



دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده مهندسی برق

پروژه کارشناسی
گروه کنترل

طراحی و ساخت ساعت مچی هوشمند
با قابلیت تحلیل حرکات دست و پایش سلامت

نگارش
سلمان عامی مطلق

استاد راهنما
دکتر محمداعظم خسروی

۱۴۰۱ مرداد

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ



دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

به نام خدا

تعهدنامه اصالت اثر

تاریخ: مرداد ۱۴۰۱

اینجانب سلمان عامی مطلق متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظرارت و راهنمایی استادی دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مأخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است. در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مأخذ بلامانع است.

سلمان عامی مطلق

امضا

سپاسگزاری

بر خود لازم می‌دانم که از زحمات جناب آقای دکتر محمداعظم خسروی که در مراحل انجام این پروژه به عنوان استاد راهنمای و استاد مشاور در کنار بندۀ بودند و حمایت همه جانبه از من داشتند، تقدیر و تشکر به عمل آورم. امید است که توانسته باشم اندکی از الطافشان را جبران کنم.

از پدر و مادر عزیزم، که بیان تشکر از ایشان، از دامنه‌ی لغات فراگرفته در زندگی ام خارج است، کمال تشکر را دارم و امیدوارم ذره‌ای از زحمات بی‌ منتشران را جبران کرده باشم.

در پایان از استاد گرانقدر مهندس امیرحسن آشنایی، مهندس فرید کاویانی، دکتر علی وزیری، آقای امیرحسین آقاجری و مهندس سعید دیاری بابت راهنمایی‌های بی‌چشمداشت و دلسوزانه‌شان، تقدیر و تشکر می‌نمایم.

سلامان عامی مطلق
۱۴۰۱ مرداد

چکیده

امروزه جای تجهیزاتی از قبیل تلفن همراه و ساعت های هوشمند در زندگی مردم این دوره باز شده و استفاده رایجی دارند. لذا بهبود تعاملات انسان و سامانه های هوشمند می تواند برگ برنده ای برای این صنعت باشد. یکی از جنبه های این تعامل، برقراری ارتباط بین ساعت هوشمند، تلفن همراه و پایش حرکات فیزیکی است. در این پروژه ابتدا یک ساعت هوشمند به صورت کامل طراحی شده است. این طراحی شامل سخت افزار، نرم افزار و طراحی مکانیکی است. این دستگاه از بستر بلوتوث برای برقراری ارتباط با تلفن همراه استفاده می کند، دارای حسگر شتاب، حسگر سلامت (پالس اکسی متر)، بازر، موتور ایجاد لرزش، شارژر باتری لیتیومی، کلیدهای لمسی و صفحه نمایش است. به کمک حسگر شتاب و پردازش سیگنال آن، متغیرهای فضایی دست اندازه گیری شده و به کمک پیاده سازی دو فیلتر کالمون، اطلاعات حسگر فیلتر شدن. این تشخیص حرکت برای مواردی مثل شمارش گام و روشن شدن صفحه نمایش در صورت بالا آمدن دست استفاده شده اند. برای پیاده سازی این فیلتر از تکنیک جاگذاری مقدار نهایی ضرایب استفاده شده است که باعث کاهش چشمگیر حجم محاسبات، کاهش حافظه مورد نیاز و افزایش سرعت اجرا است. کار انجام شده پیش رو، نقطه ای پایانی برای این پروژه نیست و می توان ایده های زیادی برای پیشرفت و مسیر آینده ای این پروژه متصور شد. هدایت یک بازوی رباتیک بر اساس حرکت دست به کمک فناوری اینترنت اشیاء مثالی از این ایده ها است.

واژه های کلیدی:

ساعت هوشمند، فیلتر کالمون، ریز برد ارزش دهنده، تلفن همراه هوشمند، دستگاه پوشیدنی

فهرست مطالب

۱	مقدمه	۱
۳	۱-۱ زیربخش مقدمه	
۴	۲ سخت افزار و الکترونیک	۲
۵	۱-۲ کلیت شماتیک	
۶	۲-۲ فیبر مدار چاپی	
۷	۳-۲ هسته‌ی پردازشی	
۹	۴-۲ درگاه بلوتوث	
۱۱	۵-۲ حسگر شتاب خطی و سرعت زاویه‌ای	
۱۲	۶-۲ حسگر PPG	
۱۳	۷-۲ صفحه نمایش	
۱۴	۸-۲ درگاه ارتباط سریال	
۱۵	۹-۲ تعذیه و مدیریت توان	
۱۵	۱-۹-۲ باتری	
۱۵	۲-۹-۲ شارژ و مدیریت توان	
۱۸	۱۰-۲ کلیدهای لمسی	
۱۹	۱۱-۲ بازار	
۲۰	۱۲-۲ موتور ایجاد لرزش	
۲۲	۳ مکانیک و طراحی صنعتی	۳
۲۳	۱-۳ بدنی اصلی	
۲۳	۱-۱-۳ نسخه‌های اولیه	
۲۴	۲-۱-۳ نسخه نهایی	
۲۷	۲-۳ دریچه‌ی پشتی	
۲۸	۳-۳ اتصالات	
۲۹	۱-۳-۳ کلیدهای لمسی	
۳۰	۲-۳-۳ صفحه نمایش	
۳۰	۳-۳-۳ پی‌سی‌بی	
۳۱	۴-۳-۳ باتری	
۳۱	۵-۳-۳ PPG حسگر	
۳۲	۶-۳-۳ دریچه‌ی پشتی	
۳۳	۴ نرم افزار	
۳۵	۱-۴ تنظیمات سخت افزاری	

۳۶	۱-۱-۴ کلاک
۳۶	۲-۱-۴ برنامه‌ریزی و اشکال‌زدایی
۳۷	GPIO ۳-۱-۴
۳۸	RTC ۴-۱-۴
۳۸	۵-۱-۴ تایمیرها
۴۲	ADC ۶-۱-۴
۴۳	I2C ۷-۱-۴
۴۳	USART ۸-۱-۴
۴۴	۲-۴ معماری
۴۴	۱-۲-۴ ماشین حالت
۵۰	۲-۲-۴ وقفه‌ها
۵۳	۳-۲-۴ لایه‌های نرم‌افزاری
۵۹	۳-۴ پردازش داده‌های حسگر حرکتی
۶۰	۱-۳-۴ فیلتر کالمن
۶۷	۲-۳-۴ تشخیص گام
۶۷	۳-۳-۴ تشخیص چرخش دست
۶۷	۴-۴ ارتباطات
۶۸	۵ نتایج و جمع‌بندی
۷۰	منابع و مراجع

فهرست اشکال

صفحه

شکل

۱-۲	شماییک کلی ساعت و نحوه اتصال بخش‌های مختلف به یکدیگر	۵
۲-۲	پی‌سی‌بی طراحی شده در نرم‌افزار Altium Designer	۶
۳-۲	تصاویری از پی‌سی‌بی پروژه	۷
۴-۲	تصاویری از پردازنده STM32F030	۸
۵-۲	شماییک مربوط به بخش ریزپردازنده	۹
۶-۲	تصاویری از ماژول بلوتوث HC-05	۱۰
۷-۲	شماییک مربوط به بخش بلوتوث	۱۰
۸-۲	تصاویر حسگر حرکتی	۱۱
۹-۲	شماییک مربوط به بخش حسگر حرکتی	۱۱
۱۰-۲	تصاویر حسگر PPG	۱۲
۱۱-۲	شماییک مربوط به بخش حسگر PPG	۱۳
۱۲-۲	تصاویر صفحه‌ی نمایش	۱۳
۱۳-۲	شماییک مربوط به بخش نمایشگر	۱۴
۱۴-۲	تصاویر مربوط به درگاه ارتباط سریال	۱۴
۱۵-۲	باتری انتخاب شده برای پروژه	۱۵
۱۶-۲	شماییکی ساده برای توضیح بخش مدیریت توان	۱۶
۱۷-۲	تصاویر بخش تغذیه	۱۷
۱۸-۲	تصاویر بخش کلیدهای لمسی	۱۸
۱۹-۲	شماییک مربوط به بخش کلیدهای لمسی	۱۸
۲۰-۲	تصاویر بازر	۱۹
۲۱-۲	شماییک مربوط به بخش بازر	۲۰
۲۲-۲	تصاویر موتور ایجاد لرزش	۲۰
۲۳-۲	شماییک مربوط به بخش ایجاد لرزش	۲۱
۱-۳	تصاویر بدنی اصلی نسخه‌ی اول	۲۳
۲-۳	تصاویر بدنی اصلی نسخه‌ی دوم	۲۴
۳-۳	تصاویر بدنی اصلی نسخه‌ی سوم	۲۴
۴-۳	تصاویر محل نصب صفحه نمایش در بدن	۲۵
۵-۳	تصاویر محل نصب پی‌سی‌بی در بدن	۲۵
۶-۳	تصویر محل اتصال USB در بدن	۲۶
۷-۳	تصویر شیارهای عبور هوا	۲۶
۸-۳	تصاویر محل نصب بند	۲۷

۹-۳	تصویر چاپ شدهی نسخهی نهایی بدن از جنس رزین و شفاف	۲۷
۱۰-۳	تصویر استوانه‌های اتصال دریچه به بدن	۲۸
۱۱-۳	تصاویر محل نصب حسگر PPG	۲۸
۱۲-۳	تصویر چاپ شدهی نسخهی نهایی دریچه‌ی پشتی از جنس رزین و شفاف	۲۹
۱۳-۳	تصاویر اتصال کلیدهای لمسی به بدن	۲۹
۱۴-۳	تصویر اتصال صفحه نمایش به بدن	۳۰
۱۵-۳	تصویر اتصال پی‌سی‌بی به بدن	۳۰
۱۶-۳	تصاویر اتصال باتری	۳۱
۱۷-۳	تصاویر اتصال حسگر PPG	۳۱
۱۸-۳	تصاویر اتصال دریچه‌ی پشتی	۳۲
۱۹-۳	تصاویر بدنی کامل	۳۲
۱-۴	نحوی تخصیص پایه‌های مختلف ریزپردازنده در نرم‌افزار CubeMX	۳۵
۲-۴	تنظیمات کلاک	۳۶
۳-۴	Debugging	۳۶
۴-۴	تنظیمات GPIO	۳۷
۵-۴	تنظیمات RTC	۳۸
۶-۴	تنظیمات دو تایمر	۳۹
۷-۴	تنظیمات تایمر شش	۳۹
۸-۴	تنظیمات تایمر چهارده	۴۰
۹-۴	تنظیمات تایمر پانزده	۴۰
۱۰-۴	تنظیمات تایمر شانزده	۴۱
۱۱-۴	تنظیمات تایمر هفده	۴۱
۱۲-۴	تنظیمات مبدل آنالوگ به دیجیتال	۴۲
۱۳-۴	I2C	۴۳
۱۴-۴	USART	۴۳
۱۵-۴	حالت‌های ماشین حالت	۴۴
۱۶-۴	صفحه‌ی در انتظار تایید	۴۵
۱۷-۴	صفحه‌ی خوش‌آمد گویی	۴۶
۱۸-۴	صفحه‌ی اصلی	۴۶
۱۹-۴	صفحه‌ی حسگر سامت	۴۷
۲۰-۴	صفحه‌ی حسگر حرکتی	۴۸
۲۱-۴	صفحه‌ی درباره‌ی ما	۴۹
۲۲-۴	صفحه‌ی آلام	۴۹

۵۰	۲۳-۴ نمونه کدی از بخش وقفه‌ی خارجی
۵۱	۲۴-۴ نمونه کدی از بخش ذخیره‌ی داده‌ی حسگر سلامت
۵۲	۲۵-۴ نمونه کدی از بخش وقفه‌ی کالمن
۵۲	۲۶-۴ کد روتین وقفه‌ی تایمر صفحه نمایش
۵۳	۲۷-۴ کد روتین وقفه‌ی DMA مربوطه به ADC
۵۳	۲۸-۴ لایه‌های نرم‌افزاری
۵۴	۲۹-۴ توابع کتابخانه‌ی xfer
۵۴	۳۰-۴ بخشی از کتابخانه‌ی fonts
۵۵	۳۱-۴ بخشی از کتابخانه‌ی notes
۵۵	۳۲-۴ پروتوتایپ توابع کتابخانه‌ی MAX30102
۵۶	۳۳-۴ پروتوتایپ توابع کتابخانه‌ی MPU6050
۵۶	۳۴-۴ نت‌های «ای ایران»
۵۷	۳۵-۴ پروتوتایپ توابع کتابخانه‌ی SSD1306
۵۷	۳۶-۴ پروتوتایپ توابع کتابخانه‌ی kalman
۵۷	۳۷-۴ پروتوتایپ توابع کتابخانه‌ی pages
۵۸	۳۸-۴ تصاویر مربوطه به کتابخانه‌ی system
۵۹	۳۹-۴ بلوک دیاگرام زیر سیستم تشخیص حرکت
۵۹	۴۰-۴ خروجی حسگر حرکتی. سه محور مربوط به شتاب خطی و سه محور مربوط به سرعت زاویه‌ای
۶۳	۴۱-۴ تصویر معادلات غیر ماتریسی کالمن به زبان C برای سیستم حرکت دورانی
۶۴	۴۲-۴ هیستوگرام داده‌های حسگر حرکتی در حال سکون
۶۵	۴۳-۴ خروجی خام شتاب خطی در مقایسه با مقدار فیلتر شده در هر سه محور
۶۶	۴۴-۴ اجرای فیلتر کالمن روی ریزپردازنده و نتایج زمان-واقعی آن

فصل اول

مقدمه

۱-۱ زیربخش مقدمه

دنتردتشی برتنش ر قرنس رند ردن رتندبقدتلتمشنقلد

فصل دوم

سخت افزار و الکترونیک

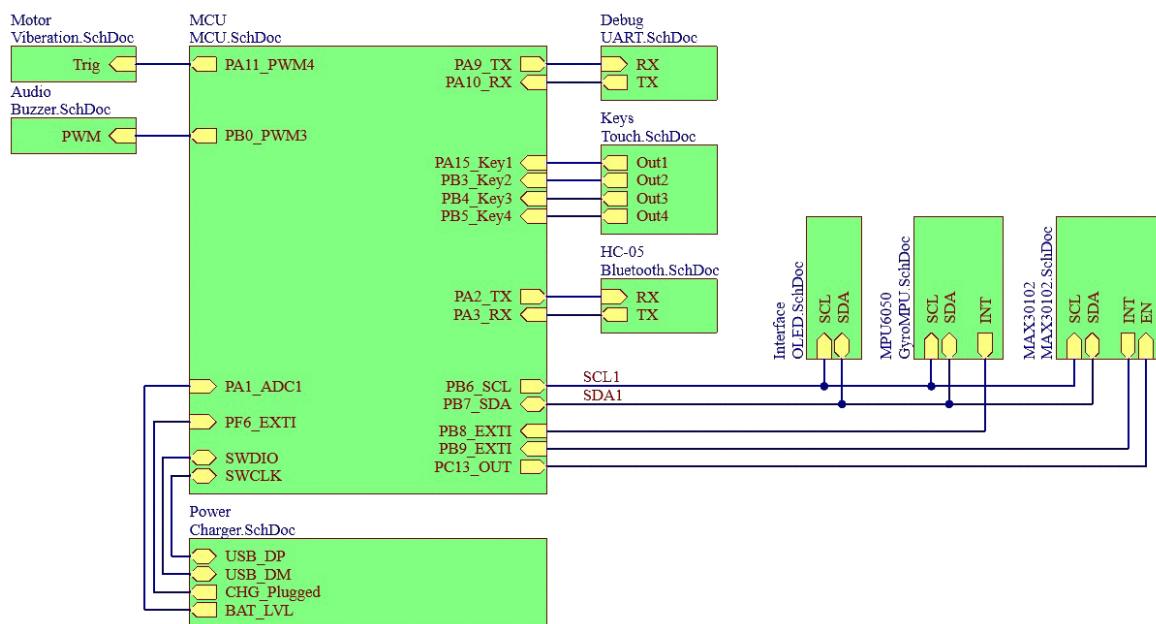
همانطور که گفته شد، این پروژه شامل سه قسمت اصلی سخت افزار، مکانیک و نرم افزار است. در این فصل، به تشریح سخت افزار می پردازیم.

۱-۲ کلیت شماتیک

شکل ۱-۲ کلیت شماتیک و بلوک دیاگرام آن را نشان می دهد. هر کدام از زیربخش ها به تفصیل در ادامه بررسی شده اند.

این زیربخش ها شامل موارد ذیل هستند:

- ۷. صفحه نمایش
- ۸. تغذیه و مدیریت توان
- ۹. کلیدهای لمسی
- ۱۰. بازر
- ۱۱. موتور ایجاد لرزش
- ۱. فیبر مدار چاپی
- ۲. هسته پردازشی
- ۳. درگاه بلوتوث
- ۴. درگاه ارتباط سریال
- ۵. حسگر PPG
- ۶. حسگر حرکتی



شکل ۱-۲: شماتیک کلی ساعت و نحوه اتصال بخش های مختلف به یکدیگر

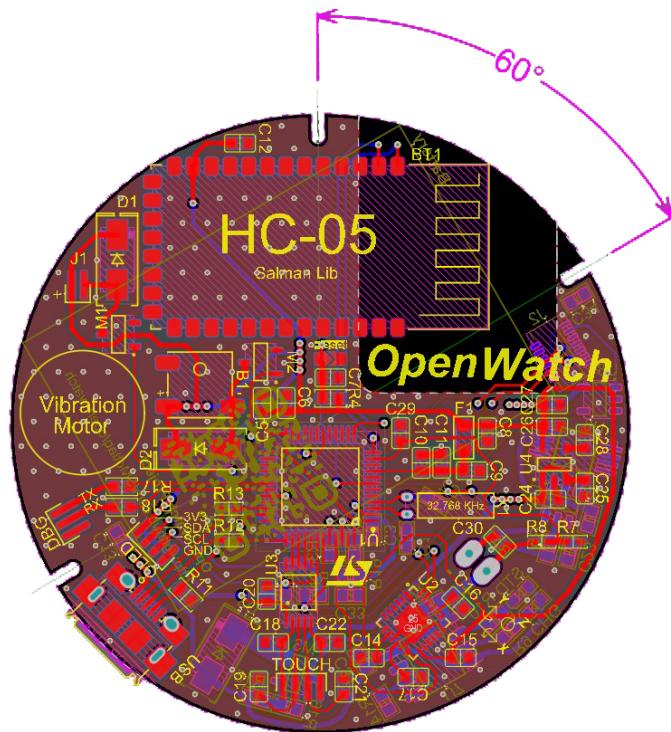
۲-۲ فیبر مدار چاپی

فیبر مدار چاپی یا PCB^۱ صفحه‌ای است معمولاً از جنس فیبر FR-4 که با دو لایه‌ی نازک مس (معمولاً به ضخامت ۳۵ میکرون) در طرفيین پوشیده است. طرحی که طراح به کارخانه‌ی چاپ پی‌سی‌بی ارسال می‌کند روی این ورقه‌ها پياده می‌شود. سپس لایه‌ی محافظ معمولاً سizer زنگ به نام Solder mask روی آن اضافه می‌شود که برای زیبایی بخشی به کار و محافظت از مس در مقابل خوردگی و اکسایش است.

در اين پروژه از يك پی‌سی‌بی چهارلایه استفاده شده است. به دليل فشردگی بالاي طرح و قطعات، همچنین برای بهبود كيفيت سيگنال‌ها و كاهش اثر نویز، دو صفحه‌ی زمين در لایه‌های ۲ و ۳ تعبيه شده است. اين صفحه‌ها با كوتاه کردن مسیر جريان برگشتی باعث بهبود كيفيت سيگنال و كاهش اثر نویز می‌شوند. همچنین تأثير چشم‌گيری در سهولت مسیرکشی پی‌سی‌بی دارند.

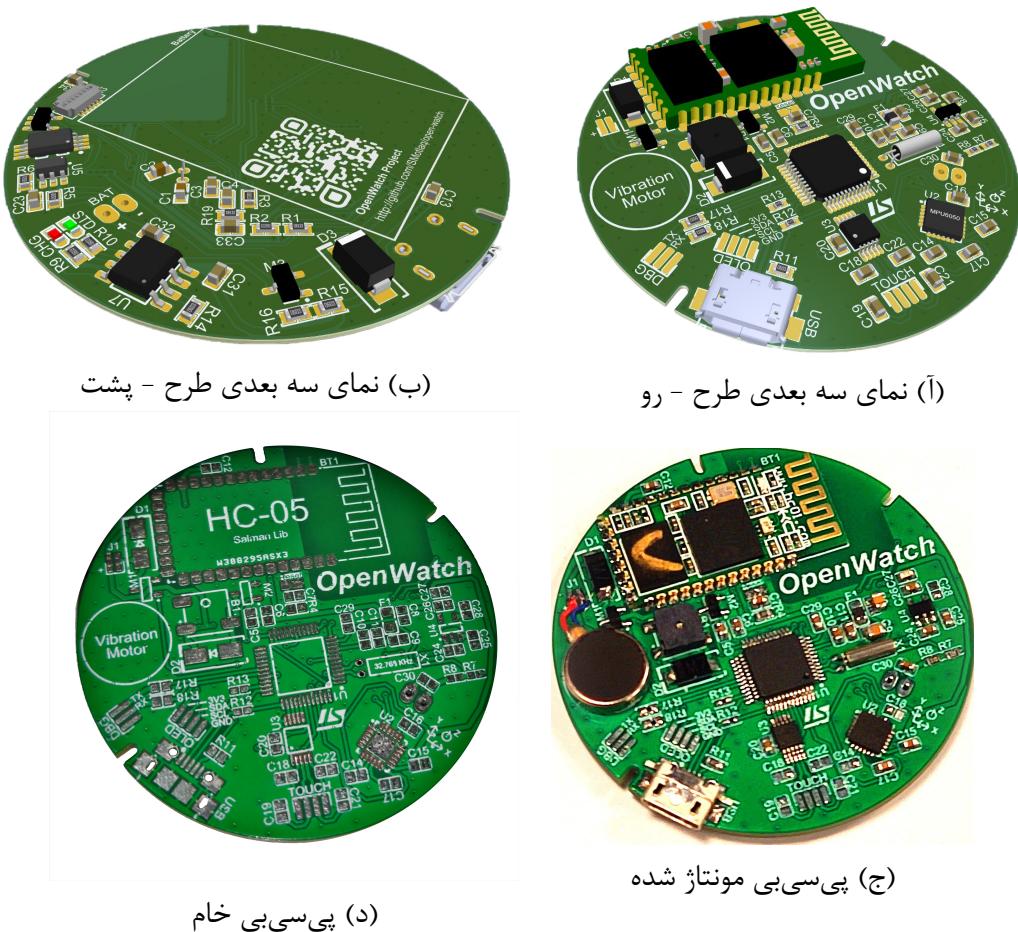
پی‌سی‌بی های این پروژه - به رایگان - توسط شرکت PCBWay^[۴] چاپ شده است که از بزرگترین و مجهرترین کارخانه‌های چاپ پی‌سی‌بی در کشور چین است.

شکل ۲-۲ تصویر پی‌سی‌بی طراحی شده در نرمافزار Altium Designer را نشان می‌دهد. تصاویر مربوط به پی‌سی‌بی در شکل ۲-۳ قابل مشاهده است.



شکل ۲-۲: پی‌سی‌بی طراحی شده در نرمافزار Altium Designer

Printed Circiut Board^۱



شکل ۳-۲: تصاویری از پی‌سی‌بی پروژه

۳-۲ هسته‌ی پردازشی

برای انتخاب پردازنده‌ی مناسب باید موارد ذیل را مدنظر داشت:

۱. مقدار حافظه‌ی فلاش^۲:

برنامه‌ای که برای پردازنده نوشته می‌شود در حافظه‌ی فلاش ذخیره می‌شود. پس این حافظه مشخص می‌کند چه حجمی از برنامه در این پردازنده جا می‌شود.

۲. مقدار حافظه‌ی رم^۳:

این حافظه، یک حافظه‌ی موقت است که متغیرها، اشاره‌گرهای^۴، نقطه‌ی بازگشت توابع و مقادیر ثبات‌ها^۵ به طور موقت در آن نوشته می‌شود. مقدار رم موردنیاز باید با توجه به حجم متغیرها و پیچیدگی عملیاتی و محاسباتی برنامه تعیین شود.

Flash^۲

RAM^۳

Pointers^۴

Registers^۵

۳. تعداد پایه ها و مدارهای واسط^۵:

پردازنده های مختلف تنوع زیادی در نوع و تعداد مدارهای واسط ارائه می دهند. با توجه به تعداد سخت افزارهای جانبی، باید تعداد پایه و نوع مدارهای واسط موردنیاز تعیین شود.

۴. پکیج:

پکیج های مختلف نمایانگر شکل ظاهری پردازنده است. برخی پکیج ها ابعاد بزرگی دارند و برخی دیگر به قدری کوچک هستند که پایه های پردازنده در زیر تراشه تعییه می شوند تا فضای کمتری اشغال کند. در انتخاب پکیج باید محدودیت فضای پی سی بی را مدنظر قرار داد.

