



دانشگاه تهران
دانشکده‌ی مهندسی برق و کامپیوتر



گزارش نهایی پژوهه سیستم‌های سایبر-فیزیکی LED چرخان با قابلیت نمایش دائم فیلم و لحظه‌ای عکس از طریق شبکه

گروه:
محسن فیاض - ژیوار صورتی - علی کرامتی پور

تاریخ:
1399/1/4

اساتید راهنمای:
دکتر مهدی کارگهی
دکتر مهدی مدرسی

1398-1399

فهرست

3	شرح کلی پروژه	❖
3	اهداف پروژه	❖
4	سخت افزار	❖
4	نرم افزار	❖
7	طراحی مفهومی	❖
7	طرح کلی (مشتمل بر اجزای درشت دانه‌ی سیستم)	❖
10	پیاده سازی (الگوریتم و نحوه‌ی کار کردن اجزای سیستم)	❖
10	شکست کار بین اعضای تیم	❖
11	سخت افزار	❖
17	نرم افزار	❖
23	تغییرات نسبت به فاز پروپوزال	❖
23	چالش‌های پروژه	❖
23	تجربیات منجر به تغییر در تصمیم‌گیری	❖
23	➤ تغییر تخته نگهدارنده آرمیچر	➤
25	➤ تغییر آرمیچر	➤
26	➤ تغییر باتری	➤
28	➤ تغییر سنسور از مغناطیسی به نور	➤
29	➤ تغییر پین مورد استفاده برای خواندن مقدار سنسور	➤
30	➤ تغییر پروتکل ارسال اطلاعات از UDP به TCP	➤
31	تجارب ناموفق	❖
31	نرديكتريين نمونه‌های مشابه	❖
32	تست عملکرد	❖
32	طرح تست	❖
32	نحوه اجرای تست (محیط تست و شرایط و)	❖
33	نتایج تست	❖
43	تحلیل نتایج	❖
50	نتیجه گیری	❖
51	هزینه نهایی	❖
51	پیوست‌های فنی	❖
52	مقالات و مراجع مورد استفاده	❖

❖ شرح کلی پروژه

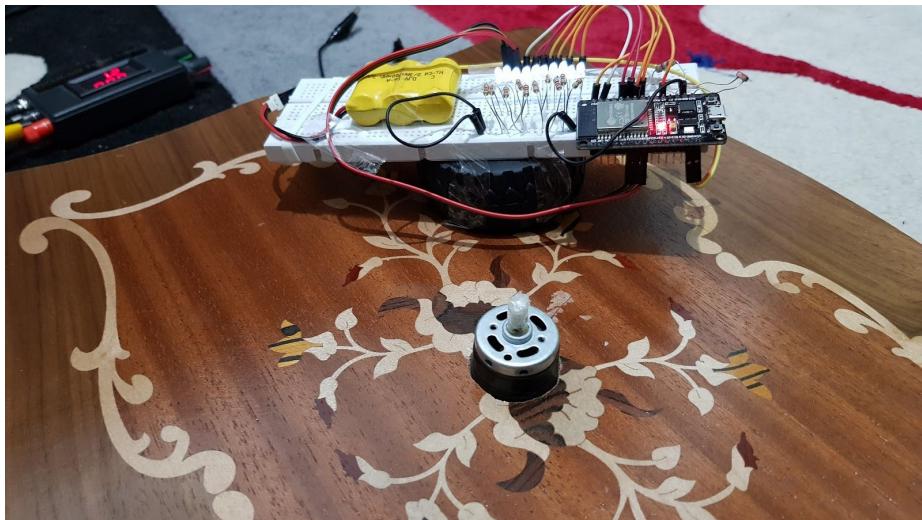
● اهداف پروژه

هدف پروژه ساخت سخت افزار یک نمایشگر گردان با تعداد اندکی LED و سپس ارسال اطلاعات به صورت آنلاین از یک گوشی اندرویدی روی شبکه و نمایش مداوم آن با توجه به محدودیت های بورد و اندروید است.

فیلم معرفی پروژه

¹[Online Real-Time Propeller display | Embedded Course Project](https://www.youtube.com/watch?v=wf1WOrsO_tA)

● ساخت افزار



ساخت افزار پروژه شامل دو بخش اصلی آرمیچر و بوردی که روی آن می چرخد است. روی بورد 10 عدد LED قرار داده شده که در بخش نرم افزار به راحتی می توان تعداد آنها را تغییر داد و 10 عدد ثابتی نیست که در این صورت می توان با افزایش تعداد LED ها در هر زمان به راحتی دقت نمایش تصاویر را افزایش داد؛ به بیان بهتر پروژه به صورتی انجام گرفته است که به راحتی قابلیت scalability داشته باشد.

قسمت اصلی و مرکزی بورد، یک system on a chip microcontroller ESP32 است که یک wifi داخلی است.

همچنین برای درک و ثبت زمان یک دور چرخیدن توسط بورد، یک عدد سنسور نور نیز قرار داده شده.

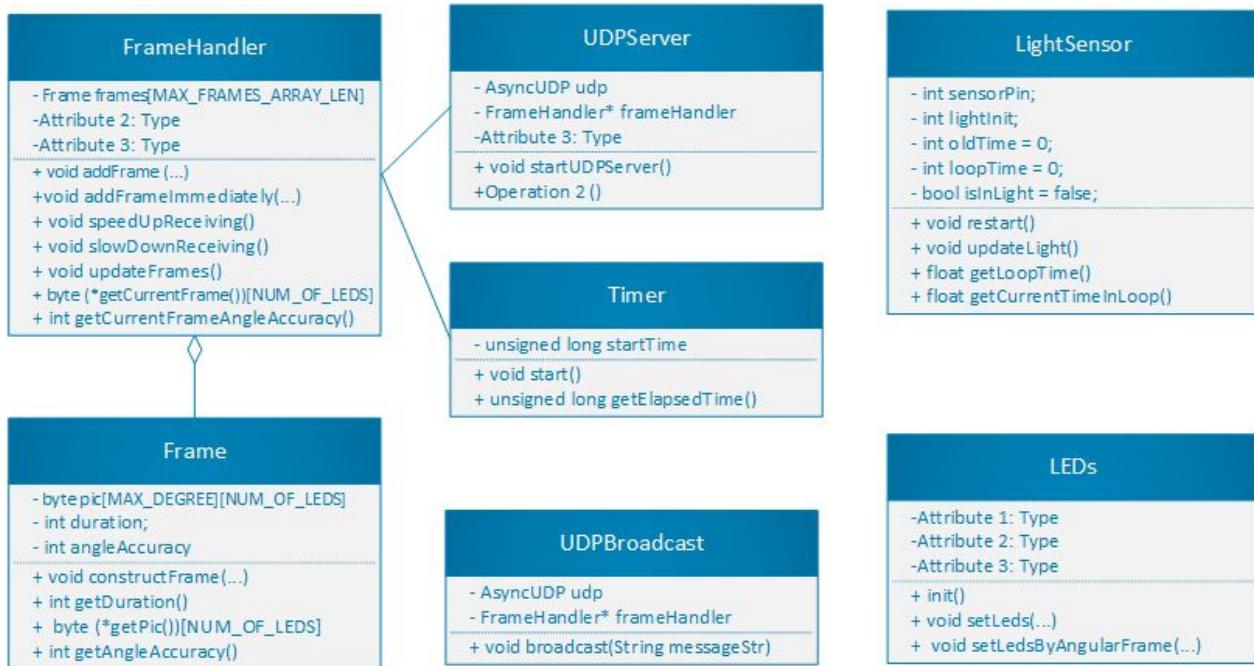
¹ https://www.youtube.com/watch?v=wf1WOrsO_tA

● نرم افزار

لینک git کد های نوشته شده:

<https://gitlab.com/mz77.co/realtime-propeller-led-display>

○ نرم افزار بورد ESP32

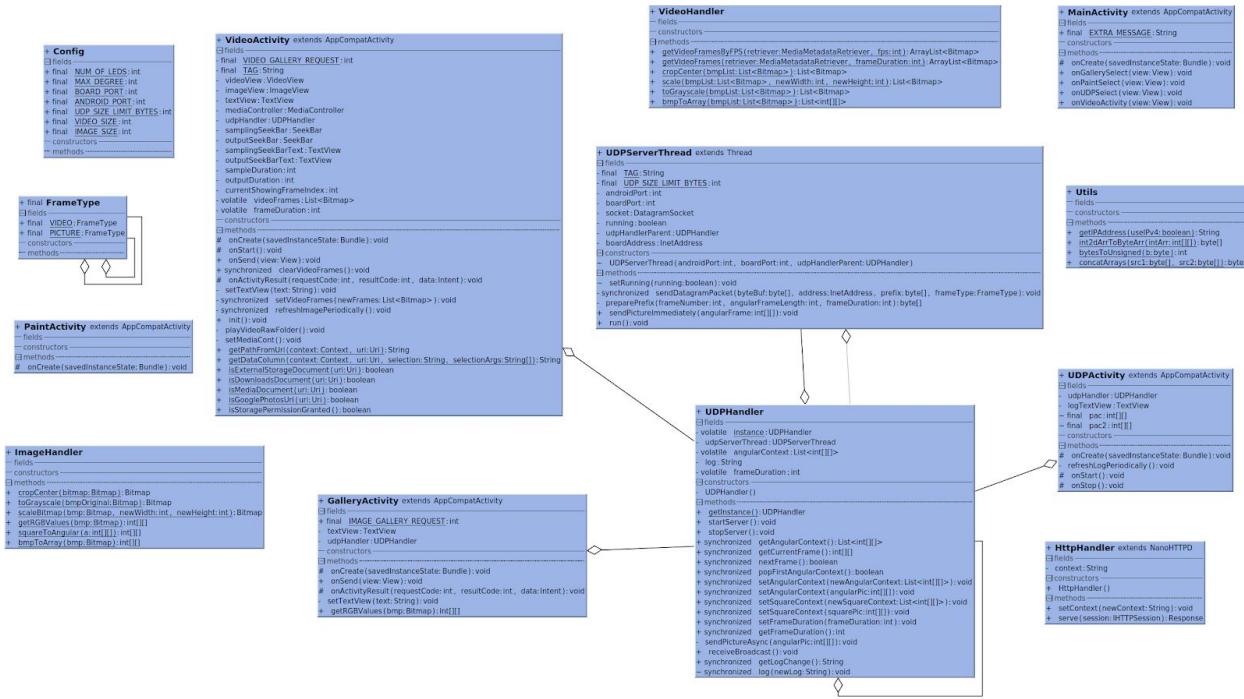


کد اجرایی روی بورد به صورت شیء گرا طراحی شده که طراحی UML آن در بالا

دیده می شود.

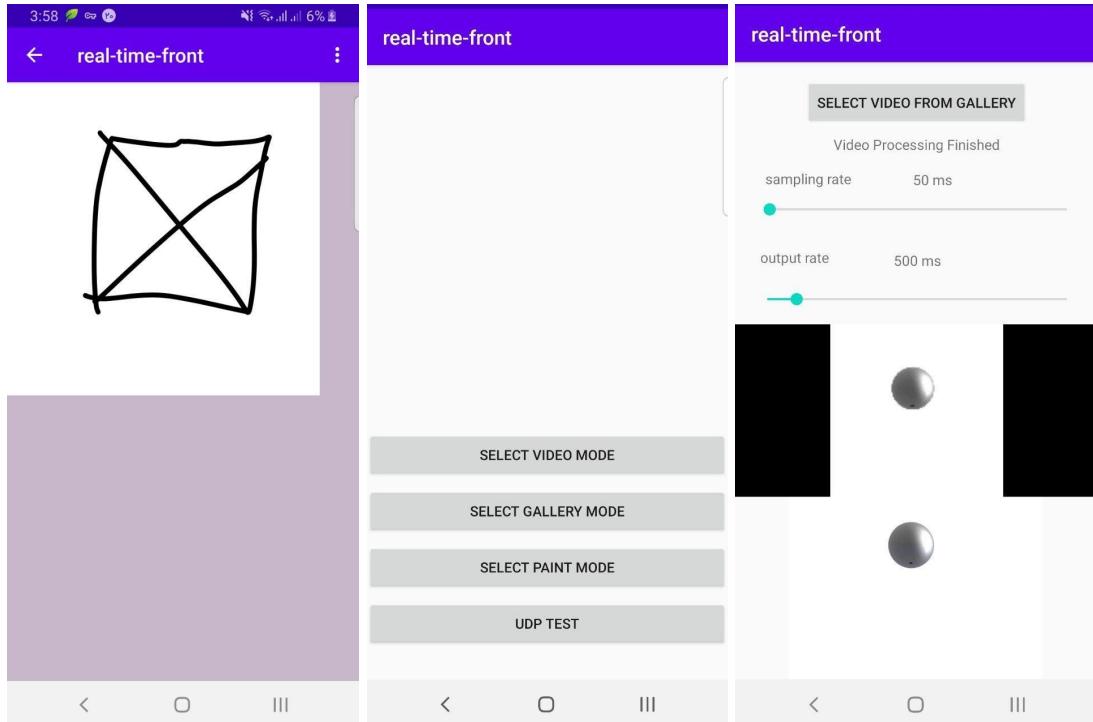
این کد برای راحتی در محیط CLion پیاده سازی شده و در محیط Arduino IDE کامپایل و کارایی آن روی بورد تست شده است.

○ نرم افزار اندروید



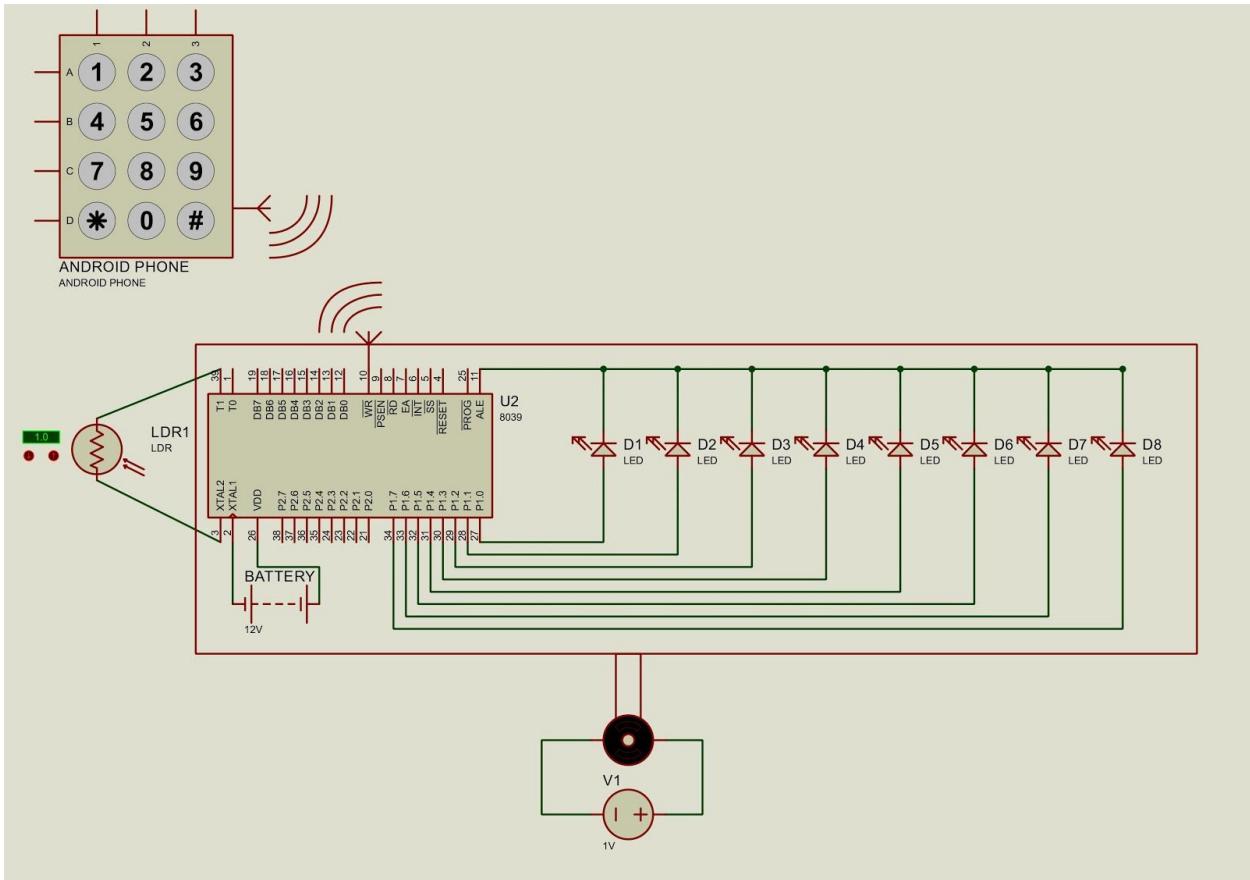
محیط پیاده سازی این بخش در Android Studio بوده و روی Emulator و حداقل 3

دستگاه متغیر تست شده است.



