



دانشگاه تهران  
دانشکده مهندسی برق و کامپیوتر



## گزارش نهایی پروژه سیستم‌های سایبر فیزیکی

بازوی رباتیک با قابلیت کنترل بی‌درنگ

اعضاي گروه :

فرزاد حبیبی (۸۱۰۱۹۵۳۸۳) ، یاسمن جعفری (۸۱۰۱۹۵۳۷۶) و صدف صادقیان (۸۱۰۱۹۵۴۱۹)

تاریخ :

۱۰ تیرماه ۱۳۹۸

اساتید :

دکتر مهدی کارگهی

دکتر مهدی مدرسی

۱۳۹۷ - ۱۳۹۸

## فهرست

۴	..... شرح کلی پژوهه	۱.
۴	..... اهداف پژوهه	○
۴	..... سخت‌افزار	○
۵	..... نرم‌افزار	○
۶	..... طراحی مفهومی	۲.
۶	..... طرح کلی	○
۶	..... ورودی	○
۷	..... پردازش	○
۹	..... شبکه و انتقال اطلاعات	○
۹	..... خروجی‌ها	○
۱۰	..... ملاحظات فیزیکی	○
۱۰	..... نوع حساسیت به زمان	○
۱۱	..... حجم داده مورد تعامل	○
۱۲	..... پیاده‌سازی	۳.
۱۲	..... شکست کار بین اعضای تیم	○
۱۳	..... سخت‌افزار	○
۱۵	• دلیل انتخاب	•
۱۸	• اتصالات	•
۱۹	• ورودی و خروجی‌ها	•
۲۰	○ نرم‌افزار	
۲۰	• پلتفرم و محیط توسعه	
۲۰	• الگوریتم سیستم	
۲۲	• ورودی‌ها و خروجی‌ها	
۲۳	• نحوه ارتباط با اجزای سخت‌افزار	
۲۳	۴. تغییرات نسبت به فاز پروپوزال	
۲۳	○ چالش‌ها پس از فاز پروپوزال	
۲۴	• تجارب ناموفق	
۲۴	۵. چالش‌های موجود در فرآیند پژوهه	
۲۴	○ چالش‌های پژوهه	
۲۴	• تجربیات منجر به تغییر در تصمیم گیری	
۲۵	۶. نزدیک‌ترین نمونه‌های مشابه	
۲۶	۷. تست عملکرد	
۲۶	○ تست شبیه‌سازی نرم‌افزاری	
۲۶	• طرح تست	

۲۶	• نحوه اجرای تست
۲۶	• نتایج تست
۲۷	• تحلیل نتایج
۲۷	◦ تست چرخش موتورهای بازوی رباتیک روی سطح مشابه با ماکت
۲۷	• طرح تست
۲۷	• نحوه اجرای تست
۲۷	• نتایج تست
۲۸	• تحلیل نتایج
۲۸	◦ تست ضبط حرکت توسط بازو
۲۸	• طرح تست
۲۸	• نحوه اجرای تست
۲۸	• نتایج تست
۲۹	• تحلیل نتایج
۲۹	◦ تست باز و بسته شدن نگهدارنده همزمان با حرکت کننده آن روی ماکت
۲۹	• طرح تست
۲۹	• نحوه اجرای تست
۲۹	• نتایج تست
۳۰	• تحلیل نتایج
۳۰	◦ تست اجرای حرکات بصورت متواالی
۳۰	• طرح تست
۳۰	• نحوه اجرای تست
۳۰	• نتایج تست
۳۰	• تحلیل نتایج
۳۰	◦ تست اتمام اجرای حرکات با فشردن دکمه اجرا برای بار دوم
۳۰	• طرح تست
۳۱	• نحوه اجرای تست
۳۱	• نتایج تست
۳۱	• تحلیل نتایج
۳۱	.۸ هزینه نهایی
۳۲	.۹ پیوستهای فنی
۳۴	.۱۰ مقالات و مراجع مورد استفاده

## ۱. شرح کلی پروژه

بازوهای رباتیک جهت جابه‌جا کردن اجسام در صنعت‌های مختلف به کار می‌روند. این بازو حرکات یک ماکت مشابه خود با اندازه دلخواه را به صورت بی‌درنگ<sup>۱</sup> تقلید می‌کند. در نهایت توانایی ضبط کردن حرکات انجام شده در یک بازه زمانی و تکرار آنها به صورت دقیق را دارد.

از آنجا که به علت صرفه‌جویی در منابع، همیشه این امکان وجود ندارد که ماشین‌هایی تک‌منظوره ساخته شوند، طراحی ماشین‌هایی که بتوانند برنامه‌ریزی شوند و کارهای موردنظر را یاد گرفته و سپس به تعداد دلخواه تکرار کنند، می‌تواند سودمند باشد. همچنین این کارها ممکن است نیاز به اجرا در ابعاد بسیار بزرگ یا کوچک داشته باشند و کنترل آنها از طریق یک ماکت با اندازه دلخواه، بسیار ساده‌تر و مفید می‌باشد. در نتیجه می‌توان با بهبود و تغییر این بازوی رباتیک برای کارکردهای متفاوت، استفاده‌های بسیاری در صنایع مختلف برای آن یافت.

### ○ اهداف پروژه

- کمک به صنعت پزشکی جهت جراحی‌های حساس که نیاز به دقت بسیار زیاد در ابعاد کوچک دارند. جراح می‌تواند با یک ماکت بزرگ‌تر به جراحی بپردازد و جراحی توسط یک بازوی اصلی در ابعاد کوچک‌تر انجام شود.
- استفاده در کارخانه‌ها برای انجام فرآیندهای تکراری و کارهایی که نیازمند نیروی زیاد هستند. فرآیندهای تکراری از این جهت ساده‌تر می‌شود که مهندس می‌تواند با استفاده از ماکت، یک کار را ضبط کند و به صورت عینی نتیجه آن را مشاهده کند. از طرفی کارهایی که نیازمند نیروی زیاد هستند هم می‌توانند با استفاده از ماکت انجام شوند و بازوی اصلی با قدرت بیشتر آن کار را انجام دهد.
- استفاده در محیط‌های خطرناک که قابلیت حضور در آن‌ها وجود ندارد. مهندسین با استفاده از ماکت‌های مشابه بازوی اصلی می‌توانند آن‌ها را در محیط‌های خطرناک کنترل کنند.

### ○ سخت‌افزار

واحد پردازشی در این سیستم بورد آردوئینو<sup>2</sup> مدل Uno می‌باشد. این بورد از یک پردازنده با فرکانس 16MHz استفاده می‌کند.

سنسورهای مورد استفاده در این سیستم <sup>۴</sup> عدد پتانسیومتر<sup>۳</sup> با ظرفیت 20K می‌باشند. با تغییر مقدار مقاومت آن‌ها حالت فعال‌کننده‌ها<sup>۴</sup> تغییر داده می‌شود.

فعال‌کننده‌های اصلی در این سیستم چهار موتور Servo مدل SG90 Tower Pro می‌باشند. این موتورها ظرفیت ۱۸۰ درجه دارند و توانایی چرخش در این بازه را دارند.

<sup>1</sup> Real Time

<sup>2</sup> Arduino Uno

<sup>3</sup> Potentiometer

<sup>4</sup> Actuator

برای تامین نیروی موردنیاز چهار موتور، از ۴ باتری قلمی استفاده شده است. همین طور جهت ارسال سیگنال‌های کنترلی از ۲ دکمهٔ فشاری<sup>۵</sup> استفاده می‌شود.

برای ساخت ماتک و بازوی اصلی هم از یک ورقه چوب بالسا با ضخامت ۵ میلی‌متر استفاده شده است.

نرم افزار

زبان برنامه‌نویسی مورد استفاده برای آردوئینو زبان C++ می‌باشد. برای استفاده از این زبان و Program کردن روی بورد آردوئینو، در Visual Studio Code محیط ویرایشگر از افزونه‌ی Platform IO استفاده شده است. همینطور از کتابخانه‌های این پلتفرم برای بهره‌وری بهتر استفاده کردیم.

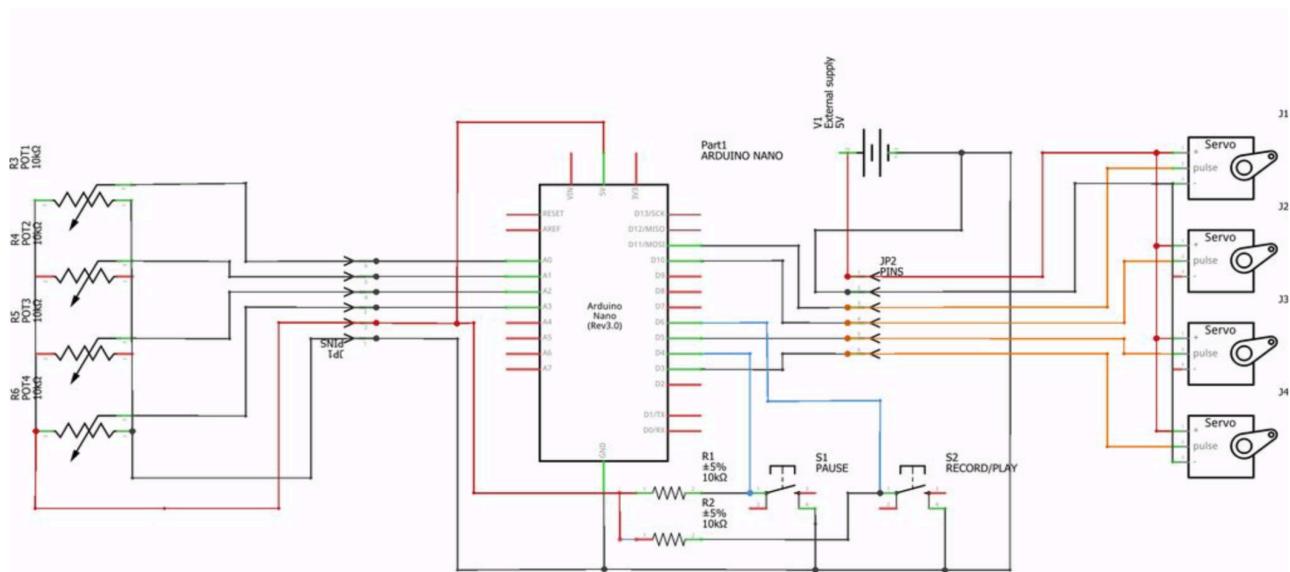
برای شبیه‌سازی کد نوشته شده به زبان C++ از نرم‌افزار Proteus استفاده شده است.

در Proteus ، ۴ سروو موتور و یک آردنینو Uno را در محیط آوردمیم. پایه‌های اول (بالایی) آنها را به VCC و پایه‌سوم آنها را به Ground وصل کردیم و پایه‌های دوم این چهار سروو را به پایه‌های ۳، ۵، ۶ و ۱۰ برد آردنینو (که از پایه‌های PWM برد هستند) متصل می‌کنیم.

در ادامه ۴ پتانسیومتر را به محیط اضافه می‌کنیم. یکی از سرهای آن‌ها را به VCC و سر دیگرشان را به Ground متصل می‌کنیم و خروجی هایشان به پایه‌های A0، A1، A2 و A3 برد آردینو وصل می‌کنیم.

همچنین تمامی Ground ها را به پایه GND برد آردینو و تمامی VCC ها را به VCC برد آردینو وصل می کنیم.

و در نهایت برای شبیه سازی دو دکمه کنترلی ربات، دو push button را به محیط اضافه می کنیم و یک سر آن ها را به دو پایه ۸ و ۹ برد که از جنس pull up هستند متصل می کنیم و سر دیگر شان را به Ground وصل می کنیم.