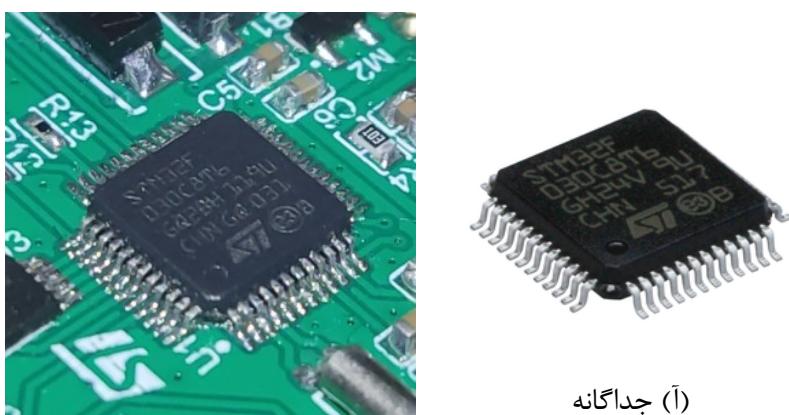
۵. موجودی بازار:

یکی از مهمترین چالش های مهندسان الکترونیک در ایران، موجودی بازار است. خیلی از قطعاتی که طراح به آن ها نیاز دارد در بازار ایران پیدا نمی شود یا قیمت بالایی دارد. یا باید به وارد کردن قطعه و تاخیر چند ماهه تن داد یا باید طرح را عوض کرد تا با قطعات موجود در بازار قابل پیاده سازی باشد.

۶. قیمت:

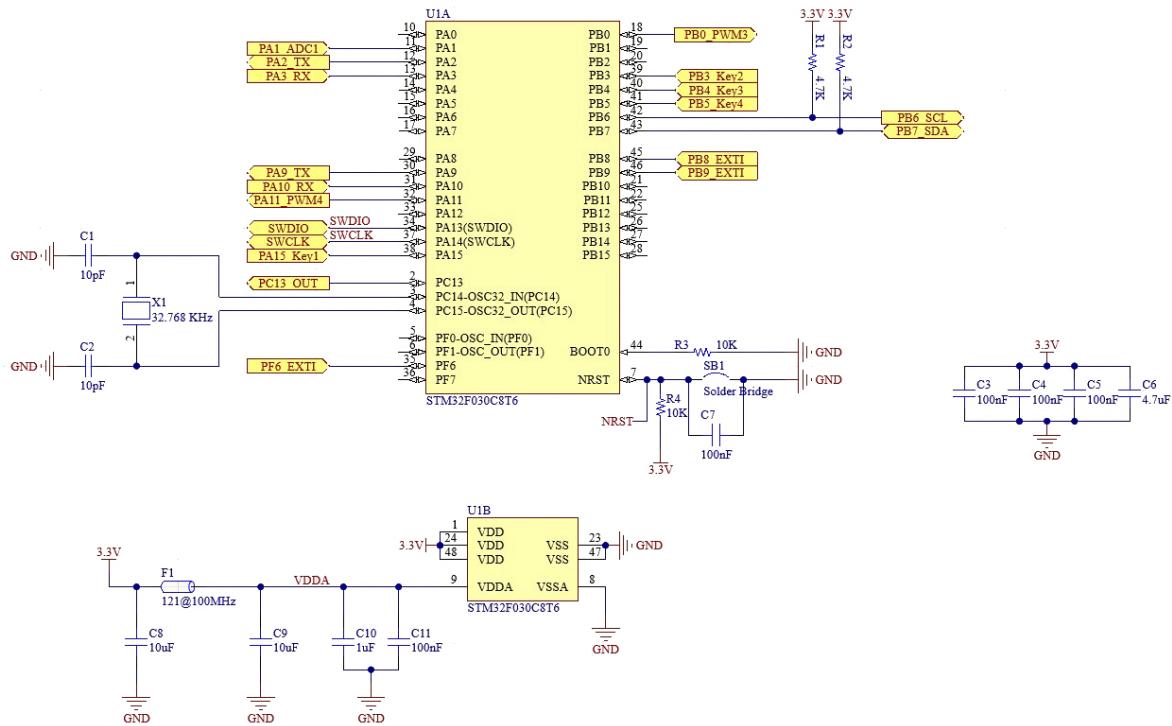
بدیهی است که یکی از قیود طراحی، قیمت تمام شده است. طراح باید در انتخاب پردازنده طوری عمل کند که با کمترین قیمت، بهترین تطابق را با قیود بالا ایجاد کند.

در نهایت با بررسی موارد فوق، پردازنده انتخاب شده در این پروژه STM32F030C8 است. این پردازنده محصول شرکت ST که هسته‌ی ARM Cortex-M0 ۳۲ بیتی دارد. شکل ۴-۲ آ تصویر واقعی این پردازنده و شکل ۴-۲ ب تصویر آن را بر روی پی سی بی ساعت نشان می دهد.



(ب) مونتاژ شده روی برد پروژه

شکل ۴-۲: تصاویری از پردازنده STM32F030



شکل ۲-۵: شماتیک مربوط به بخش ریز پردازنده

شکل ۲-۵ شماتیک مداری بخش پردازنده را نشان می‌دهد. یک کریستال ۳۲۷۶۸ هرتزی وظیفهٔ تنظیم فرکانس بخش RTC^۸ را بر عهده دارد. ساعت سیستم توسط این واحد ذخیره و تنظیم می‌شود. تغذیهٔ بخش ADC^۹ توسط چند خازن و یک فریت بید^{۱۰} فیلتر شده است. تغذیهٔ خود پردازنده نیز توسط ۴ خازن (مطابق با دستور کارخانه سازنده) فیلتر شده است.

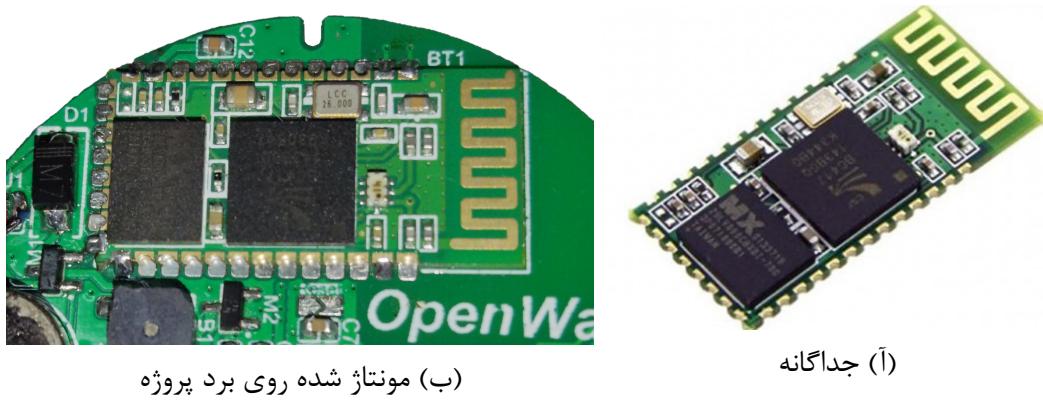
۴-۲ درگاه بلوتوث

ارتباط ساعت با تلفن همراه از طریق درگاه بلوتوث^{۱۱} است. قیود انتخاب بلوتوث هم تا حدی مشابه قیود انتخاب پردازنده (ابتدا بخش ۳-۲) است. با در نظر گرفتن شرایط بازار، قیمت و عملکرد مژوپهای مختلف، نهایتاً مژوال HC-05 تصویر این مژوال و شکل ۲-۶-ع^{۱۲} تصویر آن را بر روی پی‌سی‌بی ساعت نشان می‌دهد.

یکی از نکات مهمی که در طراحی پی‌سی‌بی برای مژوالهای مخابراتی وجود دارد این است که در نزدیکی آنتن این مژوال‌ها نباید هادی جریان الکتریکی وجود داشته باشد. در غیر این صورت خاصیت خازنی بین آنتن و این هادی باعث تغییر مشخصه‌های آنتن می‌شود و باعث اختلال در عملکرد آنتن

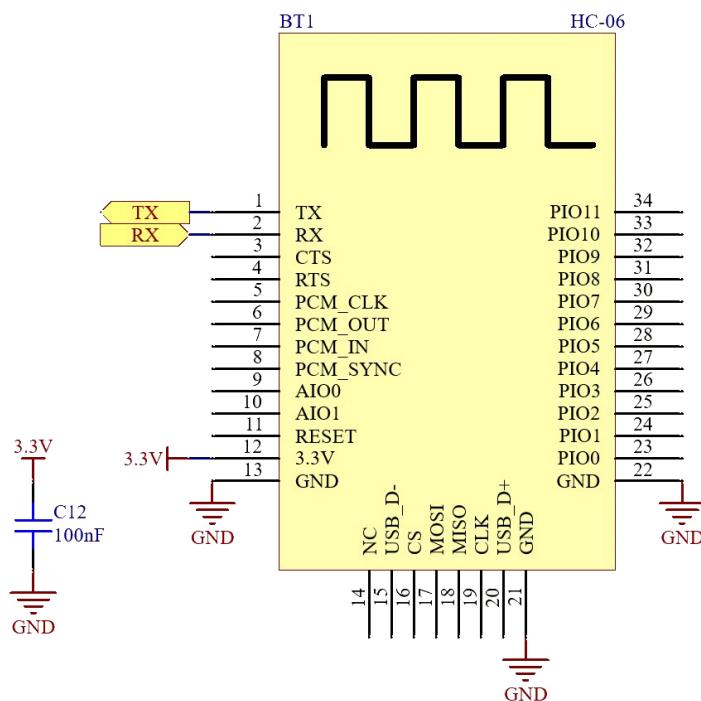
Real Time Clock^۸
Analog to Digital Converter^۹
Ferrite Bead^{۱۰}
Bluetooth^{۱۱}

می گردد. همانطور که در شکل ۲-۲ مشاهده می شود، مس های اطراف آنتن حذف شده اند تا عملکرد بلوتوث دچار مشکل نشود.



شکل ۲-۶: تصاویری از مازول بلوتوث HC-05

شکل ۷-۲ شماتیک مداری بخش بلوتوث را نشان می دهد. پروتکل ارتباطی این مازول با پردازنده پروتکل UART^{۱۲} است که با دو پین RX و TX به پردازنده متصل می شود. پریفرال UART2 در پردازنده به ارتباط با بلوتوث اختصاص دارد. تغذیه مازول نیز با یک خازن ۱۰۰ نانوفارادی فیلتر شده است.



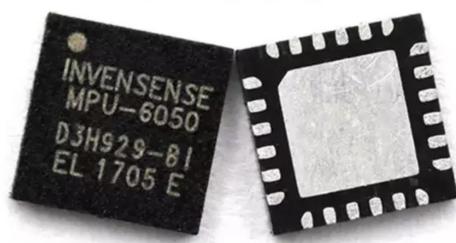
شکل ۷-۲: شماتیک مربوط به بخش بلوتوث

۵-۲ حسگر شتاب خطی و سرعت زاویه‌ای

برای اندازه‌گیری مشخصه‌های حرکتی باید سراغ IMU^{۱۳}ها رفت. IMU‌ها وسیله‌های الکترونیکی هستند که با استفاده‌ی ترکیبی از شتاب‌سنج‌ها، ژیروسکوپ‌ها و گاهی اوقات مغناطیس‌سنج‌ها، مشخصه‌های حرکتی را اندازه‌گیری و گزارش می‌کنند [۶]. در این پژوهه از حسگر MPU6050 به این منظور استفاده شده است. این حسگر علی‌رغم قیمت نسبتاً پایین، دقت و سرعت مناسبی دارد. مشخصات فنی این حسگر در ضمیمه ؟ موجود است. شکل ۸-۲ تصویر این حسگر و شکل ۸-۲ آ تصویر آن را بر روی پی‌سی‌بی ساعت نشان می‌دهد.



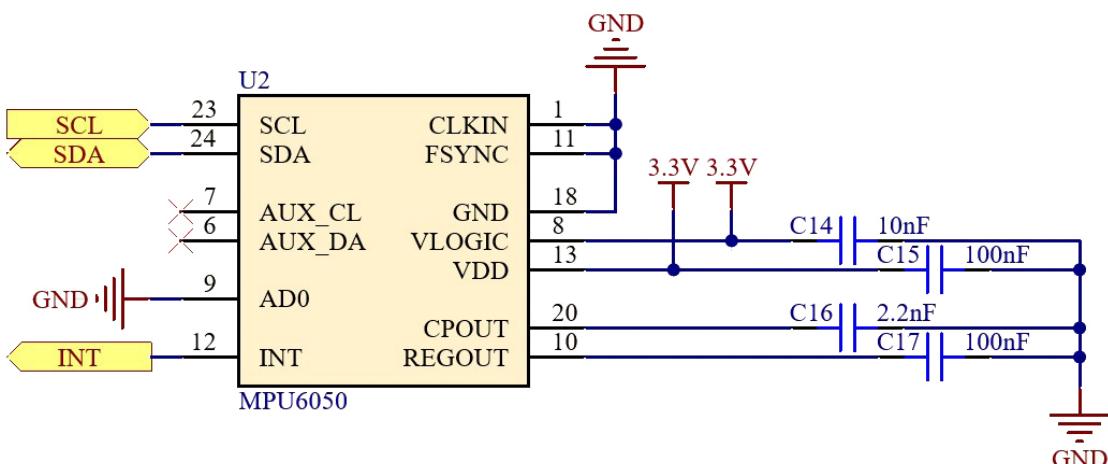
(ب) مونتاژ شده روی برد پروژه



(آ) جداگانه

شکل ۸-۲: تصاویر حسگر حرکتی

شکل ۹-۲ شماتیک مداری حسگر MPU6050 را نشان می‌دهد. خازن‌ها طبق دستور کارخانه به حسگر متصل شده‌اند. درگاه ارتباطی این حسگر، بس I2C است. به همین دلیل I2C پردازنده به این حسگر متصل است.

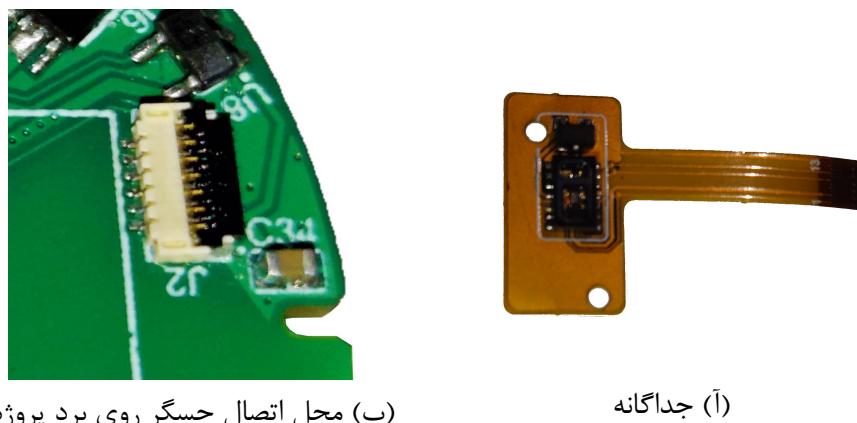


شکل ۹-۲: شماتیک مربوط به بخش حسگر حرکتی

۶-۲ حسگر PPG

در ابتدا ساختار کلی حسگرهای PPG^{۱۴} را بررسی می‌کنیم. این حسگرها شامل یک یا چند دیود نشرده‌نده‌ی نور^{۱۵} هستند که به همراه گیرنده‌ی نوری در یک بسته قرار دارند. وظیفه‌ی این حسگرها این است که نوری را به داخل بفت بدن بفرستند و بازتاب آن را دریافت کنند. با پردازش سیگنال دریافتی می‌توان به اطلاعاتی مانند ضربان قلب و سطح اکسیژن خون دست یافت.

بسته به کاربرد موردنظر، نورهای مختلفی در این سنسورها استقاده می‌شود. حسگر استفاده شده در این پروژه MAX30102 نام دارد. این حسگر دارای دو نور قرمز و مادون قرمز است. ساختار منعطف این حسگر کمک می‌کند تا بتوان به سادگی آن را در بدن نصب کرد. متاسفانه این حسگر در بازار ایران موجود نبود و مجبور به وارد کردن آن از چین شدم. تصویر این حسگر در شکل ۲-۱۰^{۱۶} و محل اتصال آن روی پی‌سی‌بی در شکل ۲-۱۰^{۱۷} قابل مشاهده است.



شکل ۲-۱۰: تصاویر حسگر PPG

شکل ۱۱-۲ شماتیک مداری حسگر MAX30102 را نشان می‌دهد. همانطور که دیده می‌شود یک آیسی RT9742GGJ5 بر سر راه تغذیه‌ی سنسور قرار دارد. این آیسی یک سوییچ ماسفت^{۱۸} است که علاوه بر قطع و وصل تغذیه، مصرف کننده را در مقابل مواردی چون کمبود ولتاژ، اضافه جریان و افزایش دما محافظت می‌کند. از آنجا که نیازی نیست این حسگر دائم روشن باشد و نمونه برداری کند، این سوییچ کمک می‌کند تا بتوان حسگر را روشن یا خاموش کرد. کانکتوری که برای حسگر استفاده شده یک کانکتور FPC^{۱۹} ۱۳ پین است که متاسفانه آن هم در بازار ایران موجود نبود و وارد شد. درگاه ارتباطی این حسگر، بس^{۲۰} I2C^{۲۱} است. به همین دلیل پردازنده به این حسگر متصل است.

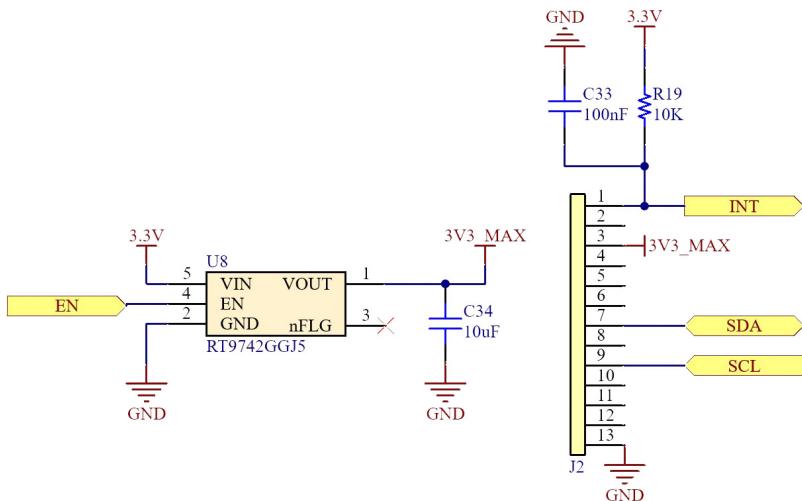
^{۱۴} Photoplethysmogram یا تغییر حجم سنجی نوری

^{۱۵} Light-emitting diode (LED)

^{۱۶} Metal–Oxide–Semiconductor Field-Effect Transistor

^{۱۷} Bus

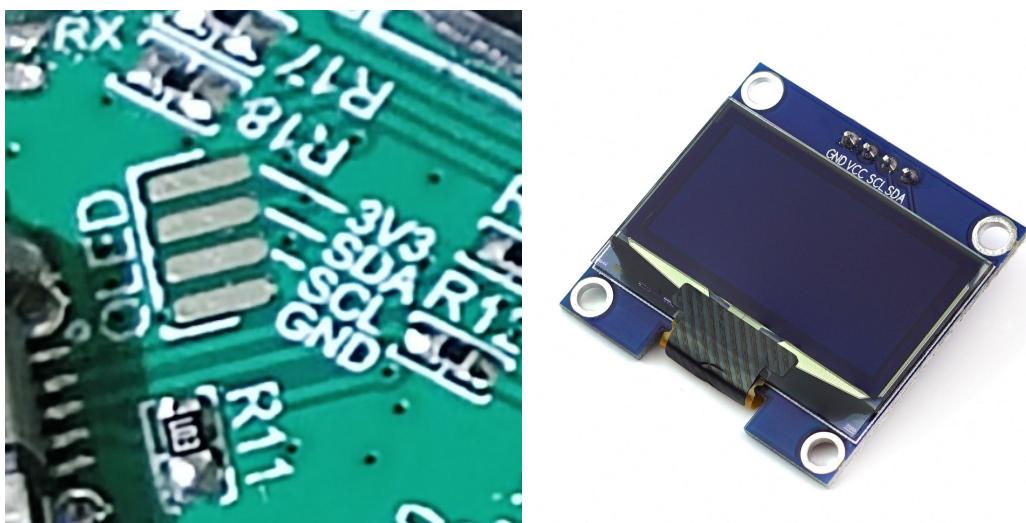
^{۱۸} Inter-Integrated Circuit



شکل ۱۱-۲: شماتیک مربوط به بخش حسگر PPG

۷-۲ صفحه نمایش

صفحه‌ی نمایش از اصلی‌ترین قسمت‌های یک ساعت هوشمند است. صفحه‌ی نمایش باید مصرف کمی داشته باشد، راهاندازی آن دشوار نباشد و تعداد پایه‌ی زیادی هم نیاز نداشته باشد تا بتوان آن را در ساعت استفاده کرد. از این رو یک صفحه‌ی نمایش^{۱۹} OLED ۱۳۰۶ SSD1306 راهاندازی می‌شود. شکل ۱۲-۲ تصویر این نمایشگر و شکل ۱۲-۲ ب تصویر محل اتصال سیم‌های آن را بر روی پی‌سی‌بی ساعت نشان می‌دهد.



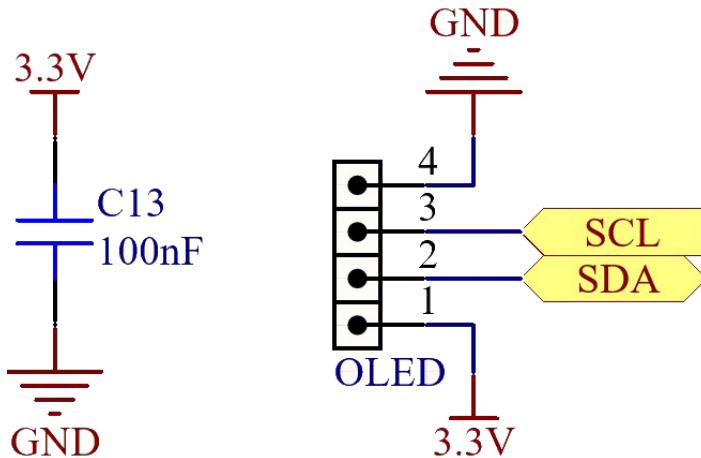
(ب) محل اتصال نمایشگر روی برد پروژه

(آ) جدایگانه

شکل ۱۲-۲: تصاویر صفحه‌ی نمایش

^{۱۹} صفحه‌ی نمایش‌های OLED این‌گونه هستند که فقط پیکسل‌های موردنیاز روشن می‌شوند و دیگر نیازی نیست که نور پشتی به تمام پیکسل‌ها بتابد تا دیده شوند. این نوع نمایشگرها مصرف کمتر و زیباتری بیشتری دارند.

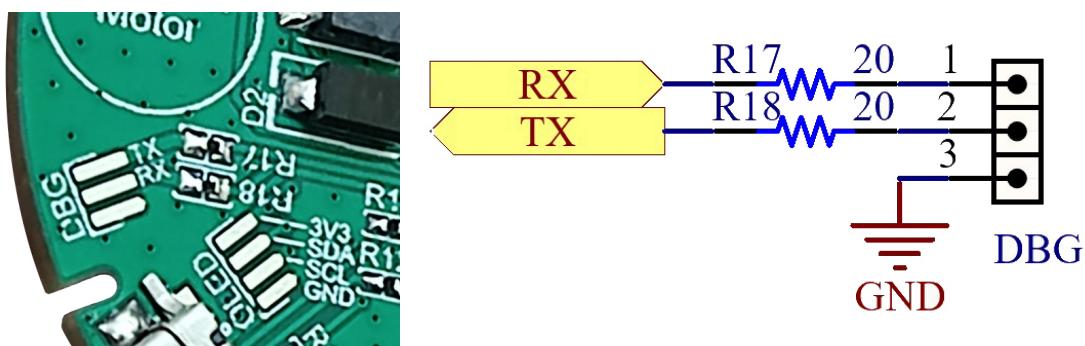
شکل ۲-۱۳: شماتیک مداری بخش نمایشگر را نشان می‌دهد. درگاه ارتباطی این نمایشگر، بس I2C است. به همین دلیل I2C1 پردازنده به این نمایشگر متصل است.



شکل ۲-۱۳: شماتیک مربوط به بخش نمایشگر

۸-۲ درگاه ارتباط سریال

پریفرال UART1 در پردازنده با دو پین RX و TX از روی پی‌سی‌بی خارج شده‌اند. کاربرد این دو پین اشکال یابی^{۲۰} و ارتباط سرعت بالا بین ساعت و رایانه است. این بخش کاربردی در عملکرد کلی ساعت ندارد و صرفاً روند توسعه را تسريع می‌کند. شکل ۲-۱۴: شماتیک مداری و شکل ۲-۱۴: تصویر واقعی این دو پین را نشان می‌دهد.