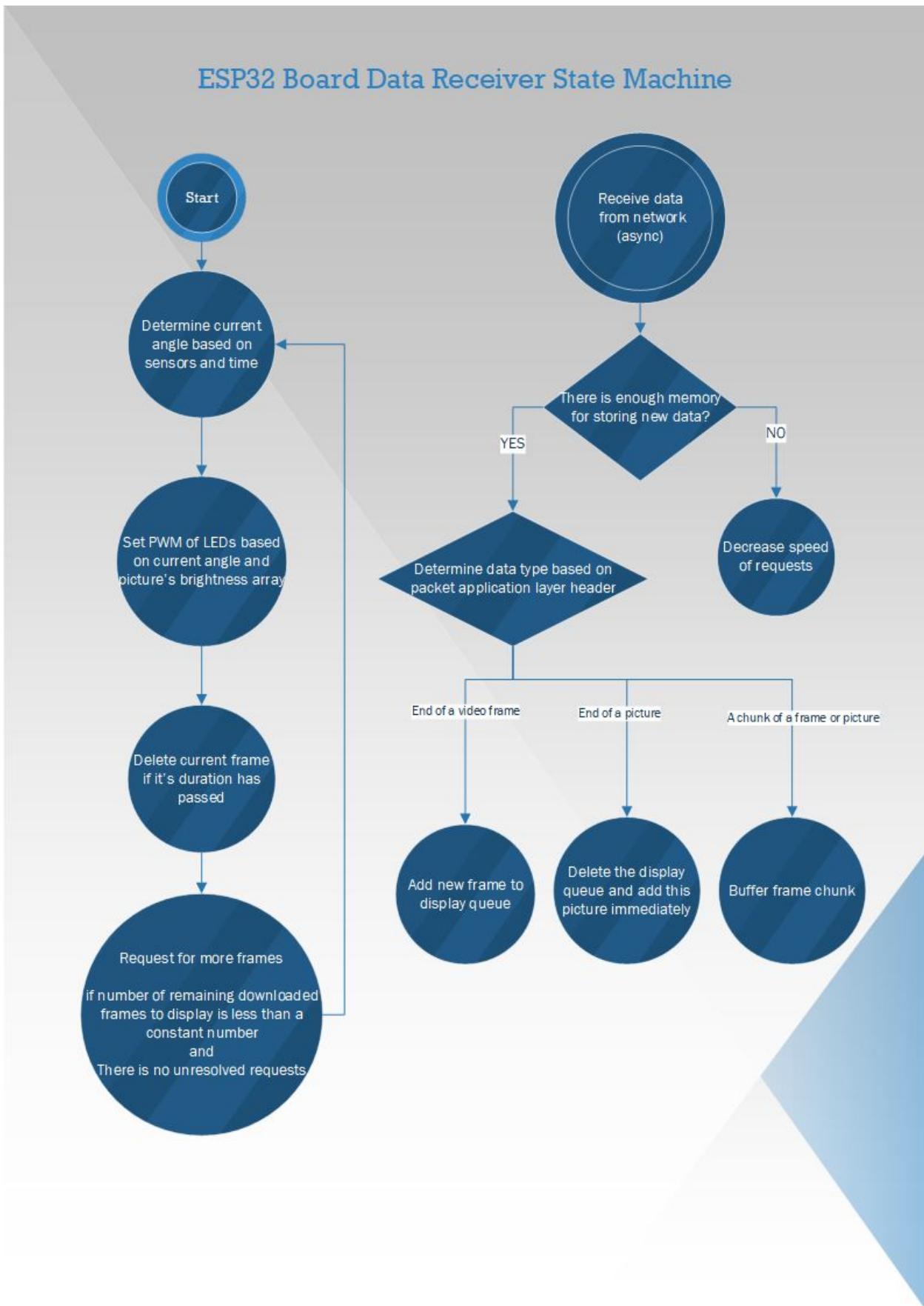
❖ طراحی مفهومی

- طرح کلی (مشتمل بر اجزای درشت دانه‌ی سیستم)



همانطور که در شکل مشخص است کامپونت ها یک عدد گوشی اندرویدی، یک عدد بورد ESP32 با مژول WIFI داخلی، یک عدد سنسور نور، 10 عدد LED با اندازه 3mm، یک عدد آرمیچر 12v یک منبع تغذیه برای چرخاندن آرمیچر با سرعت های متفاوت و یک عدد باتری 5v روی بورد برای تامین انرژی ESP32 هستند.

جريان داده در دو سمت اندرويد و ESP32 در دو شکل زیر توضیح داده شده است.





❖ پیاده سازی (الگوریتم و نحوه‌ی کار کردن اجزای سیستم)

● شکست کار بین اعضای تیم

نام	کارهای محول شده
محسن فیاض	❖ خرید، ساخت و طراحی سخت افزار شامل آرمیچر، LEDها، بورد، و ... ❖ پیاده سازی کد نرم افزاری روی ESP32 و تست اجرای آن ❖ پیاده سازی و تست کلاس مدیریت کننده UDP در سمت سخت افزار و نرم افزار و متصل کردن بورد به اندروید از این طریق ❖ بهینه سازی سرعت ارسال داده ها بر اساس تست های انجام شده روی شبکه داخلی ❖ نوشتمن گزارش ❖ ساخت فیلم معرفی پروژه
ژیوار صورتی	❖ پیاده سازی نرم افزار اندرویدی ❖ پیاده سازی صفحه دریافت ویدیو ❖ پیاده سازی کلاس ها و متدهای مورد نیاز برای تحلیل و تغییر عکسها و فیلمها روی اندروید ❖ نوشتمن بخش اندروید گزارش
علی کرامتی پور	❖ پیاده سازی صفحه paint روی اندروید ❖ پیاده سازی صفحه تایپ دایره ای روی اندروید و متدهای مورد نیاز برای دایره های کردن متن ❖ پیاده سازی نرم افزار اندرویدی ❖ نوشتمن بخش اندروید گزارش

● سخت افزار

○ آرمیچر 12V



■ دلیل انتخاب

علت انتخاب این قطعه قدرت مناسب آن نسبت به آرمیچر های ولتاژ کمتر و همچنین شکل دایره ای متقارن آن که برای سفت کردن درون چوب مناسب تر است. برای اینکار دور آن چند لایه چسب نیز کشیده شد تا درون چوب محکم تر قرار گیرد و حداقل لرزش را داشته باشد.

وزن نسبتاً زیاد بود بورده که روی این قطعه میچرخد باعث لرزش های زیادی می شود که مهار کردن آن از چالش های این پروژه بوده است. یکی از علل مهم این وزن زیاد باتری 5V است که علت استفاده از آن در ادامه گفته خواهد شد.



■ اتصالات

این قطعه از زیر با دو سیم به منبع تغذیه متصل می شود.

■ ورودی خروجی ها

ورودی توان و خروجی حرکت چرخشی است!

○ باتری 4.8V



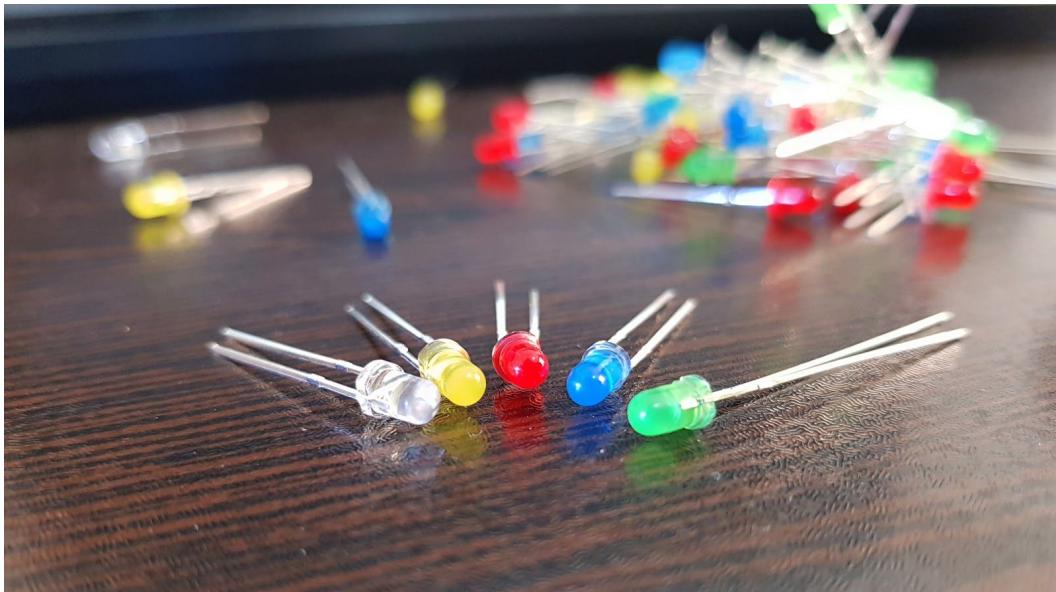
■ دلیل انتخاب

بورد ESP32 وقتی بخواهد از قابلیت های WIFI استفاده کند حتما باید با منبعی حدود 5V تغذیه شود و گزنه روشن نخواهد شد. به همین دلیل نزدیکترین باتری موجود باتری های 4.8V ماشین های کنترلی بود که از آن استفاده شد. در بازار های اینترنتی باتری سبک تر با این ولتاژ پیدا نشد که در بخش های بعدی مفصل توضیح داده می شود.

■ اتصالات

باتری به دو پین VIN و GND بورد ESP32 متصل می شود.

LED 3mm °



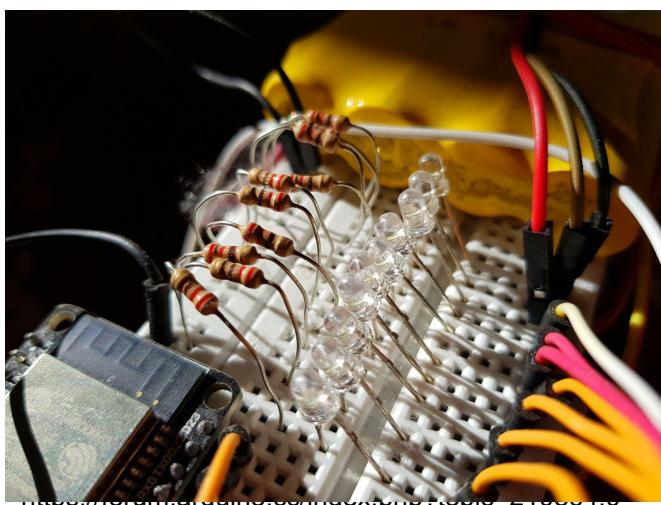
■ دلیل انتخاب

دلیل اول سایز کوچکتر آنها نسبت به مشابه های 5mm است که باعث می شود در چرخش پره، بتوانیم با دقت بیشتری زوایا را در نمایش تفکیک کنیم.

علت دوم آن بود که باید ردیف LED کاملاً یکسان باشد تا در دید ناظر تصویر قابل تشخیص باشد به همین دلیل بسته 100 عددی از LED های یکسان خریده شد. (متاسفانه در زمان خرید بسته تک رنگ موجود نبود و مجبور شدیم چند رنگ بخریم)

طبق تحقیقاتی که انجام شد سرعت روشن و خاموش شدن LED ها در حد چند 10 نانو ثانیه است که نسبت به دقت زمانی ms² که ما برای نمایش فرم ها نیاز داریم بسیار بهتر است. (منبع)

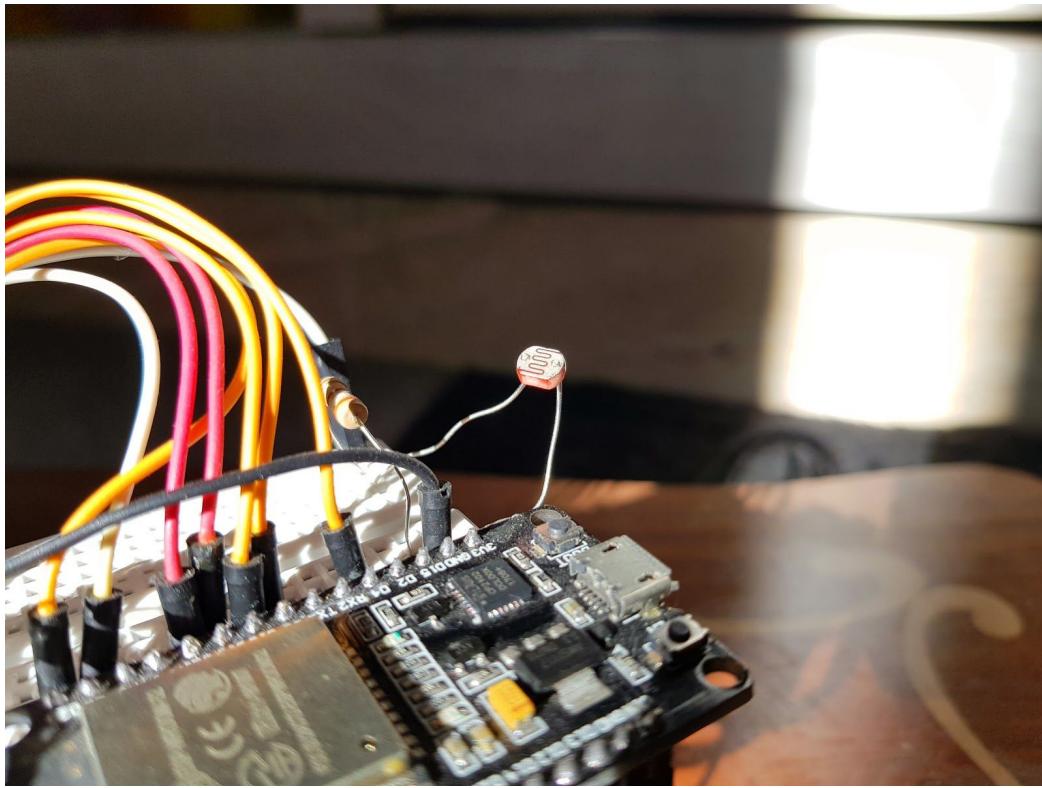
■ اتصالات



این LED ها از یک طرف به پین های مشخص شده از ESP32 متصل می شوند و از طرف دیگر به یک مقاومت تا جریان زیادی کشیده نشود.