## 5 Push Button

## ۲. طراحی مفهومی

### ○ طرح کلی

اجزای سخت افزاری طرح به صورت بالا با یکدیگر ارتباط دارند. چهار موتور servo به پایه‌های دیجیتال<sup>6</sup> بورد آردوئینو متصل هستند و چهار پتانسیومتر روی ماکت نیز به پایه‌های آنالوگ<sup>7</sup> وصل شده‌اند. همچنین دو پایه دیجیتال از برد به دو کلید وصل شدند، یکی از آنها برای کنترل شروع یا قطع ضبط حرکات و دیگری برای اجرای حرکات ضبط شده توسط بازوی اصلی می‌باشد. تمامی ارتباط‌های قطعات سیستم به وسیله سیم است.

بطور کلی، عملکرد اجرا به این صورت است که ورودی‌ها از پتانسیومترها خوانده شده و به آردوئینو داده می‌شود. این اطلاعات، ابتدا فیلتر شده و نویزهای آن تا حد ممکن حذف می‌شود و سپس پردازش‌های لازم روی این اطلاعات توسط برد آردوئینو انجام می‌گیرد. سپس جایگاه جدید محاسبه شده موتورها به موتورها منتقل می‌شود.

حالت دو کلید فشاری در بازه‌های زمانی مشخص (۲۰ میلی ثانیه یکبار) بررسی می‌شود: اگر کلید اول در زمانی که سیستم در حالت ضبط کردن نیست فشرده شود، سیستم به حالت ضبط کردن می‌رود و حرکات انجام شده در این بازه را فشرده‌سازی و ذخیره می‌کند. اگر کلید اول زمانی که سیستم در حالت ضبط کردن است فشرده شود، سیستم از حالت ضبط کردن بیرون می‌آید. اگر کلید دوم فشرده شود، بازوی رباتیک حرکات ضبط شده در حافظه‌اش را مشابه با همان حرکات انجام می‌دهد.

### ○ ورودی

۱. برای دریافت ورودی‌ها در این سیستم از پتانسیومترهای ۲۰ کیلو اهم استفاده می‌شود. در واقع در ماکتی که ربات وظیفه تقلید از آن را دارد چهار عدد پتانسیومتر وجود دارد که با فرکانس نمونه‌برداری مناسب (فرکانس ۲۵ هرتز) مقدار این پتانسیومترها خوانده می‌شود. خواندن از پتانسیومترها دو فاز دارد:

• **Analog Read** که به منظور خواندن از مقدار پتانسیومترها و دریافت مقاومت آن‌ها می‌باشد. که کمیتی آنالوگ است.

• **Mapping** در این مرحله باید مقدار آنالوگ ورودی پس از اعمال فیلترها به مقادیر دیجیتالی که قابلیت انجام محاسبات داشته باشند تغییر داده شوند.

برای انجام عملیات نگاشت<sup>8</sup> از تابع Map در زبان برنامه‌نویسی C++ مربوط به آردوئینو استفاده می‌کنیم. نگارش<sup>9</sup> این تابع به صورت زیر می‌باشد.

#### Syntax

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

#### Parameters

**value**: the number to map.

**fromLow**: the lower bound of the value's current range.

**fromHigh**: the upper bound of the value's current range.

**toLow**: the lower bound of the value's target range.

**toHigh**: the upper bound of the value's target range.

طبق سند اطلاعات<sup>10</sup> این تابع مقدار را از بازه‌ی [fromLow, fromHigh] به بازه‌ی [toLow, toHigh] نگاشت می‌دهد.

با توجه به زاویه‌ای که هر کدام از پتانسیومترها می‌تواند بچرخد بازه‌ی اولیه متغیر است اما بازه نهایی به دلیل این که پتانسیومترها نهایت ۱۸۰ درجه می‌چرخند، از ۰ تا ۱۸۰ درجه محدود می‌باشد.

۲. دومین ورودی در سیستم، دکمه‌های فشاری برای دریافت فرمان ضبط و اجرا می‌باشد. روش دریافت وضعیت این دکمه‌ها سرکشی<sup>11</sup> می‌باشد، یعنی سر زمان‌هایی با فرکانس مناسب هر بار مقدار این سنسور خوانده می‌شود تا در صورت لزوم عملیات متناظر با آن انجام شود.

این دو سنسور، سنسورهای ساده‌ای هستند که به سادگی در بازار قابل تهیه می‌باشند. ضمن این که سنسورهای پتانسیومتر تنوع زیادی دارند و می‌توان از پتانسیومترهایی با مقاومت کمتر هم بهره برد. اما پتانسیومترهای ۲۰ کیلو اهمی به دلیل مقاومت بالا خواندن از آنها راحت‌تر می‌باشد.

## ○ پردازش

قسمت پردازشی در این بازوی رباتیک، بورد آردوئینو مدل Uno می‌باشد. این بورد بر پایه‌ی میکروکنترلر مدل ATMega328 می‌باشد و ۱۴ پین ورودی/خروجی دارد. و از یک تشیدگر سرامیکی ۱۶ مگاهرتز بهره می‌برد. برای استفاده از این بورد از سندهای زیر استفاده شد.

<https://www.arduino.cc/en/Main/arduinoBoardUno/>

[https://www.arduino.cc/en/uploads/Main/Arduino\\_Uino\\_Rev3-schematic.pdf](https://www.arduino.cc/en/uploads/Main/Arduino_Uino_Rev3-schematic.pdf)

دلیل اصلی انتخاب این مدل از واحد پردازشی این است که این برد آردوینو دارای یک ADC<sup>12</sup> می‌باشد که به تعدادی پایه ورودی روی برد وصل شده‌اند. پایه‌های A0 تا A5 که با analogRead تابع از آن استفاده می‌شود. این prescaler می‌توان از مقادیر متنوعی ازها بین ۲ تا ۱۲۸ استفاده کرد که سرعت کلک<sup>13</sup> پردازنده را برای بدست آوردن سرعت ADC clock تقسیم می‌کند.

پس مقدار زمان تبدیل<sup>14</sup> با بردمان که کلک پردازنده‌اش ۱۶ مگاهرتز است به شکل زیر تعیین می‌شود:

<sup>10</sup> <https://www.arduino.cc/reference/en/language/functions/math/map/>

<sup>11</sup> Polling

<sup>12</sup> Analog to digital converter

<sup>13</sup> Clock

<sup>14</sup> Conversion

### Prescaler

$$2 \cdot 13 \cdot \frac{1}{16E6} = 0.000001625sec(1.623\mu sec)$$

$$4 \cdot 13 \cdot \frac{1}{16E6} = 0.00000325sec(3.25\mu sec)$$

$$8 \cdot 13 \cdot \frac{1}{16E6} = 0.0000065sec(6.5\mu sec)$$

$$16 \cdot 13 \cdot \frac{1}{16E6} = 0.000013sec(13\mu sec)$$

$$32 \cdot 13 \cdot \frac{1}{16E6} = 0.000026sec(26\mu sec)$$

$$64 \cdot 13 \cdot \frac{1}{16E6} = 0.000052sec(52\mu sec)$$

$$128 \cdot 13 \cdot \frac{1}{16E6} = 0.000104sec(104\mu sec)$$

پس برای تعداد تبدیل در واحد زمان خواهیم داشت:

Prescaler	Conversions/sec
2	615,385
4	307,692
8	153,846
16	76,923
32	38,462
64	19,231
128	9,615

پس مقدار prescaler طبق ADC clock می‌شود:

Prescaler	ADC Clock Frequency (kHz)
2	8000
4	4000
8	2000
16	1000
32	500
64	250
128	125

طبق سند اطلاعات<sup>15</sup> مقدار ۱۲۸ برای prescaler حداکثر رزولوشن<sup>16</sup> را ارائه می‌دهد که مقدار پیش‌فرض<sup>17</sup> برای آن است.

## ◦ شبکه و انتقال اطلاعات

برقراری ارتباط و انتقال اطلاعات به دستگاه‌های جانبی از پورت سریال استفاده می‌شود. این اطلاعات بیشتر ماهیت اطلاعی دارند و برای اموری مثل دیباگ و یا نمایش حالت سیستم استفاده می‌شوند.

از پایه‌های با قابلیت PWM در آردوئینو برای انتقال اطلاعات به موتورها استفاده می‌شود. جهت این امر برای راحتی کار از کتابخانه‌های جانبی نظری Servo.h<sup>18</sup> استفاده می‌شود.

برای خواندن اطلاعات از سنسورها از پایه‌های آنالوگ آردوئینو استفاده می‌شود. این اطلاعات می‌توانند توسطتابع AnalogRead موجود در کتابخانه‌ی اصلی آردوئینو استفاده کرد. همین طور برای دریافت اطلاعات مربوط به دکمه‌ها می‌توان از بقیه‌ی پایه‌های دیجیتال بهره برد و از تابع DigitalRead برای خواندن آن‌ها استفاده کرد.

## ◦ خروجی‌ها

خروجی‌ها در این سیستم از طریق ۴ موتور سروو به محیط فیزیکی استفاده می‌شود.  
مدل این موتورها SG90 Tower Pro می‌باشد. سند اطلاعات این موتور در لینک زیر آمده است.

[http://www.ee.ic.ac.uk/pcheung/teaching/DE1\\_EE/stores/sg90\\_datasheet.pdf](http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf)

قسمتی از اطلاعات این سند به شرح زیر می‌باشد :

Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

همان‌طور که مشاهده می‌شود نکته قابل توجه در رابطه به این سروو موتورها زاویه‌ی ۱۸۰ درجه چرخش آنها می‌باشد. وزن این سروو موتورها به دلیل پلاستیکی بودن بسیار سبک است که برای سیستم ما که یک بازو می‌باشد مناسب‌ترین انتخاب می‌باشد.

<sup>15</sup> Datasheet

<sup>16</sup> Resolution

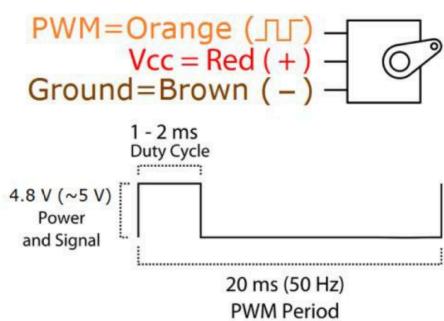
<sup>17</sup> Default

<sup>18</sup> <https://platformio.org/lib/show/883/Servo>

از نظر اندازه و وزن این سروو موتور یکی از سبک‌ترین و کوچک‌ترین ها در بازار می‌باشد. همانطور که مشاهده می‌شود به نسبت وزن کم این سروو مقدار پاور خروجی برای جابه‌جایی اشیای سبک مناسب می‌باشد.

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

درجه‌بندی سروو به شکل بالا می‌باشد که با پالس ۱۰۰۰ میکروثانیه به زاویه منفی ۹۰ و با پالس ۲۰۰۰ میکرو ثانیه به زاویه ۹۰ تغییر می‌کند. همانطور که از اطلاعات بالا مشخص است این پالس‌ها دقیق نمی‌باشند و باید با توجه به هر سروو خصوصی سازی شوند.



مقدار PWM این موتور توسط سیم نارنجی مشخص می‌شود. در بخش پیاده‌سازی باید Duty Cycle مربوط به چرخش موتور را مشخص کنیم. این مقدار توسط کتابخانه Servo در زبان برنامه‌نویسی C++ به سادگی قابل تنظیم است.

۴ عدد از این مدل سروو مورد نیاز است که به خاطر معروف بودن، به راحتی در بازار قابل پیدا کردن است.

## ○ ملاحظات فیزیکی

اگر بازوی رباتیک در محیط‌هایی که مناسب نیست قرار داده شود ممکن است آن‌طور که مورد انتظار است حرکت نکند.

این بازوی رباتیک یک حد در توانایی بلند کردن اجسام دارد و تا آن حد می‌تواند اجسام را بلند کند. این حد بسته به موتورهای استفاده شده می‌تواند مقادیر مختلفی داشته باشد.

