ثبت گردند.

- مسیرهای کنترل امپدانس باید در یک لایه مسیرکشی شوند.
- ویاها نباید در مسیرهای کنترل امپدانس به کار روند مگر برای اتصال پدهای قطعه
- مسیرهای کنترل امپدانس باید برای ساختار end-to-end طراحی شوند.
- مسیرهای کنترل امپدانس ممکن است برای ساختار multidrop طراحی شوند.
- هنگام استفاده از ساختار multidrop ، محاسبه امپدانس باید شامل بارهای جدا افزوده شده باشد.

مایکرو-استریپ و استریپ-لاین

- Microstrip یا stripline باید برای مسیرهای کنترل امپدانس تنها استفاده شوند.
- Microstrip تفاضلی و stripline تفاضلی باید برای مسیرهای کنترل امپدانس تفاضلی استفاده شوند.
- یک نمونه از stripline تفاضلی تزویج لبه در شکل ۶.۸ نشان داده شده است. این شکل اندازه های مشخصه های مربوطه برای محاسبه امپدانس را نیز نشان می دهد که بدین قرار است:

H : فاصله بین صفحات درساخت

H۱ : ضخامت لایه بین ترک ها و صفحه

W : پهنای بالای ترک

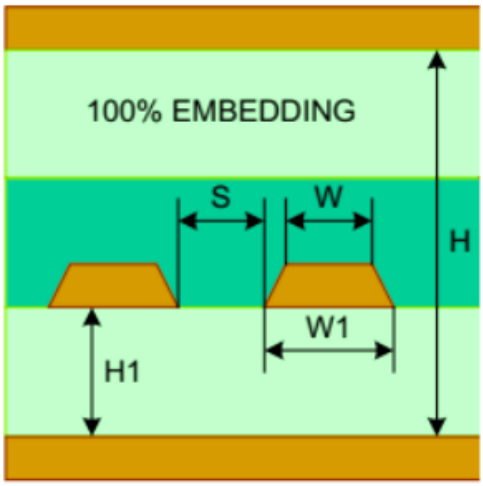
W۱ : پهنای پایین ترک

S : فاصله بین پایین دو track

- ترک های تفاضلی باید به صورت موازی و بدون تغییر فاصله بینشان مسیرکشی شوند مگر در ناحیه پایانی که دو track به ناچار از هم جدا می شوند.

۲۴.۲.۸ مدارچاپی دیجیتال

مدارهای دیجیتال همان هایی اند که اطالعات روی یک خط می تواند تنها دو مقدار ۰ یا ۱ باشند. آنها به طور عمومی در مقابل نویز پایدارتر اند نسبت به خطوط آنالوگ. آنها می توانند حاالت گذرای حساس داشته باشند، اگر سیگنال ورودی monotone نباشد.



شکل ۶.۸: ترک‌های کنترل امپدانس.

حساسیت سیگنالهای دیجیتال بدین صورت مشخص می شود: غیرحساس، حساس و بسیارحساس طرح مدارچاپی باید ناحیه بندی شود. این امر با استفاده از مدیریت منطقه ای انجام می شود. بالک های گوناگون در طرح برد جدا می شوند. این بالک ها می تواند به speed low یا ثابت، آنالوگ، خطوط تغذیه، دیجیتال سریال پر سرعت، و سیگنالهای RF تقسیم بندی شوند.

- مدار سرعت پایین باید از مدار دیجیتال پرسرعت جدا شود.
- سیگنالهای پرسرعت نباید در میان ناحیه کم سرعت مسیرکشی شوند.
- صفحات زمین باید در کاربردهای دیجیتال پرسرعت استفاده شوند.
- صفحات زمین باید در همه کاربردهای دیجیتال به کار گرفته شوند.
- سیگنالهای پرسرعت باید روی صفحات بدون فاصله و شکاف مسیرکشی شوند. (به عنوان نمونه، در لایه های صفحه تغذیه، شکاف و فاصله رخ می دهد).
- ناحیه حلقه سیگنالهای دیجیتال باید به کمترین حد برسد.
- ناحیه حلقه باید کم شود تا نویز جریان کاهش یابد. این امر می تواند با مسیرکشی سیگنال در زیر یا روی صفحه زمین به عنوان مسیربرگشت محقق گردد.
- مدار و قطعات برای سیگنالهای سرعت بالا باید نزدیک به هم باشند.
- طول مسیرکشی در کاربردهای دیجیتال برای سیگنالهای پرسرعت باید کم شود، مگر هنگامی که زمان-بندی استفاده شود.
- کاهش طول مسیرهای سیگنال، خصوصا برای ارتباطات موازی پرسرعت بین قطعات برای کاهش هم-شنوایی، نویز ولتاژ، و انطباق امپدانس انجام می شود. همچنین این کار به فشرده شدن طرح مدارچاپی کمک می کند. تکنولوژی

تعیین می کند این کم کردن چقدر باشد. برای نمونه، زمانهای صعود و نزول از فرکانس بحرانی تر اند. طول ترک های کمتر از طول بحرانی تعیین شده در جدول ۴-۵ از IPC-2251 یک نمونه از کم کردن است. هنگامی که از طول بحرانی فراتر برویم، ترک ها به عنوان خطوط انتقال مهم می شوند. این مورد نیز در بخش d.۴.۴.۲ از IPC-2251 توضیح داده شده است.

حساسیت سیگنالهای دیجیتال

- سیگنالهایی که حساس به تزویج نیستند به عنوان غیر-بحرانی تعیین می شوند. به عنوان مثال سیگنالهای گذرگاه داده و گذرگاه آدرس غیر-بحرانی اند.
- سیگنالهای غیر-بحرانی ممکن است به صورت دسته ای با هم بدون در نظرگرفتن حاالت گذرای بین آنها مسیرکشی شوند.
- حاالت گذرا عبارتند از overshoot ، undershoot ، ringing ، بازتابش خطوط، و هم-شنوایی حاالت گذرا معموال تا سطح قابل قبول پیش از زمان نمونه گیری از بین می روند یا کاهش می یابند.
- از تزویج مستقیم بین دسته های سیگنالهای غیر-بحرانی باید پرهیز شود.
- سیگنالهایی که نیاز به یک شکل موج تک نوا در آستانه ولتاژ در گیرنده ها دارند باید به ۱۱ عنوان "بحرانی" تعیین شوند. به عنوان نمونه سیگنالهای کالک و سیگنالهای write بحرانی اند.
- مسیرکشی سیگنالهای حساس باید با حفظ یکپارچگی سیگنال در پایان باشد. ابزارهای یکپارچگی سیگنال می تواند برای شبیه سازی رفتار مدارچاپی پس از مسیرکشی استفاده شوند.
- یکپارچگی سیگنالهای بحرانی باید با تست الکتریکی مدارچاپی بازبینی شوند.
- سیگنالهایی که کارایی مدار را تحت تاثیر قرار می دهند به عنوان "فوق-بحرانی" مشخص ، ringing یا تاخیر انتشار می ۱۲ می شوند. این کارایی همان پاسخ نویز، هم-شنوایی، لغزش باشد. به عنوان نمونه سیگنالهای PLL ، ولتاژ پایین و سیگنالهای LVDS فرکانس باال، فوق بحرانی اند.
- سیگنالهای فوق بحرانی، باید بدون هم جواری با دیگر سیگنالها مسیرکشی شوند یا اینکه داشته باشند. ۱۳ حفاظ جدا کردن دیگر سیگنالهای فوق-بحرانی برای پیشگیری از هم شنوایی و نویز انجام می شود. جدا کردن می تواند با صفحات و ردیف هایی از via های حفاظ محقق شود. یکپارچگی سیگنالهای فوق-بحرانی باید با تست الکتریکی مدارچاپی بازبینی شود.
- مدارات آنالوگ و دیجیتال روی یک مدارچاپی باید از یکدیگر با مدیریت منطقه ای و حفاظ گذاری جدا شوند. صفحات مدارات آنالوگ و دیجیتال باید از هم جدا باشند. نتیجه یکی بودن صفحات آنالوگ و دیجیتال می تواند تزویج AC و

نویز دیجیتال بر روی سیگنال آنالوگ باشد.

- صفحات مدارهای آنالوگ و دیجیتال می تواند با هم یکی شوند، البته در مواردی که جدا کردن صفحات، ضرر بیشتری بر کارآیی یا عملکرد یا EMC داشته باشد.
- در مواردی که صفحات آنالوگ و دیجیتال با هم یکی شوند، پیوندزدایی منبع تغذیه باید اعمال شود.
- سیگنالهای دیجیتال نباید در منطقه آنالوگ مسیرکشی شوند. مسیرکشی دقیق سیگنالهای آنالوگ و دیجیتال برای DAC ، ADC و MUX مهم است.

۲۵.۲.۸ پوشش نهایی مدارچاپی

متالیزه

الف- لایه های رسانا یا با روکش قلع-سرب و یا روکش طال روی نیکل آبکاری شوند.

ب- بردهایی که نیاز به پوشش نهایی ترکیبی دارند، باید این نواحی از هم حداقل ۵ میلی‌متر.۰ فاصله داشته باشند.

پوشش لحیم کاری

پوشش لحیم کاری از دیرباز برای پیشگیری از جاری شدن لحیم از نواحی آرایه ای و bone-dog به ویاهای سرتاسری استفاده شده است.

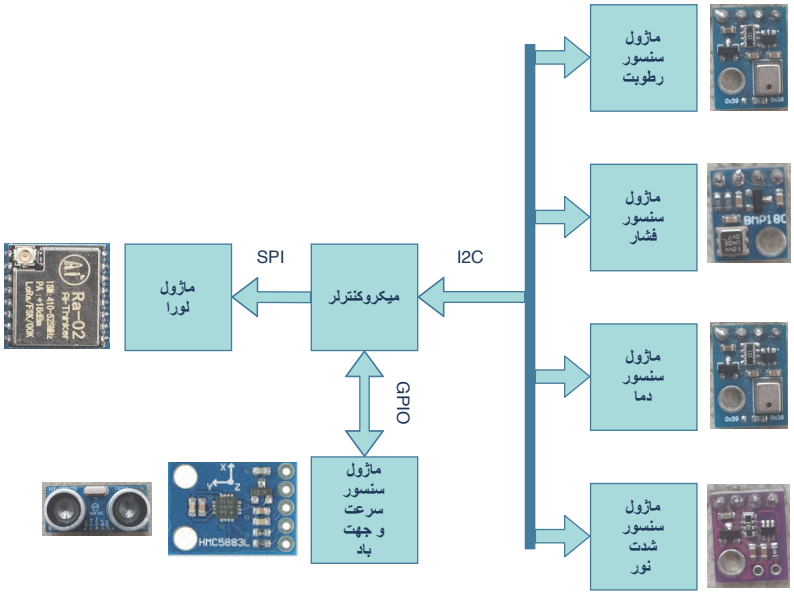
۲۶.۲.۸ برخی اصطالحات فنی

Etching : یک مرحله در ساخت مدارچاپی است که در آن، مس های اضافی از روی برد زدوده می شوند تا به طرح مسی که باید بماند برسیم. عموما etching به صورت شیمیایی و با اسیدکاری انجام می شود، اما گاهی اوقات برای بردهای نمونه از cnc و milling نیز برای این کار استفاده می کنند.

ساختار prepreg محور: ساختاری که تعداد prepreg ها از core ها بیشتر باشد، یا به عبارتی چون تعداد لایه های مس زوج است، تعداد لایه های دی الکتریک فرد است، در این ساختار، نصف تعداد لایه ها به عالوه ۱ می شود. ing Copper بدین معنی است که بر روی طرح مس هایی ایجاد کنید که به جایی وصل نیستند و فقط برای بهبود ساخت مدارچاپی کاربرد دارند.

۲۷.۲.۸ نحوه ی طراحی بخش سنسور

به طور کلی هدف از طراحی بخش سنسور، دریافت داده‌های مربوط به سنسورهای دماسنج، رطوبت سنج، فشارسنج، شدت نور، قطب نمای دیجیتال و آلتراسونیک (به منظور اندازه گیری سرعت باد) و ارسال آنها توسط لورا به سمت ایستگاه می‌باشد. بلوک دیاگرام کلی بخش سنسور در شکل ۷.۸ نمایش داده شده است.



شکل ۷.۸: بلوک دیاگرام بخش سنسور.

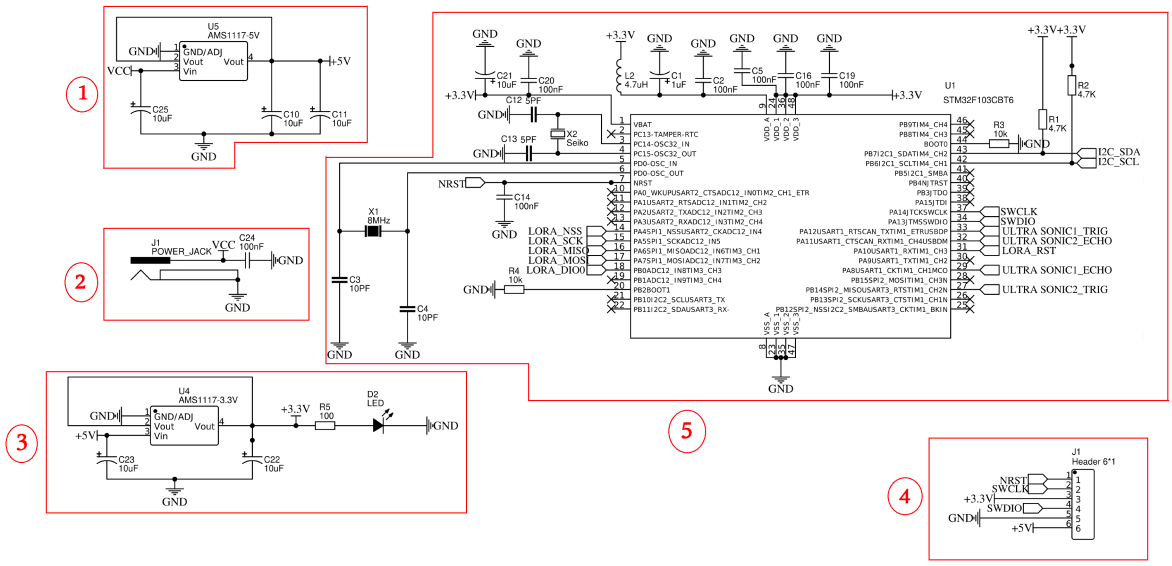
با توجه به بلوک دیاگرام شکل شکل ۷.۸ سنسورهای دماسنج، رطوبت سنج، فشار سنج، شدت نور و قطب نمای دیجیتال توسط پروتکل I2C، ماژول‌های آلتراسونیک توسط پین‌های خروجی و ورودی (GPIO) و ماژول لورا توسط پروتکل SPI به میکروکنترلر متصل می‌شوند.

۲۸.۲.۸ نقشه ی شماتیک بخش سنسور

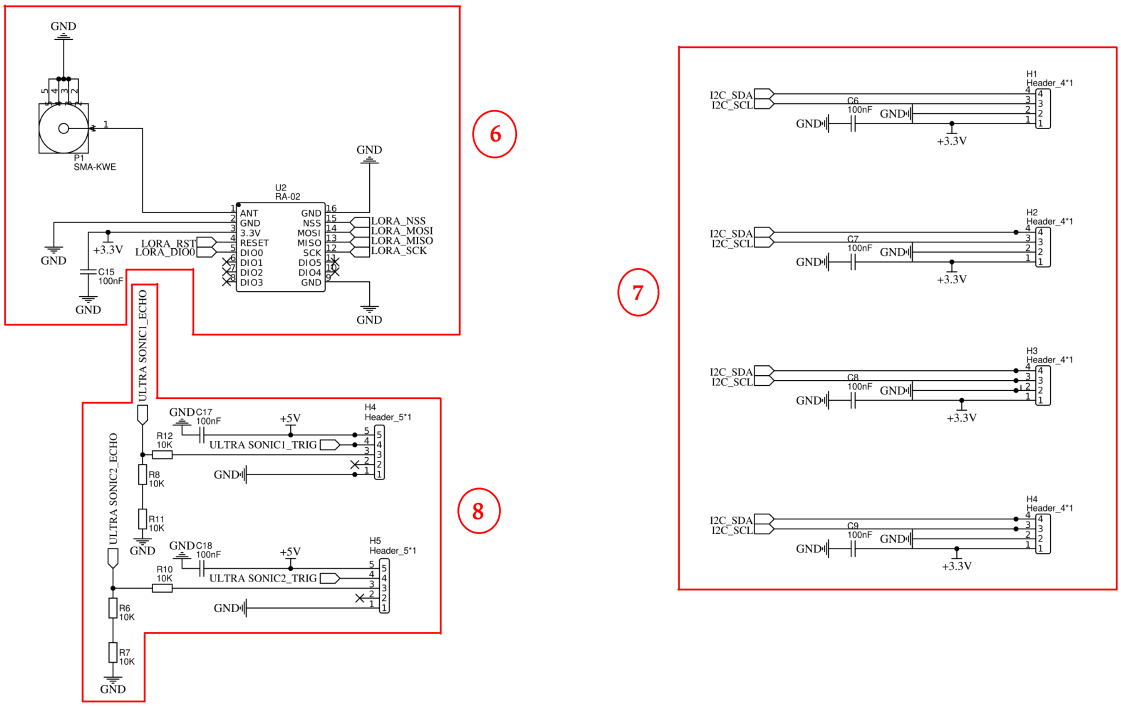
۱- این بخش، اولین بخش مربوط به قسمت تغذیه (پاور) سمت سنسور می‌باشد. در این بخش از یک رگولاتور ۵ ولت برای تغذیه‌ی ماژول آلتراسونیک استفاده شده است. ورودی رگولاتور، تغذیه‌ای است که از بیرون به مدار اعمال می‌شود که حداقل می‌بایست ۶ ولت باشد و خروجی آن ولتاژ رگوله شده‌ی ۵ ولت می‌باشد. خازن‌های قرار داده شده در ورودی و خروجی رگولاتور به منظور کاهش ریپل ولتاژ ورودی به رگولاتور و کم کردن نویز خروجی رگولاتور استفاده شده است.

۲- این بخش، دومین بخش مربوط به قسمت تغذیه (پاور) سمت سنسور می‌باشد. در این بخش از یک جک پاور برای

اتصال تغذیه‌ی ورودی به مدار استفاده شده است.



شکل ۸.۸: تصویر شماتیک برد مدارچاپی بخش سنسور قسمت اول.



شکل ۹.۸: تصویر شماتیک برد مدارچاپی بخش سنسور قسمت دوم.

۳- این بخش، سومین بخش مربوط به قسمت تغذیه (پاور) سمت سنسور می‌باشد. این بخش همانند بخش ۱ می‌باشد. تنها تفاوت آن در ولتاژ خروجی رگولاتور است که ۳٫۳ ولت می‌باشد. ورودی این رگولاتور ولتاژ خروجی رگولاتور بخش

اول می‌باشد و همچنین در خروجی این رگولاتور یک دیود نورانی قرار داده شده است که با متصل کردن تغذیه‌ی ورودی روش می‌شود.

۴- این بخش که یک هدر ۱x۶ می‌باشد برای پروگرم و دیباگ کردن میکروکنترلر قرار داده شده است.

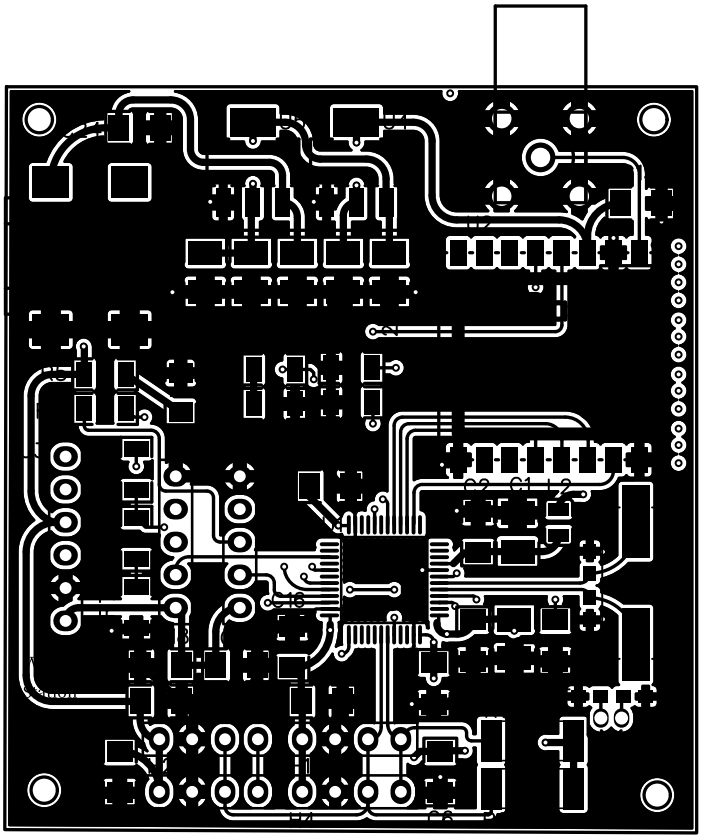
۵- در این بخش که اصلی ترین قسمت مدار می‌باشد از خازن‌های ۱۰۰ نانوفاراد و ۱ میکروفاراد به منظور کاهش نویزهای فرکانس بالا و پایین استفاده شده است. همچنین در تغذیه ی ورودی میکروکنترلر علاوه بر این خازن‌ها از سلف به منظور کاهش ریپل جریان ورودی استفاده شده است. در این بخش از دو عدد کیرستال که یکی برای مدار اصلی و یکی برای قسمت RTC استفاده شده است که خازن‌های آن مطابق با دیتاشیت میکروکنترلر انتخاب شده است. همانطور که در شکل مشاهده می‌شود قسمت دیتا و کلاک واحد I2C با مقاومت‌های ۴/۷ کیلو پول-آپ شده است.

۶- این بخش مربوط به ماژول لورا می‌باشد. در این قسمت خازن برای کاهش نویز ورودی به ماژول استفاده شده است و همچنین یک کانکتور برای اتصال آنتن به ماژول قرار داده شده است.

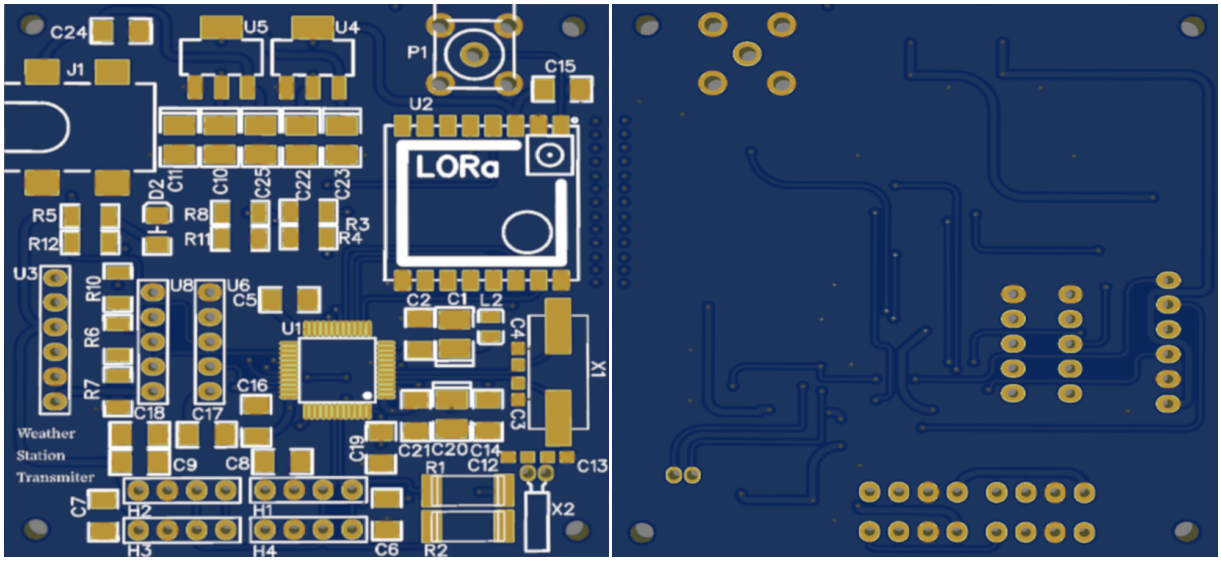
۷- در این بخش از چهار عدد هدر ۱x۴ به منظور اتصال سنسورها به مدار استفاده شده است. همچنین از خازن برای کاهش نویز ورودی به مدار و استفاده شده است.

۸- در این بخش از دو عدد هدر ۱x۵ به منظور اتصال ماژول آلتراسونیک به مدار استفاده شده است. همچنین از خازن برای کاهش نویز ورودی به مدار و از تقسیم مقاومتی برای اتصال ماژول به میکروکنترلر استفاده شده است.

تصویر بردمدارچاپی طراحی‌شده برای سمت سنسور این پروژه در شکل ۱۱.۸ آمده است. همچنین تصاویر شماتیک آن در شکل‌های ۸.۸ و ۹.۸ آمده است.



شکل ۸.۱۰: تصویر برد مدار چاپی طراحی شده برای بخش سنسور.



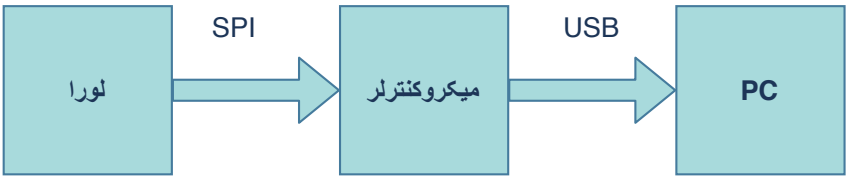
(آ) پشت برد بخش سنسور

(ب) روی برد بخش سنسور

شکل ۸.۱۱: تصاویر بردمدارچاپی طراحی‌شده برای بخش سنسور.

۳.۸ نحوه ی طراحی بخش ایستگاه

به طور کلی هدف از طراحی بخش ایستگاه، دریافت داده‌های ارسالی از ماژول لورا و مانیتور کردن آنها در PC می‌باشد. بلوک دیاگرام کلی بخش ایستگاه در شکل ۱۲.۸ نمایش داده شده است.



شکل ۱۲.۸: بلوک دیاگرام بخش ایستگاه.

با توجه به بلوک دیاگرام شکل ۱۲.۸ داده‌های دریافتی از ماژول لورا توسط میکروکنترلر دریافت شده و توسط پروتکل USB به رایانه فرستاده می‌شود و رایانه نیز توسط اپ طراحی شده دیتاهای دریافت شده را دریافت و مانتیور می‌کند.

۱.۳.۸ نقشه ی شماتیک بخش ایستگاه

۱- در این بخش که اصلی ترین قسمت مدار می‌باشد از خازن‌های ۱۰۰ نانوفاراد و ۱ میکروفاراد به منظور کاهش نویزهای فرکانس بالا و پایین استفاده شده است. همچنین در تغذیه ی ورودی میکروکنترلر علاوه بر این خازن‌ها از سلف به منظور کاهش ریپل جریان ورودی استفاده شده است. در این بخش از دو عدد کیرستال که یکی برای مدار اصلی و یکی برای قسمت RTC استفاده شده است که خازن‌های آن مطابق با دیتاشیت میکروکنترلر انتخاب شده است. همانطور که در شکل مشاهده می‌شود قسمت دیتا و کلاک واحد I2C با مقاومت‌های ۴٫۷ کیلو پول-آپ شده است.

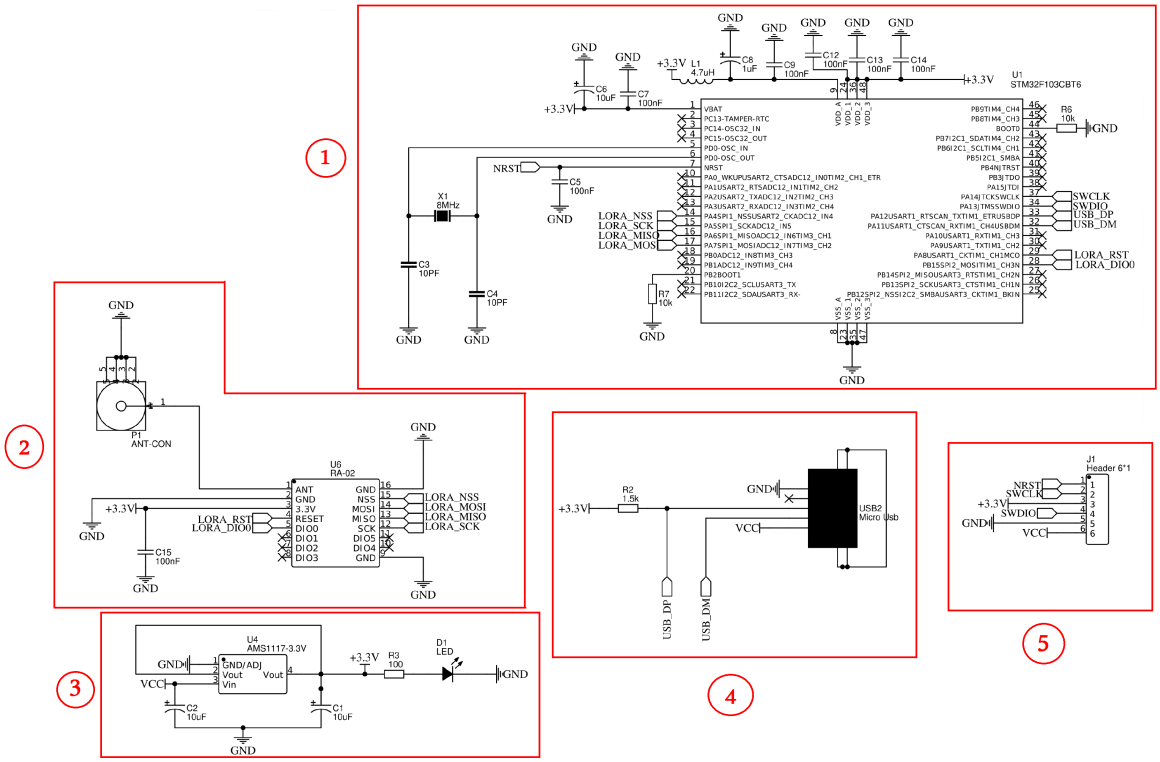
۲- این بخش مربوط به ماژول لورا می‌باشد. در این قسمت خازن برای کاهش نویز ورودی به ماژول استفاده شده است و همچنین یک کانکتور برای اتصال آنتن به ماژول قرار داده شده است.

۳- در این بخش از یک رگولاتور ۳٫۳ ولت برای تغذیه‌ی ماژول آلتراسونیک استفاده شده است. ورودی رگولاتور، تغذیه‌ای است که از بیرون به مدار اعمال می‌شود که حداقل می‌بایست ۴٫۳ ولت باشد و خروجی آن ولتاژ رگوله شده‌ی ۳٫۳ ولت می‌باشد. خازن‌های قرار داده شده در ورودی و خروجی رگولاتور به منظور کاهش ریپل ولتاژ ورودی به رگولاتور و کم کردن نویز خروجی رگولاتور استفاده شده است. همچنین در خروجی این رگولاتور یک دیود نورانی قرار داده شده است که با متصل کردن تغذیه‌ی ورودی روش می‌شود.

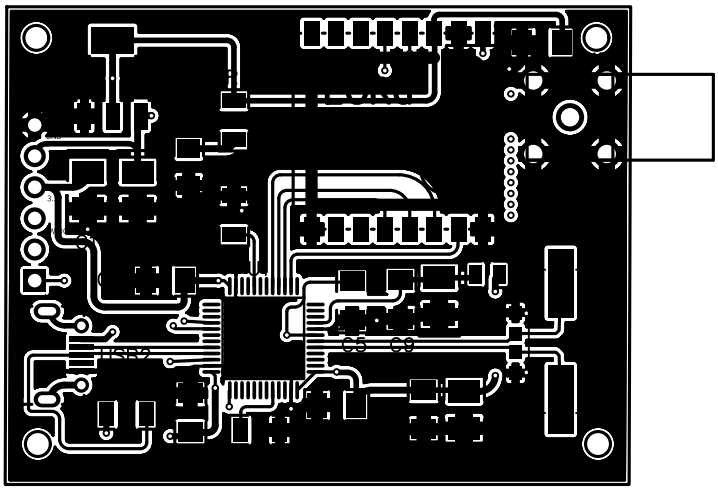
۴- در این بخش از یک کانکتور MICRO USB برای برقراری ارتباط بین میکروکنترلر و رایانه استفاده شده است.

۵- این بخش که یک هدر ۱x۶ می‌باشد برای پروگرم و دیباگ کردن میکروکنترلر قرار داده شده است.