(ب) پین‌های مربوطه

(آ) شماتیک

شکل ۲-۱۴: تصاویر مربوط به درگاه ارتباط سریال

۹-۲ تغذیه و مدیریت توان

بخش تغذیه و مدیریت توان این ساعت هوشمند از دو قسمت تشکیل شده است. بخش تغذیه که شامل باتری است و بخش شارژ که وظیفه‌ی شارژ باتری و تغذیه‌ی مدار را در صورت اتصال شارژر بر عهده دارد.

۱-۹-۲ باتری

برای انتخاب باتری قیودی چون ظرفیت، ولتاژ، ابعاد و قیمت مطرح است. باتری‌ای برای این پروژه مناسب است که در حداقل ابعاد، حداقل ظرفیت را داشته باشد، در بازار موجود باشد، قیمت مناسبی داشته باشد و یک سلول باشد (ولتاژ ۳.۷ ولت). با این تفاسیر باتری‌ای که در شکل ۱۵-۲ مشاهده می‌شود برای این پروژه انتخاب شد. یک باتری لیتیوم-یون یک سلولی که ولتاژ نامی آن ۳.۷ ولت است با ظرفیت ۴۰۰ میلی‌آمپرساعت.



شکل ۱۵-۲: باتری انتخاب شده برای پروژه

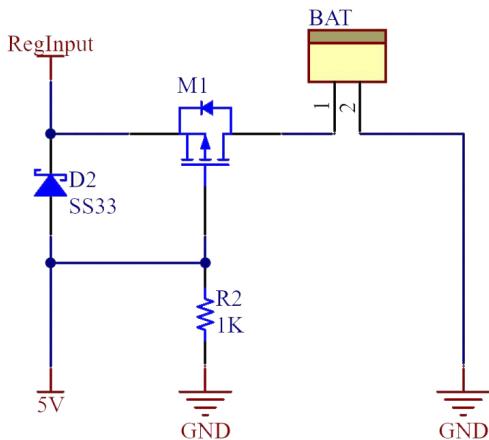
۲-۹-۲ شارژ و مدیریت توان

برای شارژ باتری از یک آیسی شارژ به نام TP4056 استفاده شده است که می‌تواند باتری‌های لیتیومی را شارژ کند. در کنار آن یک آیسی کنترلر به نام DW01 قرار دارد که ولتاژ باتری را پایش می‌کند. باتری‌های یک سلولی ولتاژ نامی ۳.۷ ولت دارند. اما در حالت شارژ کامل حدود ۴.۲ و در حالت خالی حدود ۳ ولت هستند. این کنترلر وظیفه دارد در صورت تجاوز ولتاژ باتری از این محدوده، آن را به کمک دو

ماضت از مدار خارج کند. اینگونه با تری هیچگاه دچار اضافه ولتاژ^{۲۱} یا کمبود ولتاژ^{۲۲} نمی‌شود و آسیبی نمی‌بیند. این مدارات در شکل ۱۷-۲ قابل مشاهده‌اند.

تغذیه‌ی مدارهای ساعت از یک رگولاتور ۳.۰۳ ولت تأمین می‌شود. بحث مدیریت توان به این مسئله می‌پردازد که ورودی این رگولاتور از کدام مسیر تغذیه شود. هنگامی که با تری شارژ دارد و شارژر متصل نیست، باید ورودی با تری مستقیماً به رگولاتور وارد شود. هنگامی که شارژر متصل می‌شود، باید ورودی برق شارژر مستقیماً به رگولاتور برود و با تری از مدار خارج شود که از طریق برق ورودی شارژ شود. این تغییر مسیر و جابجایی توسط یک ماضت AO3401 صورت می‌گیرد.

این ماضت یک ماضت P-Channel است. در صورتی هدایت می‌کند که شرط $V_{GS} < V_{th} = -0.9$ برقرار باشد. حال به شکل ۱۶-۲ توجه کنید. در صورتی که برق ورودی (۵ ولت) متصل باشد، ولتاژ ۵ ولت می‌شود. از طرفی ورودی رگولاتور از طریق دیود D2 که شاتکی است و 0.2 ولت افت دارد، متصل شده است. در این صورت $V_{GS} = 0.2$ ولت است و باعث می‌شود که ماضت قطع شود. در این حالت ولتاژ سورس حدود ۴.۸ است. از آنجا که ولتاژ با تری نهایتاً ۴.۳ ولت است لذا دیود داخلی ماضت نیز هدایت نمی‌کند و همه چیز درست است. حال اگر تغذیه‌ی خارجی قطع شود، گیت ماضت به دلیل وجود مقاومت Pull-Down صفر می‌شود. اکنون کافی است تا ولتاژ سورس اندکی مثبت شود (حدود ۰.۹ ولت) تا ماضت هدایت کند. این ولتاژ از طریق دیود داخلی ماضت از سمت با تری ایجاد می‌شود. حالا ماضت روشن است و ورودی رگولاتور از طریق با تری تغذیه می‌شود.



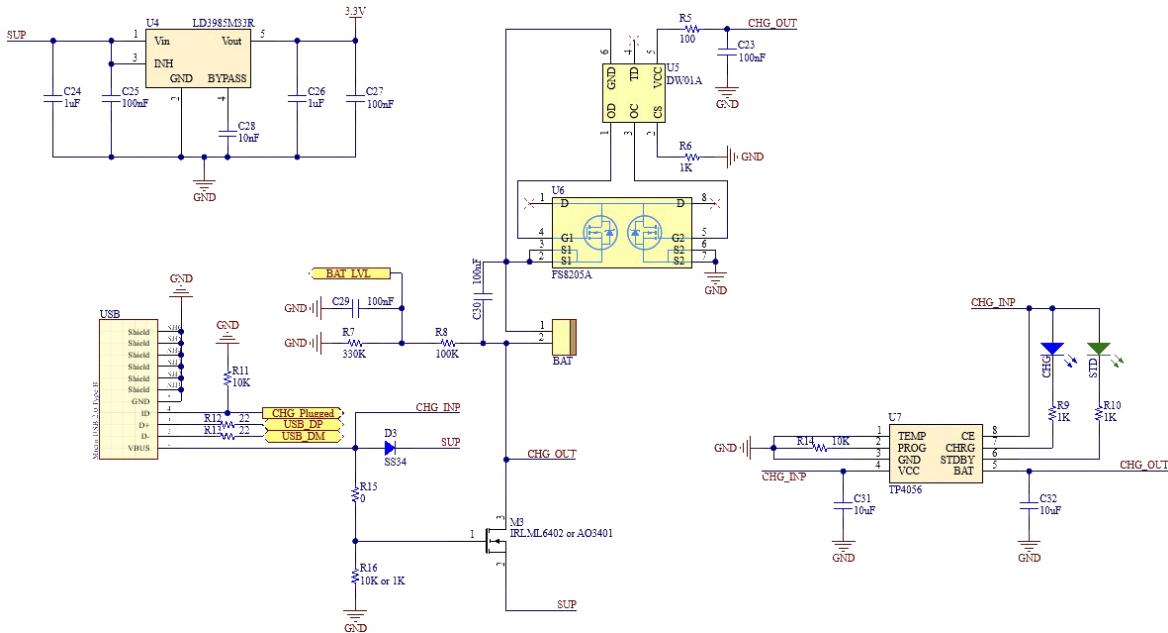
شکل ۱۶-۲: شماتیکی ساده برای توضیح بخش مدیریت توان

به طور خلاصه می‌توان حالات کاری مدار را اینگونه توضیح داد:

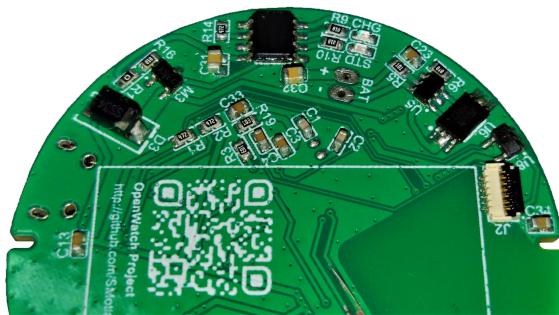
۱. شارژر متصل است: ماضت خاموش است، با تری از مدار خارج شده و به شارژر متصل است
۲. شارژر متصل نیست: ماضت روشن است، با تری در مدار است و به شارژر متصل نیست

Over voltage^{۲۱}

Under voltage^{۲۲}



(ا) شماتیک مدار شارژ و مدیریت توان



(ب) مونتاژ شده روی برد پروژه به همراه جای باتری

شکل ۱۷-۲: تصاویر بخش تغذیه

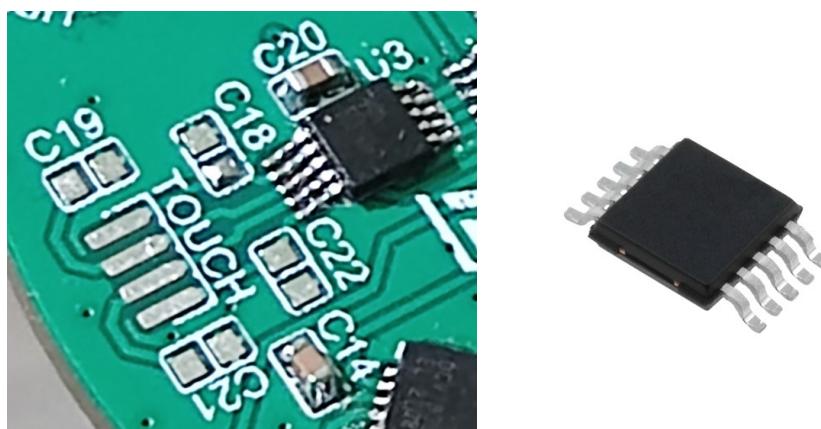
دو عدد LED به رنگ‌های قرمز و آبی به آیسی شارژ متصلند که در صورت وصل شدن ساعت به شارژر روشن می‌شوند. قرمز برای هنگامی است که باتری در حال شارژ است، آبی برای هنگامی که باتری به طور کامل شارژ شده است.

برای قرائت مقدار شارژ باتری، از یک تقسیم مقاومتی با نسبت سه چهارم (به صورت دقیق 0.76°) استفاده شده است که ولتاژ حداکثر ۴.۳ ولتی باتری را به ۳.۳ ولت می‌رساند که توسط مبدل آنالوگ به دیجیتال ^{۲۳} قابل خواندن است.

^{۲۳} ADC (Analog to Digital Converter)

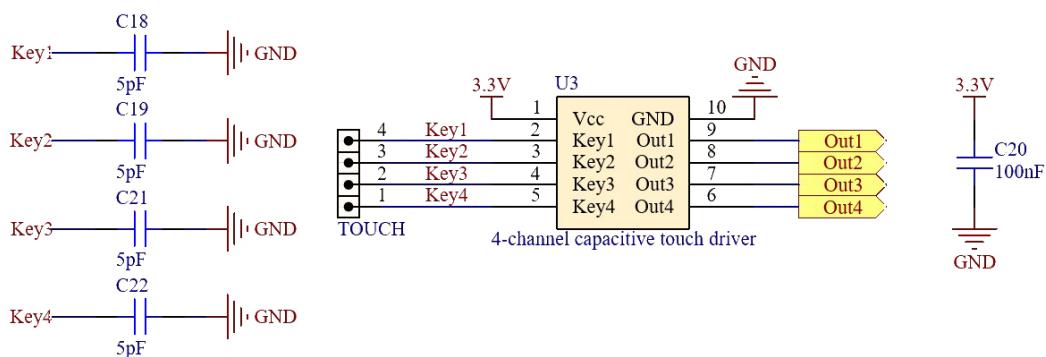
۱۰-۲ کلیدهای لمسی

بر روی بدنی ساعت و زیر صفحه ای اصلی، چهار سیم پیج تعییه شده است که باید بتوان با لمس آنها، با ساعت تعامل کرد و به آن ورودی داد. برای راه اندازی این کلیدها از یک آیسی به نام BS814A-1 استفاده شده است. این آیسی ساخت شرکت Holtek است و یکی از بهترین گزینه ها برای راه اندازی کلید لمسی است. شکل ۱۸-۲ تصویر این آیسی و شکل ۱۸-۳ تصویر آن را بر روی پی سی بی ساعت نشان می دهد.



شکل ۱۸-۲: تصاویر بخش کلیدهای لمسی

شکل ۱۹-۲ شماتیک مداری بخش کلیدهای لمسی را نشان می دهد. این آیسی ده پایه است. دو پایه برای تغذیه دارد، ۴ پایه برای اتصال به کلیدها و ۴ پایه برای اتصال به پردازنده. برای تنظیم حساسیت کلیدها می توان از خازن هایی موازی کلیدها بهره برد. در اینجا خازنی روی برد قرار نگرفته است زیرا نیازی به کاهش حساسیت نبود.

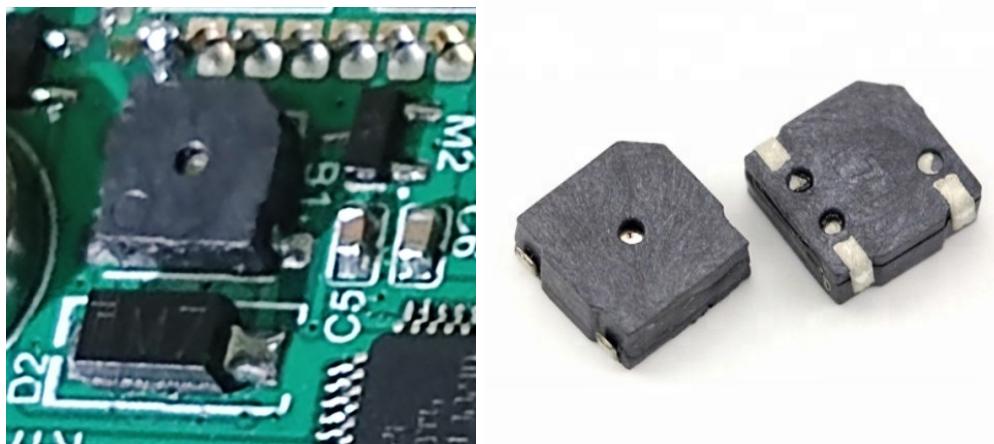


شکل ۱۹-۲: شماتیک مربوط به بخش کلیدهای لمسی

۱۱-۲ بازr

برای ایجاد صدا و هشدارهای صوتی در ساعت، از یک بازr غیرفعال^{۲۴} استفاده شده است. بازرهای پسیو، بازرهایی هستند که نوسان ساز^{۲۵} داخلی ندارند و تنها با خاصیت پیزو الکتریک^{۲۶} کار می‌کنند. بدین صورت که اگر ولتاژ به آن اعمال شود، صفحه‌ی آن جابجا می‌شود و با قطع ولتاژ به مکان اولیه باز می‌گردد.

حال اگر این قطع و وصل ولتاژ با فرکانس مشخصی صورت گیرد، بازr نیز صدایی با همان فرکانس تولید می‌کند. بدیهی است که هارمونیک‌های بالاتر نیز در این صدا وجود دارد زیرا موج ورودی به بازr مربعی است؛ اما فرکانس غالب همان فرکانس اصلی موج مربعی خواهد بود. شکل ۲۰-۲ آ تصویر بازr و شکل ۲۰-۲ ب تصویر آن را بر روی پی‌سی‌بی ساعت نشان می‌دهد.



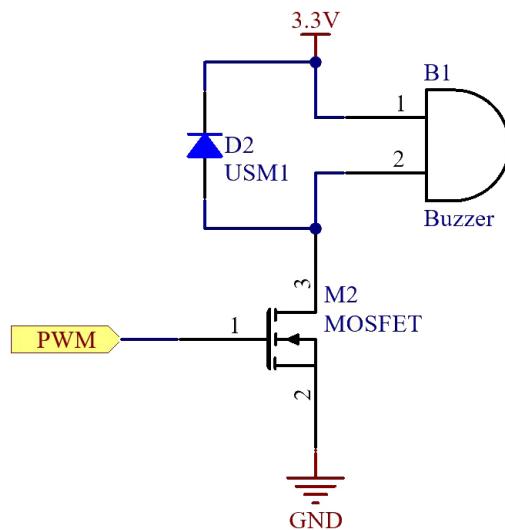
(ب) مونتاژ شده روی برد پروژه

(آ) جداگانه

شکل ۲۰-۲: تصاویر بازr

شکل ۲۱-۲ شماتیک مداری بازr را نشان می‌دهد. برای راه اندازی بازr و تولید صدا، از یک سوییج ماسفت برای قطع و وصل ولتاژ استفاده شده است. دیودی که با بازr موازی شده از ورود جریان برگشتی آن به ماسفت هنگام قطع و وصل ولتاژ جلوگیری می‌کند. برای کنترل فرکانس صدا می‌توان از اعمال موج PWM^{۲۷} به بازr بهره برد. لذا پایه‌ی فرمان این مدار به خروجی PWM تایмер ۳ در پردازنده متصل شده است.

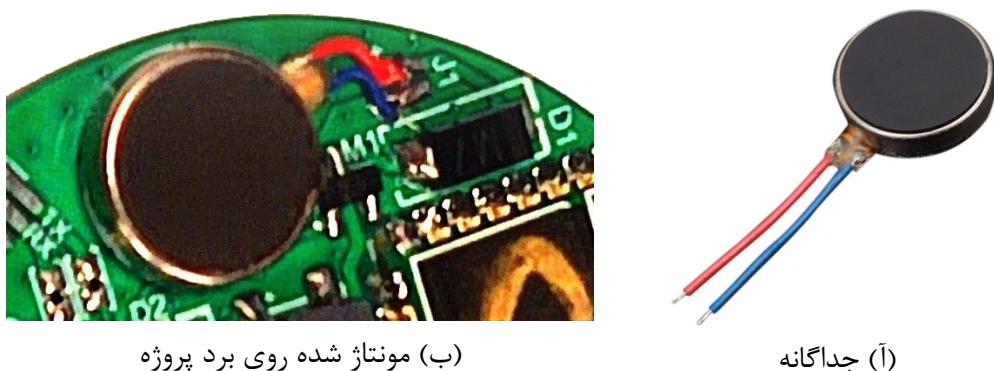
Passive^{۲۴}
Oscillator^{۲۵}
Piezoelectric^{۲۶}
Pulse Width Modulation^{۲۷}



شکل ۲۱-۲: شماتیک مربوط به بخش باز

۱۲-۲ موتور ایجاد لرزش

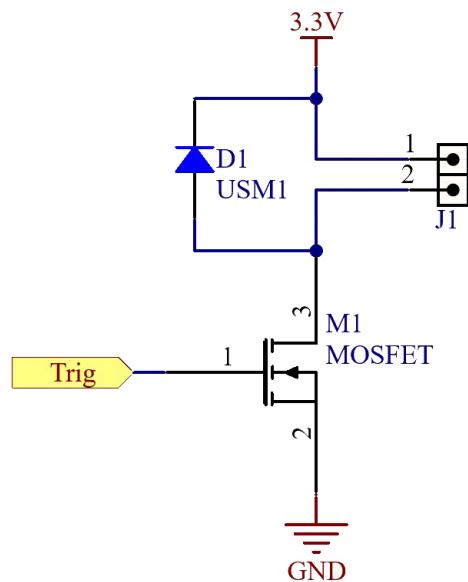
برای ایجاد لرزش ^{۲۸} در ساعت، مشابه تلفن‌های همراه، از یک موتور مخصوص استفاده شده است. موتورهای ایجاد لرزش معمولاً یک موتور DC ساده هستند که یک بار نامتقارن به آن‌ها متصل است. از آنجا که مرکز جرم این بار خارج از شفت موتور است، چرخش آن باعث ایجاد گشتاوری دوران می‌شود که لرزش را ایجاد می‌کند. شکل ۲-۲۶ تصویر این موتور و شکل ۲-۶ب تصویر آن را بر روی پی‌سی‌بی ساعت نشان می‌دهد.



شکل ۲-۲: تصاویر موتور ایجاد لرزش

شکل ۲۳-۲ شماتیک مداری بخش ایجاد لرزش را نشان می‌دهد. این موتور برای کار به 90° میلی‌آمپر جریان الکتریکی احتیاج دارد. طبیعتاً پردازنده نمی‌تواند این جریان را تأمین کند. لذا از یک کلید ماسفت برای قطع و وصل موتور استفاده شده است. دیودی که با موتور موازی شده از ورود جریان برگشتی موتور به ماسفت هنگام خاموش شدن موتور جلوگیری می‌کند. برای کنترل سرعت موتور می‌توان از اعمال موج

PWM به موتور بھرہ بردا. لذا پایه‌ی فرمان این مدار به خروجی PWM تایمر ۱ در پردازنده متصل شده است.



شکل ۲۳-۲: شماتیک مربوط به بخش ایجاد لرزش

فصل سوم

مکانیک و طراحی صنعتی

این فصل به تشریح بدنی ساعت و قسمت‌های مکانیکی آن می‌پردازد. قسمت‌های مکانیکی شامل بدنی اصلی، دریچه‌ی پشتی و نحوه‌ی سرهم شدن قطعات دیگر است.

۱-۳ بدنی اصلی

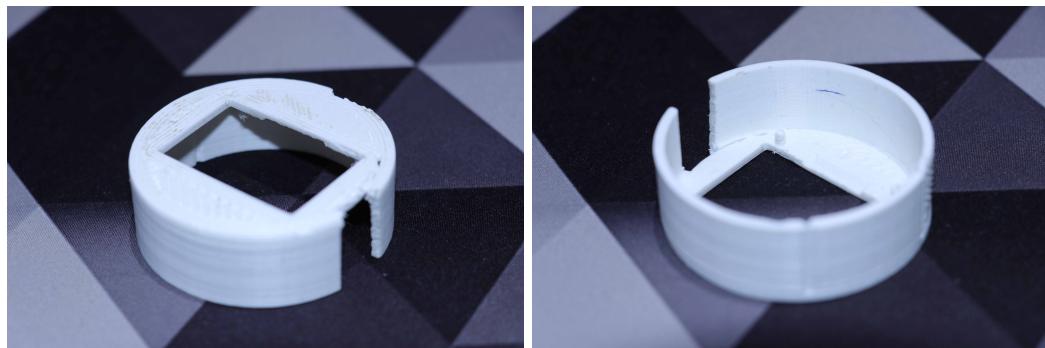
بدنی اصلی به قسمتی اطلاق می‌شود که از بیرون دیده می‌شود و بزرگترین قطعه است. طراحی این قطعه از صفر در نرم‌افزار Solid Works انجام شده است که مطرح‌ترین نرم‌افزار در زمینه‌ی مکانیک و طراحی صنعتی است. برای ساخت بدنی و بخش‌های مکانیکی نیز از فناوری چاپ سه بعدی بهره بردم. بدنی اصلی باید:

۱. محلی برای نصب صفحه نمایش داشته باشد.
۲. بتواند پی‌سی‌بی را درون خود جا دهد و مانع چرخش و جابجایی آن شود.
۳. محلی برای اتصال کابل USB داشته باشد.
۴. محلی برای جریان هوا داشته باشد. زیرا مدار شارژ باعث افزایش دما می‌شود.
۵. محلی برای اتصال بند داشته باشد.
۶. زیبایی بصری داشته باشد
۷. بیش از حد بزرگ نباشد.

برای حصول موارد فوق، چندین نمونه بدنی مختلف طراحی و چاپ شد تا در هر نسخه، بهبودی نسبت به نسخه‌ی قبلی حاصل شود تا هر چه بهتر شروط فوق ارضا شوند.

۱-۳ نسخه‌های اولیه

در ابتدا یک نمونه‌ی اولیه برای تست کلی بدنی و جانمایی پی‌سی‌بی و صفحه نمایش طراحی و ساخته شد. تصویر این نمونه‌ی اولیه را می‌توان در شکل ؟ مشاهده کرد.

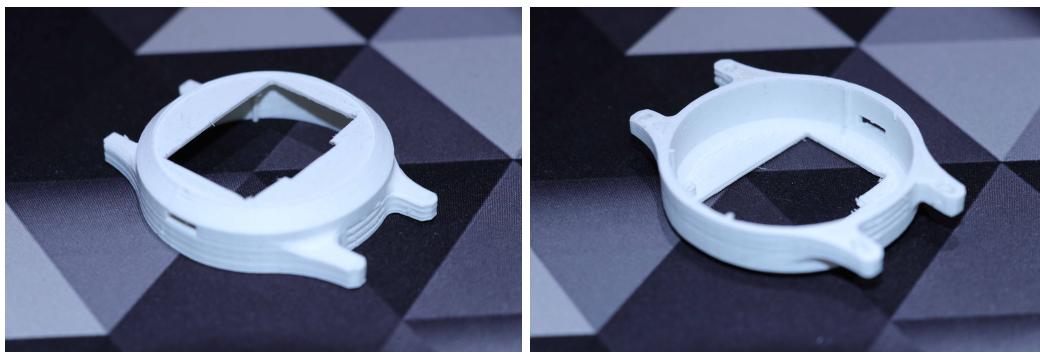


(ب) نمای روی رو

(آ) نمای پشت

شکل ۱-۳: تصاویر بدنی اصلی نسخه‌ی اول

سپس بعد از نهایی شدن طرح کلی، جزئیات طرح تکمیل شد و نسخه‌ی دوم بدن به چاپ رسید. تصویر این نسخه در شکل ؟ دیده می‌شود.