## ○ نوع حساسیت به زمان

هدف ما در این پروژه این است که حرکت بازوی رباتیک همزمان با ماکت باشد در نتیجه لازم است اطلاعات حرکت ماکت با سرعت کافی توسط سیم به بازو انتقال یابد به شکلی که این حرکت به صورت بی‌درنگ باشد. به دلیل همین تقلید بی‌درنگ لازم است اطلاعات ورودی با نرخ مناسب از پتانسیومترها خوانده شود.

برای رسیدن به هدف مان از روش زمان بندی PTT<sup>19</sup> استفاده کردیم به این شکل که ربات سر دوره<sup>20</sup>های مشخصی وضعیت پتانسیومترها را می‌خواند و پس از انجام محاسبات لازم روی سرووهای نویسد. همچنین در دوره‌های مشخصی وضعیت کلیدهای فشاری را بررسی می‌کنیم.

<sup>19</sup> Pure Time Triggered

<sup>20</sup> Cycle

## ○ حجم داده مورد تعامل

یک مقدار بولین<sup>21</sup> برای نگهداری حالت برنامه (ذخیره کردن و یا ذخیره نکردن) مورد استفاده قرار می‌گیرد.

دو آرایه دو بعدی از بایت<sup>22</sup> به ابعاد ۴ (تعداد سروو موتورها) و یک ماکریم برای ذخیره (این جا ۲۰۰) که یکی از آن‌ها برای ذخیره کردن وضعیت هر کدام از پتانسیومترها استفاده می‌شود و دیگری برای نگهداشتن زمانی که هر کدام از سروو‌ها باید در وضعیت متناظرش در آرایه اول باقی بماند.

یک آرایه از بایت به ابعاد یک در ۴ (تعداد سروو موتورها) که برای نگهداشتن جایگاه آخرین عنصر قابل استفاده برای نوشتن در حالت ضبط کردن در آرایه هر کدام از سروو موتورها استفاده می‌شود.

یک آرایه از int به اندازه ۴ (تعداد سرووهای) برای محاسبه خروجی ای که روی سرووهای نوشته می‌شود.

یک آرایه از int به ابعاد یک در ۴ (تعداد سروو موتورها) که برای نگهداشتن جایگاه آخرین عنصر خوانده شده از آرایه هر کدام از سروو موتورها در حالت اجرای حرکات ضبط شده، استفاده می‌شود.

یک آرایه از بایت به ابعاد یک در ۴ (تعداد سروو موتورها) که برای نگهداشتن مقدار زمان اجرا شده از این حالتی که سروو در آن قرار دارد استفاده می‌شود.

یک آرایه از بایت به اندازه ۴ که حاوی مقدارهای هر کدام از سرووهای در حالت اجرای حرکات ضبط شده است.

یک int که برای نگهداشتن زمان حرکات ضبط شده استفاده می‌شود.

یک int که در زمان ضبط کردن حرکات مقدار ماکریم ای که از آرایه هر سروو استفاده شده است را بدست می‌آورد تا چک کند آیا باز هم حافظه برای ذخیره حرکات بیشتر باقی مانده است یا خیر.

یک آرایه از کلاس Servo به اندازه ۴ که برای نگهداری اطلاعات چهار سروو موتور روی بازوی رباتیک استفاده می‌شود.

### مجموع حجم استفاده شده:

مجموعاً ۱۶۱۲ مقدار بایت، ۱۰ مقدار int استفاده شده است که

$$1612 \cdot 1B + 10 \cdot 2B = 1632B$$

و به اندازه ی چهار کلاس Servo حافظه مصرف می‌کند و همان‌طور که قبلاً نیز اشاره شده است Atmega 32 دارای ۲ کیلوبایت حافظه برای ذخیره‌سازی می‌باشد، که تقریباً تمام این مقدار حافظه مصرف خواهد شد. در نتیجه سایز ماکریم برای آرایه‌ها بیشتر از ۲۰۰ نمیتواند باشد.

---

<sup>21</sup> Boolean

<sup>22</sup> Byte

### ۳. پیاده‌سازی

#### ◦ شکست کار بین اعضای تیم

کارهای مختلف بین اعضای تیم پخش شدند و هر کدام از اعضای بخش‌های مختلف پروژه را بر عهده گرفتند. این کارها به صورت جدول زیر پخش شدند.

اعضای گروه:

فرزاد حبیبی (عضو شماره ۱)

یاسمن جعفری (عضو شماره ۲)

صفد صادقیان (عضو شماره ۳)

وظیفه	افراد درگیر	زمان تخمینی	زمان صرف شده
ساخت ماکت	۱ و ۲ و ۳	۲ روز	۴ روز
ساخت بدنه اصلی روبات	۱	۲ روز	۲ روز
کد آردوئینو - بخش خواندن ورودی	۳	۱ روز	۱ روز
کد آردوئینو - بخش انجام محاسبات روی ورودی	۲	۱ روز	۲ روز
کد آردوئینو - بخش خروجی دادن به صورت همزمان	۱	۱ روز	۲.۵ روز
کد آردوئینو - بخش مربوط به ذخیره‌سازی حرکت	۱	۱ روز	۳ روز
شیوه‌سازی در نرم‌افزار Proteus	۲	کمتر از ۱ روز	کمتر از ۱ روز
فیلتر و حذف نویز	۳	۲ روز	۲ روز
بهینه‌سازی	۱ و ۲ و ۳	۱ روز	۳ روز
تست و رفع اشکالات احتمالی	۱ و ۲ و ۳	۱ روز	۳ روز
تهیه گزارش کار	۱ و ۲ و ۳	۲ روز	۲ روز

## ○ سخت افزار

سخت افزار استفاده شده به صورت دقیق به شکل زیر می‌باشد.

### آردوینو مدل Uno

مدل میکروکنترلر : ATmega328

ورودی ولتاژ : 6-20V

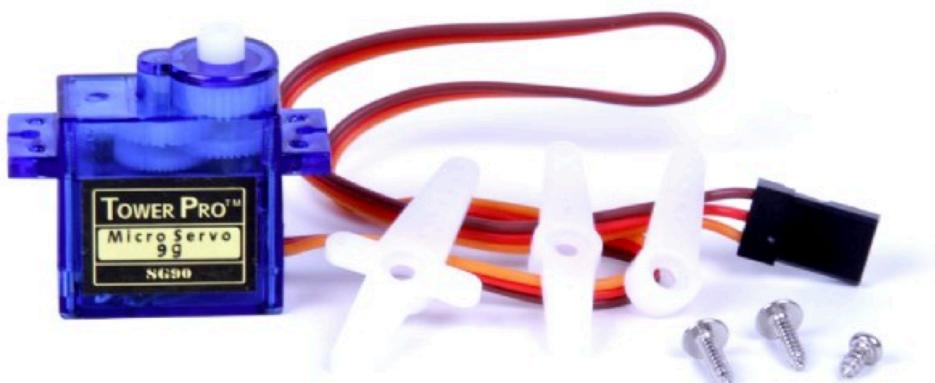
فرکانس ساعت<sup>23</sup> : ۱۶ مگاهرتز

تعداد پایه‌های ورودی/خروجی : ۱۴ عدد (۶ پایه جهت خروجی PWM به کار می‌رود)

حافظه‌ی Flash : 32KB

حافظه‌ی SRAM : 2KB

### ٤ عدد سرво مدل SG90 Tower Pro



زاویه چرخش : ۱۸۰ درجه

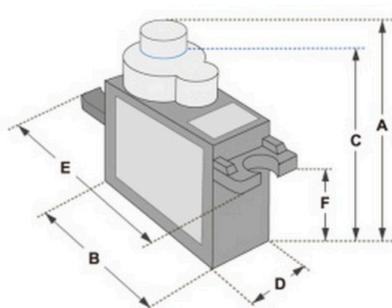
ورودی 6 V - 4.8 : ولتاژ

وزن به همراه کابل‌ها : 14.7g

گشتاور : 2.5 kg/cm

<sup>23</sup> Clock

: اندازه‌ها



A (mm) : 32

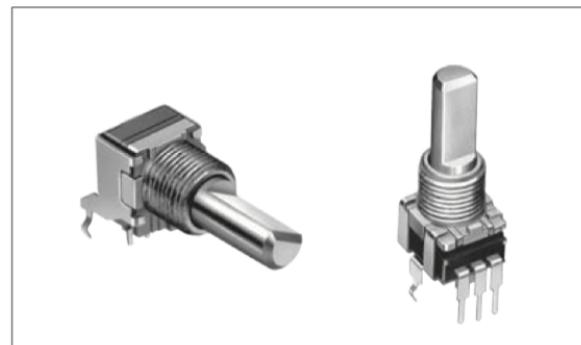
B (mm) : 23

C (mm) : 28.5

D (mm) : 12

E (mm) : 32

F (mm) : 19.5



۴ عدد پتانسیومتر

مدل : ALPS RK09L

ظرفیت : ۲۰ کیلواهرم

بیشینه‌ی زاویه‌ی چرخش : ۳۰۰ درجه

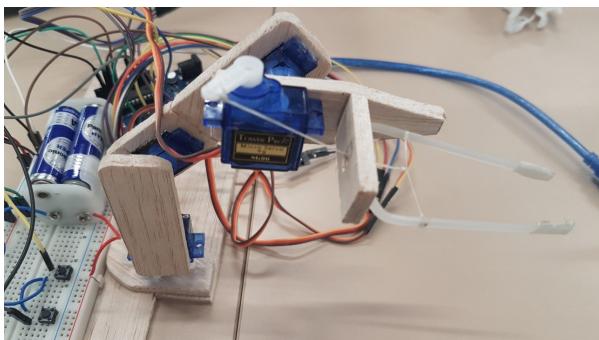


یک Breadboard و چند سیم جهت اتصال اجزای سخت‌افزاری به  
یکدیگر

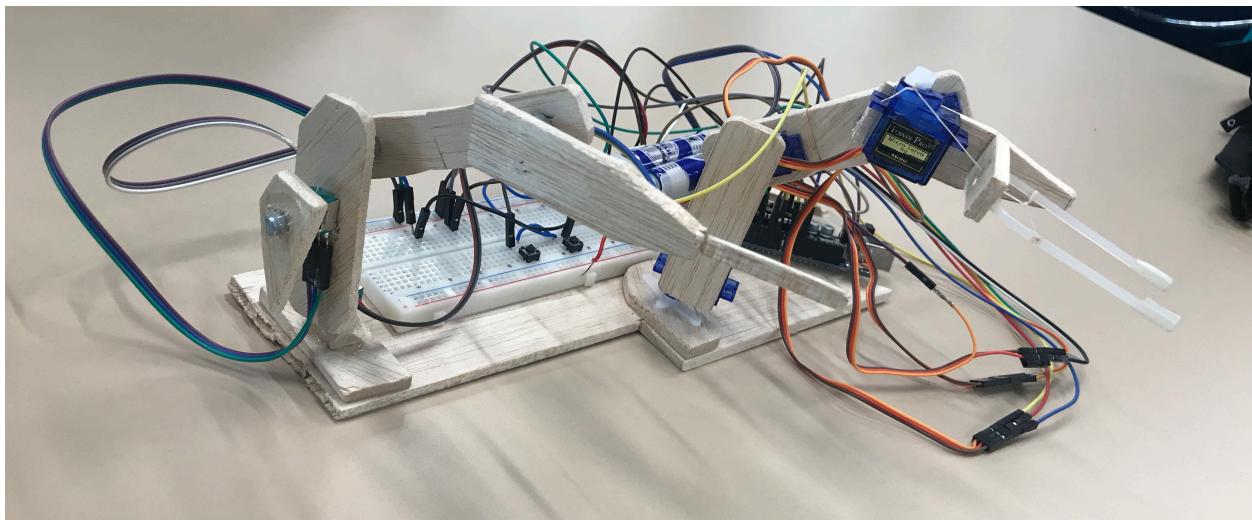
چوب بالسا جهت ساخت ماکت و بازوی اصلی



در نهایت بعد از ساخت بازوی اصلی به شکل زیر درآمد.



برای ساخت نگهدارنده<sup>24</sup> از یک تکه پلاستیک با قابلیت انعطاف استفاده شده است.  
همینطور روبات اصلی در نهایت به شکل زیر قابل نمایش می‌باشد.



## • دلیل انتخاب

دلیل انتخاب هر کدام از اجزای سخت‌افزاری به شرح ذیل می‌باشد.

### سروروهای مدل SG9 Tower Pro

طبق اطلاعاتی که در بالا آمده است این مدلی از سرورو موتور وزن بسیار کمی دارد و در روبات‌های حساس به کار می‌رود. با توجه به جنس سیستم ما این وزن بهترین وزن انتخابی بود. یک بازوی رباتیک باید نهایت از طرفی یکی از دلایل اصلی انتخاب این سروروها زاویه چرخش آن‌ها می‌باشد. یک بازوی رباتیک باید نهایت جابه‌جایی و توانایی مانور را داشته باشد. در این وزن از سروروها این مدل از سرورو بیشترین زاویه‌ی چرخش را دارد.

<sup>24</sup> Gripper

می‌توان گفت این مدل از سرو و راحت‌ترین سرو و برای کار کردن می‌باشد و بیشتر برای افراد مبتدی که قبل از آن با سرو و ها کار نکرده‌اند به کار می‌رود. یکی دیگر از دلایل اصلی انتخاب این سرو و ها قیمت آن‌ها می‌باشد. این مدل ارزان‌ترین مدل سرو و در بازار ایران می‌باشد.

### پتانسیومترهای مدل ALPS RK09L

دلیل اصلی انتخاب این پتانسیومترها کیفیت ساخت آن‌ها نسبت به قیمت آن‌ها می‌باشد. این پتانسیومترها که توسط یک شرکت ژاپنی ساخته می‌شوند از کیفیت بسیار بالایی بهره می‌برند. یکی از مشکلات اصلی پتانسیومترها هرز شدن پیچ آن‌ها بعد از مدتی استفاده می‌باشد.

یکی از دلایل دیگر کوچک بودن پایه‌ی پتانسیومترها می‌باشد. از آنجایی که آن‌ها باید در ماکت اصلی جا شوند و صرفاً حکم مفصل را برای آن داشته باشند، ضروریست که کوچک باشند.

در نهایت قیمت که یکی از فاکتورهای اصلی در انتخاب سخت افزار می‌باشد می‌تواند دلیل انتخاب این پتانسیومترها باشد.

### آردوئینو Uno

از آنجایی که سیستم ما بیشتر یک مدل از این بازوها می‌باشد، لازم نیست که از بوردهای پردازشی با توان پردازشی بالا استفاده کرد.

دلیل اصلی انتخاب بورد آردوئینو کافی بودن ورودی‌های آنالوگ آن برای کاربرد ما می‌باشد. ما به ۴ ورودی نیاز داریم که این مدل از آردوئینو ۵ ورودی آنالوگ دارد که برای نیاز ما کافیست. از طرفی در بخش طراحی مفهومی نرخ خواندن از این ورودی‌ها اندازه‌گیری شد و اثبات شد که برای نیاز ما این نرخ کافی می‌باشد.

دلیل دیگر استفاده از Arduino Uno نرخ نمونه برداری موردنیاز ما در این پژوهه است که طبق مقاله‌ای که در ادامه ذکر می‌شود ۲۵ هرتز می‌باشد. همچنین میزان حافظه رم ۲ کیلو‌بایتی آن می‌تواند نیازهای ما را تا حد خوبی برآورده می‌کند.

سرعت این پردازنده ۱۶ مگاهرتز می‌باشد و نیاز ۲۵ هرتز را برآورده می‌کند.

### چرا نرخ نمونه‌برداری ۲۵ هرتز می‌باشد؟

دلیل اصلی انتخاب این نرخ نمونه‌برداری تحقیقاتی است که در حوزه‌ی بازو های رباتیک انجام شده است. برای مثال به متن این مقاله که در رابطه با فرکانس‌های حرکتی بازوی انسان و بررسی آن می‌باشد توجه کنید:

152.3 – 154.0 and 164.0 – 169.9 Hz. The results further showed that the hand-arm is subjected to repeated extension and compression along the  $z_h$ -axis in the 10.9 – 17.3 Hz frequency range, which is close to the frequency of maximum weight (12.5 Hz) in the frequency weighting recommended in the International Standard Organization ISO 5349-1, Mechanical vibration and

طبق این مقاله بیشترین فرکانسی که بازوی انسان می‌تواند حرکت کند ۱۲/۵ هرتز می‌باشد. در غیر اینصورت آسیب می‌بیند. بنابراین ما فرکانس نمونه‌برداری را طبق نرخ نایکوئیست باید دو برابر این فرکانس انتخاب می‌کردیم. هرتر انتخاب شد ۲۵ بنابراین فرکانس نمونه‌برداری

## چرا ۲ کیلوبایت برای ذخیره‌سازی اطلاعات کافی می‌باشد؟

در ابتدا ما تصور می‌کردیم از حافظه‌ی ۳۲ کیلوبایتی آردوئینو توانایی استفاده کامل را داریم و بنابراین می‌توانیم تا حجم خیلی خوبی اطلاعات ذخیره کنیم. اما بعد از ورود به پروژه و مشاهده‌ی سند اطلاعاتی این بورد مشاهده کردیم که این حافظه تنها در اختیار BootLoader می‌باشد و ما توانایی تغییر و دسترسی به آن را نداریم. و تنها با یک حافظه‌ی رم ۲ کیلوبایتی می‌توانیم اطلاعات را ذخیره کنیم. بعد از محاسبات به این نتیجه رسیدیم که در صورتی که از این حافظه نهایت بهره‌وری را داشته باشیم می‌توانیم تا حد خوبی اطلاعات ذخیره کنیم.

یکی این که لازم نیست همه‌ی حالات سیستم را ذخیره کنیم و می‌توانیم از هر ۴ نمونه یک نمونه را ذخیره کنیم و باقی آن‌ها را تخمین بزنیم. همینطور از آنجایی که همیشه تمام سرووها حرکت نمی‌کنند می‌توان در صورتی که هر کدام از حالات تغییر نکرد، زمان آن حالت را ذخیره کرد. از طرفی از آنجایی که حالات ذخیره‌سازی سیستم اعدادی کوچکتر از ۱۸۰ درجه می‌باشند، می‌توان به جای استفاده از ساختمنداده‌هایی<sup>25</sup> مثل int یا long می‌توان از ساختمنداده‌ی byte استفاده کرد.

در صورتی که این دو شرط برقرار باشند می‌توان از این حافظه‌ی کم که تنها توانایی ذخیره‌سازی ۱۰۰ حالت را دارد در بدترین حالت ۸۰۰ حالت سیستم را ذخیره‌سازی کرد که حدود یک دقیقه می‌باشد و کافی است.

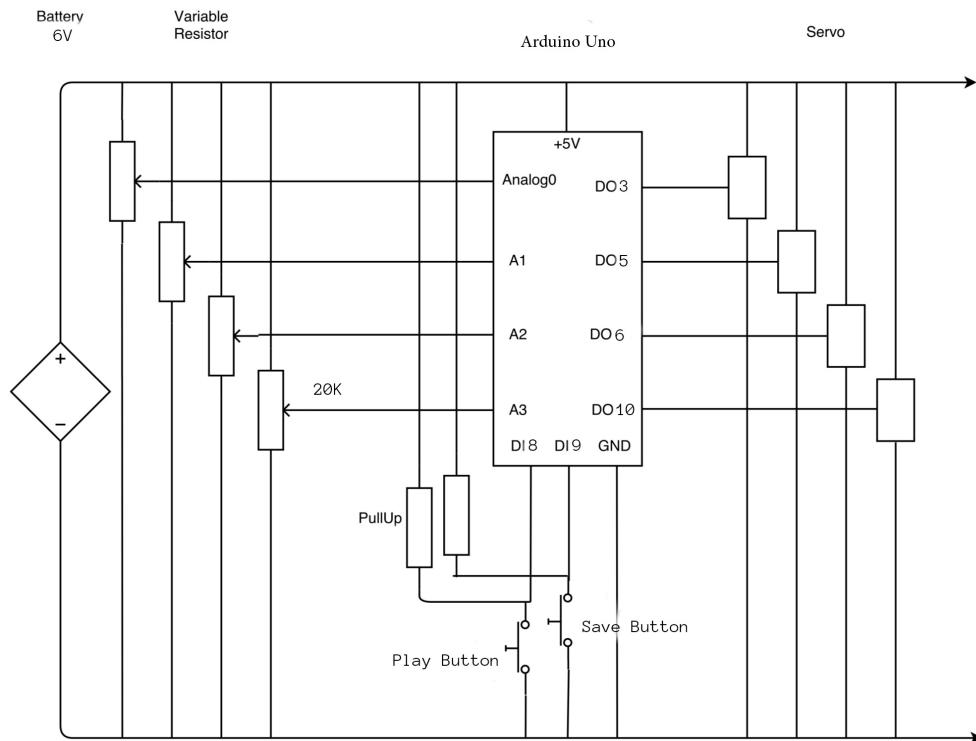
## چوب بالسا

دلیل اصلی انتخاب چوب بالسا وزن بسیار سبک جهت ساخت بازوها می‌باشد. این چوب به راحتی برش داده می‌شود و قیمت بسیار مناسبی دارد.

<sup>25</sup> Data Structure

## • اتصالات

شمای اتصالات مدار سیستم به شکل زیر می‌باشد.

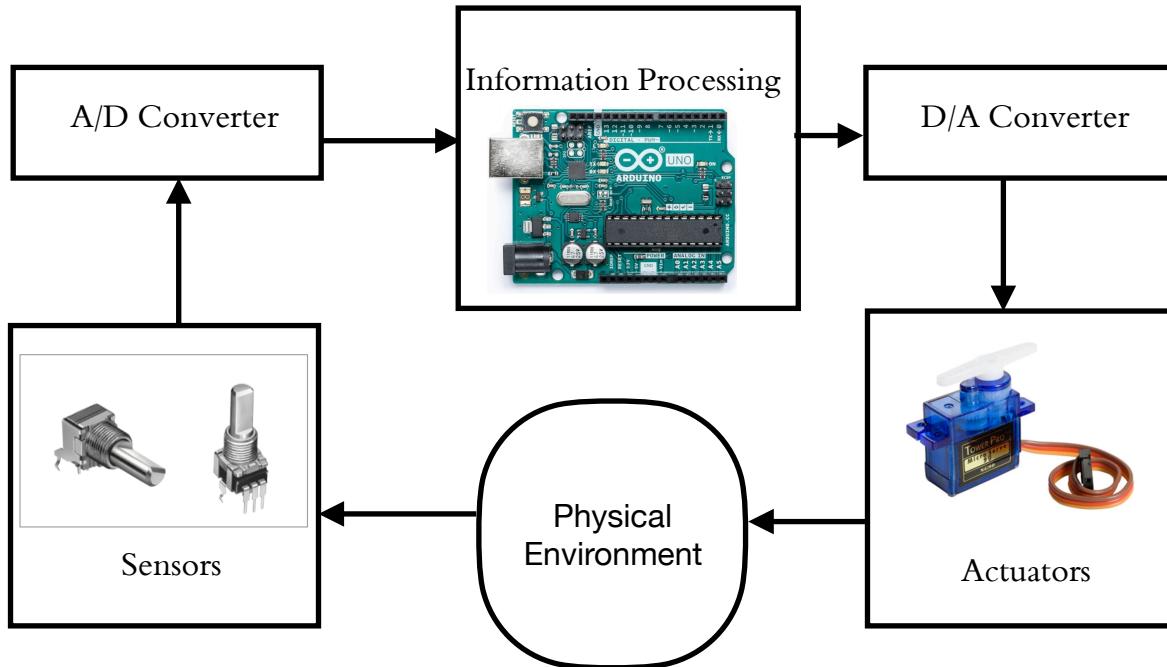


همانطور که مشاهده می‌شود ۴ پتانسیومتر توسط پایه‌های آنالوگ به بورد آردوئینو متصل می‌شوند. دو دکمه‌ی مقاومت به عنوان مقاومت ۲ فشاری هم توسط به پایه‌های دیجیتال ۸ و ۹ در بورد آردوئینو متصل شده‌اند. لازم به ذکر است در مدار اصلی به جای استفاده از مقاومت PullUp از مقاومت‌های درونی خود آردوینو استفاده شده است. موتورهای سرورو هم به پایه‌های دیجیتال خروجی متصل شده‌اند. شماره‌ی این پایه‌ها به ترتیب ۳، ۶، ۵ و ۱۰ می‌باشد. همینطور این پایه‌ها از PWM پشتیبانی می‌کنند.