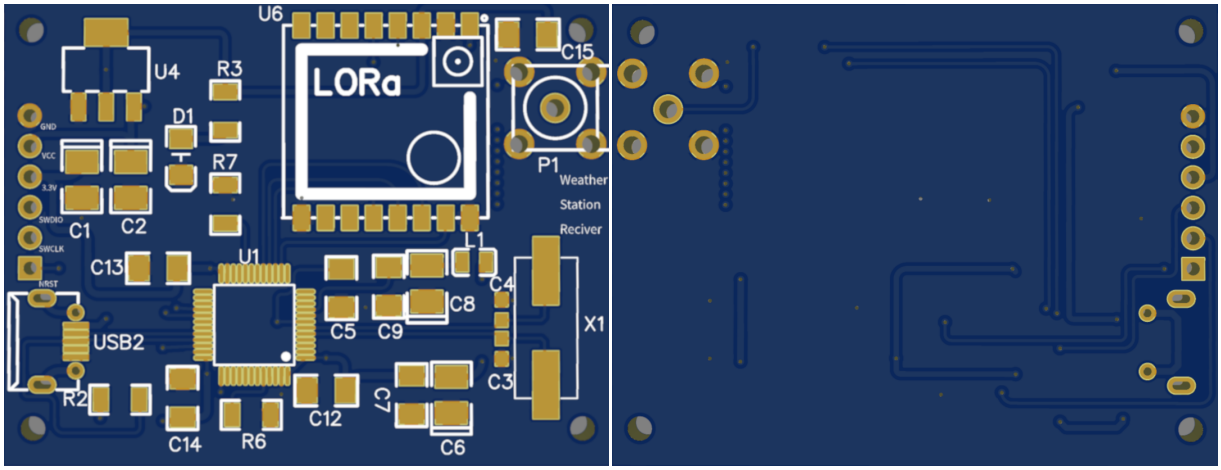
تصویر بردمدارچاپی طراحی‌شده برای سمت ایستگاه این پروژه در شکل ۱۵.۸ آمده است. همچنین تصاویر شماتیک آن در شکل ۱۳.۸ آمده است.



شکل ۱۳.۸: تصویر شماتیک برد مدارچاپی سمت ایستگاه.



شکل ۱۴.۸: تصویر برد مدار چاپی طراحی شده برای بخش ایستگاه.



(آ) پشت برد بخش ایستگاه

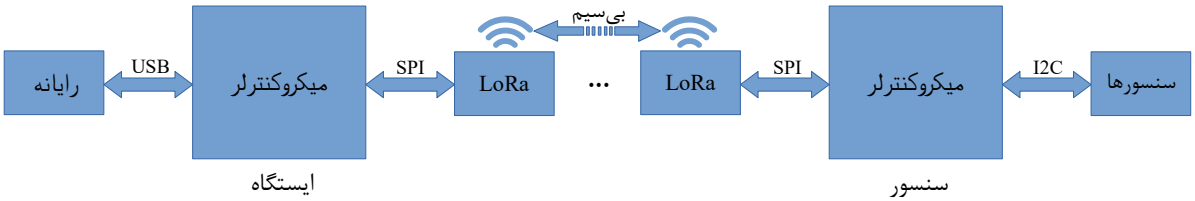
(ب) روی برد بخش ایستگاه

شکل ۱۵.۸: تصاویر بردمدارچاپی طراحی‌شده برای بخش ایستگاه.

فصل ۹

تشریح پروژه

۱.۹ مقدمه



شکل ۱.۹: بلوک دیاگرام اجزای سیستم و نحوه ارتباط اجزای مختلف با یکدیگر.

این سیستم به دو دستگاه اصلی تقسیم می‌شود، یک دستگاه جهت جمع‌آوری اطلاعات جوی برروی یک میله در ارتفاع ۱۰ متری سطح زمین قرار می‌گیرد و اطلاعات جوی نظیر دما، رطوبت، فشار، شدت نور، سرعت و جهت باد را از سنسورهای مربوطه جمع‌آوری کرده و به‌صورت بی‌سیم به دستگاه دیگر، که در ایستگاه اصلی قرار دارد، مخابره می‌کند؛ سپس اطلاعات دریافت شده در سمت دستگاه دوم جهت ثبت و ذخیره به رایانه منتقل می‌شود. در اینجا به دستگاه اول که وظیفه جمع‌آوری اطلاعات جوی را دارد سنسور و دستگاه دوم که وظیفه دریافت اطلاعات مخابره

شده و انتقال به رایانه را دارد ایستگاه می‌گوییم. بلوک دیاگرام کلی این سیستم و نحوه ارتباط بخش‌های مختلف با یکدیگر در شکل ۱.۹ آمده است.

عملکرد کلی سیستم جمع‌آوری داده‌ها در سمت سنسور، ارسال اطلاعات از طریق لورا به سمت ایستگاه و نمایش اطلاعات دریافت شده در سمت ایستگاه برروی رایانه است. به‌طورکلی این سیستم به دو بخش سنسور و ایستگاه تقسیم می‌شود که بخش ایستگاه خود به دو بخش دستگاه گیرنده و برنامه دسکتاپ (Desktop) قابل‌تقسیم است.

۲.۹ سنسور

فلوچارت فرآیندهای سمت سنسور در تصویر ۲.۹ قابل مشاهده است شرح آن به قرار زیر است:

- بعد از روشن شدن دستگاه و فعالسازی بخش‌های موردنیاز (peripherals)، میکروکنترلر از طریق I^2C با سنسور BMP180 ارتباط برقرار می‌کند و ضرایب کالیبراسیو را از سنسور BMP180 می‌خواند این ضرایب برای محاسبه دما و فشار مورداستفاده قرار می‌گیرند [۹].

- میکروکنترلر با برقراری ارتباط از طریق I^2C با سنسور MAX44009 مقدار رجیستر تنظیمات این سنسور را 0x00 تنظیم می‌کند. در این حالت سنسور در پایین‌ترین سطح مصرف توان خوده قرار می‌گیرد و هر ۸۰۰ میلی‌ثانیه یکبار میزان شدت نور را اندازه‌گیری می‌کند [۱۰].

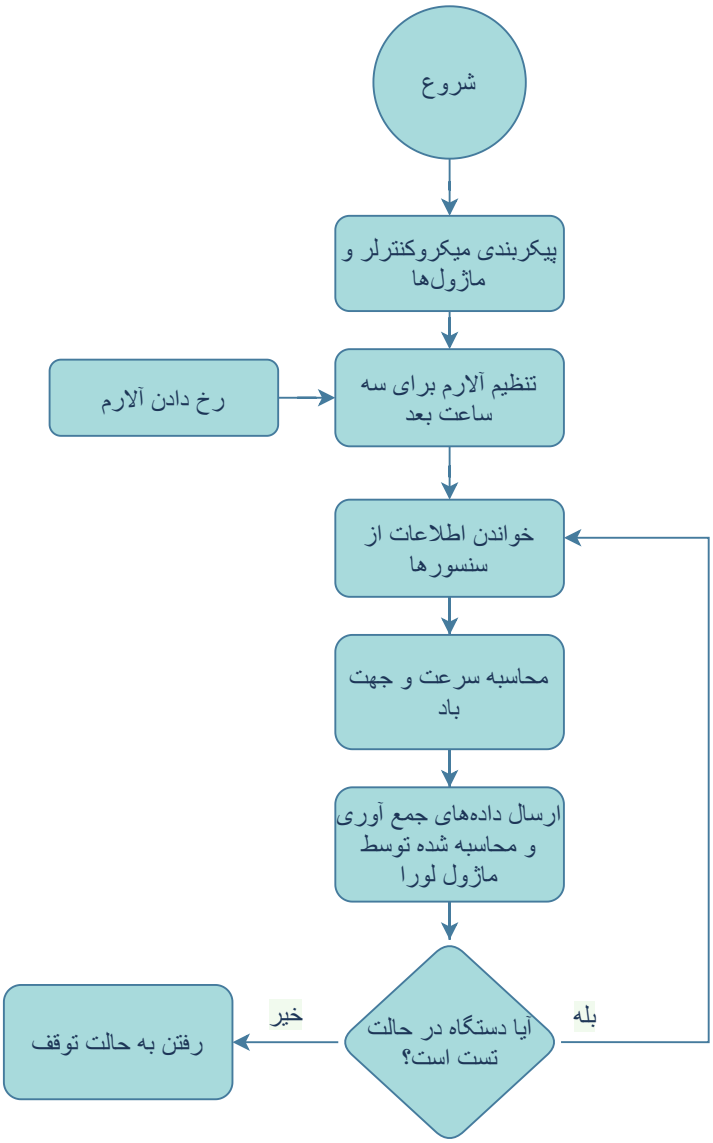
- با برقراری ارتباطی از طریق I^2C با سنسور HMC5883L و تنظیم رجیستر تنظیمات، سنسور در حالت آماده‌به‌کار و نرخ نمونه‌برداری ۵۰ هرتز قرار می‌گیرد [۱۱].

- رجیسترهای تنظیمات ماژول LoRa با برقراری ارتباط از طریق SPI تنظیم‌شده و این ماژول در حالت آماده‌به‌کار قرار می‌گیرد. فرکانس این ماژول روی ۴۳۳ مگاهرتز، توان آن روی ۲۰ dBm، ضریب بخش آن روی ۱۰ و پهنای باند آن روی ۳۱٫۲ کیلوهرتز تنظیم می‌گردد (به‌منظور دریافت اطلاعات ارسالی در سمت ایستگاه نیز دقیقاً همین تنظیمات فرکانس، پهنای باند و ضریب پخش باید اعمال شوند) [۱۲].

- برنامه وارد حلقه اصلی کار خود شده و دما و فشار را با استفاده از سنسور BMP180 اندازه‌گیری می‌کند.

- پس از اندازه‌گیری شدت نور با استفاده از سنسور MAX44009 [۱۰] و اندازه‌گیری رطوبت هوا به‌وسیله سنسور AHT10 [۱۳]، جهت جغرافیایی توسط سنسور QMC5883L [۱۱] به‌دست می‌آید.

- سرعت باد روی دو محور x و y با استفاده از سنسور HCSR05 و با توجه به رابطه‌ی ۳.۷ محاسبه می‌شود. سپس زاویه و شدت باد با توجه به‌سرعت باد روی هر دو محور با توجه به رابطه ۴.۷ به‌دست می‌آید.
- اطلاعات جمع‌آوری و محاسبه‌شده از طریق ماژول لورا برای گیرنده سمت ایستگاه ارسال می‌شود.
- میکروکنترلر و سنسورها در حالت توقف[1] قرار داده می‌شوند و پس از سه ساعت با رخ‌دادن آلارم[2] میکروکنترلر از حالت توقف خارج شده و فرآیند دریافت و ارسال داده‌ها را تکرار می‌کند.
- پس از هر بار خارج شدن از حالت توقف، آلارم بعدی برای سه ساعت بعد تنظیم می‌شود.



شکل ۲.۹: فلوچارت برنامه سمت سنسور.

[1] Stop
[2] Alarm

۱.۲.۹ نحوه خواندن دما و فشار از سنسور BMP180

در ابتدا لازم است ضرایب کالیبراسیون از سنسور BMP180 دریافت شود. روش ارتباط این ماژول با میکروکنترلر از طریق پروتکل I2C است. جهت خواندن مقادیر ضرایب کالیبراسیون باید آدرس MSB رجیستر متناظر با هر ضریب را با مود Read به آدرس ماژول (0x77) ارسال نماییم و ماژول در پاسخ ۱۶ بیت داده را به ما بر می‌گرداند که مقدار ضریب کالیبراسیون مورد نظر است. آدرس رجیسترهای ضرایب کالیبراسیون در تصویر شکل ۳.۹ آمده است.

	BMP085 reg adr	
Parameter	MSB	LSB
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

شکل ۳.۹: آدرس رجیستر ضرایب کالیبراسیون سنسور BMP180.

خواندن دما

به جهت خواندن مقدار دما باید مقدار 0x2E را در رجیستر کنترل 0xF4 بنویسیم. با این کار سنسور در حالت اندازه‌گیری دمای محیط قرار می‌گیرد و پس از حداکثر ۴٫۵ میلی ثانیه این داده برای خواندن آماده می‌شود. این کار با ارسال مقدار آدرس رجیستر کنترل یعنی 0xF4 در مود Write به آدرس ماژول (0x77) و پس از آن ارسال مقدار کنترل مورد نظر یعنی 0x2E انجام می‌شود. پس از اندازه‌گیری دما مقدار اندازه‌گیری شده در رجیستری با آدرس 0xF6(MSB) و 0xF7(LSB) در ماژول ذخیره می‌شود. به جهت خواندن مقدار اندازه‌گیری شده پس از حداقل ۴٫۵ میلی ثانیه بعد از ارسال مقدار کنترلر 0x2E به ماژول باید مقدار MSB رجیستر داده‌هایعنی 0xF6 به آدرس سنسور (0x77) در مود Read فرستاده شود تا ماژول در پاسخ ۱۶ بیت داده‌های اندازه گیری شده را که در این حالت مقدار دمای محیط است

را به ما برگرداند (ابتدا مقدار رجیستر MSB و سپس مقدار LSB برگشت داده می‌شود). سپس مقدار دقیق دمای محیط با دقت ۱.۰ درجه سانتی گراد از طریق فرمول زیر قابل محاسبه خواهد بود:

$$UT = MSB << 8 + LSB$$
$$X1 = (UT - AC6) * AC5/2^{15}$$
$$X2 = MC * 2^{11}/(X1 + MD)$$
$$B5 = X1 + X2$$
$$T = (B5 + 8)/2^4$$

که $AC6$، $AC5$، MD و MC ضرایب کالیبراسیون، MSB مقدار MSB رجیستر داده‌های دریافت شده، LSB مقدار LSB رجیستر داده‌های دریافت شده و T دمای محیط بر حسب سانتی‌گراد خواد بود.

خواندن فشار

طریقه خواندن فشار از ماژول نیز مشابه خواندن دما است. در ابتدا لازم باید مقدار رجیستر کنترل ماژول را با مقدار مورد نظر (که در اینجا مقدار متناظر با دریافت فشار است) پر کرد و پس از زمانی مشخص داده‌های اندازه گیری شده را از رجیسترهای داده ماژول خواند. در حالت خواندن فشار بر خلاف خواندن دما ماژول حالت‌های نمونه‌بردادی مختلفی را ارائه می‌کند که باید بر اساس نیاز مقدار مربوط به هر مود نمونه برداری که مد نظر بود را در رجیستری کنترل نوشت تا ماژول در حالت نمونه برداری مورد نظر قرار گیرد. با افزایش نرخ نمونه برداری دقت مقدار اندازه‌گیری شده بیشتر شده ولی زمانی که ماژول صرف اندازه‌گیری می‌کند نیز افزایش می‌یابد. در جدول ۲.۹ مقدار رجیستر کنترلی متناظر با هر مود نمونه برداری و حداکثر زمان لازم برای انجام نمونه برداری آورده شده است.

جدول ۱.۹: حالت‌های نمون برداری فشار ماژول BMP180.

حالت نمونه برداری	مقدار رجیستر کنترلی	حداکثر زمان اندازه‌گیری (میلی‌ثانیه)
۰	0x34	۴٫۵
۱	0x74	۷٫۵
۲	0xB4	۱۳٫۵
۳	0xF4	۲۵٫۵

در این پروژه استفاده از مود نمونه برداری ۰ برای ما کفایت می‌کند. برای محاسبه فشار اندازه گیری دمای هوا نیز

لازم است پس ابتدا لازم است قبل از درخواست اندازه گیری فشار دمای محیط توسط روشی که قبل تر گفته شد اندازه گیری شود. به جهت خواندن مقدار فشار باید مقدار 0x34 را در رجیستر کنترل 0xF4 بنویسیم (مود ۰). با این کار سنسور در حالت اندازه‌گیری فشار قرار می‌گیرد و پس از حداکثر ۴٫۵ میلی ثانیه (با توجه به جدول ۲.۹ و در مود ۰) این داده برای خواندن آماده می‌شود. این کار با ارسال مقدار آدرس رجیستر کنترل یعنی 0xF4 در مود Write به آدرس ماژول (0x77) و پس از آن ارسال مقدار کنترل مورد نظر یعنی 0x34 انجام می‌شود. پس از اندازه‌گیری فشار مقدار اندازه‌گیری شده در رجیستری با آدرس 0xF6(MSB)، 0xF7(LSB) و 0xF8(XLSB) در ماژول ذخیره می‌شود. به جهت خواندن مقدار اندازه‌گیری شده پس از حداقل ۴٫۵ میلی ثانیه بعد از ارسال مقدار کنترل 0x34 (در مود ۰) به ماژول باید مقدار MSB رجیستر داده‌هایعنی 0xF6 به آدرس سنسور (0x77) در مود Read فرستاده شود تا ماژول در پاسخ ۱۶ بیت داده‌های اندازه گیری شده را به ما برگرداند (ابتدا مقدار رجیستر MSB و سپس مقدار LSB برگشت داده می‌شود) و سپس باید مقدار XLSB رجیستر داده‌هایعنی 0xF8 به آدرس سنسور (0x77) در مود Read فرستاده شود تا ماژول در پاسخ ۸ بیت داده را به ما برگرداند. سپس مقدار دقیق فشار محیط با دقت ۱ پاسکال از طریق فرمول زیر قابل محاسبه خواهد بود:

$$UP = (MSB << 16 + LSB << 8 + XLSB) >> (8 - oss)$$

$$B6 = B5 - 4000$$

$$X1 = (B2 * (B6 * B6/2^{12}))/2^{11}$$

$$X2 = AC2 * B6/2^{11}$$

$$X3 = X1 + X2$$

$$B3 = ((AC1 * 4 + X3) << oss + 2)/4$$

$$X1 = AC3 * B6/2^{13}$$

$$X2 = (B1 * (B6 * B6/2^{12}))/2^{16}$$

$$X3 = ((X1 + X2) + 2)/2^{2}$$

$$B4 = AC4 * (unsigendlong)(X3 + 32768)/2^{15}$$

$$B7 = ((unsignedlong)UP - B3) * (50000 >> oss)$$

$$if(B7 < 0x80000000)\{p = (B7 * 2)/B4\}$$

$$else\{p = (B7/B4) * 2\}$$
$$X1 = (p/2^8) * (p/2^8)$$
$$X1 = (X1 * 3038)/2^{16}$$
$$X2 = (-7357 * p)/2^{16}$$
$$p = p + (X1 + X2 + 3791)/2^{16}$$

که $B5$ مقدار بدست آمده از روابط محاسبه دما است که در بخش قبلی بیان شده است، oss عدد مود نرخ نمونه برداری (از ۰ تا ۳ مطابق جدول ۲.۹)، $AC1$، $AC2$، $AC3$، $AC4$، $B1$ و $B2$ ضرایب کالیبراسیون، MSB مقدار MSB رجیستر داده‌های دریافت شده، LSB مقدار LSB رجیستر داده‌های دریافت شده ، $XLSB$ مقدار رجیستر داده XLSB و در نهایت p مقدار فشار اندازه‌گیری شده بر حسب پاسکال خواهد بود.

۲.۲.۹ نحوه خواندن شدت نور از سنسور MAX44009

در ابتدا پس از روشن شدن ماژول لازم است که حالت کاری ماژول و تنظیمات اولیه آن انجام شود. این کار با نوشتن رجیستر کنترلی این ماژول با آدرس 0x02 انجام می‌شود. بیت هشتم این رجیستر به بین CONT معروف است. این بیت مسئول تعیین مود کارکرد است به طوری که اگر مقدار 0 در آن نوشته شود ماژول هر ۸۰۰ میلی‌ثانیه مقدار نور محیط را اندازه‌گیری کرده و در رجیتسرهای خود ذخیره می‌کند و اگر مقدار 1 در آن نوشته شده باشد ماژول بدون وقفه و پشت‌سرهم اندازه‌گیری نور را انجام می‌دهد. برای این پروژه استفاده از مود 0 مناسب تر است و ازین مود استفاده خواهیم کرد. بیت هفتم این رجیستر به بیت MANUAL معروف است و در صورتی که با 1 پر شود کاربر قادر به مشخص کردن مقادیر CDR و TIM[2:0] خواهد بود و در غیر این صورت خوده سیستم در مورد این مقادیر تصمیم گیری می‌کند. مقدار CDR که بیت چهارم رجیستر کنترل است، مشخص کننده مقدار مقسم مقادیر خوانده شده از ADC است و در صورتی که با 0 پر شده باشد هیچ عملیات تقسیمی رخ نداده و مقادیر ADC عینا در رجیسترها کپی می‌شوند و در صورتی که با مقدار 1 پر شده باشد مقادیر تقسیم بر ۸ می‌شوند (در مواقعی که نور محیط بسیار زیاد است کاربرد دارد). مقدار TIM[2:0] نیز که بیت اول تا سوم رجیستر کنترل را به خود اختصاص داده بیان کننده زمان نمونه برداری و اذغام است که مقادیر متناظر با حالت‌های مختلف قابل انتخاب برای این بیت‌ها در جدول ۲.۹ آمده است. در حالت خودکار زمان‌هایی بین ۱۰۰ تا ۸۰۰ میلی ثانیه انتخاب می‌شوند و در حالت دستی میتوان از ۲۵.۶ میلی ثانیه تا ۸۰۰ میلی ثانیه را انتخاب کرد. حالت ۸۰۰ میلی ثانیه مناسب محیط با نور بسیارکم و حالت ۲۵.۶ میلی ثانیه

مناسب محیط‌هایی با نور بسیار زیاد است.

جدول ۲.۹: مقادیر بیت‌های TIM[2:0].

TIM[2:0]	زمان نمونه برداری و ادغام (میلی‌ثانیه)
000	۸۰۰
001	۴۰۰
010	۲۰۰
011	۱۰۰
100	۵۰
101	۲۵
110	۱۲٫۵
111	۶٫۲۵

در این پروژه به دلیل ثابت نبودن شرایط محیطی انتخاب حالت خودکار (مقدار 0 برای MANUAL) مناسب ترین روش ممکن است. با این کار مقادیر CDR و TIM[2:0] توسط خوده ماژول و با توجه به شرایط محیطی در هر اندازه گیری مشخص می‌شود. پس در این پروژه لازم است پس از روشن شدن ماژول مقدار 0x00 در رجیستر کنترل به آدرس 0x02 نوشته شود. این کار با ارسال آدرس رجیستر (0x02) در مود Write به آدرس ماژول (0x4A) و پس از آن ارسال مقدار 0x00 انجام می‌شود. با این کار ماژول ماکسیموم هر ۸۰۰ میلی ثانیه یک بار دیتاهای جدید را اندازه‌گیری کرده و در رجیسترهایی با آدرس 0x03(MSB) و 0x04(LSB) ذخیره می‌کند به جهت خواندن این مقادیر لازم است آدرس رجیستر به آدرس ماژول در مود Read فرستاده شود تا ماژول در پاسخ ۸ بیت دیتای این رجیسترها را به ما برگرداند. پس از دریافت این مقادیر شدت نور بر حسب لوکس از طریق روابط زیر قابل محاسبه خواهد بود:

$$Lux = (2^{Exponent} \times Mantissa) \times 0.045$$

که مقادیر $Exponent$ و $Mantissa$ از روی مقادیر رجیسترهای 0x03(MSB) و 0x04(LSB) و از طریق روابط زیر و جدول ۳.۹ بدست می‌آیند.

$$Exponent = 8 \times E3 + 4 \times E2 + 2 \times E1 + E0$$

جدول ۳.۹: مقدار رجیسترهای 0x03(MSB) و 0x04(LSB).

رجیستر	$bit0$	$bit1$	$bit2$	$bit3$	$bit4$	$bit5$	$bit6$	$bit7$
0x03	$M4$	$M5$	$M6$	$M7$	$E0$	$E1$	$E2$	$E3$
0x04	$M0$	$M1$	$M2$	$M3$	-	-	-	-

$$Mantissa = 128 \times M7 + 64 \times M6 + 32 \times M5 + 16 \times M4 + 8 \times M3 + 4 \times M2 + 2 \times M1 + M0$$

۳.۲.۹ نحوه دریافت جهت جغرافیایی از سنسور QMC5883L

در ابتدا پس از روشن شدن ماژول لازم است رجیستر کانفیک این ماژول تنظیم شده و ماژول در مود کاری مورد نظر قرار بگیرد. بیت‌ها و تنظیمات متناظر با اطلاعات پر شده در این رجیستر در جدول تصویر ۴.۹ قابل مشاهده است.

Addr	7	6	5	4	3	2	1	0
09H	OSR[1:0]		RNG[1:0]		ODR[1:0]		MODE[1:0]	

Reg.	Definition	00	01	10	11
Mode	Mode Control	Standby	Continuous	Reserve	Reserve
ODR	Output Data Rate	10Hz	50Hz	100Hz	200Hz
RNG	Full Scale	2G	8G	Reserve	Reserve
OSR	Over Sample Ratio	512	256	128	64

شکل ۴.۹: رجیستر کانفیگ ماژول QMC5883L.

برای شروع به کار و نمونه برداری، مود کاری باید در حالت continuous قرار بگیرد. برای این پروژه ریت خروجی ۱۰ یا ۵۰ هرتز و ریت نمونه برداری ۲ گاوس مناسب اند. مقدار OSR را نیز بیشترین حالت یعنی ۵۱۲ در نظر می‌گیریم. پس مقدار رجسیتر کانفیگ باید با 0x05 پر شود. برای این کار باید مقدار آدرس رجیستر کانفیگ (0x09) را به در مود Write به آدرس ماژول (0x0D) ارسال کنیم و پس از آن مقدار مورد نظر را (0x05) ارسال نماییم. با این کار ماژول وارد مود کار بدون وقفه خود شده و پس از اندازه گیری و ذخیره اطلاعات در رجیسترهای خود اولین بیت رجیستر حالت خود به آدرس 0x06 را به نشانه وجود داده برای دریافت ۱ می‌کند. پس با خواندن رجیستر حالت این ماژول میتوانیم از وجود یا عدم وجود داده برای دریافت باخبر شویم. این کار را با ارسال آدرس رجیستر حالت (0x06) در مود Read به آدرس ماژول (0x0D) انجام می‌دهیم و ماژول در پاسخ ۸ بیت داده‌های رجیستر حالت را به ما برمی‌گرداند. در صورت ۱ بودن اولین بیت رجیستر حالت داده‌ها برای دریافت آماده‌اند و لازم است آن‌ها را از رجیسترهای داده بخوانیم. رجیسترهای 0x00(LSB) و 0x01(MSB) حاوی اطلاعات محور X، رجیسترهای 0x02(LSB) و

0x03(MSB) حاوی اطلاعات محور Y و رجیسترهای 0x05(LSB) و 0x06(MSB) حاوی اطلاعات محور Z هستند. با دریافت این اطلاعات به راحتی با انجام محاسبات ریاضی میتوان زاویا با محورهای مختصات را بدست آورد. در این پروژه زاویه بردار روی صفحه X و متن‌لاتینY با بردار X برای ما اهمیت دارد که با توجه به فرمول زیر بدست می‌آید:

$$angle = \arctan \frac{Y}{X}$$

۴.۲.۹ نحوه خواندن رطوبت از سنسور AHT10

این سنسور بر خلاف سنسورهای دیگه نیازی به کانفیگ برای اندازه‌گیری و کالیبراسیون ندارد و پس از روشن شدن میتوان آن را در حالت دریافت اطلاعات قرار داد و پس از حداقل ۷۵ میلی‌ثانیه داده‌ها را از این سنسور خواند. این سنسور علاوه بر رطوبت دمای هوا را نیز اندازه گیری می‌کند ولی دقت آن در اندازه‌گیری دمای هوا به اندازه ماژول BMP180 نیست و از این رو داده‌های دمای هوایی که این سنسور اندازه‌گیری می‌کند را در این پروژه نادیده می‌گیریم.

برای قرار گرفتن ماژول در مود اندازه‌گیری باید دیتاهای 0xAC، 0x33 و 0x00 را به همین ترتیب به آدرس ماژول (0x38) ارسال کرد. با این کار ماژول ر حالت اندازه‌گیری قرار میگیرد و حداقل ۷۵ میلی‌ثانیه بعد دیتاهای اندازه‌گیری شده را در قالب ۴۸ بیت به ما برمی‌گرداند این دیتاها را هر هشت بیت با نام Data0 تا Data5 نام گذاری می‌کنیم. فشار نسبی هوا برحسب درصد از طریق روابط زیر و با استفاده از مقادیر برگشت داده شده از ماژول قابل محاسبه خواهد بود:

$$S_{RH} = Data1 \times 2^{12} + Data2 \times 2^{4} + Data3/2^{4}$$
$$RH = (S_{RH} * 100/2^{20});$$

که $Data1$ تا $Data3$ مقادیر برگشت داده شده از سنسور و RH مقدار رطوبت نسبی برحسب درصد است.

۵.۲.۹ نحوه محاسبه سرعت و جهت باد

جهت محاسبه سرعت و جهت باد ابتدا باید سرعت باد روی دو محور X و Y را بدست آورد. برای این کار در ابتدا نیاز به محاسبه سرعت صوت خواهد بود. سرعت صوت با توجه به روابطی که در بخش ۶.۷ بیان شده است به عنوان

تابعی از دمای هوا، رطوبت و فشار قابل محاسبه است. با وجود فاصله‌ای ثابت و مشخص بین فرستنده و گیرنده‌های آلتراسونیک،بنا به روابط بیان شده در بخش ۶.۷، زمانی که طول می‌کشد موج آلتراسونیک از فرستنده به گیرنده برسد با سرعت صورت به اضافه سرعت باد متناسب خواهد بود. پس با داشتن سرعت صوت و فاصله بین گیرنده و فرستنده و محاسبه زمانی که طول می‌کشد موج آلتراسونیک از فرستنده به گیرنده برسد می‌توان سرعت باد را محاسبه کرد.

برای محاسبه زمانی که طول می‌کشد تا موج آلتراسونیک از فرستنده به گیرنده برسد از ماژول HCSR05 بهره گرفتیم. این ماژول با دریافت پالسی با طول حداقل ۱۰ میکرو ثانیه، موج آلتراسونیکی با فرکانس ۴۰ کیلوهرتز تولید می‌کند. این ماژول هنگامی که پالس آلتراسونیک را تولید و ارسال می‌کند خروجی خود را ۱ می‌کند و آن را تا زمانی که پالس آلتراسونیک را از گیرنده خود دریافت کند ۱ نگه‌می‌دارد. در حقیقت با اندازه‌گیری طول پالس تولید شده در خروجی این ماژول زمانی که طول می‌کشد پالس آلتراسونیک از گیرنده به فرستنده برسد بدست می‌آید (این زمان به زمان پرواز معروف است). این زمان را با استفاده از واحد تایمر در میکروکنترلر اندازه‌گیری میکنیم به طوری که زمانی که پالسی با لبه بالا رونده روی پایه خروجی ماژول رخ داد شمارش تایمر را شروع کرده و هر یک میکرو ثانیه (دقت خروجی ماژول در این حد است) شمارش می‌کند و سپس زمانی که پالسی با لبه پایین رونده رخ داد شمارش را متوقف می‌کند و عدد شمارنده در این حالت همان زمان به اصطلاح پرواز ما خواهد بود.

با محاسبه زمان پرواز و با توجه به روابط ارائه شده در بخش ۶.۷ میتوان سرعت باد را بدست آورد. برای محاسبه جهت باد لازم است سرعت باد روی هر دو محور مختصاتی X و Y بدست بیاید. با داشتن سرعت باد روی هر دو محور میتوان این مقادیر را به دستگاه مختصات قطبی برد تا اندازه و زاویه وزش باد نسبت به محور X مختصات را در دست داشت. در این حالت اندازه در مختصات قطبی همان سرعت باد مد نظر ما است و با جمع کردن زاویه بدست آمده در این حالت با زاویه بدست آمده از سنسور قطب نما ($angle$) جهت باد بدست می‌آید. البته لازم است در این حالت محورهای مختصات سنسور باد منطبق با محورهای مختصات سنسور قطب‌نما باشند در غیر این صورت باید اختلاف زاویه این دو محور مختصات را نیز در محاسبات لحاظ کرد.