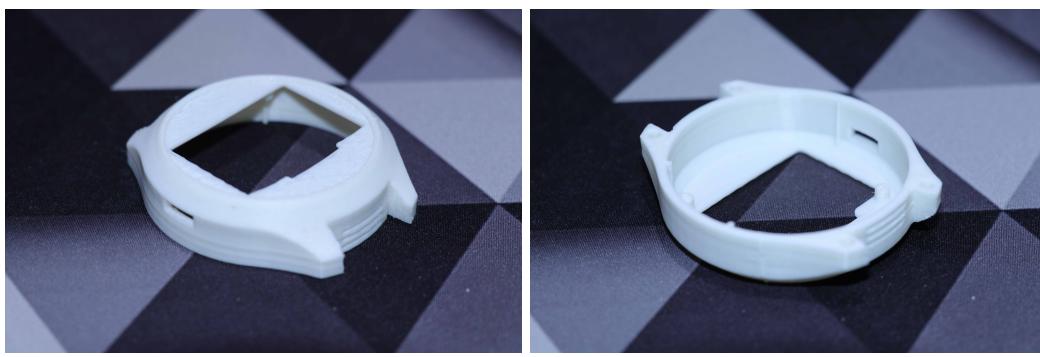


(ب) نمای رو برو

(آ) نمای پشت

شکل ۳-۲: تصاویر بدنی اصلی نسخه‌ی دوم

این نسخه ایراداتی داشت، از جمله اینکه قسمت مربوط به محل اتصال بند بیش از حد بزرگ بود از زیبایی بصری می‌کاهید. بعد از برطرف نمودن ایرادات این نسخه، نسخه‌ی سوم به چاپ رسید که شکل ؟ آن را نشان می‌دهد.



(ب) نمای رو برو

(آ) نمای پشت

شکل ۳-۳: تصاویر بدنی اصلی نسخه‌ی سوم

این نسخه تقریبا تمام شرایط فوق را ارضا می‌کرد. جانمایی حسگر PPG تنها موردی بود که باید انجام میشد. بعد از افروden این قسمت، نسخه‌ی چهارم و نهایی آماده شد.

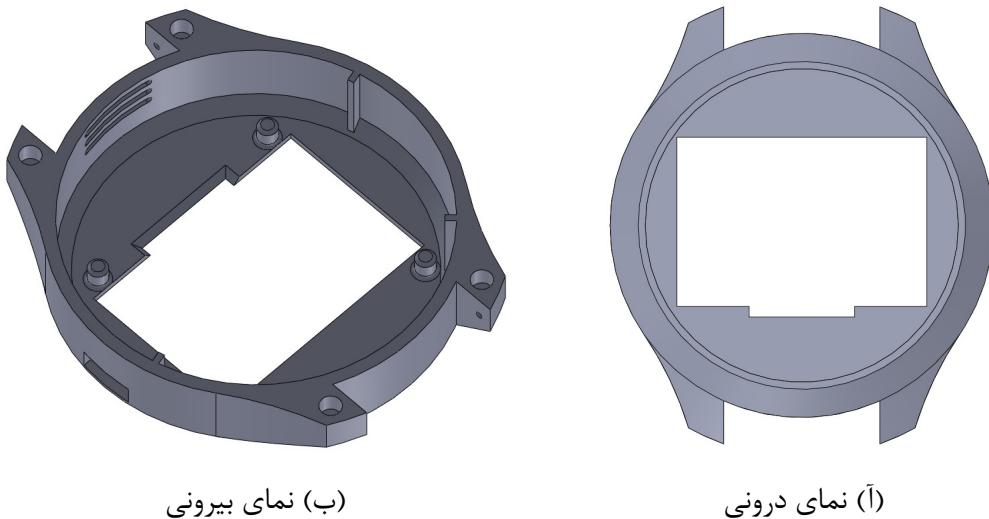
۲-۱-۳ نسخه نهایی

شروط بحث شده در بالا را برای نسخه‌ی نهایی بررسی می‌کنیم.

۱. محل نصب صفحه نمایش:

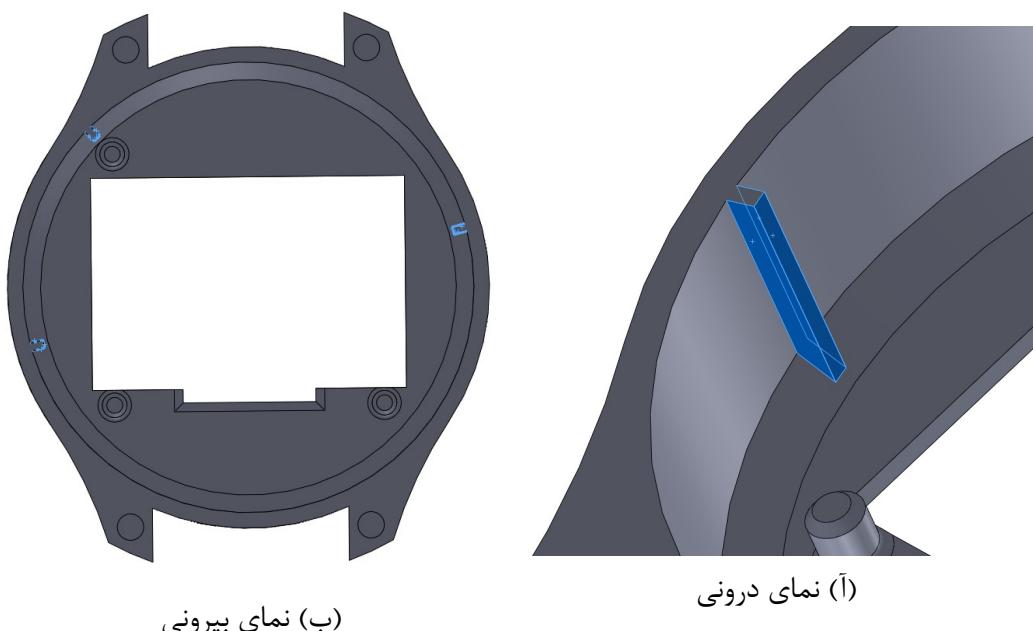
از سمت بیرون یک مستطیل خالی شده است که تا صفحه نمایش در آن قرار گیرد. از داخل هم سه استوانه منطبق بر سه سوراخ صفحه نمایش وجود دارد تا آن را در جای خود نگه دارد. اگر به

تصویر صفحه نمایش (شکل ۱۲-۲) دقیق کنید، پایین آن زائداتی برای اتصال سیم‌های فلت به صفحه نمایش قرار دارد. این زائدات به شکل یک برش مستطیلی کوچک از صفحه فوکانی بدنه جدا شده است. شکل ۳-۴ این قسمت‌ها را به خوبی نشان می‌دهد.



شکل ۳-۴: تصاویر محل نصب صفحه نمایش در بدنه

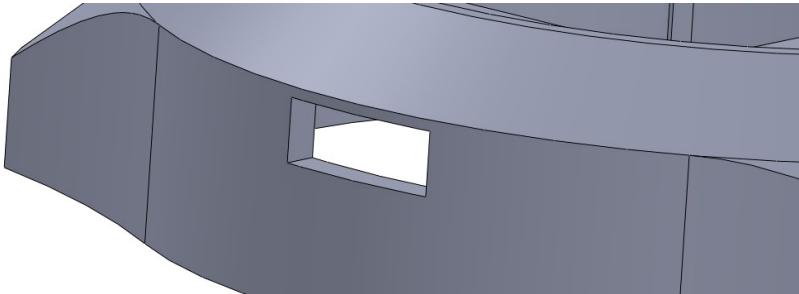
۲. جانمایی پی‌سی‌بی و جلوگیری از چرخش آن:
بر روی پی‌سی‌بی سه شیار کوچک وجود دارد. متناظر با آن روی بدنه نیز سه زائده با همان ابعاد تعبیه شده است. این سه زائدات مانع چرخش پی‌سی‌بی در جای خود می‌شوند. تصویر این نگهدارنده‌ها در شکل ۵-۳ قابل مشاهده است.



شکل ۳-۵: تصاویر محل نصب پی‌سی‌بی در بدنه

۳. محل اتصال کابل USB:

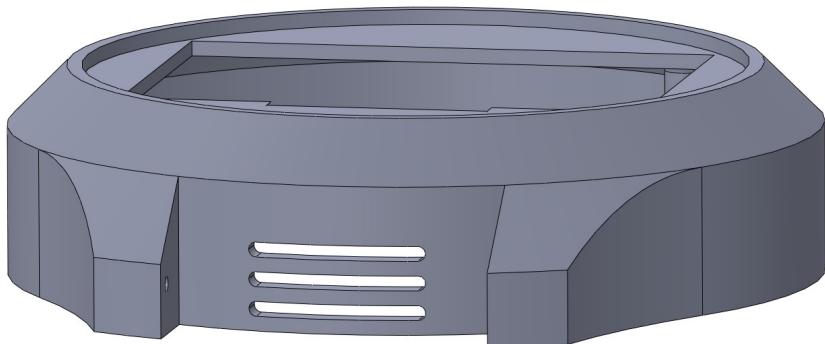
بر سطح جانبی بدنه یک سوراخ مستطیل شکل به ابعاد کانکتور micro USB تعبیه شده است.
شکل ۶-۳ محل آن روی بدنه را نمایش می‌دهد.



شکل ۶-۳: تصویر محل اتصال USB در بدنه

۴. محل عبور هوا:

در نزدیکی قطعات مدار شارژ سه شیار برای عبور هوا وجود دارد تا قطعات داخلی به علت بالا رفتن دما آسیب نبینند. این شیارها در شکل ۷-۳ قابل مشاهده‌اند.



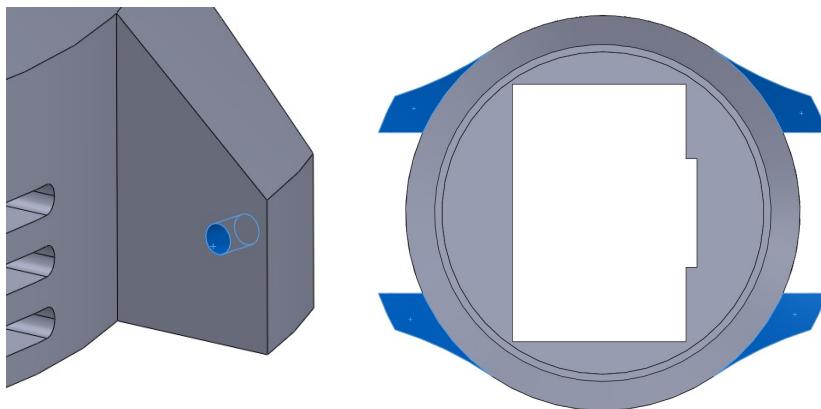
شکل ۷-۳: تصویر شیارهای عبور هوا

۵. محل اتصال بند:

بندهای موردنظر برای این ساعت مربوط به ساعت هوشمند Haylou است. این بندها از جنس پلاستیک هستند. برای اتصال بند به بدنه‌ی اصلی چهار بازوی کوچک به بدنه اضافه شده است. داخل آن‌ها سوراخ‌هایی کوچکی موجود است که پین بند در آن‌ها قرار گیرد. تصویر این بازوها و محل نصب پین را شکل ۸-۳ مشاهده می‌کنید.

۶. زیبایی بصری: جای تردید نیست که این طرح زیبا است :)

۷. ابعاد دقیق طرح در بخش‌های بعدی تشریخ خواهد شد؛ اما در مورد تناسب ابعاد، قطر دایره‌ی این ساعت حدود ۵ سانتی‌متر است که در مقایسه با ساعت‌های موجود در بازار، مقدار بزرگی نیست و معقول است.



شکل ۳-۸: تصاویر محل نصب بند

در نهایت تصویر بدنی اصلی در شکل ۹-۲ دیده می‌شود. چاپ این نسخه توسط شرکت PCBWay [۴] به رایگان انجام شده است. جنس بدن از رزین است که باعث می‌شود شفاف باشد و داخل آن دیده شود.

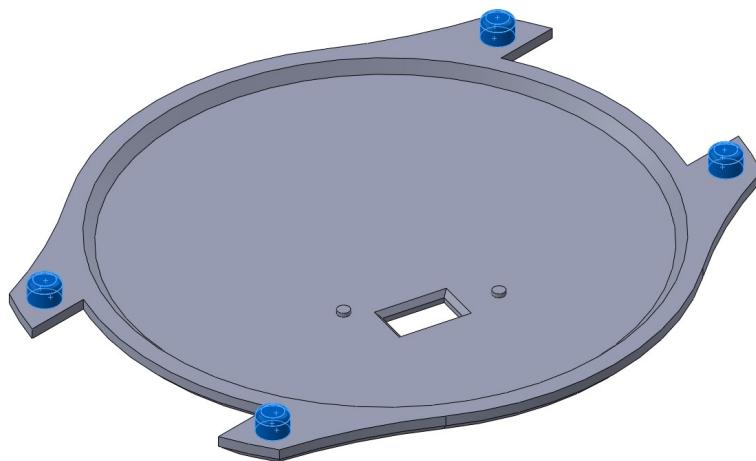


شکل ۳-۹: تصویر چاپ شده نسخهٔ نهایی بدن از جنس رزین و شفاف

۲-۳ دریچه‌ی پشتی

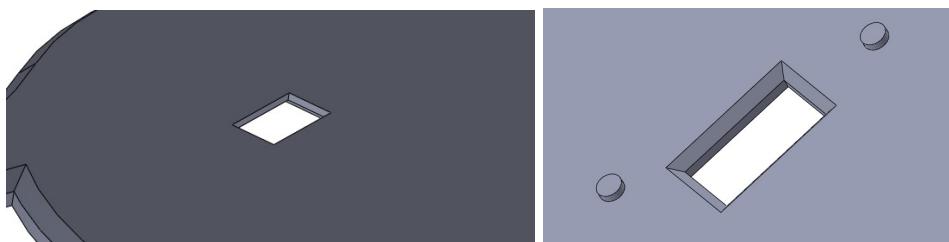
دربیچه‌ی پشتی وظیفه دارد تا بر پشت بدنی اصلی نصب شود و قطعات و تجهیزات داخل ساعت را محافظت کند. از طرفی محل نصب حسگر PPG نیز هست. این دو وظیفه را شرح می‌دهیم.

۱. اتصال به بدنی اصلی: برای اتصال به بدنی اصلی، چهار استوانه‌ی کوچک بر روی دریچه طراحی شده است. این استوانه‌ها منطبق بر سوراخ‌های روی بدن است. این سوراخ‌ها در شکل ۵-۳ به خوبی دیده می‌شوند. بدین شکل دریچه به بدن متصل می‌شود. تصویر دریچه در شکل ۳-۱۰ نشان داده شده است.



شکل ۳-۱۰: تصویر استوانه‌های اتصال دریچه به بدنه

۲. اتصال حسگر PPG: حسگر باید جایی نصب شود که با پوست تماس مستقیم داشته باشد. لذا بر روی دریچه سوراخ مستطیل شکلی ایجاد شده است تا محل نصب حسگر باشد. دو زانده‌ی کوچک نیز در نزدیکی آن قرار دارد تا منطبق بر سوراخ‌های روی حسگر باشد. این سوراخ‌ها در شکل ۲-۱۵ قابل مشاهده‌اند. شکل ۱۱-۳ محل نصب حسگر را نشان می‌دهد.



شکل ۱۱-۳: تصاویر محل نصب حسگر PPG

در نهایت تصویر دریچه‌ی پشتی در شکل ۱۲-۳ دیده می‌شود. چاپ این نسخه نیز توسط شرکت PCBWay [۴] به رایگان انجام شده است. جنس بدنه از رزین است که باعث می‌شود شفاف باشد و داخل آن دیده شود.

۳-۳ اتصالات

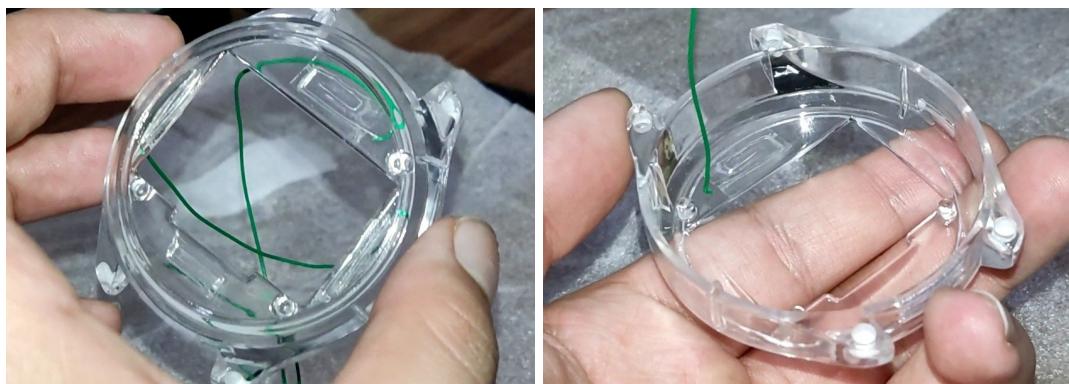
تمامی اتصالات مکانیکی در این بخش توضیح داده می‌شود.



شکل ۱۲-۳: تصویر چاپ شده‌ی نسخه‌ی نهایی دریچه‌ی پشتی از جنس رزین و شفاف

۱-۳-۳ کلیدهای لمسی

باید چهار قطعه سیم به شکل حلقوی سیم پیچی شده و از داخل به بدنه‌ی اصلی چسبانده شود. بدین منظور از سیم‌های وایر رپ^۱ استفاده شده است. همانطور که در شکل ۱۳-۳ دیده می‌شود این سیم‌ها به کمک چسب نواری به بدنه‌ی اصلی متصل شده‌اند. کلیدهای لمسی به کمک خاصیت خازنی کار می‌کنند. سر دیگر سیم‌ها به محل اتصال کلیدهای لمسی در شکل ۱۸-۲ ب لحیم می‌شوند. با توجه به این که می‌توان لایه‌ی نازک فوقانی را به صورت یک دی‌الکتریک برای خازن در نظر گرفت، می‌توان گفت با اینکه کلیدها پشت صفحه نصب شده‌اند اما باز هم به درستی کار خواهند کرد.



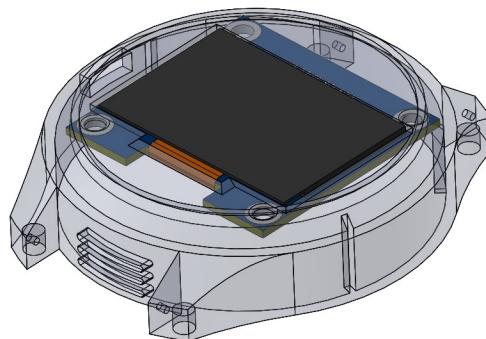
(آ) یک کلید نصب شده بر پشت سطح فوقانی (ب) چهار کلید نصب شده در چهار سمت بدنه

شکل ۱۳-۳: تصاویر اتصال کلیدهای لمسی به بدنه

Wire wrap^۱

۲-۳-۳ صفحه نمایش

همانطور که بالاتر بحث شد، صفحه نمایش با قرار گیری در سه استوانه‌ی موجود در بدنه سر جای خود محکم می‌شود. شکل ۱۴-۳ گویای این موضوع است.

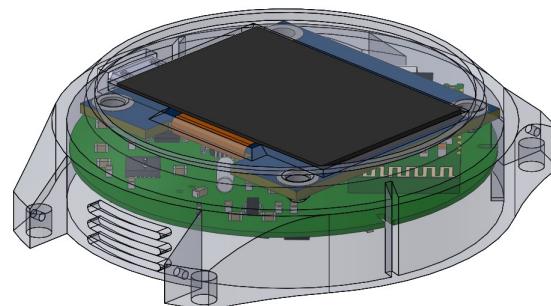


شکل ۱۴-۳: تصویر اتصال صفحه نمایش به بدنه

۳-۳-۳ پی‌سی‌بی

همانطور که بالاتر بحث شد، پی‌سی‌بی با قرار گیری روی سه زائدی موجود در بدنه سر جای خود محکم می‌شود. شکل ۱۵-۳ گویای این موضوع است.

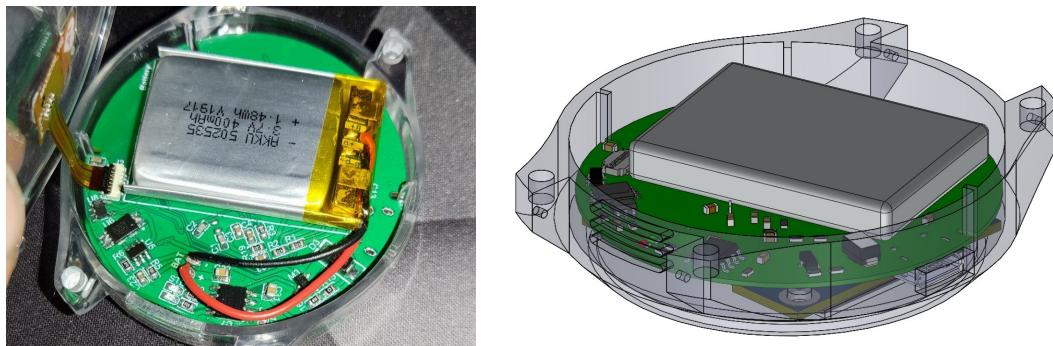
علت آن که برای اتصال صفحه نمایش یک استوانه از چهار استوانه حذف شده است آن است که استوانه‌ی حذف شده دقیقاً بالای موتور ایجاد لرزش قرار داشت و در صورت وجود، به آن گیر می‌کرد و مانع جا گیری صحیح پی‌سی‌بی می‌شد.
برای جلوگیری از اتصال ناخواسته و اتصال کوتاه بین صفحه نمایش و پی‌سی‌بی، یک ورق نازک فوم بین این دو قرار گرفته است.



شکل ۱۵-۳: تصویر اتصال پی‌سی‌بی به بدنه

۴-۳-۳ باتری

باتری ساعت به کمک نوار چسب به پی‌سی‌بی می‌چسبد و در جای خود محکم می‌شود. سیم‌های آن به محل خود لحیم شده و اتصال برقرار می‌شود. شکل ۱۶-۳ این اتصال را نشان می‌دهد.

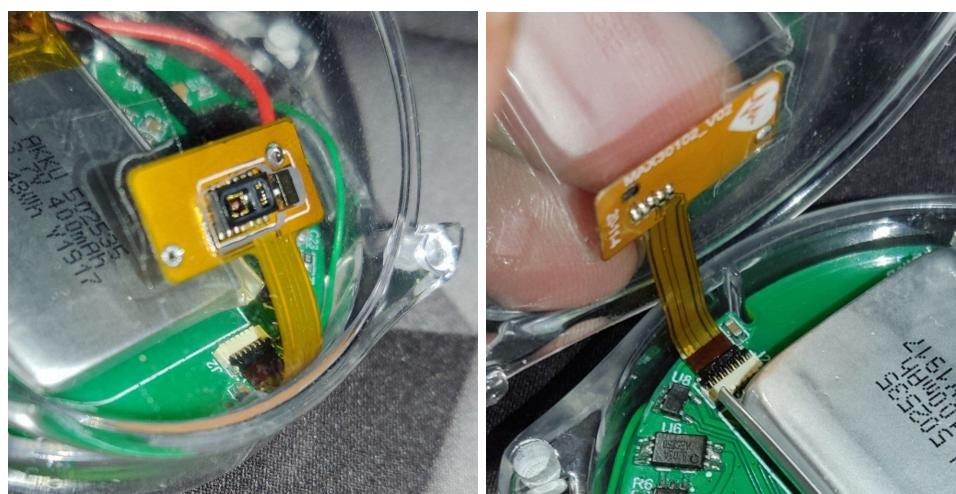


(آ) تصویر اتصال باتری در سالیدورکز
(ب) تصویر واقعی باتری و اتصال آن

شکل ۱۶-۳: تصاویر اتصال باتری

۵-۳-۳ حسگر PPG

این حسگر روی دریچه‌ی پشتی نصب می‌شود. همانطور که در شکل ۱۷-۳ مشاهده می‌شود این حسگر به دریچه متصل است و سمت دیگر آن در جای مخصوص خود به پی‌سی‌بی اتصال دارد. سوراخ مستطیل شکل روی دریچه و ضخامت مناسب آن باعث می‌شود سطح حسگر مماس سطح بیرونی دریچه باشد. اینگونه نه بیرون زدگی دارد تا آسیب ببیند و نه سطح آن از پوست فاصله می‌گیرد.