برای اتصال این قسمت‌ها به یکدیگر از یک BreadBoard استفاده شده است.

- ورودی و خروجی‌ها

### Hardware In a Loop



ورودی‌های این سیستم دو بخش کلی هستند.

پتانسیومترها : این سنسورها که سنسورهای اصلی سیستم می‌باشند، اطلاعات محیط فیزیکی را دریافت می‌کند. این اطلاعات از طریق مقاومت آن‌ها سنجیده می‌شود. مقادیر آن‌ها توسط مبدل آنالوگ به دیجیتال آردوبینو به مقدار دیجیتالی که یک عدد بین ۰ تا ۱۰۲۳ می‌باشد.

دکمه‌ها : این دکمه‌ها که وظیفه‌ی دریافت سیگنال‌های کنترلی را دارند به صورت سرکشی بررسی می‌شوند.

### پردازش :

بعد از دریافت ورودی‌ها هر بار اطلاعات گرفته شده به عنوان ورودی توسط آردوبینو دریافت می‌شوند و پردازش‌های لازم روی آن انجام می‌شود. این پردازش‌ها شامل پیدا کردن مکان هر کدام از سرووها می‌باشد.

## خروجی‌ها :

در نهایت هر کدام از مقادیر پردازش شده با استفاده از مبدل دیجیتال به آنالوگ به هر کدام از موتورهای سرورو خروجی اعمال می‌شود.

خروجی‌های اصلی در سیستم ما موتورهای سرورو می‌باشد. این موتورها ورودی خود را تحت عنوان PWM دریافت می‌کنند که توسط کتابخانه‌ی آردوئینو تولید می‌شوند.

## ◦ نرم‌افزار

### • پلتفرم و محیط توسعه

پلتفرم مورد استفاده در این سیستم پلتفرم Arduino می‌باشد. این پلتفرم بر پایه‌ی زبان C++ می‌باشد.

برای محیط توسعه از افزونه‌ی PlatformIO استفاده می‌شود. این افزونه بر روی ویرایشگر Visual Studio Code قابل نصب است و امکاناتی نظیر کتابخانه‌های مختلف و متنوع، قابلیت عیب‌یابی<sup>26</sup> و استفاده از زبان سطح بالای C++ به جای زبان سطح پایین‌تر C اشاره کرد.

وب‌سایت این افزونه در زیر آمده است :

<https://platformio.org>

برای شبیه‌سازی کد زده شده از نرم‌افزار Proteus استفاده شده است. این نرم‌افزار محیطی مشابه با محیط واقعی برای تست کردن کد ایجاد می‌کند.

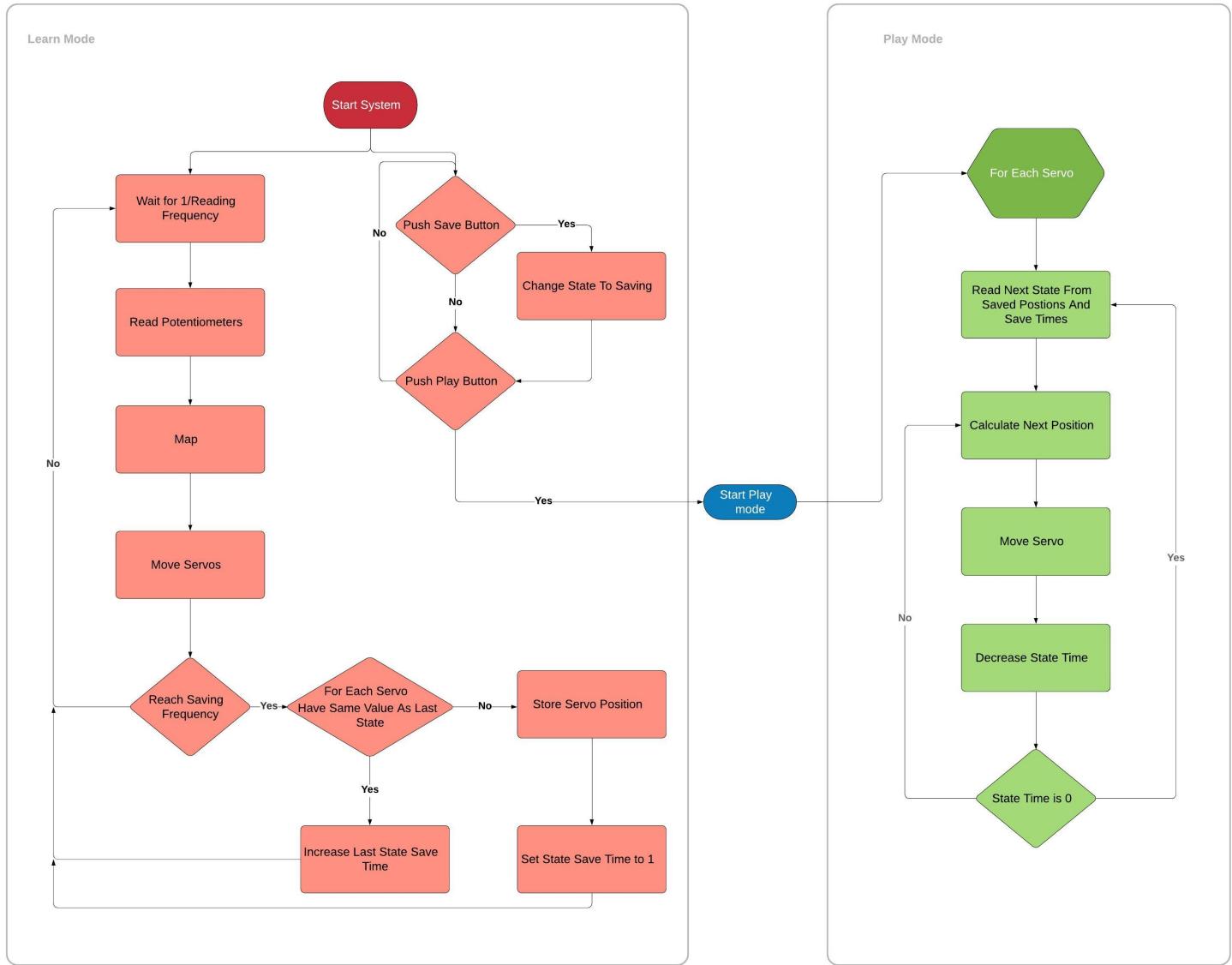
### • الگوریتم سیستم

به طور کلی الگوریتم این سیستم در زیر آمده است. این الگوریتم یک شمای کلی می‌باشد که از دو بخش ضبط و اجرا در سیستم تشکیل شده است.

در حالت کلی سیستم برای زمان‌بندی<sup>27</sup> هر کدام از تسک‌های موجود در سیستم از مدل زمان‌بندی Pure Time بهره می‌برد. Trigger

<sup>26</sup> Debug

<sup>27</sup> Scheduling



ابتدا توضیحاتی در رابطه به این مدل زمان‌بندی می‌دهیم. در این مدل از زمان‌بندی اجازه‌ی هیچ‌گونه وقفه<sup>28</sup> خارجی به سیستم داده نمی‌شود. این مدل از زمان‌بندی به صورت ثابت<sup>29</sup> است و این باعث می‌شود که یک زمان‌بندی کاملاً قطعی<sup>30</sup> ارائه شود. تمام ورودی‌ها توسط سرکشی با سیستم ارتباط برقرار می‌کنند.

بنابراین باید سر زمان‌های خاصی ورودی‌های سیستم خوانده شوند و با توجه به آن‌ها خروجی هر کدام از سرووها محاسبه شود.

<sup>28</sup> Interrupt

<sup>29</sup>Fixed

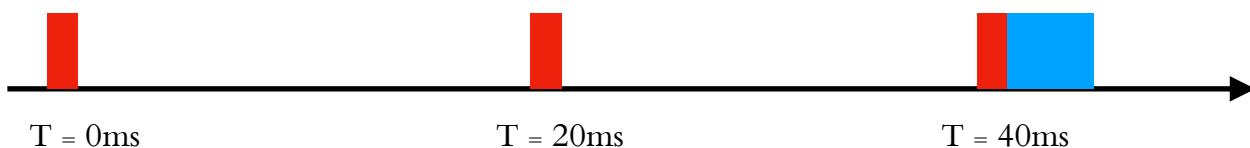
<sup>30</sup> Deterministic

برای اینکار از کتابخانه Ticker استفاده شده است. با استفاده از این کتابخانه می‌توان یک تابع را به عنوان یک Service routine تعریف کرد و سر زمان‌های مشخص به صورت پریودیک اجرا شود.

در برنامه‌ی ما دو تسک کلی وجود دارد. یکی خواندن ورودی‌ها و ذخیره‌سازی در صورت نیاز. یکی هم چک کردن وضعیت دکمه‌ها و رسیدگی به آن‌ها. این دو تایمر با دو فرکانس مختلف به شکل زیر پیاده‌سازی می‌شوند.

```
#define BUTTON_PERIOD 20 // In Milliseconds => 50 Hz
#define TIMER_PERIOD 40 // In Milliseconds => 25 Hz
Ticker read_timer(readAndWrite, TIMER_PERIOD, ENDLESS_TIMER, MILLIS);
Ticker button_timer(checkButton, BUTTON_PERIOD, ENDLESS_TIMER, MILLIS);
```

بنابراین زمان‌بندی کلی سیستم به شکل زیر انجام می‌شود.



در این شکل تسک‌های آبی خواندن از پتانسیومترها، نوشتن در سرووها و در صورت لزوم ضبط آن‌ها می‌باشد. و تسک‌های قرمز مربوط به چک کردن دکمه‌ها می‌باشد.

برای ذخیره‌سازی اطلاعات در هنگام ضبط از دو آرایه دو بعدی برای هر کدام از پتانسیومترها ورودی استفاده شده است.

```
byte servoSaved[SERVO_COUNT][MAX_SAVED_VALUES];
byte servoTimePassed[SERVO_COUNT][MAX_SAVED_VALUES];
```

این آرایه‌ها برای بهترین بهره‌وری از جنس byte می‌باشند. دلیل این انتخاب استفاده از کمترین حافظه‌ی ممکن برای هر کدام از حالت‌های ذخیره‌سازی است.

یک آرایه برای ذخیره‌سازی هر کدام از حالت‌های بازو استفاده می‌شود. در صورتی که حالت جدید برای ذخیره‌سازی تکراری بود به جای ذخیره‌سازی آن در یک داده‌ی جدید در آن آرایه، در آرایه موازی خانه‌ی مربوط به آن را یکی افزایش می‌دهیم. در این حالت در صورتی که بازو ثابت بماند اطلاعات زیادی نوشته نمی‌شود و حافظه هدر نمی‌رود.

## • ورودی‌ها و خروجی‌ها

برای دریافت ورودی‌ها بسته به نوع خواندن ورودی هر کدام به یک شکل گرفته می‌شوند. ورودی‌های پتانسیومتر از آنجایی که آنالوگ می‌باشند باید از طریق تابع AnalogRead خوانده شوند. به این صورت که در هر بار خواندن از پتانسیومترها یک مقدار از بین ۰ تا ۱۰۲۳ توسط این تابع گزارش می‌شود.