سنسور نور

◦



■ دلیل انتخاب

مهم ترین علت انتخاب این سنسور این است که می‌توان با انداختن نور از فاصله با پره چرخان نتیجه مطلوب را گرفت. در این مورد در بخش‌های توضیحات بیشتری خواهد داده شد.

روش تشخیص یک دور به این صورت است که یک threshold برای شدت نور طبق نور اولیه محیط (وقتی بورد روشن می‌شود شدت نور اولیه ذخیره می‌شود. همچنین با استفاده از قابلیت touch بورد ESP32 اگر پین شماره 12 لمس شود این مقدار دهی دوباره انجام می‌شود) در نظر گرفته می‌شود و هر وقت سنسور از آن بالاتر رود و بعد پایین بیاید نقطه مورد نظر مشخص می‌شود.

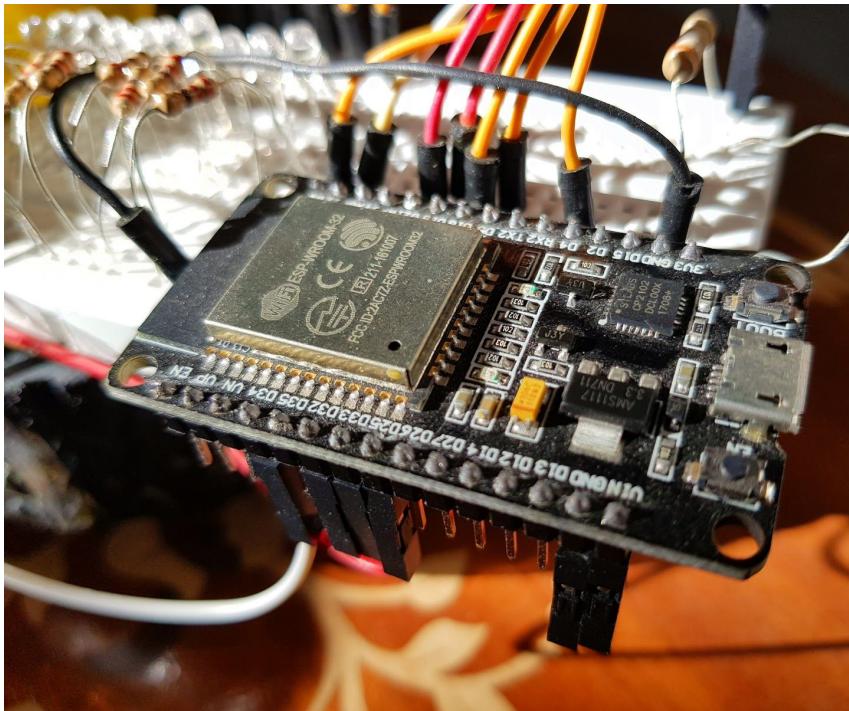
زمان پاسخگویی LDR بین 2 تا 50 میلی ثانیه است ([منبع³](#)) که یعنی برای اینکه در کمتر از یک دور کامل نتیجه دیده شود، سرعت چرخش پره می‌تواند بین 20 تا 500 دور در ثانیه طبق زمان پاسخگویی باشد.

■ اتصالات

این سنسور به یک مقاومت و پین خواندن Analog بورد ESP32 متصل است.

³ <http://lednique.com/opto-isolators-2/light-dependent-resistor-ldr/>

ESP32-WROOM-32 ○



■ دلیل انتخاب

WIFI ●

مهم ترین علت انتخاب این برد وجود مازول WIFI داخل خود ESP32 است که کار با شبکه را بسیار آسان تر و سریعتر می‌کند، چون اگر مازول جدایی برای اینکار استفاده می‌شد، باید سریار ارتباط میکروکنترلر و آن مازول خارجی را هم متحمل می‌شدیم، در حالی که الان این ارتباط به صورت داخلی پیاده سازی شده که سرعت بیشتری دارد.

FreeRTOS & Cores ●

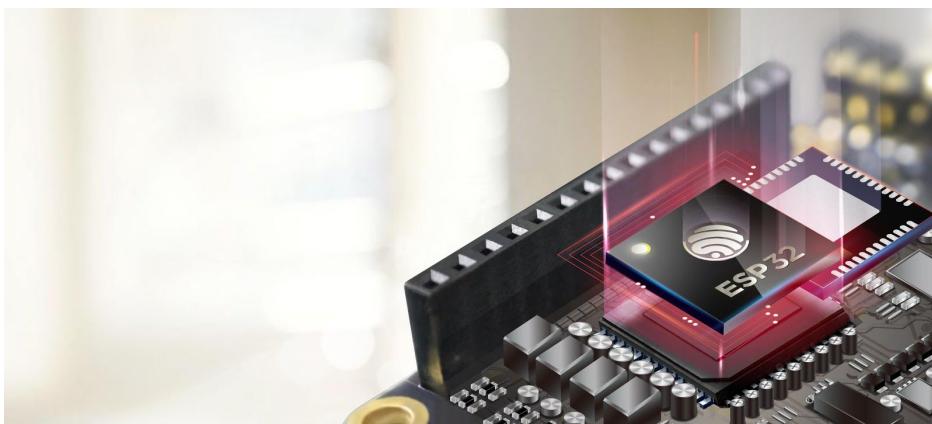
از طرفی این برد دارای دو هسته است و می‌توان از FreeRTOS روی آن استفاده کرد که برای انجام این پروژه ارزشمند بوده است. یکی از بخش‌های پروژه در سمت میکروکنترلر، هم روند پیش بدن نمایش فریم‌ها روی LED‌ها و دریافت و تحلیل فریم‌های ورودی از شبکه بوده است. در این کار قطعاً از thread‌ها استفاده شده است اما لزوماً از هر دو هسته استفاده نشده است. در این مورد باید توجه داشت که طبق تست‌هایی که انجام شد سرعت اجرای دستورات نوشته شده روی هسته شماره 1 سریعتر از هسته شماره 0 این میکروکنترلر است.

⁴ فرکانس کاری هسته این میکروکنترلر حداقل 240MHz است. (منبع)

⁴ https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

SRAM ●

این بورد دارای 520KB SRAM برای ذخیره دیتا و instruction دارد. این موضوع باعث شد تا حداکثر 31 عدد فریم آن هم به صورت فشرده شده بتوان نگه داشت که توضیحات آن در چالش ها خواهد آمد. همچنین برای ذخیره لیست فریم ها از یک آرایه با سایز ثابت استفاده می شود چون در سیستم های real-time و embedded به علت محدودیت حافظه و امکانات، اصلا نباید از حافظه متغیر استفاده کرد. این تصمیم بر ثبات اجرای برنامه بسیار تاثیر دارد.



■ اتصالات

این میکروکنترلر به باتری، سنسور نور، تمام LED ها و با استفاده از شبکه به گوشی اندرویدی متصل است.

■ ورودی خروجی ها

ورودی های مورد نیاز برای ثابت نگاه داشتن تصویر و نظرخیدن آن یک ورودی سنسور است که پس از هر دور 360 درجه ای پره، باید به کنترلر اطلاع داده شود. برای این منظور می توان از سنسور های مختلفی مانند سنسور اثر هال یا سنسور نور استفاده کرد که در طی ساخت پروژه تصمیم گیری خواهد شد.

ورودی دیگر پروژه، اطلاعاتی است که از شبکه دریافت خواهد شد و باید روی پره نمایش داده شود.

● نرم افزار

○ نرم افزار ESP32

■ سیستم عامل و محیط توسعه

سیستم عامل روی میکروکنترلر FreeRTOS است که از xtask در این سیستم عامل استفاده شده است.

محیط توسعه در دو محیط [Arduino IDE](#) و [CLion](#) انجام شد. مورد اول از نظر تمیزی کد و پیشنهادات مناسب در هنگام کد زدن این پروژه را سریعتر می‌کرد و مورد دوم وظیفه کامپایل و فرستادن instruction ها به میکروکنترلر از طریق پورت سریال را داشت.

■ ورودی ها و خروجی ها

ورودی های نرم افزار مقدار خوانده شده از پین آتالوگ متصل به سنسور نور، و اطلاعات فرستاده شده از طریق شبکه است. خروجی ها، کنترل روشنابی LED ها و درخواست فریم های بیشتر از طریق شبکه است.

○ نرم افزار اندروید

■ سیستم عامل و محیط توسعه

اندروید بر پایه نسخه اصلاح شده‌ای از هسته لینوکس و دیگر نرم‌افزارهای متن باز طراحی شده است. نرم‌افزارهای جانی اندرویدی با استفاده از زبان جاوا نوشته می‌شوند و برای ارتباط با لایه‌های زیرین سیستم عامل می‌توانند از کتابخانه‌های جاوای اندروید استفاده کنند. بخش رابط کاربری سیستم عامل اندروید با زبان جاوا نوشته شده است و بسیاری از برنامه‌های اندروید هم با جاوا نوشته شده‌اند. اما این سیستم عامل، Java Virtual Machine ندارد.

برای اجرای برنامه‌های جاوا روی این سیستم عامل، کدهای جاوا به کدهای Dalvik تبدیل می‌شوند و سپس روی Dalvik virtual machine اجرا می‌شوند. دالویک یک ماشین مجازی جاوای است که برای سیستم عامل اندروید بهینه شده است تا هم RAM و هم CPU را کمتری مصرف کند. برنامه‌های جاوای معمولی هم که روی گوشی‌های دیگر اجرا می‌شوند با استفاده از نرم‌افزار شبیه‌ساز جاوا مانند J2ME LOADER روی این سیستم عامل قابل اجرا هستند.

همینطور محیط برنامه‌نویسی که برای این پروژه و کدهای اندرویدی استفاده کردیم android studio می‌باشد که بر پایه محیط IntelliJ طراحی شده و با توجه به داشتن محیطی برای مدیریت همه activity ها و همینطور resource ها و layout های استفاده شده در قسمت تست نیز می‌تواند emulator ها را مدیریت کند تا با هر تغییری در کد بتوان دوباره برنامه را تست و بررسی کرد.

■ ورودی ها و خروجی ها

ورودی ها در برنامه اندرویدی در قسمت های مختلف با یکدیگر مختلف هستند با توجه به اینکه در قسمت های مختلف می توان به صفحات مختلفی مراجعه کرد و به شکل های مختلفی به برنامه ورودی داد به عنوان مثال می توان از طریق انتخاب ویدیو یا عکس از گالری کاربر به این شکل ورودی به بورد و قسمت سخت افزاری را مهیا کرد و یا اینکه می توان به صورت در لحظه و آنی از طریق کشید یک نقاشی در صفحه برنامه ورودی آن را به بورد انتقال داد.

در قسمتی که می توان عکس یا ویدیو در برنامه انتخاب کرد روند کار به این شکل انجام می شود که در ابتدا پس از انتخاب عکس یا فیلم مورد نظر URI منبع مورد نظر گرفته می شود پس از آن می توان با استفاده از آن، فایل مربوطه را پیدا کرد. پس از آن نیز می توان با استخراج frame ها و همینطور bitmap (فرمتی که برای ذخیره عکس و یا pixel ها استفاده می شود)، استفاده و دست کاری آن ها تصویری مورد نظر و مناسب برای ارسال به بورد آماده کرد. به عنوان مثال می توان تصویری به شکل سیاه و سفید و یا حتی به شکل مربعی که برای استفاده بورد مناسب تر است بدست آورد. پس از آن نیز می توان با تبدیل این bitmap ها به آرایه ای از رنگ های مختلف مربوط به هر پیکسل و همینطور پس از آن نگاشت این اطلاعات که مربوط به تصاویر مربعی هستند به زاویه ای، اطلاعات مناسب برای نمایش در بورد را به آن ارسال کرد.

■ نحوه ارتباط با اجزای سخت افزار

از طریق کلاس Udp Handler

◦ توضیح پروتکل ساخته شده برای انتقال فریم ها

اول لازم به توضیح است که هر فریم ارسالی به شکل 360 در 10 است. یعنی مقدار روشنایی هر کدام از 10 تا LED موجود (بین 0 تا 255) به ازای هر یک از 360 درجه. بنابراین هر عدد 1 بایت فضا می خواهد و در مجموع هر فریم $10 \times 360 = 3600$ که می شود 3600 بایت است.

به علت محدودیت اندازه بسته های udp، مقدار مطمئن برای ارسال تا مشکلی پیش نیاید حدود 100 بایت در هر بسته است. به همین دلیل هر بسته به چندین بسته کوچکتر 1000 بایتی شکسته می شود. (در اینجا می شود 3 بسته 1000 بایتی و یک بسته 600 بایتی) برای اینکه در مقصد، این بسته ها به هم متصل شوند و یک فریم را بسازند، نیاز به تعریف header هایی برای هر بسته است.

اطلاعاتی که اول هر بسته فرستاده می‌شود به شرح زیر است:

PREFIX[0]	PREFIX[1]	PREFIX[2]	PREFIX[3]	PAYLOAD
COMMAND {F, E, I}	PACKET INDEX	FRAME DURATION (LSB)	FRAME DURATION (MSB)	(1000B)

COMMAND . 1 (یک بایت)

این مقدار نمایش دهنده این است که محتویات بسته چیست و باید چگونه با آن برخورد کرد. مقادیری که پیاده سازی شده اند به شرح زیر است:

"F" . a

مشخص کننده ارسال بسته ای میانی از یک Frame است. بورد با دریافت آن، می‌فهمد که باید بسته را ذخیره کند، و منتظر بسته های بعدی باشد تا فریم کامل ساخته شود.

"E" . b

این دستور نشان دهنده پایان یا همان END of Frame است. بورد با دریافت آن می‌فهمد که بسته های قبلی با رسیدن این بسته به اتمام می‌رسد. بنابراین اطلاعات بسته های قبلی را به این بسته متصل می‌کند و فریم کامل شده را به کلاس FrameHandler می‌دهد تا در موقع مناسب نمایش داده شود.

"I" . c

این دستور نشان دهنده پایان یک عکس است که باید سریعاً نمایش داده شود، یعنی End of Immediate Frame.

بورد با دریافت آن، همان کارهای مشابه دستور E را انجام می‌دهد با این تفاوت که به کلاس FrameHandler اطلاع می‌دهد که این فریم باید به صورت آنی و در لحظه نمایش داده شود.

یعنی این دستور تمام فریم های بافر شده قبلی را پاک می‌کند و فریم جدید را جایگزین می‌کند.

این دستور در ارسال عکس و مهمتر از آن در حالت paint استفاده می‌شود. نکته مهم آن است که ممکن است همین یک فریم به اشتباه ارسال شود، یا اطلاعاتش مشکلی داشته باشد، به همین دلیل پس از این که دریافت شد نباید همیشه همین فریم را نمایش داد.

برای همین، این عکس هم مانند فریم فیلم دیده می شود، و پس از مدتی، دوباره درخواست فریم جدید داده می شود، که اندروید در این حالت همان عکس را دوباره می فرستد. با این کار اگر تصویر بار اول اشتباه ارسال شود، در دفعات بعدی جبران می شود.

2. PACKET INDEX (یک بایت)

این بخش، شماره فریم ارسالی را مشخص می کند. این عدد بین ۰ تا تعداد فریم درخواستی منهای یک است. قصد از گذاشتن این مورد، توانایی دریافت فریم ها با ترتیب درست بود، اما با توجه به اینکه بدون استفاده از آن نتایج مطلوبی دیده شد نیازی به سربار مرتب سازی نبود.