(آ) نمای درونی
(ب) نمای بیرونی

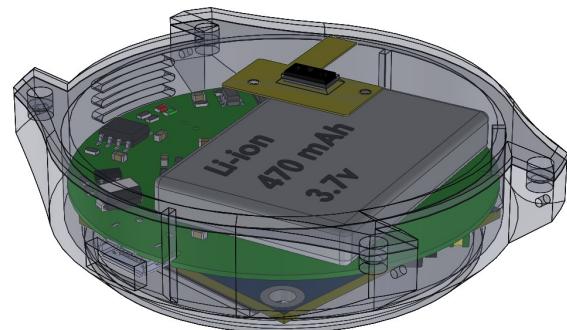
شکل ۱۷-۳: تصاویر اتصال حسگر PPG

۶-۳-۳ دریچه‌ی پشتی

دریچه‌ی پشتی مطابق با شکل ۱۸-۳ به کمک چهار استوانه‌ی موجود روی آن به بدنی اصلی متصل می‌شود.



(ب) تصویر واقعی دریچه و اتصال آن



(آ) تصویر اتصال دریچه در سالیدورکز

شکل ۳: تصاویر اتصال دریچه‌ی پشتی

در نهایت بدنی و مکانیک ساعت تکمیل شد و ظاهر نهایی ساعت به شکل ۱۹-۳ در آمد.



(ب) نمای زیر



(آ) نمای رو

شکل ۱۹-۳: تصاویر بدنی کامل

فصل چهارم

نرم افزار

در این فصل به جزئیات نرم افزار پرداخته می شود. منظور از نرم افزار، کدهایی است که نوشته شده‌اند تا توسط پردازنده‌ی روی ساعت اجرا شوند. در اصطلاح فنی به این بخش سفت‌افزار^۱ نیز می‌گویند. بین سفت‌افزار (برنامه‌ای که روی ریزپردازنده‌ها اجرا می‌شود) و نرم افزارهایی که برای سیستم‌های سطح بالا مانند رایانه نوشته می‌شود، تفاوت‌های بسیاری وجود دارد. برای مثال به چند نمونه از این تفاوت‌ها اشاره می‌کنیم:

۱. عدم وجود سیستم عامل:

برنامه‌هایی که به زبان‌های مختلفی مانند C یا Python برای رایانه‌ها نوشته می‌شوند، توسط سیستم عامل اجرا می‌شوند. سیستم عامل وظیفه دارد تا برنامه‌های متنوعی که روی سیستم وجود دارد را زمان‌بندی کند، با مدیریت صحیح حافظه آن‌ها را اجرا کند و ارتباط با سخت‌افزار را نیز بر عهده بگیرد. اما در حالتی که برای یک ریزپردازنده برنامه می‌نویسیم، دیگر سیستم عاملی وجود ندارد. بلکه تمام برنامه خط به خط و مستقیماً توسط ریزپردازنده اجرا می‌شود.

۲. ارتباط مستقیم با سخت‌افزار:

در سیستم‌های سطح بالا که به سیستم عامل مجهzenد، ارتباط با سخت‌افزار نیز بر عهده‌ی سیستم عامل است. در صورتی که نیاز باشد با دستگاه‌های ورودی/خروجی ارتباطی برقرار شود، از طریق توابع سیستمی موجود در سیستم عامل این ارتباط ایجاد می‌شود. اما هنگام نوشتن برنامه برای ریزپردازنده، باید لایه‌ی ارتباط با سخت‌افزار را نیز نوشت. زیرا سیستم عاملی وجود ندارد و تک تک ارتباطات سخت‌افزاری باید توسط برنامه نویس راهاندازی شود.

۳. مدیریت حافظه:

هنگامی که در برنامه‌ای به زیان C می‌خواهیم از حافظه استفاده کنیم، باید با دستور malloc و نظایر آن، به سیستم عامل درخواست دهیم تا آدرس مناسبی را به برنامه‌ی ما اختصاص دهد. همچنین در صورتی که بخواهیم در آدرسی که به برنامه‌ی ما تعلق ندارد مقداری را بنویسیم یا از آن بخوانیم، سیستم عامل مانع می‌شود و با خطای عدم دسترسی مواجه می‌شویم. اما در موردی که با ریزپردازنده‌ها سر و کار داشته باشیم دیگر نیازی به درخواست و اختصاص حافظه نیست. چرا که سیستم عاملی وجود ندارد و تمام حافظه در اختیار برنامه‌ی ما است. در این حالت مدیریت حافظه امری بسیار حساس است. زیرا چیزی وجود ندارد تا مانع دسترسی‌های اشتباه شود و ممکن است با خطا در مدیریت حافظه، عملکرد برنامه دچار اختلالات جدی شود.

۴. محدودیت منابع:

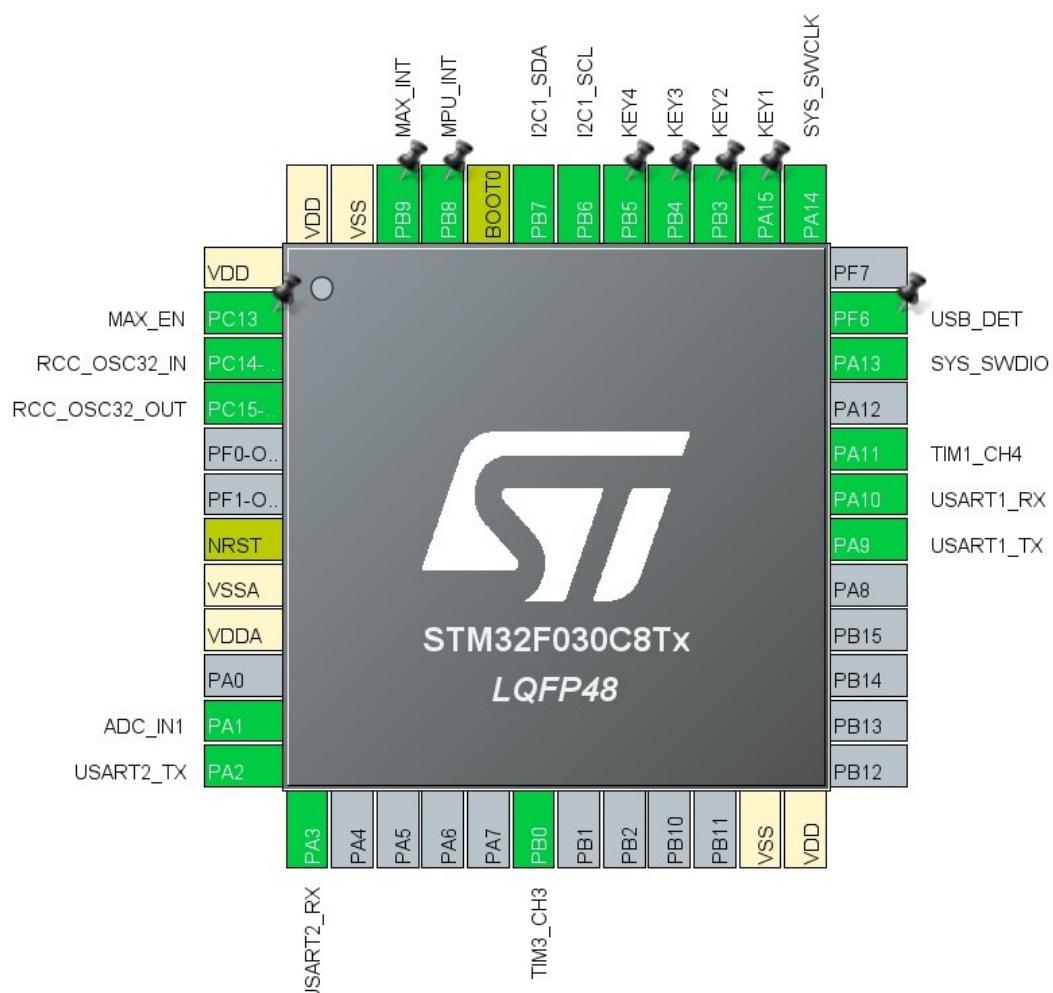
پردازنده‌ی رایانه‌ها معمولاً پردازنده‌های قوی‌ای هستند و سرعت بالایی دارند. همچنین حافظه‌های موجود نیز بسیار زیاد هستند. حال آنکه در پردازنده‌ها این منابع بسیار بسیار محدود هستند. همانطور که در ضمیمه‌ی ؟ قابل مشاهده است، مقدار حافظه‌ی فلش ریزپردازنده‌ی ما فقط ۶۴ کیلوبایت است. به این معنی که حجم برنامه بعد از کامپایل نباید از این مقدار تجاوز کند. این

Firmware^۱

محدودیت هرگز در نرم افزارهای رایج وجود ندارد. این اتفاق باعث می شود تا برنامه نویسی برای ریز پردازنده ها دشوار تر شود و نیاز به دانش بیشتری داشته باشد تا بتوان با بهینه سازی های مختلف و الگوریتم های بهتر، حجم برنامه را کاهش داد. در ادامه یکی از این بهینه سازی ها در رابطه با پیاده سازی فیلتر کالمن توضیح داده خواهد شد.

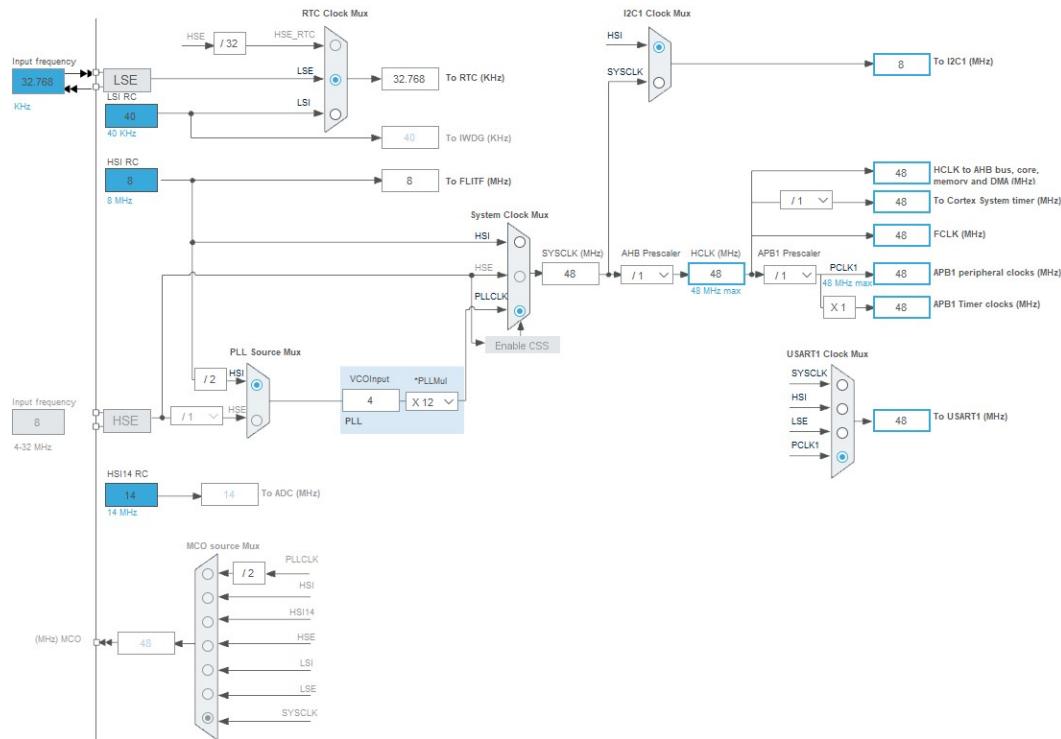
۱-۴ تنظیمات سخت افزاری

همانطور که بالاتر اشاره شد، ارتباط با سخت افزار باید توسط برنامه رسیدگی شود. برای این هدف از نرم افزار CubeMX استفاده شد. شکل ۱-۴ نحوه ا اختصاص پایه ها را نشان می دهد. هر کدام از بخش ها به تفصیل در ادامه بررسی می شود.



۱-۱-۴ کلاک

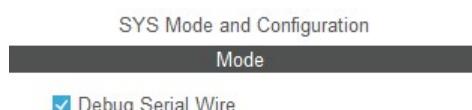
تنظیمات کلاک به صورت شکل ۲-۴ است. نوسان ساز اصلی روی ۸ مگاهرتز داخلی تنظیم شده است و برای ایجاد کلاک بخش ساعت و تاریخ، از یک کریستال خارجی با فرکانس ۳۲.۷۶۸ کلیوهرتز استفاده شده است. کلاک بخش های مختلف نیز در همین تصویر قابل مشاهده است.



شکل ۲-۴: تنظیمات کلاک

۲-۱-۴ برنامه ریزی و اشکال زدایی

برای بحث برنامه ریزی^۲ و اشکال زدایی^۳ از پروتکل Serial Wire استفاده شده است که در شکل ۳-۴ تنظیم آن دیده می شود.



شکل ۳-۴: تنظیمات Debugging

^۲ انتقال برنامه‌ی نوشته شده به ریزپردازنده یا به اصطلاح Programming Debugging^۳

GPIO ۳-۱-۴

GPIO ها پایه هایی هستند که به عنوان ورودی / خروجی دیجیتال مورد استفاده قرار می گیرند. در این پروژه از GPIO های متعددی استفاده شده است که در شکل ۴-۴ فهرست آنها قابل مشاهده است. در ادامه هر مورد توضیح داده می شود.

Pin Name	Signal on Pin	GPIO output level	GPIO mode	GPIO Pull-up/Pull-down	Maximum output speed	Fast Mode	User Label
PA15	n/a	n/a	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	n/a	KEY1
PB3	n/a	n/a	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	n/a	KEY2
PB4	n/a	n/a	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	n/a	KEY3
PB5	n/a	n/a	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	n/a	KEY4
PB8	n/a	n/a	External Interrupt Mode with Rising edge trigger detection	No pull-up and no pull-down	n/a	n/a	MPU_INT
PB9	n/a	n/a	External Interrupt Mode with Falling edge trigger detection	No pull-up and no pull-down	n/a	n/a	MAX_INT
PC13	n/a	Low	Output Push Pull	No pull-up and no pull-down	Low	n/a	MAX_EN

شکل ۴-۴: تنظیمات GPIO

وقفه های خارجی

وقفه های خارجی ^۴ پایه هایی هستند که در صورت وقوع رویداد خاصی، کار عادی پردازنده را متوقف می کنند و یک تابع به خصوص را به اجرا در می آورند. بعد از اجرای روتین وقفه، پردازنده به کار قبلی خود ادامه می دهد.

رویدادهای مورد استفاده در این پروژه دو مورد است: ۱ - لبهی پایین رونده ^۵ ۲ - لبهی بالا رونده ^۶ همانطور که در شکل ۴-۴ مشهود است، پایه های ۳، ۴، ۵ و ۸ از پورت B و پایهی ۱۵ از پورت A برای لبهی بالارونده تنظیم شده اند. پایهی ۹ از پورت B نیز به لبهی پایین رونده حساس است. اینکه کدام پایه توسط کدام بخش تحریک می شود در ستون User label شکل ۴-۴ دیده می شود. برچسب های KEY مربوطه به کلیدهای لمسی روی بدنه هستند. در صورت لمس شدن هر کلید، تابع مخصوص آن اجرا می شود. برچسب MPU مربوط به حسگر حرکتی است. این پایه وقتی فعال می شود که داده های جدید حسگر آماده شده باشد. برچسب MAX نیز به همین شکل عمل می کند. هرگاه دادهی حسگر PPG آمادهی قرائت باشد، این وقفه فعال می شود.

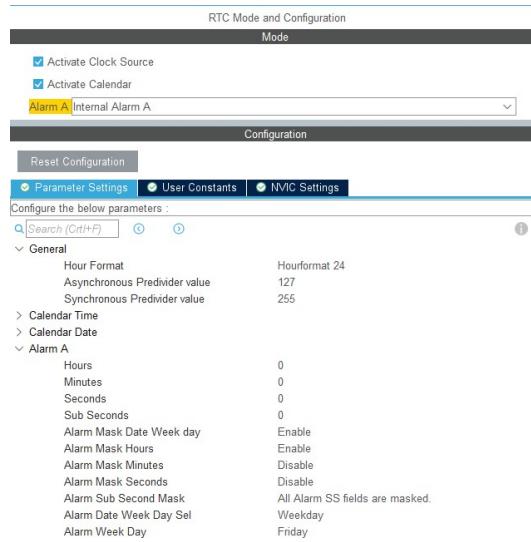
خروجی دیجیتال

پایهی ۱۳ از پورت C به صورت خروجی دیجیتال تعریف شده است. این خروجی به پایهی فعال ساز سوییج ماسفتی متصل است که حسگر PPG را روشن می کند (شکل ۱۱-۲). برای روشن یا خاموش کردن این حسگر، کافی است این پایه را صفر یا یک کرد.

External interrupts^۴Falling Edge^۵Rising Edge^۶

RTC ۴-۱-۴

واحد RTC برای نگهداری و کار با ساعت و تاریخ است. تنظیمات این بخش در شکل ۵-۴ دیده می‌شود. در این بخش می‌توان یک آلام هم فعال کرد که در ساعت و روز مشخصی یک وقفه را فعال کند. در اینجا آلام هم فعال شده است.



شکل ۴-۵: تنظیمات RTC

۵-۱-۴ تایمرها

این ریزپردازنه هفت تایмер دارد که در این پروژه هر هفت تایمر استفاده شده‌اند. در ادامه کاربرد و تنظیمات هر تایمر شرح داده می‌شود.

TIM1

تایمر شماره‌ی یک مطابق با تنظیمات شکل ۴-۶ راهاندازی شده است تا بتواند PWM موردنیاز برای کنترل دور موتور ایجاد لرزش را تولید کند. البته مقدار رجیستر ARR^۴ در برنامه به صورت پویا تغییر می‌کند.

TIM3

تایمر شماره‌ی یک مطابق با تنظیمات شکل ۴-۶ ب راهاندازی شده است تا بتواند PWM موردنیاز برای تنظیم فرکانس صدای باز را تولید کند. البته مقدار رجیستر ARR در برنامه به صورت پویا تغییر می‌کند.

Real-time Clock^۷

رجیستری است که برای تنظیم فرکانس اصلی تایمر استفاده می‌شود Auto-Reload Register^۸

TIM3 Mode and Configuration

Mode

Slave Mode: Disable
Trigger Source: Disable
 Internal Clock
Channel1: Disable
Channel2: Disable
Channel3: PWM Generation CH3
Channel4: Disable
Combined Channels: Disable

Configuration

Reset Configuration

NVIC Settings DMA Settings GPIO Settings
 Parameter Settings (Selected) User Constants

Configure the below parameters :

Search (Ctrl+F)

Counter Settings

- Prescaler (PSC - 16 bits value) 48-1
- Counter Mode Up
- Counter Period (AutoReload ... 1000-1
- Internal Clock Division (CKD) No Division
- auto-reload preload Disable

Trigger Output (TRGO) Parameters

PWM Generation Channel 3

Mode	Pulse (16 bits value)	Output compare preload	Fast Mode	CH Polarity
PWM mode 1	500	Enable	Disable	High

TIM1 Mode and Configuration

Mode

Slave Mode: Disable
Trigger Source: Disable
Clock Source: Internal Clock
Channel1: Disable
Channel2: Disable
Channel3: Disable
Channel4: PWM Generation CH4
Combined Channels: Disable

Configuration

Reset Configuration

NVIC Settings DMA Settings GPIO Settings
 Parameter Settings (Selected) User Constants

Configure the below parameters :

Search (Ctrl+F)

Counter Settings

- Prescaler (PSC - 16 bits value) 48-1
- Counter Mode Up
- Counter Period (AutoReload Register - 16 bits value) 1000-1
- Internal Clock Division (CKD) No Division
- Repetition Counter (RCR - 8 bits value) 0
- auto-reload preload Disable

Trigger Output (TRGO) Parameters

Break And Dead Time management - BRK Configuration

Break And Dead Time management - Output Configuration

PWM Generation Channel 4

Mode	Pulse (16 bits value)	Output compare preload	Fast Mode	CH Polarity	CH Idle State
PWM mode 1	300	Enable	Disable	High	Reset

(ب) تنظیمات تایمر سه

(آ) تنظیمات تایمر یک

شکل ۴-۶: تنظیمات دو تایمر

TIM6

تایمر شماره‌ی شش مطابق شکل ۴-۷ تنظیم شده است تا هر یک میلی ثانیه یک وقفه را فعال کند. این وقفه توابع مربوط به فیلتر کالمن را اجرا می‌کند. در واقع می‌توان گفت دوره تناوب نمونه‌گیری و پردازش سیستم کالمن یک میلی ثانیه است.

TIM6 Mode and Configuration

Mode

Activated
 One Pulse Mode

Configuration

Reset Configuration

User Constants NVIC Settings DMA Settings
 Parameter Settings (Selected)

Configure the below parameters :

Search (Ctrl+F)

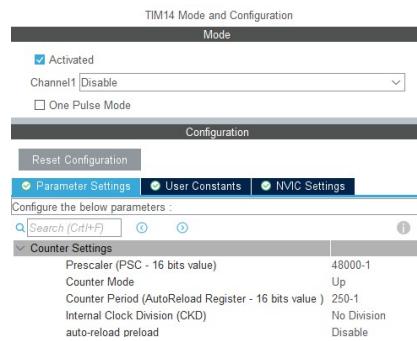
Counter Settings

- Prescaler (PSC - 16 bits value) 48-1
- Counter Mode Up
- Counter Period (AutoReload Register - 16 bits value) 1000-1
- auto-reload preload Disable

شکل ۴-۷: تنظیمات تایمر شش

TIM14

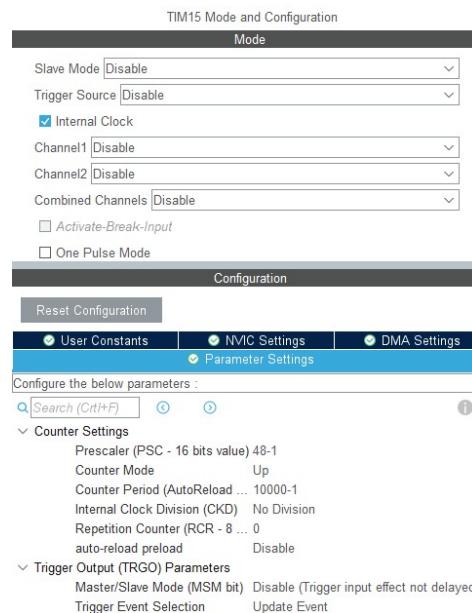
تایمر شماره‌ی چهارده مطابق شکل ۸-۴ تنظیم شده است تا هر 25° میلی ثانیه یک وقفه را فعال کند. این وقفه مربوط به توابع سیستمی است. در این توابع مواردی از قبیل تازه‌سازی صفحه نمایش و تنظیم برخی متغیرها انجام می‌شود.



شکل ۸-۴: تنظیمات تایمر چهارده

TIM15

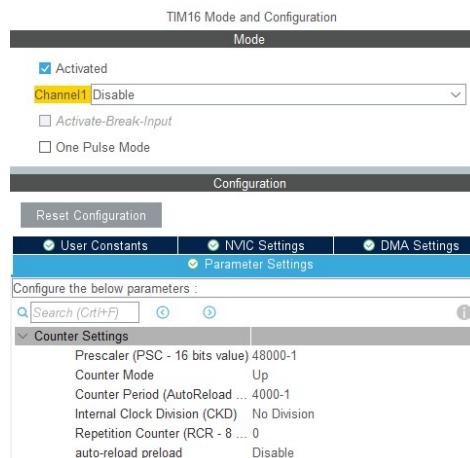
تایمر شماره‌ی پانزده مطابق شکل ۹-۴ تنظیم شده است تا هر 10° میلی ثانیه به واحد ADC فرمان تبدیل دهد. به این معنی که واحد آنالوگ به دیجیتال فقط در صورتی یک نمونه‌برداری را شروع می‌کند که تایمر شماره ۱۵ به آن فرمان دهد. اینگونه یک نمونه‌برداری دقیق زمانی ایجاد می‌شود.



شکل ۹-۴: تنظیمات تایمر پانزده

TIM16

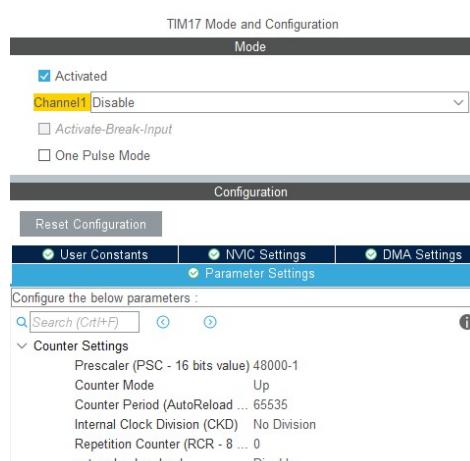
تایمر شماره‌ی شانزده مطابق شکل ۱۰-۴ تنظیم شده است تا هر ۴ ثانیه یک وقفه را فعال کند. این وقفه برای خاموش کردن صفحه نمایش است. هرگاه کاربر به مدت ۴ ثانیه با ساعت تعامل نداشته باشد این وقفه صفحه را خاموش می‌کند.