برای دریافت ورودی‌های مربوط به دکمه نیز از تابع digitalRead استفاده می‌شود. در دریافت اطلاعات مربوط به دکمه ممکن است که بعد از فشردن دکمه، به دلیل سریع اجرا شدن پریوید خواندن از دکمه دوباره مقدار دکمه در حالت فشرده شده خوانده شود، در حالی که دکمه تنها یک بار فشرده شده است. برای این حالت بعد از اولین دریافت ورودی دکمه تا جایی که ورودی تغییر نکند از ورودی مربوط به آن خوانده می‌شود.

برای اعمال هر کدام از خروجی‌ها روی موتورهای سرورو از کتابخانه Servo استفاده کرده‌ایم. در این حالت برای هر کدام از سروروها به صورت جداگانه کافیست تا زاویه‌ی آن‌ها را اعمال کنیم. برای مثال فرض کنید که می‌خواهیم سرورو شماره‌ی ۳<sup>31</sup>pwm درجه باشد، در نتیجه کافی است ۱۲۰ در زاویه مربوط به write این عدد را روی سرورو کنیم. به همین علت است که هر زاویه‌ای که از ورودی می‌خوانیم ابتدا به عرض پالس<sup>32</sup> متناظرش تبدیل کرده و سپس آن را می‌نویسیم.

## ● نحوه ارتباط با اجزای سخت افزار

در ابتدا کد هر کدام از سروروها را به یک پایه pwm بورد و همچنین مینیمم pwm متناظر با کوچکترین زاویه سرورو (که این مقدار به صورت پیش‌فرض ۵۴۴ میکروثانیه است) و ماکزیمم pwm متناظر با بزرگترین زاویه سرورو (که به شکل پیش‌فرض ۲۴۰۰ میکروثانیه است) را به آن attach می‌کنیم. در نتیجه از این پس خروجی‌هایمان را به این شکل روی محیط خارج و از طریق نوشتن pwm روی پایه مربوط به آن سرورو اعمال می‌کنیم.

در همین زمان تعدادی از پایه‌های آنالوگ بورد را به صورت ورودی تعریف می‌کنیم تا از روی این پایه‌ها مقدار هر کدام از این پتانسیومترها را که عددی بین ۰ تا ۱۰۲۳ است، به عنوان ورودی سیستم بگیریم.

همچنین دو پین را به عنوان input pullup برای دو دکمه فشاری تعریف می‌کنیم و این دو نیز دو ورودی در نرم‌افزارمان هستند که کاری که سیستم باید انجام دهد را مشخص می‌کند.

پس به طور کلی سیستم با خواندن از پایه‌های ورودی‌اش که متصل به پتانسیومترها هستند، وضعیت ماکت را می‌فهمد و پس از انجام محاسبات با نوشتن روی پایه‌های خروجی که به سروروها متصل است آن‌ها را کنترل می‌کند.

## ٤. تغییرات نسبت به فاز پروپوزال

### ○ چالش‌ها پس از فاز پروپوزال

پس از تحویل فاز پروپوزال برخی از تصمیمات قبلی که بر اساس اطلاعات تئوری گرفته شده بود، در عمل با مشکل مواجه شد و برای رفع آن مجبور به ایجاد تغییراتی شدیم.

<sup>31</sup> Pulse Width Modulation

<sup>32</sup> Pulse Width

## ● تجارب ناموفق

در ابتدای پیاده‌سازی، فرض ما بر استفاده از حافظه فلاش در بورد آردوئینو بود که یک حافظه ۳۲ کیلوبایتی است و تخمین‌های خود برای حد بالای مدت ذخیره سازی را بر اساس همین مقدار در نظر گرفتیم اما در فضای عملیاتی و هنگام ذخیره‌سازی روی سخت‌افزار اصلی متوجه شدیم که این حافظه تنها برای ذخیره‌سازی مقادیر ثابت می‌تواند استفاده شود و به صورت غیرقابل تغییر است و تنها boot loader می‌تواند در آن بنویسد. بنابراین مقادیری که در لحظه از تغییرات در ماکت بدست می‌آید تا در آینده تکرار کند در این حافظه قابل ذخیره‌سازی نیست.

این مسئله چالش بزرگی برای ما بود، چراکه حافظه قابل استفاده در این زمینه حافظه تصادفی استاتیک<sup>33</sup> است که در آردوئینو مدل استفاده شده تنها ۲ کیلوبایت است و این به معنای کاهش قابل توجه حافظه و یک شانزدهم برابر شدن آن است.

با توجه به این چالش به وجود آمده، ما نیازمند روش‌هایی برای استفاده هر چه بیشتر و کارآمدتر از حافظه بودیم. به این منظور بهینه‌سازی‌هایی از جمله تخمین برخی حرکات میانی از روی حرکات قبلی و بعدی در بازه‌هایی بسیار کوتاه به جای ذخیره‌سازی تمام حرکات انجام دادیم.

هم‌چنین به جای نگهداری حرکات بسیار جزئی یا هنگام سکون بازوها این اطلاعات را ذخیره نکرده و تنها بازه‌ی زمانی که بازو در این حرکت باقی مانده است را ذخیره می‌کنیم که این منجر به صرفه‌جویی در میزان حافظه می‌شود.

علاوه براین در ابتدای اطلاعات حرکت را به صورت عدد ۱۶ بیتی نگه می‌داشتم که پس از بررسی‌های مختلف متوجه شدیم که بدون از دست دادن اطلاعات قابل توجه، می‌توان به جای ۲ بایت این اطلاعات را در ۱ بایت یا ۸ بیت ذخیره کرد که همین موضوع حافظه قابل استفاده ما را ۲ برابر می‌کند.

## ۵. چالش‌های موجود در فرآیند پژوهه

### ○ چالش‌های پژوهه

در فرآیند انجام این پژوهه، چالش‌های گوناگونی به وجود آمد که باعث تصمیم‌گیری‌هایی متفاوت و جدید شد. برخی تصمیمات اولیه منجر به نتایجی ناموفق شده و برخی دیگر می‌توانست بهبود یابد و نتایج بهتری را داشته باشد.

## ● تجربیات منجر به تغییر در تصمیم‌گیری

از جمله اولین چالش‌های مربوط به پیاده‌سازی، ساخت بدن اصلی و ماکت بود. در ساخت بدن اصلی باید از مواد اولیه سبک استفاده کرد تا موتورها بتوانند به راحتی بازوها را حرکت دهند. ایده اولیه برای این موضوع استفاده از چاپ سه‌بعدی بود که پس از بررسی‌های بیشتر به این نتیجه رسیدیم که برای نهایی نشدن طرح در گام‌های اولیه و امکان انعطاف بیشتر در صورت هرگونه تغییر در تصمیمات پیاده‌سازی بهتر است از روشی منعطف‌تر استفاده کنیم و به این منظور پس از پرس‌وجوهایی در این مورد، تصمیم به استفاده از چوب بالسا برای ساخت بدن گرفتیم. چالش بعدی در این زمینه، چگونگی برش این چوب بود. با توجه به کمبود تجربه در این زمینه، در تلاش‌های اولیه با شکست مواجه شده و هر بار در مراحل مختلف چوب‌ها می‌شکست و بازوی مناسبی ساخته نشد. تصمیم بعدی در این زمینه، استفاده

<sup>33</sup> Static Random Access Memory (SRAM)

از برش لیزر بود که این تصمیم هم عملی نشد چراکه این روش برای نوع چوب ما و ضخامت آن مناسب نبود. در نهایت پس از جستجوهای متعدد در اینترنت و مشاهده ویدیوهای مربوطه در YouTube، نحوه صحیح برش این چوب را یادگرفتیم و بازوها را ساختیم.

مسئله دیگر یافتن نرخ نمونه‌برداری مناسب بود. برای این نرخ انتظار می‌رود نمونه‌ها بیش از حد نباشد زیرا اطلاعات اضافه به معنای مصرف حافظه بیشتر و مدت ضبط کمتر است و در عین حال باید تعداد نمونه‌ها کافی باشد تا حرکات دقیق و پیوسته و بدون جهش انجام شود و حرکات ماکت و بازوی اصلی تا حد امکان به یکدیگر نزدیک باشند. در ابتدا مقادیر مختلف را برای نرخ نمونه‌برداری را امتحان کردیم و نتایج را مقایسه کردیم اما این روش تنها برای ایجاد ذهنیت مناسب بود و به اندازه کافی دقیق نبود. به این منظور با جستجوهای مختلف و خواندن بخش‌هایی از چندین مقاله حداکثر نرخ حرکت بازوی انسان که منجر به آسیب رسیدن به آن نمی‌شد را یافته و با توجه به نرخ نایکوئیست، نرخ نمونه‌برداری را تعیین کردیم.

چالش دیگری که داشتیم وجود چندین موتور بود که همه آن‌ها همانگ با یکدیگر عمل کرده و نباید روی کار همدیگر تاثیر می‌گذاشتند. در ابتدا فرض می‌کردیم که تمام موتورها می‌توانند با مقادیر اولیه یکسان مقداردهی شوند اما هنگامی که شروع به نوشتند که آن‌ها کردیم متوجه شدیم که هر یک از موتورها باید به صورت مستقل و جداگانه کالیبر شود. به این منظور مقادیر مناسب هر یک را به صورت جداگانه بدست آوردیم و محاسبات مربوط به آن‌ها را در کد به صورت آفلاین لحاظ کردیم.

## ۶. نزدیک ترین نمونه‌های مشابه

در اینترنت نمونه‌های مختلف از بازوی رباتیک وجود دارد که هر کدام ویژگی‌های مختص به خود را دارند. بخش مشترک همه آن‌ها وجود سنسورها و فعال‌کننده‌ها است. همه آن‌ها از یک سری سنسور، اطلاعات را از محیط اطراف خود می‌گیرند. برخی از آن‌ها به صورت دستی مقدار سنسورها (مثلاً پتانسیومتر) را عوض می‌کنند و برخی دیگر مانند پژوهه ما یک ماکت دارند که با حرکت دادن بازوی‌های آن ماکت، مقدار سنسور تغییر کرده و این تغییرات را به یک واحد پردازشی مانند آردوینو منتقل می‌کنند که این اطلاعات را پردازش می‌کند و اطلاعات مربوط به حرکات بازوی اصلی را محاسبه و به فعال‌کننده‌ها منتقل می‌کنند.

تفاوت اصلی این پژوهه‌ها با پژوهه‌های مربوط به بخش ذخیره‌سازی، حداکثر کردن مدت زمان ذخیره و تکرار بدون اشتباہ حرکات است.

از جمله نمونه‌های مشابه که در فرآیند انجام پژوهه مفید بودند می‌توان به موارد زیر اشاره کرد:

<https://www.youtube.com/watch?v=Ogm3ITBxkL8>

<https://www.youtube.com/watch?v=bLnAJ-mSElE>

<https://github.com/mmittek/micro-servo-robot-arm>

<https://www.robotshop.com/community/robots/show/mini-robotic-arm>

## ۷. تست عملکرد

### ○ تست شبیه‌سازی نرم‌افزاری

#### ● طرح تست

در ابتدا بعد از آماده‌سازی کد بهتر است که قبل از ورود به محیط اجرایی کد مربوطه در محیط شبیه‌سازی تست بشود. این باعث می‌شود تا از صحت کد با خبر شویم و اگر مشکلی مربوط به محیط اجرا موجود باشد آن را به راحتی متوجه شویم.

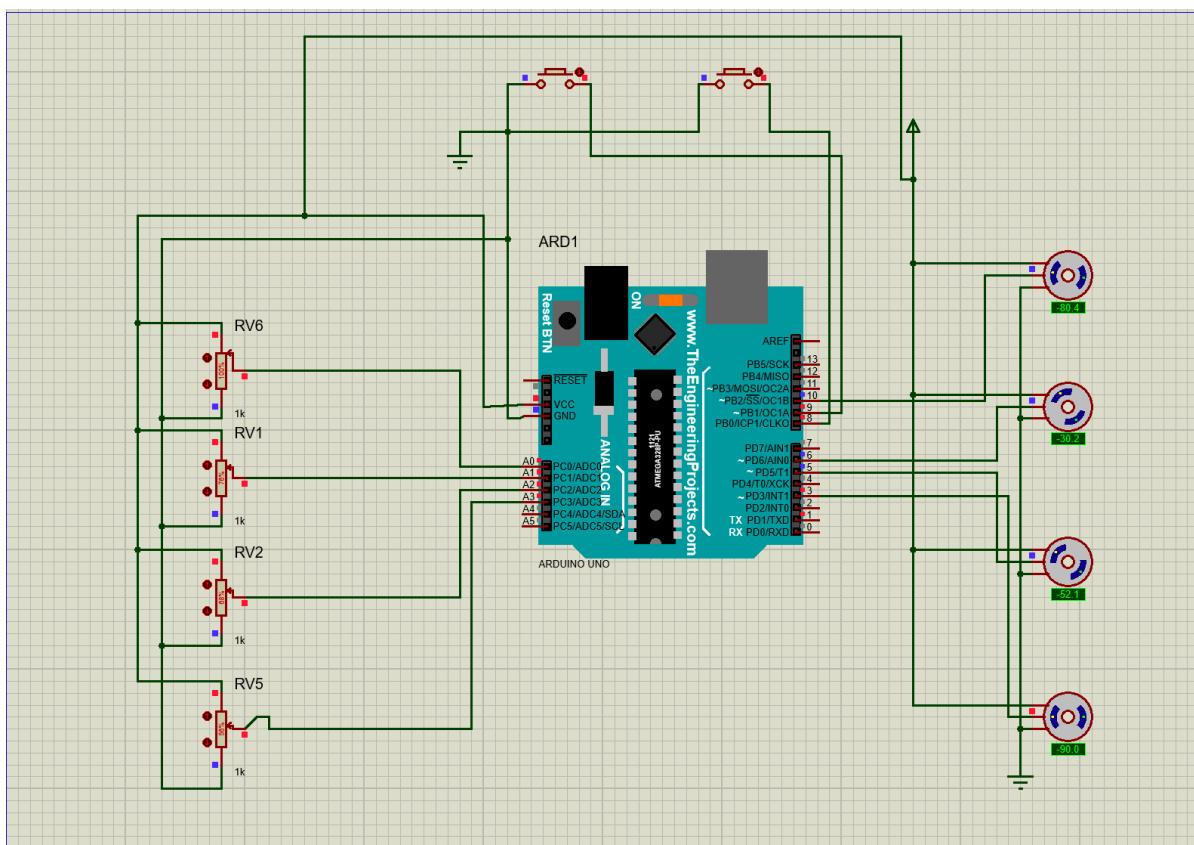
محیط اصلی سیستم را در نرم‌افزار طراحی می‌کنیم و کدهای مربوطه را قرار می‌دهیم. تغییر مقدار پتانسیومترها باید باعث تغییر مقدار سرووها شود. همینطور قسمت ضبط و اجرا نیز باید به درستی کار کند و در صورت فشردن دکمه هر کدام از اعمال انجام شوند.

#### ● نحوه اجرای تست

محیط نرم‌افزار را آماده می‌کنیم. سپس بعد از اجرای شبیه‌سازی مقدار چند تا از پتانسیومترها را تغییر می‌دهیم. در این حالت باید هر کدام از سرووها هم متناظر با آن تغییر کنند. سپس با فشردن دکمه ضبط آن را تکرار می‌کنیم و در نهایت با فشردن دکمه پخش، باید حرکت ضبط شده اجرا بشود.

#### ● نتایج تست

هر کدام از سرووها با تغییر پتانسیومترها مقدارشان تغییر می‌کند.



بعد از فشردن دکمه‌ی ضبط تغییرات انجام شده به درستی ضبط می‌شوند و با فشردن دکمه‌ی اجرا پخش می‌شوند.

## ● تحلیل نتایج

در محیط شبیه‌سازی شده کد به درستی اجرا می‌شود.

### ○ تست چرخش موتورهای بازوی رباتیک روی سطح مشابه با ماکت

#### ● طرح تست

تست چرخش بدن بازوی اصلی روی سطح مشابه با ماکت به این صورت است که طی حرکت هر یک از بخش‌های ماکت که منجر به تغییر مقدار پتانسیومتر می‌شود، موتور متناظر آن در بازوی اصلی همان میزان چرخش را تقلید کند.

بازوی اصلی شامل ۴ موتور است که ۳ تای آن‌ها حرکات چرخشی دارند و باید در این تست، موفق شوند.

ابتدا هر یک از این ۳ موتور را به صورت جداگانه تست می‌کنیم. به این صورت که هر یک از پتانسیومترها را با موتور متناظرش در نظر گرفته و بقیه را ثابت می‌گذاریم. انتظار می‌رود که با چرخش هر بخش از ماکت، تنها همان بخش از بازوی اصلی به صورت همزمان همان مقدار و در همان جهت بچرخد.

این تست ابتدا با مقادیر مرزی انجام می‌شود. به این صورت که ابتدا چرخش ۱۸۰ درجه (از ۹۰ تا ۹۰) را بررسی کرده و سپس مقدارهای میانی را بررسی می‌کنیم.

#### ● نحوه اجرای تست

به ترتیب پتانسیومتر متصل به هریک از بخش‌های ماکت را برای تست انتخاب می‌کنیم.

در مرحله اول تست، پتانسیومتر مربوطه را در یک جهت تا انتهای می‌چرخانیم. این مقدار را مقدار صفر برای موتور مربوطه در نظر گرفته و سپس پتانسیومتر را تا انتهای دیگر چرخانده و آن را مقدار ۱۸۰ برای موتور در نظر می‌گیریم. در کد این بخش، اندازه‌های ذکر شده را به عنوان مقادیر اولیه و نهایی در ایجاد تناظر گذاشته و این تناظر برای یافتن مقادیر میانی استفاده شده است.

مفصل متناظر ماکت را برای هر یک از پتانسیومترها چرخانده و درستی چرخش موتور و هم‌زمانی حرکت آن با ماکت را در مقادیر مرزی و میانی بررسی می‌کنیم.

#### ● نتایج تست

یکی از موتورها به درستی حرکت کرده ولی حرکت بقیه آن‌ها دقیق نیست و تا مرزهای خود جلو نمی‌رونند.

جهت چرخش همه موتورها درست و هم‌جهت با چرخش پتانسیومتر متناظر است.

در عمل با توجه به ساختار ماکت و بازوی اصلی، نمی‌توان برخی از بخش‌ها را تا حد مرزی پتانسیومتر چرخاند.

## • تحلیل نتایج

حرکت درست یکی از موتورها و غیر دقیق بودن حرکت بقیه آن‌ها ناشی از کالیبره نکردن مستقل هریک از آن‌ها است. هر یک از موتورها باید به تنها یک کالیبره شده و به مقادیر دقیق مربوط به پتانسیومتر مربوطه تناظر داده شود. حالیکه در تست اولیه این مقادیر تنها برای یک موتور و پتانسیومتر بدست آمده و برای بقیه هم استفاده شده بود که منجر به غیر دقیق بودن حرکات آن‌ها می‌شد.

جهت چرخش درست موتورها به این معناست که حدود مرزی بدست آمده با ترتیب درستی برای ایجاد تناظر استفاده شده‌اند.

با توجه به ساختار ماکت و بازوی اصلی، در مفصل‌ها نمی‌توان از یک انتها تا حد دیگر پتانسیومتر چرخش را انجام داد و باید این مقادیر که به عنوان حدود ایجاد تناظر در نظر گرفته می‌شوند، با مدل و کاربرد هماهنگ شوند. پس از رفع اشکالات ذکر شده، تست را تکرار کرده و نتایج موفقیت‌آمیز بود.

## ◦ تست ضبط حرکت توسط بازو

### • طرح تست

در تست اولیه مربوط به ذخیره‌سازی حرکات، هدف ذخیره حرکات و توانایی یک بار تکرار حرکات با فشردن دکمه اجرا است. به این صورت که ابتدا دکمه ذخیره‌سازی را فشار داده و ضبط آغاز می‌شود. سپس با فشردن دوباره همان دکمه پایان ضبط اعلام می‌شود.

حال برای بررسی درستی و کامل بودن اطلاعات ضبط شده، دکمه اجرا را فشار داده و تمام حرکات ضبط شده باید به ترتیب اجرا شوند و پس از تمام شدن حرکات متوقف شود.

حال اگر دکمه اجرا دوباره فشرده شود، حرکات باید بار دیگر به صورت کامل اجرا شوند.

بخش دیگر این تست، سنجش مدتی است که یک بخش می‌تواند ثابت بماند و در عمل، پس از اجرای حرکات ضبط شده مدت زمان سکون به درستی تکرار شود.

### • نحوه اجرای تست

در چندین مرحله تست را انجام می‌دهیم. ابتدا برای مدت کوتاهی و در حد ۵ ثانیه حرکات را ضبط و تکرار می‌کنیم. در هر مرحله در صورت موفقیت‌آمیز بودن ضبط قبلی، مدت ضبط را دو برابر کرده و فرآیند ذخیره و تکرار را انجام می‌دهیم.

برای بخش دوم تست، مشابه قبل عمل کرده و زمان‌های مختلف را می‌سنجیم.

### • نتایج تست

در اولین تست، تنها قادر به ذخیره ۸ ثانیه بودیم. به طوری که پس از ۸ ثانیه حرکات به درستی ذخیره شده و قابلیت تکرار داشت اما پس از آن در صورت تلاش برای ذخیره بیشتر، سیستم از کار افتاده و هیچ واکنش دیگری نشان نمی‌داد.

در بخش دوم تست، اگر هر بخش حدود ۴۰ ثانیه ساکن باشد مشکلی در تکرار آن به وجود نیامد اما پس از آن زمان آن به هم می‌ریخت.

### • تحلیل نتایج

علت کم بودن مدت زمان ذخیره‌سازی عدم امکان استفاده از حافظه ۳۲ کیلوبایتی فلاش است. از آنجاکه اطلاعات مربوط به حرکات به صورت استاتیک نیست، در حافظه SRAM که تنها دارای ۲ کیلوبایت ظرفیت است، ذخیره می‌شود.

برای حل این مشکل و استفاده بهینه از حافظه راهکارهای مختلفی ارائه کردیم که توضیح کامل آن‌ها در بخش تجارب ناموفق آورده شده است.

پس از اعمال تعییرات ذکر شده، این زمان افزایشی قابل توجه داشت. به طوری که با توجه به اقدامات ذکر شده دیگر مدت زمان ضبط، ثابت نبوده و به شدت تحت تاثیر نوع و سرعت حرکات است اما در بدترین حالت توانایی ذخیره‌ی حدود ۴۰ ثانیه را داشته و این مقدار می‌تواند تا چندین دقيقه افزایش یابد.

هم‌چنین برای بخش دوم این تست، نتایج با محاسبات تئوری انجام شده هماهنگ بود:

$$2^8 * 40 = 40960ms = 40.9s$$

## ○ تست باز و بسته شدن نگهدارنده<sup>۳۴</sup> همزمان با حرکت کننده آن روی ماکت

### • طرح تست

یکی از پتانسیومترها و موتور متناظر برای باز و بسته کردن نگهدارنده استفاده می‌شوند. به این صورت که یک نخ در اطراف سر موتور بسته می‌شود تا با چرخش موتور، کشیده یا رها شود و در نتیجه این حرکت، نگهدارنده را باز و بسته کند. مشابه بخش قبل، یک تناظر بین مقادیر پتانسیومتر و موتور ایجاد کرده و درستی حرکت و همزمانی آن را با مقادیر مختلف امتحان می‌کنیم.