۶.۲.۹ نحوه ارسال اطلاعات از طریق لورا

در ابتدا پس از روشن کردن لورا باید مقادیر رجیسترهای کانفیگ آن را با مقادیر مورد نیاز پر کرد و آن را در مود کاری مورد نیاز قرار داد. به جهت نوشتن رجیسترهای کانفیگ لورا باید در ابتدا این ماژول را در مود اسلیپ قرار داد. تنها در این حالت مقادیر نوشته شده به عنوان کانفیگ روی ماژول اعمال می‌شوند. برای این کار باید مقدار رجیستر RegOpMode با آدرس 0x01 با مقدار 0x08 پر شود. برای این کار پس از انتخاب چیپ با پایه NSS، مقدار 0x01 و سپس 0x08 روی پایه MOSI ارسال می‌شود. با این کار ماژول لورا در حالت اسپیلپ و آماده اعمال تنظیمات قرار می‌گیرد. حال

مقدار رجیسترهای RegFrMsb، RegFrMid و RegFrLsb به آدرس‌های 0x06، 0x07 و 0x08 را که تنظیم کننده فرکانس موج رادیویی اند را با مقادیر 0x6C، 0x40 و 0x00 پر میکنیم (این مقدار از تقسیم فرکانس مورد نظر بر ۶۱٫۰۳۵۱۵۶۲۵ بدست آمده است). با این کار فرکانس کاری روی ۴۳۳ مگاهرتز تنظیم می‌شود. حال باید رجیستر RegPaConfig به آدرس 0x09 را مقدار دهی کنیم. این رجیستر تنظیم کننده توان خروجی است. برای توان خروجی ۲۰ dB باید مقدار 0x8F را در این رجیستر قرار دهیم. برای قرار دادن پهنای باند ماژول در حالت ۲۵۰ کیلو هرتز لازم است ۴ بیت آخر رجیستر RegModemConfig1 به آدرس 0x1D با مقدار 0b1000 پر شود. همچنین برای تعیین ضریب پخش ۱۰ لازم است ۴ بیت آخر رجیستر رجیستر RegModemConfig2 به آدرس 0x1E با مقدار 0b1010 پر شود. به جهت دریافت وضعیت ارسال پیام از طریق پایه‌های DIO لازم است بیت اول رجیستر RegDioMapping2 با مقدار 1 پر شود. در این حالت در صورت ارسال داده‌ها پایه DIO0 فعال می‌شود و ما با چک کردن وضعیت این پایه بعد از اعمال دستور ارسال می‌توانیم از وضعیت ارسال بسته باخبر شویم.

31	شناسه دستگاه فرستنده	0
63	شناسه دستگاه گیرنده	32
95	دما	64
127	فشار	96
159	شدت نور	128
192	رطوبت	160
223	سرعت باد	192
255	جهت باد	224

شکل ۵.۹: ساختار داده‌های موجود در سمت سنسور.

پس از تنظیم رجیسترهای کانفیگ ماژول را به حالت اسندبای میبریم در این حالت دیگر رجیسترهای کانفیگ قابل تغییر نخواهند بود. برای این کار مقدار 0x09 را درون رجیستر RegOpMode می‌نویسیم. جهت ارسال داده باید رجیستر RegFifo را با داده مورد نظر پر کرد و سپس مود کاری را در حالت Tx قرار داد. برای قرار دادن ماژول در مود Tx تنها کافیست مقدار 0x8B را درون رجیستر RegOpMode (با آدرس 0x01) نوشت. داده‌های دریافت شده در سمت سنسور در آرایه‌ای از نوع float ذخیره شده اند. ساختار این داده‌ها را در تصویر fig:sensorData مشاهده می‌کنید. برای جای دادن داده‌ها در رجیستر FIFO ماژول لورا لازم است داده‌ها در ساختاری ۸ بیتی (آرایه‌ای از اعداد بدون علامت ۸ بیتی) برای ماژول ارسال شوند از این رو مقادیر موجود در آرایه داده‌ها هر ۸ بیت به هشت بیت

به ماژول لورا و رجیستر FIFO این ماژول فرستاده می‌شوند. ساختار داده‌های ارسالی به رجیستر FIFO لورا نیز در تصویر ۶.۹ قابل مشاهده است. این داده‌ها با همین ساختار به طرف ایستگاه فرستاده و دریافت می‌شوند و لازم است پس از دریافت دوباره به ساختار اصلی و قبلی خود تبدیل شوند.

0	شناسه دستگاه فرستنده بخش اول	8
8	شناسه دستگاه فرستنده بخش دوم	15
16	شناسه دستگاه فرستنده بخش سوم	23
24	شناسه دستگاه فرستنده بخش چهارم	31
⋮	⋮	⋮
224	جهت باد بخش اول	231
232	جهت باد بخش دوم	239
240	جهت باد بخش سوم	247
248	بهت باد بخش چهارم	255

شکل ۶.۹: ساختار داده‌های ارسالی از طریق لورا.

۷.۲.۹ پیکربندی I2C

برای نوشتن کدهای این پروژه از لایبری HAL استفاده می‌کنیم. در این صورت برای فعال کردن بخش I2C تنها لازم است این بخش را در نرم افزار STM32CubeMX فعال کنیم. تنظیمات این بخش مثل سرعت و نوع آدرس وابسته به دستگاه‌های Slave متصل به I2C است و باید با توجه به مواردی که آن‌ها پشتیبانی می‌کنند تنظیمات این قسمت را پر کرد. خوشبختانه تمامی سنسورهای استفاده شده برای این پروژه تنظیمات یکسانی را پشتیبانی می‌کنند و می‌توان تنها با یک I2C با تمامی آن‌ها ارتباط برقرار و دیتاهای لازم را دریافت کرد. سنسورهای استفاده شده همگی از ماکسیموم سرعت قابل استفاده یعنی ۴۰۰ کیلوهرتز پشتیبانی می‌کنند اما در مورد این پروژه سرعت تبادل داده‌ها با سنسورها تفاوت زیادی نمی‌کند. پس در نتیجه سرعت ۱۰۰ کیلوهرتز را برای تست وتوسعه به امید کمتر کردن خطاهای ممکن انتخاب می‌کنیم. تصویر تنظیمات این بخش در نرم افزار STM32CubeMX در شکل ۷.۹ آمده است.

Master Features	
I2C Speed Mode	Standard Mode
I2C Clock Speed (Hz)	100000
Slave Features	
Clock No Stretch Mode	Disabled
Primary Address Length...	7-bit
Dual Address Acknowle...	Disabled
Primary slave address	0
General Call address de...	Disabled

شکل ۷.۹: تنظیمات I2C.

۸.۲.۹ پیکربندی SPI

به سبب اسفاده از لایبری HAL و نرم افزار STM32CubeMX برای فعال کردن این بخش از میکروکنترلر تنها کافی است این بخش را در نرم اقزار STM32CubeMX فعال کنیم. برای این کار SPI مورد نظر را در حالت Full-Duplex Master قرار می‌دهیم. تنظیم پارامترهای دیگر بستگی به تنظیمات قابل پشتیبانی برای Slaveها دارد. در این حالت برای ماژول لورا تنظیمات تصویر ۸.۹ را اعمال می‌کنیم. ماژول لورا سرعت‌های تبادل بالاتر را نیز ساپورت می‌کند ولی در این پروژه سرعت تبادلات تفاوت زیادی ایجاد نمی‌کند از این رو برای تست وتوسعه به استفاده از سرعت‌های تبادل پایین تر بسنده می‌کتیم.

Basic Parameters	
Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First
Clock Parameters	
Prescaler (for Baud Rate)	128
Baud Rate	562.5 KBits/s
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge
Advanced Parameters	
CRC Calculation	Disabled
NSS Signal Type	Software

شکل ۸.۹: تنظیمات SPI.

۹.۲.۹ حالت توقف میکروکنترلر

در این پروژه لازم است اطلاعات سنسورها هر ۳ ساعت یکبار جمع آوری و از طریق ماژول لورا مخابره شود و در بین این ساعات میکرو عملکرد دیگری نخواد داشت. برای ذخیره انرژی میتوان میکرو را در بین این ساعات در حالت توقف یا Stop قرار داد. در حالت توقف مصرف انرژی میکرو کنترلر با غیر فعال کردن کلاک‌ها بسیار کاهش می‌یابد. در این مود مقادیر رم و رجیسترها حفظ می‌شود و فقط کلاک‌ها غیر فعال می‌شوند. بعد از برخواستن از این حالت سیستم از کلاک HSI به عنوان منبع کلاک استفاده می‌کند که لازم است در این حالت برای برگشت به حالت کار عادی تنظیمات کلاک دوباره اعمال شوند. در این حالت میکروکنترلر تنها مصرفی در حدود ۰٫۸ میکروآمپر را دارا می‌باشد (با فعال بودن RTC).

تفاوت این حالت با حالت Standby که در آن میکرو کمترین میزان مصرف انرژی خود را دارد در این است که در حالت Standby مقادیر رجیسترها و رم ذخیره نمی‌شود (فقط رجیسترهای مربوط به حالت Standby دست نخورده باقی می‌مانند و باقی اطلاعات از دست می‌روند) و میکرو پس از برخواستن از این حالت وضعیتی مشابه وضعیت ریست شدن را خواهد داشت اما می‌توان مصرف را در حالتی مشابه تا ۰٫۵۷ میکروآمپر کاهش داد.

برای خارج کردن میکروکنترلر از این حالت می‌توان از اینتراپت خارجی و یا آلارم واحد RTC استفاده کرد. در این پروژه ما از آلارم واحد RTC استفاده می‌کنیم. اولین آلارم را میتوان با استفاده از نرم افزار STM32CubeMX در بخش RTC تنظیم کرد و پس از آن با رخ دادن هر آلارم باید آلارم بعدی را برای سه ساعت بعد تنظیم کرد.

۳.۹ ایستگاه

فلوچارت فرآیندهای سمت ایستگاه در تصویر ۹.۹ قابل مشاهده است شرح آن به قرار زیر است:

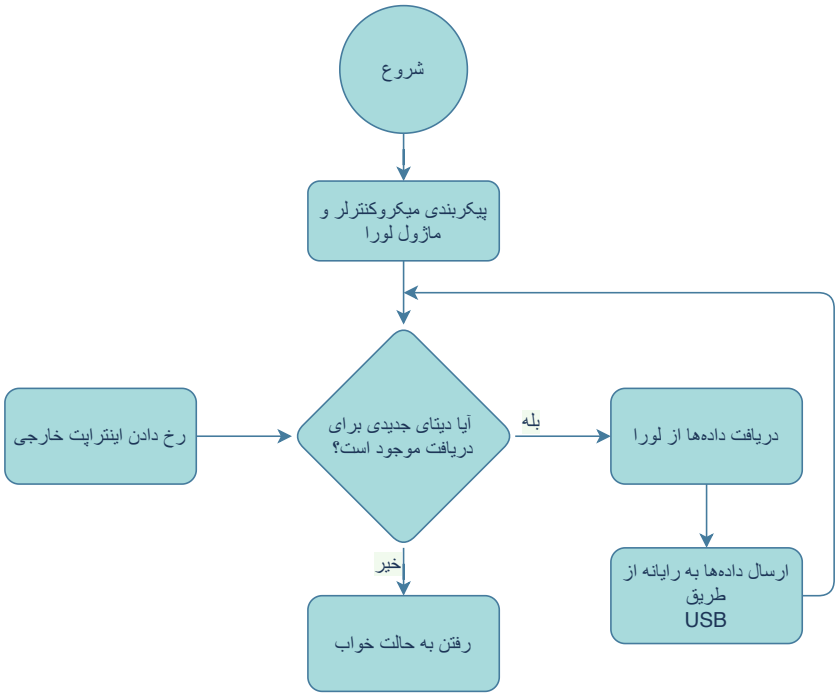
- با اتصال دستگاه از طریق کابل USB به رایانه میکروکنترلر پس از فعال‌سازی بخش‌های موردنیاز، از طریق SPI با ماژول لورا ارتباط برقرار کرده و رجیسترهای تنظیمات را با اطلاعات مشابه با سمت سنسور پر می‌کند و ماژول لورا را در حالت دریافت اطلاعات بدون توقف [۳] قرارمی‌هد [۱۲].

- میکرو به حلقه اصلی کار خود واردشده و پس از چک کردن وجود دیتای دریافتی در ماژول لورا، در صورت عدم وجود دیتا به حالت خواب [۴] رفته و تا زمان دریافت دیتا در همان حالت باقی می‌ماند.

[3] Continuously

[4] Sleep

- با دریافت دیتا توسط ماژول لورا وقفه‌ای خارجی رخ‌داده و میکرو را از حالت خواب بیدار می‌کند.
- میکرو کنترلر با برقراری ارتباط از طریق SPI با ماژول لورا دیتای دریافت شده را خوانده و پس از بررسی یکسان بودن شناسه دریافت‌کننده با شناسه خود آن را از طریق USB به رایانه ارسال می‌کند.
- در صورت عدم وجود دیتاهای دیگر، میکرو کنترلر به حالت خواب رفته و تا رخ دادن وقفه بعدی، که نشان‌دهنده دریافت اطلاعات توسط ماژول لورا است، در همان حال باقی می‌ماند.



شکل ۹.۹: فلوچارت برنامه سمت ایستگاه.

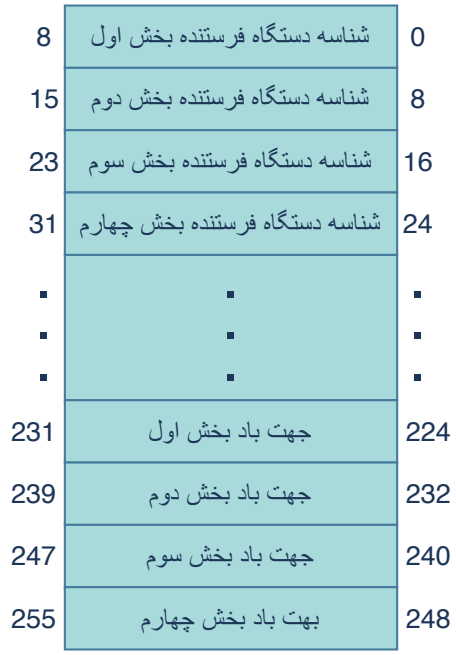
۱.۳.۹ نحوه دریافت اطلاعات از لورا

برای اینکه ماژول لورا بتواند در این سمت اطلاعات ارسال شده را دریافت کند باید دقیقا همان تنظیماتی که در سمت سنسور برای آن اعمال شده در این سمت نیز اعمال شود. با وجود کوچک ترین تفاوتی بین پارامترهای فرکانس، پهنای‌باند و ضریب‌پخش بین دو سمت فرستنده و گیرنده دیگر ماژول لورا قادر نخواهد بود اطلاعات ارسال شده را دریافت کند. پس تمامی این پارامترها دقیقا مطابق بخش ۶.۲.۹ در ماژول تنظیم خواهند شد. همچنین پیکربندی SPI همانند پیکربندی بخش سنسور تنظیم شده است.

پس از تنظیم این پارامترها ماژول لورا در این سمت باید در مود دریافت اطلاعات بدون توقف قرار داده شود. در این حالت در صورت دریافت اطلاعات ماژول روی پایه‌های DIO خود با فعال کردن یکی از پایه‌ها (که قابل تنظیم

است کدام پایه فعال شود) رخ دادن دریافت اطلاعات را اطلاع می‌دهد. در این حالت ماژول از کار باز نمی‌ایستد و دوباره منتظر دریافت اطلاعات می‌ماند. برای قرار دادن ماژول در این مود باید رجیستر RegOpMode به آدرس لاتین۰x۰۱ با مقدار 0x8D پر شود.

در این صورت در صورت رخ دادن دریافت داده میکروکنترلر از طریق اینتراپت خارجی‌ای که به پایه DIO ماژول وصل شده است از رخ دادن دریافت اطلاعات باخبر شده و می‌تواند جهت خواندن اطلاعات دریافت شده اقدام نماید. برای خواندن اطلاعات دریافت شده در ابتدا باید طول اطلاعات دریافت شده را از رجیستری با عنوان RegRxNbBytes با آدرس 0x13 دریافت کرد (ماکسیموم طول بسته قابل دریافت ۲۵۶ بسته است). پس از آن با داشتن طول بسته می‌توان دیتای دریافت شده را از رجیسترهای FIFO (با آدرس 0x00) خواند. پس از خواندن اطلاعات موجود لازم است فلگ مربوط به اینتراپ را در ماژول ریست کرد برای این کار کافیست مقدار 0xFF را درون رجیستر RegIrqFlags به آدرس 0x12 نوشت.



شکل ۱۰.۹: ساختار داده‌های دریافت شده از طریق لورا.

داده‌های دریافت شده در این قسمت ساختاری مشابه ساختار داده‌های ارسال شده دارند (در تصویر ۱۰.۹ قابل مشاهده است). در این قسمت لازم است قبل از ارسال اطلاعات به رایانه شناسه دریافت کننده با شناسه موجود در دستگاه سمت ایستگاه چک شود. این کار برای جلو گیری از ثبت چند باره اطلاعات و یا جلو گیری از ثبت اطلاعات نادرست برای دستگاه‌هایی که همپوشانی رادیویی منطقه‌ای دارند لازم است. برای این کار باید ابتدا ساختار داده‌های دریافت شده که شامل ۸x۳۲ داده از نوع عدد بدون علامت می‌شود به ساختار اصلی که ۳۲x۸ داده‌ای و از نوع float بوده است تبدیل شود این کار به سادگی و با تبدیل هر ۴ جایگاه آرایه دیتاهای دریافت شده به نوع float انجام می‌شود.

تصویر داده‌ها پس از تبدیل به ساختار اصلی در شکل ۱۱.۹ آمده است.

31	شناسه دستگاه فرستنده	0
63	شناسه دستگاه گیرنده	32
95	دما	64
127	فشار	96
159	شدت نور	128
192	رطوبت	160
223	سرعت باد	192
255	جهت باد	224

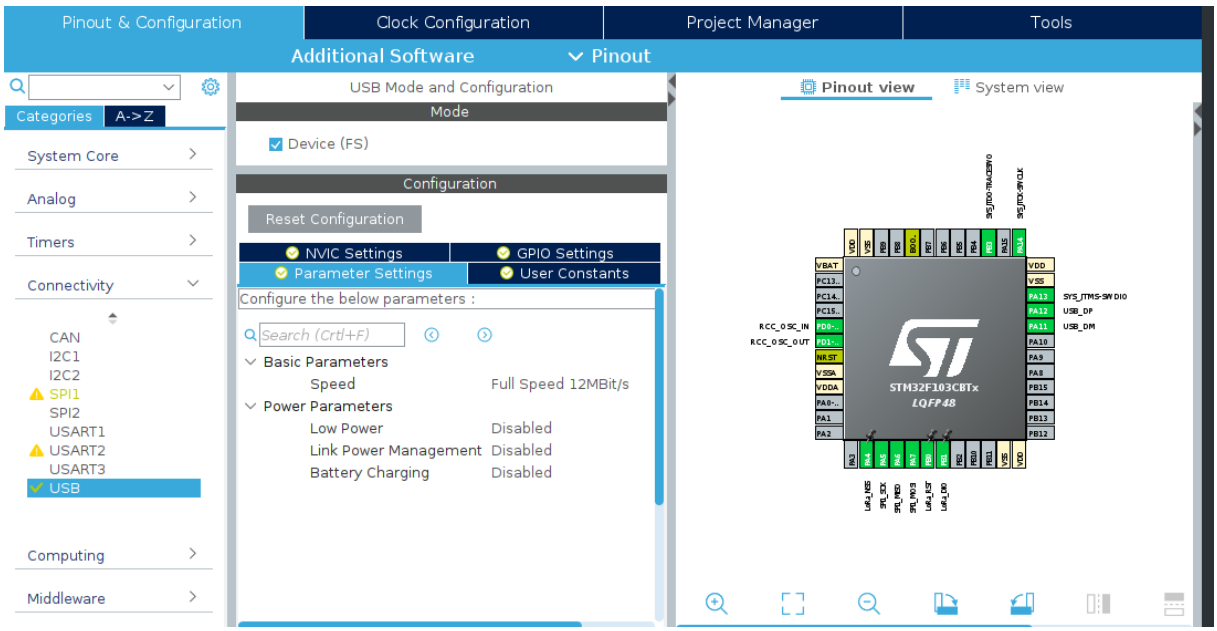
شکل ۱۱.۹: ساختار داده‌ها پس از تبدیل به ساختار اصلی در سمت ایستگاه.

پس از این تبدیل شناسه دستگاه گیرنده که در دومین جایگاه آرایه داده‌های تبدیل شده قرار دارد با شناسه ثبت شده در دستگاه سمت ایستگاه بررسی شده و در صورتی که با یکدیگر مطابقت داشتند داده‌های دریافت شده از طریق USB به رایانه ارسال می‌شوند.

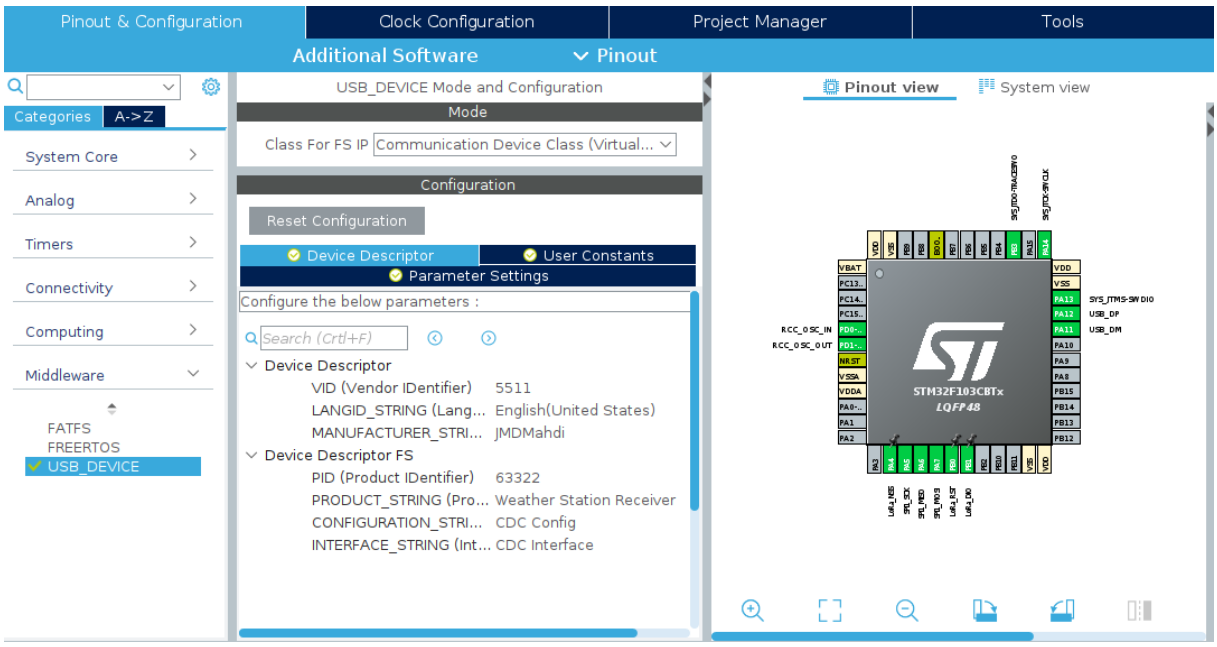
۲.۳.۹ نحوه ارسال اطلاعات از طریق USB

از انجایی که در این پروژه ار کتابخانه HAL استفاده می‌کنیم فعال کردن واحد USB به سادگی همانند فعال کردن بخش‌های I2C و SPI از طریق نرم‌افزار STM32CubeMX قابل انجام است. برای این کار کافیست تیک Device در قسمت USB زده شود و سرعت یو اس بی مشخص شود. تصویر این تنظیمات در شکل ۱۲.۹ قابل مشاهده است. پس از آن لازم است کلاس یو اس بی را مشخص کنیم. برای این کار تنها لازم است از قسمت Middleware به قسمت USB_Device رفته و در قسمت Class for FS IP کلاس مورد نظر خود را انتخاب کنیم. تفاوت و کاربر هر یک از این کلاس‌ها در بخش ۴.۶ توضیح داده شده‌است. در اینجا ما قصد استفاده از کلاس CDC یا Communication Device Class را داریم. در این حالت (کلاس CDC و یواس‌بی فول اسپید) میتوانیم دیتاهای بالک ۶۴ بایتی را از طریق USB مخابره کنیم. تصویر تنظیمات بخش کلاس در شکل ۱۳.۹ آمده است. در این قسمت لازم است مقادیر VID و PID را مقادیری یکتا و مختص این نوع دستگاه وارد کنیم. مقدار VID شناسه سازنده و مقدار PID شناسه این نوع دستگاه است. در هنگام ثبت محصول به عنوان محصول نهایی و قابل عرضه در بازار لازم است عملکرد یواس‌بی دستگاه مورد آزمایش واقع شده و با پس از دریافت گواهینامه‌های معتبر

امکان خرید شناسه برای دستگاه میسر خواهد بود. البته شرکت‌هایی نیز هستند که میتوان از آن‌ها PID دریافت کرد ولی در این صورت باید از VIDای که به نام آن شرکت است استفاده نمود. برای تست و توسعه می‌توان هر مقدار PID و VIDای که توسط دستگاه‌های متصل به رایانه خود استفاده نمی‌شود استفاده کرد. مقادیر Manufacture String و Product String نیز مقادیری هستند که در صورت اتصال دستگاه به رایانه به عنوان نام سازنده دستگاه و نم دستگاه نمایش داده می‌شود.



شکل ۱۲.۹: تنظیمات USB در نرمافزار STM32CubeMX.



شکل ۱۳.۹: تنظیمات USB_Device در نرمافزار STM32CubeMX.

جهت ارسال داده‌ها از طریق کتابخانه HAL، به سادگی می‌توان از دستور CDC_Transmit_FS استفاده کرد. این دستور دو پارامتر داده ارسالی و سایز داده را دریافت کرده و داده را از طریق USB به دستگاه Host مخابره می‌کند. نوع داده‌های دریافتی در این قسمت نیز همانند داده‌های دریافتی در ماژول لورا آرایه‌ای از داده‌های ۸ بیتی بدون علامت است. پس میتوان همان داده‌های دریافت شده از طریق لورا را نیز مستقیما از طریق همین دستور و بدون هیچ تبدیل اضافه از طریق USB به رایانه ارسال کرد. این تابع به طور خودکار در صورتی که سایز داده‌های ارسالی بیشتر از ۶۴ بایت باشد داده ارسالی را به چندین بسته بالک ۶۴ بایتی تقسیم می‌کند. در اینجا ما ۳۲ بایت داده را از لورا دریافت می‌کنیم و دقیقا همان داده‌ها را میخواهیم ارسال کنیم پس داده‌ها در یک بسته بالک به خوبی جای می‌گیرند. تصویر این داده‌های در شکل ۱۰.۹ در بخش‌های قبلی آمده است.

۳.۳.۹ حالت خواب میکروکنترلر

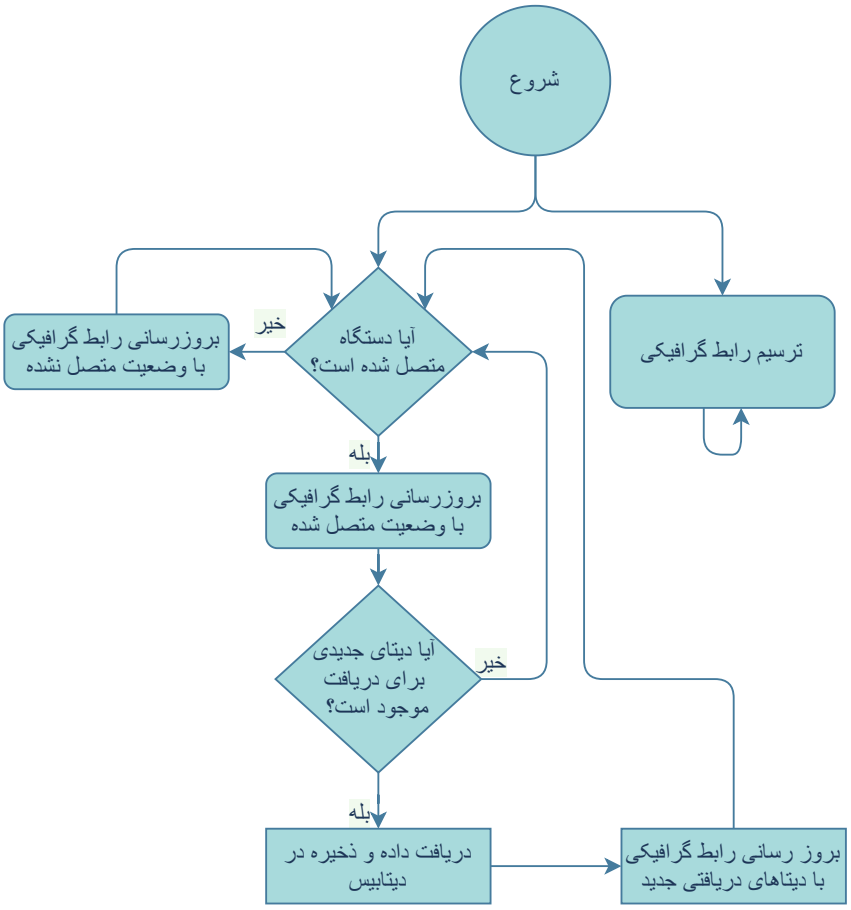
داده‌های ارسالی ار طریق ماژول لورا در سمت سنسور هر سه ساعت یک بار ارسال می‌شوند و در سمت سنسور نیز ان داده‌ها در سه ساعت یک‌بار دریافت می‌شوند به همین جهت و دقیقا مشابه سمت سنسور به دلیل تمایل به ذخیره انرژی در بین این ساعات بی‌کاری میکروکنترلر را در حالت خواب قرار می‌دهیم. تفاوت حالت خواب با حالت توقف که در سمت سنسور از آن استفاده کردیم در این است که در این حالت میکروکنترلر می‌تواند با هریک از peripheralهای فعال از خواب برحیزد و به کار خود ادامه دهد و پس از اتمام کار دوباره به خواب برود. در این سمت به دلیل استفاده از یواس‌بی از این مود استفاده کردیه‌ایم تا در صورت رخ دادن هر گونه خطای سخت افزاری در اتصالات و یا ریست‌های نرم افزاری در سمت هاست پس از رفع و ثابت شدن وضعیت مجبور به قطع و وصل کردن کابل یو اس بی متصل به دستگاه نباشم.

در این حالت مصرف میکروکنترلر چیزی در حدود ۴۰۰ میکروآمپر خواهد بود. البته که ماژول لورا در این حالت در مود دریافت بدون وقفه خود قرار دارد و در صورت دریافت اطلاعات جدید میکروکنترلر را با اعمال اینتراپت خارجی از خواب بیدار می‌کند تا دیتای دریافت شده را گرفته و از طریق یو اس بی به رایانه منتقل نماید.

۴.۹ برنامه دسکتاپ

برنامه دسکتاپ که با زبان Python نوشته شده است، متشکل از سه بخش Home، Charts و Log می‌باشد. در بخش Home اطلاعات آخرین دیتای دریافت شده به نمایش درمی‌آید. در بخش Charts نمودارهای دیتاهای دریافتی در بازه قابل‌تعیین توسط کاربر به نمایش درمی‌آید. تمام رخدادهایی که در ارتباط با دستگاه رخ می‌دهد نظیر

دریافت دیتای جدید و یا اتصال یا قطع اتصال دستگاه در تب Log با ذکر زمان لیست می‌شوند.



شکل ۱۴.۹: فلوچارت برنامه دسکتاپ.