شکل ۱۰-۴: تنظیمات تایمر شانزده

TIM17

تایمر شماره‌ی هفده مطابق شکل ۱۱-۴ تنظیم شده است. این تایمر وقفه ندارد و به صورت یک زمان‌سنج به کار می‌رود. از این تایмер برای ساخت تابع تأخیر استفاده شده است. حداکثر تأخیر قابل تولید توسط این تایمر، دو به توان ۱۶ یعنی ۶۵۵۳۶ میلی ثانیه است.

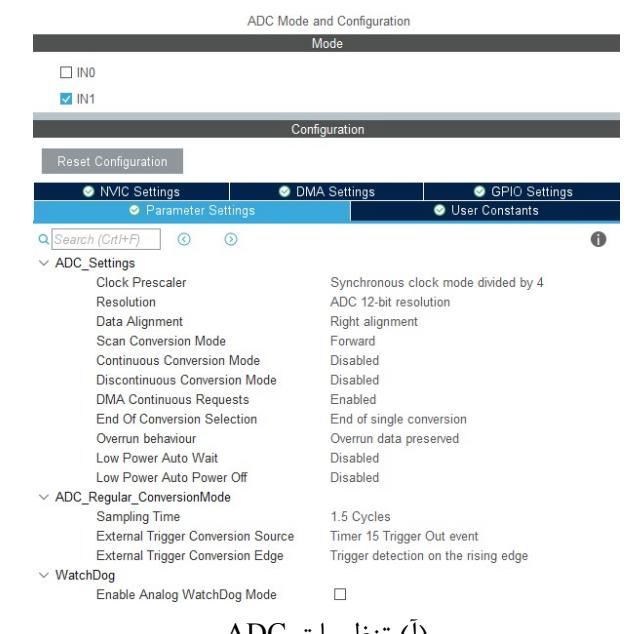


شکل ۱۱-۴: تنظیمات تایمر هفده

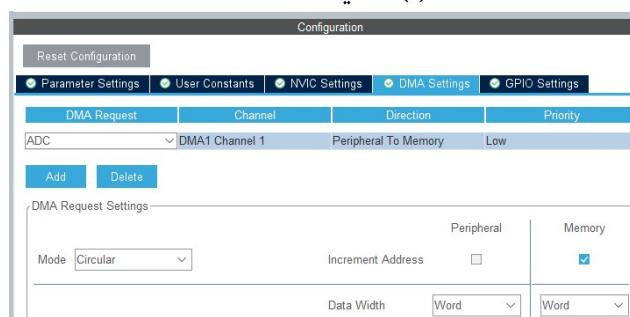
ADC ۶-۱-۴

مبدل آنالوگ به دیجیتال در این پردازنده ۱۲ بیتی است. یعنی ولتاژ ورودی را از بازه‌ی ° تا ۳.۳ ولت به بازه‌ی ° تا ۴۰۹۵ نگاشت می‌کند. اینگونه می‌توان با یک ضرب و تقسیم ساده مقدار ولتاژ ورودی را خواند. شکل ۱۲-۴ تنظیمات این واحد را نشان می‌دهد. همانطور که گفته شد، این واحد به کمک تایمر ۱۵ فعال می‌شود و با هر فرمان آن یک نمونه برمی‌دارد. برای ذخیره‌ی این نمونه‌ها، ADC را با DMA^۹ کوپل می‌کنیم. اینگونه هر بار که ADC یک نمونه را تبدیل کرد، به طور مستقیم و بدون دخالت پردازنده آن را در یک آرایه ذخیره می‌کند. بعد از اینکه آرایه پر شد نیز با فعال کردن یک وقفه به ما خبر می‌دهد تا عملیات پردازشی روی آن انجام گیرد.

این ساختار که ADC با تایمر فعال شود و با DMA کار کند، حرفه‌ای ترین ساختار راهاندازی این واحد است. تنظیمات DMA در شکل ۱۲-۴ ب دیده می‌شود.



(آ) تنظیمات ADC

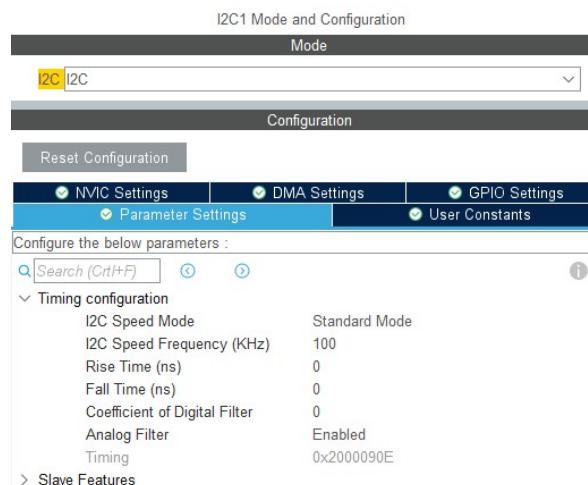


(ب) تنظیمات DMA

شکل ۱۲-۴: تنظیمات مبدل آنالوگ به دیجیتال

I2C ۷-۱-۴

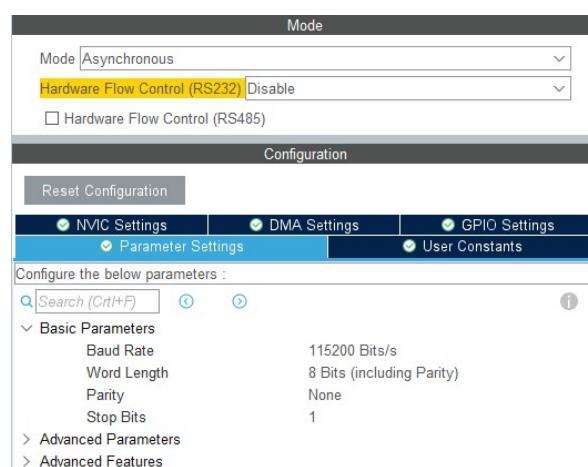
واحد I2C مطابق شکل ۱۳-۴ تنظیم شده است. این واحد یک واحد ارتباطی تحت پروتکل I2C است.



شکل ۱۳-۴: تنظیمات I2C

USART ۸-۱-۴

واحد USART1 و USART2 مطابق شکل ۱۴-۴ تنظیم شده و تنظیمات مشابهی دارند. با این تفاوت که یک واحد DMA نیز وصل است تا داده‌ی دریافتی بلوتوث را مستقیماً در حافظه ذخیره کند و در صورت تکمیل شدن پیام، وقفه‌ی مربوطه را فعال کند تا داده‌ی دریافتی پردازش شود.



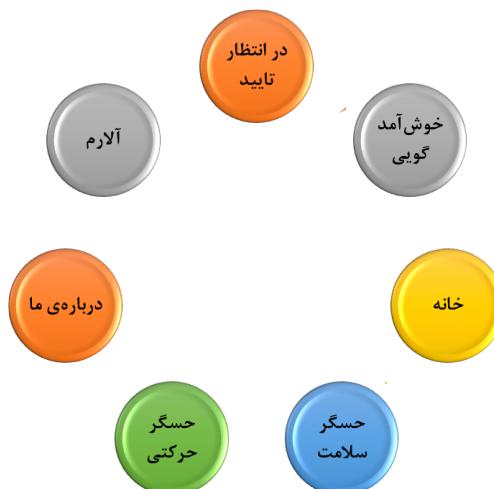
شکل ۱۴-۴: تنظیمات USART

۲-۴ معماری

برنامه‌ی نوشته شده در این پروژه مشابه برنامه‌های عادی ریزپردازندگان نیست. در اکثر قریب به اتفاق پروژه‌ها، معماری به این صورت است که ابتدا چند تابع اجرا می‌شوند تا تنظیمات کلی را انجام دهند، سپس یک حلقه‌ی (1) وجود دارد تا روتین اصلی برنامه به صورت مداوم در آن اجرا شود. در این معماری وظیفه‌های^{۱۰} مختلف به طور غیر همزمان اجرا شده و مشکلات زمان‌بندی دارند. اما معماری این پروژه یک معماری رویداد محور^{۱۱} است. در این معماری حلقه‌ی (1) while خالی است و تمام وظیفه‌ها در توابع وقفه رسیدگی می‌شوند. در کنار این معماری رویداد محور، یک ماشین حالت محدود یا به اختصار FSM^{۱۲} نیز وجود دارد تا حالت‌های کاری مختلف را اجرا کند. در ادامه بخش‌های مختلف این معماری شرح داده می‌شود.

۱-۲-۴ ماشین حالت

ماشین حالت مفهومی است برای توصیف و تشریح عملکرد یک سیستم که حالت‌های کاری مختلفی دارد. این مفهوم نشان می‌دهد که چگونه هر حالت به حالتی دیگر تبدیل می‌شود و در هر حالت خروجی سیستم چیست.



شکل ۱۵-۴: حالت‌های ماشین حالت

در این پروژه ماشین حالت را تعمیم داده‌ام. به این صورت که انتقال بین حالت‌ها همچنان به کمک یک سیگنال کلک صورت می‌گیرد، اما دیگر هر حالت خروجی ثابتی ندارد، بلکه مجموعه‌ای ای توابع در هر حالت دائمًا اجرا می‌شوند. شکل ۱۵-۴^{۱۳} حالت‌ها و نام آن‌ها را نشان می‌دهد.

Tasks^{۱۰}
Event-based architecture^{۱۱}
Finite State Machine^{۱۲}

حالت در انتظار تایید

• روتین اصلی:

۱. نمایش صفحه‌ی درخواست تایید مطابق شکل ۱۶-۴

۲. متوقف کردن تایمر خاموش کردن صفحه نمایش

• کلید بالا فشرده شود:

۱. یک کردن متغیر ack در ساختار سیستم

۲. فراخوانیتابع سیستمی ارسال تاییدیه

• کلید پایین فشرده شود:

۱. صفر کردن متغیر ack در ساختار سیستم

۲. فراخوانیتابع سیستمی ارسال عدم تایید

۳. تنظیم حالت روی خانه

۴. فعال کردن صفحه نمایش



شكل ۱۶-۴: صفحه‌ی در انتظار تایید

حالت خوش آمد گویی

• روتین اصلی:

۱. نمایش صفحه‌ی خوش آمد گویی به مدت ۴ ثانیه ۱۷-۴

۲. تنظیم حالت به عنوان خانه

۳. فعال کردن تایمر خاموش کردن صفحه



شکل ۱۷-۴: صفحه‌ی خوش‌آمد گویی

حالت خانه

- روتین اصلی:
 ۱. قرائت ساعت و تاریخ
 ۲. قرائت میزان شارژ باتری
 ۳. قرائت وضعیت اتصال به گوشی
 ۴. نمایش صفحه‌ی اصلی مطابق شکل ۱۸-۴

• کلید بالا فشرده شود:

۱. تنظیم حالت روی درباره‌ی ما

• کلید پایین فشرده شود:

۱. متوقف کردن تایمر فیلتر کالمون
۲. متوقف کردن تایمر ADC
۳. روشن کردن حسگر PPG و تنظیم آن
۴. تنظیم حالت روی حسگر سلامت



(آ) در حالتی که اتصال با گوشی برقرار است (ب) در حالتی که اتصال با گوشی برقرار نیست

شکل ۱۸-۴: صفحه‌ی اصلی

حالت حسگر سلامت

- روئین اصلی:

۱۹-۴. نمایش صفحه‌ی حسگر سلامت مطابق شکل

۲. متوقف کردن تایمر خاموش کردن صفحه نمایش

- کلید بالا فشرده شود:

۱. خاموش کردن حسگر سلامت

۲. روشن کردن تایمر فیلتر کالمن

۳. روشن کردن تایمر ADC

۴. روشن کردن تایمر خاموش شدن صفحه نمایش

۵. تنظیم حالت روی خانه

- کلید پایین فشرده شود:

۱. خاموش کردن حسگر سلامت

۲. روشن کردن تایمر فیلتر کالمن

۳. روشن کردن تایمر ADC

۴. تنظیم حالت روی حسگر حرکتی



(آ) در حالتی که اتصال با گوشی برقرار است (ب) در حالتی که اتصال با گوشی برقرار نیست

شکل ۱۹-۴: صفحه‌ی حسگر سامت

حالت حسگر حرکتی

- روئین اصلی:

۲۰-۴. نمایش صفحه‌ی حسگر حرکتی مطابق شکل

۲. متوقف کردن تایمر خاموش کردن صفحه نمایش

• کلید بالا فشرده شود:

۱. متوقف کردن تایمر فیلتر کالمن

۲. متوقف کردن تایمر ADC

۳. روشن کردن حسگر PPG و تنظیم آن

۴. تنظیم حالت روی حسگر سلامت

• کلید پایین فشرده شود:

۱. تنظیم حالت روی درباره‌ی ما



شکل ۴-۲۰: صفحه‌ی حسگر حرکتی

حالت درباره‌ی ما

• روتین اصلی:

۱. نمایش صفحه‌ی درباره‌ی ما مطابق شکل ۴-۲۱

۲. متوقف کردن تایمر خاموش کردن صفحه نمایش

• کلید بالا فشرده شود:

۱. تنظیم حالت روی حسگر حرکتی

• کلید پایین فشرده شود:

۱. روشن کردن تایمر خاموش شدن صفحه نمایش

۲. تنظیم حالت روی خانه



شکل ۲۱-۴: صفحه‌ی درباره‌ی ما

حالت آلام

- روتین اصلی:

۱. نمایش صفحه‌ی آلام مطابق شکل ۲۲-۴
۲. متوقف کردن تایمر خاموش کردن صفحه نمایش

- هر کلیدی که فشرد شود:

۱. پایین آوردن پرچم آلام
۲. روشن کردن تایمر فیلتر کالمن
۳. روشن کردن تایمر خاموش شدن صفحه نمایش
۴. تنظیم حالت روی خانه



شکل ۲۲-۴: صفحه‌ی آلام

تمامی اشکال موجود در صفحات پیکسل به پیکسل طراحی شده و از هیچ شکل آماده‌ای در این پروژه استفاده نشده است.

۲-۲-۴ وقفه‌ها

منابع وقفه در این پروژه ۵ چیز است:

۱. وقفه خارجی^{۱۳}: همانطور که در بخش ۳-۱-۴ بحث شد، ۵ پایه از پایه‌های GPIO برای فعالسازی وقفه خارجی هستند:

(آ) کلیدهای لمسی: بعد از فعال شدن وقفه، با توجه به اینکه سیستم در چه حالتی از حالت‌های هفتگانه قرار دارد، توابع مشخصی اجرا می‌شوند. شرح کامل این اقدامات در بخش ۱-۲-۴ داده شده است. شکل ۲۳-۴ یک نمونه از این کدها را نشان می‌دهد.

```

515     else if(GPIO_Pin == KEY4_Pin){
516         DEBUG("UP PRESSED \n\r");
517         if(!getScreenEable(&sys)){
518             //
519         }
520     else{
521         SCREEN_TIMER_RESET;
522         switch (sys.state)
523         {
524             case ack_waiting:
525                 // accept
526                 sys.ack = 1;
527                 TxGenerator(&sys);
528                 break;
529             case hello:
530                 //
531                 break;
532             case home:
533                 sys.state = about;
534                 break;
535             case bloody_hell:
536                 max30102_turn_off();
537                 HAL_TIM_Base_Start_IT(&BATERY_TIMER);
538                 HAL_TIM_Base_Start_IT(&KALMAN_TIMER);
539
540                 sys.state = home;
541                 SCREEN_TIMER_RESET;
542                 HAL_TIM_Base_Start_IT(&SCREEN_TIMER);
543                 break;
544             case pedomodo:
545                 HAL_TIM_Base_Stop(&KALMAN_TIMER);
546                 HAL_TIM_Base_Stop(&BATERY_TIMER);
547                 max30102_turn_on();
548                 sys.state = bloody_hell;
549                 break;
550             case about:
551                 sys.state = pedomodo;
552                 break;
553             case ringing:
554                 SCREEN_TIMER_RESET;
555                 setRinging(&sys, 0);
556                 HAL_TIM_Base_Start_IT(&KALMAN_TIMER);
557                 HAL_TIM_Base_Start_IT(&SCREEN_TIMER);
558                 sys.state = home;
559                 break;
560             }
561             beep_beep(&sys, 500, 100);
562         }
563     }

```

شکل ۲۳-۴: نمونه کدی از بخش وقفه خارجی

(ب) وقفهی حسگر PPG: هنگامی که حسگر داده‌ی جدیدی داشته باشد، با فعال کردن یک پایه، وقفه‌ای را فعال می‌کند. در روتین این وقفه داده‌های حسگر قرائت می‌شود و به ترتیب در یک آرایه ذخیره می‌شوند. هنگامی که این آرایه پر شود، توسط بلوتوث به گوشی ارسال می‌گردد. شکل ۲۴-۴ بخش مربوط به ذخیره در آرایه و ارسال آن را نشان می‌دهد.

```

202 void append_PPG_buffer(System* sys, uint32_t ir, uint32_t red){
203     uint8_t irLSB, irMSB, redLSB, redMSB;
204
205     irLSB = (uint8_t)((ir & 0x000000FF) >> 0);
206     irMSB = (uint8_t)((ir & 0x0000FF00) >> 8);
207
208     redLSB = (uint8_t)((red & 0x000000FF) >> 0);
209     redMSB = (uint8_t)((red & 0x0000FF00) >> 8);
210
211     sys->TxBuffer[ 10 + sys->ppg_data_address_pointer*4 + 0 ] = irLSB;
212     sys->TxBuffer[ 10 + sys->ppg_data_address_pointer*4 + 1 ] = irMSB;
213     sys->TxBuffer[ 10 + sys->ppg_data_address_pointer*4 + 2 ] = redLSB;
214     sys->TxBuffer[ 10 + sys->ppg_data_address_pointer*4 + 3 ] = redMSB;
215
216     if(sys->ppg_data_address_pointer != 199){
217         sys->ppg_data_address_pointer++;
218     }
219     else{
220         sys->ppg_data_address_pointer=0;
221         TxGenerator(sys);
222     }
223 }
```

شکل ۲۴-۴: نمونه کدی از بخش ذخیره‌ی داده‌ی حسگر سلامت

۲. وقفه‌ی تایمر: همانطور که گفته شد سه تایمر از هفت تایمر برای فعالسازی وقفه به کار رفته‌اند:

(آ) تایمر فیلتر کالمن: این تایمر هر یک میلی ثانیه وقفه را فعال می‌کند. در روتین وقفه ابتدا اطلاعات حسگر حرکتی خوانده می‌شود. سپس این اطلاعات برای فیلتر شدن به توابع کالمن داده می‌شود. شرح این توابع در بخش‌هایی بعدی آورده شده است. سپس اطلاعات فیلتر شده وارد مرحله‌ی پردازش می‌شوند. این پردازش‌ها شامل تشخیص سرعت زاویه‌ای دست حول آرنج و تشخیص گام است. کد این بخش در شکل ۲۵-۴ قابل مشاهده است.

(ب) تایمر صفحه نمایش: هرگاه وقفه‌ی این تایمر فعال شود به این معنی است که چهار ثانیه از آخرین تعامل کاربر با ساعت گذشته است و باید صفحه نمایش خاموش شود. شکل ۲۶-۴ کد این قسمت را نشان می‌دهد.

(ج) تایمر اصلی سیستم: این تایمر هر 25° میلی ثانیه سرریز می‌شود. در نتیجه هر 25° میلی ثانیه روتین وقفه اجرا می‌شود. در روتین وقفه‌ی این تایمر ابتدا بررسی می‌شود که آیا صفحه نمایش باید روشن شود یا خیر. اگر نیاز به روشن شدن بود، با توجه به حالت سیستم و ماشین حالتی که در بخش ۱-۲-۴ توضیح داده شد، روتین حالت مربوطه را اجرا می‌کند.

۳. وقفه‌ی آلام: هنگامی که ساعت و روز بر ساعت و روز آلام منطبق شود، این وقفه فعال می‌گردد. در روتین این وقفه پرچم آلام یک می‌شود، تایمر کالمن خاموش می‌شود، صفحه نمایش روشن می‌شود و حالت سیستم به آلام تغییر می‌کند.

```

382 // Filtering -----
383 /* Accelerometer */
384 KAccel.z_x = myAccelRaw.x;
385 #if ALLXYZ_MODE
386     KAccel.z_y = myAccelRaw.y;
387     KAccel.z_z = myAccelRaw.z;
388 #endif
389 kalman_scaler_update(&KAccel);
390
391 /* Gyroscope */
392 #if ALLXYZ_MODE
393     KGyro.z_x = myGyroRaw.x;
394     KGyro.z_z = myGyroRaw.z;
395 #endif
396     KGyro.z_y = myGyroRaw.y;
397     kalman_2d1r_update(&KGyro);
398
399 /* Pedometer */
400 if((KAccel.x_hat >= 1000)&&(!getPedFlag(&sys))){
401     setPedFlag(&sys,1);
402     sys.pedometer++;
403     //DEBUG("%d\n\r",sys.pedometer);
404 }
405 else if((KAccel.x_hat < 100)&&(getPedFlag(&sys))){
406     setPedFlag(&sys,0);
407 }
408
409 /* Enable screen */
410 if((KGyro.wy_hat >= 4000)&&(!getScreenFlag(&sys))){
411     setScreenFlag(&sys,1);
412 }
413 else if((KAccel.x_hat < 400)&&(getScreenFlag(&sys))){
414     setScreenFlag(&sys,0);
415     HAL_TIM_Base_Start_IT(&SCREEN_TIMER);
416     setScreenEable(&sys,1);
417 }
418
419 }
420

```

شکل ۴-۲۵: نمونه کدی از بخش وقفه‌ی کالمن

```

504 } else if(htim == &SCREEN_TIMER){
505     HAL_TIM_Base_Stop(&SCREEN_TIMER);
506     setScreenEable(&sys, 0);
507     SCREEN_TIMER_RESET;
508 }
509

```

شکل ۴-۲۶: کد روتین وقفه‌ی تایمر صفحه نمایش

۴. وقفه‌ی DMA مربوطه به ADC: تنظیمات DMA مربوط به ADC به گونه‌ای است که هر ۱۰ میلی ثانیه یک نمونه می‌گیرد و آن را در آرایه‌ای ذخیره می‌کند. هرگاه این آرایه که طول آن ۱۲۸ است پر شود، این وقفه فعال می‌شود. در روتین وقفه می‌خواهیم میانگین این ۱۲۸ نمونه را به عنوان عددنهایی گزارش کنیم. میانگین ساده‌ترین راه جلوگیری از ورود نویز به سیستم است.

پس مطابق شکل ۲۷-۴ ابتدا مجموع این ۱۲۸ داده را حساب می‌کنیم. سپس باید این حاصل را بر تعداد نمونه‌ها تقسیم کنیم. فلسفه‌ی انتخاب عدد ۱۲۸ برای تعداد نمونه‌ها اینجا مشخص می‌شود. می‌دانیم تقسیمات اعشاری به دلیل پیچیدگی‌های محاسباتی، مقداری زیادی از حافظه را اشغال می‌کنند و سرعت کمتری دارند. پس بهتر است تا جای ممکن از آن‌ها دوری کنیم.

از طرفی می‌دانیم که اگر مقسوم‌علیه توان دو باشد، می‌توان عملیات تقسیم را با شیفت مقسوم انجام داد. در اینجا جای اینکه مجموع را بر ۱۲۸ تقسیم کنیم، کافی است آن را ۷ بیت به سمت راست شیفت دهیم تا تقسیم انجام شود. اینگونه هم سرعت بسیار بالایی دارد هم حافظه را بیهوده اشغال نمی‌کند.

```

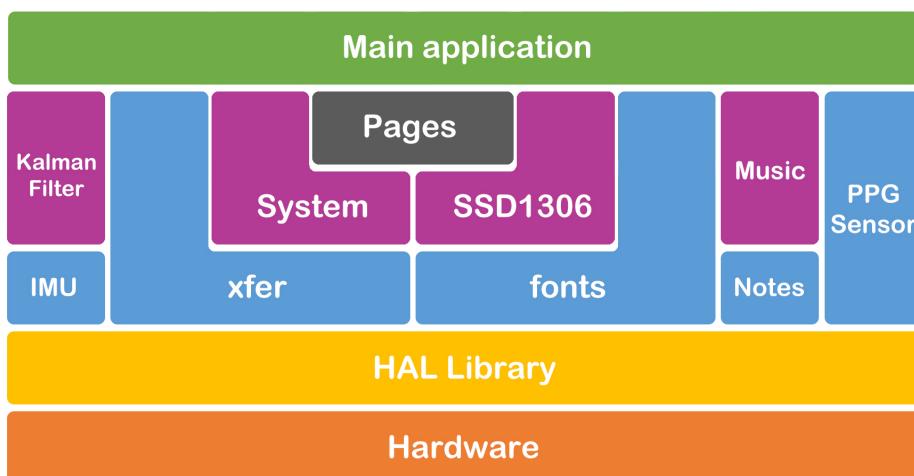
708 void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDefDef* hadc){
709     sum = 0;
710     for(uint32_t i = 0; i<128; i++)
711         sum += adcVals[i];
712     sys.voltage = sum >> 7;
713 }
714 }
```

شکل ۲۷-۴: کد روتین وقفه‌ی DMA مربوطه به ADC

۵. وقفه‌ی DMA مربوطه به USART: تنظیمات DMA مربوط به گونه‌ای است که هرگاه تعداد ۲۱ بایت داده توسط واحد ارتباط سریال (بلوتوث) دریافت شود، وقفه‌ی DMA مربوطه فعال می‌شود. در روتین این وقفه این داده تحلیل می‌شود و بسته به نوع آن، اقدام مناسب با آن صورت می‌گیرد. جزئیات مربوط به بسته‌ها و نحوه‌ی کدگشایی آن‌ها در بخش ۴-۴ توضیح داده خواهد شد.