3. FRAME DURATION (دو بایت)

این بخش مشخص کننده مدت نمایش فریم ارسالی است. این عدد می تواند بین ۰ تا ۶۵,۵۳۶ باشد، البته در این مورد هم روی بورد محدودیت گذاشته شده که حداقل هر فریم ۵ ثانیه مهلت نمایش دارد و در صورتی که مدتی بیش از ۵۰۰۰ms داده شود، همان ۵۰۰۰ در نظر گرفته می شود.

همچنین باید توجه داشت که عکس ها نیز این مدت را دارند، و بورد به صورت دوره ای همان عکس را درخواست و می کند و از گوشی می گیرد تا اگر در یک بار گرفتن مشکلی پیش آمده بود بطرف شود.

● بخش متن real-time

در این قسمت کاربر با تایپ کردن متن ورودی با حداکثر ۱۲ کاراکتر از حروف الفبا در یک فیلد `editText`، متن را به صورت real-time بر روی بورد مشاهده می‌کند.

برای پیاده‌سازی این قابلیت، ابتدا سعی بر دایره‌ای کردن یک تصویر عادی از یک نوشته داشتیم.



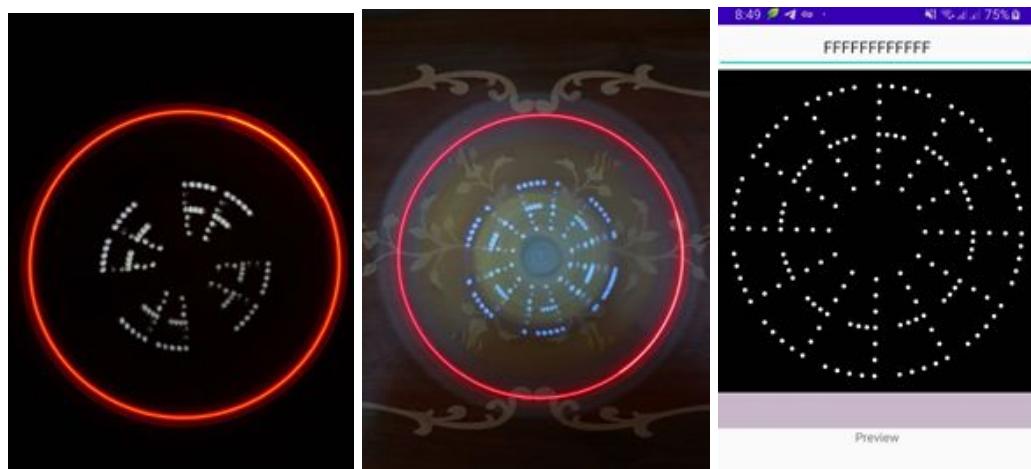
همینطور که در این تصویر می‌بینیم نوشته را به یک نوشته دایره‌ای تبدیل کردیم. اما به دلیل کمبود LED ها، تصویر درست مشاهده نمی‌شد.

در اقدام بعدی تمامی اطلاعات برای روشن و خاموش بودن LED ها را برای هر حرف الفبا انگلیسی را `hardcode` کردیم. از آنجا که آرایه نهایی ارسال شده در تابع `setAngularContext` ابعاد ۳۶۰ در ۱۰ دارد، ارتفاع هر حرفمان حداکثر ۱۰ می‌باشد. با توجه به تجربه گذشته، با کوچک کردن شعاع دایره خالی درون نوشته، نوشته شکل خوبی به خود نمی‌گیرد و قسمت زیرین تمامی حروف در مرکز دایره مشاهده می‌شوند. با دانستن این سعی شد که حروف را در ارتفاعی کمتر از ۱۰ و در ارتفاع ۷ پیاده سازی شود. همچنین عرض هر حرف را ۵ در نظر گرفتیم. بنابراین کاربر با وارد کردن متن یک آرایه ۳۶۰ در ۱۰ ساخته شده و برای برد ارسال می‌شود.

در طول این پروژه، به دلیل قرنطینه با مشکلاتی روبرو شدیم که در انتهای مجبور به شبیه سازی بورد در محیط نرم افزاری شدیم که شکل نهایی را در محیط نرم افزاری نیز مشاهده بکنیم.

طبق شبیه‌سازی، اگر برای هر حرف عرض ۵ را در نظر می‌گفته‌یم تصویر خیلی فشرده و غیر خوانایی از حروف مشاهده می‌کردیم. با تغییر این زاویه به ۲۵ تصویر خوانا تر شد. همچنین ۵ زاویه خالی برای فاصله گذاری بین حروف در نظر گرفته شد.

طبق شبیه‌سازی انجام شده تمام خروجی ها ایده‌آل بود اما در هنگام انتقال به سخت‌افزار خروجی متفاوتی گرفته شد



همانطور که در بالا مشاهده می‌شود خروجی شبیه سازی شده با خروجی سخت‌افزار هم خوانی ندارد و در بعضی زاویه ها هیچی مشاهده نمی‌شود ! با توجه به اینکه آرایه نهایی کاملا درست بود متوجه این مشکل شدیم که سخت‌افزار به دلیل محدودیت ها در دقت گاهی ممکن است بعضی زوایا را تشخیص ندهد و از روی آن بگذرد. همانطور که گفتیم در هر حرف در ۲۵ زاویه نمایش داده می‌شود اما به این صورت که زاویه ها پیوسته نبودند و تنها زوایا ۰ و ۱۵ و ۳۰ و ۴۵ و ۶۰ و ۷۵ و ۹۰ به مقدار دهی می‌شدند. با مقدار دهی زوایا ۱ و ۲ و ۳ و ۴ به مقدار نشان داده شده در زاویه ۰ ، ۶۰ و ۷۵ و ۹۰ به مقدار آرایه در زاویه ۵ و ... به نتیجه مطلوب رسیدیم و تصاویر آن را در زیر مشاهده می‌کنید.



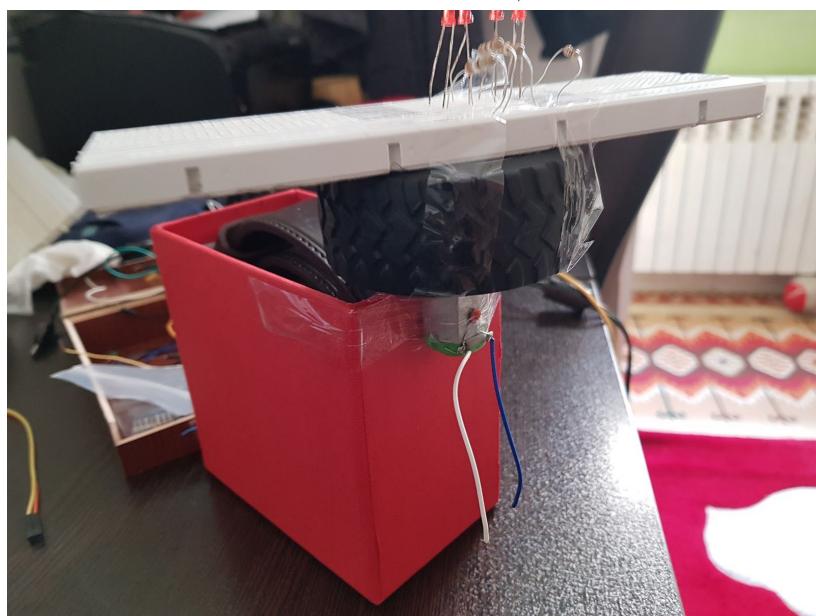
❖ تغییرات نسبت به فاز پروپوزال

- چالش های پژوهش

● تجربیات منجر به تغییر در تصمیم گیری

○ تغییر تخته نگهدارنده آرمیچر

در ابتدای ساخت قطعه های لازم برای چرخش پره، از یک قطعه مقواوی استفاده شده بود.



این قطعه برای چرخش های با سرعت بالا مناسب نبود زیرا لرزش بسیار زیادی به آن منتقل می شد که تحمل آن را نداشت. البته برای نمایش تصاویر ثابت مناسب بود که تصویر زیر نمونه از این مورد است.



این مشکل باعث شد تا یک قطعه چوب سنگین را جایگزین پایه قبلی کنیم.

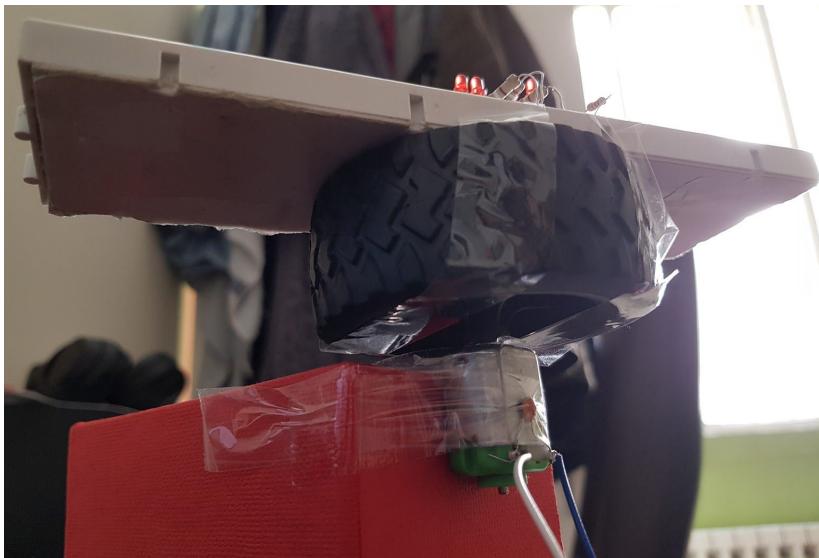


برای محکم کردن آرمیچر در این پایه جدید لازم بود تا بخشی دایره ای درون چوب سوراخ شود که به علت محدودیت ابزار های موجود، این کار با دریل انجام شد.



۰ تغییر آرمیچر

در ابتدای ساخت پره و پایه آن، از یک آرمیچر ۵v که از یک ماشین اسباب بازی برداشته شده بود استفاده می شد.



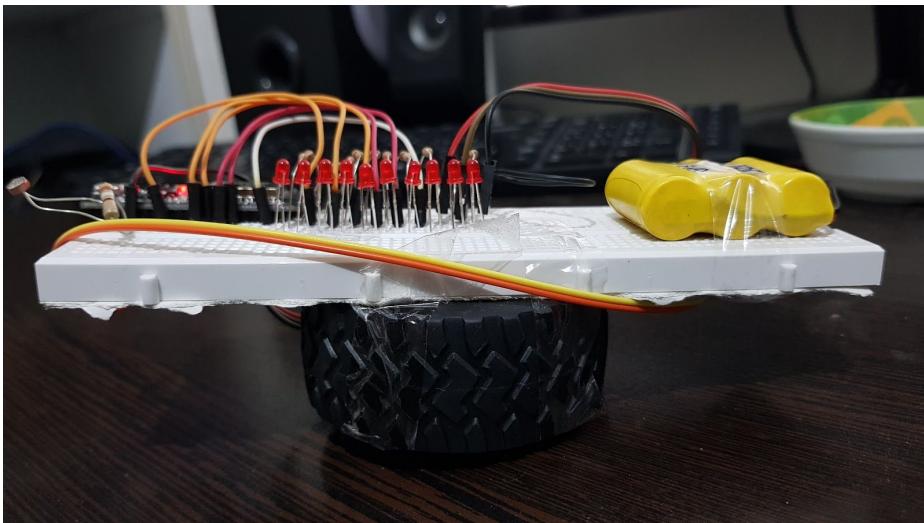
ضعیف بودن این آرمیچر باعث می شد هم نتوان سرعت بالایی از چرخش پره انتظار داشت، همچنین در چرخیدن های مداوم (حدود چند دقیقه) این آرمیچر داغ می شد و این مورد باعث کمتر شدن توانایی آن نیز می شد.

به همین دلیل همزمان با استفاده از تخته چوبی، یک آرمیچر ۱۲v نیز خریده شد که دیگر هیچ کدام از مشکلات گفته شده را ندارد و الان محدود کننده سرعت نه آرمیچر، بلکه لرزش شدید پره در سرعت های خیلی زیاد است.



۰ تغییر باتری

در ابتدای کار از یک باتری 3.6v استفاده می شد که در شکل زیر نیز مشخص است.



این باتری وزن کمی داشت و چرخاندن آن روی پره راحت بود. از این جهت بهترین گزینه برای استفاده بود.



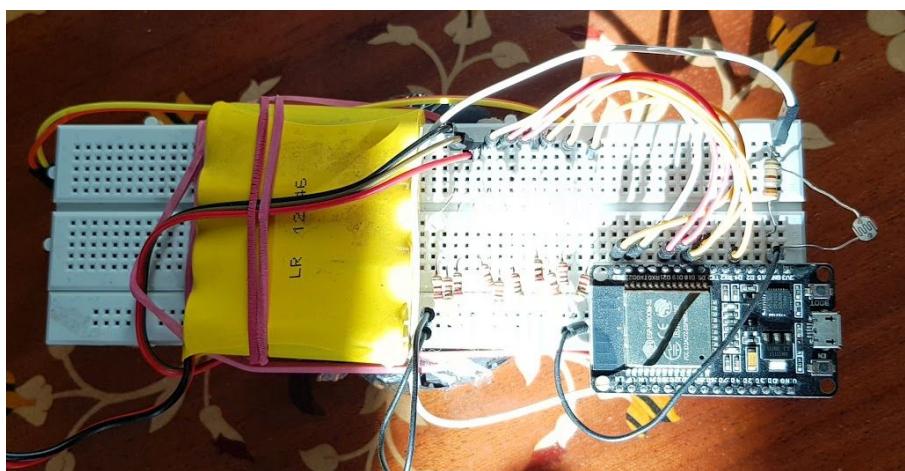
اما مشکل از زمانی به وجود آمد که بخش WIFI را روی ESP32 فعال کردیم. بعد از آن، دیگر باتری نمی توانست، میکروکنترلر را روشن کند.^۵

^۵ Brownout detector was triggered

به همین دلیل باتری را به یک عدد باتری 4.8v تغییر دادیم که مشکل برطرف شد.

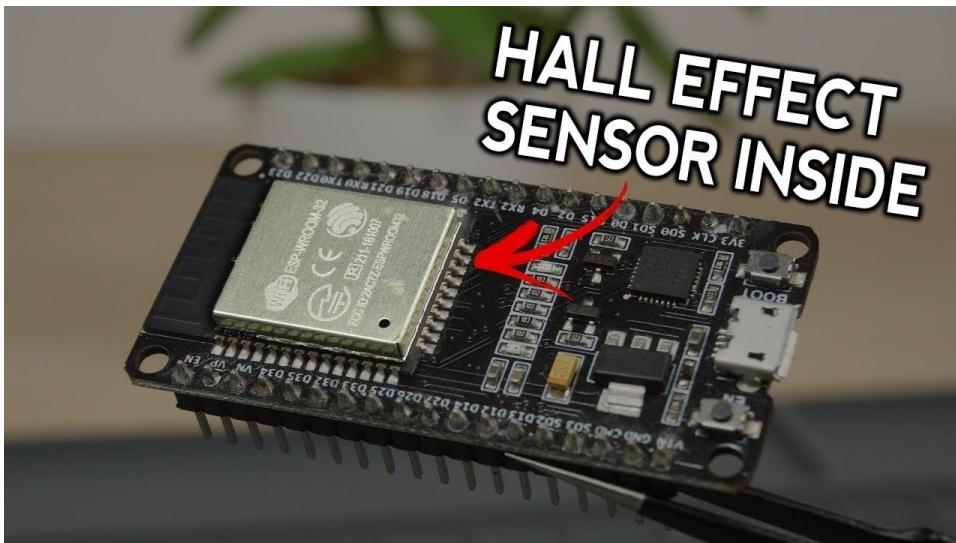


که البته سنگین تر بودن این باتری باعث مشکلاتی در تعادل مناسب پره و چرخش آن می شود که با تلاش برای قرار دادن آن در بخش وسطی پره سعی شده تا مرکز ثقل پره در مرکز آن باشد.