فلوچارت برنامه دسکتاپ در شکل fig:USBFlowChart آمده است. نحوه عملکرد برنامه دسکتاپ به شرح زیر است:

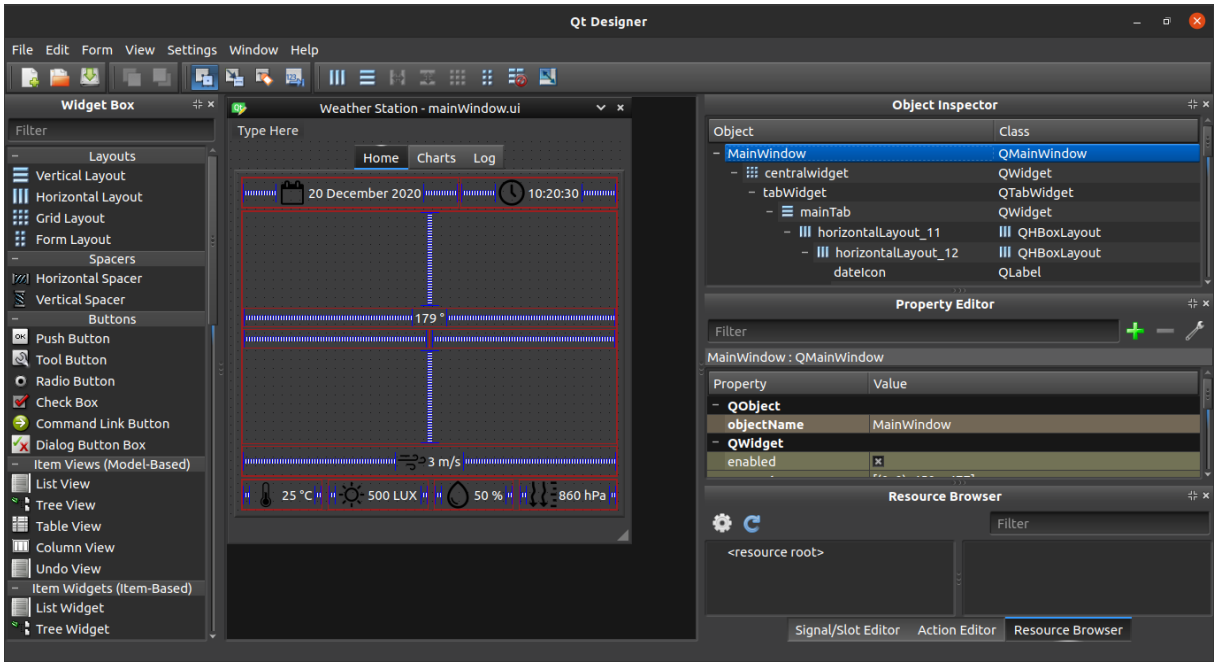
- با اجرای برنامه Thread اصلی وظیفه ترسیم رابط گرافیکی برنامه[5]، که با PyQt5 پیاده‌سازی شده است، را برعهده می‌گیرد.
- در همین حین Thread دیگر به کمک کتابخانه libusb مسئول بررسی وضعیت اتصال دستگاه به رایانه و دریافت اطلاعات ارسالی از دستگاه می‌شود.
- در صورت تغییر وضعیت اتصال و یا دریافت اطلاعات، مشخصات آن در تب Log ثبت می‌شود و برای کاربر قابل‌مشاهده خواهد بود.

[5] Graphical user interface (GUI)

- هنگام دریافت اطلاعات جدید از طریق USB علاوه بر نمایش در تب اصلی برنامه، به‌روزرسانی تب Charts با اطلاعات جدید و ثبت رخ داد در تب Log، مشخصات کامل آن در دیتابیس SQLite در کنار فایل اجرایی برنامه نیز ذخیره می‌شود.
- در تب Charts با انتخاب بازه زمانی، نمودارها باتوجه به اطلاعات ثبت‌شده آن بازه زمانی در دیتابیس بروز می‌شوند.

۱.۴.۹ نحوه طراحی رابط کاربری

از برنامه Qt Designer برای طراحی محیط GUI برنامه استفاده شده است که تصویری از محیط این نرم افزار را در شکل ۱۵.۹ مشاهده می‌کنید. در این برنامه به راحتی با درگ و دراپ کردن ویجت‌ها در بخش‌های مورد نظر میتوان رابط کاربری برنامه خود را ساخت. در پنل سمت راست این برنامه نیز می‌توان جزئیات بیشتری از هر ویجت را دید و به تنظیمات بیشتری دسترسی داشت.



شکل ۱۵.۹: طراحی محیط برنامه با نرم افزار Qt Designer.

برای این برنامه یک ویجت از نوع tabWidget با ۳ تب اضافه می‌کنیم. این تب‌ها همان سه بخش اصلی برنامه یعنی Home، Charts و Log را در بر می‌گیرند. در این صورت با کلیک کردن روی هر تب اطلاعات موجود در آن تب را نشان می‌دهند.

جهت نمایش چارت‌ها در تب Charts از ویجتی با عنوان chartWidget استفاده می‌کنیم. برای افزودن

اطلاعات به این نمودارها از سری‌هایی از نوع QLineSeries استفاده می‌کنیم. محور افقی این نمودارها را زمان ثبت شدن دیتاها در دیتابیس و محور عموی را مقدار داده‌های ذخیره شده در دیتابیس در نظر می‌گیریم. برای هر یک از داده‌های دما، رطوبت، فشار، سرعت باد، جهت باد و شدت نور نموداری از همین نوع و مستقل رسم می‌کنیم. در بالای نمودارها دو ویجت از نوع datePicker قرار داده شده است که کاربر با انتخاب آن میتواند تاریخ شروع و پایان را مشخص کند با مشخص کردن تاریخ شروع و پایان داده‌های نمودارها با اطلاعات متناظر با بازه انتخابی بروز می‌شوند. در اینجا تفاوتی ندارد که در کدام باکس تاریخ شروع و در کدام باکس تاریخ پایان را وارد کند برنامه طوری نوشته شده است که به طور خودکار تاریخی که از دیگری کوچک تر است را به عنوان تاریخ شروع و تاریخ دیگر را به عنوان تاریخ پایان در نظر بگیرد. همچنین به دلیل اینکه ارتفاع این نمودارها زیاد است آن‌ها را در ویجتی از نوع scrollArea قرار داده‌ایم تا امکان اسکرول کردن به این تب اضافه شود.

در تب Log یک ویجت از نوع چک باکس و پایین آن یک ویجت از نوع textBrowser قرار داده شده است. لاگ‌های رخ دادهای برنامه نظیر قطع و وصل دستگاه، دریافت اضلاعات جدید و وضعیت افزودن دیتاها به دیتابیس در این قسمت با زمان دقیق رخ داد لیست خواهند شد. همچنین در صورتی که تیک چک باکس بالای باکس متنی زده شده باشد در صورت افزوده شدن ردیف‌های متنی جدید اسکرول بار باکس متن به آخرین خط اضافه شده منتقل می‌شود و در غیر این صورت (در صورتی که چک‌باکس تیک زده نشده باشد) اسکرول باکس متنی تغییری نخواد کرد.

۲.۴.۹ ساختار دیتابیس

به جهت ذخیره داده‌های از دیتابیس SQLite استفاده می‌کنیم. دلیل این امر پشتیبانی اکثر زبان‌ها و سیستم عامل‌ها از این نوع دیتابیس و سادگی و قابل حمل بودن آن است. ساختار جدول دیتابیس SQLiteای که اطلاعات پس از دریافت از طریق USB در آن ذخیره می‌شود در تصویر ۱۶.۹ آمده است.

weather_records		CREATE TABLE "weather_records" ("id" INTEGER PRIMARY KEY
id	INTEGER	"id" INTEGER PRIMARY KEY AUTOINCREMENT
sensor_id	INTEGER	"sensor_id" INTEGER NOT NULL DEFAULT 0
station_id	INTEGER	"station_id" INTEGER NOT NULL DEFAULT 0
temperature	REAL	"temperature" REAL NOT NULL DEFAULT 0
pressure	REAL	"pressure" REAL NOT NULL DEFAULT 0
light_intensity	REAL	"light_intensity" REAL NOT NULL DEFAULT 0
humidity	REAL	"humidity" REAL NOT NULL DEFAULT 0
wind_speed	REAL	"wind_speed" REAL NOT NULL DEFAULT 0
wind_direction	REAL	"wind_direction" REAL NOT NULL DEFAULT 0
timestamp	INTEGER	"timestamp" INTEGER NOT NULL

شکل ۱۶.۹: ساختار جدول دیتابیس.

در برنامه پس از اجرا در صورتی که فایل دیتابیس ویا جدول دیتابیس موجود نباشد فایل و جدول دیتابیس ساخته می‌شود. این فایل در کنار فایل اجرایی برنامه در پوشه DB و با عنوان weatherStation.db قرار می‌گیرد. این فایل را می‌توان به سادگی با نرم افزارهای مدیریت دیتابیس Sqlite باز کرد و دیتاهای ذخیره شده را ویرایش کرد و یا به سیستم و دیتابیس‌های دیگر منتقل کرد. همچنین می‌توان با اکثر زبان‌های برنامه نویسی کوئری‌های مورد نظر را روی این فایل و دیتابیس به جهت دریافت یا ثبت داده‌ها اجرا کرد.

پارامتر id در این جدول پارامتری عددی صحیح و یکتا است که به طور خودکار با افزودن دیتاهای جدید مقدار آن یک واحد افزایش می‌یابد. در این صورت هر دیتا ثبت شده علاوه بر زمان ثبت، که در پارامتر timestamp ثبت می‌شود، شناسه‌ای یکتا (id) برای خود خواهند داشت. پارامتر timestamp مقداری عددی صحیح می‌پذیرد و در آن مقدار ثانیه‌های گذشته از تاریخ مرجع Jan 01 1970 تا تاریخ مورد نظر قرار داده می‌شود. به این نوع نحوه ثبت زمان، که ثانیه‌های گذشته از تاریخ Jan 01 1970 تا تاریخ مورد نظر برای ثبت است، Unix Time و یا Timestamp گفته می‌شود. مابقی پارامترها مقادیر دریافت شده از USB را درون خود جای می‌دهند و به دلیل اینکه نوع داده‌های دریافت شده float بوده است این پارامترها مقادیری از نوع اعداد حقیقی دارند که این نوع اعداد را می‌تواند بدون مشکل در خود جای دهد.

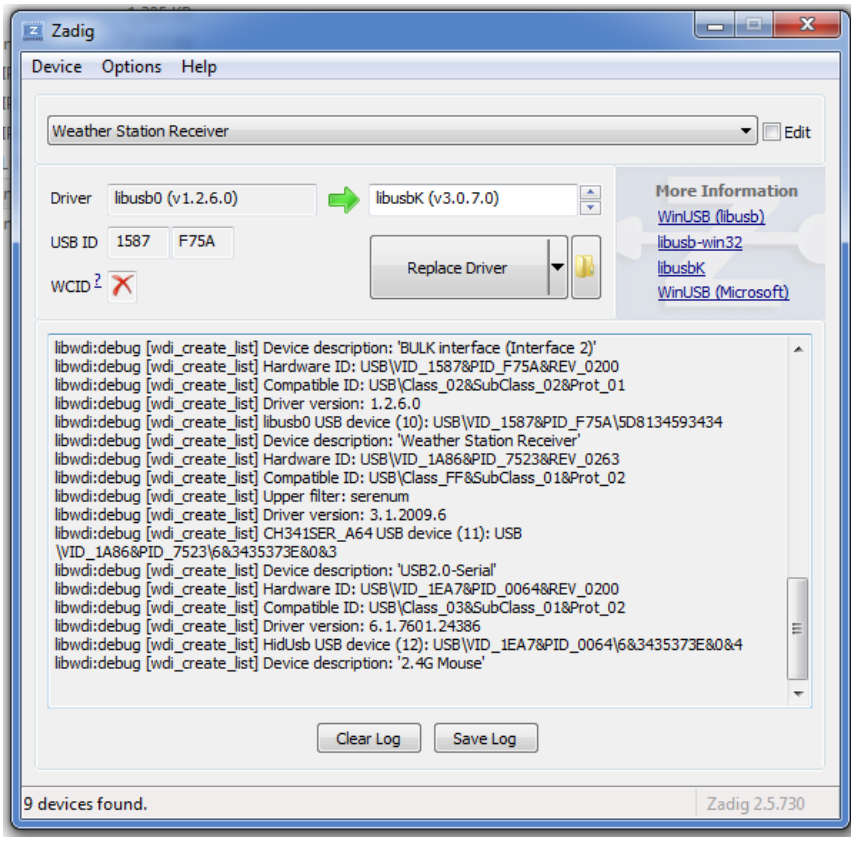
پس از اتمام طراحی این برنامه (Qt Designer) خروجی‌ای با پسوند UI به ما می‌دهد. این فایل را میتوانیم با ابزارهای دیگری که در کنار این نرم افزار ارائه می‌شود به کدهای قابل استفاده در زبان‌های برنامه نویسی قابل پشتیبانی تبدیل کرد. برای مثال برای تبدیل این فایل به کدهای قابل استفاده در پایتون از دستور زیر استفاده می‌کنیم.

```
pyuic5 mainWindow.ui -o mainWindow.py
```

که با دریافت فایل mainWindow.ui فایل خروجی mainWindow.py را به ما تحویل می‌دهد که از آن می‌توانیم مستقیما در برنامه خود استفاده کنیم.

۳.۴.۹ نحوه دریافت اطلاعات از طریق USB

برای دریافت اطلاعات ارسال شده از سخت افزار سمت ایستگاه به رایانه لازم است در ابتدا رایانه کلاس دستگاه را بشناسد تا بتواند با آن به درستی ارتباط برقرار کند. برای کلاس CDC استفاده شده بهترین درایور قابل استفاده که تمامی امکانات این کلاس را پوشش دهد درایور LibUSB است. این درایور در سیستم‌های لینوکسی و مبتنی بر یونیکس به طور دیفالت روی سیستم موجود است و فقط کافیست دسترسی برنامه به یو اس بی از طریف تنظیمات udev باز شود تا برنامه بتواند دستگاه متصل شده را دیده و با آن ارتباط برقرار کند. در محیط ویندوز این مسئله کمی متفاوت است و باید حتما به صورت جداگانه درایور LibUSB برای این دستگاه نصب شود. از نرم افزار Zadig



شکل ۱۷.۹: تصویری از محیط نرم افزار نصب درایور Zadig.

می‌توان به این منظور استفاده کرد. تصویری از محیط این نرم افزار در هنگام نصب درایور در شکل ۱۷.۹ آمده است.

به جهت برنامه نویسی این قسمت از کتابخانه PyUSB در زبان برنامه نویسی پایتون استفاده شده است. پس از افزودن این کتابخانه به پروژه لازم است در ابتدا بک‌اند این کتابخانه نیز مشخص گردد. بک‌اند این کتابخانه باید متناسب با ورژن درایور نصب شده انتخاب گردد. در سیستم‌عامل‌های مبتنی بر یونیکس به جهت وجود درایور LibUSB این کار لازم نیست و به طور خودکار این کار توسط خود کتابخانه انجام می‌شود.

چک کردن وضعیت اتصال یو اس بی به دستگاه و یا دریافت اطلاعات باید در Thread جدا از Thread اصلی که مسئول ترسیم رابط گرافیکی برنامه است انجام شود. در غیر ان صورت در هنگام انجام فرآیندهای چک کردن وضعیت و دریافت اطلاعات رابط کاربری برنامه قفل شده و امکان کلیک کردن روی دکمه‌ها و رفتن به بخش‌های دیگر برنامه ممکن نیست. تمامی فرآیندهای چک کردن ارتباط با دستگاه و دریافت اطلاعات از آن با استفاده از کتابخانه PyUSB انجام می‌شود.

اطلاعاتی که در این بخش دریافت می‌شود اطلاعاتی از نوع bytearray و یا Uint8Array هستند که ساختار تصویر ۱۸.۹ را دارند (که همان ساختار ارسال شده از طرق لورا است). برای دست‌یابی به دیتاهای اولیه لازم است این داده‌ها را به لیستی از اعداد float ۳۲ بیتی تبدیل کنیم. این کار با کنار هم قرار دادن هر ۴ بایت از این داده‌ها میسر

می‌شود. برای این کار از کتابخونه numpy و توابعی که به همین منظور دارد استفاده می‌کنیم تا درنهایت به ساختار تصویر ۱۹.۹ برسیم.

0	شناسه دستگاه فرستنده بخش اول	8
8	شناسه دستگاه فرستنده بخش دوم	15
16	شناسه دستگاه فرستنده بخش سوم	23
24	شناسه دستگاه فرستنده بخش چهارم	31
⋮	⋮	⋮
224	جهت باد بخش اول	231
232	جهت باد بخش دوم	239
240	جهت باد بخش سوم	247
248	بهت باد بخش چهارم	255

شکل ۱۸.۹: ساختار داده‌های دریافت شده از طریق USB.

0	شناسه دستگاه فرستنده	31
32	شناسه دستگاه گیرنده	63
64	دما	95
96	فشار	127
128	شدت نور	159
160	رطوبت	192
192	سرعت باد	223
224	جهت باد	255

شکل ۱۹.۹: ساختار داده‌ها پس از تبدیل به ساختار اصلی در برنامه دسکتاپ.

در این صورت ساختاری مشابه ساختار اطلاعات جمع آوری شده در سمت سنسور را داریم که دقیقا با همان اطلاعات جمع آوری شده پر شده است و میتوانیم مستقیما با دسترسی به خانه‌های مختلف آن مقادیر مورد نظر را گرفته و در دیتابیس برای استفاده‌های بعدی ذخیره کنیم.

فصل ۱۰

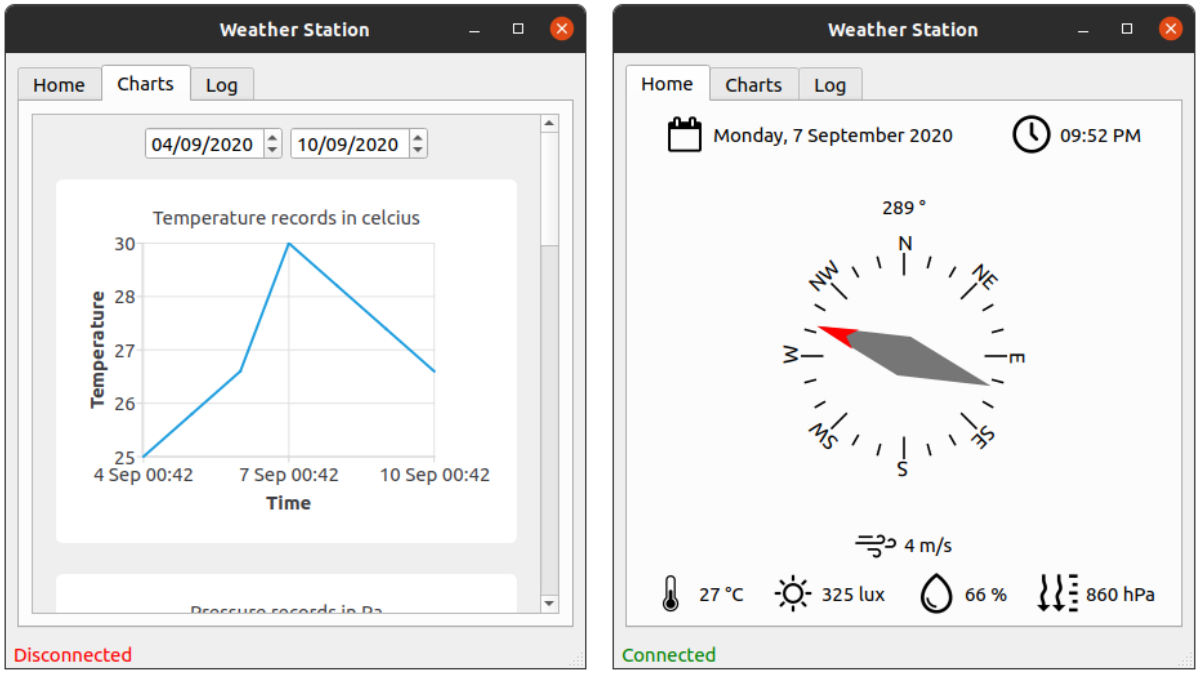
نتایج

۱.۱۰ مقدمه

سیستم سنسور (سیستمی که وظیفه جمع‌آوری دیتاها از سنسورها و مخابره آن‌ها را دارد) اطلاعات را جمع‌آوری کرده و برای سیستم ایستگاه (سیستمی که وظیفه دریافت داده‌های ارسال شده و ارسال آن‌ها به رایانه را دارد) از طریق ماژول لورا مخابره می‌کند. در بخش ایستگاه انتظار می‌رود این اطلاعات بلافاصله دریافت شده و برای رایانه ارسال شود. در رایانه برنامه‌ای که برای همین منظور نوشته شده‌است، وظیفه دریافت اطلاعات ارسال شده از طریق USB و ذخیره آن در دیتابیس را دارد. در این حالت انتظار می‌رود دیتاهایی که در نهایت برروی رایانه در دیتابیس ذخیره می‌شوند دقیقا مطابق با دیتاهایی باشد که دستگاه سنسور آن‌ها را جمع آوری و مخابره کرده است. همچنین انتظار می‌رود دستگاه ایستگاه قابلیت Plug & Play را داشته باشد به طوری که با اتصال دستگاه به USB رایانه در صورت باز بودن برنامه این اتصال به طور خودکار شناسایی شده و برنامه در وضعیت متصل و آماده دریافت داده‌ها قرار بگیرد.

۲.۱۰ تست برنامه دسکتاپ

خروجی اطلاعات جمع‌آوری و مخابره شده توسط دستگاه بر روی رایانه و به کمک برنامه ساخته‌شده به همین منظور در سمت ایستگاه قابل‌مشاهده می‌باشد. جهت انجام بررسی‌های جزئی و ابتدایی به جهت پیش‌بینی وضعیت آب‌وهوایی برروی دیتاهای دریافتی، می‌توان از نمودار دیتاهای دریافتی که در تب Charts برنامه قائل مشاهده است استفاده نمود. همچنین مشخصات و جزئیات آخرین دیتای دریافت شده نیز در تب Home این نرم‌افزار قابل‌مشاهده است. تصاویری از محیط برنامه در شکل ۱.۱۰ آمده است.



(آ) نمایش آخرین دیتاهای دریافتی برنامه در تب Home.

(ب) نمایش نمودارهای اطلاعات دریافت شده در تب Charts.

شکل ۱.۱۰: تصاویر محیط برنامه دسکتاپ.

همان طور که در تصویر ۱.۱۰ مشخص است در صورت اتصال دستگاه به رایانه در نوار وضعیت عبارت سبز رنگ Connected (تصویر سمت راست) و در صورت عدم اتصال دستگاه به رایانه در نوار وضعیت عبارت قرمز رنگ Disconnected (تصویر سمت چپ) درج می‌شود. همچنین در تب Charts با انتخاب بازه تاریخ نمودارهای مربوط به آن بازه به نمایش در خواهند آمد.

۳.۱۰ تست برد ماژول لورا

در انجام تست‌های عملی مشخص شد برد مفید ماژول لورا علاوه بر وابستگی‌ای که به پارامترهای پهنای‌باند، ضریب پخش و توان دارد، به‌شدت به نوع آنتن وابسته است و نیازمند توجه ویژه‌ای به مسئله تطبیق امپدانس ترک‌های آنتن خروجی در طراحی بردمدارچاپی است. در نهایت طبق آزمایش صورت گرفته در منطقه مسکونی، در فرکانس ۴۳۳ مگاهرتز، پهنای باند ۸.۲۰ کیلوهرتز، ضریب پخش ۱۰، توان dBm۲۰ و یک آنتن دست‌ساز، تا فاصله حدود ۲ کیلومتری می‌توان دیتاهای ارسالی را در سمت ایستگاه دریافت کرد.

۴.۱۰ تست سنسور سنجش سرعت و جهت باد

به دلیل در دسترس نبودن معیار دقیقی برای سنجش سرعت باد، نتیجه‌گیری در مورد دقت اندازه‌گیری سرعت باد اشتباه است اما با انجام آزمایشات متعدد دقت اندازه‌گیری جهت باد با سرعت متوسط و در دمای اتاق ۴± درجه به‌دست آمد. همچنین مشخص شد تغییر فاصله فرستنده و گیرنده‌های آلتراسونیک از یکدیگر و از زمین، عاملی مؤثر در تعیین دقت اندازه‌گیری و ماکسیموم سرعت قابل‌اندازه‌گیری است. به‌طوری‌که با نزدیک‌تر قرار دادن فرستنده و گیرنده (تا حداقل ۴ سانتی‌متر) سرعت قابل‌اندازه‌گیری ماکسیموم و دقت اندازه‌گیری مینیموم می‌شود.

سورس‌کد تمامی پخش‌های پروژه به‌صورت متن‌باز در وبگاه GitHub به نشانی زیر در دسترس است:

https://github.com/jmdmahdi/Weather-Station

مراجع

مراجع

[1] Wikipedia, “Weather station,” 2020. [Online; accessed August 13, 2020].

[2] mehrnews, “mehrnews,” 2020. [Online; accessed Oct 9, 2020].

[3] S. Schiavon, T. Hoyt, and A. Piccioli, “Web application for thermal comfort visualization and calculation according to ASHRAE standard 55,” *Building Simulation*, vol.7, pp.321–334, Dec. 2013.

[4] P. Chandran, R. Bhakthavatchalu, and P. P. Kumar, “Time of flight measurement system for an ultrasonic anemometer,” in *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pp.734–737, 2016.

[5] O. Cramer, “The variation of the specific heat ratio and the speed of sound in air with temperature, pressure, humidity, and co2 concentration,” *The Journal of the Acoustical Society of America*, vol.93, no.5, pp.2510–2516, 1993.

[6] K. Rasmussen, “Calculation methods for the physical properties of air used in the calibration of microphones,” 1997.

[7] R. S. Davis, “Equation for the determination of the density of moist air (1981/91),” *Metrologia*, vol.29, no.1, pp.67–70, 1992.

[8] STMicroelectronics, “An2586 application note,” *Doc ID 13675 Rev 7*, 2011.

[9] B. Sensortec, “Digital pressure sensor,” *May 7th*, 2015.

[10] M. Integrated, “Max44009 industry’s lowest-power ambient light sensor with adc,” *19-5719; Rev 0; 1/11*.

[11] Honeywell, “3-axis digital compass ic hmc5883l,” *Form 900405 Rev E*, 2013.

[12] SEMTEH, “Sx1276/77/78/79 - 137 mhz to 1020 mhz low power long range transceiver,” *Rev. 4*, 2015.

[13] ASAIR, “Aht10 technical manual,” *V1.1*, 2018.

پیوست

سورس کد ایستگاه

main.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f1xx_hal.h"

```

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/
#define LoRa_NSS_Pin GPIO_PIN_4
#define LoRa_NSS_GPIO_Port GPIOA
#define LoRa_RST_Pin GPIO_PIN_0
#define LoRa_RST_GPIO_Port GPIOB
#define LoRa_DIO_Pin GPIO_PIN_1
#define LoRa_DIO_GPIO_Port GPIOB
#define LoRa_DIO_EXTI_IRQn EXTI1_IRQn
/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

main.c

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.c
 * @brief          : Main program body
 ******************************************************************************
 * @attention
 *
 * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
```

```
10  * All rights reserved.</center></h2>
11  *
12  * This software component is licensed by ST under BSD 3-Clause license,
13  * the "License"; You may not use this file except in compliance with the
14  * License. You may obtain a copy of the License at:
15  *                        opensource.org/licenses/BSD-3-Clause
16  *
17  ******************************************************************************
18  */
19 /* USER CODE END Header */
20
21 /* Includes ------------------------------------------------------------------*/
22 #include "main.h"
23 #include "usb_device.h"
24
25 /* Private includes ----------------------------------------------------------*/
26 /* USER CODE BEGIN Includes */
27 #include <stdlib.h>
28 #include <string.h>
29 #include <math.h>
30 #include "usbd_cdc_if.h"
31 #include "SX1278.h"
32 /* USER CODE END Includes */
33
34 /* Private typedef -----------------------------------------------------------*/
35 /* USER CODE BEGIN PTD */
36
37 /* USER CODE END PTD */
38
39 /* Private define ------------------------------------------------------------*/
40 /* USER CODE BEGIN PD */
41
42 // LoRa device id
43 #define DEVICE_ID 65535
44
45 /* USER CODE END PD */
46
47 /* Private macro -------------------------------------------------------------*/
48 /* USER CODE BEGIN PM */
49
50 /* USER CODE END PM */
51
52 /* Private variables ---------------------------------------------------------*/
53 SPI_HandleTypeDef hspi1;
54
55 /* USER CODE BEGIN PV */
56 uint8_t LoRa_Len;
57 uint8_t Buffer[32];
58 float RXData[8];
59 SX1278_hw_t SX1278_hw;
60 SX1278_t SX1278;
61 /* USER CODE END PV */
62
63 /* Private function prototypes -----------------------------------------------*/
64 void SystemClock_Config(void);
65 static void MX_GPIO_Init(void);
66 static void MX_SPI1_Init(void);
67 /* USER CODE BEGIN PFP */
```

```
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void) {
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_SPI1_Init();
  MX_USB_DEVICE_Init();
  /* USER CODE BEGIN 2 */

  // Initialize LoRa
  SX1278_hw.dio0.port = LoRa_DIO_GPIO_Port;
  SX1278_hw.dio0.pin = LoRa_DIO_Pin;
  SX1278_hw.nss.port = LoRa_NSS_GPIO_Port;
  SX1278_hw.nss.pin = LoRa_NSS_Pin;
  SX1278_hw.reset.port = LoRa_RST_GPIO_Port;
  SX1278_hw.reset.pin = LoRa_RST_Pin;
  SX1278_hw.spi = &hspi1;
  SX1278.hw = &SX1278_hw;
  SX1278_begin(&SX1278, SX1278_433MHZ, SX1278_POWER_20DBM, SX1278_LORA_SF_10,
      SX1278_LORA_BW_250KHZ, 32);

  // Set LoRa on Continuous RX mode
  SX1278_LoRaEntryRx(&SX1278, 32, 2000);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
```

```
  while (1) {
    /* After waking up from sleep or powerup */

    // Check if is there any data received on LoRa
    LoRa_Len = SX1278_LoRaRxPacket(&SX1278);
    while (LoRa_Len > 0) { // Read received data while data present
      // Read data
      SX1278_read(&SX1278, Buffer, LoRa_Len);
      // convert to float array to check receiver device id before sending over USB
      memcpy(RXData, Buffer, sizeof(Buffer));
      /**
       * RXData is as follow:
       * RXData[0]: Sender device id
       * RXData[1]: Receiver device id
       * RXData[2]: Temperature
       * RXData[3]: Pressure
       * RXData[4]: Light intensity
       * RXData[5]: Humidity
       * RXData[6]: Wind speed
       * RXData[7]: Wind direction
       */
      // Check if receiver device id matches with this device and ignore data if not
      if ((uint16_t) RXData[1] == DEVICE_ID) {
        // Send received data over USB
        CDC_Transmit_FS(Buffer, sizeof(Buffer));
      }

      // Recheck if data present
      LoRa_Len = SX1278_LoRaRxPacket(&SX1278);
    }

    // Sleep until interrupt accrued
    HAL_SuspendTick();
    HAL_PWR_EnterSLEEPMode(PWR_LOWPOWERREGULATOR_ON, PWR_SLEEPENTRY_WFI);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void) {
  RCC_OscInitTypeDef RCC_OscInitStruct = { 0 };
  RCC_ClkInitTypeDef RCC_ClkInitStruct = { 0 };
  RCC_PeriphCLKInitTypeDef PeriphClkInit = { 0 };

  /** Initializes the CPU, AHB and APB busses clocks
   */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
```

```
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB busses clocks
   */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
      | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK) {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USB;
  PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_PLL_DIV1_5;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK) {
    Error_Handler();
  }
}

/**
 * @brief SPI1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_SPI1_Init(void) {

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */

  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_128;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK) {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

```

```
}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void) {
  GPIO_InitTypeDef GPIO_InitStruct = { 0 };

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LoRa_NSS_GPIO_Port, LoRa_NSS_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LoRa_RST_GPIO_Port, LoRa_RST_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : LoRa_NSS_Pin */
  GPIO_InitStruct.Pin = LoRa_NSS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LoRa_NSS_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LoRa_RST_Pin */
  GPIO_InitStruct.Pin = LoRa_RST_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LoRa_RST_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LoRa_DIO_Pin */
  GPIO_InitStruct.Pin = LoRa_DIO_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(LoRa_DIO_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI1_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI1_IRQn);

}

/* USER CODE BEGIN 4 */

/**
 * @brief On external interrupt accrued this function will be called.
 * @param GPIO_Pin
 * @retval None
 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
  // Check if interrupt is from LoRa DIO
  if (GPIO_Pin == GPIO_PIN_1) {
    // Enable tick
```

```
    HAL_ResumeTick();
  }
}

/* USER CODE END 4 */

/**
 * @brief  This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void) {
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

SX1278.h

از این فایل به طور مشترک در سمت سنسور نیز استفاده شده است.