۳-۲-۴ لایه‌های نرم افزاری

ساختار کد به صورت لایه‌ای است. به این معنی که هر قطعه کد یا کتابخانه از توابع موجود در کتابخانه‌های لایه‌ی پایین استفاده می‌کند. شکل ۲۸-۴ نحوه‌ی لایه‌بندی این پروژه را نشان می‌دهد.



شکل ۲۸-۴: لایه‌های نرم افزاری

پایین‌ترین لایه، لایه‌ی سخت‌افزار است. رجیسترها که برای کار با سخت‌افزار مورد نیاز است در این لایه قرار دارد. از آنجا کار با رجیسترها به صورت مستقیم اگرچه در مواردی بهتر و بهینه‌تر است، اما دشواری‌هایی دارد که در اکثر موارد مانع استفاده از آن می‌شود. این پروژه نیز از این قاعده مستثنی نیست. برای ساده کردن کار با سخت‌افزار، از یک کتابخانه به نام HAL^{۱۴} بهره بردیم که به کمک توابع موجود در آن کار با سخت‌افزار ساده می‌شود. بقیه‌ی لایه‌ها روی لایه‌ی HAL سوار می‌شوند. دیگر لایه‌ها شامل کتابخانه‌هایی است که برای بخش‌های مختلف پروژه نوشته شده است. در ادامه به صورت کلی به توضیح این کتابخانه‌ها پرداخته می‌شود.

^{۱۴} Hardware Abstraction Layer

کتابخانه xfer

این کتابخانه شامل دو تابع است که وظیفه ارسال داده ها از طریق سخت افزار را برعهده دارد. تصویر این توابع در شکل ۲۹-۴ دیده می شود.

```

1 #include "xfer.h"
2
3 void rawXfer(uint8_t* data, uint16_t length){
4     #if BLE_DEBUG
5         HAL_UART_Transmit(&BLUETOOTH_UART, data, length, 1000);
6     #else
7         HAL_UART_Transmit(&DEBUG_UART, data, length, 5000);
8     #endif
9 }
10
11 void DEBUG(const char* _str, ...){
12     #if DEBUG_ENABLE
13         va_list args;
14         va_start(args, _str);
15         char buffer[50];
16         memset(buffer, 0, 50);
17         int buffer_size = vsprintf(buffer, _str, args);
18         #if BLE_DEBUG
19             HAL_UART_Transmit(&BLUETOOTH_UART, (uint8_t*)buffer, buffer_size, 1000);
20         #else
21             HAL_UART_Transmit(&DEBUG_UART, (uint8_t*)buffer, buffer_size, 5000);
22         #endif
23     #endif
24 }

```

شکل ۲۹-۴: توابع کتابخانه xfer

کتابخانه fonts

این کتابخانه شامل فونت ها و اشکالی است که روی صفحه نمایش نشان داده خواهد شد. تصویر بخشی از این کدها در شکل ۳۰-۴ دیده می شود.

```

191 0x0000, // v 86 67
192 0x0000, // w 87 68
193 0x0000, // x 88 69
194 0x0000, // y 89 70
195 0x0000, // z 90 71
196 0x0388, 0x0780, 0x0600, 0x0500, 0x0400, 0x0300, 0x0200, 0x0100, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, // { 91 0
197 0x0000, 0x0600, // | 92 72
198 0x3300, 0x1C00, 0x0C00, // } 93 0
199 0x0000, // ~ 94 0
200 ;
201 */
202 static const uint16_t Font16x26 [] = {
203     0x0000, // Ascii = []
204     0x0000, // Ascii = []
205     0x03E0, // Ascii = []
206     0x1E00, // Ascii = []
207     0x1E1C, // Ascii = []
208     0x03FC, 0x03FC, 0x01FE, 0x01E9, // Ascii = [ ]
209     0x03E0, 0x0707, 0x0E70, // Ascii = [ ]
210     0x07E0, 0x0F80, 0x0F78, // Ascii = [ ]
211     0x03E0, // Ascii = [ ]
212     0x03E0, // Ascii = [ ]
213     0x07E0, 0x07C0, // Ascii = [ ]
214     0x03E0, 0x03C0, // Ascii = [ ]
215     0x0000, // Ascii = []
216     0x0000, // Ascii = []

```

شکل ۳۰-۴: بخشی از کتابخانه fonts

کتابخانه notes

این کتابخانه شامل نگاشتی از نت‌های موسیقی به فرکانس آن‌ها است. تصویر بخشی از این کدها در شکل ۳۱-۴ دیده می‌شود.

```

44 #define NOTE_FS4 370
45 #define NOTE_G4 392
46 #define NOTE_GS4 415
47 #define NOTE_A4 440
48 #define NOTE_AS4 466
49 #define NOTE_B4 494
50 #define NOTE_C5 523
51 #define NOTE_CS5 554

```

شکل ۳۱-۴: بخشی از کتابخانه notes

کتابخانه max30102

این کتابخانه برای کار با حسگر PPG است که توسط آقای Pan نوشته شده است [۳]. شکل ۳۲-۴ نام توابع موجود در این کتابخانه را نشان می‌دهد.

```

123 __weak void max30102_plot(uint32_t ir_sample, uint32_t red_sample);
124
125 void max30102_init(max30102_t *obj, I2C_HandleTypeDef *hi2c);
126 void max30102_write(max30102_t *obj, uint8_t reg, uint8_t *buf, uint16_t buflen);
127 void max30102_read(max30102_t *obj, uint8_t reg, uint8_t *buf, uint16_t buflen);
128
129 void max30102_reset(max30102_t *obj);
130
131 void max30102_set_a_full(max30102_t *obj, uint8_t enable);
132 void max30102_set_ppg_rdy(max30102_t *obj, uint8_t enable);
133 void max30102_set_alc_ovf(max30102_t *obj, uint8_t enable);
134 void max30102_set_die_temp_rdy(max30102_t *obj, uint8_t enable);
135 void max30102_set_die_temp_en(max30102_t *obj, uint8_t enable);
136
137 void max30102_on_interrupt(max30102_t *obj);
138 uint8_t max30102_has_interrupt(max30102_t *obj);
139 void max30102_interrupt_handler(max30102_t *obj);
140
141 void max30102_shutdown(max30102_t *obj, uint8_t shdn);
142
143 void max30102_set_mode(max30102_t *obj, max30102_mode_t mode);
144 void max30102_set_sampling_rate(max30102_t *obj, max30102_sr_t sr);
145
146 void max30102_set_led_pulse_width(max30102_t *obj, max30102_led_pw_t pw);
147 void max30102_set_adc_resolution(max30102_t *obj, max30102_adc_t adc);
148
149 void max30102_set_led_current_1(max30102_t *obj, float ma);
150 void max30102_set_led_current_2(max30102_t *obj, float ma);
151 void max30102_set_multi_led_slot_1_2(max30102_t *obj, max30102_multi_led_ctrl_t slot1, max30102_multi_led_ctrl_t slot2);
152 void max30102_set_multi_led_slot_3_4(max30102_t *obj, max30102_multi_led_ctrl_t slot3, max30102_multi_led_ctrl_t slot4);
153
154 void max30102_set_fifo_config(max30102_t *obj, max30102_smp_ave_t smp_ave, uint8_t roll_over_en, uint8_t fifo_a_full);
155 void max30102_clear_fifo(max30102_t *obj);
156 void max30102_read_fifo(max30102_t *obj);
157
158 void max30102_read_temp(max30102_t *obj, int8_t *temp_int, uint8_t *temp_frac);
159

```

شکل ۳۲-۴: پروتوتایپ توابع کتابخانه MAX30102

کتابخانه MPU6050

این کتابخانه برای کار با حسگر MPU6050 است که توسط آقای محمد یعقوب نوشته شده است
[۱]. شکل ۳۳-۴ نام توابع موجود در این کتابخانه را نشان می‌دهد.

```

123 //Function Prototype
124 //1- i2c Handler
125 void MPU6050_Init(I2C_HandleTypeDef *I2Chnd);
126 //2- i2c Read
127 void I2C_Read(uint8_t ADDR, uint8_t *i2cBuf, uint8_t NofData);
128 //3- i2c Write 8 Bit
129 void I2C_Write8(uint8_t ADDR, uint8_t data);
130 //4- MPU6050 Initialization Configuration
131 void MPU6050_Config(MPU_ConfigTypeDef *config);
132 //5- Get Sample Rate Divider
133 uint8_t MPU6050_Get_SMPRT_DIV(void);
134 //6- Set Sample Rate Divider
135 void MPU6050_Set_SMPRT_DIV(uint8_t SMPRTvalue);
136 //7- External Frame Sync.
137 uint8_t MPU6050_Get_FSYNC(void);
138 //8- Set External Frame Sync.
139 void MPU6050_Set_FSYNC(enum EXT_SYNC_SET_ENUM ext_Sync);
140 //9- Get Accel Raw Data
141 void MPU6050_Get_RawData(RawData_Def *rawDefA, RawData_Def *rawDefG);
...

```

شکل ۳۳-۴: پروتوتایپ توابع کتابخانه MPU6050

کتابخانه music

در این کتابخانه موسیقی «ای ایران ای مرز پرگهر» به صورت نت‌های موسیقی نوشته شده است تا هنگام فعال شدن آلارم نواخته شود. شکل ۳۴-۴ این نت‌ها را نشان می‌دهد.

```

3 int melody[] = {
4 NOTE_D4, NOTE_A4, NOTE_A4, NOTE_A4, NOTE_D4,
5 NOTE_D4, NOTE_A4, NOTE_A4, NOTE_A4, NOTE_D4,
6 NOTE_A4, NOTE_AS4, NOTE_C5, NOTE_A4,
7 NOTE_D5, NOTE_C5, NOTE_C5, NOTE_AS4, NOTE_AS4, NOTE_A4, NOTE_A4, NOTE_G4,
8 NOTE_F4, NOTE_G4, NOTE_C5, NOTE_A4,
9 NOTE_D4, NOTE_C4, NOTE_D4, NOTE_E4, NOTE_F4, NOTE_E4, NOTE_F4, NOTE_G4,
10 NOTE_A4, NOTE_C5, NOTE_AS4, NOTE_A4, NOTE_G4, NOTE_F4, NOTE_E4, NOTE_D4,
11 NOTE_C4, NOTE_G4, NOTE_G4, NOTE_A4, NOTE_A4,
12 NOTE_AS4, NOTE_A4, NOTE_G4, NOTE_F4, NOTE_G4,
13 NOTE_A4, NOTE_G4, NOTE_F4, NOTE_E4, NOTE_F4,
14 NOTE_G4, NOTE_F4, NOTE_E4, NOTE_D4,
15 NOTE_AS4, NOTE_A4, NOTE_A4, NOTE_G4, NOTE_G4, NOTE_F4, NOTE_F4, NOTE_E4,
16 NOTE_A4, NOTE_G4, NOTE_G4, NOTE_F4, NOTE_F4, NOTE_E4, NOTE_E4, NOTE_D4,
17 NOTE_F4, NOTE_F4, NOTE_G4, NOTE_A4
18 };
19
20 int noteDurations[] = {
21 4, 8, 8, 4, 4,
22 4, 8, 8, 4, 4,
23 4, 8, 8, 2,
24 8, 8, 8, 8, 8, 8, 8,
25 4, 8, 8, 2,
26 8, 8, 8, 8, 8, 8, 8,
27 8, 8, 8, 8, 8, 8, 8,
28 4, 8, 8, 4, 4,
29 4, 8, 8, 4, 4,
30 4, 8, 8, 4, 4,
31 4, 8, 8, 2,
32 8, 8, 8, 8, 8, 8, 8,
33 8, 8, 8, 8, 8, 8, 8,
34 4, 8, 8, 2
35 };

```

شکل ۳۴-۴: نت‌های «ای ایران»

کتابخانه‌ی SSD1306

Olivier Van den Eede این کتابخانه برای کار با صفحه نمایش‌های با درایور SSD1306 است که توسط آقای Olivier Van den Eede نوشته شده است [۲]. شکل ۳۵-۴ نام توابع موجود در این کتابخانه را نشان می‌دهد.

```

59 |     uint8_t ssd1306_Init(I2C_HandleTypeDef *hi2c);
60 |     void ssd1306_UpdateScreen(I2C_HandleTypeDef *hi2c);
61 |     void ssd1306_Fill(SSD1306_COLOR color);
62 |     void ssd1306_DrawPixel(uint8_t x, uint8_t y, SSD1306_COLOR color);
63 |     void ssd1306_DrawShape(uint8_t x, uint8_t y, ShapeDef shape);
64 |     char ssd1306_WriteChar(char ch, FontDef Font, SSD1306_COLOR color);
65 |     char ssd1306_WriteString(char* str, FontDef Font, SSD1306_COLOR color);
66 |     void ssd1306_SetCursor(uint8_t x, uint8_t y);
67 |     void ssd1306_InvertColors(void);

```

شکل ۳۵-۴: پروتوتایپ توابع کتابخانه‌ی SSD1306

کتابخانه‌ی Kalman

در این کتابخانه توابع و روابط فیلتر کالمن پیاده‌سازی شده است. توضیحات مفصل این توابع در بخش ۳-۴ داده خواهد شد. شکل ۳۶-۴ نام توابع موجود در این کتابخانه را نشان می‌دهد.

```

65 |     void kalman_scaler_init(KalmanScaler* hkalman, float Q, float R);
66 |     void kalman_2d1r_init(Kalman2D1R* hkalman, float Q, float R);
67 |
68 |     void kalman_scaler_update(KalmanScaler* hkalman);
69 |     void kalman_2d1r_update(Kalman2D1R* hkalman);

```

شکل ۳۶-۴: پروتوتایپ توابع کتابخانه‌ی kalman

کتابخانه‌ی Pages

در این کتابخانه توابعی که بسته به حالت سیستم، صفحه‌ی مورد نظر را روی صفحه نمایش ترسیم می‌کنند. شکل ۳۷-۴ نام توابع موجود در این کتابخانه را نشان می‌دهد.

```

9  |     void render_ack_waiting(System* sys);
10 |     void render_home(System* sys);
11 |     void render_connected(System* sys);
12 |     void render_bloody_hell(System* sys);
13 |     void render_pedometo(System* sys);
14 |     void render_about(System* sys);
15 |     void render_ringing(System* sys);

```

شکل ۳۷-۴: پروتوتایپ توابع کتابخانه‌ی pages

کتابخانه System

می توان گفت این کتابخانه حیاتی ترین کتابخانه توابع مربوط به سیستم قرار دارد. همچنین یک ساختار^{۱۵} تعریف شده تا متغیرهای سیستمی را شامل شود. شکل ۴-۳۸ نام توابع موجود در این کتابخانه و شکل ۴-۳۸-ب تعریف این ساختار را نشان می دهد.

```

69 // Constructor
70 System sys_init(void);
71
72 // Get Flags
73 uint8_t getConnected(System* sys);
74 uint8_t getRinging(System* sys);
75 uint8_t getSHReached(System* sys);
76 uint8_t getScreenFlag(System* sys);
77 uint8_t getScreenEable(System* sys);
78 uint8_t getPedFlag(System* sys);
79 uint8_t getVibEnable(System* sys);
80 uint8_t getSoundEnable(System* sys);
81
82 // SetFlags
83 void setConnected(System* sys, uint8_t state);
84 void setRinging(System* sys, uint8_t state);
85 void setSHReached(System* sys, uint8_t state);
86 void setScreenFlag(System* sys, uint8_t state);
87 void setScreenEable(System* sys, uint8_t state);
88 void setPedFlag(System* sys, uint8_t state);
89 void setVibEnable(System* sys, uint8_t state);
90 void setSoundEnable(System* sys, uint8_t state);
91
92 // Main functions
93 void RxParser(System* sys);
94 void TxGenerator(System* sys);
95 void addAck2buffer(System* sys);
96 void addPedometer2buffer(System* sys);
97 void addBattery2buffer(System* sys);
98 void clear_PPG_buffer(System* sys);
99 void append_PPG_buffer(System* sys, uint32_t ir, uint32_t red);
100 void setAlarm(uint8_t Hours, uint8_t Minutes, uint8_t Seconds, uint8_t weekday, uint32_t masks);
101 float map(uint16_t x, uint16_t in_min, uint16_t in_max, uint16_t out_min, uint16_t out_max);
102

```

(آ) پروتوتایپ توابع کتابخانه system

```

27 enum system_states {ack_waiting, hello, home, bloody_hell, pedomodo, about, ringing};
28
29 typedef struct System_t{
30
31     uint32_t pedometer;
32     float pace_size;
33
34     uint16_t voltage;
35     uint8_t battery;
36
37     uint16_t ppg_data_address_pointer;
38     uint8_t heart_beat;
39     float SpO2;
40
41     RTC_DateTypeDef Date;
42     RTC_TimeTypeDef Time;
43
44     uint8_t RxBuffer[RX_BUFFER_SIZE];
45     uint8_t TxBuffer[TX_BUFFER_SIZE];
46
47     uint8_t flags;
48     // Flags:
49     // -----
50     // | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
51     // | isConnected | isRinging | isSHReached | screenFlag | screenEnable | pedFlag | vibEnable | soundEnable |
52     // -----
53
54
55     int8_t display_calibration_x;
56     int8_t display_calibration_y;
57
58     enum system_states state;
59
60     char user_name[18];
61     uint8_t name_xpos;
62
63     uint8_t ack;
64
65 } System;
66

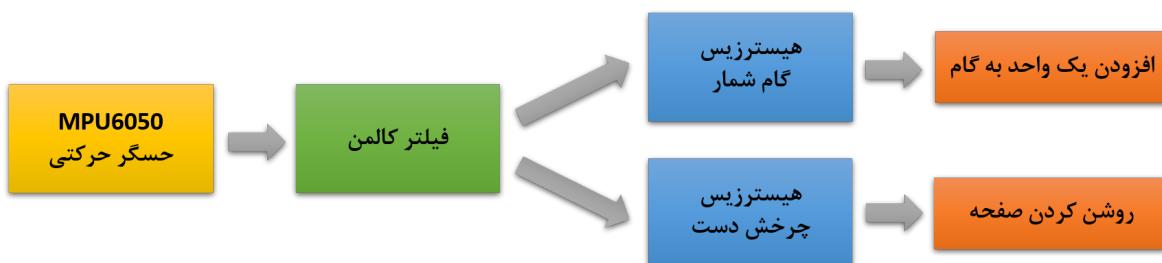
```

(ب) متغیرهای struct سیستم

شکل ۴-۳۸: تصاویر مربوطه به کتابخانه system

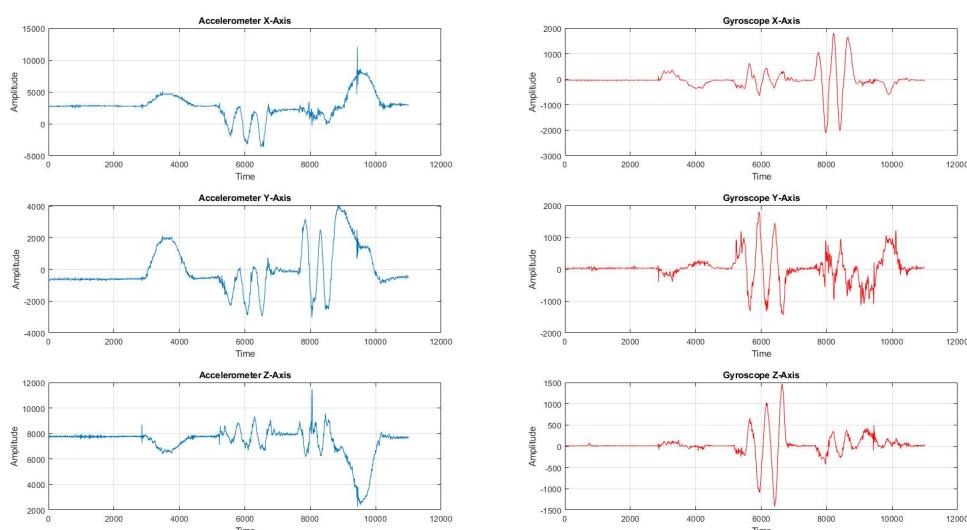
۳-۴ پردازش داده های حسگر حرکتی

دو مورد از وظایفی که برای ساعت تعریف شده است، شمردن تعداد گامها و روشن کردن صفحه نمایش در صورت بالا آمدن دست است. برای انجام این وظایف از حسگر حرکتی MPU6050 استفاده شده است. اما می دانیم ادوات اندازه گیری دارای نویز و نایقینی های مختلف هستند. حسگر فوق نیز از این قاعده مستثنی نیست. لذا قبل از هر چیز باید داده های آن فیلتر شود تا نویزو اثرات مخرب دیگر تا جای ممکن بر پردازش ها اثر نگذارند. به این منظور یک فیلتر کالمن طراحی و پیاده سازی شده است. شکل ۳۹-۴ بلوک دیاگرام زیر سیستم تشخیص حرکت را نشان می دهد.



شکل ۳۹-۴: بلوک دیاگرام زیر سیستم تشخیص حرکت

برای طراحی بلوک دیاگرام فوق، ابتدا نویز حسگر را بشناسیم. بدین منظور چند ثانیه از داده های حسگر توسط ساعت نمونه برداری شد و توسط بلوتوث به نرم افزار متلب ارسال شد. این نمونه ها به مدت یازده ثانیه ضبط شدند. در این مدت حرکات تصادفی و متنوعی را به ساعت اعمال کردم. شکل ۴۰-۴ خروجی حسگر را در این مدت نشان می دهد.



شکل ۴۰-۴: خروجی حسگر حرکتی. سه محور مربوط به شتاب خطی و سه محور مربوط به سرعت زاویه ای

همانطور که در خروجی حسگر مشهود است، مقدار نویز قابل توجه است و می‌تواند پردازش‌ها را دچار خطا کند. از این رو به طراحی فیلتر کالمن می‌پردازیم.

۱-۳-۴ فیلتر کالمن

گام اول طراحی فیلتر کالمن، مدلسازی سیستم است. حسگر حرکتی سیستمی است کنترل نشده. لذا حلقه‌ی کنترل‌ای در کار نیست. می‌توان گفت ورودی این حسگر شتاب حرکت دست است. در نتیجه سیستم شتاب سنج حسگر را می‌توان به صورت معادله‌ی ۱-۴ مدل کرد که در آن x موقعیت، v سرعت خطی، a شتاب خطی و T فاصله‌ی زمانی دو نمونه یا همان تناوب نمونه‌برداری یا گسترش‌سازی است.

$$x[n+1] = \frac{1}{\zeta} a[n]T^2 + v[n]T + x[n] \quad (1-4)$$

سیستم سرعت زاویه‌ای نیز به طور مشابه به صورت معادله‌ی ۲-۴ مدل می‌شود که در آن θ زاویه، ω سرعت زاویه‌ای، α شتاب زاویه‌ای و T فاصله‌ی زمانی دو نمونه یا همان تناوب نمونه‌برداری یا گسترش‌سازی است.

$$\theta[n+1] = \frac{1}{\zeta} \alpha[n]T^2 + \omega[n]T + \theta[n] \quad (2-4)$$

از آنجا که حرکت دست، یا به عبارتی تغییرات شتاب خطی/زاویه‌ای نامشخص است، می‌توان آن را به صورت اغتشاش ورودی مدل کرد. در نتیجه تغییرات شتاب اغتشاشی است که به سیستم وارد می‌شود تا متغیرهای سیستم (موقعیت و سرعت) را تغییر دهد. بنابراین می‌توان فرم فضای حالت را برای این سیستم‌ها به صورت معادلات ۳-۴ و ۴-۴ نوشت:

$$\begin{bmatrix} x[n+1] \\ v[n+1] \\ a[n+1] \end{bmatrix} = \begin{bmatrix} 1 & T & \frac{1}{\zeta} T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x[n] \\ v[n] \\ a[n] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w_{linear}[n] \quad (3-4)$$

$$\begin{bmatrix} \theta[n+1] \\ \omega[n+1] \\ \alpha[n+1] \end{bmatrix} = \begin{bmatrix} 1 & T & \frac{1}{\zeta} T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta[n] \\ \omega[n] \\ \alpha[n] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w_{rotational}[n] \quad (4-4)$$

گام دوم مدلسازی مشاهده از حالت‌های سیستم است. همانطور که از معادلات فوق مشخص است، ماتریس‌های حالت (G) و ورودی (ϕ) در هر دو حالت خطی و زاویه‌ای یکسان هستند و می‌توان مدل یکسانی به هر دو حرکت نسبت داد. اما از طرفی می‌دانیم که خروجی حسگر برای حالت خطی، شتاب و برای حالت زاویه‌ای، سرعت است. در نتیجه معادلات استاتیکی مشاهده به شکل معادلات ۵-۴ و ۶-۴

خواهند بود.

$$z_{linear} = [0 \ 0 \ 1] \begin{bmatrix} x \\ v \\ a \end{bmatrix} + v_{linear} \quad (5-4)$$

$$z_{rotational} = [0 \ 1 \ 0] \begin{bmatrix} \theta \\ \omega \\ \alpha \end{bmatrix} + v_{rotational} \quad (6-4)$$

نکته‌ی مهمی در رابطه با حسگر سرعت زاویه‌ای وجود دارد. حسگرهای MEMS^{۱۶} که برای اندازه‌گیری سرعت زاویه‌ای استفاده می‌شوند، یک عبارت ثابت^{۱۷} را در خروجی با مقدار واقعی جمع می‌کنند. این عبارت بایاس را می‌توان به عنوان یک حالت جدید به مدل سیستم افزود. در این حالت مدل نهایی به صورت معادلات ۷-۴ و ۸-۴ خواهد بود.

$$\begin{bmatrix} \theta[n+1] \\ \omega[n+1] \\ \alpha[n+1] \\ b[n+1] \end{bmatrix} = \begin{bmatrix} 1 & T & \frac{1}{2}T^2 & 0 \\ 0 & 1 & T & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta[n] \\ \omega[n] \\ \alpha[n] \\ b[n] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} w_{rotational}[n] \quad (7-4)$$

$$z_{rotational} = [0 \ 1 \ 0 \ 1] \begin{bmatrix} \theta \\ \omega \\ \alpha \\ b \end{bmatrix} + v_{rotational} \quad (8-4)$$

در این پژوهه لازم است تا صرفا الگوی «کلی» تغییرات شتاب خطی و الگوی «کلی» تغییرات سرعت زاویه‌ای مورد بررسی قرار گیرند. از این رو فرض‌های ذیل معقول‌اند:

۱. عبارت بایاس مقدار کوچکی است؛ از طرفی تأثیری نیز در الگوی تغییرات ندارد لذا می‌توان از آن صرف‌نظر کرد.
۲. در تحلیل حرکت خطی، نیازی به تخمین موقعیت و سرعت نیست و صرفا شتاب اهمیت دارد، لذا می‌توان حالت‌های موقعیت و سرعت را از معادله‌ی حالت حذف کرد.
۳. در تحلیل حرکت دورانی نیازی به تخمین زاویه نیست و صرفا سرعت زاویه‌ای اهمیت دارد. لذا می‌توان حالت زاویه را از معادله‌ی حالت حذف کرد.

با اعمال مفروضات فوق، معادلات حرکت خطی به شکل معادله‌ی ۹-۴ و معادلات حرکت دورانی به شکل معادله‌ی ۱۰-۴ در می‌آیند.

$$\begin{aligned} a[n+1] &= a[n] + w_{linear}[n] \\ z_{rotational} &= a + v_{linear} \end{aligned} \quad (9-4)$$

$$\begin{aligned} \begin{bmatrix} \omega[n+1] \\ \alpha[n+1] \end{bmatrix} &= \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega[n] \\ \alpha[n] \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_{rotational}[n] \\ z_{rotational} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \omega \\ \alpha \end{bmatrix} + v_{rotational} \end{aligned} \quad (10-4)$$

گام سوم طراحی معادلات کالمن برای دو سیستم بالا است. معادلات کالمن در حالت کلی به صورت [۵] است ۱۱-۴.

$$\begin{aligned} \hat{x}(N+1) &= \phi(N)\hat{x}(N) + K(N)(z(N) - H(N)\phi(N)\hat{x}(N)) \\ K &= \Sigma(N+1|N)H(N)^T(H(N)\Sigma(N+1|N)H(N)^T + R(N))^{-1} \\ \Sigma(N+1|N) &= \phi(N)\Sigma(N|N)\phi(N)^T + G(N)Q(N)G(N)^T \\ \Sigma(N+1|N+1) &= (I - K(N)H(N))\Sigma(N+1|N)(I - K(N)H(N))^T + K(N)R(N)K(N)^T \end{aligned} \quad (11-4)$$

از آنجا که ماتریس‌های دو سیستم متفاوت هستند، هر کدام مستلزم بکارگیری معادلات متفاوتی برای فیلتر شدن هستند. لذا هر کدام جداگانه شرح داده می‌شوند.

- حرکت خطی:

از آنجا که معادلات حرکت خطی به صورت اسکالر درآمده‌اند و از طرفی سیستم نیز نامتغیر با زمان است، می‌توان ماتریس‌های سیستم را به صورت ذیل خلاصه کرد:

$$\phi = 1 \quad , \quad G = 1 \quad , \quad H = 1 \quad (12-4)$$

در صورتی که مقادیر ۱۲-۴ را در ۱۱-۴ جایگذاری کنیم، با کمی ساده سازی به معادلات ۱۳-۴ می رسیم.

$$\begin{aligned}\hat{a}[n+1] &= \frac{R}{\Sigma(N|N) + Q + R} \hat{a}[n] + \frac{\Sigma(N|N) + Q}{\Sigma(N|N) + Q + R} z[n] \\ \Sigma(N+1|N+1) &= \frac{(\Sigma(N|N) + Q)R}{\Sigma(N|N) + Q + R}\end{aligned}\quad (13-4)$$

این معادلات برای پیاده سازی در ریز پردازنده مناسباند و به راحتی اجزا می شوند.

- حرکت دورانی:

سیستم حرکت دورانی یک سیستم مرتبه دوی نام تغیر با زمان است. لذا ماتریس های سیستم به صورت زیر خلاصه می شود:

$$\phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad (14-4)$$

```

99 void kalman_2dir_update(Kalman2DIR* hkalman){
100    float t = SAMPLING_TIME;
101
102    #if LONNEM_MODE==0
103        hkalman->Pnn1[0][0] = hkalman->pre_Pnn1[0][0] + 2*I*hkalman->pre_Pnn1[0][1] + I*I*hkalman->pre_Pnn1[1][1];
104        hkalman->Pnn1[0][1] = hkalman->pre_Pnn1[0][1] + I*hkalman->pre_Pnn1[1][1];
105        hkalman->Pnn1[1][0] = hkalman->pre_Pnn1[1][0];
106        hkalman->Pnn1[1][1] = hkalman->pre_Pnn1[1][1] + hkalman->Q;
107
108        hkalman->X[0] = hkalman->Pnn1[0][0]/((float)(hkalman->Pnn1[0][0] + hkalman->R));
109        hkalman->X[1] = hkalman->Pnn1[1][0]/((float)(hkalman->Pnn1[0][0] + hkalman->R));
110    #else
111        hkalman->X[0] = 0.024835;
112        hkalman->X[1] = 0.312276;
113    #endif
114
115    #if ALLXYZ_MODE
116        hkalman->wx_hat = hkalman->X[0]*hkalman->wx_hat + hkalman->wx_hat - hkalman->X[0]*I*hkalman->dxw_hat + I*hkalman->dxw_hat + hkalman->X[0]*hkalman->z_x;
117        hkalman->dxw_hat = hkalman->X[1]*hkalman->wx_hat - hkalman->X[1]*I*hkalman->dxw_hat + hkalman->dxw_hat + hkalman->X[1]*hkalman->z_x;
118        hkalman->pre_wx_hat = hkalman->wx_hat;
119        hkalman->pre_dxw_hat = hkalman->dxw_hat;
120
121        hkalman->wy_hat = hkalman->X[0]*hkalman->wy_hat + hkalman->wy_hat - hkalman->X[0]*I*hkalman->dwy_hat + I*hkalman->dwy_hat + hkalman->X[0]*hkalman->z_y;
122        hkalman->dyw_hat = hkalman->X[1]*hkalman->wy_hat - hkalman->X[1]*I*hkalman->dwy_hat + hkalman->dwy_hat + hkalman->X[1]*hkalman->z_y;
123        hkalman->pre_wy_hat = hkalman->wy_hat;
124        hkalman->pre_dyw_hat = hkalman->dyw_hat;
125    #endif
126
127    hkalman->wy_hat = -hkalman->X[0]*hkalman->wy_hat + hkalman->wy_hat - hkalman->X[0]*I*hkalman->dwy_hat + I*hkalman->dwy_hat + hkalman->X[0]*hkalman->z_y;
128    hkalman->dyw_hat = -hkalman->X[1]*hkalman->wy_hat - hkalman->X[1]*I*hkalman->dwy_hat + hkalman->dwy_hat + hkalman->X[1]*hkalman->z_y;
129    hkalman->pre_wy_hat = hkalman->wy_hat;
130    hkalman->pre_dyw_hat = hkalman->dyw_hat;
131
132    #if LONNEM_MODE==0
133        hkalman->Pnn1[0][0] = pow(hkalman->X[0],2)*hkalman->Pnn1[0][0] + pow(hkalman->X[0],2)*hkalman->R - 2*hkalman->X[0]*hkalman->Pnn1[0][0] + hkalman->Pnn1[0][0];
134        hkalman->Pnn1[1][0] = hkalman->X[0]*hkalman->X[1]*hkalman->Pnn1[0][0] + hkalman->X[0]*hkalman->X[1]*hkalman->R - hkalman->X[0]*hkalman->Pnn1[0][1] - hkalman->X[0]*hkalman->Pnn1[0][0] + hkalman->Pnn1[0][1];
135        hkalman->Pnn1[0][1] = hkalman->X[1]*hkalman->X[0]*hkalman->Pnn1[0][0];
136        hkalman->Pnn1[1][1] = pow(hkalman->X[1],2)*hkalman->Pnn1[0][0] + pow(hkalman->X[1],2)*hkalman->R - 2*hkalman->X[1]*hkalman->Pnn1[0][1] + hkalman->Pnn1[1][1];
137
138        hkalman->pre_Pnn1[0][0] = hkalman->Pnn1[0][0];
139        hkalman->pre_Pnn1[1][0] = hkalman->Pnn1[0][0];
140        hkalman->pre_Pnn1[0][1] = hkalman->Pnn1[0][1];
141        hkalman->pre_Pnn1[1][1] = hkalman->Pnn1[1][1];
142    #endif

```

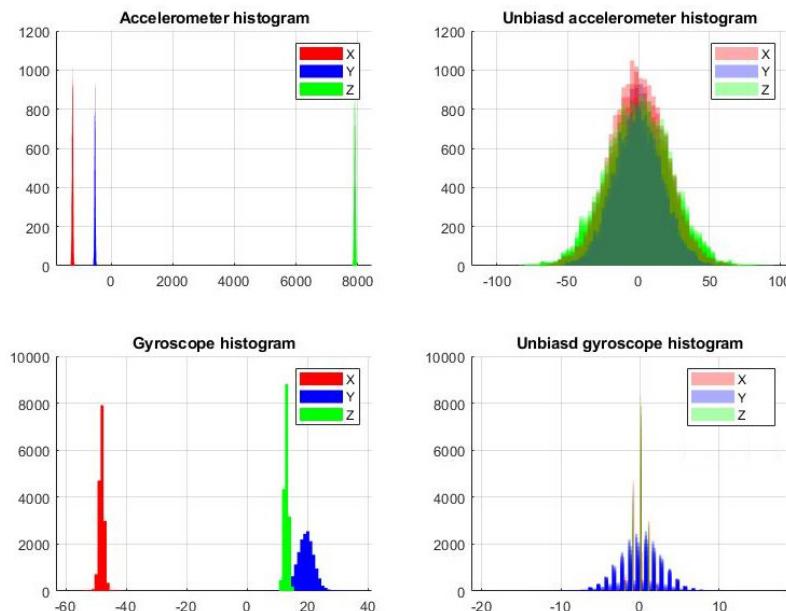
شکل ۴-۴: تصویر معادلات غیر ماتریسی کالمن به زبان C برای سیستم حرکت دورانی

جاگذاری ۱۴-۴ در معادلات کالمن و محاسبه مقادیر تخمین زده شده کار ساده ای است. اما از آنجا که باید این محاسبات در ریز پردازنده اجرا شوند، دیگر امکان محاسبات ماتریسی به صورت عمومی وجود ندارد. زیرا مقدار حافظه و قدرت پردازشی بسیار محدود است. بدین جهت به کمک نرم افزار متلب و کتابخانه محاسبات سمبولیک آن، رابطه هایی که برای پیاده سازی به صورت غیر ماتریسی مورد نیاز است استخراج شد. این معادلات بسیار بزرگ هستند. لذا به تصویر این

معادلات در برنامه بسنده می‌کنیم. شکل پیاده‌شده‌ی این معادلات در زبان C به صورت شکل ۴۱-۴ است.

حال معادلات کالمن برای سیستم‌های موردنظر به دست آمده‌اند. اکنون باید ماتریس‌های کوواریانس خطا (R و Q) را به دست آورد. ماتریس R کوواریانس نویز اندازه‌گیری یا همان v در معادلات ۹-۴ و ۱۰-۴ است. از آن جا که معادلات مشاهده همگی اسکالر هستند می‌توان R را یک عدد در نظر گرفت و به جای لفظ کوواریانس از واریانس استفاده کرد.

برای اندازه‌گیری واریانس نویز، حسگر را به صورت ساکن روی میز گذاشتم. سپس نمونه‌های آن را توسط بلوتوث به متلب منتقل کردم. بعد از چند دقیقه نمونه‌برداری، بافت‌نگاشت^{۱۸} داده‌ها رسم شد. نتیجه مطابق شکل ۴۲-۴ است. سپس با فراخوانی تابع var واریانس داده‌ها به دست می‌آید. نهایتاً میانگین واریانس نویز اندازه‌گیری برای حرکت خطی برای هر سه محور مقدار $R_{linear} = 90^{\circ}$ و برای حرکت دورانی مقدار $R_{rotational} = 1$ محاسبه شد.



شکل ۴۲-۴: هیستوگرام داده‌های حسگر حرکتی در حال سکون

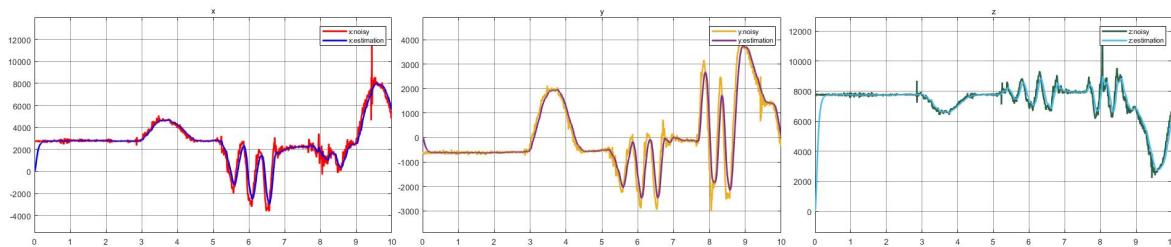
ماتریس Q اما در اکثر موارد مقداری قابل اندازه‌گیری نیست. شتاب نامعین حرکات دست به صورت اغتشاش ورودی مدل شده است که فرآیندی کاملاً نامعین است. اما با توجه به تأثیر مقدار Q بر عملکرد فیلتر، می‌توان مقدار مناسبی را پیدا کرد. می‌دانیم هر چه کوواریانس اغتشاش ورودی را بزرگتر فرض کنیم، بدین معنی است که فیلتر بیشتر بر مشاهدات متکی باشد نه به پیش‌بینی. همچنین به عکس، اگر مقدار Q کم باشد، فیلتر وزن بیشتری به پیش‌بینی خود از سیستم می‌دهد و اثر مشاهدات کمتر می‌شود. هرچه Q کمتر باشد، عملکرد فیلتر لخت‌تر و کندتر خواهد بود. هرچه Q بزرگتر شود فیلتر حالت تهاجمی‌تری^{۱۹} به خود می‌گیرد.

Histogram^{۱۸}

Aggressive^{۱۹}

حال با در نظر گرفتن این مفهوم و عملکرد، برای تعیین Q از روش سعی و خطا استفاده شد. بدین صورت که فیلتر کالمن با مقادیر مختلف Q روی داده‌های از پیش ضبط شده شبیه‌سازی شد. سپس مقداری انتخاب شد که نه خیلی تهاجمی باشد و نه خیلی کند و حافظه‌دار. در نهایت برای حرکت خطی مقدار $Q_{linear} = 0.8$ و برای حرکت دورانی مقدار $Q_{rotational} = 1.0$ انتخاب شد.

شکل ۴۳-۴ عملکرد فیلتر را روی داده‌های ضبط شده حرکت خطی نشان می‌دهد. همانطور که دیده می‌شود نویزهای موجود به خوبی حذف شده‌اند و داده‌ی خروجی فیلتر نویز مشهودی ندارد.



شکل ۴۳-۴: خروجی خام شتاب خطی در مقایسه با مقدار فیلتر شده در هر سه محور

حال که تمام مقادیر مورد نیاز فیلتر محاسبه شده و به دست آمدند، نوبت به پیاده‌سازی و بررسی عملکرد آن در ریزپردازنده می‌رسد. تمام معادلات مطابق آن چه در بالا گفته شد به صورت کد C در ریزپردازنده پیاده شد. اما نکته‌ای وجود دارد.

می‌دانیم که معادلات سیستم دورانی (معادلات ۴-۱۰) نامتغیر با زمان هستند. لذا ماتریس‌های اصلی سیستم (ϕ , G , H , R و Q) نیز با زمان تغییر نمی‌کنند. حال به معادلات کالمن (۱۱-۴) توجه کنید. معادلات مربوط به بهره‌ی کالمن (K) و کوواریانس‌های خطای $(\Sigma(N+1|N+1))$ و $(\Sigma(N+1|N+1))$ فقط تابع ماتریس‌های سیستم هستند و از حالت‌های سیستم مستقلند.

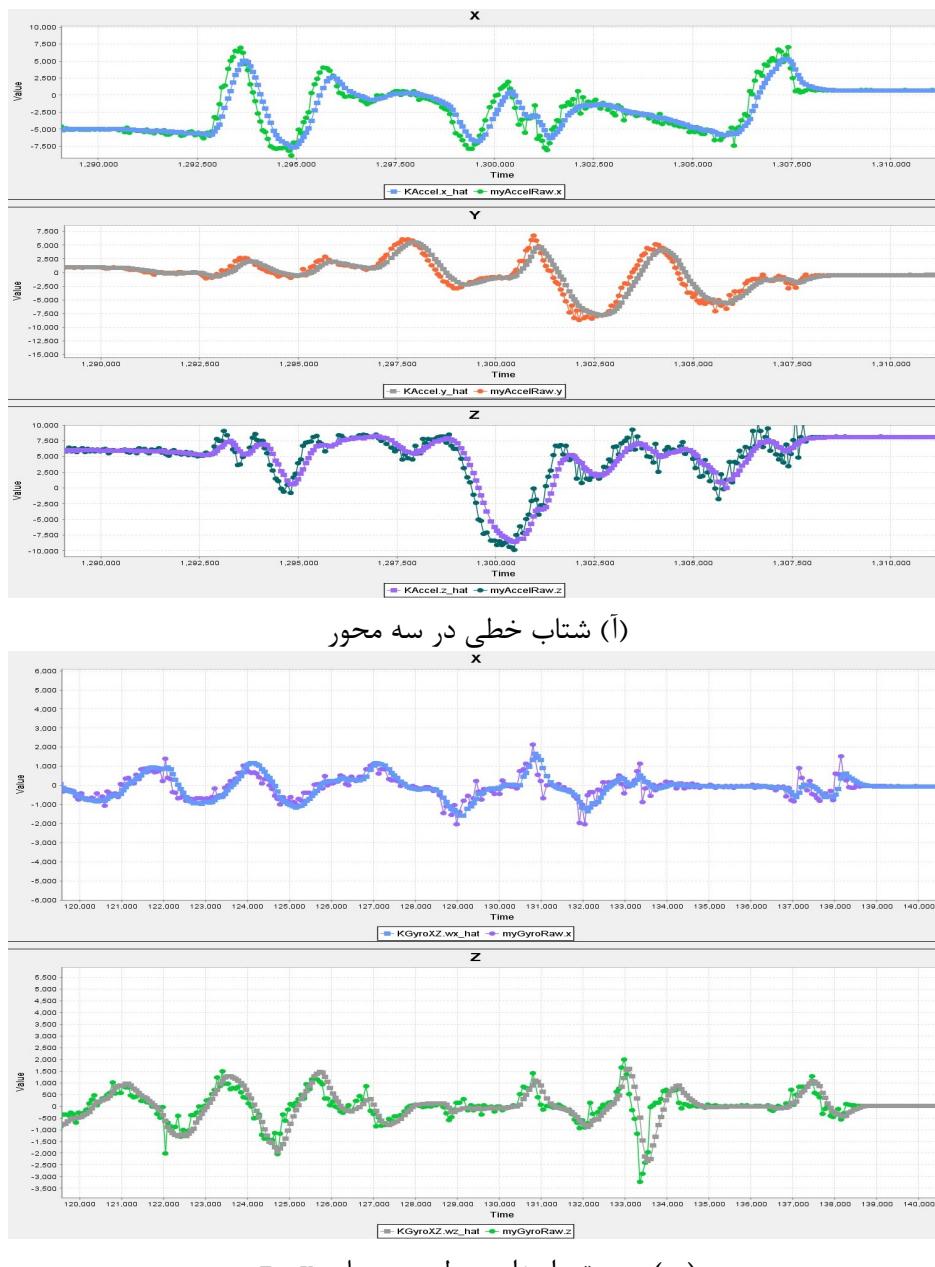
از طرفی می‌دانیم شرط کافی برای آنکه $(\Sigma(N+1|N+1)) = (\Sigma(\infty))$ موجود باشد آن است که سیستم کنترل‌پذیر و رویت‌پذیر باشد و R و Q نیز مثبت باشند [۵]. در اینجا شرط $R, Q > 0$ برقرار است. در رابطه با سیستم حرکت دورانی داریم:

$$Ctrb(\phi, G) = \begin{bmatrix} 0 & T \\ 1 & 1 \end{bmatrix}, \quad Obsv(\phi, H) = \begin{bmatrix} 1 & 0 \\ 1 & T \end{bmatrix}$$

که هر دو مرتبه کامل هستند. در نتیجه هر دو سیستم کنترل‌پذیر و رویت‌پذیرند. بنابر این $(\Sigma(\infty))$ موجود است که در نتیجه‌ی آن می‌توان گفت کوواریانس خطای پیش‌بینی و بهره‌ی کالمن نیز دارای حالت دائم و مقدار نهایی هستند.

دانستن اینکه بهره‌ی کالمن مقدار نهایی دارد و محاسبه‌ی آن مقدار، کمک می‌کند تا حالت گذرای فیلتر از بین برود و فیلتر از همان لحظه‌ی اول در بهینه‌ترین حالت کار کند. البته بزرگترین حسن آن است که دیگر نیازی به محاسبه‌ی K , $(\Sigma(N+1|N+1))$ و $(\Sigma(N+1|N+1))$ نیست. این امر حجم محاسبات را به شدت کاهش می‌دهد و سرعت پردازش را زیاد می‌کند. لذا مقدار نهایی بهره‌ی کالمن توسط نرم‌افزار متلب محاسبه شد و مستقیماً در معادله‌ی تخمین قرار داده شد. این کار حجم برنامه را

۱۴ کیلوبایت کاهش داد (مجدد متذکر می‌شوم که حجم کل حافظه ۶۴ کیلوبایت است!). در نهایت با پیاده‌سازی فیلتر کالمن با روش جایگذاری مقداری نهایی بهره، اندازه‌گیری شتاب خطی و سرعت زاویه‌ای و فیلتر کردن آن بر روی ریزپردازنده به پایان می‌رسد. شکل ۴-۴ نتیجه‌ی اجرای زمان-واقعی فیلتر برای شتاب خطی و شکل ۴-۴ ب نتیجه را برای سرعت زاویه‌ای نشان می‌دهند.



شکل ۴-۴: اجرای فیلتر کالمن روی ریزپردازنده و نتایج زمان-واقعی آن

۲-۳-۴ تشخیص گام

۳-۳-۴ تشخیص چرخش دست

۴-۴ ارتباطات

فصل پنجم

نتایج و جمع‌بندی

نتیجه

مراجع و مراجع

- [1] Mohammad Yaqoob. Mpu6050 hal based library. <https://github.com/MYaqoobEmbedded/STM32-Tutorials/tree/master/Tutorial%2035%20-%20MPU6050%20IMU%20Module>.
- [2] Olivier Van den Eede. Ssd1306 hal based library. <https://github.com/4ilo/ssd1306-stm32HAL>.
- [3] Pan. Max30102 hal based library. https://github.com/eepj/MAX30102_for_STM32_HAL.
- [4] PCBWay. Pcbway website. <https://www.pcbway.com/>.
- [5] Schwepppe, Fred C. Uncertain dynamic systems. Prentice-Hall Englewood Cliffs, N.J, 1973.
- [6] Wikipedia contributors. Inertial measurement unit — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Inertial_measurement_unit.