۰ تغییر سنسور از مغناطیسی به نور

در ابتدای پروژه، طبق نمونه های مشابهی که دیده بودیم، ما هم می خواستیم از یک سنسور اثر هال و یک آهنربا برای تشخیص زمان یک دور چرخش پره استفاده کنیم. روی خود ESP32 یک سنسور اثر هال موجود است.



با بررسی آن و خواندن مقدارش، متوجه شدیم که آهنربا باید بسیار به این سنسور نزدیک باشد تا تغییر معناداری در مقدار خوانده شده دیده شود.

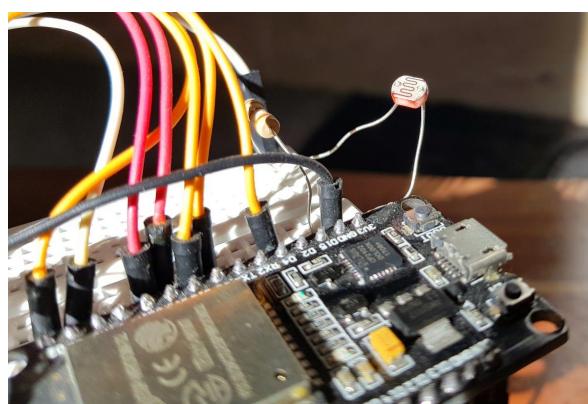
به ۳ علت این گرینه جایش را به یک سنسور ساده LDR داد.

اولین علت همین نزدیکی بسیار دقیق آهنربا به سنسور برای تاثیرگذاری بود.

دومین علت که از معلول های بخش اول است، پیش آمدن مشکل چسبیدن آهنربا به باقی میکروکنترلر، پین های آهنی آن و بقیه اجزای پره، و یا حتی کششی که ایجاد می کند بود.

و سومین علت ناموجود بودن سنسور اثر هال در فروشگاه ها، و وفور سنسور شدت نور.

بنابراین تصمیم بر این شد تا با استفاده از یک سنسور نور ساده، و یک چراغ قوه، این مشکل برطرف شود.



۰ تغییر پین مورد استفاده برای خواندن مقدار سنسور

در ابتدای کار از پین شماره 15 برای خواندن مقدار سنسور نور استفاده می شد. اما پس از فعال کردن بخش RF میکروکنترلر این مقدار روی این سنسور دیگر به درستی خوانده نمی شد. طبق بررسی ها، این میکروکنترلر دو عدد ADC دارد.^۶ ADC1 و ADC2 همچنین GPIO15 از ADC2 استفاده می کند. از طرفی علت این اتفاق این بوده که WIFI نیز از ADC2 استفاده می کند و به همین دلیل همزمان نمی توان از analogRead استفاده کرد.^۷ به همین دلیل به جای استفاده از این پین، از GPIO34 استفاده کردیم که به ADC1 متصل است.

^۶ <https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>

^۷ <https://github.com/espressif/arduino-esp32/issues/102>

۰ تغییر پروتکل ارسال اطلاعات از TCP به UDP

ابدای پروژه قصد داشتیم تا با استفاده از TCP از ویژگی های مفید آن مثل اطمینان از رسیدن همه اطلاعات، و به ترتیب رسیدنشان استفاده کنیم.

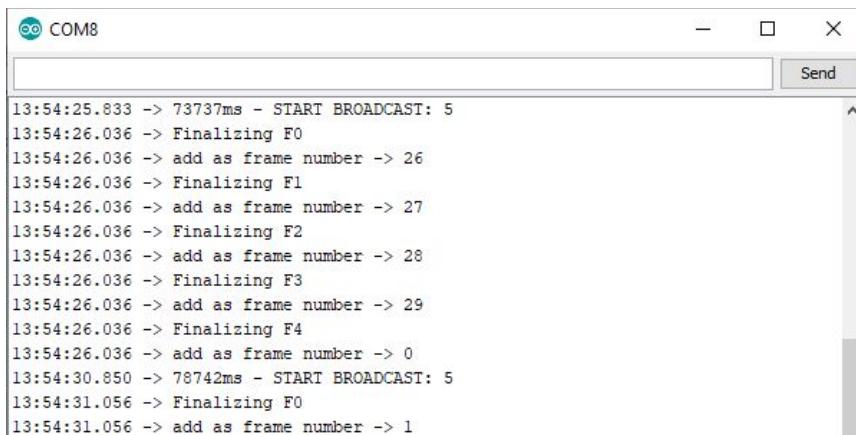
اما پس از تحقیقات و بیشتر آشنا شدن با Congestion Control و جزئیات این پروتکل، تصمیم گرفتیم مانند اکثر سرویس های stream، ما هم از UDP استفاده کنیم.

```
long m = millis();
HTTPClient http;
http.begin("http://192.168.1.7:8080/"); //HTTP
// start connection and send HTTP header
int httpCode = http.GET();
Serial.print(String(millis()-m));
Serial.println("ms");
// httpCode will be negative on error
if(httpCode > 0) {
    // file found at server
    if(httpCode == HTTP_CODE_OK) {
        String payload = http.getString();
        Serial.print(sizeof(payload));
        Serial.println("B");
    }
}
```

03:18:50.885 -> 12B
03:18:53.973 -> Message
03:18:54.587 -> 607ms
03:18:54.587 -> 12B
03:18:57.616 -> Message
03:19:00.637 -> 3009ms
03:19:00.637 -> 12B
03:19:03.621 -> Message
03:19:04.337 -> 692ms
03:19:04.337 -> 12B
03:19:07.343 -> Message
03:19:07.683 -> 361ms
03:19:07.683 -> 12B
03:19:10.699 -> Message

Autoscroll Show timestamp

همانطور که دیده می شود، با استفاده از HTTP برای درخواست و دریافت یک بسته یکسان، بین 3009ms تا 361ms تاخیر دیده می شود. این موضوع برای کاربرد ما که real-time است اصلا مناسب نیست.



اما در UDP دیده می شود از زمان ارسال درخواست بسته، تا دریافت بسته، حدود 203 تا 206ms زمان می برد. (تست های بیشتر هم انجام شده که میانگین همین مقدار است و به علت محدودیت گزارش همه عکس ها آورده نشده)

طبق این نتایج UDP نه تنها همواره حدود زمانی ثابتی برای رسیدن بسته ها دارد (با توجه به اینکه روی یک شبکه داخلی با ترافیک کم و bandwidth زیاد اجرا می شود)، همچنین در استفاده ای ما سرعت بهتری نیز نسبت به HTTP دارد.

۰ تغییر ساختار فریم های ارسالی از مربعی به دایره‌ای

در ابتدا قصد ما این بود که فریم ها را به همان صورت مربعی اصلی به بورد بفرستیم و بورد خودش از روی آن، چیزهایی را که باید در هر زاویه نشان دهد استنتاج کند.

اما پس از مواجه شدن با سرعت کم UDP و تاثیر حجم اطلاعات روی delay و همچنین مهمتر از آن، حافظه RAM بسیار کم میکروکنترلر که کلا حدود ۵ فریم مربعی را می‌توانست نگه دارد، تصمیم گرفتیم این ساختار را به دایره ای تغییر دهیم. در ساختار دایره‌ای هر فریم به شکل یک آرایه دو بعدی به اندازی 10×360 است که ۳۶۰ نشان دهنده ۳۶۰ درجه، و ۱۰ نیز نشان دهنده هر LED است. مقدار هر خانه این فریم ها می‌تواند عددی بین ۰ تا ۲۵۵ باشد که بعبارتی یک بایت باشد. با این تصمیم و تست روی میکروکنترلر دیده شد که حداکثر می‌توان ۳۱ فریم به این شکل از حافظه گرفت که برای بافر کردن فیلم‌های مقدار کافی می‌باشد.

● تجرب ناموفق

یک باتری لیتیوم یون سبک خردباری شد، اما به علت اینکه ولتاژ آن کم بود و همچنین سوکت آن متناسب با تغییه اجزای پروژه نبود بی استفاده شد و در آخر از همان باتری سنگین اسیاب بازی استفاده کردیم. از طرفی باتری های بعد از ۳V حدود ۷V است و قیمت آنها بسیار گران است که باعث محدودیت ما در استفاده از باتری شد.

❖ نزدیکترین نمونه های مشابه

در همه مواردی که ما دیدیم، نمونه های مشابه صرفا تا مرحله‌ی ساخت یک پره چرخان و نمایش شکلی ساده مثل ساعت پیش می‌رفتند. از این موارد برای ساخت بهتر پره، و یا محل قرار گیری سنسور استفاده شد. اما از نظر بخش پیچیده تر پروژه ما که ارتباط با شبکه بود نمونه‌ای ندیدیم و خودمان با طراحی اختصاصی این بخش را پیاده سازی کردیم.

فیلم و سندهای نمونه‌های مشابه:

- <https://www.youtube.com/watch?v=JrcKJ0djQN8>
- <https://www.youtube.com/watch?v=FoLSfqmOjYo>
- <https://www.youtube.com/watch?v=JV0KobgGwBk>
- <https://thestempedia.com/project/diy-led-pov-display/>
- <https://www.electronicshub.org/pov-display-using-arduino/>

❖ تست عملکرد

● طرح تست

ویژگی اصلی که باید توسط پروژه ما تحويل داده می شد و در پروپوزال گفته شده بود، صرفا نمایش یک فیلم از شبکه به صورت مدام بود. البته که سخت ترین بخش پروژه هم همان بخش بود، اما تیم ما به همان ویژگی بستنده نکرد و بخش های زیر را نیز به ویژگی های قابل تحويل پروژه اضافه کرد:

حالت **paint** که با کشیدن شکل روی صفحه نمایش به صورت **real-time** روی نمایشگر چرخان نیز نمایش داده می شود.

مورد دیگر نمایش هر عکسی از گالری گوشی اندرویدی است.

و مورد دیگر، نمایش هر فیلم دلخواه از روی گالری گوشی است. (به جای نمایش فیلم خاصی که از قبل آماده شده) بنابراین تست ما شامل بخش های زیر بود:

1. تست نمایش فیلم دلخواه با قابلیت تنظیم فریم ریت

2. تست نمایش یک عکس از گالری

3. تست نمایش به هنگام نقاشی کشیده شده در حالت **paint**

● نحوه اجرای تست (محیط تست و شرایط و)

برای دریافت اطلاعات زمانی، از لگ های روی اندروید و همچنین لاغ هایی که روی میکروکنترلر چاپ کردیم، استفاده کردیم.

برای فیلم برداری و عکس برداری از نتایج کار، باید این کار در شرایط نوری بسیار تاریک انجام می شد.

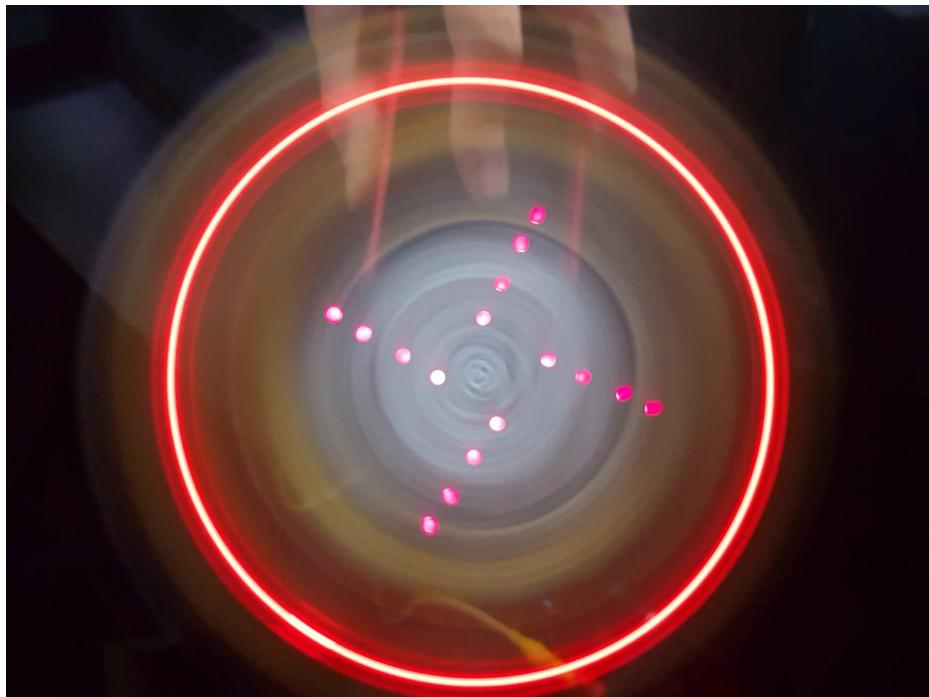
عکس برداری نسبت به فیلم گرفتن راحت تر بود، چون با تنظیم دیافراگم و ISO روی دوربین می توانستیم به خروجی دلخواه برسیم.

اما برای فیلمبرداری این تنظیمات به صورت خودکار و طبق محیط فیلمبرداری انجام می شوند که باعث شد مجبور باشیم محیط را بسیار تاریک کنیم تا شکل نمایش داده شده در دوربین دیده شود.

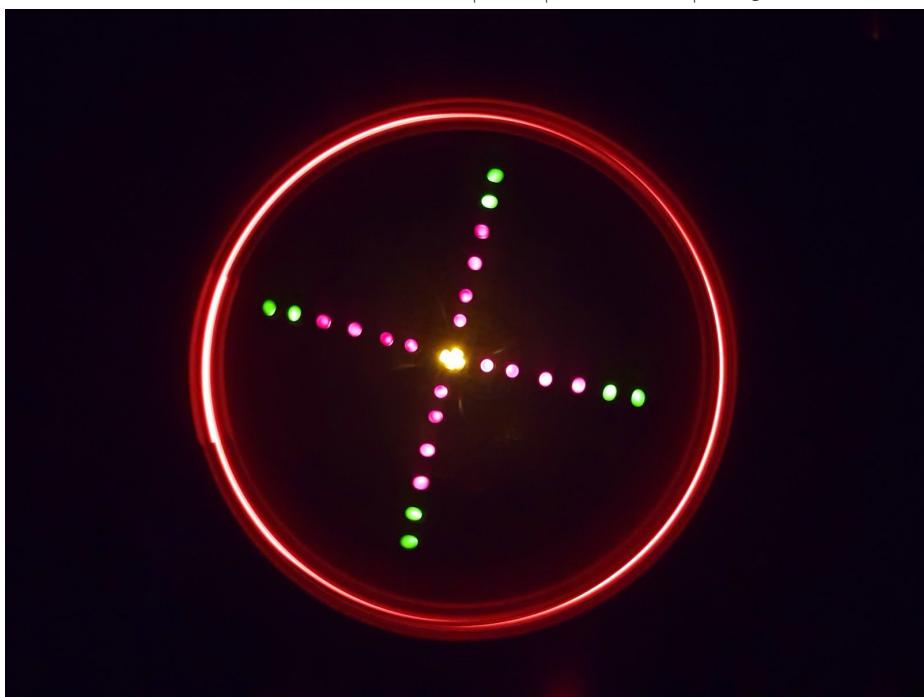
● نتایج تست

روند پیشرفت نتایج:

در اولین تست ها، صرفا با یک کد ساده در 4 جهت LED ها را روشن کردیم تا نتیجه را ببینیم.



که همانطور که دیده می شود هم تعداد LED ها کم بود و هم زاویه مناسب را نداشتند.