### • نحوه اجرای تست

بدون ایجاد تغییر در مقادیر مربوط به بقیه پتانسیومترها در دو مرحله عملکرد نگهدارنده را بررسی می‌کنیم ، در مرحله اول، هم‌زمانی، هماهنگی و درستی جهت و اندازه بررسی می‌شود. در مرحله دوم، درستی تکرار حرکت نگهدارنده در حالت ضبط را بررسی می‌کنیم.

### • نتایج تست

در مرحله اول، تست با موفقیت پشت سر گذاشته شد و حرکات نگهدارنده با جهت درست و مقدار مناسب و به صورت همزمان با ماکت به درستی انجام شد.

<sup>34</sup> Gripper

در مرحله دوم، مشاهده شد که حرکات مربوط به نگهدارنده به درستی تکرار نمی‌شود و بعضی اوقات نگهدارنده پس از بسته شدن آن، بلا فاصله دوباره باز می‌شود و در همان حالت باقی می‌ماند.

### • تحلیل نتایج

پس از بررسی موضوع و خواندن مقادیر تناظر داده شده در موقع رخ دادن این خطأ، متوجه شدیم که مقادیر اشتباہی ذخیره و تکرار شده‌اند. درواقع در هنگام تکرار هم زمان حرکات، از آنجا که مقادیر به صورت اعداد ۱۶ بیتی خوانده و تکرار می‌شوند مشکلی پیش نمی‌آمد اما در هنگام تکرار، از آنجا که ذخیره به صورت ۸ بیتی صورت می‌گرفت، زمانی که چرخش پتانسیومتر از حد مشخصی فراتر می‌رفت، ۸ بیت برای ذخیره آن کافی نبوده و با سریز مواجه می‌شدیم. این مشکل به دلیل ایجاد تناظر نامناسب به وجود آمده و با اصلاح این مقادیر مشکل حل شده و تست با موفقیت گذرانده شد.

## ○ تست اجرای حرکات بصورت متوالی

### • طرح تست

یکی از ویژگی‌های مورد انتظار در این پژوهه، امکان تکرار حرکات ضبط شده به صورت متوالی و بدون نیاز به دخالت عامل خارجی است. به این صورت که پس از ضبط حرکات و فشردن دکمه اجرا، بازوی اصلی حرکات را به صورت متوالی تکرار کرده و تنها زمانی متوقف شود که دکمه اجرا برای بار دوم فشرده شود. هم‌چنین در طی تکرار حرکات به صورت متوالی، باید در هر دور اجرا، حرکات با همان مقادیر و سرعت اولیه تکرار شوند و نباید در اثر تکرار این مقادیر به هم بریزند.

### • نحوه اجرای تست

ابتدا یک مجموعه حرکت را انتخاب کرده و آن را ضبط می‌کنیم. سپس با فشردن دکمه اجرا، تکرار حرکات باید به صورت متوالی و با همان مقادیر و سرعت‌های اولیه انجام شود.

### • نتایج تست

حرکات به درستی و به صورت متوالی تکرار شده و تنها در صورت قطع اتصال روبات و منبع، این حرکات متوقف می‌شود.

### • تحلیل نتایج

کد مربوطه به درستی عمل کرده و حرکات با کمترین خطأ تکرار می‌شوند.

## ○ تست اتمام اجرای حرکات با فشردن دکمه اجرا برای بار دوم

### • طرح تست

در تست قسمت قبل، ذکر شد که تکرار حرکات در صورت قطع اتصال از منبع متوقف می‌گردد. در این بخش، عملکرد فشردن دکمه اجرا برای بار دوم تست می‌گردد. هم‌چنین پس از توقف، در صورت فشردن دوباره دکمه اجرا، حرکات از ابتدا دوباره آغاز می‌شود.

## • نحوه اجرای تست

ابتدا یک مجموعه حرکت را انتخاب کرده و آن را ضبط می‌کنیم. سپس با فشردن دکمه اجرا، تکرار حرکات آغاز می‌گردد. حال برای توقف اجرا بار دیگر دکمه را فشار می‌دهیم و تمامی حرکات باید متوقف شود.

## • نتایج تست

با فشردن دکمه اجرا برای بار دوم و پس از آن، هیچ تغییری در بازوی اصلی به وجود نیامد و بازو همچنان به تکرار حرکات خود ادامه داد.

## • تحلیل نتایج

پس از بررسی کد و خطایابی<sup>35</sup>، متوجه شدیم که بروز این مشکل به علت انسدادی<sup>36</sup> بودن بخشی از کد است که منجر به عدم توجه به فشرده شدن دکمه در حین اجرا می‌شد.

پس از رفع این مشکل، تست با موفقیت گذرانده شد.

## ۸. هزینه نهایی

هزینه تمامی قطعات استفاده شده در جدول زیر ذکر شده است:

نام محصول	تعداد	قیمت واحد	قیمت کل
موتور Servo SG90	۴ عدد	۲۱ هزار تومان	۸۴ هزار تومان
آردوینو Uno	۱ عدد	۷۰ هزار تومان	۷۰ هزار تومان
باتری	۱۲ عدد	۲ هزار تومان	۲۴ هزار تومان
پتانسیومتر 20K	۵ عدد	۴ هزار تومان	۲۰ هزار تومان
چسب	۱ عدد	۲ هزار تومان	۲۰ هزار تومان
سیم و دکمه	—	—	۱۹ هزار تومان
چوب بالسا	۱ ورقه	۱۶ هزار تومان	۱۶ هزار تومان
جاباتری	۱ عدد	۵ هزار تومان	۵ هزار تومان
بست کمربندی	۳ عدد	هزار تومان	۳ هزار تومان

جمع کل: ۲۶۱ هزار تومان

<sup>35</sup> debug

<sup>36</sup> blocking

## ۹. پیوست‌های فنی

```

if (maxSaved < MAX_SAVED_VALUES && saving) {
    if (checkSavingPeriod % SAVE_UNIT_PERIOD == 0) {
        updateSaved(x[0], 0);
        updateSaved(x[1], 1);
        updateSaved(x[2], 2);
        updateSaved(x[3], 3);
    }
    checkSavingPeriod++;
}

```

همان طور که در قسمتی از کد که در بالا آمده است مشاهده می‌کنید، در زمان ضبط کردن حرکات ماکت، تمامی آن‌ها ضبط نمی‌شوند بلکه آن‌ها در دوره‌ی خاصی ضبط می‌شوند تا حجم کمتری در حافظه ذخیره شود.

```

byte servoValue[4];
for (byte i = 0; i < SERVO_COUNT; i++) {
    servoValue[i] =
        servoTimePassed[i][servoIndex[i]] - servoSlotRunnedTime[i] == 1
            ? servoSaved[i][servoIndex[i]] +
                (j * ((servoSaved[i][servoIndex[i]] + 1) -
                    servoSaved[i][servoIndex[i]])) /
                    SAVE_UNIT_PERIOD)
            : servoSaved[i][servoIndex[i]];
}

```

در زمان اجرا با توجه به این‌که همه‌ی مقادیر ذخیره‌سازی نمی‌شوند، باید هر بار مقادیر ذخیره نشده تخمین زده بشوند. این تخمین به صورت بالا صورت می‌گیرد.

```

if (digitalRead(PLAY_BUTTON_PIN) == LOW) {
    while (digitalRead(PLAY_BUTTON_PIN) == LOW) {
    }
    playing = !playing;
    return;
}

```

هر کدام از دکمه‌ها ممکن است یک بار فشرده شوند در حالی که با توجه به نرخ نمونه‌برداری بالا چندین بار مقدار آن‌ها یکسان خوانده بشود. برای این‌کار از یک حلقه استفاده می‌شود و تا زمانی که مقدار خوانده شده از دکمه تغییری نکند کاری نمی‌کند.

```
#define SERVO_0_MIN_MICROSECOND 590
#define SERVO_0_MAX_MICROSECOND 2245
#define SERVO_1_MIN_MICROSECOND 800
#define SERVO_1_MAX_MICROSECOND 2400
#define SERVO_2_MIN_MICROSECOND 675
#define SERVO_2_MAX_MICROSECOND 2525
#define SERVO_3_MIN_MICROSECOND 707
#define SERVO_3_MAX_MICROSECOND 2445
```

هر کدام از سرووها با توجه به شرایط ساختشان ورودی دادن به آن‌ها فرق می‌کند. برای اینکار باید با توجه به زاویه‌ای که هر کدام از سرووها می‌توانند داشته باشند باید بیشینه و کمینه‌ی زمان ان‌ها در PWM را به آن‌ها بدهیم. برای این‌کار به صورت دستی هر کدام از اعداد مربوط به سرووها را پیدا کردیم. این اعداد در نهایت به شکل بالا می‌باشند.

```
#define POT_0_MAX 991
#define POT_0_MIN 60
#define POT_1_MAX 962
#define POT_1_MIN 100
#define POT_2_MAX 1012
#define POT_2_MIN 64
#define POT_3_MAX 230
#define POT_3_MIN 0
```

زاویه‌ی چرخش هر کدام از پتانسیومترها ۳۰۰ درجه است. این درحالیست که زاویه‌ی چرخش سرووها ۱۸۰ درجه می‌باشد. برای اینکه چرخش هر کدام از پتانسیومترها متناظر با چرخش هر کدام از سرووها باشد باید یک بازه از چرخش پتانسیومترها جهت چرخاندن سرووها استفاده شود.

```
bool hasSameValue(int newValue, int index) {
    if (lastSaved[index] == 0)
        return false;
    return servoSaved[index][lastSaved[index] - 1] >= newValue - SAVE_THRESHOLD
&&
    servoSaved[index][lastSaved[index] - 1] <= newValue +
SAVE_THRESHOLD;
}
```

گاهی اوقات داده‌ی تکراری جهت ذخیره پیش می‌آید. در این حالت نباید داده‌ی تکراری ذخیره شود. برای چک کردن این حالت خاص از تابع بالا استفاده می‌شود که با استفاده از یک Threshold کار خود را انجام می‌دهد.

## ۱۰. مقالات و مراجع مورد استفاده

از جمله مراجع اولیه و مهم این پروژه، برگه اطلاعات<sup>37</sup> موتورها و آردوئینو است. در این برگه‌ها نحوه کار و ارتباط با هر یک از اجزا است که اطلاعات اولیه مورد نیاز برای انجام این پروژه است. لینک‌های مربوطه در زیر آمده است:

- <https://www.arduino.cc>
- <https://www.flickr.com/photos/28521811@N04/8520970405/in/album-72157632703854644/>
- [http://www.ee.ic.ac.uk/pcheung/teaching/DE1\\_EE/stores/sg90\\_datasheet.pdf](http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf)
- [https://www.mouser.com/Search/Refine?Keyword=130634798&FS=True&Ntk=P\\_MarCom&Tb.datasheets](https://www.mouser.com/Search/Refine?Keyword=130634798&FS=True&Ntk=P_MarCom&Tb.datasheets)

از دیگر مراجع و مقالات استفاده شده می‌توان به لینک‌های زیر اشاره کرد:

- <https://www.labcenter.com>
- <https://www.intorobotics.com/tutorial-how-to-control-the-tower-pro-sg90-servo-with-arduino-uno>
- [https://www.researchgate.net/publication/269690255\\_Design\\_Analysis\\_and\\_Implementation\\_of\\_a\\_Robotic\\_Arm-The\\_Animator](https://www.researchgate.net/publication/269690255_Design_Analysis_and_Implementation_of_a_Robotic_Arm-The_Animator)
- [https://www.researchgate.net/publication/283546707\\_Natural\\_Frequencies\\_of\\_the\\_Human\\_Hand-Arm\\_System\\_using\\_Finite\\_Element\\_Method\\_and\\_Experimental\\_Modal\\_Analysis](https://www.researchgate.net/publication/283546707_Natural_Frequencies_of_the_Human_Hand-Arm_System_using_Finite_Element_Method_and_Experimental_Modal_Analysis)

<sup>37</sup> datasheet