```
/**
 * Author Wojciech Domski <Wojciech.Domski@gmail.com>
 * www: www.Domski.pl
 *
 * work based on DORJI.COM sample code and
 * https://github.com/realspinner/SX1278_LoRa
 */

#ifndef __SX1278_H__
#define __SX1278_H__

#include <stdint.h>
#include <stdbool.h>
#include "stm32f1xx_hal.h"

#define SX1278_MAX_PACKET 256
#define SX1278_DEFAULT_TIMEOUT      3000

//Error Coding rate (CR)setting
```

```
20 #define SX1278_CR_4_5
21 //#define SX1278_CR_4_6
22 //#define SX1278_CR_4_7
23 //#define SX1278_CR_4_8
24 #ifdef   SX1278_CR_4_5
25 #define SX1278_CR  0x01
26 #else
27 #ifdef   SX1278_CR_4_6
28 #define SX1278_CR     0x02
29 #else
30 #ifdef   SX1278_CR_4_7
31 #define SX1278_CR     0x03
32 #else
33 #ifdef   SX1278_CR_4_8
34 #define SX1278_CR     0x04
35 #endif
36 #endif
37 #endif
38 #endif
39
40 //CRC Enable
41 #define SX1278_CRC_EN
42 #ifdef  SX1278_CRC_EN
43 #define SX1278_CRC    0x01
44 #else
45 #define SX1278_CRC    0x00
46 #endif
47 //RFM98 Internal registers Address
48 /*******************LoRa mode***************************/
49 #define LR_RegFifo                                  0x00
50 // Common settings
51 #define LR_RegOpMode                                0x01
52 #define LR_RegFrMsb                                 0x06
53 #define LR_RegFrMid                                 0x07
54 #define LR_RegFrLsb                                 0x08
55 // Tx settings
56 #define LR_RegPaConfig                              0x09
57 #define LR_RegPaRamp                                0x0A
58 #define LR_RegOcp                                   0x0B
59 // Rx settings
60 #define LR_RegLna                                   0x0C
61 // LoRa registers
62 #define LR_RegFifoAddrPtr                           0x0D
63 #define LR_RegFifoTxBaseAddr                        0x0E
64 #define LR_RegFifoRxBaseAddr                        0x0F
65 #define LR_RegFifoRxCurrentaddr                     0x10
66 #define LR_RegIrqFlagsMask                          0x11
67 #define LR_RegIrqFlags                              0x12
68 #define LR_RegRxNbBytes                             0x13
69 #define LR_RegRxHeaderCntValueMsb                   0x14
70 #define LR_RegRxHeaderCntValueLsb                   0x15
71 #define LR_RegRxPacketCntValueMsb                   0x16
72 #define LR_RegRxPacketCntValueLsb                   0x17
73 #define LR_RegModemStat                             0x18
74 #define LR_RegPktSnrValue                           0x19
75 #define LR_RegPktRssiValue                          0x1A
76 #define LR_RegRssiValue                             0x1B
77 #define LR_RegHopChannel                            0x1C
```

```
#define LR_RegModemConfig1                    0x1D
#define LR_RegModemConfig2                    0x1E
#define LR_RegSymbTimeoutLsb                  0x1F
#define LR_RegPreambleMsb                     0x20
#define LR_RegPreambleLsb                     0x21
#define LR_RegPayloadLength                   0x22
#define LR_RegMaxPayloadLength                0x23
#define LR_RegHopPeriod                       0x24
#define LR_RegFifoRxByteAddr                  0x25
// I/O settings
#define REG_LR_DIOMAPPING1                    0x40
#define REG_LR_DIOMAPPING2                    0x41
// Version
#define REG_LR_VERSION                        0x42
// Additional settings
#define REG_LR_PLLHOP                         0x44
#define REG_LR_TCXO                           0x4B
#define REG_LR_PADAC                          0x4D
#define REG_LR_FORMERTEMP                     0x5B
#define REG_LR_AGCREF                         0x61
#define REG_LR_AGCTHRESH1                     0x62
#define REG_LR_AGCTHRESH2                     0x63
#define REG_LR_AGCTHRESH3                     0x64

/*******************FSK/ook mode***************************/
#define  RegFIFO              0x00
#define  RegOpMode            0x01
#define  RegBitRateMsb        0x02
#define  RegBitRateLsb        0x03
#define  RegFdevMsb           0x04
#define  RegFdevLsb           0x05
#define  RegFreqMsb           0x06
#define  RegFreqMid           0x07
#define  RegFreqLsb           0x08
#define  RegPaConfig          0x09
#define  RegPaRamp            0x0a
#define  RegOcp               0x0b
#define  RegLna               0x0c
#define  RegRxConfig          0x0d
#define  RegRssiConfig        0x0e
#define  RegRssiCollision     0x0f
#define  RegRssiThresh        0x10
#define  RegRssiValue         0x11
#define  RegRxBw              0x12
#define  RegAfcBw             0x13
#define  RegOokPeak           0x14
#define  RegOokFix            0x15
#define  RegOokAvg            0x16
#define  RegAfcFei            0x1a
#define  RegAfcMsb            0x1b
#define  RegAfcLsb            0x1c
#define  RegFeiMsb            0x1d
#define  RegFeiLsb            0x1e
#define  RegPreambleDetect    0x1f
#define  RegRxTimeout1        0x20
#define  RegRxTimeout2        0x21
#define  RegRxTimeout3        0x22
#define  RegRxDelay           0x23
```

```
#define  RegOsc                0x24
#define  RegPreambleMsb        0x25
#define  RegPreambleLsb        0x26
#define  RegSyncConfig         0x27
#define  RegSyncValue1         0x28
#define  RegSyncValue2         0x29
#define  RegSyncValue3         0x2a
#define  RegSyncValue4         0x2b
#define  RegSyncValue5         0x2c
#define  RegSyncValue6         0x2d
#define  RegSyncValue7         0x2e
#define  RegSyncValue8         0x2f
#define  RegPacketConfig1      0x30
#define  RegPacketConfig2      0x31
#define  RegPayloadLength      0x32
#define  RegNodeAdrs           0x33
#define  RegBroadcastAdrs      0x34
#define  RegFifoThresh         0x35
#define  RegSeqConfig1         0x36
#define  RegSeqConfig2         0x37
#define  RegTimerResol         0x38
#define  RegTimer1Coef         0x39
#define  RegTimer2Coef         0x3a
#define  RegImageCal           0x3b
#define  RegTemp               0x3c
#define  RegLowBat             0x3d
#define  RegIrqFlags1          0x3e
#define  RegIrqFlags2          0x3f
#define  RegDioMapping1        0x40
#define  RegDioMapping2        0x41
#define  RegVersion            0x42
#define  RegPllHop             0x44
#define  RegPaDac              0x4d
#define  RegBitRateFrac        0x5d

/*******************************************************
 **Parameter table define
 *******************************************************/
#define SX1278_433MHZ       0

static const uint8_t SX1278_Frequency[1][3] = { { 0x6C, 0x40, 0x00 }, //433MHz
    (433000000.0 / 61.03515625)
    };

#define SX1278_POWER_20DBM      0

static const uint8_t SX1278_Power[4] = { 0x8f, //20dbm

    };

#define SX1278_LORA_SF_6      0
#define SX1278_LORA_SF_7      1
#define SX1278_LORA_SF_8      2
#define SX1278_LORA_SF_9      3
#define SX1278_LORA_SF_10     4
#define SX1278_LORA_SF_11     5
#define SX1278_LORA_SF_12     6

```

```
static const uint8_t SX1278_SpreadFactor[7] = { 6, 7, 8, 9, 10, 11, 12 };

#define SX1278_LORA_BW_7_8KHZ    0
#define SX1278_LORA_BW_10_4KHZ    1
#define SX1278_LORA_BW_15_6KHZ    2
#define SX1278_LORA_BW_20_8KHZ    3
#define SX1278_LORA_BW_31_2KHZ    4
#define SX1278_LORA_BW_41_7KHZ    5
#define SX1278_LORA_BW_62_5KHZ    6
#define SX1278_LORA_BW_125KHZ    7
#define SX1278_LORA_BW_250KHZ    8
#define SX1278_LORA_BW_500KHZ    9

static const uint8_t SX1278_LoRaBandwidth[10] = { 0, //    7.8KHz,
    1, //   10.4KHz,
    2, //   15.6KHz,
    3, //   20.8KHz,
    4, //   31.2KHz,
    5, //   41.7KHz,
    6, //   62.5KHz,
    7, // 125.0KHz,
    8, // 250.0KHz,
    9  // 500.0KHz
    };

typedef enum _SX1278_STATUS {
  SLEEP, STANDBY, TX, RX
} SX1278_Status_t;

typedef struct {
  int pin;
  void * port;
} SX1278_hw_dio_t;

typedef struct {
  SX1278_hw_dio_t reset;
  SX1278_hw_dio_t dio0;
  SX1278_hw_dio_t nss;
  void * spi;
} SX1278_hw_t;

typedef struct {
  SX1278_hw_t * hw;

  uint8_t frequency;
  uint8_t power;
  uint8_t LoRa_Rate;
  uint8_t LoRa_BW;
  uint8_t packetLength;

  SX1278_Status_t status;

  uint8_t rxBuffer[SX1278_MAX_PACKET];
  uint8_t readBytes;
} SX1278_t;

//hardware
__weak void SX1278_hw_init(SX1278_hw_t * hw);
```

```
__weak void SX1278_hw_SetNSS(SX1278_hw_t * hw, int value);
__weak void SX1278_hw_Reset(SX1278_hw_t * hw);
__weak void SX1278_hw_SPICommand(SX1278_hw_t * hw, uint8_t cmd);
__weak uint8_t SX1278_hw_SPIReadByte(SX1278_hw_t * hw);
__weak void SX1278_hw_DelayMs(uint32_t msec);
__weak int SX1278_hw_GetDIO0(SX1278_hw_t * hw);

//logic

uint8_t SX1278_SPIRead(SX1278_t * module, uint8_t addr);
void SX1278_SPIWrite(SX1278_t * module, uint8_t addr, uint8_t cmd);
void SX1278_SPIBurstRead(SX1278_t * module, uint8_t addr, uint8_t *rxBuf,
    uint8_t length);
void SX1278_SPIBurstWrite(SX1278_t * module, uint8_t addr, uint8_t *txBuf,
    uint8_t length);
void SX1278_DIO0_InterruptHandler(SX1278_t * module);

void SX1278_config(SX1278_t * module, uint8_t frequency, uint8_t power,
    uint8_t LoRa_Rate, uint8_t LoRa_BW);
void SX1278_defaultConfig(SX1278_t * module);

void SX1278_entryLoRa(SX1278_t * module);
void SX1278_clearLoRaIrq(SX1278_t * module);
int SX1278_LoRaEntryRx(SX1278_t * module, uint8_t length, uint32_t timeout);
uint8_t SX1278_LoRaRxPacket(SX1278_t * module);
int SX1278_LoRaEntryTx(SX1278_t * module, uint8_t length, uint32_t timeout);
int SX1278_LoRaTxPacket(SX1278_t * module, uint8_t *txBuf, uint8_t length,
    uint32_t timeout);

void SX1278_begin(SX1278_t * module, uint8_t frequency, uint8_t power,
    uint8_t LoRa_Rate, uint8_t LoRa_BW, uint8_t packetLength);

int SX1278_transmit(SX1278_t * module, uint8_t *txBuf, uint8_t length,
    uint32_t timeout);
int SX1278_(SX1278_t * module, uint8_t length, uint32_t timeoutT);
uint8_t SX1278_available(SX1278_t * module);
uint8_t SX1278_read(SX1278_t * module, uint8_t *rxBuf, uint8_t length);

uint8_t SX1278_RSSI_LoRa(SX1278_t * module);
uint8_t SX1278_RSSI(SX1278_t * module);

void SX1278_standby(SX1278_t * module);
void SX1278_sleep(SX1278_t * module);

#endif
```

SX1278.c

از این فایل به طور مشترک در سمت سنسور نیز استفاده شده است.

```
/**
 * Author Wojciech Domski <Wojciech.Domski@gmail.com>
 * www: www.Domski.pl
 *
 * work based on DORJI.COM sample code and
 * https://github.com/realspinner/SX1278_LoRa
 */

```

```
#include "SX1278.h"
#include <string.h>


//////////////////////////////////
// logic
//////////////////////////////////

__weak void SX1278_hw_init(SX1278_hw_t * hw) {
  SX1278_hw_SetNSS(hw, 1);
  HAL_GPIO_WritePin(hw->reset.port, hw->reset.pin, GPIO_PIN_SET);
}

__weak void SX1278_hw_SetNSS(SX1278_hw_t * hw, int value) {
  HAL_GPIO_WritePin(hw->nss.port, hw->nss.pin,
      (value == 1) ? GPIO_PIN_SET : GPIO_PIN_RESET);
}

__weak void SX1278_hw_Reset(SX1278_hw_t * hw) {
  SX1278_hw_SetNSS(hw, 1);
  HAL_GPIO_WritePin(hw->reset.port, hw->reset.pin, GPIO_PIN_RESET);

  SX1278_hw_DelayMs(1);

  HAL_GPIO_WritePin(hw->reset.port, hw->reset.pin, GPIO_PIN_SET);

  SX1278_hw_DelayMs(100);
}

__weak void SX1278_hw_SPICommand(SX1278_hw_t * hw, uint8_t cmd) {
  SX1278_hw_SetNSS(hw, 0);
  HAL_SPI_Transmit(hw->spi, &cmd, 1, 1000);
  while (HAL_SPI_GetState(hw->spi) != HAL_SPI_STATE_READY)
    ;
}

__weak uint8_t SX1278_hw_SPIReadByte(SX1278_hw_t * hw) {
  uint8_t txByte = 0x00;
  uint8_t rxByte = 0x00;

  SX1278_hw_SetNSS(hw, 0);
  HAL_SPI_TransmitReceive(hw->spi, &txByte, &rxByte, 1, 1000);
  while (HAL_SPI_GetState(hw->spi) != HAL_SPI_STATE_READY)
    ;
  return rxByte;
}

__weak void SX1278_hw_DelayMs(uint32_t msec) {
  HAL_Delay(msec);
}

__weak int SX1278_hw_GetDIO0(SX1278_hw_t * hw) {
  return (HAL_GPIO_ReadPin(hw->dio0.port, hw->dio0.pin) == GPIO_PIN_SET);
}

//////////////////////////////////
// logic
//////////////////////////////////
```

```
uint8_t SX1278_SPIRead(SX1278_t * module, uint8_t addr) {
  uint8_t tmp;
  SX1278_hw_SPICommand(module->hw, addr);
  tmp = SX1278_hw_SPIReadByte(module->hw);
  SX1278_hw_SetNSS(module->hw, 1);
  return tmp;
}

void SX1278_SPIWrite(SX1278_t * module, uint8_t addr, uint8_t cmd) {
  SX1278_hw_SetNSS(module->hw, 0);
  SX1278_hw_SPICommand(module->hw, addr | 0x80);
  SX1278_hw_SPICommand(module->hw, cmd);
  SX1278_hw_SetNSS(module->hw, 1);
}

void SX1278_SPIBurstRead(SX1278_t * module, uint8_t addr, uint8_t* rxBuf,
    uint8_t length) {
  uint8_t i;
  if (length <= 1) {
    return;
  } else {
    SX1278_hw_SetNSS(module->hw, 0);
    SX1278_hw_SPICommand(module->hw, addr);
    for (i = 0; i < length; i++) {
      *(rxBuf + i) = SX1278_hw_SPIReadByte(module->hw);
    }
    SX1278_hw_SetNSS(module->hw, 1);
  }
}

void SX1278_SPIBurstWrite(SX1278_t * module, uint8_t addr, uint8_t* txBuf,
    uint8_t length) {
  unsigned char i;
  if (length <= 1) {
    return;
  } else {
    SX1278_hw_SetNSS(module->hw, 0);
    SX1278_hw_SPICommand(module->hw, addr | 0x80);
    for (i = 0; i < length; i++) {
      SX1278_hw_SPICommand(module->hw, *(txBuf + i));
    }
    SX1278_hw_SetNSS(module->hw, 1);
  }
}

void SX1278_defaultConfig(SX1278_t * module) {
  SX1278_config(module, module->frequency, module->power, module->LoRa_Rate,
      module->LoRa_BW);
}

void SX1278_config(SX1278_t * module, uint8_t frequency, uint8_t power,
    uint8_t LoRa_Rate, uint8_t LoRa_BW) {
  SX1278_sleep(module); //Change modem mode Must in Sleep mode
  SX1278_hw_DelayMs(15);

  SX1278_entryLoRa(module);
```

```
  //SX1278_SPIWrite(module, 0x5904); //?? Change digital regulator form 1.6V to 1.47V: see errata note

  SX1278_SPIBurstWrite(module, LR_RegFrMsb,
      (uint8_t*) SX1278_Frequency[frequency], 3); //setting  frequency parameter

  //setting base parameter
  SX1278_SPIWrite(module, LR_RegPaConfig, SX1278_Power[power]); //Setting output power parameter

  SX1278_SPIWrite(module, LR_RegOcp, 0x0B);       //RegOcp,Close Ocp
  SX1278_SPIWrite(module, LR_RegLna, 0x23);     //RegLNA,High & LNA Enable
  if (SX1278_SpreadFactor[LoRa_Rate] == 6) {   //SFactor=6
    uint8_t tmp;
    SX1278_SPIWrite(module,
    LR_RegModemConfig1,
        ((SX1278_LoRaBandwidth[LoRa_BW] << 4) + (SX1278_CR << 1) + 0x01)); //Implicit Enable CRC Enable(0x02) & Error Coding rate 4/5(0x01), 4/6(0x02), 4/7(0x03), 4/8(0x04)

    SX1278_SPIWrite(module,
    LR_RegModemConfig2,
        ((SX1278_SpreadFactor[LoRa_Rate] << 4) + (SX1278_CRC << 2)
            + 0x03));

    tmp = SX1278_SPIRead(module, 0x31);
    tmp &= 0xF8;
    tmp |= 0x05;
    SX1278_SPIWrite(module, 0x31, tmp);
    SX1278_SPIWrite(module, 0x37, 0x0C);
  } else {
    SX1278_SPIWrite(module,
    LR_RegModemConfig1,
        ((SX1278_LoRaBandwidth[LoRa_BW] << 4) + (SX1278_CR << 1) + 0x00)); //Explicit Enable CRC Enable(0x02) & Error Coding rate 4/5(0x01), 4/6(0x02), 4/7(0x03), 4/8(0x04)

    SX1278_SPIWrite(module,
    LR_RegModemConfig2,
        ((SX1278_SpreadFactor[LoRa_Rate] << 4) + (SX1278_CRC << 2)
            + 0x03)); //SFactor & LNA gain set by the internal AGC loop
  }

  SX1278_SPIWrite(module, LR_RegSymbTimeoutLsb, 0xFF); //RegSymbTimeoutLsb Timeout = 0x3FF(Max)
  SX1278_SPIWrite(module, LR_RegPreambleMsb, 0x00); //RegPreambleMsb
  SX1278_SPIWrite(module, LR_RegPreambleLsb, 12); //RegPreambleLsb 8+4=12byte Preamble
  SX1278_SPIWrite(module, REG_LR_DIOMAPPING2, 0x01); //RegDioMapping2 DIO5=00, DIO4=01
  module->readBytes = 0;
  SX1278_standby(module); //Entry standby mode
}

void SX1278_standby(SX1278_t * module) {
  SX1278_SPIWrite(module, LR_RegOpMode, 0x09);
  module->status = STANDBY;
}

void SX1278_sleep(SX1278_t * module) {
```

```
  SX1278_SPIWrite(module, LR_RegOpMode, 0x08);
  module->status = SLEEP;
}

void SX1278_entryLoRa(SX1278_t * module) {
  SX1278_SPIWrite(module, LR_RegOpMode, 0x88);
}

void SX1278_clearLoRaIrq(SX1278_t * module) {
  SX1278_SPIWrite(module, LR_RegIrqFlags, 0xFF);
}

int SX1278_LoRaEntryRx(SX1278_t * module, uint8_t length, uint32_t timeout) {
  uint8_t addr;

  module->packetLength = length;

  SX1278_defaultConfig(module);    //Setting base parameter
  SX1278_SPIWrite(module, REG_LR_PADAC, 0x87);  //Normal and RX
  SX1278_SPIWrite(module, LR_RegHopPeriod, 0xFF); //No FHSS
  SX1278_SPIWrite(module, REG_LR_DIOMAPPING1, 0x01);//DIO=00,DIO1=00,DIO2=00, DIO3=01
  SX1278_SPIWrite(module, LR_RegIrqFlagsMask, 0x3F);//Open RxDone interrupt & Timeout
  SX1278_clearLoRaIrq(module);
  SX1278_SPIWrite(module, LR_RegPayloadLength, length);//Payload Length 21byte(this
    register must difine when the data long of one byte in SF is 6)
  addr = SX1278_SPIRead(module, LR_RegFifoRxBaseAddr); //Read RxBaseAddr
  SX1278_SPIWrite(module, LR_RegFifoAddrPtr, addr); //RxBaseAddr->FiFoAddrPtr
  SX1278_SPIWrite(module, LR_RegOpMode, 0x8d);  //Mode//Low Frequency Mode
  module->readBytes = 0;

  while (1) {
    if ((SX1278_SPIRead(module, LR_RegModemStat) & 0x04) == 0x04) { //Rx-on going
    RegModemStat
      module->status = RX;
      return 1;
    }
    if (--timeout == 0) {
      SX1278_hw_Reset(module->hw);
      SX1278_defaultConfig(module);
      return 0;
    }
    SX1278_hw_DelayMs(1);
  }
}

uint8_t SX1278_LoRaRxPacket(SX1278_t * module) {
  unsigned char addr;
  unsigned char packet_size;

  if (SX1278_hw_GetDIO0(module->hw)) {
    memset(module->rxBuffer, 0x00, SX1278_MAX_PACKET);

    addr = SX1278_SPIRead(module, LR_RegFifoRxCurrentaddr); //last packet addr
    SX1278_SPIWrite(module, LR_RegFifoAddrPtr, addr); //RxBaseAddr -> FiFoAddrPtr

    if (module->LoRa_Rate == SX1278_LORA_SF_6) { //When SpreadFactor is six,will used
    Implicit Header mode(Excluding internal packet length)
      packet_size = module->packetLength;
```

```
    } else {
      packet_size = SX1278_SPIRead(module, LR_RegRxNbBytes); //Number for received
    bytes
    }

    SX1278_SPIBurstRead(module, 0x00, module->rxBuffer, packet_size);
    module->readBytes = packet_size;
    SX1278_clearLoRaIrq(module);
  }
  return module->readBytes;
}

int SX1278_LoRaEntryTx(SX1278_t * module, uint8_t length, uint32_t timeout) {
  uint8_t addr;
  uint8_t temp;

  module->packetLength = length;

  SX1278_defaultConfig(module); //setting base parameter
  SX1278_SPIWrite(module, REG_LR_PADAC, 0x87);  //Tx for 20dBm
  SX1278_SPIWrite(module, LR_RegHopPeriod, 0x00); //RegHopPeriod NO FHSS
  SX1278_SPIWrite(module, REG_LR_DIOMAPPING1, 0x41); //DIO0=01, DIO1=00,DIO2=00, DIO3
    =01
  SX1278_clearLoRaIrq(module);
  SX1278_SPIWrite(module, LR_RegIrqFlagsMask, 0xF7); //Open TxDone interrupt
  SX1278_SPIWrite(module, LR_RegPayloadLength, length); //RegPayloadLength 21byte
  addr = SX1278_SPIRead(module, LR_RegFifoTxBaseAddr); //RegFiFoTxBaseAddr
  SX1278_SPIWrite(module, LR_RegFifoAddrPtr, addr); //RegFifoAddrPtr

  while (1) {
    temp = SX1278_SPIRead(module, LR_RegPayloadLength);
    if (temp == length) {
      module->status = TX;
      return 1;
    }

    if (--timeout == 0) {
      SX1278_hw_Reset(module->hw);
      SX1278_defaultConfig(module);
      return 0;
    }
  }
}

int SX1278_LoRaTxPacket(SX1278_t * module, uint8_t* txBuffer, uint8_t length,
    uint32_t timeout) {
  SX1278_SPIBurstWrite(module, 0x00, txBuffer, length);
  SX1278_SPIWrite(module, LR_RegOpMode, 0x8b);  //Tx Mode
  while (1) {
    if (SX1278_hw_GetDIO0(module->hw)) { //if(Get_NIRQ()) //Packet send over
      SX1278_SPIRead(module, LR_RegIrqFlags);
      SX1278_clearLoRaIrq(module); //Clear irq
      SX1278_standby(module); //Entry Standby mode
      return 1;
    }

    if (--timeout == 0) {
      SX1278_hw_Reset(module->hw);
```

```
      SX1278_defaultConfig(module);
      return 0;
    }
    SX1278_hw_DelayMs(1);
  }
}

void SX1278_begin(SX1278_t * module, uint8_t frequency, uint8_t power,
    uint8_t LoRa_Rate, uint8_t LoRa_BW, uint8_t packetLength) {
  SX1278_hw_init(module->hw);
  module->frequency = frequency;
  module->power = power;
  module->LoRa_Rate = LoRa_Rate;
  module->LoRa_BW = LoRa_BW;
  module->packetLength = packetLength;
  SX1278_defaultConfig(module);
}

int SX1278_transmit(SX1278_t * module, uint8_t* txBuf, uint8_t length,
    uint32_t timeout) {
  if (SX1278_LoRaEntryTx(module, length, timeout)) {
    return SX1278_LoRaTxPacket(module, txBuf, length, timeout);
  }
  return 0;
}

int SX1278_receive(SX1278_t * module, uint8_t length, uint32_t timeout) {
  return SX1278_LoRaEntryRx(module, length, timeout);
}

uint8_t SX1278_available(SX1278_t * module) {
  return SX1278_LoRaRxPacket(module);
}

uint8_t SX1278_read(SX1278_t * module, uint8_t* rxBuf, uint8_t length) {
  if (length != module->readBytes)
    length = module->readBytes;
  memcpy(rxBuf, module->rxBuffer, length);
  rxBuf[length] = '\0';
  module->readBytes = 0;
  return length;
}

uint8_t SX1278_RSSI_LoRa(SX1278_t * module) {
  uint32_t temp = 10;
  temp = SX1278_SPIRead(module, LR_RegRssiValue); //Read RegRssiValue, Rssi value
  temp = temp + 127 - 137; //127:Max RSSI, 137:RSSI offset
  return (uint8_t) temp;
}

uint8_t SX1278_RSSI(SX1278_t * module) {
  uint8_t temp = 0xff;
  temp = SX1278_SPIRead(module, 0x11);
  temp = 127 - (temp >> 1); //127:Max RSSI
  return temp;
}
```

سورس کد سنسور

main.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f1xx_hal.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
```

```
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines ------------------------------------------------------------*/
#define LoRa_NSS_Pin GPIO_PIN_4
#define LoRa_NSS_GPIO_Port GPIOA
#define LoRa_RST_Pin GPIO_PIN_4
#define LoRa_RST_GPIO_Port GPIOC
#define LoRa_DIO_Pin GPIO_PIN_5
#define LoRa_DIO_GPIO_Port GPIOC
#define X_axis_echo_Pin GPIO_PIN_9
#define X_axis_echo_GPIO_Port GPIOE
#define X_axis_trigger_Pin GPIO_PIN_10
#define X_axis_trigger_GPIO_Port GPIOE
#define Y_axis_echo_Pin GPIO_PIN_11
#define Y_axis_echo_GPIO_Port GPIOE
#define Y_axis_trigger_Pin GPIO_PIN_12
#define Y_axis_trigger_GPIO_Port GPIOE
#define workingMode_Pin GPIO_PIN_6
#define workingMode_GPIO_Port GPIOC
/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

main.c

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.c
 * @brief          : Weather Station Sensor Side Codes
 ******************************************************************************
 * @attention
 *
 * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST under BSD 3-Clause license,
 * the "License"; You may not use this file except in compliance with the
 * License. You may obtain a copy of the License at:
 *                        opensource.org/licenses/BSD-3-Clause
 *
 ******************************************************************************
 */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
```

```
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <string.h>
#include "BMP180.h"
#include "MAX44009.h"
#include "AHT10.h"
#include "QMC5883L.h"
#include "HCSR05.h"
#include "SX1278.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

// Transducers distance
#define HCSR05_DISTANCE 6

// Angle difference between wind sensor x axis and compass x axis
#define HCSR05_ANGLE_DIFFERENCE 0

// LoRa device id
#define DEVICE_ID 12345

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;

RTC_HandleTypeDef hrtc;

SPI_HandleTypeDef hspi1;

TIM_HandleTypeDef htim1;

/* USER CODE BEGIN PV */
float BMP180_Temperature, lux;
int32_t BMP180_Pressure;
int16_t compass_x, compass_y, compass_z;
int8_t TempHum[2];
float_t X_axis_wind_speed = 0;
float_t Y_axis_wind_speed = 0;
float_t wind[2];
float TXData[8];
uint8_t Buffer[32];
SX1278_hw_t SX1278_hw;
SX1278_t SX1278;
```

```
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_RTC_Init(void);
static void MX_SPI1_Init(void);
static void MX_TIM1_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */

/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void) {
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_RTC_Init();
  MX_SPI1_Init();
  MX_TIM1_Init();
  /* USER CODE BEGIN 2 */

  //BMP180 init
  BMP180_Init(&hi2c1);
  BMP180_SetOversampling(BMP180_LOW);
  BMP180_UpdateCalibrationData();

  // MAx44009 init
  MAX44009_Begin(&hi2c1);

```

```
138  // HMC5883L init
139  QMC5883L_Init(&hi2c1);
140  QMC5883L_Set_Sampling_Rate(50);
141
142  // AHT10 init
143  AHT10_Init(&hi2c1);
144
145  // HCSR05 delay init
146  HCSR05_Init();
147
148  // Initialize LoRa
149  SX1278_hw.dio0.port = LoRa_DIO_GPIO_Port;
150  SX1278_hw.dio0.pin = LoRa_DIO_Pin;
151  SX1278_hw.nss.port = LoRa_NSS_GPIO_Port;
152  SX1278_hw.nss.pin = LoRa_NSS_Pin;
153  SX1278_hw.reset.port = LoRa_RST_GPIO_Port;
154  SX1278_hw.reset.pin = LoRa_RST_Pin;
155  SX1278_hw.spi = &hspi1;
156  SX1278.hw = &SX1278_hw;
157  SX1278_begin(&SX1278, SX1278_433MHZ, SX1278_POWER_20DBM, SX1278_LORA_SF_10,
158      SX1278_LORA_BW_250KHZ, 32);
159
160  SX1278_LoRaEntryTx(&SX1278, 32, 2000);
161
162  // Setting TXData constant values
163  TXData[0] = DEVICE_ID; // Sender device id
164  TXData[1] = 65535; // Receiver device id
165
166  /* USER CODE END 2 */
167
168  /* Infinite loop */
169  /* USER CODE BEGIN WHILE */
170  while (1) {
171    // Reads temperature.
172    BMP180_Temperature = BMP180_GetTemperature();
173    // Setting TXData
174    TXData[2] = BMP180_Temperature; // Temperature
175
176    // Reads pressure.
177    BMP180_Pressure = BMP180_GetPressure();
178    // Setting TXData
179    TXData[3] = BMP180_Pressure; // Pressure
180
181    // Reads light intensity.
182    lux = MAX44009_Get_Lux();
183    // Setting TXData
184    TXData[4] = lux; // Light intensity
185
186    // Reads temperature and hum.
187    AHT10_GetTemperature_hum(TempHum);
188    // Setting TXData
189    TXData[5] = TempHum[1]; // Humidity
190
191    // Reads compass.
192    QMC5883L_ReadRaw(&compass_x, &compass_y, &compass_z);
193
194    // Get X axis wind speed
195    HCSR05_Ready(X_axis_echo_GPIO_Port, X_axis_echo_Pin, &htim1,
```

```
196         TIM_CHANNEL_1, HCSR05_DISTANCE, HCSR05_ANGLE_DIFFERENCE);
197     X_axis_wind_speed = HCSR05_Get_WindSpeed((int8_t) BMP180_Temperature,
198         (uint32_t) BMP180_Pressure, (int8_t) TempHum[1]);
199
200     // Get Y axis wind speed
201     HCSR05_Ready(Y_axis_echo_GPIO_Port, Y_axis_echo_Pin, &htim1,
202         TIM_CHANNEL_2, HCSR05_DISTANCE, HCSR05_ANGLE_DIFFERENCE);
203     Y_axis_wind_speed = HCSR05_Get_WindSpeed((int8_t) BMP180_Temperature,
204         (uint32_t) BMP180_Pressure, (int8_t) TempHum[1]);
205
206     // Calculate wind speed and direction
207     HCSR05_Calculate_WindSpeedNdAngle(X_axis_wind_speed, Y_axis_wind_speed,
208         compass_x, compass_y, wind);
209     // Setting TXData
210     TXData[6] = wind[0]; // Wind speed
211     TXData[7] = wind[1]; // Wind direction
212
213     // Transmit TXData
214     memcpy(Buffer, TXData, sizeof(TXData));
215     SX1278_transmit(&SX1278, Buffer, sizeof(Buffer), 3000);
216
217     // Check workingMode pin status
218     // GPIO_PIN_SET: Normal working mode
219     // GPIO_PIN_RESET: Continuous mode
220     if (HAL_GPIO_ReadPin(workingMode_GPIO_Port, workingMode_Pin)
221         == GPIO_PIN_SET) {
222       // It is on normal working mode
223
224       /* Taking a rest after a hard work */
225
226       // Put LoRa to sleep
227       SX1278_sleep(&SX1278);
228
229       // Put compass to sleep
230       QMC5883L_Sleep();
231
232       // Other sensors are in sleep mode by default
233
234       // Stop tick to reduce MCU power usage
235       HAL_SuspendTick();
236       // Enter stop mode and wait until interrupt occurred
237       HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);
238
239       // After alarm interrupt
240       // Reset RTC
241       MX_RTC_Init();
242     }
243
244     /* USER CODE END WHILE */
245
246     /* USER CODE BEGIN 3 */
247   }
248   /* USER CODE END 3 */
249 }
250
251 /**
252  * @brief System Clock Configuration
253  * @retval None
```

```
 */
void SystemClock_Config(void) {
  RCC_OscInitTypeDef RCC_OscInitStruct = { 0 };
  RCC_ClkInitTypeDef RCC_ClkInitStruct = { 0 };
  RCC_PeriphCLKInitTypeDef PeriphClkInit = { 0 };

  /** Initializes the CPU, AHB and APB busses clocks
   */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_LSI
      | RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV5;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.LSIState = RCC_LSI_ON;
  RCC_OscInitStruct.Prediv1Source = RCC_PREDIV1_SOURCE_PLL2;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  RCC_OscInitStruct.PLL2.PLL2State = RCC_PLL2_ON;
  RCC_OscInitStruct.PLL2.PLL2MUL = RCC_PLL2_MUL8;
  RCC_OscInitStruct.PLL2.HSEPrediv2Value = RCC_HSE_PREDIV2_DIV5;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB busses clocks
   */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
      | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK) {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_RTC;
  PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_LSI;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK) {
    Error_Handler();
  }
  /** Configure the Systick interrupt time
   */
  __HAL_RCC_PLLI2S_ENABLE();
}

/**
 * @brief I2C1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_I2C1_Init(void) {

  /* USER CODE BEGIN I2C1_Init 0 */

  /* USER CODE END I2C1_Init 0 */

  /* USER CODE BEGIN I2C1_Init 1 */
```

```
  /* USER CODE END I2C1_Init 1 */
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK) {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_Init 2 */

}

/**
 * @brief RTC Initialization Function
 * @param None
 * @retval None
 */
static void MX_RTC_Init(void) {

  /* USER CODE BEGIN RTC_Init 0 */

  /* USER CODE END RTC_Init 0 */

  RTC_TimeTypeDef sTime = { 0 };
  RTC_DateTypeDef DateToUpdate = { 0 };
  RTC_AlarmTypeDef sAlarm = { 0 };

  /* USER CODE BEGIN RTC_Init 1 */

  /* USER CODE END RTC_Init 1 */
  /** Initialize RTC Only
   */
  hrtc.Instance = RTC;
  hrtc.Init.AsynchPrediv = RTC_AUTO_1_SECOND;
  hrtc.Init.OutPut = RTC_OUTPUTSOURCE_ALARM;
  if (HAL_RTC_Init(&hrtc) != HAL_OK) {
    Error_Handler();
  }

  /* USER CODE BEGIN Check_RTC_BKUP */

  /* USER CODE END Check_RTC_BKUP */

  /** Initialize RTC and set the Time and Date
   */
  sTime.Hours = 0;
  sTime.Minutes = 0;
  sTime.Seconds = 0;

  if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BIN) != HAL_OK) {
```

```
    Error_Handler();
  }
  DateToUpdate.WeekDay = RTC_WEEKDAY_MONDAY;
  DateToUpdate.Month = RTC_MONTH_JANUARY;
  DateToUpdate.Date = 1;
  DateToUpdate.Year = 0;

  if (HAL_RTC_SetDate(&hrtc, &DateToUpdate, RTC_FORMAT_BIN) != HAL_OK) {
    Error_Handler();
  }
  /** Enable the Alarm A
   */
  sAlarm.AlarmTime.Hours = 3;
  sAlarm.AlarmTime.Minutes = 0;
  sAlarm.AlarmTime.Seconds = 0;
  sAlarm.Alarm = RTC_ALARM_A;
  if (HAL_RTC_SetAlarm_IT(&hrtc, &sAlarm, RTC_FORMAT_BIN) != HAL_OK) {
    Error_Handler();
  }
  /* USER CODE BEGIN RTC_Init 2 */

  /* USER CODE END RTC_Init 2 */

}

/**
 * @brief SPI1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_SPI1_Init(void) {

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */

  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_128;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK) {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */
```

```
}

/**
 * @brief TIM1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM1_Init(void) {

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = { 0 };
  TIM_MasterConfigTypeDef sMasterConfig = { 0 };
  TIM_IC_InitTypeDef sConfigIC = { 0 };

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 72 - 1;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 65535 - 1;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK) {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK) {
    Error_Handler();
  }
  if (HAL_TIM_IC_Init(&htim1) != HAL_OK) {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig)
      != HAL_OK) {
    Error_Handler();
  }
  sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_RISING;
  sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  sConfigIC.ICFilter = 0;
  if (HAL_TIM_IC_ConfigChannel(&htim1, &sConfigIC, TIM_CHANNEL_1) != HAL_OK) {
    Error_Handler();
  }
  if (HAL_TIM_IC_ConfigChannel(&htim1, &sConfigIC, TIM_CHANNEL_2) != HAL_OK) {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */

```

```
}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void) {
  GPIO_InitTypeDef GPIO_InitStruct = { 0 };

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LoRa_NSS_GPIO_Port, LoRa_NSS_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LoRa_RST_GPIO_Port, LoRa_RST_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOE, X_axis_trigger_Pin | Y_axis_trigger_Pin,
      GPIO_PIN_RESET);

  /*Configure GPIO pin : LoRa_NSS_Pin */
  GPIO_InitStruct.Pin = LoRa_NSS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LoRa_NSS_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LoRa_RST_Pin */
  GPIO_InitStruct.Pin = LoRa_RST_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LoRa_RST_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : LoRa_DIO_Pin workingMode_Pin */
  GPIO_InitStruct.Pin = LoRa_DIO_Pin | workingMode_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : X_axis_trigger_Pin Y_axis_trigger_Pin */
  GPIO_InitStruct.Pin = X_axis_trigger_Pin | Y_axis_trigger_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/**
```

```
 * @brief Timer Capture Callback
 * @param htim
 * @retval None
 */
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {
  HCSR05_TIM_Callback(htim);
}

/**
 * @brief Alarm Event Callback Function
 * @param hrtc
 * @retval None
 */
void HAL_RTC_AlarmAEventCallback(RTC_HandleTypeDef *hrtc) {
  SystemClock_Config();
  HAL_ResumeTick();
}

/* USER CODE END 4 */

/**
 * @brief  This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void) {
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

SX1278.h

این فایل به طور مشترک در بخش ایستگاه نیز استفاده شده است و سورس‌کد آن در بخش قبل آمده است.

SX1278.c

این فایل به طور مشترک در بخش ایستگاه نیز استفاده شده است و سورس‌کد آن در بخش قبل آمده است.

BMP180.h

```
/* An STM32 HAL library written for the BMP180 temperature/pressure sensor. */
/* Library by @eepj www.github.com/eepj */
#ifndef BMP180_H
#define BMP180_H

#include "stm32f1xx_hal.h"

#define BMP180_I2C_ADDR          0x77
#define BMP180_I2C_TIMEOUT         1000
#define BMP180_CONTROL_REG         0xf4
#define BMP180_MSB_REG           0xf6
#define BMP180_LSB_REG           0xf7
#define BMP180_XLSB_REG          0xf8
#define BMP180_SOFT_RESET_REG      0xe0
#define BMP180_SOFT_RESET        0xb6

typedef enum BMP180_OSS {
  BMP180_LOW, BMP180_STANDARD, BMP180_HIGH, BMP180_ULTRA,
} BMP180_OSS;

typedef enum BMP180_CalIndex {
  BMP180_INDEX_AC1,
  BMP180_INDEX_AC2,
  BMP180_INDEX_AC3,
  BMP180_INDEX_AC4,
  BMP180_INDEX_AC5,
  BMP180_INDEX_AC6,
  BMP180_INDEX_B1,
  BMP180_INDEX_B2,
  BMP180_INDEX_MB,
  BMP180_INDEX_MC,
  BMP180_INDEX_MD,
} BMP180_CalIndex;

typedef struct BMP180_EEPROM {
  short BMP180_AC1;
  short BMP180_AC2;
  short BMP180_AC3;
  unsigned short BMP180_AC4;
  unsigned short BMP180_AC5;
  unsigned short BMP180_AC6;
  short BMP180_B1;
  short BMP180_B2;
  short BMP180_MB;
  short BMP180_MC;
  short BMP180_MD;
} BMP180_EEPROM;

extern I2C_HandleTypeDef *_bmp180_ui2c;
extern BMP180_EEPROM _bmp180_eeprom;
extern BMP180_OSS _bmp180_oss;

extern const uint8_t BMP180_EEPROM_ADDR_MSB[11];
extern const uint8_t BMP180_EEPROM_ADDR_LSB[11];

extern const uint8_t BMP180_CMD_TEMP;
```

```
extern const uint8_t BMP180_DELAY_TEMP;
extern const uint8_t BMP180_CMD_PRES[4];
extern const uint8_t BMP180_DELAY_PRES[4];

void BMP180_Init(I2C_HandleTypeDef *hi2c);

void BMP180_SetOversampling(BMP180_OSS oss);

void BMP180_UpdateCalibrationData(void);

void BMP180_WriteReg(uint8_t reg, uint8_t cmd);
uint8_t BMP180_ReadReg8(uint8_t regm);
void BMP180_ReadReg(uint8_t regm, uint8_t * result);


int32_t BMP180_GetRawTemperature(void);
int32_t BMP180_GetPressure(void);

float BMP180_GetTemperature(void);

int32_t BMP180_GetUT(void);
int32_t BMP180_GetUP(void);

#endif
```

BMP180.c

```
#include "BMP180.h"

I2C_HandleTypeDef *_bmp180_ui2c;
BMP180_EEPROM _bmp180_eeprom;
BMP180_OSS _bmp180_oss;

const uint8_t BMP180_EEPROM_ADDR_MSB[11] = { 0xaa, 0xac, 0xae, 0xb0, 0xb2, 0xb4, 0xb6,
    0xb8, 0xba, 0xbc, 0xbe };
const uint8_t BMP180_EEPROM_ADDR_LSB[11] = { 0xab, 0xad, 0xaf, 0xb1, 0xb3, 0xb5, 0xb7,
    0xb9, 0xbb, 0xbd, 0xbf };

const uint8_t BMP180_CMD_TEMP = 0x2e;
const uint8_t BMP180_DELAY_TEMP = 5;
const uint8_t BMP180_CMD_PRES[4] = { 0x34, 0x74, 0xb4, 0xf4 };
const uint8_t BMP180_DELAY_PRES[4] = { 5, 8, 14, 26 };

/**
 * @brief Initializes the BMP180 temperature/pressure sensor.
 * @param hi2c User I2C handle pointer.
 */
void BMP180_Init(I2C_HandleTypeDef *hi2c) {
  _bmp180_ui2c = hi2c;
}

/**
 * @param oss Enum, oversampling setting.
 * @note Available resolutions: BMP180_LOW, BMP180_STANDARD, BMP180_HIGH, BMP180_ULTRA
    .
 * @note Refer to section 3.3.1 of datasheet.
 */
void BMP180_SetOversampling(BMP180_OSS oss) {
```

```
  _bmp180_oss = oss;
}

/**
 * @brief Updates calibration data.
 * @note Must be called once before main loop.
 */
void BMP180_UpdateCalibrationData(void) {
  uint8_t result[2];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_AC1], (uint8_t *)result);
  _bmp180_eeprom.BMP180_AC1 =(result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_AC2], (uint8_t *)result);
  _bmp180_eeprom.BMP180_AC2 =(result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_AC3], (uint8_t *)result);
  _bmp180_eeprom.BMP180_AC3 = (result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_AC4], (uint8_t *)result);
  _bmp180_eeprom.BMP180_AC4 = (result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_AC5], (uint8_t *)result);
  _bmp180_eeprom.BMP180_AC5 = (result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_AC6], (uint8_t *)result);
  _bmp180_eeprom.BMP180_AC6 = (result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_B1], (uint8_t *)result);
  _bmp180_eeprom.BMP180_B1 = (result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_B2], (uint8_t *)result);
  _bmp180_eeprom.BMP180_B2 = (result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_MB], (uint8_t *)result);
  _bmp180_eeprom.BMP180_MB = (result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_MC], (uint8_t *)result);
  _bmp180_eeprom.BMP180_MC = (result[0] << 8) | result[1];
  BMP180_ReadReg(BMP180_EEPROM_ADDR_MSB[BMP180_INDEX_MD], (uint8_t *)result);
  _bmp180_eeprom.BMP180_MD = (result[0] << 8) | result[1];
}

/**
 * @brief Writes to a specific register.
 * @param reg Address of register to write to.
 * @param cmd Byte to write.
 */
void BMP180_WriteReg(uint8_t reg, uint8_t cmd) {
  uint8_t arr[2] = { reg, cmd };
  HAL_I2C_Master_Transmit(_bmp180_ui2c, BMP180_I2C_ADDR << 1, arr, 2,
    BMP180_I2C_TIMEOUT);
}

/**
 * @brief Reads from a specific register.
 * @param reg Address of register to read from.
 * @return Byte read.
 */
uint8_t BMP180_ReadReg8(uint8_t reg) {
  HAL_I2C_Master_Transmit(_bmp180_ui2c, BMP180_I2C_ADDR << 1, &reg, 1,
    BMP180_I2C_TIMEOUT);
  uint8_t result;
  HAL_I2C_Master_Receive(_bmp180_ui2c, BMP180_I2C_ADDR << 1, &result, 1,
    BMP180_I2C_TIMEOUT);
  return result;
}

```

```
/**
 * @brief Reads from a specific register.
 * @param reg Address of register to read from.
 * @return Byte read.
 */
void BMP180_ReadReg(uint8_t reg, uint8_t *result) {
  HAL_I2C_Master_Transmit(_bmp180_ui2c, BMP180_I2C_ADDR << 1, &reg, 1,
    BMP180_I2C_TIMEOUT);
  HAL_I2C_Master_Receive(_bmp180_ui2c, BMP180_I2C_ADDR << 1, (uint8_t*)result, 2,
    BMP180_I2C_TIMEOUT);
}

/**
 * @brief Measures and calculates temperature.
 * @return Temperature in 0.1 (1/10) degrees Celsius.
 */
int32_t BMP180_GetRawTemperature(void) {
  BMP180_WriteReg(BMP180_CONTROL_REG, BMP180_CMD_TEMP);
  HAL_Delay(BMP180_DELAY_TEMP);
  uint8_t result[2];
  BMP180_ReadReg(BMP180_MSB_REG, (uint8_t *)result);
  int32_t ut = (result[0] << 8) | result[1];
  int32_t x1 = (ut - _bmp180_eeprom.BMP180_AC6) * _bmp180_eeprom.BMP180_AC5 / (1 <<
    15);
  int32_t x2 = (_bmp180_eeprom.BMP180_MC * (1 << 11)) / (x1 + _bmp180_eeprom.BMP180_MD
    );
  int32_t b5 = x1 + x2;
  return (b5 + 8) / (1 << 4);
}

/**
 * @brief Measures and calculates temperature.
 * @return Temperature in degrees Celsius.
 */
float BMP180_GetTemperature(void) {
  int32_t temp = BMP180_GetRawTemperature();
  return temp / 10.0;
}

/**
 * @brief Measures and calculates pressure.
 * @return Pressure in Pascal(Pa).
 */
int32_t BMP180_GetPressure(void) {
  BMP180_WriteReg(BMP180_CONTROL_REG, BMP180_CMD_TEMP);
  HAL_Delay(BMP180_DELAY_TEMP);
  int32_t ut = BMP180_GetUT();
  BMP180_WriteReg(BMP180_CONTROL_REG, BMP180_CMD_PRES[_bmp180_oss]);
  HAL_Delay(BMP180_DELAY_PRES[_bmp180_oss]);
  int32_t up = BMP180_GetUP();
  int32_t x1 = (ut - _bmp180_eeprom.BMP180_AC6) * _bmp180_eeprom.BMP180_AC5 / (1 <<
    15);
  int32_t x2 = (_bmp180_eeprom.BMP180_MC * (1 << 11)) / (x1 + _bmp180_eeprom.BMP180_MD
    );
  int32_t b5 = x1 + x2;
  int32_t b6 = b5 - 4000;
  x1 = (_bmp180_eeprom.BMP180_B2 * (b6 * b6 / (1 << 12))) / (1 << 11);
```

```
  x2 = _bmp180_eeprom.BMP180_AC2 * b6 / (1 << 11);
  int32_t x3 = x1 + x2;
  int32_t b3 = (((_bmp180_eeprom.BMP180_AC1 * 4 + x3) << _bmp180_oss) + 2) / 4;
  x1 = _bmp180_eeprom.BMP180_AC3 * b6 / (1 << 13);
  x2 = (_bmp180_eeprom.BMP180_B1 * (b6 * b6 / (1 << 12))) / (1 << 16);
  x3 = ((x1 + x2) + 2) / 4;
  uint32_t b4 = _bmp180_eeprom.BMP180_AC4 * (uint32_t) (x3 + 32768) / (1 << 15);
  uint32_t b7 = ((uint32_t) up - b3) * (50000 >> _bmp180_oss);
  int32_t p;
  if (b7 < 0x80000000)
    p = (b7 * 2) / b4;
  else
    p = (b7 / b4) * 2;
  x1 = (p / (1 << 8)) * (p / (1 << 8));
  x1 = (x1 * 3038) / (1 << 16);
  x2 = (-7357 * p) / (1 << 16);
  p = p + (x1 + x2 + 3791) / (1 << 4);
  return p;
}

int32_t BMP180_GetUT(void){
  uint8_t result[2];
  BMP180_ReadReg(BMP180_MSB_REG, (uint8_t *)result);
  return (result[0] << 8) | result[1];
}

int32_t BMP180_GetUP(void){
  uint8_t result1[2];
  BMP180_ReadReg(BMP180_MSB_REG, (uint8_t *)result1);
  return ((result1[0] << 16) | (result1[1] << 8) | BMP180_ReadReg8(BMP180_XLSB_REG))
    >> (8 - _bmp180_oss);
}
```

dwt_delay.h

```
/*
 * Simple microseconds delay routine, utilizing ARM's DWT
 * (Data Watchpoint and Trace Unit) and HAL library.
 * Intended to use with gcc compiler, but I hope it can be used
 * with any other C compiler across the Universe (provided that
 * ARM and CMSIS already invented) :)
 * Max K
 *
 *
 * This file is part of DWT_Delay package.
 * DWT_Delay is free software: you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License,
 * or (at your option) any later version.
 *
 * us_delay is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty
 * of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
 * the GNU General Public License for more details.
 * http://www.gnu.org/licenses/.
 */

#include <stdint.h>
```

```
#ifndef INC_DWT_DELAY_H_
#define INC_DWT_DELAY_H_

#define DWT_DELAY_NEWBIE 0

void DWT_Init(void);
void DWT_Delay(uint32_t us);

#endif /* INC_DWT_DELAY_DWT_DELAY_H_ */
```

dwt_delay.c

```
/*
 * Simple microseconds delay routine, utilizing ARM's DWT
 * (Data Watchpoint and Trace Unit) and HAL library.
 * Intended to use with gcc compiler, but I hope it can be used
 * with any other C compiler across the Universe (provided that
 * ARM and CMSIS already invented) :)
 * Max K
 *
 *
 * This file is part of DWT_Delay package.
 * DWT_Delay is free software: you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License,
 * or (at your option) any later version.
 *
 * us_delay is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty
 * of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
 * the GNU General Public License for more details.
 * http://www.gnu.org/licenses/.
 */

#include "stm32f1xx_hal.h"          // change to whatever MCU you use
#include "dwt_delay.h"

/**
 * Initialization routine.
 * You might need to enable access to DWT registers on Cortex-M7
 *   DWT->LAR = 0xC5ACCE55
 */
void DWT_Init(void)
{
    if (!(CoreDebug->DEMCR & CoreDebug_DEMCR_TRCENA_Msk)) {
        CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk;
        DWT->CYCCNT = 0;
        DWT->CTRL |= DWT_CTRL_CYCCNTENA_Msk;
    }
}

#if DWT_DELAY_NEWBIE
/**
 * If you are a newbie and see magic in DWT_Delay, consider this more
 * illustrative function, where you explicitly determine a counter
 * value when delay should stop while keeping things in bounds of uint32.
*/
```

```c
void DWT_Delay(uint32_t us) // microseconds
{
    uint32_t startTick  = DWT->CYCCNT,
             targetTick = DWT->CYCCNT + us * (SystemCoreClock/1000000);

    // Must check if target tick is out of bounds and overflowed
    if (targetTick > startTick) {
        // Not overflowed
        while (DWT->CYCCNT < targetTick);
    } else {
        // Overflowed
        while (DWT->CYCCNT > startTick || DWT->CYCCNT < targetTick);
    }
}
#else
/**
 * Delay routine itself.
 * Time is in microseconds (1/1000000th of a second), not to be
 * confused with millisecond (1/1000th).
 *
 * No need to check an overflow. Let it just tick :)
 *
 * @param uint32_t us  Number of microseconds to delay for
 */
void DWT_Delay(uint32_t us) // microseconds
{
    uint32_t startTick = DWT->CYCCNT,
             delayTicks = us * (SystemCoreClock/1000000);

    while (DWT->CYCCNT - startTick < delayTicks);
}

#endif
```

HCSR05.h

```c
/*
 * HCSR05.h
 *
 *  Created on: Aug 11, 2020
 *      Author: jmdmahdi
 */

#ifndef HCSR05_H
#define HCSR05_H

#include "stm32f1xx_hal.h"
#include "math.h"
#include "dwt_delay.h"

#ifndef PI
#define PI 3.14159265358979323846264338327950288
#endif

void HCSR05_Init(void);
void HCSR05_Calculate_SoundSpeed(int8_t T, uint32_t P, int8_t H);
void HCSR05_Calculate_TOF(void);
void HCSR05_TIM_Callback(TIM_HandleTypeDef *htim);
```

```
void HCSR05_Ready(GPIO_TypeDef *trig_port, uint16_t trig_pin, TIM_HandleTypeDef* htim,
    uint32_t timer_channel, uint8_t distance, int8_t angle_difference);
float_t HCSR05_Get_WindSpeed(int8_t T, uint32_t P, int8_t H);
float_t HCSR05_Calculate_Angle(double_t x, double_t y);
void HCSR05_Calculate_WindSpeedNdAngle(float_t X_axis, float_t Y_axis, int16_t x,
    int16_t y, float_t *result);


GPIO_TypeDef *HCSR05_Trigger_Port;
uint16_t HCSR05_Trigger_Pin;
TIM_HandleTypeDef* HCSR05_Htim;
uint32_t HCSR05_Channel;
uint8_t HCSR05_Captured;
uint32_t HCSR05_IC_First_Val;
uint32_t HCSR05_IC_Second_Val;
uint32_t HCSR05_TOF;
uint8_t HCSR05_Is_First_Val_Captured;
uint8_t HCSR05_Distance;
int8_t HCSR05_Angle_Difference;
float_t HCSR05_SoundSpeed;

#endif /* HCSR05_H */
```

HCSR05.c

```
/*
 * HCSR05.c
 *
 *  Created on: Aug 11, 2020
 *      Author: jmdmahdi
 */
#include "HCSR05.h"

/**
 * @brief Initialization sensor requirments.
 */
void HCSR05_Init(void)
{
  // Init delay function
  DWT_Init();
}

/**
 * @brief Calculate sound speed based on Owen Cramer's method.
 * @param T Temperature (Celsius).
 * @param P Air pressure (Pa).
 * @param H Relative humidity (%).
 */
void HCSR05_Calculate_SoundSpeed(int8_t T, uint32_t P, int8_t H){
  double_t Xc, Xw, V1, V2, V3, f, PSV, PSV1, PSV2, T_kel, T_sqr;

  // Variables description is in the report
  Xc = 400 * 1e-6;
  T_kel = 273.15 + T;
  T_sqr = pow(T, 2);
  f = 1.00062 + 3.141593 * 1e-8 * P + T_sqr * 5.6 * 1e-7;
  PSV1 = pow(T_kel, 2) * 1.2378847 * 1e-8 - 1.9121316 * 1e-2 * T_kel;
  PSV2 = 33.93711047 - 6.3431645 * 1e3 / T_kel;
```

```
  PSV = exp(PSV1 + PSV2);
  Xw = (H / 100) * f * PSV / P;
  V1 = 0.603055 * T + 331.5024 - T_sqr * 5.28 * 1e-4 + (0.1495874 * T + 51.471935 -
    T_sqr * 7.82 * 1e-4) * Xw;
  V2 = (-1.82 * 1e-7 + 3.73 * 1e-8 * T - T_sqr * 2.93 * 1e-10) * P + (-85.20931 -
    0.228525 * T + T_sqr * 5.91 * 1e-5) * Xc;
  V3 = pow(Xw, 2) * 2.835149 - pow(P, 2) * 2.15 * 1e-13 + pow(Xc, 2) * 29.179762 +
    4.86 * 1e-4 * Xw * P * Xc;

  HCSR05_SoundSpeed =  V1 + V2 - V3;
}

/**
 * @brief Calculate sound time of flight.
 */
void HCSR05_Calculate_TOF(void)
{
  // Send 10 microsecond pulse on trigger pin
  HAL_GPIO_WritePin(HCSR05_Trigger_Port, HCSR05_Trigger_Pin, GPIO_PIN_SET);
  DWT_Delay(10);
  HAL_GPIO_WritePin(HCSR05_Trigger_Port, HCSR05_Trigger_Pin, GPIO_PIN_RESET);

  // Start timer
  HAL_TIM_IC_Start_IT(HCSR05_Htim, HCSR05_Channel);

  // Wait for Echo pulse
  uint32_t startTick = HAL_GetTick();
  do{
    if(HCSR05_Captured) break;
  }while((HAL_GetTick() - startTick) < 500);

  // Reset state
  HCSR05_Captured = 0;
  HAL_TIM_IC_Stop_IT(HCSR05_Htim, HCSR05_Channel);
}
/**
 * @brief must put in HAL_TIM_IC_CaptureCallback.
 */
void HCSR05_TIM_Callback(TIM_HandleTypeDef *htim)
{
  // If the interrupt source is on our channel
  if (htim->Channel == HCSR05_Channel) {
    // If the first val is not captured
    if (HCSR05_Is_First_Val_Captured==0) {
      // Get first value
      HCSR05_IC_First_Val = HAL_TIM_ReadCapturedValue(htim, HCSR05_Channel);
      // Set first val captured as true
      HCSR05_Is_First_Val_Captured = 1;
      // change the polarity to falling edge
      __HAL_TIM_SET_CAPTUREPOLARITY(htim, HCSR05_Channel,
    TIM_INPUTCHANNELPOLARITY_FALLING);
    } else if (HCSR05_Is_First_Val_Captured==1) { // If first val is already captured
      // Get second value
      HCSR05_IC_Second_Val = HAL_TIM_ReadCapturedValue(htim, HCSR05_Channel);

      // Calculate HCSR05_TOF
      if (HCSR05_IC_Second_Val > HCSR05_IC_First_Val) {
        HCSR05_TOF = HCSR05_IC_Second_Val-HCSR05_IC_First_Val;
```

```
        } else if (HCSR05_IC_First_Val > HCSR05_IC_Second_Val) {
          HCSR05_TOF = (0xffff - HCSR05_IC_First_Val) + HCSR05_IC_Second_Val;
        }

        // Capturing finished
        HCSR05_Captured = 1;

        // Reset HCSR05_Is_First_Val_Captured for next use
        HCSR05_Is_First_Val_Captured = 0;
        // Reset the counter
        __HAL_TIM_SET_COUNTER(htim, 0);
        // Set polarity to rising edge for next use
        __HAL_TIM_SET_CAPTUREPOLARITY(htim, HCSR05_Channel,
    TIM_INPUTCHANNELPOLARITY_RISING);
      }
    }
}

/**
 * @brief Get things ready.
 * @param trig_port HCSR05 Trigger pin port.
 * @param trig_pin HCSR05 Trigger pin.
 * @param htim HCSR05 timer.
 * @param timer_channel HCSR05 timer channel.
 * @param distance Distance between transducers (cm).
 */
void HCSR05_Ready(GPIO_TypeDef *trig_port, uint16_t trig_pin, TIM_HandleTypeDef* htim,
    uint32_t timer_channel, uint8_t distance, int8_t angle_difference){
  // Set trigger port and pin global variables
  HCSR05_Trigger_Port = trig_port;
  HCSR05_Trigger_Pin = trig_pin;
  // Set timer global variables
  HCSR05_Htim = htim;
  HCSR05_Channel = timer_channel;
  // Set distance global variable
  HCSR05_Distance = distance;
  // Set angle difference global variable
  HCSR05_Angle_Difference = angle_difference;
  // Reset measurement Variables
  HCSR05_TOF = 0;
  HCSR05_Captured = 0;
  HCSR05_IC_First_Val = 0;
  HCSR05_IC_Second_Val = 0;
  HCSR05_TOF = 0;
  HCSR05_Is_First_Val_Captured = 0;
  HCSR05_SoundSpeed = 0;
}

/**
 * @brief Get wind speed on one axis.
 * @param T Temperature (Celsius).
 * @param P Air pressure (Pa).
 * @param H Relative humidity (%).
 */
float_t HCSR05_Get_WindSpeed(int8_t T, uint32_t P, int8_t H){
  // Calculate sound speed
  HCSR05_Calculate_SoundSpeed(T, P, H);
  // Calculate time of flight
```

```
  HCSR05_Calculate_TOF();

  // If Calculated values true
  if(HCSR05_TOF > 0 && HCSR05_SoundSpeed > 0){
    // Calculate and Return Wind Speed on one axis
    return (HCSR05_Distance * 1e4 / HCSR05_TOF) - HCSR05_SoundSpeed;
  }
  return 0;
}

/**
 * @brief Calculate angle from x and y.
 * @return Angle in degree.
 */
float_t HCSR05_Calculate_Angle(double_t x, double_t y){
  // Calculate angle and convert to degree
  return atan2(y, x) * 180.0 / PI;
}

/**
 * @brief Calculate speed R and theta.
 * @param X_axis X axis speed.
 * @param Y_axis Y axis speed.
 * @param compass_x compass measured x.
 * @param compass_y compass measured y.
 * @param result will fill with calculated speed and angle
 */
void HCSR05_Calculate_WindSpeedNdAngle(float_t X_axis, float_t Y_axis, int16_t
    compass_x, int16_t compass_y, float_t *result){
  // Calculate angle
  float_t compose_angle = HCSR05_Calculate_Angle(compass_x, compass_y);
  float_t wind_angle = HCSR05_Calculate_Angle(X_axis, Y_axis);
  float_t angle = compose_angle + wind_angle + HCSR05_Angle_Difference;
  if(angle<=0) angle += 360;
  // Fill result with R(speed) and theta(angle)
  *result = hypot(X_axis, Y_axis);
  *(result+1) = angle;
}
```

MAX44009.h

```
#ifndef MAX44009_h
#define MAX44009_h
#include "stm32f1xx_hal.h"

#define MAX44009_ADDR 0x4A<<1 // or 0x4B if A0 pin connected to Vcc

HAL_StatusTypeDef MAX44009_Begin();

float MAX44009_Get_Lux(void);

#endif
```

MAX44009.c

```
#include "MAX44009.h"

I2C_HandleTypeDef *MAX44009_hi2c;
```

```
HAL_StatusTypeDef MAX44009_Begin(I2C_HandleTypeDef *hi2c) {
  MAX44009_hi2c = hi2c;
  return HAL_I2C_Mem_Write(MAX44009_hi2c, MAX44009_ADDR, 0x02, 1, 0x00, 1,
      100);
}

float MAX44009_Get_Lux(void) {
  uint8_t data[2];

  if (HAL_I2C_Mem_Read(MAX44009_hi2c, MAX44009_ADDR, 0x03, 1, data, 1, 100)
      == HAL_OK) {
    if (HAL_I2C_Mem_Read(MAX44009_hi2c, MAX44009_ADDR, 0x04, 1, data + 1, 1,
        100) == HAL_OK) {
      // Convert the data to lux
      uint8_t exponent = data[0] >> 4;
      uint32_t mantisa = ((data[0] & 0x0F) << 4) + (data[1] & 0x0F);
      mantisa <<= exponent;
      return ((float) (mantisa) * 0.045);
    }
  }
  return 0;
}
```

QMC5883L.h

```
#ifndef QMC5883L_H
#define QMC5883L_H

#include "stm32f1xx_hal.h"
#include "math.h"

void QMC5883L_Init();
void QMC5883L_Sleep();
void QMC5883L_Reset();
int QMC5883L_Ready();
void QMC5883L_Reconfig();

int QMC5883L_Read_Heading();
int QMC5883L_ReadRaw(int16_t *x, int16_t *y, int16_t *z);

void QMC5883L_Reset_Calibration();

void QMC5883L_Set_Sampling_Rate(int rate);
void QMC5883L_Set_Range(int range);
void QMC5883L_Set_Oversampling(int ovl);

int16_t xhigh, xlow;
int16_t yhigh, ylow;
uint8_t addr;
uint8_t mode;
uint8_t rate;
uint8_t range;
uint8_t oversampling;

#endif
```

QMC5883L.c

```c
#include "QMC5883L.h"

/* The default I2C address of this chip */
#define QMC5883L_ADDR 0x0D<<1

/* Register numbers */
#define QMC5883L_X_LSB 0
#define QMC5883L_X_MSB 1
#define QMC5883L_Y_LSB 2
#define QMC5883L_Y_MSB 3
#define QMC5883L_Z_LSB 4
#define QMC5883L_Z_MSB 5
#define QMC5883L_STATUS 6
#define QMC5883L_TEMP_LSB 7
#define QMC5883L_TEMP_MSB 8
#define QMC5883L_CONFIG 9
#define QMC5883L_CONFIG2 10
#define QMC5883L_RESET 11
#define QMC5883L_RESERVED 12
#define QMC5883L_CHIP_ID 13

/* Bit values for the STATUS register */
#define QMC5883L_STATUS_DRDY 1
#define QMC5883L_STATUS_OVL 2
#define QMC5883L_STATUS_DOR 4

/* Oversampling values for the CONFIG register */
#define QMC5883L_CONFIG_OS512 0b00000000
#define QMC5883L_CONFIG_OS256 0b01000000
#define QMC5883L_CONFIG_OS128 0b10000000
#define QMC5883L_CONFIG_OS64  0b11000000

/* Range values for the CONFIG register */
#define QMC5883L_CONFIG_2GAUSS 0b00000000
#define QMC5883L_CONFIG_8GAUSS 0b00010000

/* Rate values for the CONFIG register */
#define QMC5883L_CONFIG_10HZ   0b00000000
#define QMC5883L_CONFIG_50HZ   0b00000100
#define QMC5883L_CONFIG_100HZ  0b00001000
#define QMC5883L_CONFIG_200HZ  0b00001100

/* Mode values for the CONFIG register */
#define QMC5883L_CONFIG_STANDBY 0b00000000
#define QMC5883L_CONFIG_CONT    0b00000001

/* Apparently PI isn't available in all environments. */
#ifndef PI
#define PI 3.14159265358979323846264338327950288
#endif

I2C_HandleTypeDef *QMC5883L_i2c;

void QMC5883L_Write_Register(uint16_t reg, int value) {
  HAL_I2C_Mem_Write(QMC5883L_i2c, QMC5883L_ADDR, reg, 1, (uint8_t*) value, 1,
      100);
```

```
}

void QMC5883L_Read_Register(uint8_t *data, uint16_t reg, uint16_t count) {
  HAL_I2C_Mem_Read(QMC5883L_i2c, QMC5883L_ADDR, reg, 1, data, count, 100);
}

void QMC5883L_Reconfig() {
  QMC5883L_Write_Register(QMC5883L_CONFIG,
      oversampling | range | rate | mode);
}

void QMC5883L_Reset() {
  QMC5883L_Write_Register(QMC5883L_RESET, 0x01);
  QMC5883L_Reconfig();
}

void QMC5883L_Set_Oversampling(int x) {
  switch (x) {
  case 512:
    oversampling = QMC5883L_CONFIG_OS512;
    break;
  case 256:
    oversampling = QMC5883L_CONFIG_OS256;
    break;
  case 128:
    oversampling = QMC5883L_CONFIG_OS128;
    break;
  case 64:
    oversampling = QMC5883L_CONFIG_OS64;
    break;
  }
  QMC5883L_Reconfig();
}

void QMC5883L_Set_Range(int x) {
  switch (x) {
  case 2:
    range = QMC5883L_CONFIG_2GAUSS;
    break;
  case 8:
    range = QMC5883L_CONFIG_8GAUSS;
    break;
  }
  QMC5883L_Reconfig();
}

void QMC5883L_Set_Sampling_Rate(int x) {
  switch (x) {
  case 10:
    rate = QMC5883L_CONFIG_10HZ;
    break;
  case 50:
    rate = QMC5883L_CONFIG_50HZ;
    break;
  case 100:
    rate = QMC5883L_CONFIG_100HZ;
    break;
  case 200:
```

```
    rate = QMC5883L_CONFIG_200HZ;
    break;
  }
  QMC5883L_Reconfig();
}

void QMC5883L_Init(I2C_HandleTypeDef *i2c) {
  QMC5883L_i2c = i2c;
  /* This assumes the wire library has been initialized. */
  addr = QMC5883L_ADDR;
  oversampling = QMC5883L_CONFIG_OS512;
  range = QMC5883L_CONFIG_2GAUSS;
  rate = QMC5883L_CONFIG_50HZ;
  mode = QMC5883L_CONFIG_STANDBY;
  QMC5883L_Reset();
}

void QMC5883L_Sleep() {
  mode = QMC5883L_CONFIG_STANDBY;
  QMC5883L_Reset();
}

int QMC5883L_Ready() {
  uint8_t data;
  QMC5883L_Read_Register(&data, QMC5883L_STATUS, 1);
  if (!data)
    return 0;
  return data & QMC5883L_STATUS_DRDY;
}

int QMC5883L_ReadRaw(int16_t *x, int16_t *y, int16_t *z) {
  mode = QMC5883L_CONFIG_CONT;
  QMC5883L_Reset();
  while (!QMC5883L_Ready()) {
  }

  uint8_t data[6];

  QMC5883L_Read_Register(data, QMC5883L_X_LSB, 6);

  *x = data[0] | (data[1] << 8);
  *y = data[2] | (data[3] << 8);
  *z = data[4] | (data[5] << 8);

  return 1;
}

void QMC5883L_Reset_Calibration() {
  xhigh = yhigh = 0;
  xlow = ylow = 0;
}

int QMC5883L_Read_Heading() {
  int16_t x, y, z;

  if (!QMC5883L_ReadRaw(&x, &y, &z))
    return 0;

```

```
  /* Update the observed boundaries of the measurements */

  if (x < xlow)
    xlow = x;
  if (x > xhigh)
    xhigh = x;
  if (y < ylow)
    ylow = y;
  if (y > yhigh)
    yhigh = y;

  /* Bail out if not enough data is available. */

  if (xlow == xhigh || ylow == yhigh)
    return 0;

  /* Recenter the measurement by subtracting the average */

  x -= (xhigh + xlow) / 2;
  y -= (yhigh + ylow) / 2;

  /* Rescale the measurement to the range observed. */

  float fx = (float) x / (xhigh - xlow);
  float fy = (float) y / (yhigh - ylow);

  int heading = 180.0 * atan2(fy, fx) / M_PI;
  if (heading <= 0)
    heading += 360;

  return heading;
}
```

سورس کد برنامه دسکتاپ

main.py

```
import sys, signal, platform, time, usb, faulthandler
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5.QtChart import *
from PyQt5.QtWidgets import *
from UI.mainWindow import Ui_MainWindow
from UI.compassWidget import compassWidget
from UI.chartWidget import chartWidget
from DB.db import sqlite3DB
from datetime import datetime, timedelta
from Worker.worker import Worker
from Worker.signalWakeupHandler import SignalWakeupHandler
import usb.backend.libusb1
import numpy as np

backend = usb.backend.libusb1.get_backend(find_library=lambda x: "libusb-1.0.dll")

# Properly show Qt faults
faulthandler.enable()

```

```
# Define global variables
_CloseApp = False


class MainWindow(QMainWindow):

    def __init__(self, *args, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)
        # Define DB instance
        self.DB = sqlite3DB("DB/weatherStation.db")
        # Define properties
        self.is_connected = False
        self.is_configured = False
        self.dev = None
        self.status = "checking"
        self.USBString = ""

        # Define main ui
        self.window = Ui_MainWindow()
        self.window.setupUi(self)

        # Set log textBrowser as read only
        self.window.textBrowser.setReadOnly(True)
        self.window.textBrowser.setCursorWidth(0)
        # Generate compass widget
        self.compass = compassWidget()
        self.window.compassLayout.addWidget(self.compass)

        # Restore last time window geometry
        self.settings = QSettings('JMDMahdi', 'WeatherStation')
        geometry = self.settings.value('geometry', '')
        if isinstance(geometry, QByteArray):
            self.restoreGeometry(geometry)

        # Update the home tab with last received data
        self.updateHomeTab()

        # Define chart tab charts and add them to view
        self.generateCharts()

  # Set default statusbar text
        self.window.statusbar.showMessage('Checking ...')

        # Build background thread for USB
        self.threadpool = QThreadPool()

        # Pass the function to execute in background
        self.worker = Worker(self.USB_process)
        self.worker.signals.progress.connect(self.processData)
        self.worker.signals.statusbar.connect(self.updateStatusBar)

        # Execute
        self.threadpool.start(self.worker)

    def processData(self, data):
        # Remove completed data indicators from string and pass into insertData
    function, to insert to db
        self.insertData(data)
```

```
    def USB_process(self, process_callback, UIUpdateCallBack):
        global _CloseApp
        # Check device status continuously
        while True:
            if _CloseApp:  # End thread on close signal
                # Close UI
                QApplication.quit()
                return
            if self.is_connected:
                # Wait for data from device
                try:
                    # Get USB data and pass to main thread to process
                    data = self.dev.read(0x81, 32, 2000)
                    # Convert data to uint8 array
                    dataBytes = np.array(data.tolist(), dtype=np.uint8)
                    # Convert data to float list
                    process_callback.emit(dataBytes.view(dtype=np.float32).tolist())
                except Exception as e:
                    if not ("Operation timed out" in str(e) or "Pipe error" in str(e) or "Input/Output Error" in str(e) or "No such device" in str(e)):
                        print("self.dev.read error: " + str(e))
                    # Recheck device status
                    self.check_if_device_connected(UIUpdateCallBack, True)
                    time.sleep(1)
                    pass
            else:
                # Recheck device status every 1 secound if device not connected
                time.sleep(1)
                UIUpdateCallBack.emit(self.is_connected)
                self.check_if_device_connected(UIUpdateCallBack, True)

    def device_connected(self):
        # Set device status on UI as connected
        if self.status != "connected":
            self.writeLog("Device connected\n")
            self.window.statusbar.showMessage('Connected')
            self.window.statusbar.setStyleSheet('color: green')
            self.status = "connected"

    def device_disconnected(self):
        # Set device status on UI as disconnected
        if self.status != "disconnected":
            self.writeLog("Device disconnected\n")
            self.window.statusbar.showMessage('Disconnected')
            self.window.statusbar.setStyleSheet('color: red')
            self.status = "disconnected"

    def updateStatusBar(self, status):
        if status:
            self.device_connected()
        else:
            self.device_disconnected()

    def find_device(self):
        # Search connected device for our device and return device descriptor
        dev_g = usb.core.find(backend=backend, idVendor=5511, idProduct=63322, find_all=True)
```

```
        dev_list = list(dev_g)
        if dev_list is None:
            return None
        try:
            dev = dev_list[0]
        except:
            try:
                dev = dev_list.next()
            except:
                return None
        return dev

    def check_if_device_connected(self, UIUpdateCallBack, force=False):
        # Search for our device
        self.dev = self.find_device()
        if self.dev is None:
            self.is_connected = False
            self.is_configured = False
        else:
            self.is_connected = True
            # Gain access to read the device if not done already
            if force or not self.is_configured:
                # This must be done on every connection
                self.config_device()
        # Set status on UI
        UIUpdateCallBack.emit(self.is_connected)

    def config_device(self):
        # Gain access to device
        try:
            self.dev.reset()
        except:
            self.is_configured = False
            return
            pass
        try:
            self.dev.set_configuration()
        except usb.core.USBError:
            pass
        cfg = self.dev.get_active_configuration()
        intf = usb.util.find_descriptor(cfg, bInterfaceNumber=1)
        if not intf:
            raise("Interface not found")
        if platform.system() != "Windows":
            if self.dev.is_kernel_driver_active(intf.bInterfaceNumber) is True:
                self.dev.detach_kernel_driver(intf.bInterfaceNumber)
        self.is_configured = True

    def closeEvent(self, event):
        global _CloseApp
        # Confirm close
        reply = QMessageBox.question(self, 'Window Close', 'Are you sure you want to close the window?',
                                     QMessageBox.Yes | QMessageBox.No, QMessageBox.No)
        if reply == QMessageBox.Yes:
            # Save window geometry to restore next time
            self.settings.setValue('geometry', self.saveGeometry())
            # Tell worker to stop processing
```

```
            _CloseApp = True
            # Wait for the worker to end
            self.threadpool.waitForDone()
            # Close app
            event.accept()
        else:
            event.ignore()

    def updateHomeTab(self):
        '''Update the home tab with last received data'''
        # Get last data
        lastData = self.DB.getLastRow()
        # Fill lastData with 0 if data not available
        if lastData is None:
            lastData = (0, 0, 0, 0, 0, 0, 0)

        # Convert timestamp to datetime
        date = datetime.fromtimestamp((lastData[6]))

        # Update home tab with last data
        self.window.temperatureText.setText(str(lastData[0]) + " °C")
        self.window.pressureText.setText(str(lastData[1] / 100) + " hPa")
        self.window.lightIntensityText.setText(str(lastData[2]) + " lux")
        self.window.humidityText.setText(str(lastData[3]) + " %")
        self.window.windSpeedText.setText(str(lastData[4]) + " m/s")
        self.compass.setAngle(int(lastData[5]))
        self.window.windAngleText.setText(str(lastData[5]) + " °")
        if platform.system() != "Windows":
            self.window.dateText.setText(date.strftime("%A, %-d %B %Y"))
        else:
            self.window.dateText.setText(date.strftime("%A, %e %B %Y"))
        self.window.timeText.setText(date.strftime("%I:%M %p"))

    def insertData(self, data):
        self.writeLog("Received data: " + " ,".join(map(str, data)) + " | Status: ")
        if len(data) == 8:
            self.DB.insert(data)
            self.writeLog("Successfully inserted to db\n", False)
            self.updateHomeTab()
        else:
            self.writeLog("Invalid data\n", False)

    def writeLog(self, text, insertTime=True):
        self.window.textBrowser.moveCursor(QTextCursor.End)
        if insertTime:
            self.window.textBrowser.insertPlainText(datetime.now().strftime("%a %b %d %H:%M:%S.%f %Y") + "> ")
        self.window.textBrowser.insertPlainText(text)
        if self.window.checkBox.isChecked():
            self.window.textBrowser.verticalScrollBar().setValue(self.window.textBrowser.verticalScrollBar().maximum())

    def generateCharts(self):
        # Add chart widgets
        self.temperatureChart = chartWidget(None, "Temperature records in celcius", "Time", "Temperature")
        self.window.chartTabContents.addWidget(self.temperatureChart)
```

```
        self.pressureChart = chartWidget(None, "Pressure records in Pa", "Time", "Pressure")
        self.window.chartTabContents.addWidget(self.pressureChart)
        self.LightIntensityChart = chartWidget(None, "Light intensity records in lux", "Time", "Light intensity")
        self.window.chartTabContents.addWidget(self.LightIntensityChart)
        self.humidityChart = chartWidget(None, "Humidity records in percent", "Time", "Humidity")
        self.window.chartTabContents.addWidget(self.humidityChart)
        self.windSpeedChart = chartWidget(None, "Wind speed records in m/s", "Time", "Wind speed")
        self.window.chartTabContents.addWidget(self.windSpeedChart)
        # Update dateEdits date
        today = datetime.now()
        week_ago = today - timedelta(days=7)
        self.window.firstDate.setDate(week_ago.date())
        self.window.secondDate.setDate(today.date())
        # Call updateCharts on date changes
        self.window.firstDate.dateChanged.connect(self.updateCharts)
        self.window.secondDate.dateChanged.connect(self.updateCharts)
        # Update charts series from db
        self.updateCharts()

    def updateCharts(self):
        # Get dates
        secondDate = self.window.secondDate.date().toPyDate()
        firstDate = self.window.firstDate.date().toPyDate()
        # Generate start and end date based on dates
        if firstDate < secondDate:
            startDate = datetime.combine(firstDate, datetime.min.time())
            endDate = datetime.combine(secondDate, datetime.max.time())
        else:
            endDate = datetime.combine(firstDate, datetime.max.time())
            startDate = datetime.combine(secondDate, datetime.min.time())
        # Get rows between start and end date
        rows = self.DB.getDataBetween(startDate.timestamp(), endDate.timestamp())
        # define QLineSeries instants
        temperatureSeries = QLineSeries()
        pressureSeries = QLineSeries()
        LightIntensitySeries = QLineSeries()
        humiditySeries = QLineSeries()
        windSpeedSeries = QLineSeries()
        # Fill series with db data
        for row in rows:
            temperatureSeries.append(int(row[6] * 1000), row[0])
            pressureSeries.append(int(row[6] * 1000), row[1])
            LightIntensitySeries.append(int(row[6] * 1000), row[2])
            humiditySeries.append(int(row[6] * 1000), row[3])
            windSpeedSeries.append(int(row[6] * 1000), row[4])
        # Update series with new ones
        self.temperatureChart.setSeries(temperatureSeries)
        self.pressureChart.setSeries(pressureSeries)
        self.LightIntensityChart.setSeries(LightIntensitySeries)
        self.humidityChart.setSeries(humiditySeries)
        self.windSpeedChart.setSeries(windSpeedSeries)


def closeApp(sig, frame=None):
```

```
    global _CloseApp
    # Ignore additional signals
    signal.signal(sig, signal.SIG_IGN)
    # Tell background process to stop
    _CloseApp = True
    print("\rClosing app ...")


if __name__ == "__main__":
    # Force Close with ctrl+c
    signal.signal(signal.SIGINT, closeApp)
    # Force Close on terminal close
    signal.signal(signal.SIGTERM, closeApp)
    # Start UI
    app = QApplication(["Weather Station"])
    SignalWakeupHandler(app)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

db.py

```
import sqlite3
import time


class sqlite3DB(object):
    """sqlite3 database class"""

    def __init__(self, db_location):
        """Initialize db class variables"""
        self.connection = sqlite3.connect(db_location)
        self.cursor = self.connection.cursor()
        self.create_table()

    def create_table(self):
        """Create a database table if it does not exist already"""
        self.cursor.execute('''CREATE TABLE IF NOT EXISTS "weather_records" (
  "id"  INTEGER PRIMARY KEY AUTOINCREMENT,
  "sensor_id" INTEGER NOT NULL DEFAULT 0,
  "station_id"  INTEGER NOT NULL DEFAULT 0,
  "temperature" REAL NOT NULL DEFAULT 0,
  "pressure"  REAL NOT NULL DEFAULT 0,
  "light_intensity" REAL NOT NULL DEFAULT 0,
  "humidity"  REAL NOT NULL DEFAULT 0,
  "wind_speed"  REAL NOT NULL DEFAULT 0,
  "wind_direction"  REAL NOT NULL DEFAULT 0,
  "timestamp" INTEGER NOT NULL
  );''')

    def commit(self):
        """Commit changes to database"""
        self.connection.commit()

    def getLastRow(self):
        """Returns last weather record"""
        self.cursor.execute(
```

```
            'SELECT "temperature", "pressure", "light_intensity", "humidity", "wind_speed", "wind_direction", "timestamp" FROM "weather_records" ORDER BY id DESC LIMIT 1')
        return self.cursor.fetchone()

    def insert(self, data):
        """Inserts weather station records"""
        # Make a list from data
        data = list(data)
        if len(data) == 8:
            # Add timestamp field
            data.append(time.time())
        # Insert to db
        self.cursor.execute(
            'INSERT INTO "weather_records" ("sensor_id", "station_id", "temperature", "pressure", "light_intensity", "humidity", "wind_speed", "wind_direction", "timestamp") VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?)',
            data)
        # Commit changes
        self.commit()
        # Return inserted row id
        return self.cursor.lastrowid

    def getDataBetween(self, start, end):
        """Return data in range of selected dates"""
        self.cursor.execute(
            'SELECT "temperature", "pressure", "light_intensity", "humidity", "wind_speed", "wind_direction", "timestamp" FROM "weather_records" WHERE "timestamp" > ? AND "timestamp" < ? ORDER BY id ASC',
            [start, end])
        return self.cursor.fetchall()

    def close(self):
        """Close sqlite3 connection"""
        # Commit changes before closing connection
        self.commit()
        # Close connection
        self.connection.close()

    def __del__(self):
        """Class destructor"""
        # Close db connection
        self.close()
```

chartWidget.py

```
from PyQt5.QtChart import *
from PyQt5.QtGui import *
from PyQt5.QtCore import *


class chartWidget(QChartView):
    seriesChanged = pyqtSignal(QLineSeries)

    def __init__(self, parent=None, title=None, XTitle=None, yTitle=None):
        QChartView.__init__(self, parent)

        self.XTitle = XTitle
```

```
        self.YTitle = yTitle

        self.old_series = None
        self.axisX = None
        self.axisY = None
        self._series = QLineSeries()

        self._chart = QChart()

        self._chart.setAnimationOptions(QChart.SeriesAnimations)
        self._chart.legend().setVisible(False)

        self._chart.addSeries(self._series)

        self.updateSeries()

        self._chart.setTitle(title)

        self.setChart(self._chart)
        self.setRenderHint(QPainter.Antialiasing)
        self.setMinimumSize(300, 300)

    def update(self, *__args):
        if self.old_series is not None:
            self._chart.removeSeries(self.old_series)
        self._chart.addSeries(self._series)
        self.updateSeries()

    def updateSeries(self):
        if self.axisX is not None:
            self._chart.removeAxis(self.axisX)
        self.axisX = QDateTimeAxis()
        self.axisX.setTickCount(self.calculateTick())
        self.axisX.setFormat("d MMM hh:mm")
        self.axisX.setTitleText(self.XTitle)
        self._chart.addAxis(self.axisX, Qt.AlignBottom)

        if self.axisY is not None:
            self._chart.removeAxis(self.axisY)
        self.axisY = QValueAxis()
        self.axisY.setLabelFormat("%i")
        self.axisY.setTitleText(self.YTitle)
        self._chart.addAxis(self.axisY, Qt.AlignLeft)

        self._series.attachAxis(self.axisX)
        self._series.attachAxis(self.axisY)

    def series(self):
        return self._series

    def resizeEvent(self, event):
        self.updateSeries()
        QChartView.resizeEvent(self, event)

    def calculateTick(self):
        width = self.frameGeometry().width()
        return int(round((width * 3) / 376))

```

```
    @pyqtSlot(QLineSeries)
    def setSeries(self, series):
        self.old_series = self._series
        self._series = series
        self.seriesChanged.emit(series)
        self.update()

    series = pyqtProperty(QLineSeries, series, setSeries)
```

compassWidget.py

```
import sys
from PyQt5.QtGui import *
from PyQt5.QtCore import *
from PyQt5.QtWidgets import *


class compassWidget(QWidget):
    angleChanged = pyqtSignal(float)

    def __init__(self, parent=None):

        QWidget.__init__(self, parent)

        self._angle = 0.0
        self._margins = 10
        self._pointText = {0: "N", 45: "NE", 90: "E", 135: "SE", 180: "S",
                           225: "SW", 270: "W", 315: "NW"}

        self.setWindowFlags(Qt.FramelessWindowHint)
        self.setAttribute(Qt.WA_TranslucentBackground)
        self.setStyleSheet("background:transparent;")
        self.setMinimumSize(200, 200)

    def paintEvent(self, event):

        painter = QPainter()
        painter.begin(self)
        painter.setRenderHint(QPainter.Antialiasing)

        painter.fillRect(event.rect(), self.palette().brush(QPalette.Window))
        self.drawMarkings(painter)
        self.drawNeedle(painter)

        painter.end()

    def drawMarkings(self, painter):

        painter.save()
        painter.translate(self.width() / 2, self.height() / 2)
        scale = min((self.width() - self._margins) / 120.0,
                    (self.height() - self._margins) / 120.0)
        painter.scale(scale, scale)

        font = QFont(self.font())
        font.setPixelSize(10)
        metrics = QFontMetricsF(font)

```

```
        painter.setFont(font)

        i = 0
        while i < 360:

            if i % 45 == 0:
                painter.drawLine(0, -40, 0, -50)
                painter.drawText(-metrics.width(self._pointText[i]) / 2.0, -52,
                                 self._pointText[i])
            else:
                painter.drawLine(0, -45, 0, -50)

            painter.rotate(15)
            i += 15

        painter.restore()

    def drawNeedle(self, painter):

        painter.save()
        painter.translate(self.width() / 2, self.height() / 2)
        painter.rotate(self._angle)
        scale = min((self.width() - self._margins) / 120.0,
                    (self.height() - self._margins) / 120.0)
        painter.scale(scale, scale)

        painter.setPen(QPen(Qt.NoPen))
        painter.setBrush(self.palette().brush(QPalette.Shadow))

        painter.drawPolygon(
            QPolygon([QPoint(-10, 0), QPoint(0, -45), QPoint(10, 0),
                      QPoint(0, 45), QPoint(-10, 0)])
        )

        palette = self.palette()
        palette.setColor(QPalette.Highlight, Qt.red)
        painter.setBrush(palette.brush(QPalette.Highlight))

        painter.drawPolygon(
            QPolygon([QPoint(-5, -25), QPoint(0, -45), QPoint(5, -25),
                      QPoint(0, -30), QPoint(-5, -25)])
        )

        painter.restore()

    def sizeHint(self):

        return QSize(150, 150)

    def angle(self):
        return self._angle

    @pyqtSlot(float)
    def setAngle(self, angle):

        if angle != self._angle:
            self._angle = angle
            self.angleChanged.emit(angle)
```

```
            self.update()

    angle = pyqtProperty(float, angle, setAngle)
```

worker.py

```
import sys
from PyQt5.QtCore import *
import traceback
from Worker.workerSignals import WorkerSignals


class Worker(QRunnable):
    '''
    Worker thread

    Inherits from QRunnable to handler worker thread setup, signals and wrap-up.

    :param callback: The function callback to run on this worker thread. Supplied args and
                     kwargs will be passed through to the runner.
    :type callback: function
    :param args: Arguments to pass to the callback function
    :param kwargs: Keywords to pass to the callback function

    '''

    def __init__(self, fn, *args, **kwargs):
        super(Worker, self).__init__()

        # Store constructor arguments (re-used for processing)

        self.fn = fn
        self.args = args
        self.kwargs = kwargs
        self.signals = WorkerSignals()

        # Add the callback to our kwargs
        self.kwargs['process_callback'] = self.signals.progress
        self.kwargs['UIUpdateCallBack'] = self.signals.statusbar

    @pyqtSlot()
    def run(self):
        '''
        Initialise the runner function with passed args, kwargs.
        '''

        # Retrieve args/kwargs here; and fire processing using them
        try:
            self.fn(*self.args, **self.kwargs)
        except:
            traceback.print_exc()
            exctype, value = sys.exc_info()[:2]
            self.signals.error.emit((exctype, value, traceback.format_exc()))
```

workerSignals.py

```
from PyQt5.QtCore import *
```

```python


class WorkerSignals(QObject):
    '''
    Defines the signals available from a running worker thread.

    Supported signals are:

    result
        `object` data returned from processing, anything

    progress
        `int` indicating % progress

    '''
    error = pyqtSignal(tuple)
    progress = pyqtSignal(list)
    statusbar = pyqtSignal(bool)
```

signalWakeupHandler.py

```python
import sys, signal, socket
from PyQt5.QtCore import *
from PyQt5.QtNetwork import *


class SignalWakeupHandler(QAbstractSocket):

    def __init__(self, parent=None):
        super().__init__(QAbstractSocket.UdpSocket, parent)
        self.old_fd = None
        # Create a socket pair
        self.wsock, self.rsock = socket.socketpair(type=socket.SOCK_STREAM)
        # Let Qt listen on the one end
        self.setSocketDescriptor(self.rsock.fileno())
        # And let Python write on the other end
        self.wsock.setblocking(False)
        self.old_fd = signal.set_wakeup_fd(self.wsock.fileno())
        # First Python code executed gets any exception from
        # the signal handler, so add a dummy handler first
        self.readyRead.connect(lambda: None)
        # Second handler does the real handling
        self.readyRead.connect(self._readSignal)

    def __del__(self):
        # Restore any old handler on deletion
        if self.old_fd is not None and signal and signal.set_wakeup_fd:
            signal.set_wakeup_fd(self.old_fd)

    def _readSignal(self):
        # Read the written byte.
        # Note: readyRead is blocked from occuring again until readData()
        # was called, so call it, even if you don't need the value.
        data = self.readData(1)
        # Emit a Qt signal for convenience
        self.signalReceived.emit(data[0])

    signalReceived = pyqtSignal(int)
```

mainWindow.ui

این فایل به طور خودکار با استفاده از نرم‌افزار Qt Designer ساخته شده است.

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>450</width>
    <height>477</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>450</width>
    <height>477</height>
   </size>
  </property>
  <property name="windowTitle">
   <string>Weather Station</string>
  </property>
  <property name="toolTip">
   <string/>
  </property>
  <property name="statusTip">
   <string/>
  </property>
  <property name="accessibleName">
   <string>Weather Station</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <property name="mouseTracking">
    <bool>false</bool>
   </property>
   <property name="toolTip">
    <string/>
   </property>
   <property name="autoFillBackground">
    <bool>true</bool>
   </property>
   <layout class="QGridLayout" name="gridLayout">
    <item row="0" column="0">
     <widget class="QTabWidget" name="tabWidget">
      <property name="sizePolicy">
       <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
        <horstretch>0</horstretch>
        <verstretch>0</verstretch>
       </sizepolicy>
      </property>
```

```
56        <property name="tabPosition">
57         <enum>QTabWidget::North</enum>
58        </property>
59        <property name="tabShape">
60         <enum>QTabWidget::Rounded</enum>
61        </property>
62        <property name="currentIndex">
63         <number>0</number>
64        </property>
65        <property name="elideMode">
66         <enum>Qt::ElideNone</enum>
67        </property>
68        <widget class="QWidget" name="mainTab">
69         <property name="sizePolicy">
70          <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
71           <horstretch>0</horstretch>
72           <verstretch>0</verstretch>
73          </sizepolicy>
74         </property>
75         <attribute name="title">
76          <string>Home</string>
77         </attribute>
78         <layout class="QVBoxLayout" name="verticalLayout_3">
79          <item>
80           <layout class="QHBoxLayout" name="horizontalLayout_11">
81            <item>
82             <layout class="QHBoxLayout" name="horizontalLayout_12">
83              <item>
84               <spacer name="horizontalSpacer_15">
85                <property name="orientation">
86                 <enum>Qt::Horizontal</enum>
87                </property>
88                <property name="sizeHint" stdset="0">
89                 <size>
90                  <width>40</width>
91                  <height>20</height>
92                 </size>
93                </property>
94               </spacer>
95              </item>
96              <item>
97               <widget class="QLabel" name="dateIcon">
98                <property name="maximumSize">
99                 <size>
100                 <width>32</width>
101                 <height>32</height>
102                </size>
103               </property>
104               <property name="text">
105                <string/>
106               </property>
107               <property name="pixmap">
108                <pixmap>Assets/Calendar.png</pixmap>
109               </property>
110               <property name="scaledContents">
111                <bool>true</bool>
112               </property>
113              </widget>
```

```
114            </item>
115            <item>
116             <widget class="QLabel" name="dateText">
117              <property name="text">
118               <string>20 December 2020</string>
119              </property>
120             </widget>
121            </item>
122            <item>
123             <spacer name="horizontalSpacer_16">
124              <property name="orientation">
125               <enum>Qt::Horizontal</enum>
126              </property>
127              <property name="sizeHint" stdset="0">
128               <size>
129                <width>40</width>
130                <height>20</height>
131               </size>
132              </property>
133             </spacer>
134            </item>
135           </layout>
136          </item>
137          <item>
138           <layout class="QHBoxLayout" name="horizontalLayout_13">
139            <item>
140             <spacer name="horizontalSpacer_17">
141              <property name="orientation">
142               <enum>Qt::Horizontal</enum>
143              </property>
144              <property name="sizeHint" stdset="0">
145               <size>
146                <width>40</width>
147                <height>20</height>
148               </size>
149              </property>
150             </spacer>
151            </item>
152            <item>
153             <widget class="QLabel" name="timeIcon">
154              <property name="maximumSize">
155               <size>
156                <width>32</width>
157                <height>32</height>
158               </size>
159              </property>
160              <property name="text">
161               <string/>
162              </property>
163              <property name="pixmap">
164               <pixmap>Assets/Clock.png</pixmap>
165              </property>
166              <property name="scaledContents">
167               <bool>true</bool>
168              </property>
169             </widget>
170            </item>
171            <item>
```

```
172             <widget class="QLabel" name="timeText">
173              <property name="text">
174               <string>10:20:30</string>
175              </property>
176             </widget>
177            </item>
178            <item>
179             <spacer name="horizontalSpacer_18">
180              <property name="orientation">
181               <enum>Qt::Horizontal</enum>
182              </property>
183              <property name="sizeHint" stdset="0">
184               <size>
185                <width>40</width>
186                <height>20</height>
187               </size>
188              </property>
189             </spacer>
190            </item>
191           </layout>
192          </item>
193         </layout>
194        </item>
195        <item>
196         <layout class="QVBoxLayout" name="windDetails">
197          <item>
198           <layout class="QVBoxLayout" name="compassContainer">
199            <item>
200             <spacer name="verticalSpacer">
201              <property name="orientation">
202               <enum>Qt::Vertical</enum>
203              </property>
204              <property name="sizeHint" stdset="0">
205               <size>
206                <width>20</width>
207                <height>40</height>
208               </size>
209              </property>
210             </spacer>
211            </item>
212            <item>
213             <layout class="QHBoxLayout" name="horizontalLayout_10">
214              <item>
215               <spacer name="horizontalSpacer_14">
216                <property name="orientation">
217                 <enum>Qt::Horizontal</enum>
218                </property>
219                <property name="sizeHint" stdset="0">
220                 <size>
221                  <width>40</width>
222                  <height>20</height>
223                 </size>
224                </property>
225               </spacer>
226              </item>
227              <item>
228               <widget class="QLabel" name="windAngleText">
229                <property name="text">
```

```
230                 <string>179 °</string>
231                 </property>
232                 </widget>
233                 </item>
234                 <item>
235                 <spacer name="horizontalSpacer_13">
236                 <property name="orientation">
237                 <enum>Qt::Horizontal</enum>
238                 </property>
239                 <property name="sizeHint" stdset="0">
240                 <size>
241                 <width>40</width>
242                 <height>20</height>
243                 </size>
244                 </property>
245                 </spacer>
246                 </item>
247                 </layout>
248                 </item>
249                 <item>
250                 <layout class="QHBoxLayout" name="horizontalLayout_9">
251                 <item>
252                 <spacer name="horizontalSpacer_12">
253                 <property name="orientation">
254                 <enum>Qt::Horizontal</enum>
255                 </property>
256                 <property name="sizeHint" stdset="0">
257                 <size>
258                 <width>40</width>
259                 <height>20</height>
260                 </size>
261                 </property>
262                 </spacer>
263                 </item>
264                 <item>
265                 <layout class="QVBoxLayout" name="compassLayout"/>
266                 </item>
267                 <item>
268                 <spacer name="horizontalSpacer_11">
269                 <property name="orientation">
270                 <enum>Qt::Horizontal</enum>
271                 </property>
272                 <property name="sizeHint" stdset="0">
273                 <size>
274                 <width>40</width>
275                 <height>20</height>
276                 </size>
277                 </property>
278                 </spacer>
279                 </item>
280                 </layout>
281                 </item>
282                 <item>
283                 <spacer name="verticalSpacer_2">
284                 <property name="orientation">
285                 <enum>Qt::Vertical</enum>
286                 </property>
287                 <property name="sizeHint" stdset="0">
```

```
288                 <size>
289                  <width>20</width>
290                  <height>40</height>
291                 </size>
292                </property>
293               </spacer>
294              </item>
295             </layout>
296            </item>
297            <item>
298             <layout class="QHBoxLayout" name="horizontalLayout_8">
299              <item>
300               <spacer name="horizontalSpacer_10">
301                <property name="orientation">
302                 <enum>Qt::Horizontal</enum>
303                </property>
304                <property name="sizeHint" stdset="0">
305                 <size>
306                  <width>40</width>
307                  <height>20</height>
308                 </size>
309                </property>
310               </spacer>
311              </item>
312              <item>
313               <widget class="QLabel" name="windSpeedIcon">
314                <property name="maximumSize">
315                 <size>
316                  <width>32</width>
317                  <height>32</height>
318                 </size>
319                </property>
320                <property name="text">
321                 <string/>
322                </property>
323                <property name="pixmap">
324                 <pixmap>Assets/windSpeed.png</pixmap>
325                </property>
326                <property name="scaledContents">
327                 <bool>true</bool>
328                </property>
329               </widget>
330              </item>
331              <item>
332               <widget class="QLabel" name="windSpeedText">
333                <property name="text">
334                 <string>3 m/s</string>
335                </property>
336               </widget>
337              </item>
338              <item>
339               <spacer name="horizontalSpacer_9">
340                <property name="orientation">
341                 <enum>Qt::Horizontal</enum>
342                </property>
343                <property name="sizeHint" stdset="0">
344                 <size>
345                  <width>40</width>
```

```
346                 <height>20</height>
347                 </size>
348                 </property>
349                 </spacer>
350                 </item>
351                 </layout>
352                 </item>
353                 </layout>
354                 </item>
355                 <item>
356                 <layout class="QHBoxLayout" name="horizontalLayout">
357                 <property name="spacing">
358                 <number>6</number>
359                 </property>
360                 <item>
361                 <layout class="QHBoxLayout" name="horizontalLayout_2">
362                 <item>
363                 <spacer name="horizontalSpacer">
364                 <property name="orientation">
365                 <enum>Qt::Horizontal</enum>
366                 </property>
367                 <property name="sizeHint" stdset="0">
368                 <size>
369                 <width>40</width>
370                 <height>20</height>
371                 </size>
372                 </property>
373                 </spacer>
374                 </item>
375                 <item>
376                 <widget class="QLabel" name="temperatureIcon">
377                 <property name="sizePolicy">
378                 <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
379                 <horstretch>0</horstretch>
380                 <verstretch>0</verstretch>
381                 </sizepolicy>
382                 </property>
383                 <property name="maximumSize">
384                 <size>
385                 <width>32</width>
386                 <height>32</height>
387                 </size>
388                 </property>
389                 <property name="frameShadow">
390                 <enum>QFrame::Plain</enum>
391                 </property>
392                 <property name="text">
393                 <string/>
394                 </property>
395                 <property name="textFormat">
396                 <enum>Qt::AutoText</enum>
397                 </property>
398                 <property name="pixmap">
399                 <pixmap>Assets/Temperature.png</pixmap>
400                 </property>
401                 <property name="scaledContents">
402                 <bool>true</bool>
403                 </property>
```

```
404                 <property name="wordWrap">
405                  <bool>false</bool>
406                 </property>
407                </widget>
408               </item>
409               <item>
410                <widget class="QLabel" name="temperatureText">
411                 <property name="sizePolicy">
412                  <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
413                   <horstretch>0</horstretch>
414                   <verstretch>0</verstretch>
415                  </sizepolicy>
416                 </property>
417                 <property name="text">
418                  <string>25 °C</string>
419                 </property>
420                </widget>
421               </item>
422               <item>
423                <spacer name="horizontalSpacer_2">
424                 <property name="orientation">
425                  <enum>Qt::Horizontal</enum>
426                 </property>
427                 <property name="sizeHint" stdset="0">
428                  <size>
429                   <width>40</width>
430                   <height>20</height>
431                  </size>
432                 </property>
433                </spacer>
434               </item>
435              </layout>
436             </item>
437             <item>
438              <layout class="QHBoxLayout" name="horizontalLayout_3">
439               <item>
440                <spacer name="horizontalSpacer_3">
441                 <property name="orientation">
442                  <enum>Qt::Horizontal</enum>
443                 </property>
444                 <property name="sizeHint" stdset="0">
445                  <size>
446                   <width>40</width>
447                   <height>20</height>
448                  </size>
449                 </property>
450                </spacer>
451               </item>
452               <item>
453                <widget class="QLabel" name="lightIntensityIcon">
454                 <property name="sizePolicy">
455                  <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
456                   <horstretch>0</horstretch>
457                   <verstretch>0</verstretch>
458                  </sizepolicy>
459                 </property>
460                 <property name="maximumSize">
461                  <size>
```

```
462                 <width>32</width>
463                 <height>32</height>
464                 </size>
465                 </property>
466                 <property name="text">
467                 <string/>
468                 </property>
469                 <property name="pixmap">
470                 <pixmap>Assets/Sun.png</pixmap>
471                 </property>
472                 <property name="scaledContents">
473                 <bool>true</bool>
474                 </property>
475                 </widget>
476               </item>
477               <item>
478                 <widget class="QLabel" name="lightIntensityText">
479                 <property name="text">
480                 <string>500 LUX</string>
481                 </property>
482                 </widget>
483               </item>
484               <item>
485                 <spacer name="horizontalSpacer_4">
486                 <property name="orientation">
487                 <enum>Qt::Horizontal</enum>
488                 </property>
489                 <property name="sizeHint" stdset="0">
490                 <size>
491                 <width>40</width>
492                 <height>20</height>
493                 </size>
494                 </property>
495                 </spacer>
496               </item>
497             </layout>
498           </item>
499           <item>
500             <layout class="QHBoxLayout" name="horizontalLayout_6">
501               <item>
502                 <spacer name="horizontalSpacer_5">
503                 <property name="orientation">
504                 <enum>Qt::Horizontal</enum>
505                 </property>
506                 <property name="sizeHint" stdset="0">
507                 <size>
508                 <width>40</width>
509                 <height>20</height>
510                 </size>
511                 </property>
512                 </spacer>
513               </item>
514               <item>
515                 <widget class="QLabel" name="humidityIcon">
516                 <property name="maximumSize">
517                 <size>
518                 <width>32</width>
519                 <height>32</height>
```

```
520                 </size>
521                </property>
522                <property name="text">
523                 <string/>
524                </property>
525                <property name="pixmap">
526                 <pixmap>Assets/Humidity.png</pixmap>
527                </property>
528                <property name="scaledContents">
529                 <bool>true</bool>
530                </property>
531               </widget>
532              </item>
533              <item>
534               <widget class="QLabel" name="humidityText">
535                <property name="text">
536                 <string>50 %</string>
537                </property>
538               </widget>
539              </item>
540              <item>
541               <spacer name="horizontalSpacer_6">
542                <property name="orientation">
543                 <enum>Qt::Horizontal</enum>
544                </property>
545                <property name="sizeHint" stdset="0">
546                 <size>
547                  <width>40</width>
548                  <height>20</height>
549                 </size>
550                </property>
551               </spacer>
552              </item>
553             </layout>
554            </item>
555            <item>
556             <layout class="QHBoxLayout" name="horizontalLayout_7">
557              <item>
558               <spacer name="horizontalSpacer_7">
559                <property name="orientation">
560                 <enum>Qt::Horizontal</enum>
561                </property>
562                <property name="sizeHint" stdset="0">
563                 <size>
564                  <width>40</width>
565                  <height>20</height>
566                 </size>
567                </property>
568               </spacer>
569              </item>
570              <item>
571               <widget class="QLabel" name="pressureIcon">
572                <property name="maximumSize">
573                 <size>
574                  <width>32</width>
575                  <height>32</height>
576                 </size>
577                </property>
```

```
578                <property name="text">
579                 <string/>
580                </property>
581                <property name="pixmap">
582                 <pixmap>Assets/AirPressure.png</pixmap>
583                </property>
584                <property name="scaledContents">
585                 <bool>true</bool>
586                </property>
587               </widget>
588              </item>
589              <item>
590               <widget class="QLabel" name="pressureText">
591                <property name="text">
592                 <string>860 hPa</string>
593                </property>
594               </widget>
595              </item>
596              <item>
597               <spacer name="horizontalSpacer_8">
598                <property name="orientation">
599                 <enum>Qt::Horizontal</enum>
600                </property>
601                <property name="sizeHint" stdset="0">
602                 <size>
603                  <width>40</width>
604                  <height>20</height>
605                 </size>
606                </property>
607               </spacer>
608              </item>
609             </layout>
610            </item>
611           </layout>
612          </item>
613         </layout>
614        </widget>
615        <widget class="QWidget" name="chartTab">
616         <attribute name="title">
617          <string>Charts</string>
618         </attribute>
619         <layout class="QGridLayout" name="gridLayout_2">
620          <item row="0" column="0">
621           <widget class="QScrollArea" name="scrollArea">
622            <property name="autoFillBackground">
623             <bool>true</bool>
624            </property>
625            <property name="widgetResizable">
626             <bool>true</bool>
627            </property>
628            <widget class="QWidget" name="chartTabScrollArea">
629             <property name="geometry">
630              <rect>
631               <x>0</x>
632               <y>0</y>
633               <width>420</width>
634               <height>371</height>
635              </rect>
```

```
636            </property>
637            <layout class="QVBoxLayout" name="verticalLayout_2">
638             <item>
639              <layout class="QVBoxLayout" name="chartTabContents">
640               <property name="spacing">
641                <number>6</number>
642               </property>
643               <item>
644                <layout class="QHBoxLayout" name="horizontalLayout_4">
645                 <item>
646                  <spacer name="horizontalSpacer_19">
647                   <property name="orientation">
648                    <enum>Qt::Horizontal</enum>
649                   </property>
650                   <property name="sizeHint" stdset="0">
651                    <size>
652                     <width>40</width>
653                     <height>20</height>
654                    </size>
655                   </property>
656                  </spacer>
657                 </item>
658                 <item>
659                  <widget class="QDateEdit" name="firstDate">
660                   <property name="toolTip">
661                    <string>dd/MM/yyyy</string>
662                   </property>
663                   <property name="displayFormat">
664                    <string>dd/MM/yyyy</string>
665                   </property>
666                   <property name="date">
667                    <date>
668                     <year>2020</year>
669                     <month>7</month>
670                     <day>1</day>
671                    </date>
672                   </property>
673                  </widget>
674                 </item>
675                 <item>
676                  <widget class="QDateEdit" name="secondDate">
677                   <property name="toolTip">
678                    <string>dd/MM/yyyy</string>
679                   </property>
680                   <property name="displayFormat">
681                    <string>dd/MM/yyyy</string>
682                   </property>
683                   <property name="date">
684                    <date>
685                     <year>2020</year>
686                     <month>7</month>
687                     <day>8</day>
688                    </date>
689                   </property>
690                  </widget>
691                 </item>
692                 <item>
693                  <spacer name="horizontalSpacer_20">
```

```
694                     <property name="orientation">
695                      <enum>Qt::Horizontal</enum>
696                     </property>
697                     <property name="sizeHint" stdset="0">
698                      <size>
699                       <width>40</width>
700                       <height>20</height>
701                      </size>
702                     </property>
703                    </spacer>
704                   </item>
705                  </layout>
706                 </item>
707                </layout>
708               </item>
709              </layout>
710             </widget>
711            </widget>
712           </item>
713          </layout>
714         </widget>
715         <widget class="QWidget" name="logsTab">
716          <property name="sizePolicy">
717           <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
718            <horstretch>0</horstretch>
719            <verstretch>0</verstretch>
720           </sizepolicy>
721          </property>
722          <attribute name="title">
723           <string>Log</string>
724          </attribute>
725          <layout class="QVBoxLayout" name="verticalLayout">
726           <item>
727            <widget class="QCheckBox" name="checkBox">
728             <property name="text">
729              <string>Auto scroll</string>
730             </property>
731             <property name="checked">
732              <bool>true</bool>
733             </property>
734             <property name="tristate">
735              <bool>false</bool>
736             </property>
737            </widget>
738           </item>
739           <item>
740            <widget class="QTextBrowser" name="textBrowser">
741             <property name="sizePolicy">
742              <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
743               <horstretch>0</horstretch>
744               <verstretch>0</verstretch>
745              </sizepolicy>
746             </property>
747             <property name="cursor" stdset="0">
748              <cursorShape>IBeamCursor</cursorShape>
749             </property>
750             <property name="focusPolicy">
751              <enum>Qt::NoFocus</enum>
```

```
            </property>
            <property name="acceptDrops">
             <bool>false</bool>
            </property>
            <property name="inputMethodHints">
             <set>Qt::ImhNone</set>
            </property>
            <property name="documentTitle">
             <string/>
            </property>
            <property name="readOnly">
             <bool>true</bool>
            </property>
           </widget>
          </item>
         </layout>
        </widget>
       </widget>
      </item>
     </layout>
    </widget>
    <widget class="QMenuBar" name="menubar">
     <property name="geometry">
      <rect>
       <x>0</x>
       <y>0</y>
       <width>450</width>
       <height>26</height>
      </rect>
     </property>
    </widget>
    <widget class="QStatusBar" name="statusbar"/>
   </widget>
   <resources/>
   <connections/>
</ui>
```

mainWindow.py

این فایل به طور خودکار توسط ابزار pyuic5 و با استفاده از فایل mainWindow.ui توسط دستور زیر ساخته شده است:

pyuic5 -x mainWindow.ui -o ../mainWindow.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'UI/template/mainWindow.ui'
#
# Created by: PyQt5 UI code generator 5.15.0
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
```

```
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(450, 477)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())
        MainWindow.setSizePolicy(sizePolicy)
        MainWindow.setMinimumSize(QtCore.QSize(450, 477))
        MainWindow.setToolTip("")
        MainWindow.setStatusTip("")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setMouseTracking(False)
        self.centralwidget.setToolTip("")
        self.centralwidget.setAutoFillBackground(True)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.tabWidget.sizePolicy().hasHeightForWidth())
        self.tabWidget.setSizePolicy(sizePolicy)
        self.tabWidget.setTabPosition(QtWidgets.QTabWidget.North)
        self.tabWidget.setTabShape(QtWidgets.QTabWidget.Rounded)
        self.tabWidget.setElideMode(QtCore.Qt.ElideNone)
        self.tabWidget.setObjectName("tabWidget")
        self.mainTab = QtWidgets.QWidget()
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.mainTab.sizePolicy().hasHeightForWidth())
        self.mainTab.setSizePolicy(sizePolicy)
        self.mainTab.setObjectName("mainTab")
        self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.mainTab)
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.horizontalLayout_11 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_11.setObjectName("horizontalLayout_11")
        self.horizontalLayout_12 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_12.setObjectName("horizontalLayout_12")
        spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_12.addItem(spacerItem)
        self.dateIcon = QtWidgets.QLabel(self.mainTab)
        self.dateIcon.setMaximumSize(QtCore.QSize(32, 32))
        self.dateIcon.setText("")
        self.dateIcon.setPixmap(QtGui.QPixmap("UI/template/Assets/Calendar.png"))
        self.dateIcon.setScaledContents(True)
        self.dateIcon.setObjectName("dateIcon")
        self.horizontalLayout_12.addWidget(self.dateIcon)
        self.dateText = QtWidgets.QLabel(self.mainTab)
        self.dateText.setObjectName("dateText")
        self.horizontalLayout_12.addWidget(self.dateText)
        spacerItem1 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
```

```
69          self.horizontalLayout_12.addItem(spacerItem1)
70          self.horizontalLayout_11.addLayout(self.horizontalLayout_12)
71          self.horizontalLayout_13 = QtWidgets.QHBoxLayout()
72          self.horizontalLayout_13.setObjectName("horizontalLayout_13")
73          spacerItem2 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
74          self.horizontalLayout_13.addItem(spacerItem2)
75          self.timeIcon = QtWidgets.QLabel(self.mainTab)
76          self.timeIcon.setMaximumSize(QtCore.QSize(32, 32))
77          self.timeIcon.setText("")
78          self.timeIcon.setPixmap(QtGui.QPixmap("UI/template/Assets/Clock.png"))
79          self.timeIcon.setScaledContents(True)
80          self.timeIcon.setObjectName("timeIcon")
81          self.horizontalLayout_13.addWidget(self.timeIcon)
82          self.timeText = QtWidgets.QLabel(self.mainTab)
83          self.timeText.setObjectName("timeText")
84          self.horizontalLayout_13.addWidget(self.timeText)
85          spacerItem3 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
86          self.horizontalLayout_13.addItem(spacerItem3)
87          self.horizontalLayout_11.addLayout(self.horizontalLayout_13)
88          self.verticalLayout_3.addLayout(self.horizontalLayout_11)
89          self.windDetails = QtWidgets.QVBoxLayout()
90          self.windDetails.setObjectName("windDetails")
91          self.compassContainer = QtWidgets.QVBoxLayout()
92          self.compassContainer.setObjectName("compassContainer")
93          spacerItem4 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
    QtWidgets.QSizePolicy.Expanding)
94          self.compassContainer.addItem(spacerItem4)
95          self.horizontalLayout_10 = QtWidgets.QHBoxLayout()
96          self.horizontalLayout_10.setObjectName("horizontalLayout_10")
97          spacerItem5 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
98          self.horizontalLayout_10.addItem(spacerItem5)
99          self.windAngleText = QtWidgets.QLabel(self.mainTab)
100         self.windAngleText.setObjectName("windAngleText")
101         self.horizontalLayout_10.addWidget(self.windAngleText)
102         spacerItem6 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
103         self.horizontalLayout_10.addItem(spacerItem6)
104         self.compassContainer.addLayout(self.horizontalLayout_10)
105         self.horizontalLayout_9 = QtWidgets.QHBoxLayout()
106         self.horizontalLayout_9.setObjectName("horizontalLayout_9")
107         spacerItem7 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
108         self.horizontalLayout_9.addItem(spacerItem7)
109         self.compassLayout = QtWidgets.QVBoxLayout()
110         self.compassLayout.setObjectName("compassLayout")
111         self.horizontalLayout_9.addLayout(self.compassLayout)
112         spacerItem8 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
113         self.horizontalLayout_9.addItem(spacerItem8)
114         self.compassContainer.addLayout(self.horizontalLayout_9)
115         spacerItem9 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
    QtWidgets.QSizePolicy.Expanding)
116         self.compassContainer.addItem(spacerItem9)
117         self.windDetails.addLayout(self.compassContainer)
118         self.horizontalLayout_8 = QtWidgets.QHBoxLayout()
```

```
        self.horizontalLayout_8.setObjectName("horizontalLayout_8")
        spacerItem10 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_8.addItem(spacerItem10)
        self.windSpeedIcon = QtWidgets.QLabel(self.mainTab)
        self.windSpeedIcon.setMaximumSize(QtCore.QSize(32, 32))
        self.windSpeedIcon.setText("")
        self.windSpeedIcon.setPixmap(QtGui.QPixmap("UI/template/Assets/windSpeed.png"))
        self.windSpeedIcon.setScaledContents(True)
        self.windSpeedIcon.setObjectName("windSpeedIcon")
        self.horizontalLayout_8.addWidget(self.windSpeedIcon)
        self.windSpeedText = QtWidgets.QLabel(self.mainTab)
        self.windSpeedText.setObjectName("windSpeedText")
        self.horizontalLayout_8.addWidget(self.windSpeedText)
        spacerItem11 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_8.addItem(spacerItem11)
        self.windDetails.addLayout(self.horizontalLayout_8)
        self.verticalLayout_3.addLayout(self.windDetails)
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setSpacing(6)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        spacerItem12 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_2.addItem(spacerItem12)
        self.temperatureIcon = QtWidgets.QLabel(self.mainTab)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.temperatureIcon.sizePolicy().hasHeightForWidth())
        self.temperatureIcon.setSizePolicy(sizePolicy)
        self.temperatureIcon.setMaximumSize(QtCore.QSize(32, 32))
        self.temperatureIcon.setFrameShadow(QtWidgets.QFrame.Plain)
        self.temperatureIcon.setText("")
        self.temperatureIcon.setTextFormat(QtCore.Qt.AutoText)
        self.temperatureIcon.setPixmap(QtGui.QPixmap("UI/template/Assets/Temperature.png"))
        self.temperatureIcon.setScaledContents(True)
        self.temperatureIcon.setWordWrap(False)
        self.temperatureIcon.setObjectName("temperatureIcon")
        self.horizontalLayout_2.addWidget(self.temperatureIcon)
        self.temperatureText = QtWidgets.QLabel(self.mainTab)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.temperatureText.sizePolicy().hasHeightForWidth())
        self.temperatureText.setSizePolicy(sizePolicy)
        self.temperatureText.setObjectName("temperatureText")
        self.horizontalLayout_2.addWidget(self.temperatureText)
        spacerItem13 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
```

```
        self.horizontalLayout_2.addItem(spacerItem13)
        self.horizontalLayout.addLayout(self.horizontalLayout_2)
        self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_3.setObjectName("horizontalLayout_3")
        spacerItem14 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_3.addItem(spacerItem14)
        self.lightIntensityIcon = QtWidgets.QLabel(self.mainTab)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.
    QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.lightIntensityIcon.sizePolicy().
    hasHeightForWidth())
        self.lightIntensityIcon.setSizePolicy(sizePolicy)
        self.lightIntensityIcon.setMaximumSize(QtCore.QSize(32, 32))
        self.lightIntensityIcon.setText("")
        self.lightIntensityIcon.setPixmap(QtGui.QPixmap("UI/template/Assets/Sun.png"))
        self.lightIntensityIcon.setScaledContents(True)
        self.lightIntensityIcon.setObjectName("lightIntensityIcon")
        self.horizontalLayout_3.addWidget(self.lightIntensityIcon)
        self.lightIntensityText = QtWidgets.QLabel(self.mainTab)
        self.lightIntensityText.setObjectName("lightIntensityText")
        self.horizontalLayout_3.addWidget(self.lightIntensityText)
        spacerItem15 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_3.addItem(spacerItem15)
        self.horizontalLayout.addLayout(self.horizontalLayout_3)
        self.horizontalLayout_6 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_6.setObjectName("horizontalLayout_6")
        spacerItem16 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_6.addItem(spacerItem16)
        self.humidityIcon = QtWidgets.QLabel(self.mainTab)
        self.humidityIcon.setMaximumSize(QtCore.QSize(32, 32))
        self.humidityIcon.setText("")
        self.humidityIcon.setPixmap(QtGui.QPixmap("UI/template/Assets/Humidity.png"))
        self.humidityIcon.setScaledContents(True)
        self.humidityIcon.setObjectName("humidityIcon")
        self.horizontalLayout_6.addWidget(self.humidityIcon)
        self.humidityText = QtWidgets.QLabel(self.mainTab)
        self.humidityText.setObjectName("humidityText")
        self.horizontalLayout_6.addWidget(self.humidityText)
        spacerItem17 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_6.addItem(spacerItem17)
        self.horizontalLayout.addLayout(self.horizontalLayout_6)
        self.horizontalLayout_7 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_7.setObjectName("horizontalLayout_7")
        spacerItem18 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_7.addItem(spacerItem18)
        self.pressureIcon = QtWidgets.QLabel(self.mainTab)
        self.pressureIcon.setMaximumSize(QtCore.QSize(32, 32))
        self.pressureIcon.setText("")
        self.pressureIcon.setPixmap(QtGui.QPixmap("UI/template/Assets/AirPressure.png"
    ))
        self.pressureIcon.setScaledContents(True)
```

```
        self.pressureIcon.setObjectName("pressureIcon")
        self.horizontalLayout_7.addWidget(self.pressureIcon)
        self.pressureText = QtWidgets.QLabel(self.mainTab)
        self.pressureText.setObjectName("pressureText")
        self.horizontalLayout_7.addWidget(self.pressureText)
        spacerItem19 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_7.addItem(spacerItem19)
        self.horizontalLayout.addLayout(self.horizontalLayout_7)
        self.verticalLayout_3.addLayout(self.horizontalLayout)
        self.tabWidget.addTab(self.mainTab, "")
        self.chartTab = QtWidgets.QWidget()
        self.chartTab.setObjectName("chartTab")
        self.gridLayout_2 = QtWidgets.QGridLayout(self.chartTab)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.scrollArea = QtWidgets.QScrollArea(self.chartTab)
        self.scrollArea.setAutoFillBackground(True)
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")
        self.chartTabScrollArea = QtWidgets.QWidget()
        self.chartTabScrollArea.setGeometry(QtCore.QRect(0, 0, 420, 371))
        self.chartTabScrollArea.setObjectName("chartTabScrollArea")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.chartTabScrollArea)
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.chartTabContents = QtWidgets.QVBoxLayout()
        self.chartTabContents.setSpacing(6)
        self.chartTabContents.setObjectName("chartTabContents")
        self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_4.setObjectName("horizontalLayout_4")
        spacerItem20 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_4.addItem(spacerItem20)
        self.firstDate = QtWidgets.QDateEdit(self.chartTabScrollArea)
        self.firstDate.setDate(QtCore.QDate(2020, 7, 1))
        self.firstDate.setObjectName("firstDate")
        self.horizontalLayout_4.addWidget(self.firstDate)
        self.secondDate = QtWidgets.QDateEdit(self.chartTabScrollArea)
        self.secondDate.setDate(QtCore.QDate(2020, 7, 8))
        self.secondDate.setObjectName("secondDate")
        self.horizontalLayout_4.addWidget(self.secondDate)
        spacerItem21 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
    QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_4.addItem(spacerItem21)
        self.chartTabContents.addLayout(self.horizontalLayout_4)
        self.verticalLayout_2.addLayout(self.chartTabContents)
        self.scrollArea.setWidget(self.chartTabScrollArea)
        self.gridLayout_2.addWidget(self.scrollArea, 0, 0, 1, 1)
        self.tabWidget.addTab(self.chartTab, "")
        self.logsTab = QtWidgets.QWidget()
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.
    QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.logsTab.sizePolicy().hasHeightForWidth())
        self.logsTab.setSizePolicy(sizePolicy)
        self.logsTab.setObjectName("logsTab")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.logsTab)
        self.verticalLayout.setObjectName("verticalLayout")
```

```
        self.checkBox = QtWidgets.QCheckBox(self.logsTab)
        self.checkBox.setChecked(True)
        self.checkBox.setTristate(False)
        self.checkBox.setObjectName("checkBox")
        self.verticalLayout.addWidget(self.checkBox)
        self.textBrowser = QtWidgets.QTextBrowser(self.logsTab)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.textBrowser.sizePolicy().hasHeightForWidth())
        self.textBrowser.setSizePolicy(sizePolicy)
        self.textBrowser.viewport().setProperty("cursor", QtGui.QCursor(QtCore.Qt.IBeamCursor))
        self.textBrowser.setFocusPolicy(QtCore.Qt.NoFocus)
        self.textBrowser.setAcceptDrops(False)
        self.textBrowser.setInputMethodHints(QtCore.Qt.ImhNone)
        self.textBrowser.setDocumentTitle("")
        self.textBrowser.setReadOnly(True)
        self.textBrowser.setObjectName("textBrowser")
        self.verticalLayout.addWidget(self.textBrowser)
        self.tabWidget.addTab(self.logsTab, "")
        self.gridLayout.addWidget(self.tabWidget, 0, 0, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 450, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Weather Station"))
        MainWindow.setAccessibleName(_translate("MainWindow", "Weather Station"))
        self.dateText.setText(_translate("MainWindow", "20 December 2020"))
        self.timeText.setText(_translate("MainWindow", "10:20:30"))
        self.windAngleText.setText(_translate("MainWindow", "179 °"))
        self.windSpeedText.setText(_translate("MainWindow", "3 m/s"))
        self.temperatureText.setText(_translate("MainWindow", "25 °C"))
        self.lightIntensityText.setText(_translate("MainWindow", "500 LUX"))
        self.humidityText.setText(_translate("MainWindow", "50 %"))
        self.pressureText.setText(_translate("MainWindow", "860 hPa"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.mainTab), _translate("MainWindow", "Home"))
        self.firstDate.setToolTip(_translate("MainWindow", "dd/MM/yyyy"))
        self.firstDate.setDisplayFormat(_translate("MainWindow", "dd/MM/yyyy"))
        self.secondDate.setToolTip(_translate("MainWindow", "dd/MM/yyyy"))
        self.secondDate.setDisplayFormat(_translate("MainWindow", "dd/MM/yyyy"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.chartTab), _translate("MainWindow", "Charts"))
        self.checkBox.setText(_translate("MainWindow", "Auto scroll"))
```

```
324         self.tabWidget.setTabText(self.tabWidget.indexOf(self.logsTab), _translate("
    MainWindow", "Log"))
```