پس از آن روی همان سازه قدیمی که استحکام کافی را نداشت نتیجه زیر را مشاهده کردیم

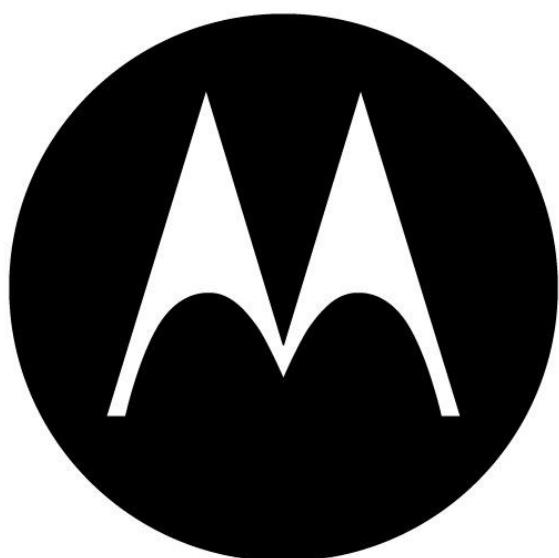
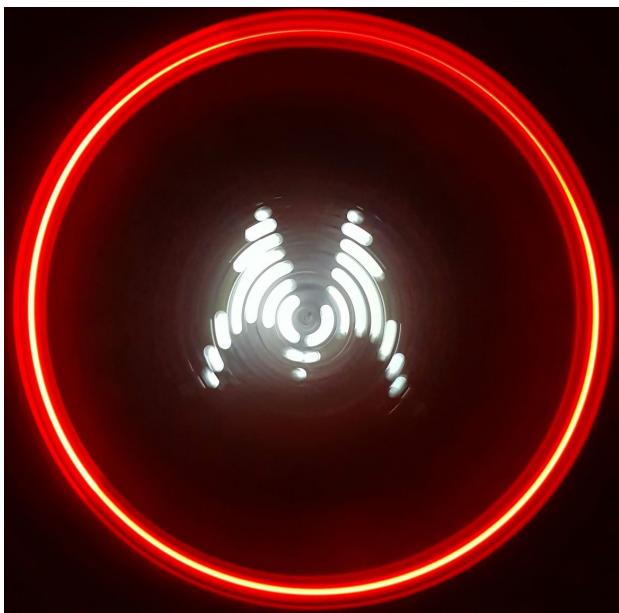


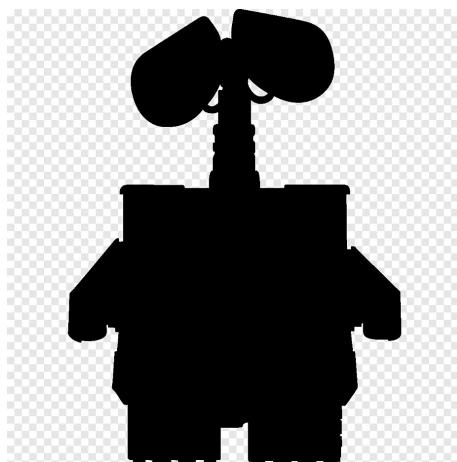
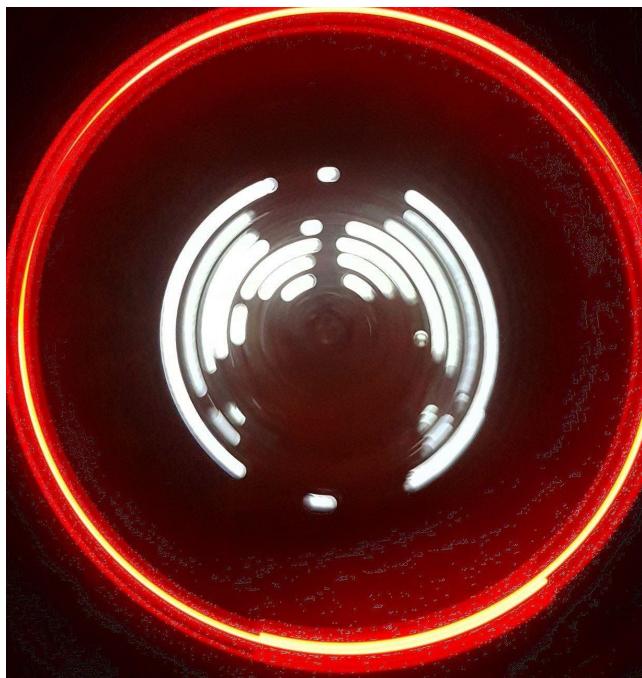
پس از آن سازه جدید را با چوب محکم تر، آرمیچر جدید و LED های جدید و بیشتر ساختیم که خروجی زیر را گرفتیم.



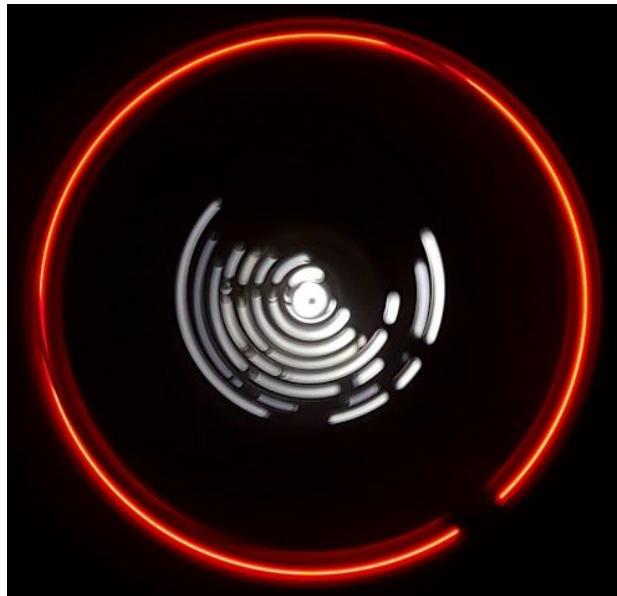
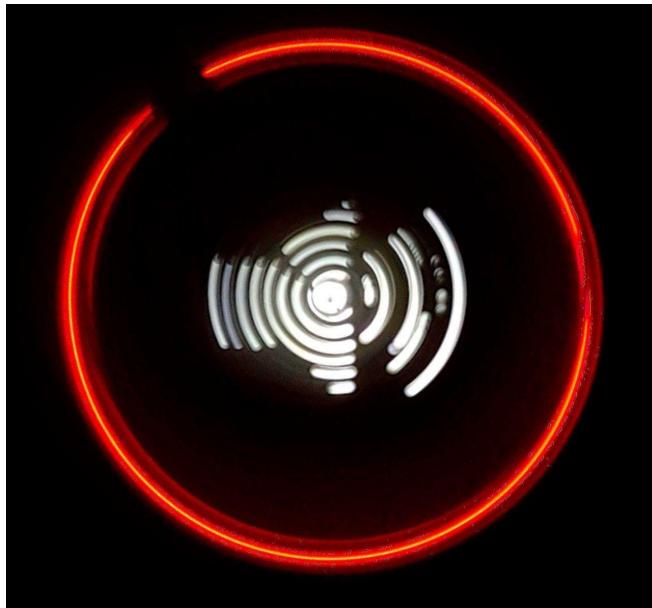
سپس پخش ارسال از شبکه را به پروژه اضافه کردیم که باعث اتصال گوشی اندرویدی به بورد شد.

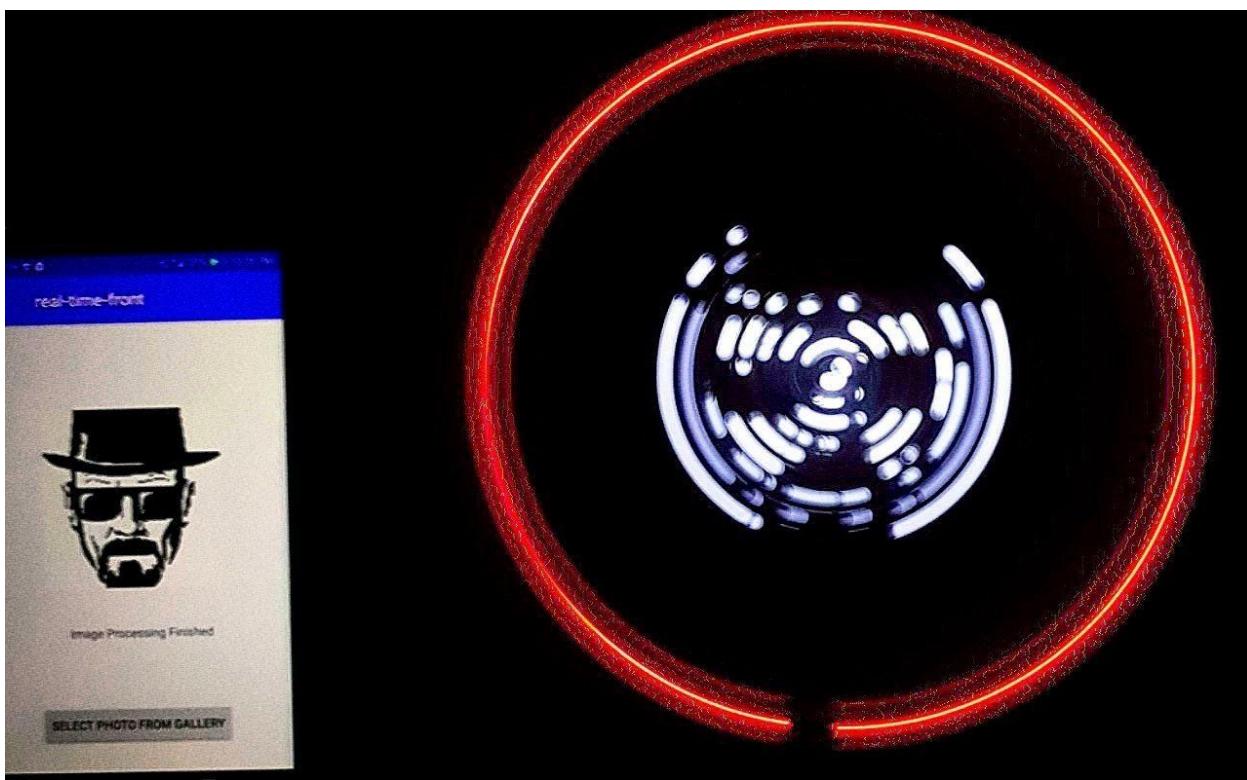
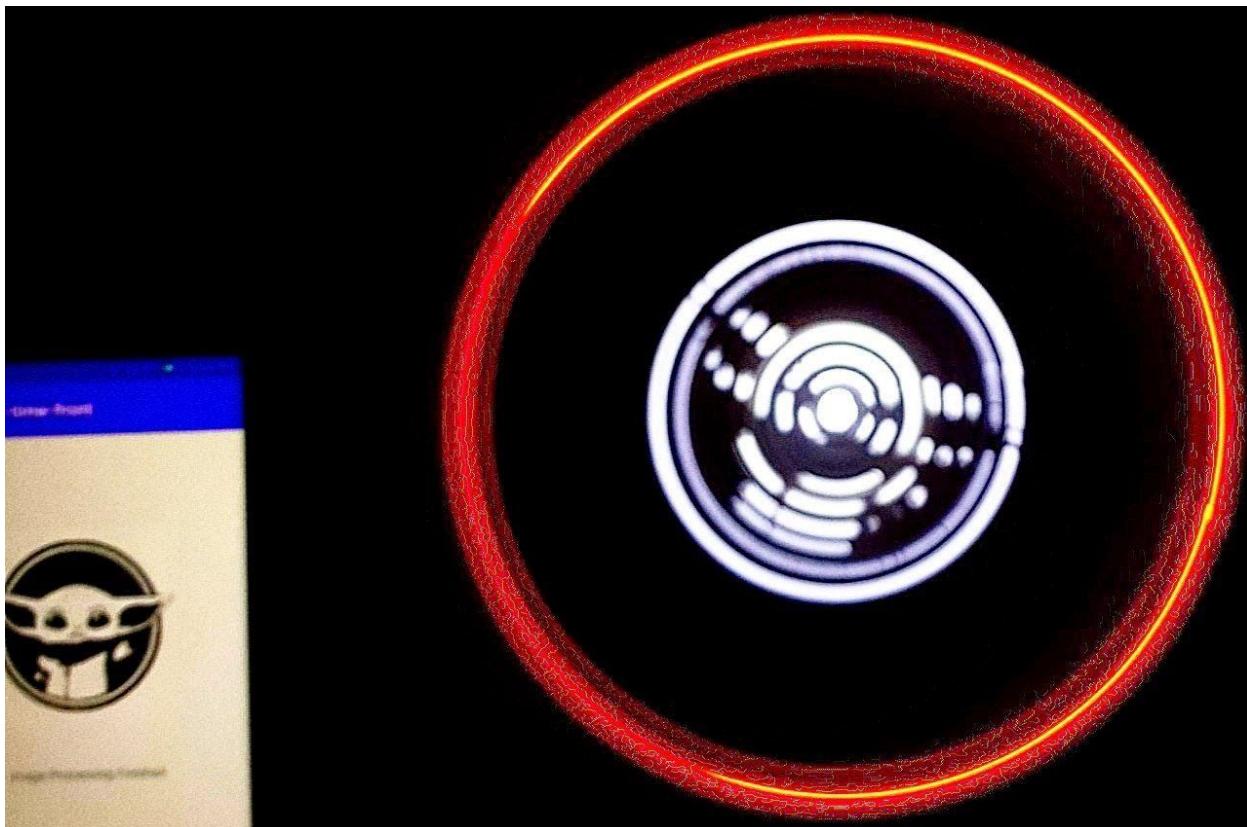
تست های نهایی بخش تصویر:

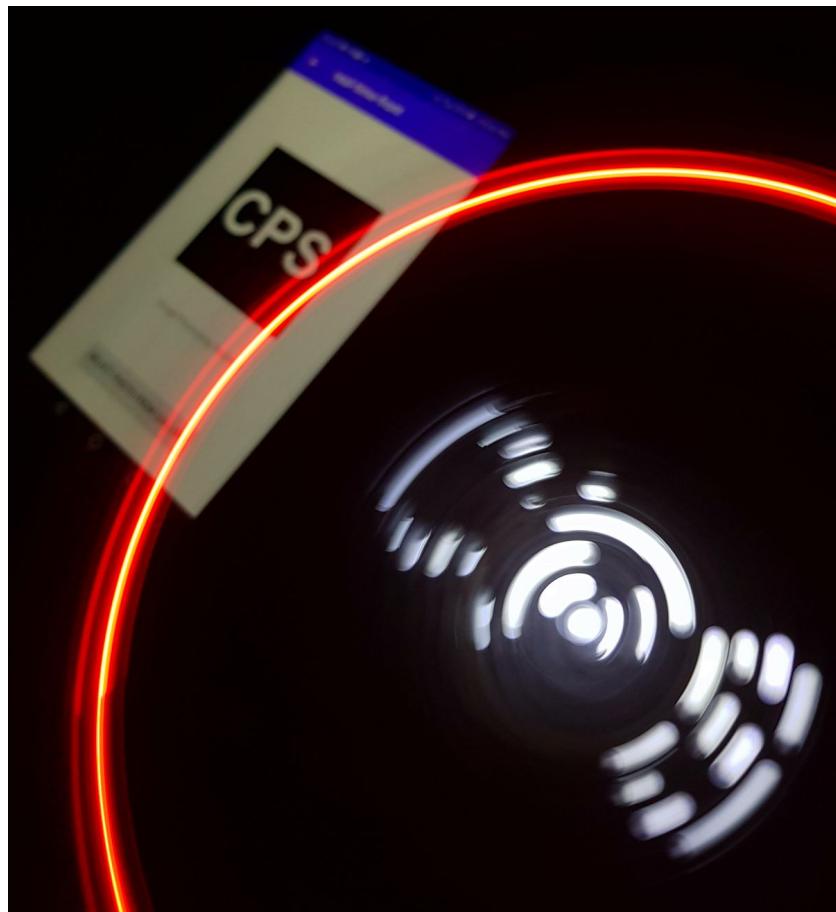
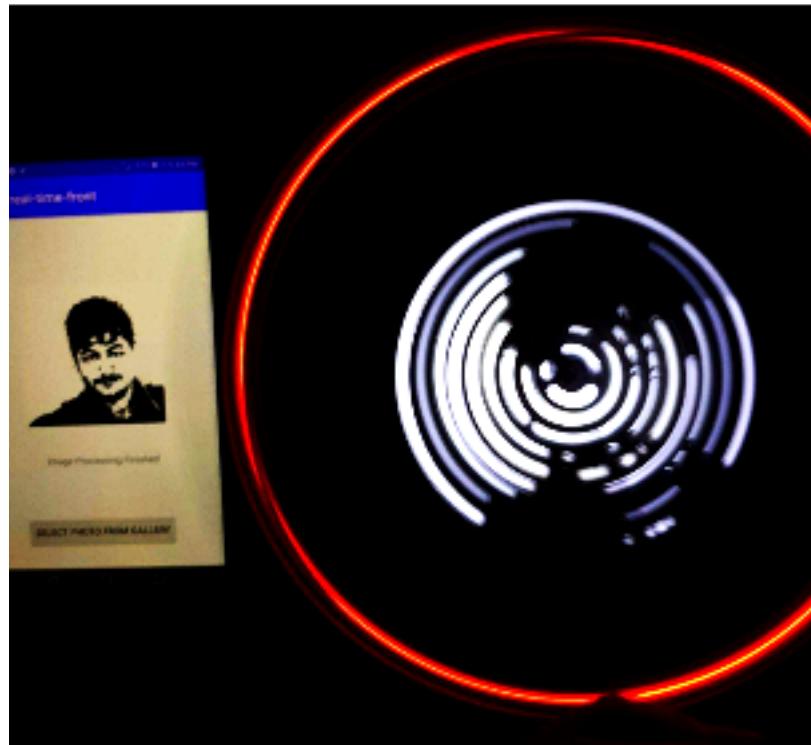




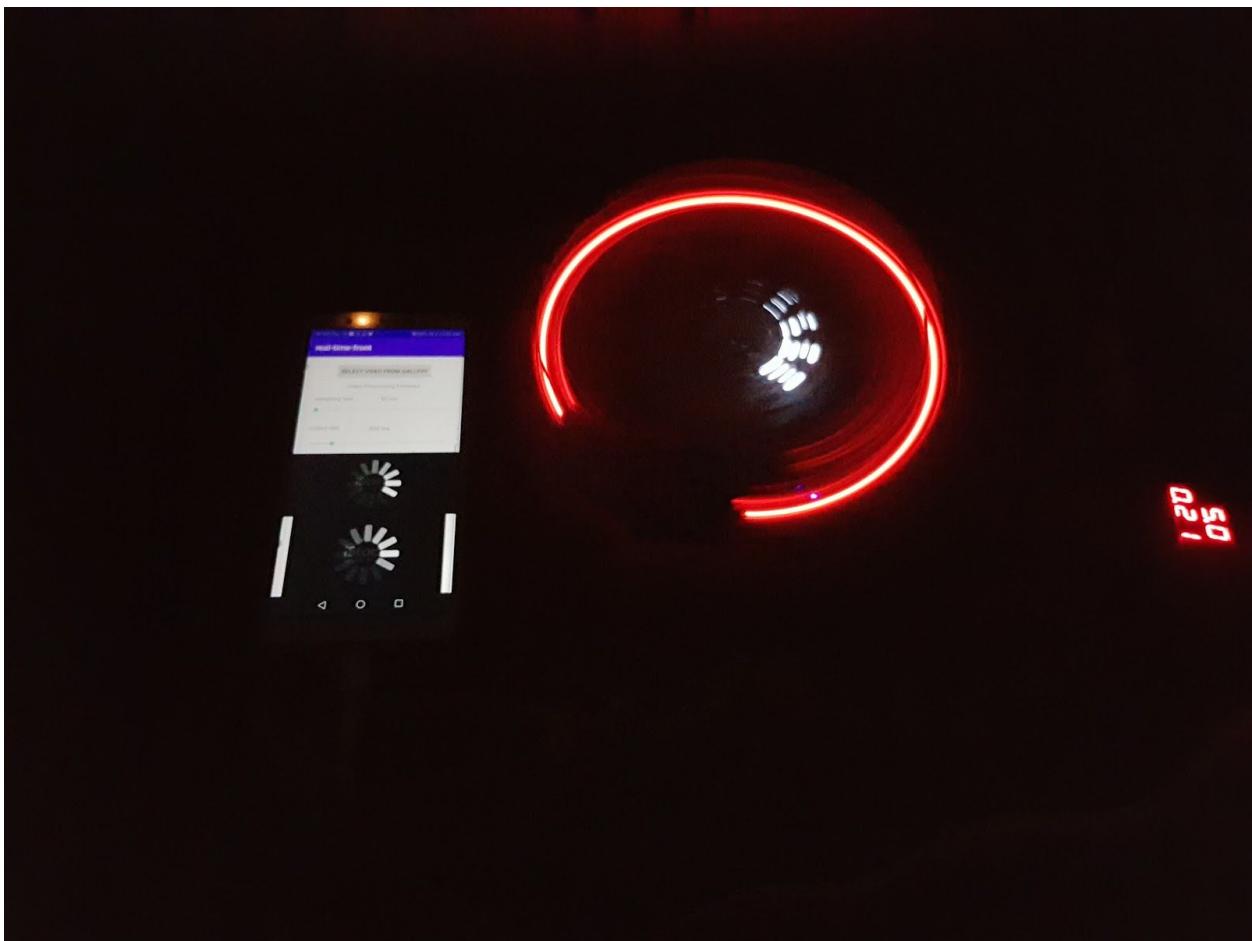








نتایج بخش فیلم:



همانطور که در بخش های قبلی توضیح داده شد، برای فیلم برداری مشکلاتی بخاطر وجود نداشتن تنظیمات پیشرفته روی این حالت وجود دارد.

البته فیلم های گویا و مشخصی از نتیجه کار گرفته شده که روی یوتیوب قرار می گیرد. برای نشان دادن مسئله، دو عکس زیر که سمت راست یک فریم از فیلم گرفته شده است دیده می شود. اگر با چشم، فیلم دیده شود، پکمن کاملا مشخص است و دارد دهانش را باز و بسته می کند، اما در یک اسکرین شات از فیلم، تنها بخشی از آن دیده می شود.



نتایج بخش :Paint



(به همان علتی که بالا توضیح داده شد، این تصویر با ترکیب چندین فریم از فیلم در photoshop ایجاد شده است)

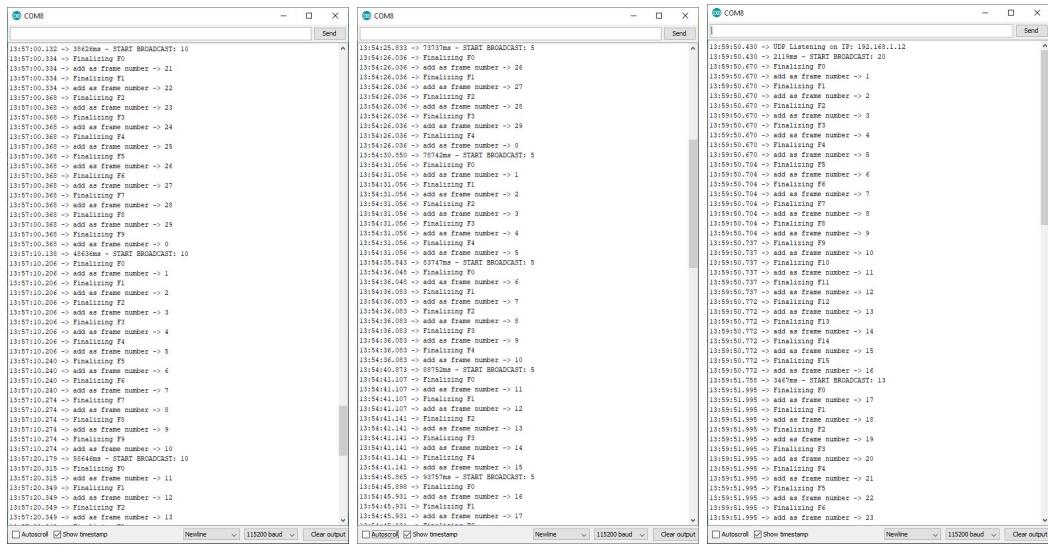
• تحلیل نتایج

□ در بخش عکس ها نتایج بسیار عالی هستند. چون تحلیل زمانی خاصی برای این حالت وجود ندارد، همین که به درستی نمایش داده می شوند کافی است.

□ در بخش فیلم، طبق اینکه روی بورد مشخص شده که فاصله دو درخواست متوالی، حداقل 1 ثانیه است و هر بار حداکثر 10 فریم درخواست شود، حداکثر فریم ریت قابل نمایش از نظر تئوری 10 فریم در ثانیه است.

البته تست هایی که با استفاده از موبایل انجام شد، قابلیت تنظیم فاصله بین هر دو فریم نمایش داده شده روی بورد را دارد که تا حدود کمترین مقدار 200ms، نمایش فیلم بدون مشکل و روان انجام می شد.

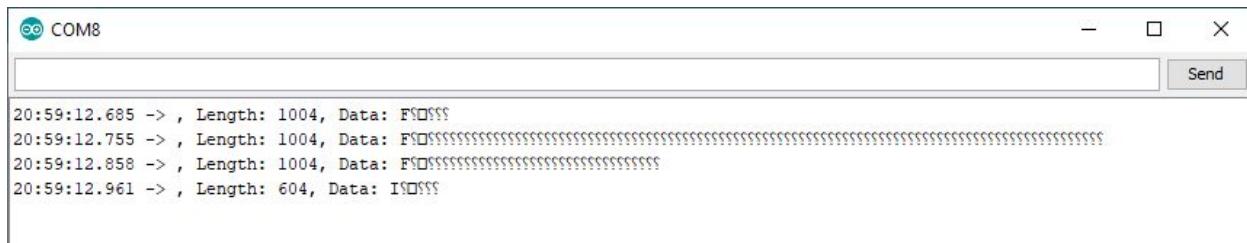
علت تنظیم عدد 10 فریم در یک ارسال، در تصاویر زیر آمده که اگر این مقدار 20 بود، تنها عدد از آن می رسید و بقیه دچار مشکل می شدند. به همین دلیل و با کمی حاشیه اطمینان، هر ارسال را 10 فریم در نظر گرفتیم.



همچین مواردی چون خارج شدن دستگاه فرستنده از شبکه نیز در نظر گرفته شده به طوری که میکروکنترلر همواره آخرین عکسی که برایش مانده را نمایش می دهد و به صورت دوره ای نیز روی پورت مشخص شده، تعداد فریمی که می خواهد را broadcast می کند. به این ترتیب هر وقت یک دستگاه به شبکه وارد شود، کار از سر گرفته می شود.

همچین اگر یک فریم به صورت ناقص برسد، حداکثر دو فریم از دست می روند و باقی فریم ها به صورت درست اضافه خواهند شد. (هر فریم به حدود 5 بخش شکسته و ارسال می شود)

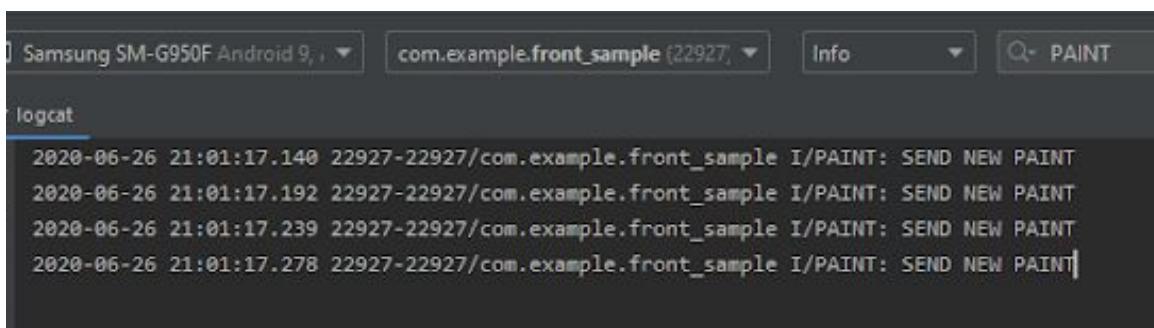
□ در بخش paint با تاخیر مناسبی نتیجه رسم روی نمایشگر ساخته شده دیده می‌شود. تاخیر این اتفاق بر اساس لاغ‌ها محاسبه شده است.



همانطور که دیده می‌شود اندروید در لحظه 9.711 تاچ را دریافت و شروع به ارسال می‌کند.
هر فریم بخاطر محدودیت سایز payload در بسته‌های udp به بسته‌های 1004 بایتی(1000 بایت دیتا و 4 بایت header مخصوص خود ما است) تقسیم و ارسال می‌شود. روی لاغ میکروکنترلر می‌بینیم که اولین بسته در 12.685 و آخرین بسته در 12.961 تحويل داده شده است.
بنابراین تاخیر ارسال و نمایش یک فریم کامل می‌شود

$$12.961 - 9.711 = 3.25\text{s}$$

برای بررسی صحت، این آزمایش را چندبار تکرار کردیم:



COM8

```

21:01:20.049 -> , Length: 1004, Data: F□
21:01:20.151 -> , Length: 1004, Data: F□?????????????????????????????????????
21:01:20.220 -> , Length: 1004, Data: F□?????????????????????????????????
21:01:20.321 -> , Length: 1004, Data: F□
21:01:20.424 -> Overload Frame!
21:01:20.424 -> , Length: 1004, Data: F□?????????????????????????????????
21:01:20.491 -> Overload Frame!
21:01:20.491 -> , Length: 1004, Data: F□?????????????????????????????
21:01:20.592 -> Overload Frame!
21:01:20.592 -> , Length: 604, Data: I□
21:01:20.661 -> , Length: 1004, Data: F□
21:01:20.728 -> , Length: 1004, Data: F□?????????????????????????????????
21:01:20.831 -> , Length: 1004, Data: F□?????????????????????????????
21:01:20.934 -> , Length: 604, Data: I□
21:01:20.967 -> , Length: 1004, Data: F□
21:01:21.069 -> , Length: 1004, Data: F□?????????????????????????????
21:01:21.137 -> , Length: 1004, Data: F□?????????????????????????
21:01:21.239 -> , Length: 604, Data: I□

```

در این موارد هم می‌بینیم که تاخیر حدود 3 ثانیه است.
البته باید این نکته را در نظر داشت که اگر درخواست از سمت میکروکنترلر بررسی شود سرعت پاسخگویی بسیار سریعتر است.

```

21:13:43.502 -> 914985ms - START BROADCAST: 10
21:13:43.706 -> , Length: 1004, Data: F\O
21:13:43.808 -> , Length: 1004, Data: F\O
21:13:43.877 -> , Length: 1004, Data: F\O
21:13:43.978 -> , Length: 604, Data: E\O?????????
21:13:44.013 -> Finalizing F0
21:13:49.022 -> 920515ms - START BROADCAST: 10
21:13:49.231 -> , Length: 1004, Data: F\O
21:13:49.335 -> , Length: 1004, Data: F\O
21:13:49.404 -> , Length: 1004, Data: F\O
21:13:49.507 -> , Length: 604, Data: E\O?????????
21:13:49.541 -> Finalizing F0
21:13:54.560 -> 926046ms - START BROADCAST: 10
21:13:55.275 -> , Length: 1004, Data: F\O
21:13:55.379 -> , Length: 1004, Data: F\O
21:13:55.448 -> , Length: 1004, Data: F\O
21:13:55.553 -> , Length: 604, Data: E\O?????????
21:13:55.586 -> Finalizing F0
21:14:00.599 -> 932085ms - START BROADCAST: 10
21:14:00.804 -> , Length: 1004, Data: F\O
21:14:00.907 -> , Length: 1004, Data: F\O
21:14:00.974 -> , Length: 1004, Data: F\O
21:14:01.078 -> , Length: 604, Data: E\O?????????
21:14:01.113 -> Finalizing F0
21:14:06.114 -> 937615ms - START BROADCAST: 10
21:14:11.133 -> 942616ms - START BROADCAST: 10
21:14:11.167 -> , Length: 1004, Data: F\O
21:14:11.272 -> , Length: 1004, Data: F\O
21:14:11.341 -> , Length: 1004, Data: F\O
21:14:11.444 -> , Length: 604, Data: E\O?????????
21:14:11.479 -> Finalizing F0

```

همانطور که دیده می شود از زمان broadcast کردن که همان ارسال درخواست از طرف بورد است، تا گرفتن کامل یک فریم(شامل چند بسته udp)، تنها حدود

$$44.013-43.502 = 511\text{ms}$$

$$49.541-49.022 = 519\text{ms}$$

طول می کشد، با اینکه این زمان، مدت رفت و برگشت بسته است نه فقط یک ارسال. علت محاسبات اشتباه قبلی که 3 ثانیه تاخیر را نشان می داد می تواند موضوع سینک نبودن ساعت اندروید و بورد باشد. چون مطمئن هستیم که رفت و برگشت بسته 500ms است پس قطعاً یک مسیر آن باید کمتر از 500ms باشد.

بنابراین تاخیر ارسال تا دریافت یک فریم کامل(چند بسته) از اندروید به بورد حدوداً 250ms است.

البته باید توجه داشت که در هر درخواست حداقل 10 فریم گرفته می شود که حدودا 40 بسته است.

```

COM8
Send
13:57:00.132 -> 38626ms - START BROADCAST: 10
13:57:00.334 -> Finalizing F0
13:57:00.334 -> add as frame number -> 21
13:57:00.334 -> Finalizing F1
13:57:00.334 -> add as frame number -> 22
13:57:00.368 -> Finalizing F2
13:57:00.368 -> add as frame number -> 23
13:57:00.368 -> Finalizing F3
13:57:00.368 -> add as frame number -> 24
13:57:00.368 -> Finalizing F4
13:57:00.368 -> add as frame number -> 25
13:57:00.368 -> Finalizing F5
13:57:00.368 -> add as frame number -> 26
13:57:00.368 -> Finalizing F6
13:57:00.368 -> add as frame number -> 27
13:57:00.368 -> Finalizing F7
13:57:00.368 -> add as frame number -> 28
13:57:00.368 -> Finalizing F8
13:57:00.368 -> add as frame number -> 29
13:57:00.368 -> Finalizing F9
13:57:00.368 -> add as frame number -> 0
13:57:10.138 -> 48636ms - START BROADCAST: 10
13:57:10.206 -> Finalizing F0
13:57:10.206 -> add as frame number -> 1
13:57:10.206 -> Finalizing F1
13:57:10.206 -> add as frame number -> 2
13:57:10.206 -> Finalizing F2
13:57:10.206 -> add as frame number -> 3
13:57:10.206 -> Finalizing F3
13:57:10.206 -> add as frame number -> 4
13:57:10.206 -> Finalizing F4
13:57:10.206 -> add as frame number -> 5
13:57:10.240 -> Finalizing F5
13:57:10.240 -> add as frame number -> 6
13:57:10.240 -> Finalizing F6
13:57:10.240 -> add as frame number -> 7
13:57:10.274 -> Finalizing F7
13:57:10.274 -> add as frame number -> 8
13:57:10.274 -> Finalizing F8
13:57:10.274 -> add as frame number -> 9
13:57:10.274 -> Finalizing F9
13:57:10.274 -> add as frame number -> 10
13:57:20.179 -> 58646ms - START BROADCAST: 10
13:57:20.315 -> Finalizing F0
13:57:20.315 -> add as frame number -> 11
13:57:20.349 -> Finalizing F1
13:57:20.349 -> add as frame number -> 12
13:57:20.349 -> Finalizing F2
13:57:20.349 -> add as frame number -> 13

```

Autoscroll Show timestamp

همانطور که دیده می شود فاصله ارسال درخواست تا دریافت کامل 10 فریم حدودا به مقدار های زیر است:

$$00.368 - 00.132 = 236\text{ms}$$

$$10.274 - 10.138 = 136\text{ms}$$

این نتایج دو مورد را نشان می دهد.

● چاپ سریال، علت کندی برنامه

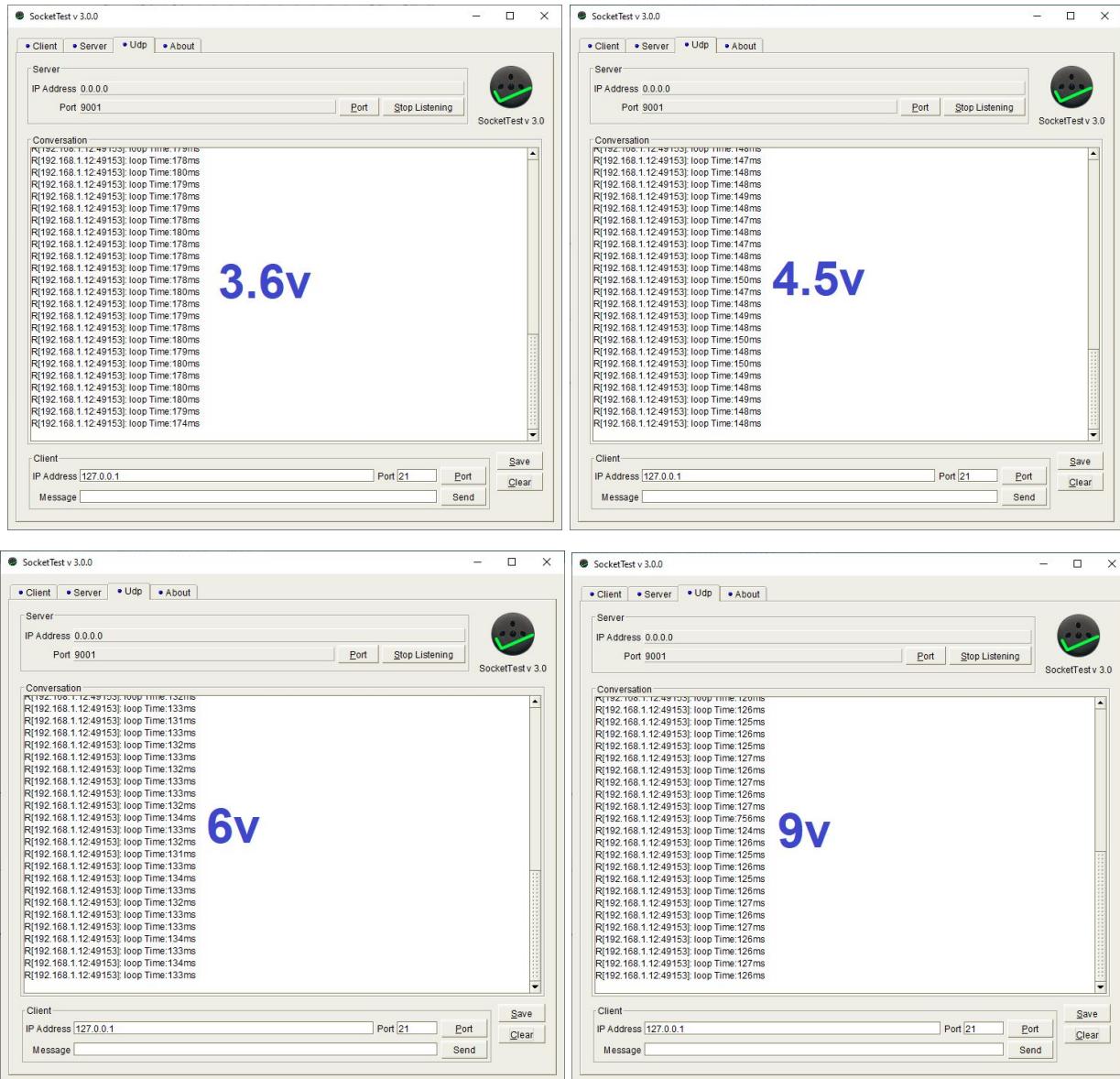
اولاً این مقادیر کمتر از 500ms که بالاتر محاسبه شد هستند. علت این مورد می تواند چاپ نکردن تمام بسته ها و مقدارشان در این حالت باشد. در آزمایش های انجام شده هم دیده شده که چاپ زیاد موارد با

استفاده از پورت سریال می‌تواند سرعت را به شدت کاهش دهد. بنابراین این زمانها نسبت به تست قبلی قابل اتکا تر هستند.

● توضیح throughput بهتر در حالت درخواست 10 فریم یکجا، به جای درخواست برای هر فریم دوما دیده می‌شود که فاصله دریافت چندین فریم بسیار کمتر از فاصله دریافت اولین فریم است. این موضوع اساس بافرینگ انجام شده است. به این شکل که به جای درخواست فریم به فریم، تعدادی فریم در یک درخواست گرفته شود و بافر شود تا نویت نمایش آن برسد. طبق تاخیر های موجود، اگر فاصله درخواست و دریافت اولین فریم را حدود 200ms بگیریم، اگر می خواستیم هر فریم را درخواست دهیم و دریافت کنیم، حداقل فاصله نمایش دو فریم، از نظر تئوری 200ms می شد، که همانطور که قبل محاسبه شد با روش بافرینگ موجود، این عدد 10 فریم در ثانیه یعنی 100ms است. قطعا در عمل، از نظر درگیری بیشتر بورد برای ارسال درخواست های متعدد و شلوغ تر شدن شبکه، روش درخواست هر فریم، سریار های بیشتری نیز خواهد داشت.

از طرفی باید به عدم قطعیت زمان دریافت هم اشاره کرد، یعنی فریم های فیلم باید با فاصله یکسان نمایش داده شوند، اگر برای این فاصله می خواستیم روی زمان ارسال و دریافت بسته ها اتکا کنیم قطعا نتیجه خوبی حاصل نمی شد. اما در روش پیاده سازی شده که همواره 20 فریم در بافر بورد موجود است و هر زمان لازم شد نمایش داده می شد این موضوع هم حل شده و فاصله نمایش فریم کاملا ثابت است.

همچنین برای تحلیل دقیق‌تر، سرعت چرخش بورد نیز اندازه گیری شد.



نتایج حدودی به دست آمده در جدول زیر نوشته شده‌اند.

	3.6v	4.5v	6v	9v
زمان هر دور	179ms	148ms	133ms	126ms
دور در ثانیه	5.58Hz	6.75Hz	7.51Hz	7.93Hz

نتیجه گیری ●

در این پژوهه، ابتدا بورد ساخته شده و مشکلات مکانیکی و اتصالات سخت افزاری حل شد. سپس نرم افزار بورد نوشته شد که شامل تنظیم زمان نمایش کدام پیکسل ها در کدام زاویه است. سپس کد اندروید زده شد، و با استفاده از برنامه نویسی سوکت، دو طرف به هم متصل شدند. سپس فیلم های سمت اندروید به اطلاعات فشرده که توسط بورد نمایش داده شوند تبدیل شدند، مهمترین مرحله این فشرده سازی تبدیل عکس از حالت مربعی به زاویه ای بود. سپس کد بورد کامل شد که فریم ها را درخواست دهد و آن ها را بافر کند، و در زمان مناسب نمایش دهد. در آخر موارد اضافه ای مانند بخش paint نیز پیاده سازی شد تا پژوهه پربار تر از مواردی که قرار بود تحويل داده شود باشد.

❖ هزینه نهایی

خیلی از موارد مثل باتری و میکروکنترلر را شخصا و از چندین سال قبل داشتیم که چون این موارد مخصوصا خود میکروکنترلر بسیار گرانتر شده بسیار به نفع ما شد.

لیست خریدهای اضافه ای که انجام شد به شرح زیر است.

مجموعا: 727.600 ریال

لیست کالاهای خریداری شده

ماقاومت 560 اهم کد 660 بسته 50 عددی | گارانتی اصالت و سلامت فیزیکی کالا

گارانتی: گارانتی اصالت و سلامت فیزیکی کالا | تعداد: 1 عدد

فروشده: کوانتوم

32,000 ریال

نوار جسب شیشه‌ای عرض 1.5 سانتی متر بسته 10 عددی | گارانتی اصالت و سلامت فیزیکی کالا

گارانتی: گارانتی اصالت و سلامت فیزیکی کالا | تعداد: 1 عدد

فروشده: جسمجی

13,000 ریال

سیم جامبر مدل MF بسته 40 عددی | سرویس ویژه دیجی کالا: 7 دوز تضمین بازگشت کالا

گارانتی: سرویس ویژه دیجی کالا: 7 دوز تضمین بازگشت کالا | تعداد: 1 عدد



فروشده: آندرودندا

159,600 ریال

269,000 ریال

دیود نوری هابراید 3 میلیپتر کد 06 بسته 50 عددی | چند رنگ | گارانتی اصالت و سلامت فیزیکی کالا

گارانتی: گارانتی اصالت و سلامت فیزیکی کالا | رنگ: چند رنگ | تعداد: 2 عدد



فروشده: آندرودندا

24,000 ریال

ماقاومت 220 اهم کد 320 بسته 50 عددی | گارانتی اصالت و سلامت فیزیکی کالا

گارانتی: گارانتی اصالت و سلامت فیزیکی کالا | تعداد: 1 عدد



فروشده: کوانتوم

24,000 ریال

ماقاومت 56 اهم کد R56 بسته 60 عددی | گارانتی اصالت و سلامت فیزیکی کالا

گارانتی: گارانتی اصالت و سلامت فیزیکی کالا | تعداد: 1 عدد



فروشده: کوانتوم

44,000 ریال

باتری لیتیوم یون مدل 651730HH ظرفیت 280 میلی آمپر ساعت | گارانتی اصالت و سلامت فیزیکی کالا

گارانتی: گارانتی اصالت و سلامت فیزیکی کالا | تعداد: 1 عدد



فروشده: راد پیام

32,000 ریال

آدمیج سشووار کد 007 | گارانتی اصالت و سلامت فیزیکی کالا

گارانتی: گارانتی اصالت و سلامت فیزیکی کالا | تعداد: 1 عدد



فروشده: کولر الکتریک

180,000 ریال

❖ پیوست های فنی

- ❖ <https://gitlab.com/mz77.co/realtime-propeller-led-display>
- ❖ https://www.youtube.com/watch?v=wflWOrsO_tA
- ❖ <https://en.wikipedia.org/wiki/ESP32>
- ❖ https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
- ❖ <https://www.espressif.com/en/products/socs/esp32/overview>
- ❖ <https://github.com/espressif/arduino-esp32/tree/master/libraries/AsyncUDP>

❖ مقالات و مراجع مورد استفاده

- ❖ https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf