دانشگاه صنعتی اصفهان

دانشکده مهندسی صنایع و سیستمها

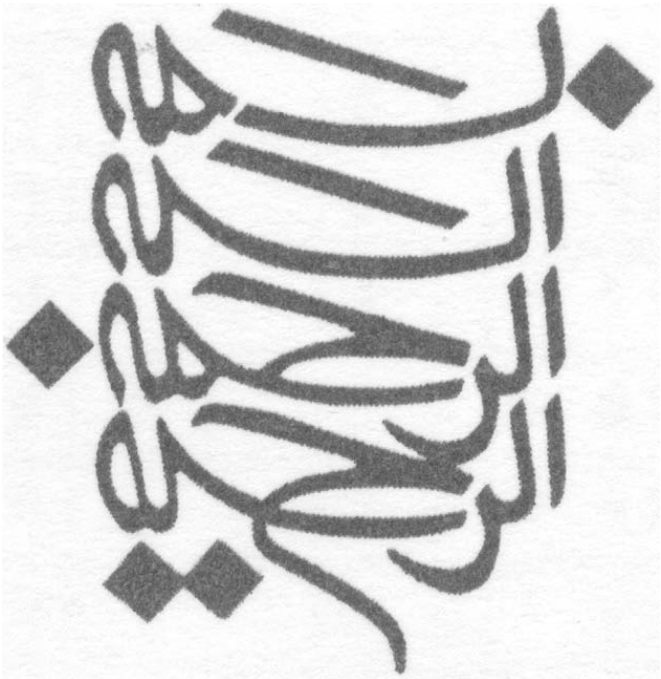
طراحی و ساخت یک بازوی روبات بردار گذار برنامه پذیر با دو درجه آزادی وانگشت قابل کنترل

پایان نامه کارشناسی ارشد مهندسی صنایع

صالح مصدق

استاد راهنما
دکتر جمشید پرویزیان

1382



دانشگاه صنعتی اصفهان
دانشکده مهندسی صنایع و سیستمها

طراحی و ساخت یک بازوی روبات بردار-گذار برنامه‌پذیر با دو درجه آزادی و انگشت قابل‌کنترل

پایان نامه کارشناسی ارشد مهندسی صنایع

صالح مصدق

استاد راهنما
دکتر جمشید پرویزیان

1382

دانشگاه صنعتی اصفهان

دانشکده مهندسی صنایع و سیستمها

پایان نامه‌کارشناسی ارشد مهندسی صنایع آقای صالح مصدق
تحت عنوان

طراحی و ساخت یک بازوی روبات بردار گذار برنامه پذیر با دو درجه آزادی وانگشت قابل کنترل

در تاریخ توسط کمیته تخصصی زیرموردبررسی و تصویب نهائی قرار گرفت .

1- استاد راهنمای پایان نامه دکتر
2- استاد مشاور پایان نامه دکتر
3- استاد داور دکتر
4- استاد داور دکتر

سرپرست تحصیلات تکمیلی دانشکده دکتر

فهرست مطالب

واژه هايكليدي :
بازوي رباتيک، بازوي بردار و گذار، ميکروکنترلر، انکودر، پرت سريال.
3 DOF Pick & Put Arm, MicroControler 89c52, Encoder, Serial Port.

پيشگفتار :
اين پروژه شروع نمي‌شد، ادامه نمي‌يافت و تمام نمي‌شد مگر آنکه فردي با ويژگيهاي منحصر به خود، حامي اين پروژه باشد، خيلي ممنون آقاي دکتر پرويزيان!
همچنين از آقاي دکتررئيسي، رئيس دانشکده صنايع وسيستمها وآقاي دکترشاهنده، سرپرست تحصيلات تکميلي دانشکده، براي نظرمساعد و همراهيشان با اين پروژه تشکرمي کنم.
بي انصافي است اگر ازآقايان نادر دهقاني زاده، احسان خياط وحسن آرش نامي برده نشود، چرا که هر موقع مشکلي را با ايشان درميان گذاردم حداکثر توان خود را براي حل آن از من دريغ نکردند.
از يگانه برادرم، حميد و يکي ازبهترين دوستانم، مازيار که غير مستقيم در پيشرفت پروژه نقش داشته‌اند، ممنونم و اميدوارم شرکت طراحان پيرامون درمسير تکامل خود گامهاي هر چه بلندتري را بردارد.
و درآخراينکه: دانشگاه صنعتي اصفهان داراي امکانات کارگاهي فراواني است (هرچنداکثرا فرسوده اند) واگر شخصي داراي توانائي استفاده از اين تجهيزات باشد واز برخورد با موانع اداري و غيراداري! نهراسد، مي تواند کليه قطعات مورد نياز براي پروژهاي ساخت را در داخل دانشگاه بسازد، کاري که براي اين پروژه انجام شد.
صالح مصدق، فروردين 1382 خورشيدي

چکیده :

در این پایان‌نامه نمونه‌اي ساده از یک بازوي گذاردن و برداشتن برنامه‌پذیر (به طور مستقل یا از طریق کامپیوتر) طراحي و ساخته شده است. دقت حرکت چرخشي، 0/5 درجه و دقت حرکت عمودي و میزان باز و بسته شدن انگشت به ترتیب 1 و 0/5 میلیمتر است. محدوده حرکت بازو نقاط موجود بر روي یک استوانه به ارتفاع 12 سانتیمتر و شعاع 40 سانتیمتر است و انگشت آن از 0 تا 11 سانتیمتر باز و بسته مي‌شود. به طور کلي سخت افزار این پروژه به عنوان محیط آزمایش و نمایش قابلیت هاي نرم افزاري آن ساخته شده است. منظور از بخش‌نرم افزاري برنامه مقیم در حافظه میکروکنترلر و برنامه گرافیکي کامپیوتري است. سخت‌افزار این بازو ازبیش از70 قطعه تشکیل شده است و بجز 2 قطعه آن که توسط ماشین فرز "CNC" ساخته شده است، بقیه توسط دستگاه‌هاي تراش و فرزمعمولي ساخته شده است.

5 برد الکتریکي وظیفه کنترل حرکات بازو را برعهده دارند. بر روي یکي از این بردها 2 مبدل آنالوگ به دیجیتال به همراه مدارات مکمل قراردارد. وظیفه این برد تبدیل موقعیت جاري پتانسیومترهاي خطي ودقیق مرتبط با دو درجه آزادي بازو (حرکت عمودي وحرکت انگشت) به فرم دیجیتال و سپس ارسال آن به میکرو از طریق 2 شیفت رجیستر است. 2 برد دیگر از این 5 برد وظیفه کنترل 2 موتورDC و یک استپ موتور را برعهده دارند. چهارمین برد مربوط به صفحه کلید است که شبکه مربوط به 40 کلید وهمچنین گیت هاي مکمل آنرا دربردارد. پنجمین برد، بردي است که میکروکنترلر بر روي آن قرار دارد و علاوه بر دریافت اطلاعات رسیده از 4 برد دیگر و اطلاعات رسیده از انکودر، وظیفه تجزیه و تحلیل حرکات ربات و فرمان دادن به موتورها و استپر و همچنین ارتباط برقرارکردن با نرم‌افزار کامپیوتري را هم برعهده دارد. میکروکنترلر استفاده شده یک "89c52" است که نزدیک به 3200 خط کد اسمبلي (تقریبا 7 کیلوبایت) را در خود جاي داده است. انکودر ساخته شده براي این بازو، یک انکودر 10 بیتي چرخشي است که داراي دقت نیم درجه است.

"RobotUI" نام نرم‌افزاري است که توسط "VC++6" طراحي و نوشته شده است و تسلط بر بازو را از طریق کامپیوتر، به طور کامل فراهم مي‌کند. به دو صورت مي‌توان به بازو برنامه داد. اولین روش استفاده از صفحه کلید متصل به بازو است و روش دیگر اتصال ربات به کامپیوتر و وارد کردن برنامه حرکت از طریق نرم افزار "RobotUI" است. برنامه با استفاده از ده G کد طراحي شده براي این بازو نوشته مي‌شود.

شاید ساده‌ترین کاربرد برای یک نمونه صنعتی ساز از این ربات، جابجائی قطعه بین ماشین‌ها در یک مجموعه CIM است که درآن ماشین‌هائی که بر روی قطعه کار انجام می‌دهند، دراطراف بازو چیده می‌شوند و

ربات به گونه‌ای برنامه‌ریزی می‌شودکه قطعات مختلف را در زمان‌های مناسب بین ماشین‌ها جابجا کند تا زمان بیکاری ماشین‌ها به حداقل رسیده و قطعات نیز طبق توالی ساخت خود بر روی ماشین‌ها قرارگیرند.

فصل اول: مقدمه

1-1 مقدمه:

اگر كلمه ربات[1] را در اينترنت جستجو كنيد، صفحات بي شماري را مي يابيد كه مي‌توان مطلب آنها را بعنوان مقدمه در اينجا آورد اما هدف اصلي اين پروژه به عمل رساندن يک طرح در زمينه طراحي بازوي رباتيک بوده است ازاين رو از تفضيل بخش هاي نظري اجتناب شده، به طراحي عملي افزوده شده است.

در ابتدا هدف آن بود تا يک نوار نقاله برنامه پذير براي يک مجموعه CIM[2] ساخته شود اما از آنجا كه اين كار را ساده يافتم پروژه را پس از چند تغيير به صورت طراحي و ساخت يک بازوي رباتيک برنامه پذير با يک درجه آزادي (از يک محل برداشتن و در محل مورد نظر به زمين گذاردن) تعريف كرديم. باپيشرفت پروژه، 1 درجه آزادي ديگر مربوط به جابجايي عمودي وكنترل انگشت هم به پروژه اضافه شد. اين سير تكميلي با اضافه كردن صفحه كليد ونمايشگر[3] (LCD) ادامه يافت تا به رابط كا مپيوتري گرافيكي رسيد وآنچه كه در حال حاضر در دانشكده صنايع دانشگاه صنعتي اصفهان قرار دارد حاصل تلاش پيوسته يک دوره 14 ماهه است.

2-1 تقسيم بندي و توصيف ربات هاي صنعتي[1] :

---

[1] ROBOT

[2] Computer Integrated Manufacturing

[3] Liquid Cristal Display

گروه صنايع رباتيک آمريکا، ربات را چنين تعريف مي-کند: يک ربات يک وسيله چندکاره قابل برنامه‌ريزي است که براي جابجائي مواد، قطعات، ابزار و وسايل خاص بر روي مسيرهاي برنامه‌ريزي شده مختلف، جهت انجام کارهاي متنوع، طراحي شده است. از اين تعريف نتيجه مي شود که ربات علاوه بر اين که بايد بتواند دستوراتي که از قبل به آن داده شده است را اجرا کند بايد به گونه‌اي باشد که اين دستورات را نيز به راحتي بتوان تغيير داد.

بازوي ربات از عضوهائي تشکيل شده است که نسبت به هم توسط اتصال هاي خطي يا چرخشي داراي حرکت هستند. ترکيب اين اتصالات پيکربندي هندسي ربات را مشخص مي کند. در جدول (1-1) ، هفت نوع عمومي پيکربندي ربات و منطقه پوشش آن را مشاهده مي کنيد.

در زير به شرح هر يک از اين پيکربنديها مي پردازيم:

پيکربندي مختصاتي:

- اين پيکربندي داراي يک منطقه پوشش مستطيل است. سه محور اصلي آن داراي حرکت خطي هستند و حرکت آنها نسبت به هم ساده است و بنابراين کنترل کردن اين حرکات راحت است. اين پيکربندي مي تواند منطقه کاري بزرگي را پوشش دهد اما اين منطقه نسبت به فضاي لازم براي خود ربات بسيار کم است. رباتهاي مختصاتي عموما براي مونتاژ استفاده مي شوند، هر چند انواع بزرگتر آنها گاهي براي جابجائي پالت و بارگذاري ماشين‌ابزار استفاده مي‌شوند.

پيکربندي سيلندري:

- اين پيکربندي داراي يک منطقه پوشش سيلندري است. دو محور آن داراي حرکت خطي هستند و ربات مي‌تواند حول يک پايه بچرخد. حرکت هاي نسبي قسمت هاي مختلف بازو ساده هستند. براي حرکت بالا پائين، چرخشي و داخل- خارج تنها لازم است تا يک محور کنترل شود. ربات هاي سيلندري پايدارند و داراي منطقه پوشش خوبي نسبت به فضاي لازم براي خود ربات هستند. قابليت دسترسي به فاصله هاي دور اين رباتها، آنها را براي جابجائي پالت و بارگذاري ماشين-ابزار مناسب مي‌سازد.

پيکربندي قطبي:

منطقه پوشش ربات هائي که داراي اين پيکربنديانند بخشي از يک کره است. اين ربات ها، ربات هاي پايداري هستند اما داراي انعطاف‌پذيري ربات هاي مفصلي نيستند.

پيکربندي مفصلي با محورهاي افقي:

ربات هائي که داراي اين پيکربنديانند شبيه بازوي انسان هستند وگستره وسيعي از کارها را مي‌توانند انجام دهند. حرکت خطي در هر راستا نيازمند داشتن مختصات جابجائي هر محور است بنابراين، اين نوع ربات نيازمند برنامه هاي کنترلي پيچيده تر است. اين ربات ها براي بلند کردن بار

تا سطوح بالا، در مقاصد انبارداري مناسب هستند. چند كارگي اين ربات ها باعث مي شود تا از آنها براي كارهاي مختلفي چون پاشش رنگ، جوشكاري، چسب زدن، مونتاژ كردن وجابجائي قطعات استفاده شود.

جدول (۱-۱) هفت نوع عمومي پيكربندي ربات و منطقه پوشش آن

منطقه پوشش	پيكربندي
مكعب مستطيل	مختصاتي
سيلندري	سيلندري
كروي	قطبي
كروي	مفصلي با محورهاي افقي
سيلندري	مفصلي با محورهاي عمودي
بخشي از كره	معلق
كروي	چند مفصلي

پيكربندي مفصلي با محورهاي عمودي:
عمده حركات آن در سطح افق است و براي كارهاي سنگين مثل عمليات فورج مناسب است.
پيكربندي معلق:
داراي ساختار با اينرسي پايين است كه از اين ويژگي براي حركت هاي سريع و شتابدار استفاده مي‌شود.
پيكربندي چند مفصلي:
- ربات هائي كه داراي اين پيكربنديانند از انعطاف پذيري فوق‌العاده اي برخوردار هستند و براي كارهاي مختلفي مي‌توان از آنها استفاده كرد. به علت انعطاف‌پذيري بالا از آنها براي انجام كارهائي كه ربات هاي معمولي قادر به انجام آن نيستند مثل جوشكاري درزها در قسمت‌هاي داخل يك اتوموبيل استفاده مي‌شود.
روشهاي انتقال نيرو:
- به دو روش كلي مي‌توان نيروي لازم براي حركت ربات ها را فراهم كرد: نيروي الكتريكي[1] و نيروي سيال[2] كه نيروي سيال خود به دو دسته نيروي هيدروليكي و نيروي بادي تقسيم مي شود.
امروزه رايج‌ترين روش اعمال نيرو استفاده از نيروي الكتريكي است كه توسط انواع مختلف موتورها چون موتورهاي سروو dc ، پله‌اي و سروو القائي اعمال مي شود. اين روش سريع و تميز است. ربات هاي هيدروليك تنها در كارهاي سنگين استفاده مي شوند. اين ربات ها نسبت به اندازه خود داراي قدرت بيشتري در برابر ربات هاي الكتريكي هستند اما كنترل آنها مشكلتر و هزينه آنها نيز بالاتر است.
ربات هاي بادي، سريع و نسبتا ارزان هستند اما كنترل آنها مشكلتر است. اين ربات ها براي مونتاژهاي سبك و عمليات بسته‌بندي مناسب هستند.
سيستم كنترل سرو[3]:
- ربات ها را به دو دسته سرو كنترل و غير سرو كنترل نيز مي‌توان تقسيم‌بندي كرد. براي آن كه به مزاياي كنترل ميكروپراسوري، دقت خوب در شرايط باري سنگين و اجراي عمليات پيچيده دست يابيم، يك كنترل سرو كامل، لازم است. در اين روش، ميزان جابجائي و شتاب با علائم فرمان مقايسه مي‌شود. تفاوت بين فرمان و عمل انجام شده، ميزان خطا است كه به عنوان اطلاعات برگشتي به ميكروكنترلر فرستاده مي شودتا دستور بعدي مطابق آن اصلاح شود. اكثر رباتهاي هيدروليكي و الكتريكي، سرو كنترل هستند. ربات هاي بادي معمولا غير سرو كنترل هستند.

---

[1] Electric Power
[2] Fluid Power
[3] Servo

کاربردها:

رایج ترین کاربرد ربات هاي سرو کنترل، جوشکاري نقطه‌اي در صنایع اتوموبیل‌سازي است. معمولا براي آن که جوش به راحتي ایجاد شود نیاز به یک ربات با 6 درجه آزادي است. همچنین نیاز به یک برنامه‌ریزي پیشرفته است تا همزمان با حرکت اتومبیل ، جوشها در محل مناسب خود ایجاد شوند.

جوشکاري قوس الکتریکي نیاز به مهارت بالاي نیروي انساني دارد علاوه برآن حرارت، دود و تشعشعات ایجاد شده باعث ناراحتي کارگر مي‌شود. بنابراین یافتن کارگري که هم داراي مهارت بالا باشد وهم حاضر به کارکردن در این شرایط باشد، مشکل است این جاست که ربات ها به بهترین شکل قابلیت هاي خود را نشان مي‌دهند. شرایط رنگ‌آمیزي هم مشابه جوشکاري قوس الکتریکي است و ربات ها به راحتي جایگزین انسان مي‌شوند.

- مورد دیگر کاربرد ربات ها، براي بارگذاري ماشین ابزار است که اساسا براي سیستم هاي تولید انعطاف پذیر(FMS [1]) و مجموعه هاي تولید یکپارچه توسط کامپیوتر (CIM) [2] کاربرد دارد. از موارد دیگر کاربرد ربات ها، مي‌توان بازرسي و تست کردن محصول، پلیسه‌گیري قطعات، چسبکاري و برش توسط آب را نام برد.

رباتي که براي این پایان نامه، طراحي شده‌است از نوع استوانه‌اي است که درجه آزادي افقي آن پیاده‌سازي نشده است.

3-1 تاریخچه کارهاي انجام شده:

با جستجو در میان پایان نامه هاي ارائه شده در دانشگاه هاي ایران، نمونه‌اي که سعي در طراحي و ساخت یک ربات صنعتي، به طور کامل داشته باشد، یافت نشد. پایان نامه هاي مرتبط به رباتیک عموما تحلیل سینماتیکي و دینامیکي یک ربا ت بوده و در مواردي هم که صحبت از ساخت به میان آمده ، یا بخشي از ربات منظور بوده است مثل پایان نامه " طراحي وساخت کنترل مفاصل ربات صنعتی در ساختار کنترل موازی مفاصل"[2] ویا بر نوع دیگری از ربات مثل ربات های متحرک تمرکز شده است مثل پایان نامه "طراحی وساخت یک ربات متحرک"[3]. در پایان نامه اول، هدف ایجاد بستری مناسب جهت آزمودن الگوریتم‌های مختلف ارائه شده برای کنترل ربات بوده است و برای اینکه بتوان هم روشهای متمرکز و هم روشهای غیرمتمرکز کنترل ربات را پیاده کرد، یک ساختار جدید مرکب ارائه شده است. در پایان نامه دوم ربات ساخته شده یک پایه متحرک است که

---

[1] Flexible Manufacturing System

[2] Computer Integrated Manufacturing

یک بازوی مکانیکی را نیز حمل می کند. حرکت ربات با کنترل جداگانه دو موتور محرک در دو طرف ربات صورت می گیرد.

فصل دوم: معرفی ربات

1-2 مشخصات بازو:

این بازو یک بازوي گذاردن و برداشتن است که محدوده حرکت آن نقاط موجود بر روي یک استوانه به ارتفاع 12 سانتیمتر و شعاع 40 سانتیمتر است و انگشت آن از 0 تا 11 سانتیمتر باز و بسته مي‌شود. دقت حرکت چرخشي آن نیم درجه بدون محدودیت جهت چرخش است. دقت حرکت عمودي آن و همچنین دقت انگشت آن به ترتیب 1 و 0/5 میلیمتر است.کنترل بازو هم از طریق کامپیوتر و هم به صورت مستقل امکانپذیر است. براي کنترل از طریق کامپیوتر برنامه اي که براي همین منظور نوشته شده اجرا شده و در محیط آن مي توان بازو را کنترل کرد. براي توضیحات بیشتر به فصل هفتم(برنامه نویسي رابط گرافیکي) وضمیمه الف مراجعه شود.

همچنین این بازو را مي توان به طور مستقل واز طریق مینی کامپیوتر موجود در پایه بازو کنترل کرد. منظور از مینی کامپیوتر یک میکروکنترلر، یک نمایشگر ویک صفحه کلید است که به صورت فشرده‌اي در پایه بازو قرار گرفته اند. براي توضیحات بیشتر به فصل هاي پنجم و ششم وضمیمه الف مراجعه شود.

2-2 روابط و ساختار اجزا بازو:

حرکت دوراني بازو توسط یک استپ موتور که توسط یک سیستم چرخدنده حلزوني به محور اصلي بازو وصل است ایجاد مي شود. جهت وسرعت چرخش در هر لحظه توسط قطار پالسي که

بوسيله ميكروكنترلر ساخته شده و با كمك برد استپر تقويت مي شودكنترل مي گردد.

از ويژگي گام استپر براي رفتن به نقطه خاصي استفاده نمي‌شود بلكه اين كار توسط بررسي اطلاعات رسيده از انكودرصورت مي گيرد. به اين صورت كه ابتدا توسط توابع مخصوص جهت حركت (ساعتگرد يا پاد ساعتگرد، بسته به اين كه كدام طرف نزديكتر است) مشخص شده وفاصله بين مبداومقصد به سه قسمت تقسيم مي شود. سپس فرمانهاي مناسب به استپرداده مي شود تا بازو قسمت اول مسير را با شتاب افزاينده طي كند تا به انتهاي آن برسد، سپس قسمت دوم را با سرعت ثابت طي كند و با رسيدن به قسمت سوم با شتاب كاهنده حركت كند تا به نقطه مطلوب برسد ومتوقف شود. چنانچه فاصله بين مبدا و مقصد كوچكتر از 10 درجه باشد اين فاصله را با سرعت كم و ثابتي طي مي كند.

در ابتدا تصميم بر اين بود تا از سيستم انكودر براي كنترل حركت هاي عمودي و حركت انگشت هم استفاده شود، اما از آنجا كه انكودر ساخته شده بزرگ بود وهمچنين براي آنكه روش ديگري نيز تجربه شود، ايده استفاده از مبدل آنالوگ به ديجيتال (ADC)[1] واستفاده از تغييرات يك پتانسيومتر براي حركت بازو مورد استفاده قرارگرفت. در اين روش خروجي حاصل از دو مبدل آنالوگ به ديجيتال توسط 2 رجيسترموازي به سريال به داخل ميكروكنترلر منتقل مي شود.

در هر بار خواندن موقعيت پتانسيومترها، 7 نمونه از هر پتانسيومتر پشت سرهم خوانده مي شود و سپس ميانگين اين 7 نمونه به عنوان موقعيت جاري پتانسيومتر مورد استفاده قرار مي‌گيرد.

مبدل هاي آنالوگ به ديجيتال 8 بيتي هستند و بنابر اين 256 موقعيت را تفكيك مي‌كنند. براي انگشت ازتمام 256 حالت استفاده مي‌شودو چون رنج حركت انگشت در ازاي يك دورچرخش پتانسيومتر 11سانتيمتر است، دقت انگشت 0/5mm است.

براي حركت عمودي بازو موقعيت پتانسيومتر بر2 تقسيم مي شود در نتيجه 128 حالت وجود دارد كه معادل با 12 سانتيمتر جابجائي عمودي است بنابر اين دقت اين جابجائي يك ميليمتر است.

توضيح ضروري:

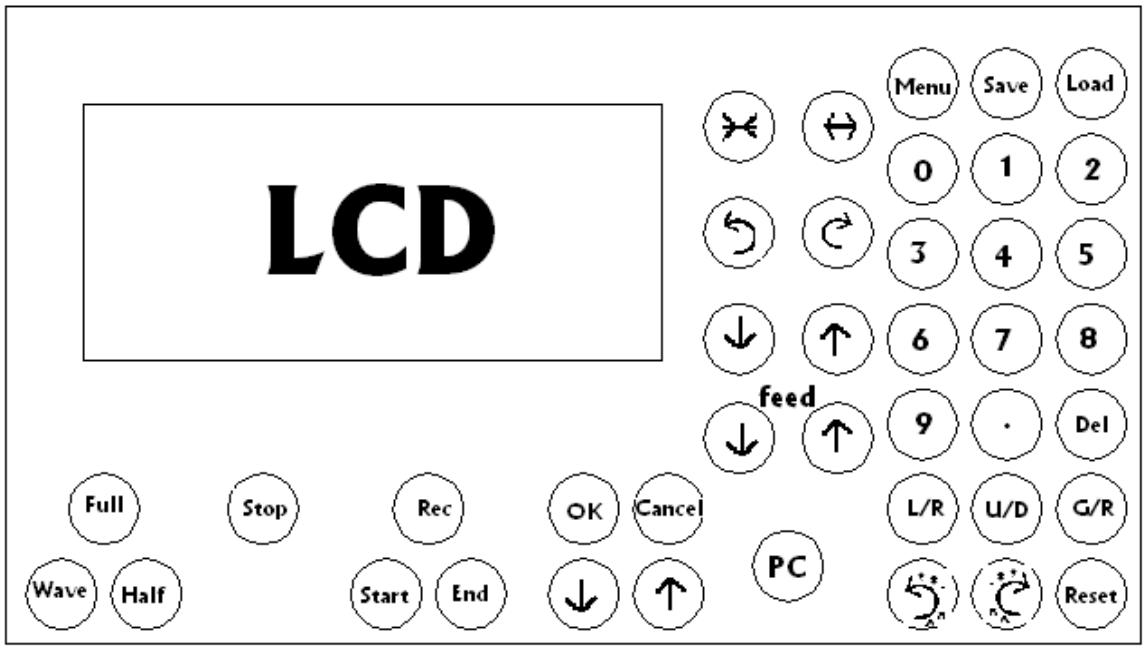
در هنگام تشريح كدهاي برنامه ها در مواردي به علت تفصيلي بودن برنامه شرح آن به درازا مي‌كشيد كه از حوصله اين متن خارج بود بنابر اين تنها به ذكر سر نخ

[1] Analog to Digital Converter

هائي از عملكرد برنامه ها اكتفا شده است. در ادامه براي آشنائي با عملكرد بازو، صفحه كليد متصل به بازو تشريح مي‌شود.

3-2 عملكردكليدها :

صفحه كليد بازو در شكل 2-1 نشان داده شده است. در زير به شرح عملكرد كليدهاي آن پرداخته مي‌شود. در نمونه اصلي متن يا تصويري متناسب با عملكرد كليد بر روي آن رسم شده است.



شكل (2-1) صفحه كليد متصل به بازو

6كليدواقع در سمت راست نمايشگر:

اين كليدها براي حركت دادن بازو به صورت دستي استفاده مي شوند.

كليدهاي Menu, Save, Load:

با زدن كليد Menu، منوئي ظاهر مي شود كه از طريق آن مي توان برنامه اي را نوشت يا ويرايش كرد (گزينه ويرايش در حال حاضر فعال نيست). كليدهاي Load وSave هم براي ذخيره يا اجراي برنامه به كار ميروند.

كليدهاي اعداد:

براي وارد كردن اعداد به كار مي رونـد.

2كليدقرارگرفته درسمت چپ كليد Reset :

اين دو كليد براي وارد كردن حروف و اعداد به كار ميروند.

كليدDel :

براي پاك كردن استفاده ميشود.

كليدهاي L/R,U/D,G/R :

براي وارد كردن سه Gكد به كارمي‌روند. اين كدها عبارتند از G براي انگشت، U براي حركت عمودي وL براي حركت چرخشي.

2كليدقرار گرفته درزير كلمه Feed :

براي كم و زياد كردن ضريب سرعت(Feed) هستند.

كليدهاي Full,Half,Wave :

مد استپرراتغيير مي دهند.

كليدهاي Rec,Start,End :

توسط اين كليدها مي توان حركت هاي دستي بازو را ثبت كرد. با زدن دكمه "Start"نام برنامه پرسيده ميشود. پس از وارد كردن نامي براي برنامه، بازو وارد مد ثبت مي شود. در اين مد ميتوان بازو را به صورت دستي حركت داد و موقعيت هاي دلخواه را توسط فشار دكمه"Rec "ثبت كرد. براي خاتمه برنامه بايد دكمه" End "را فشار داد. چنانچه حافظه برنامه پر شود، ربات اين موضوع را اطلاع داده و پس از ذخيره برنامه از اين مد خارج ميشود.

كليد"Stop" :

در حال حاضر تنها براي خروج از حالت" Run "،بدون اجراي برنامه اي، استفاده مي شود.

كليد" Reset" :

با فشار دادن اين كليد ميكرو Reset مي‌شود.

كليد" PC " :

براي اتصال به كامپيوتر استفاده مي‌شود. پس از فشاردادن اين كليد و انتخاب نرخ تبادل اطلاعات مناسب، بازو آماده فرمان گرفتن از[1] PC مي شود.

نكته مهم:

با قطع برق برنامه‌ها از بين مي روند بنابراين براي حفظ آنها بايد به PC وصل شده و برنامه ها را به روي ديسك منتقل كرد.

در پيوست 1 اطلاعات مربوط به نحوه استفاده ازاين كليدها براي ورود واجراي يك برنامه آورده شده است.

[1] Personal Computer

فصل سوم:
انکودر[1]

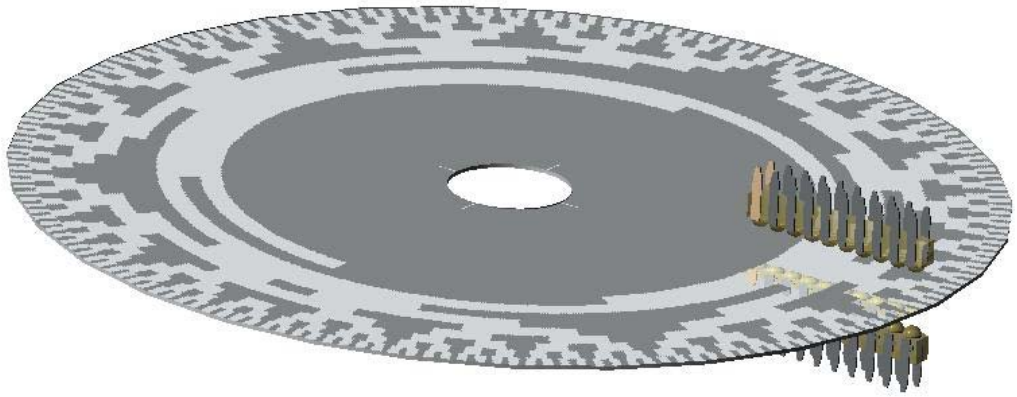
قبل از تشريح قسمتهاي مكانيكي و الكتريكي، بايد انكودراستفاده شده در اين بازوكه يک قطعه مكانيكي-الكتريكي است، توضيح داده شود[4]. انكودرها بردونوع هستند: انكودرهاي مطلق وانكودرهاي نسبي. از آنجا كه انكودر ساخته شده براي اين ربات ازنوع مطلق است دراينجا اين نوع انكودر به طور كامل شرح داده ميشود:
اين نوع انكودر از3 قسمت اصلي تشكيل شده است. يک صفحه كد، يک قسمت كدخوا ن و يک قسمت تحليل كد. خروجي حاصل از قسمت كدخوان معرف موقعيت فعلي شفت انكودر است كه در قسمت تحليل كد به فرم باينري تبديل مي شود.

صفحه كد :

صفحه كد يک صفحه شفاف به قطر 12سانتيمتر است كه به 720 قطاع تقسيم شده است و هر قطاع معرف يک موقعيت يا كد است. هر قطاع خود به 10 قسمت كه هر قسمت معرف 1 بيت است تقسيم مي‌شود. با 10 بيت مي توان به 1024 حالت دست يافت (از 0 تا 1023). براي خواندن هر موقعيت 10 لِد[2] فرستنده مادون قرمز به صورت شعاعي در يک طرف صفحه و در مقابل هر بيت قرار داده شده ودر طرف ديگر صفحه 10 فتوترانزيستور مادون قرمز متناظربا اين 10 لِد فرستنده قرار ميگيرد (شكل3-1).

---

[1] Encoder
[2] LED



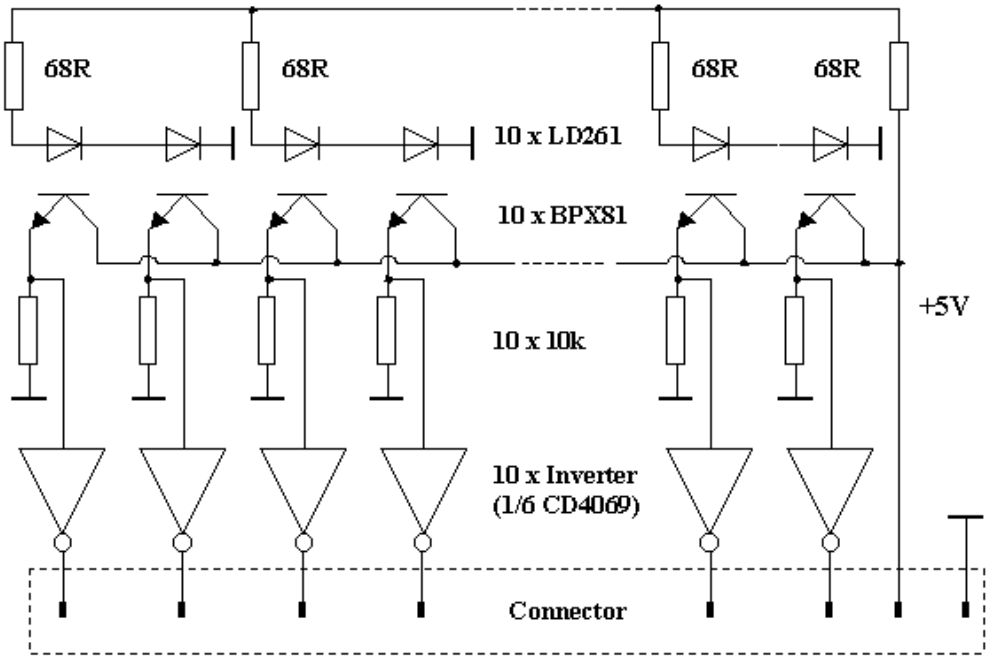
شکل (3- 1) مجموعه صفحه کد ولدهای فرستنده و گیرنده

اگر مسیر بین یک لد فرستنده وگیرنده متناظر آن قطع شود پس از اینورتر بیت صفر ودر غیر این صورت بیت 1 را داریم . اگر کد هاي 0 تا 1023 را به صورت باینري بر روي صفحه کد قرار دهیم ازآنجا که هر کد با کدهاي کناري خود در چند بیت متفاوت است، باعث بالا رفتن خطا مي‌شود. براي پرهیز از این مشکل از کد دیگري که به کد خاکستري (Gray یا reflected)معروف است، استفاده مي‌کنیم وسپس در قسمت تحلیل کد آنرا مجددا به فرم باینري برمي‌گردانیم. کدخاکستري داراي این ویژگي است که هرکد باکدهاي کناري خود تنها دریک بیت تفاوت دارد. براي تبدیل کدباینري به کد خاکستري، کد باینري را یک بیت به سمت راست شیفت مي دهیم ونتیجه را با کد اصلي جمع مي‌کنیم، بدون آنکه به بیت کَري توجه کنیم(اگر بیت کري بوجود آمد ازآن صرفنظر مي کنیم). سپس اولین بیت سمت راست نتیجه را حذف مي کنیم. آنچه باقي مي ماند، کد خاکستري مورد نظر است. براي مثال عدد11 را در نظر مي‌گیریم که کد باینري آن عبارت است از 1011، داریم:

binary: 1 0 1 1
1 0 1 1
Gray: 1 1 1 0 1

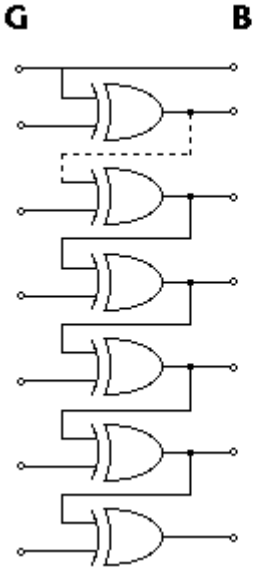
بنابر این کد خاکستري مربوط به عدد 11عبارتست از 1110. اگر کد خاکستري عدد 12 را بدست آوریم متوجه مي شویم که در تنها یک بیت با کد خاکستري عدد 11 تفاوت دارد.

در شکل زیرشماتیک قسمت الکتریکی هد نمایش داده شده است:



شكل (2-3) شماتيک قسمت الکتريکی هد

براي تبديل کد خاکستري به کد باينري معادل آن مي توان از روش سخت‌افزاري يا نرم‌افزاري استفاده کرد. در روش سخت‌افزاري از يک مجموعه گِيت XOR استفاده مي‌شود که شماتيک آن براي يک عدد 7 بيتي در زير ترسيم شده است.

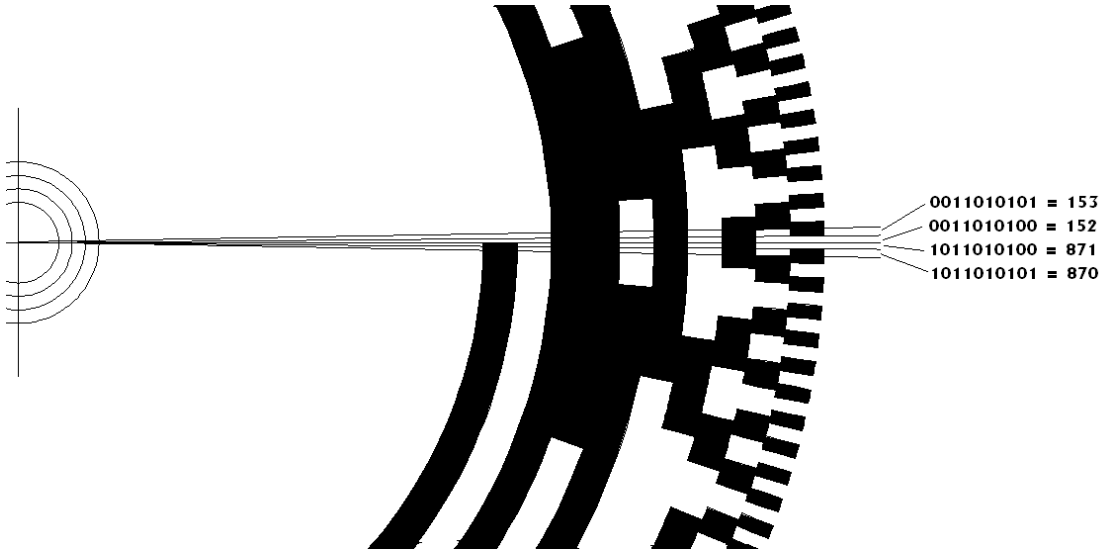


در روش نرم‌افزاري مي توان با چند دستور خواندن ومقايسه‌کردن با مقدار قبلي، به کد باينري دست يافت. در

زيريک الگوريتم براي تبديل يک عدد 10 بيتي خاکستري به معادل باينري‌آن آورده مي‌شود:

BEGIN: set B0 through B11 = 1
B11 = G11
IF B11 = G10 THEN B10 = 0
IF B10 = G9 THEN B9 = 0
IF B9 = G8 THEN B8 = 0
IF B8 = G7 THEN B7 = 0
IF B7 = G6 THEN B6 = 0
IF B6 = G5 THEN B5 = 0
IF B5 = G4 THEN B4 = 0
IF B4 = G3 THEN B3 = 0
IF B3 = G2 THEN B2 = 0
IF B2 = G1 THEN B1 = 0
IF B1 = G0 THEN B0 = 0
DONE

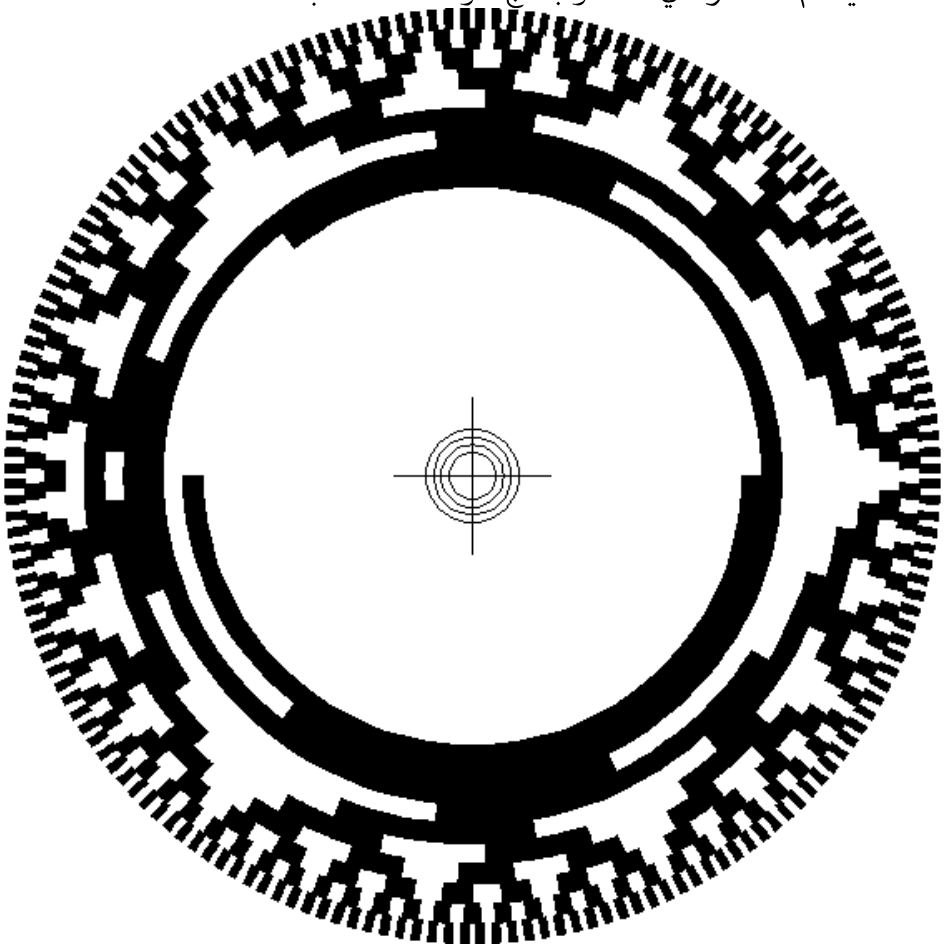
هرچند انکودر ساخته شده براي اين بازو 10بيتي است اما از آنجا که اگر صفحه کد به 1024 قطاع تقسيم شود به علت باريک شدن بيش از حد قطاع ها (بخصوص در نواحي نزديک به مرکز) امکان خواندن کد به صورت صحيح براي مجموعه فرستنده و گيرنده ميسر نمي شود، بنابراين تعداد کدها به 720 کد کاهش داده شد. کد هاي خاکستري داراي اين ويژگي هستندکه اولين وآخرين آنها نيز تنها دريک بيت با هم تفاوت دارند و چنانچه به تعداد مساوي از دو طرف اين مجموعه کدها را حذف کنيم، باز هم اين ويژگي پايداراست. ازاين ويژگي استفاده کرده و152کد از ابتدا و به همين تعداد از انتهاي کدهاحذف شد. در شکل 3-3 محل اتصال دو انتهاي کد به هم نشان داده شده است.



شکل (3-3) محل اتصال دو انتهاي کد

شکل 3-4 صفحه کدنهايي رانمايش مي‌دهد. تعداد قطاع ها دراين صفحه 720 قطاع ومعادل عددهاي 152 تا 871 به فرم خاکستري هستند. براي تهيه اين صفحه‌کد ازروش زيراستفاده شده است:

ابتدا صفحه کد در نرم‌افزار اتوکد ترسيم شده و سپس براي تهيه فيلم ليتوگرافي از آن اين فايل از طريق نرم افزار واسط Illustrator وارد نرم افزار Freehand شده و خروجي‌آن براي چاپ به ليتوگرافي تحويل شده است. اين روش باعث مي شود تا فيلم داراي اعوجاج و خطا نباشد.

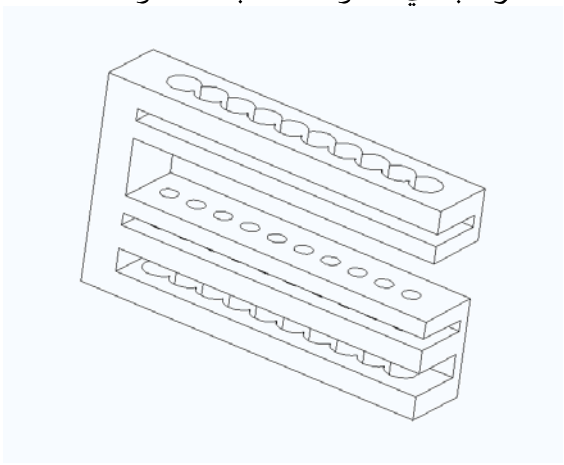


شکل (3-4) صفحه کدنهايي

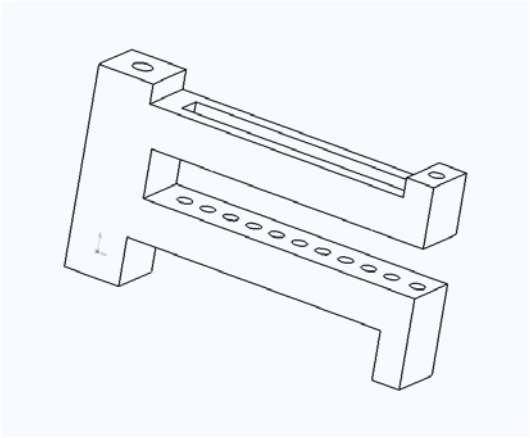
براي ساخت هدکدخوان طرحهاي مختلفي ترسيم شدکه نمونه آنهارا درشکل 3-5 مي بينيد. طرح اول داراي پيچيدگي بسيار بوده وطرح دوم هم به علت آنکه بخشي از کار توسط دستگاه فرز انجام مي‌شد، داراي دقت کافي نبود اما طرح سوم به صورت کاملا بهينه و ساده و به طورکا مل توسط ماشين فرز CNC[1] ساخته ميشود. هما نطورکه ديده ميشود اين هد از شش قطعه تشکيل شده است که اين قطعات از فيبر دو طرف مس

[1] Computer Numerical Control

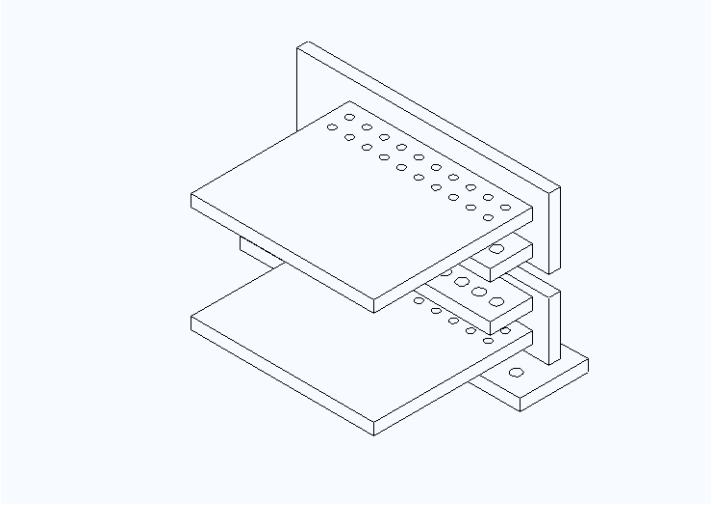
فايبرگلاس ساخته شده اندو پس از مونتاژ هد مي توان آنها را به هم لحيم كرد تا در جاي خود ثابت شوند.



الف



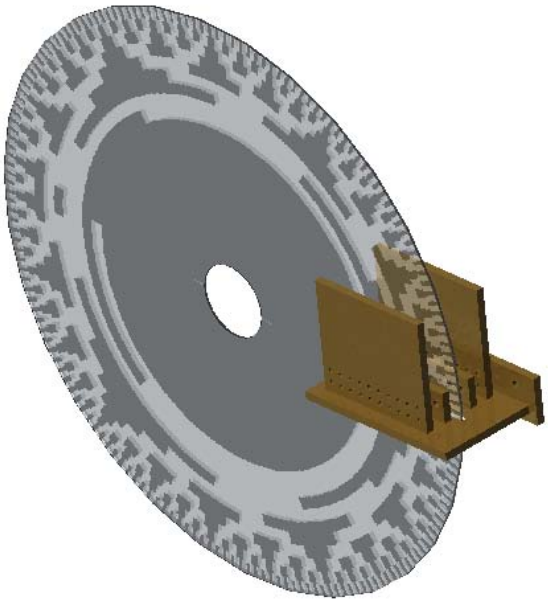
ب



پ

شكل (3-5) طرحهاي مختلف هدكدخوان

در فصل‌سوم، تصويرهايي از شبيه سازي صورت گرفته در MasterCAMمربوط به اين شش قطعه آورده شده است. پس از پايان اجراي اين برنامه، قطعات لازم براي مونتاژ 8 هد بدست مي آيد. شکل 3-6 تصويري ازنمونه ساخته شده از هد و صفحه کد را نشان ميدهند.



شکل (3-6) مجموعه هد و صفحه کد

فصل چهارم: مکانیک بازو

در صفحات بعدي نقشه ساخت و نماي سه بعدي از 78 قطعه استفاده شده در ساخت اين بازو را مشاهده مي‌کنيد. از اين تعداد 8 عدد ،قطعاتي چون بالبرينگ، موتور و مانندآنها هستند و بقيه از صفر ساخته شده‌اند. در جدول 1-4 ليست قطعات و عملکرد آنها آورده شده است.

ذکر چند نکته ضروري است:

- هيچگونه تحليل مکانيکي براي طراحي اين قطعات انجام نشده است. نبايد فراموش‌کردکه اين بازوي رباتيک يک بازوي نيمه آزمايشگاهي است، بنابراين درهنگام طراحي سعي شده است تا با امکانات موجود حداکثر توانايي بدست آيد. در هنگام طراحي يک مدل صنعتي از اين نوع تجهيزات معمولا يک سري اهداف از پيش تعيين شده، و بر اساس آنها طراحي صورت مي گيرد مثل حداقل باري که ميتوان جابجا کرد، حداقل سرعت جابجايي و... .

شکل ظاهري بازو نيز از نکته ذکر شده مستثني نيست وحاصل سليقه شخصي سازنده و محدوديت هاي سر راه وي است به طور مثال در نمونه ساخته شده نمي‌توان شئ را که در ارتفاعي پائين‌تر از 15/3 سانتيمترقراردارد، گرفته وجابجا کرد واين به دليل محدوديت جابجايي ماشين فرزي استکه چرخدنده‌شانه‌اي مربوط به حرکت عمودي بازو توسط آن ساخته شده است و اين محدوديت باعث شد تا نتوان چرخدنده شانه اي را بزرگتر از 20 سانتيمتر ساخت. شايد گفته شود که چرا از ارتفاع کل ربات براي حل اين مشکل کاسته نشده است که در جواب بايد همه تقصيرات را به گردن سليقه طراح انداخت! (به هر حال تناسبات طرح بايد حفظ شود!).

به جای انتخاب نام برای قطعات، نمای سه بعدی از قطعه در گوشه نقشه آورده شده است که با مراجعه به نقشه انفجاری، می توان از محل وعملکرد آن قطعه مطلع شد.

جدول 4-1 لیست قطعات و عملکرد آنها	
شماره قطعه	توضیح
1	پوسته در برگیرنده مجموعه چرخدنده شانه‌ای- جنس قطعه: آلومینیوم
26، 27، 28	مجموعه چرخدنده شانه‌ای و بالشتک‌های آن که حرکت عمودی ربات را ایجاد می کنند-- جنس قطعات از چپ به راست: فسفر برنز، تفلون، آهن
2	چرخ شانه‌ای --جنس قطعه: فسفر برنز
3، 4	درپوش فوقانی و درپوش تحتانی مجموعه چرخدنده شانه‌ای- جنس قطعات: آلومینیوم
5، 6	درپوش فوقانی و بدنه مجموعه انگشت - جنس قطعات: آلومینیوم
7، 8، 9،10	مجموعه گیربکس برای سیستم انگشت
11، 15، 12	سیستم پیچ و مهره که باعث باز و بسته شدن انگشت می شود. --جنس قطعات از راست به چپ: فسفر برنز، فسفر برنز، آهن
13	صفحه زیرین نگهدارنده چرخدنده های گیربکس انگشت- جنس قطعه: آهن
14	موتورDC انگشت
16، 17	پین های نگهدارنده فک های انگشت- جنس قطعه: آهن
18	بدنه اصلی انگشت که موتور، گیربکس و فک ها به آن وصل می‌شوند. - جنس قطعه: آلومینیوم
19، 20	فک های انگشت- جنس قطعه: آلومینیوم
24، 29	نگهدارنده بالشتک های چرخدنده شانه‌ای- جنس قطعات: فیبر استخوانی الیافدار
31، 34، 32	این اجزا حرکت خطی چرخدنده شانه‌ای را به حرکت دورانی پتانسیومتر تبدیل می‌کنند. ---جنس قطعات از راست به چپ: آهن، فسفر برنز، فسفر برنز
33	درپوش پتانسیمتر و سیستم وابسته به آن- جنس قطعه: آلومینیوم
21، 25	دو پروفیل تشکیل‌دهنده بازوی افقی- جنس قطعه: آلومینیوم
22	نگهدارنده موتور dc - جنس قطعه: پلاستیک
23	نگهدارنده گیربکس مربوط به حرکت عمودی- جنس قطعه: پلاستیک
35، 36	دو پروفیل تشکیل‌دهنده بازوی عمودی- جنس قطعه: آلومینیوم
37، 38	درپوش های پلاستیکی دو انتهای پروفیل ها- جنس قطعات: پلاستیک
40، 41	صفحات نگهدارنده بازوی عمودی- جنس قطعات: آلومینیوم
42	محور اصلی انتقال نیروی چرخشی- جنس قطعه: CK45

ادامه جدول 4-1	
43، 45	بلبرینگ های اطراف محور اصلی
44	بالشتک بلبرینگ های اطراف محور اصلی- جنس قطعه: آلومینیوم
39	پایه ربات که از صفحه فوقانی و 4 قطعه دیواره تشکیل شده است. - جنس قطعه: MDF
54، 57	دو قطعه‌ای که صفحه انکودر را به محور اصلی وصل می-کنند. - جنس قطعات: تفلون
55	انکودر که از 6 قطعه تشکیل شده‌است و این قطعات در فصل سوم تشریح شد. - جنس قطعات: فیبر فایبرگلاس دو طرف مس
56	صفحه انکودر
58	صفحه پلاستیکی که صفحه کلید روی آن چاپ شده است.
59	صفحه نمایش
60	بدنه صفحه کلید- جنس قطعه: آلومینیوم
49	نگهدارنده موتور پله‌ای- جنس قطعه: آلومینیوم
46	پوسته نگهدارنده برد مدار استپر که به عنوان خنک کننده آی‌سی L298 نیز عمل می‌کند. - جنس قطعه: آلومینیوم
47، 48	ترانس های منبع تغذیه
51، 52	مارپیچ و چرخ حلزونی که نیروی موتور پله‌ای را به حرکت دورانی بازو تبدیل می‌کند.
50	کف - جنس قطعه: MDF [1]
53	موتور پله‌ای

--در نقشه هاي انفجاري و ساير نقشه ها ازآوردن پيچ و مهره و واشر وقطعات مشابه ديگر خودداري شده است. استپ موتور استفاده شده دراين بازو ،يک موتور روسي است که ابعاد خارجي آن در نقشه ها آورده شده است. به جاي اين استپ موتور هر موتور ديگري که داراي ابعادنزديک به اين موتور باشد، قابل جايگزيني است. اين استپ موتور 12 ولت (حداکثر 32ولت) وداراي گام 1/8 درجه است.گشتاور خروجي آن 0/84 نيوتن متروفرکانس اسمي آن 1000 هرتز است.
- ابعاد خارجي دو DC موتور 12ولت استفاده شده در اين بازو نيز در نقشه ها آورده شده است و هر موتور مشابهي را مي توان جايگزين کرد.

سعی شده است تا آنجا که ممکن است قطعات سبک باشند. همچنین طراحی به گونه‌ای صورت گرفته است که نیاز به کمترین

[1] Medium Density Fiber

ماشینکاری باشد. به همین دلایل است که از پروفیل‌های آلومینیوم در ساخت بازو سود برده شده است.
- برای بخش هائی از پایه از MDF[1] (نوعی فیبر پوشش دار) استفاده شده است،تا ماشینکاری آن راحتتر باشد.
--جنس چرخدنده ها از فسفر برنز انتخاب شده است تا علاوه بر ماشینکاری راحتتر، از ویژگی نرمی حرکت وسایش کم آن نیز بهره برده شود.
مشخصات چرخدنده شانه اي به صورت زیراست. دراین روابط M معرف مدول، Z تعداد دنده ها، d0 قطر داخلي، dk قطر خارجي، h ارتفاع دنده هاو t فاصله گامها است:
مشخصات چرخ:

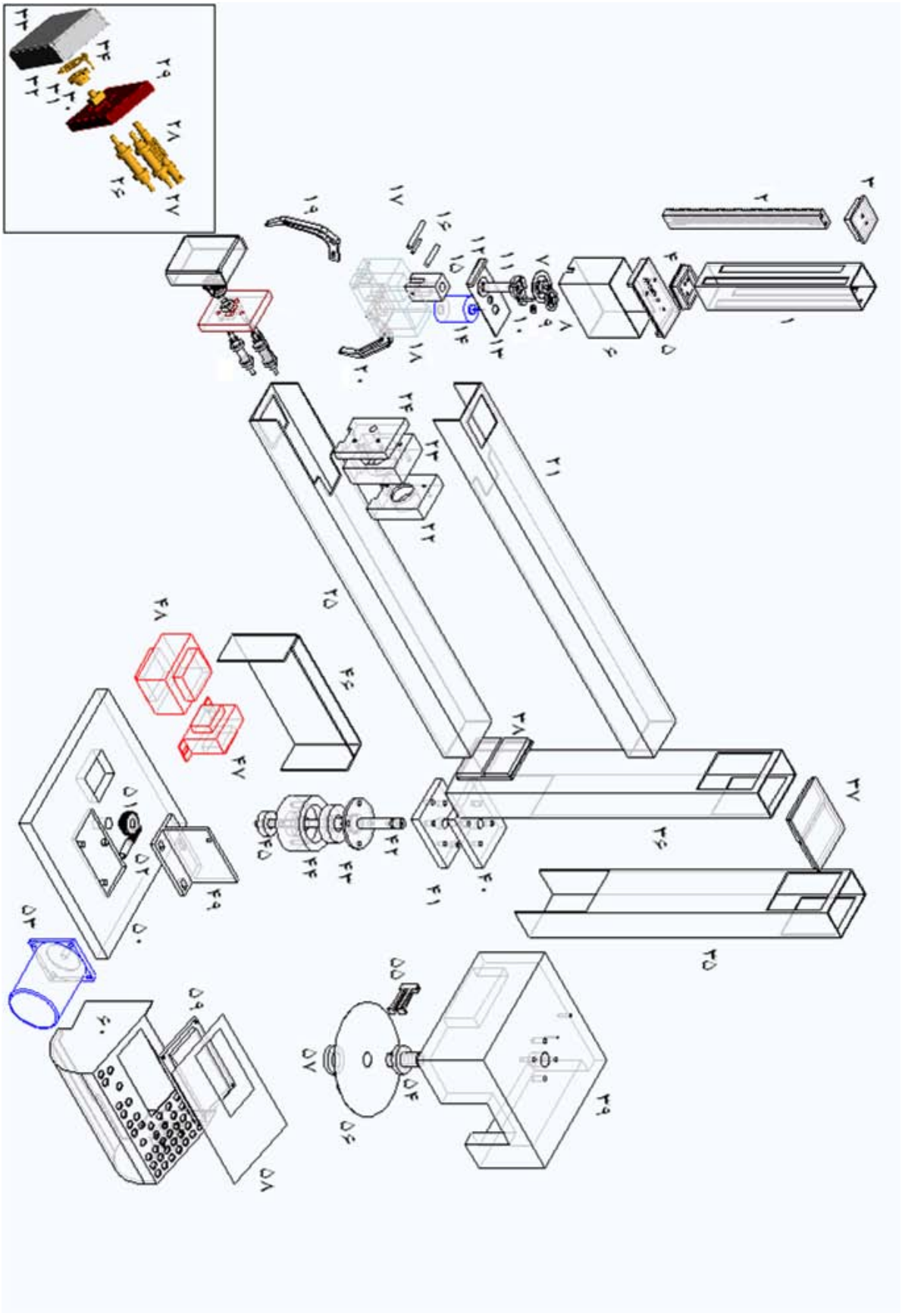
M=0.7 Z=12
d0=M*Z=0.7*12=8.4
dk=(Z+2)*M=(12+2)*0.7=9.8
h=2.16M=2.16*0.7=1.52

مشخصات شانه:

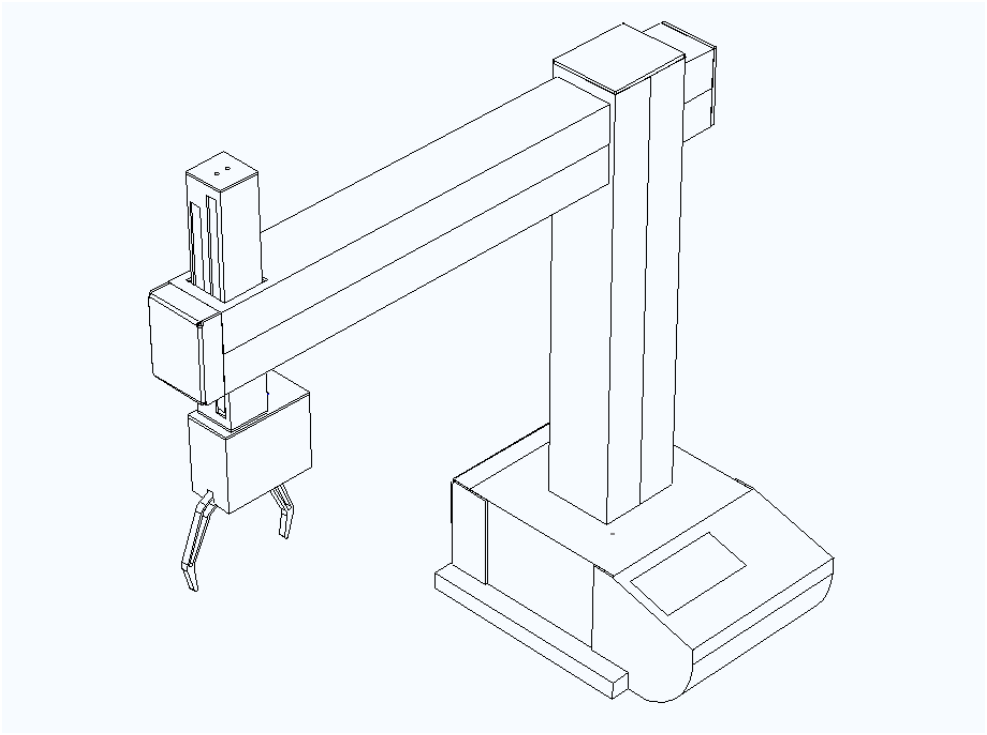
t=3.14M=3.14*0.7=2.2
h=2.16M=2.16*0.7=1.52

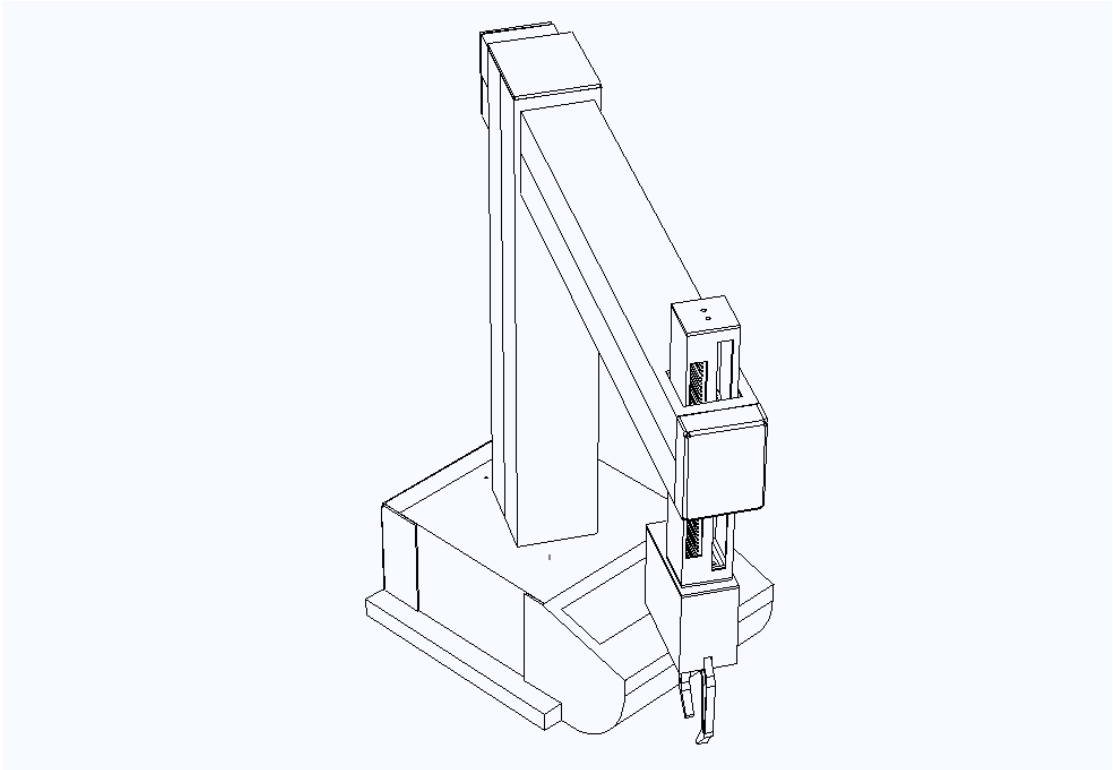
-

[1] Medium Density Fiber

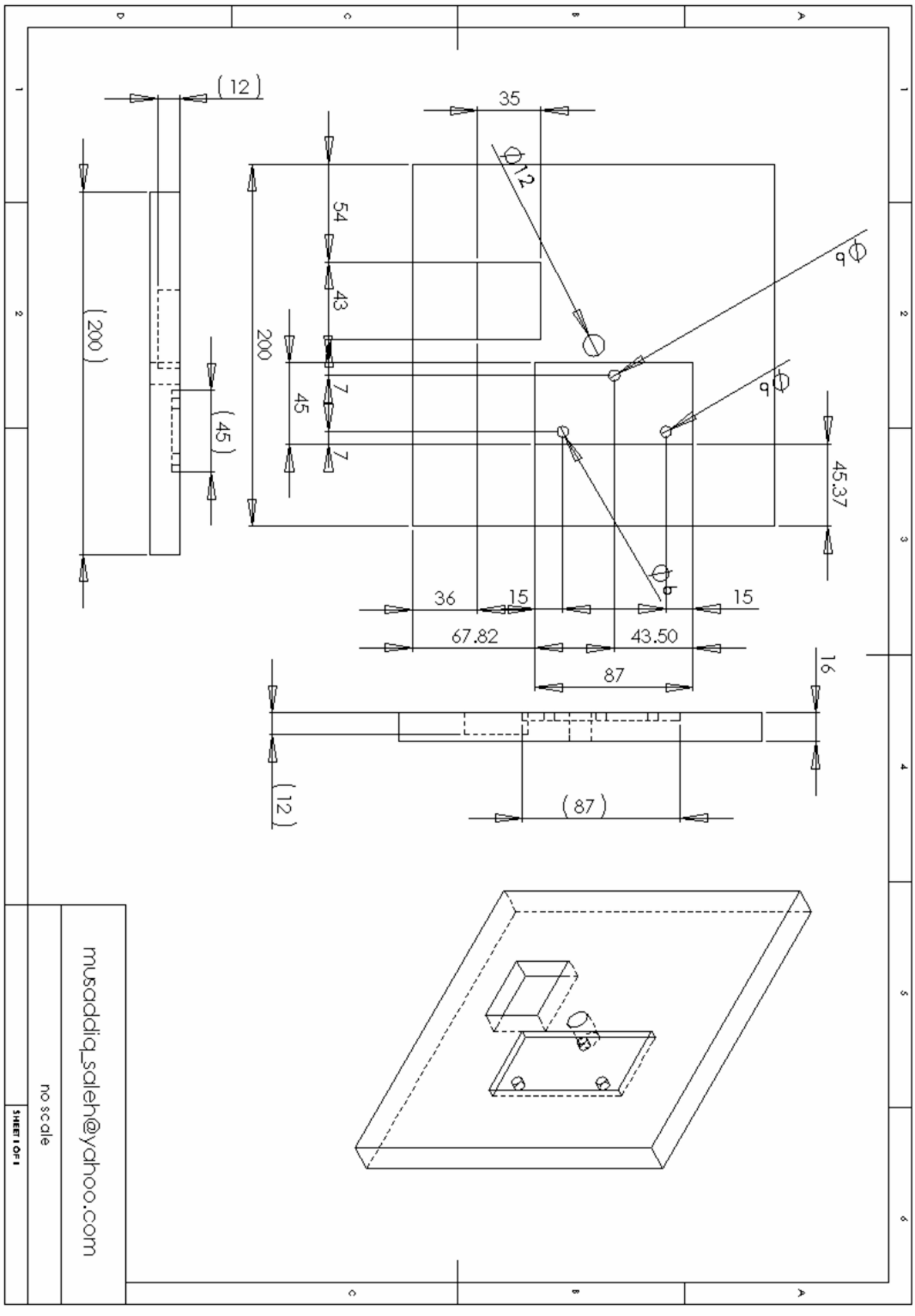


شکل(4-1) نقشه انفجاری بازو

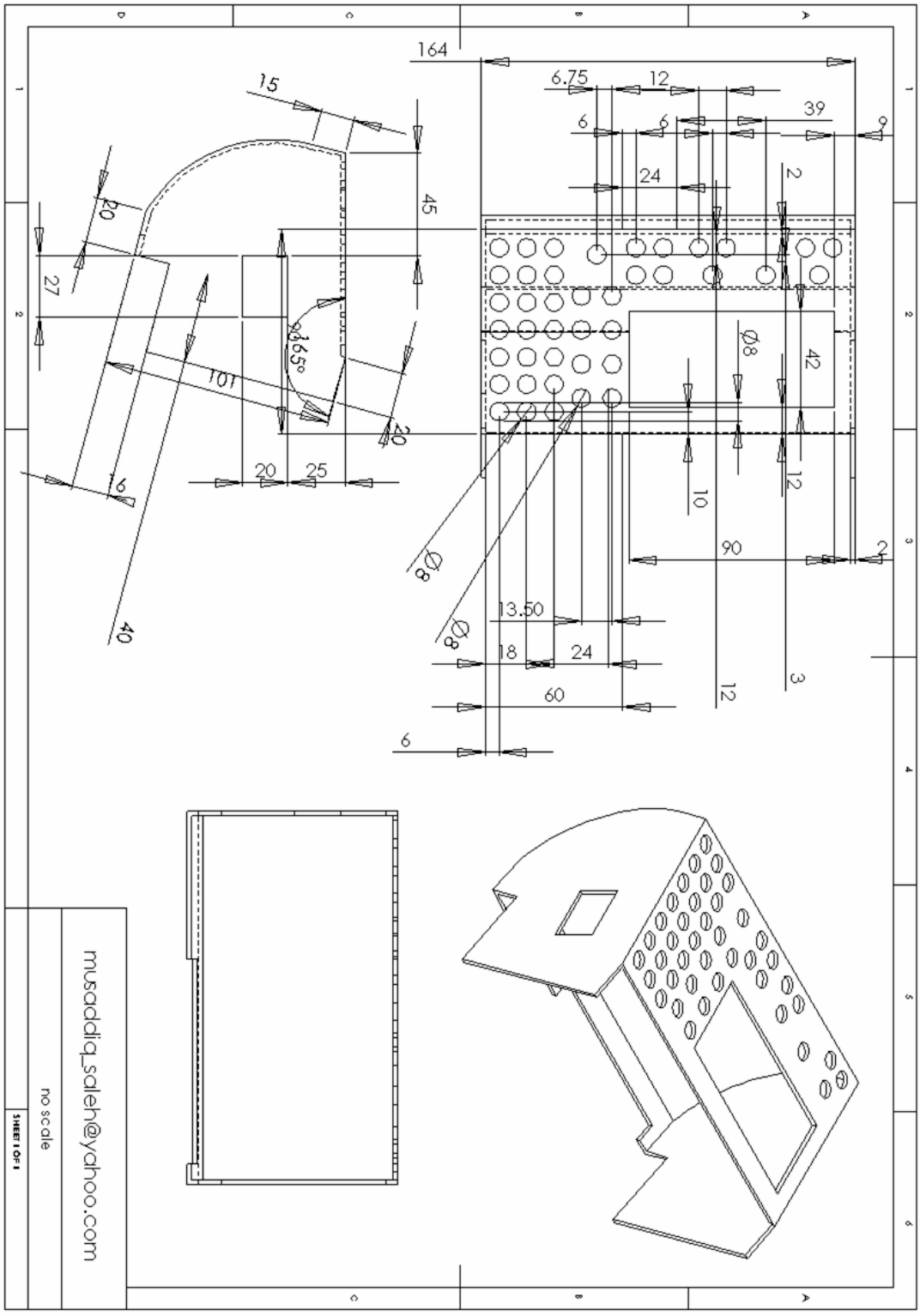




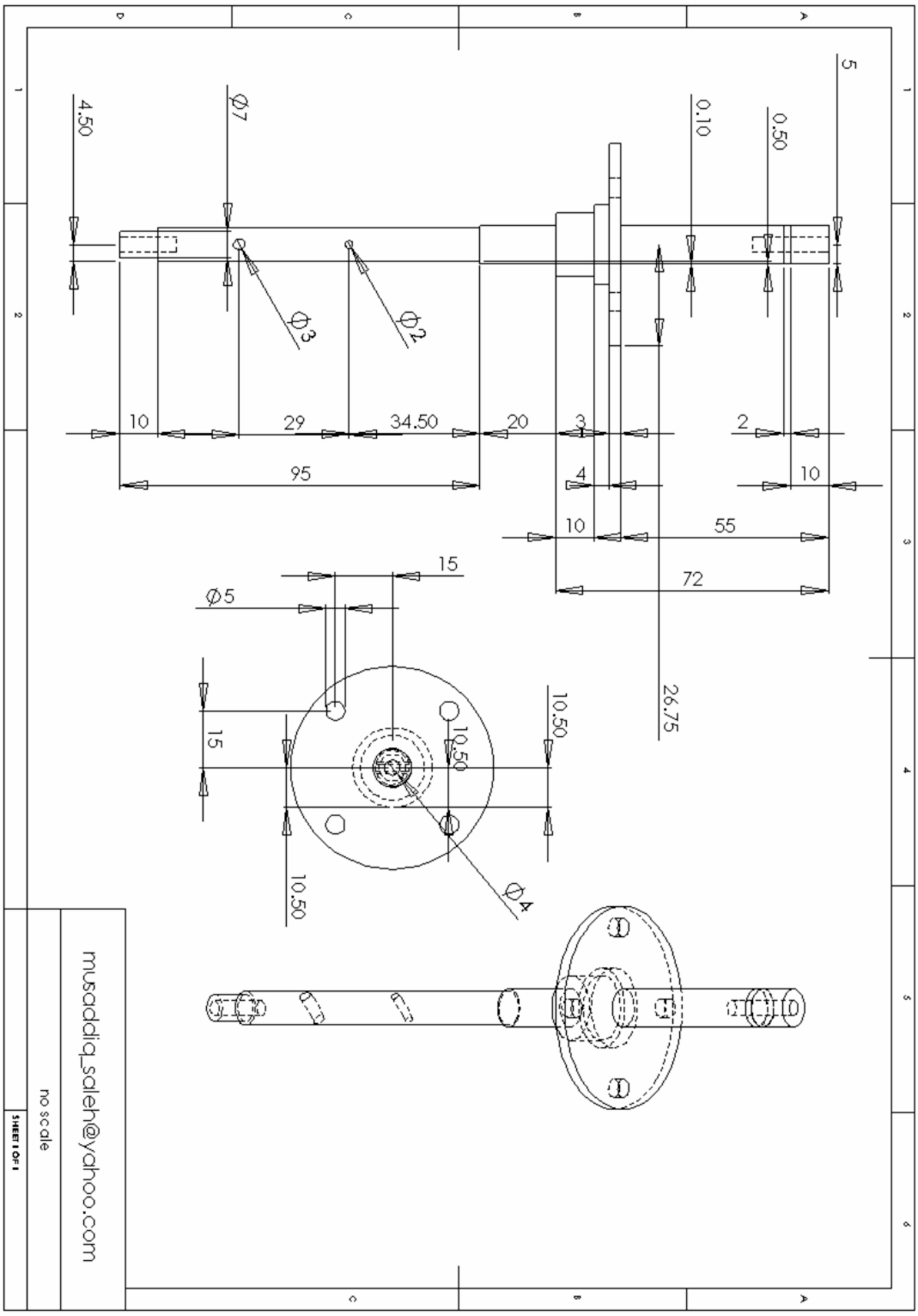
شکل(4-2) دو نمای مختلف از بازو



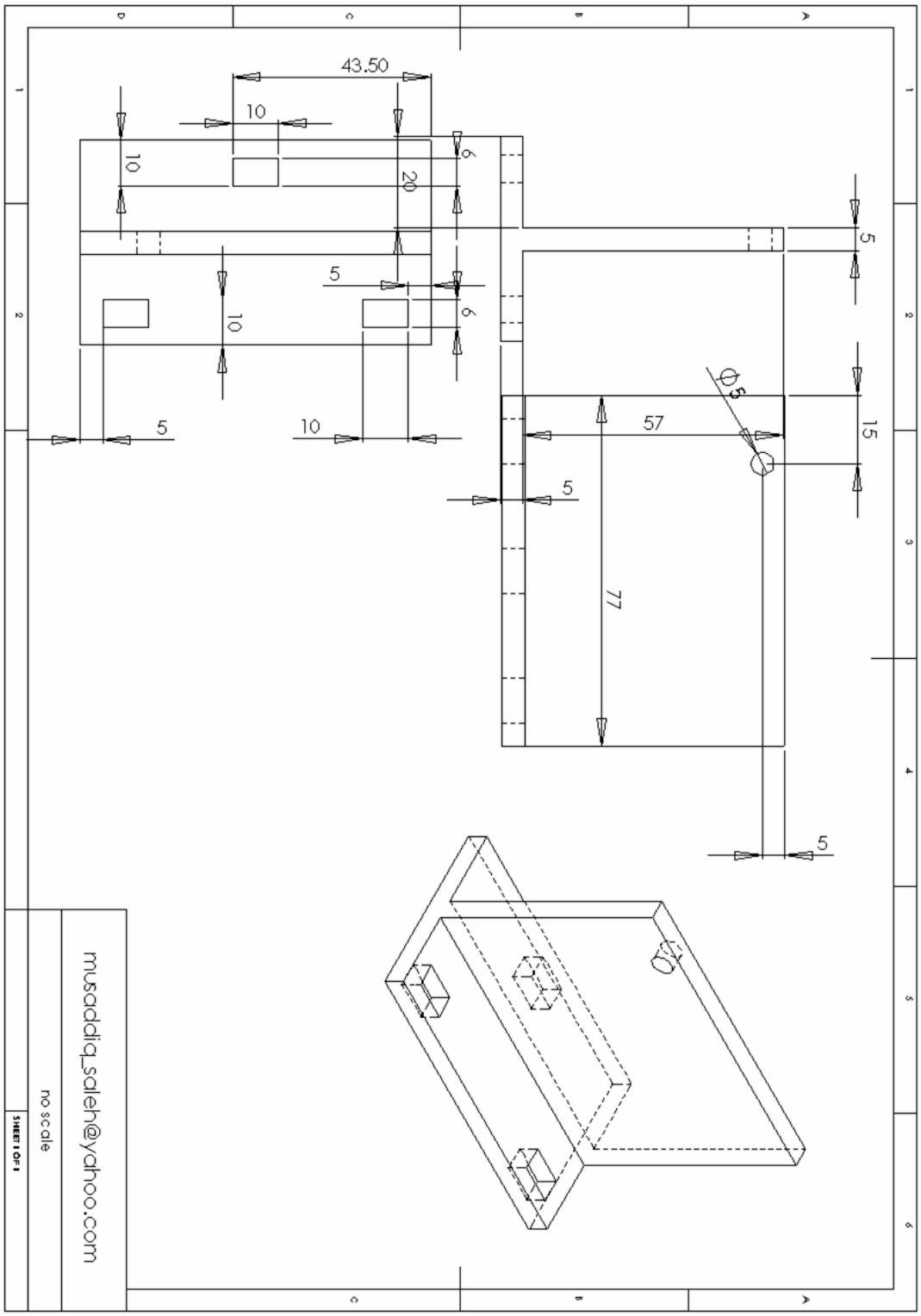
نقشه شماره (4-1) قطعه شماره 50



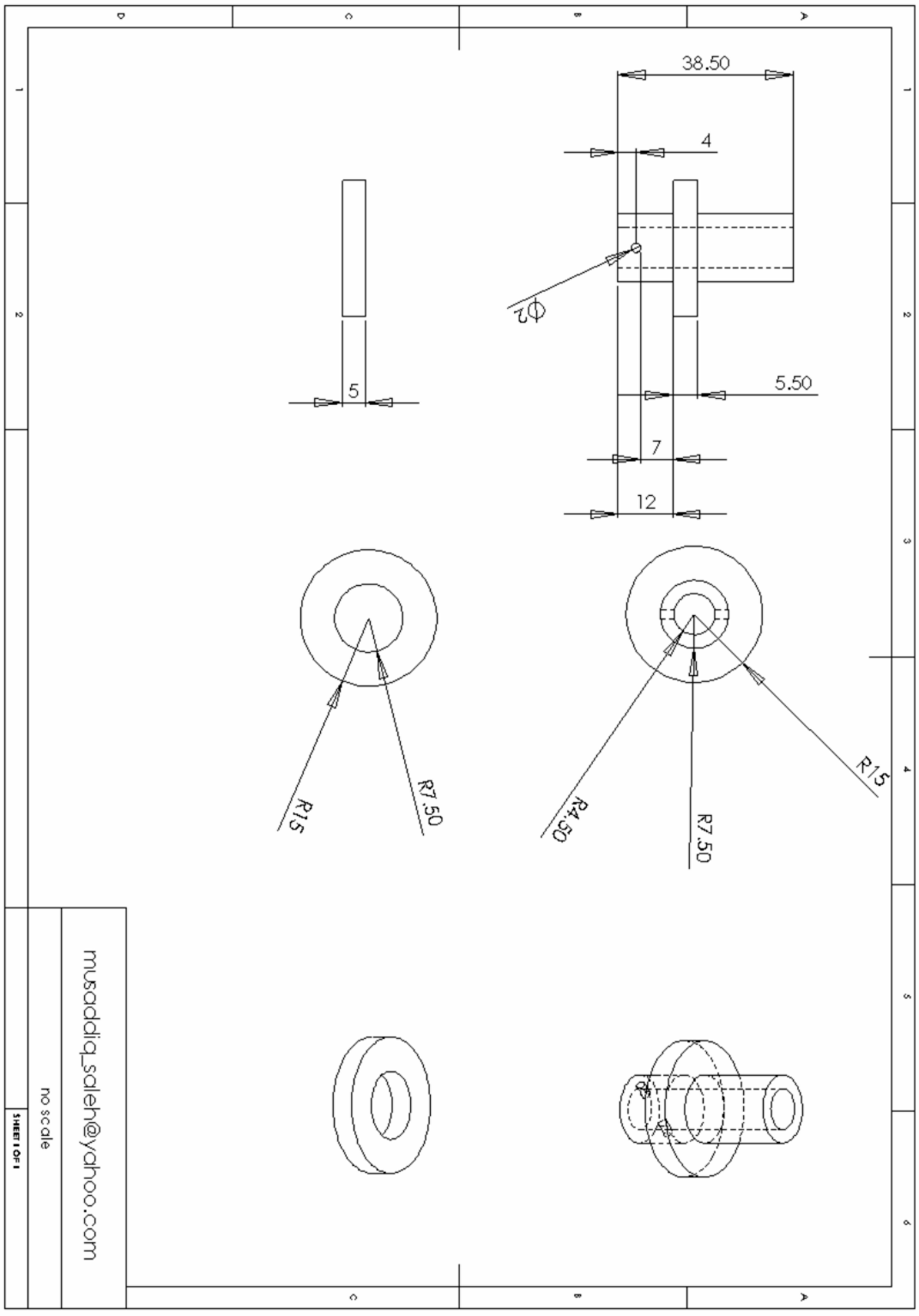
نقشه شماره (4-2) قطعه شماره 60



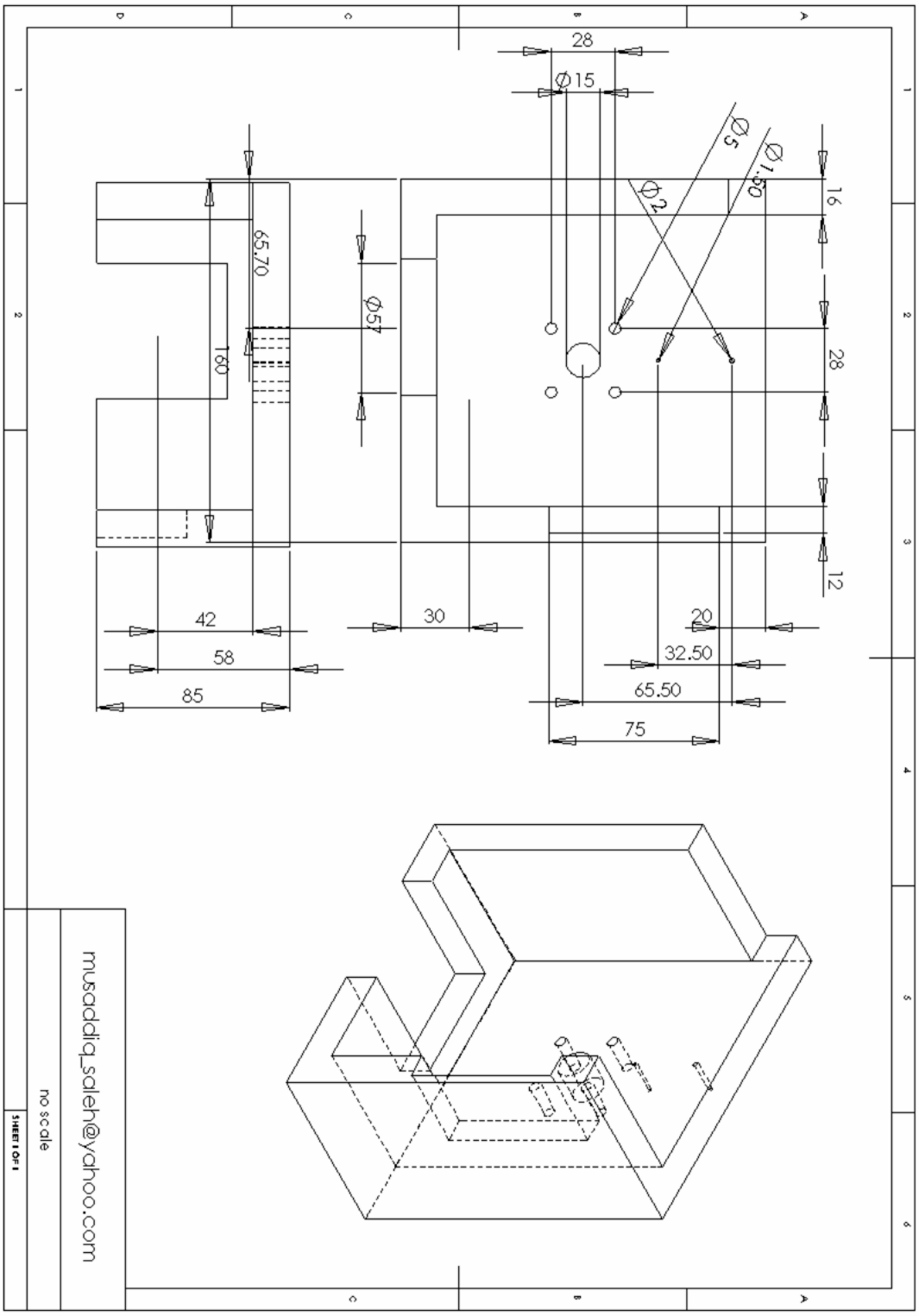
نقشه شماره (4-3) قطعه شماره 42



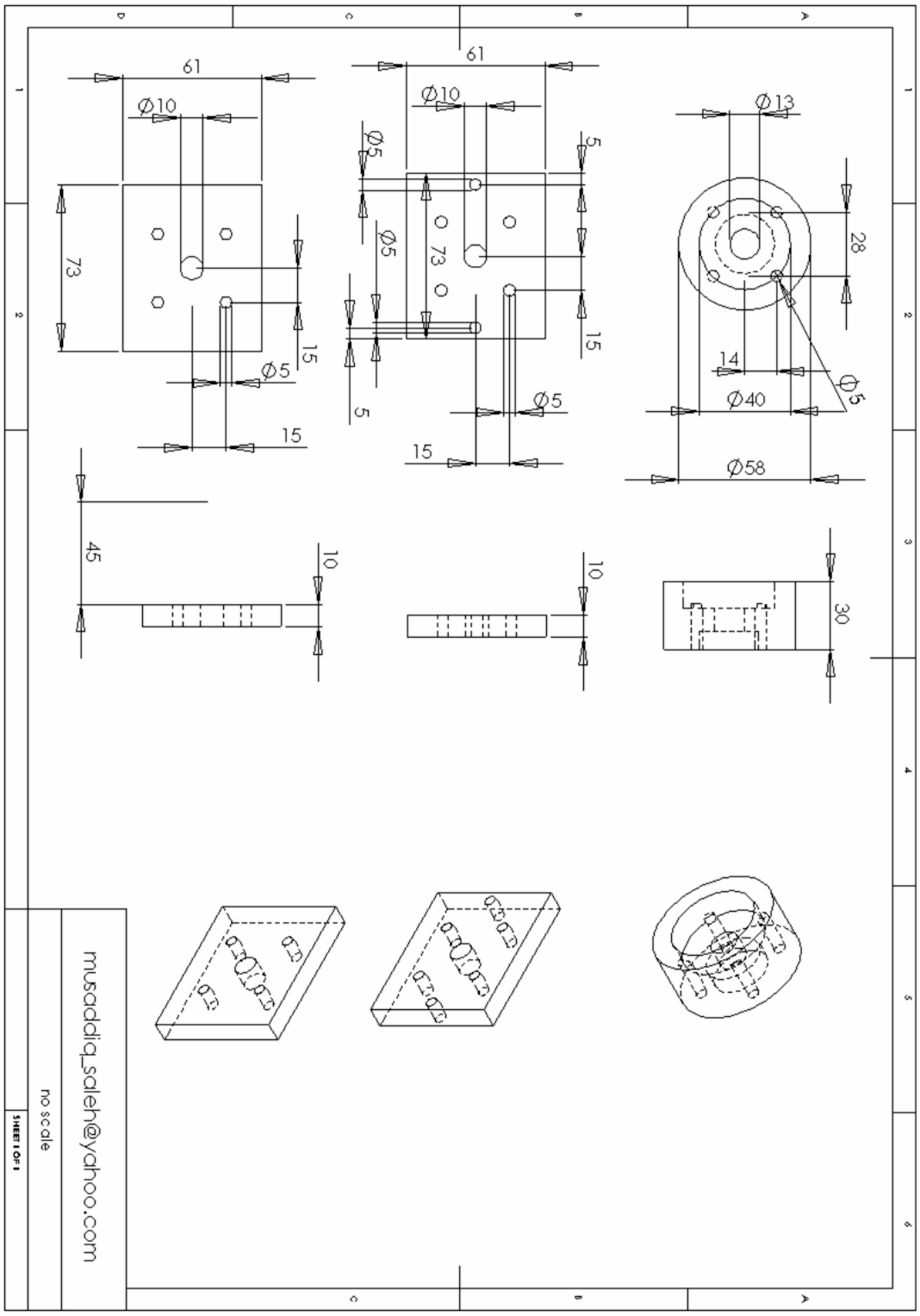
نقشه شماره (4-4) قطعه شماره 49



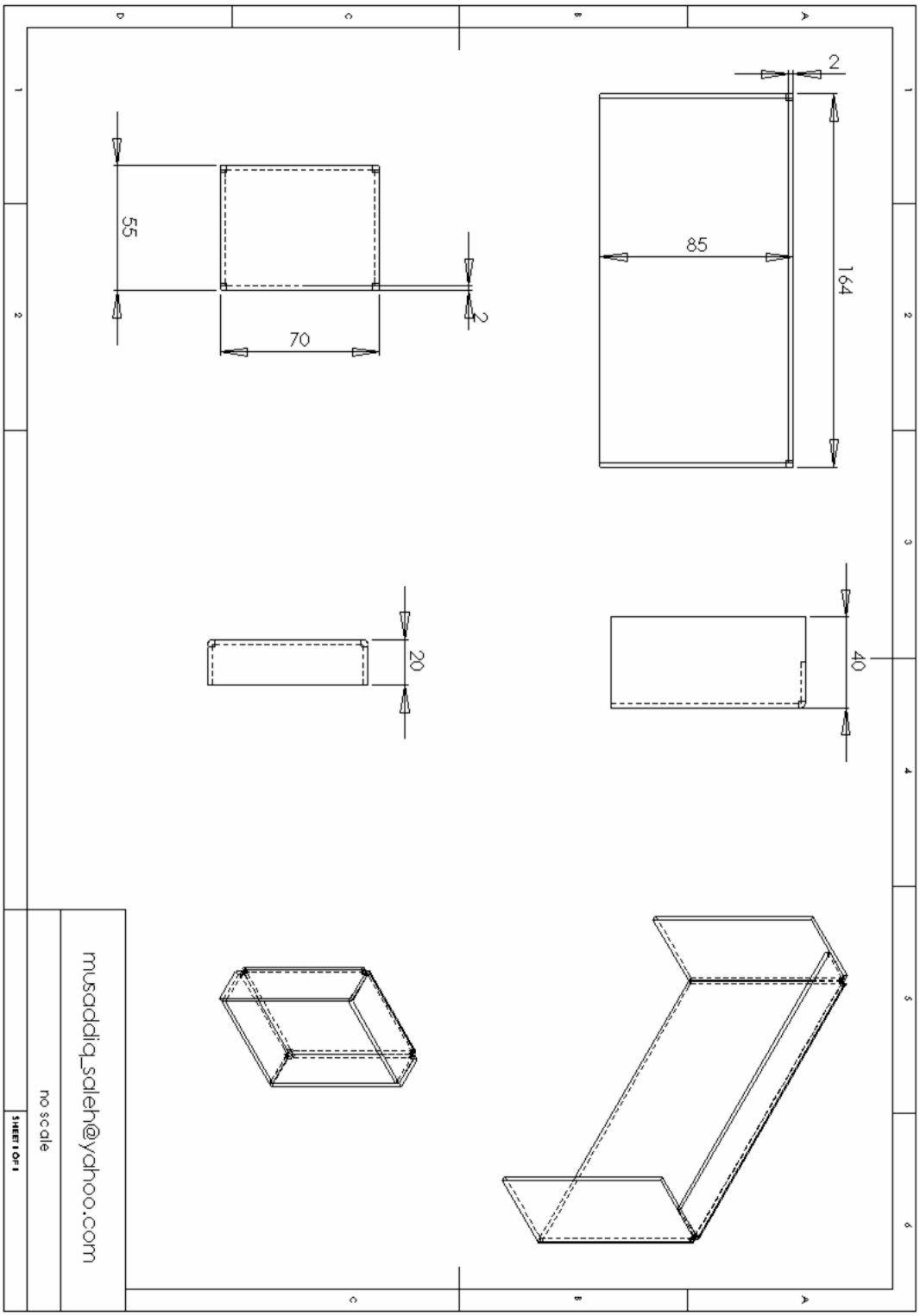
نقشه شماره (4-5) قطعات شماره 54، 57



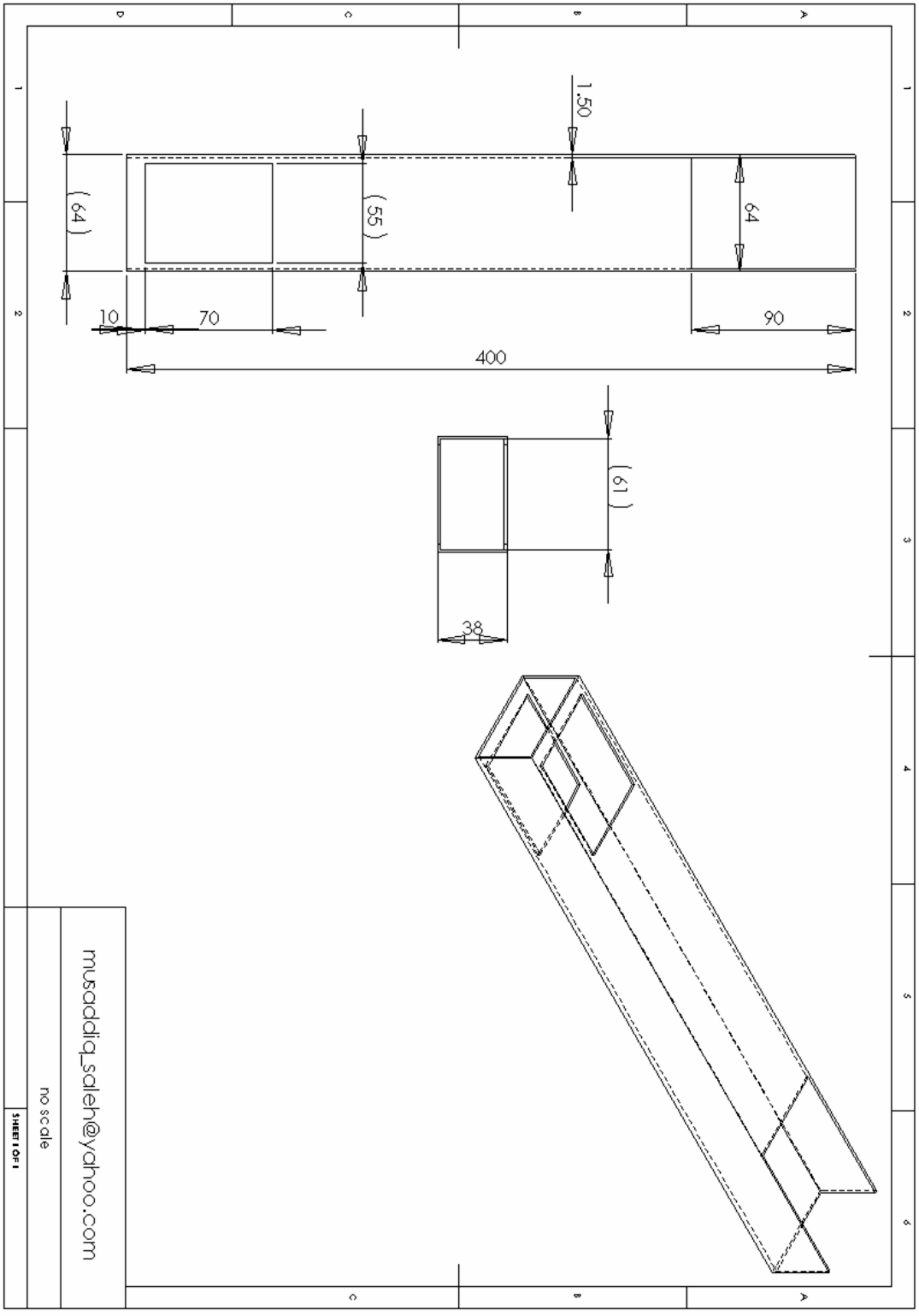
نقشه شماره (4-6) قطعه شماره 39



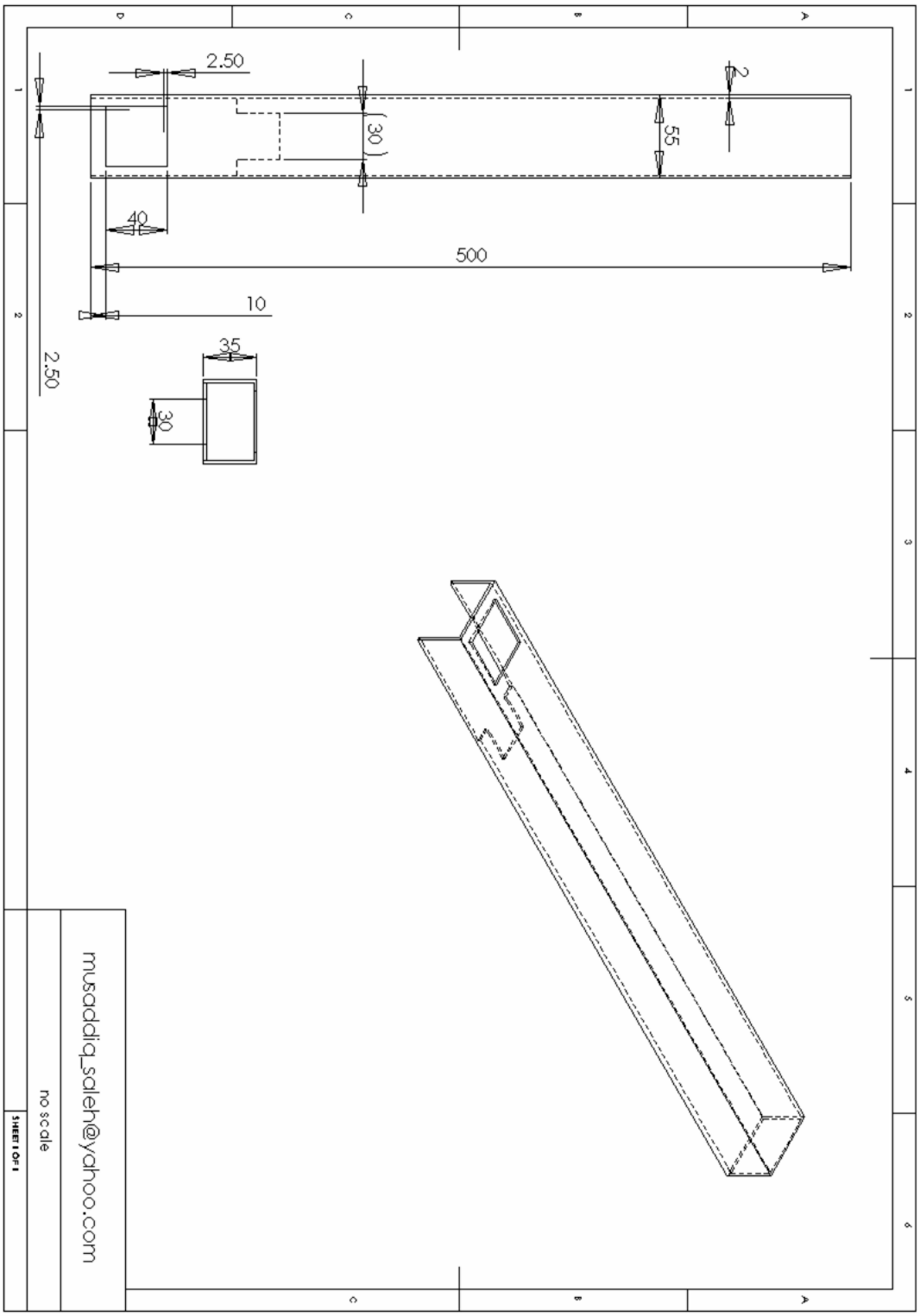
نقشه شماره (4-7) قطعات شماره 44، 40، 41



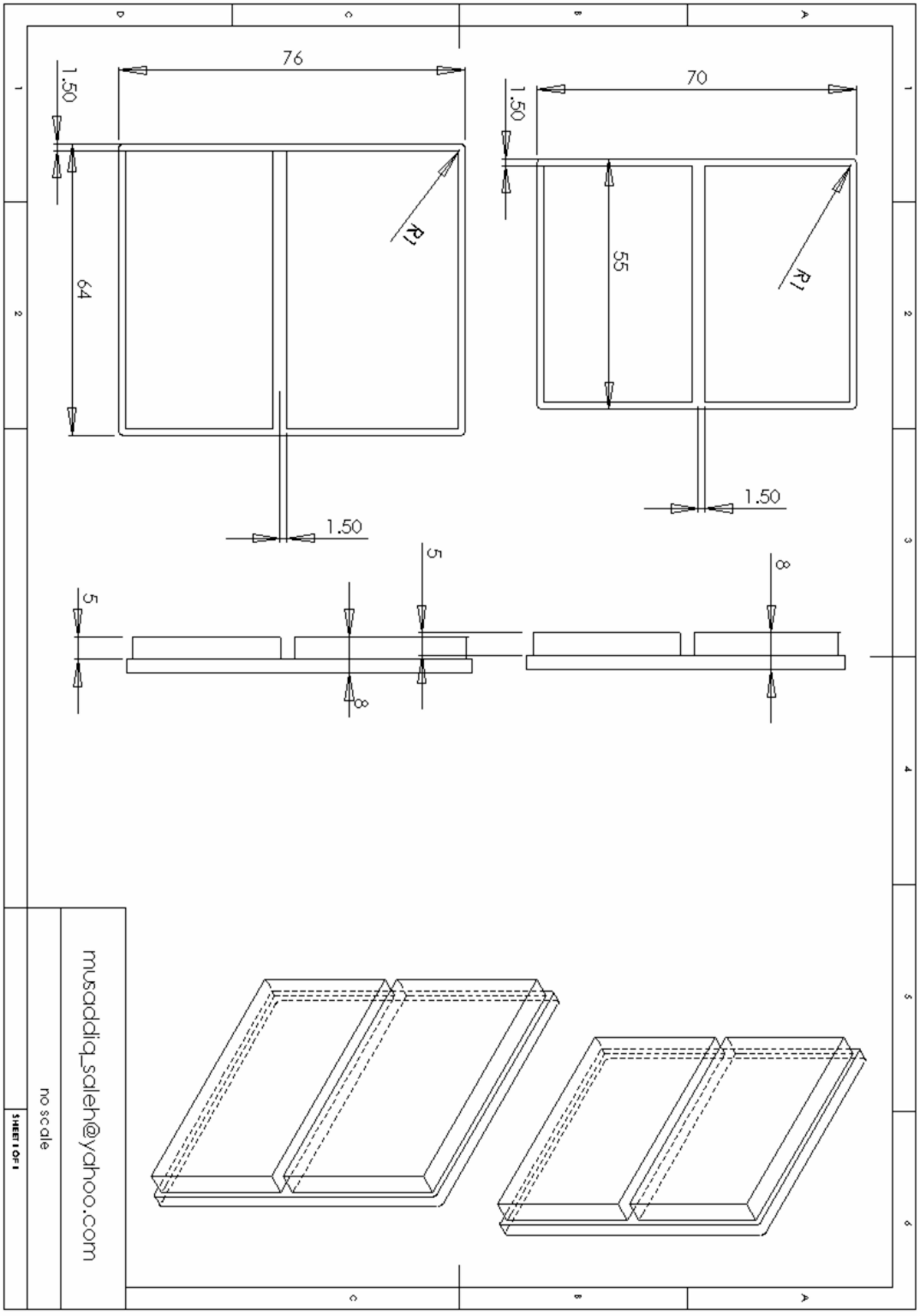
نقشه شماره (4-8) قطعات شماره 46، 33



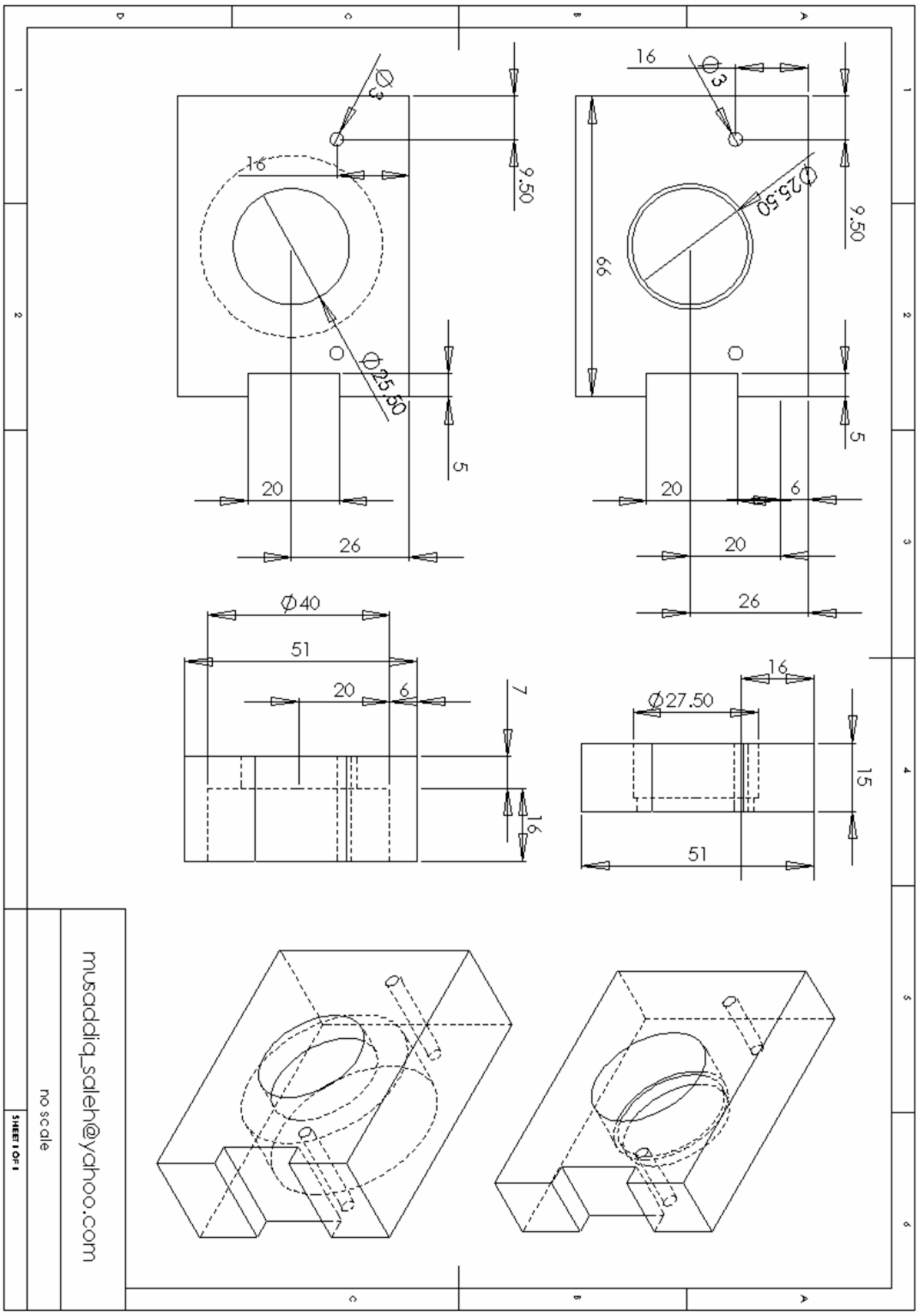
نقشه شماره (4-9) قطعات شماره 35 یا 36



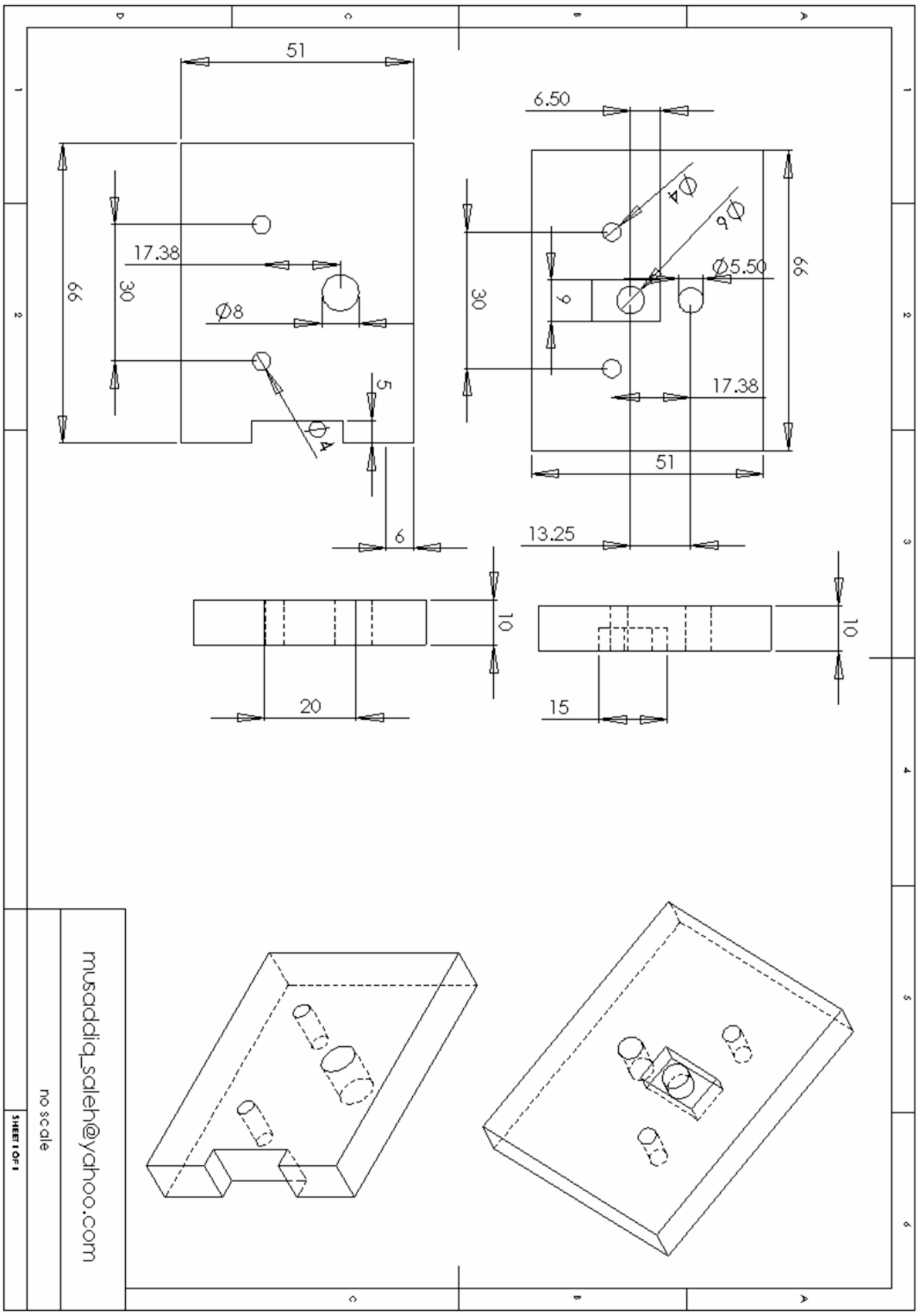
نقشه شماره (4-10) قطعات شماره 21 یا 25



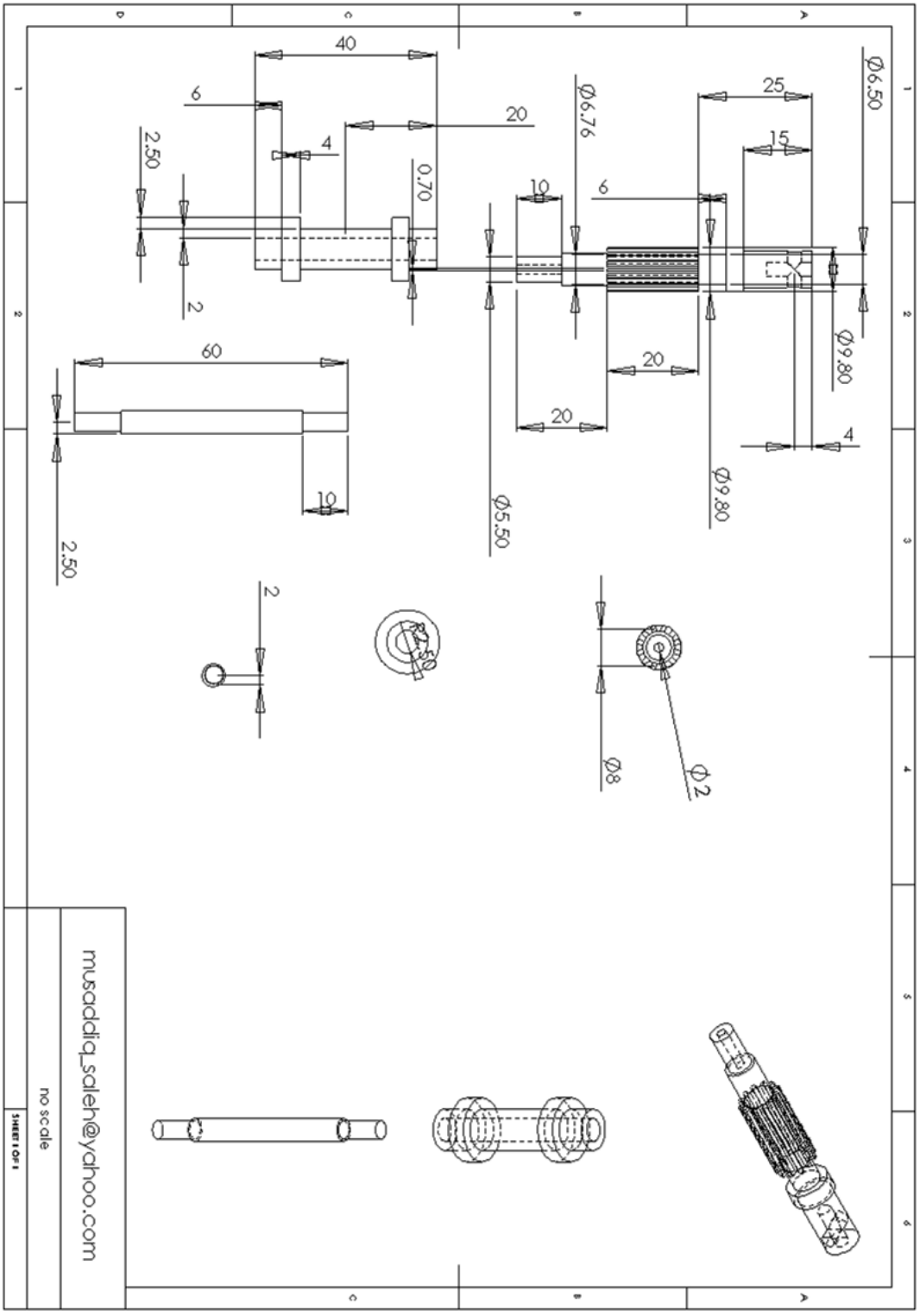
نقشه شماره (4-11) قطعات شماره 38، 37



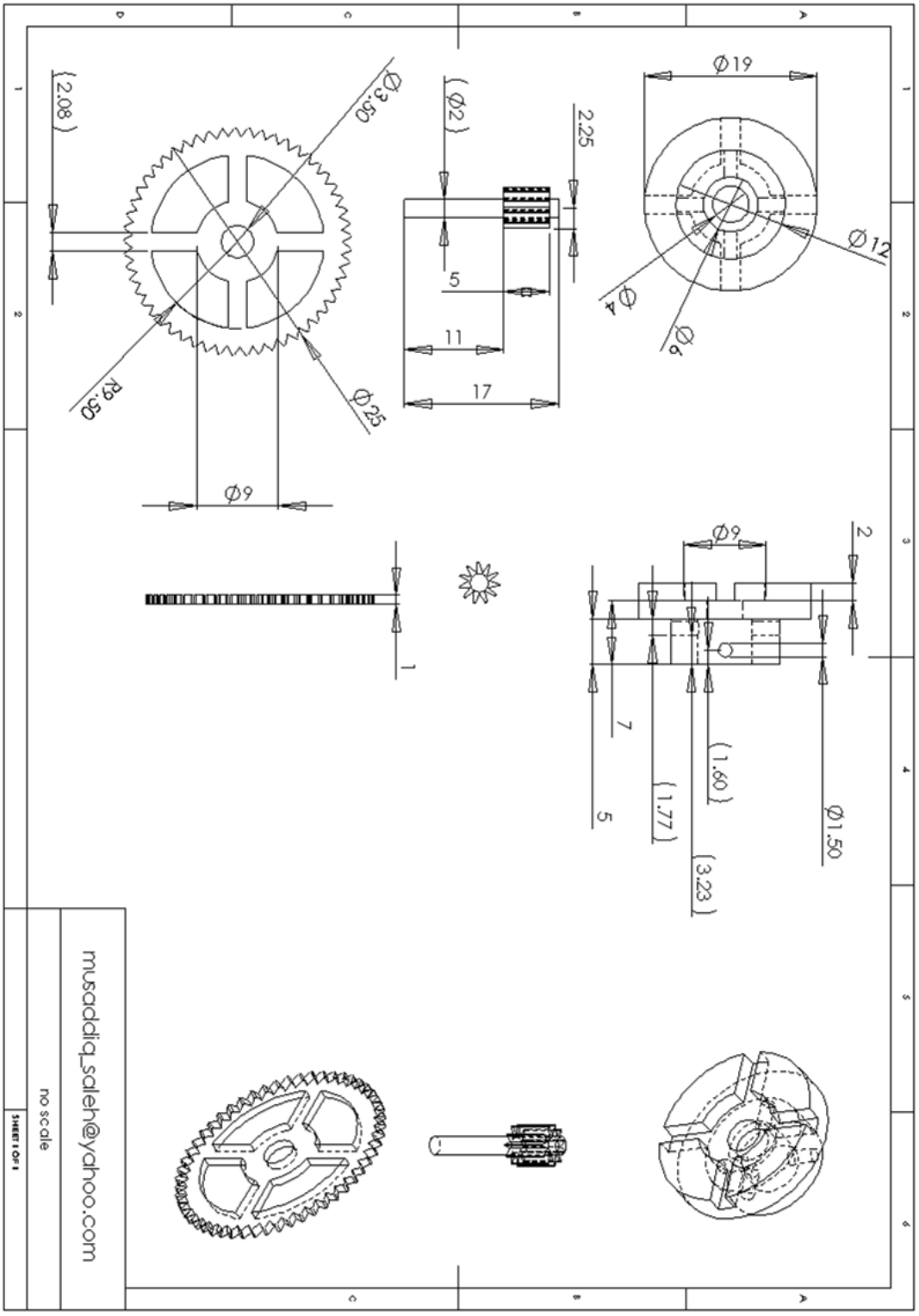
نقشه شماره (4-12) قطعات شماره 22، 23



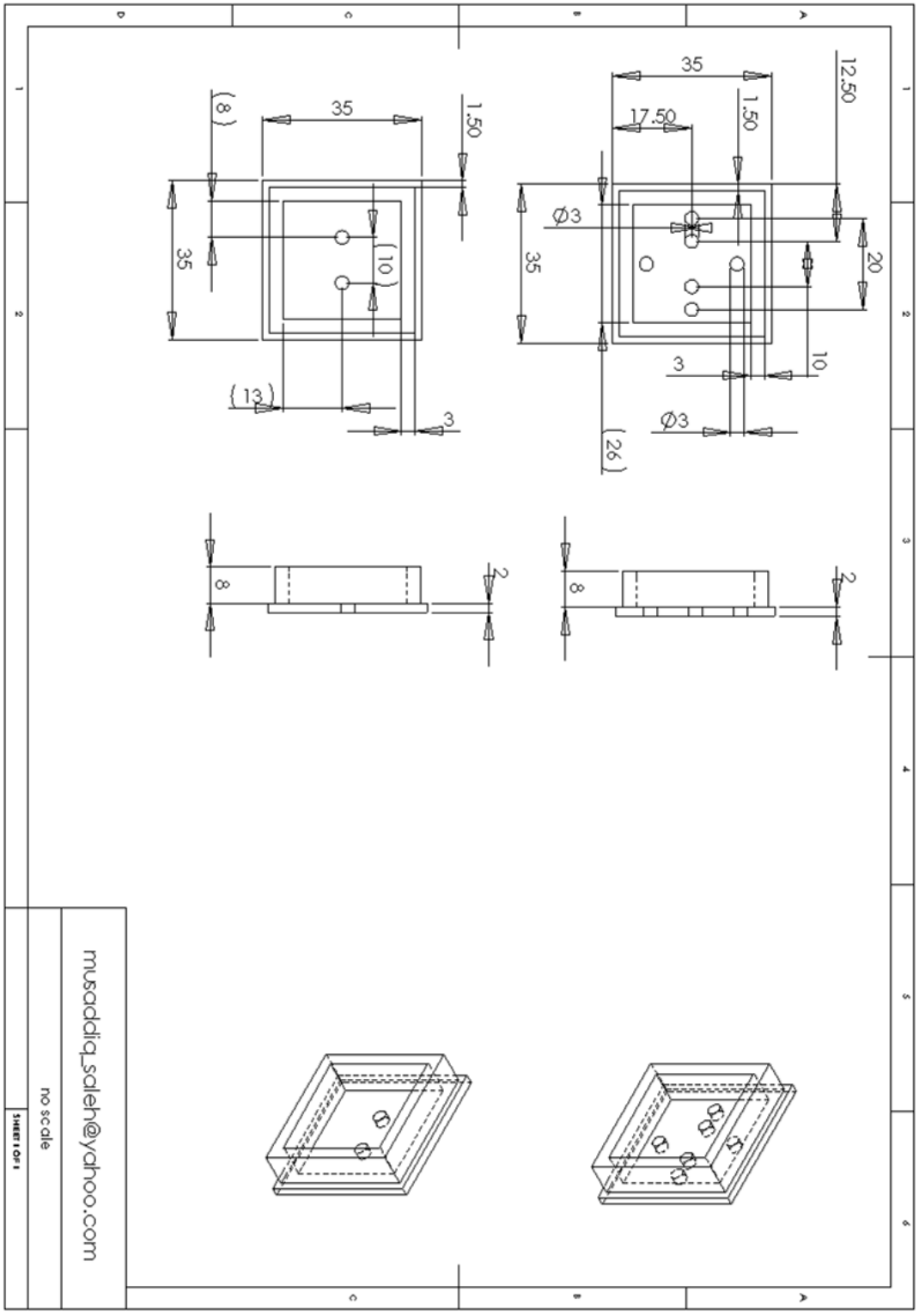
نقشه شماره (4-13) قطعات شماره 29، 24



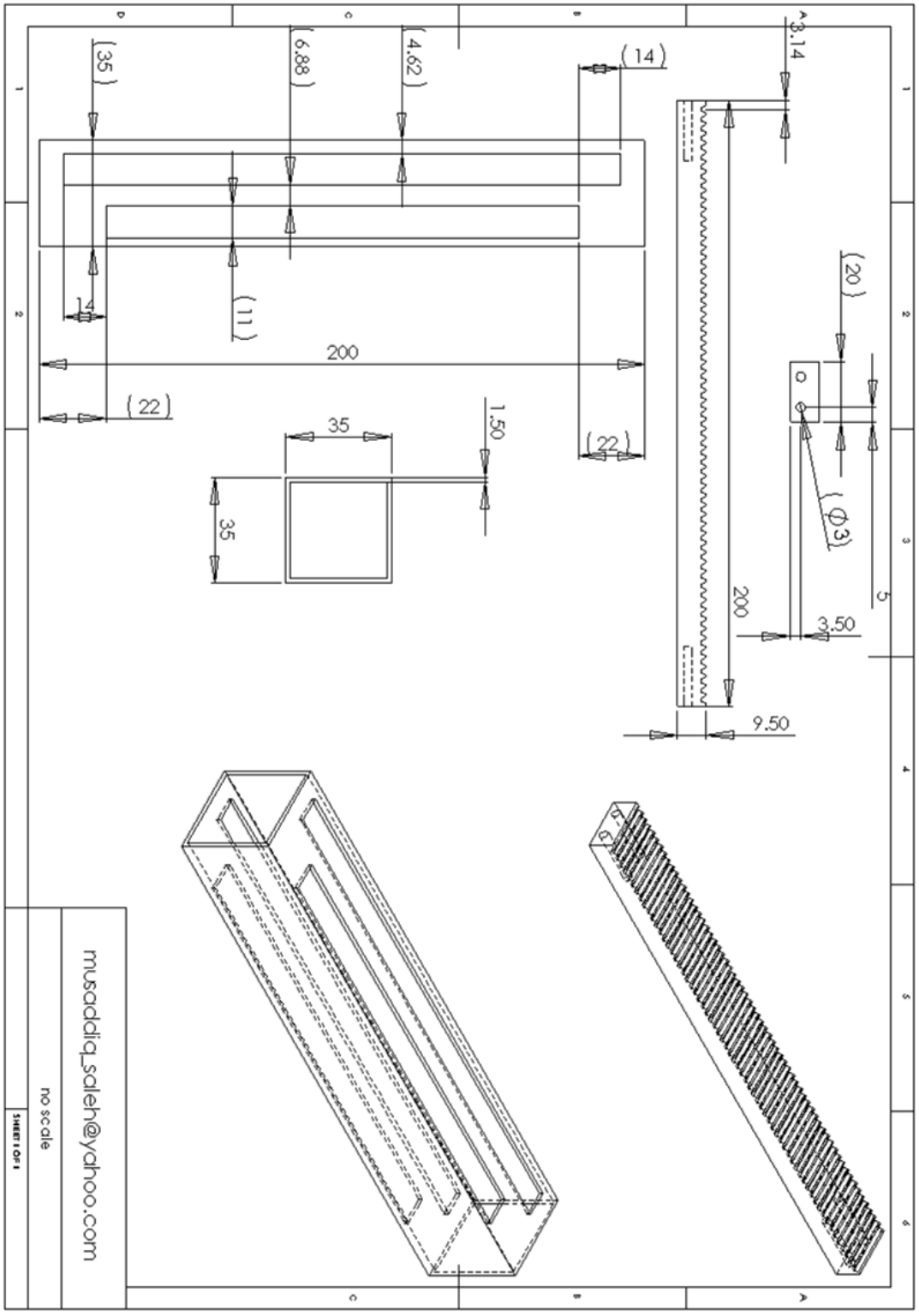
نقشه شماره (4-14) قطعات شماره 28، 27، 26



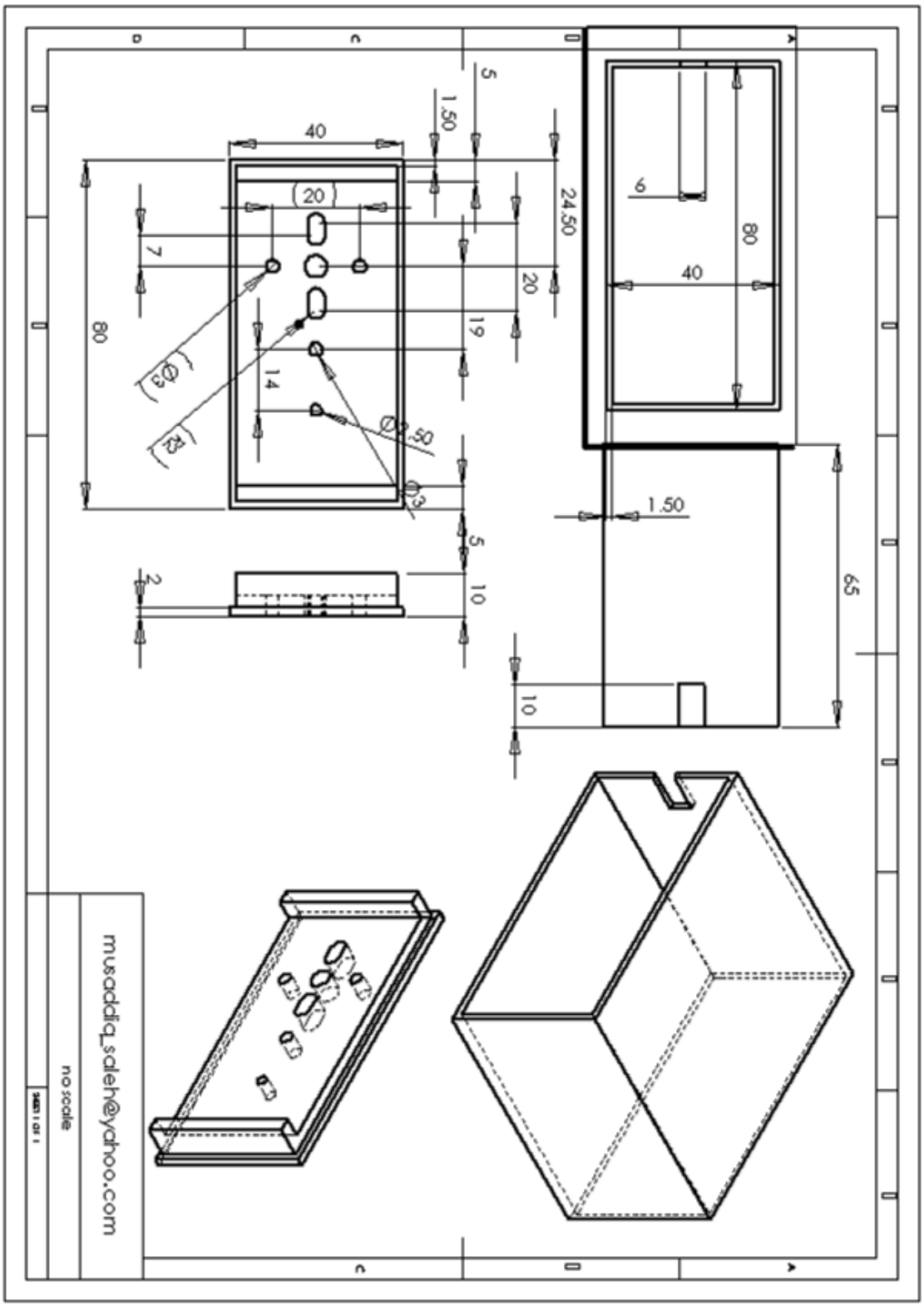
نقشه شماره (4-15) قطعات شماره 31، 34، 32



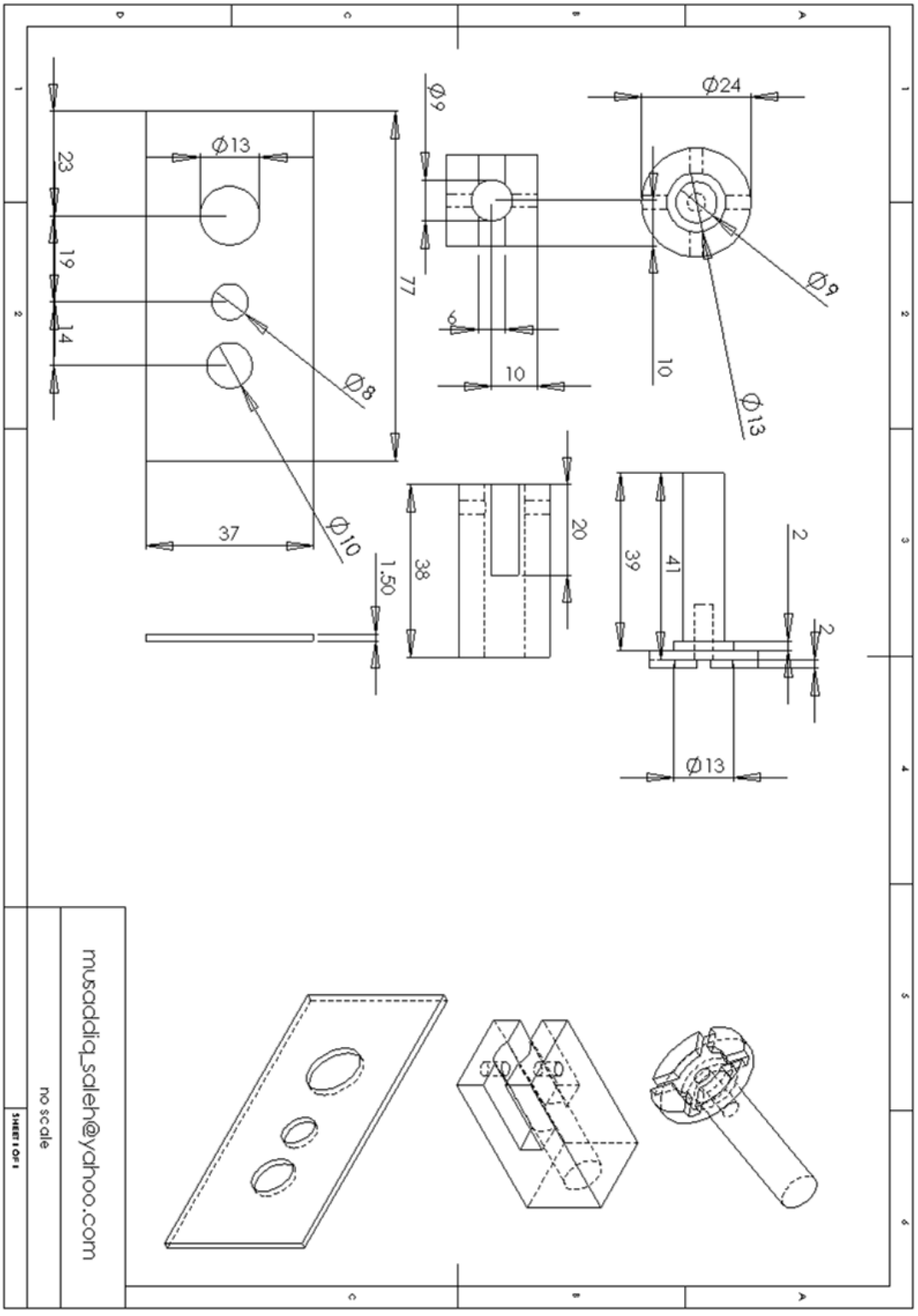
نقشه شماره (4-16) قطعات شماره 4، 3



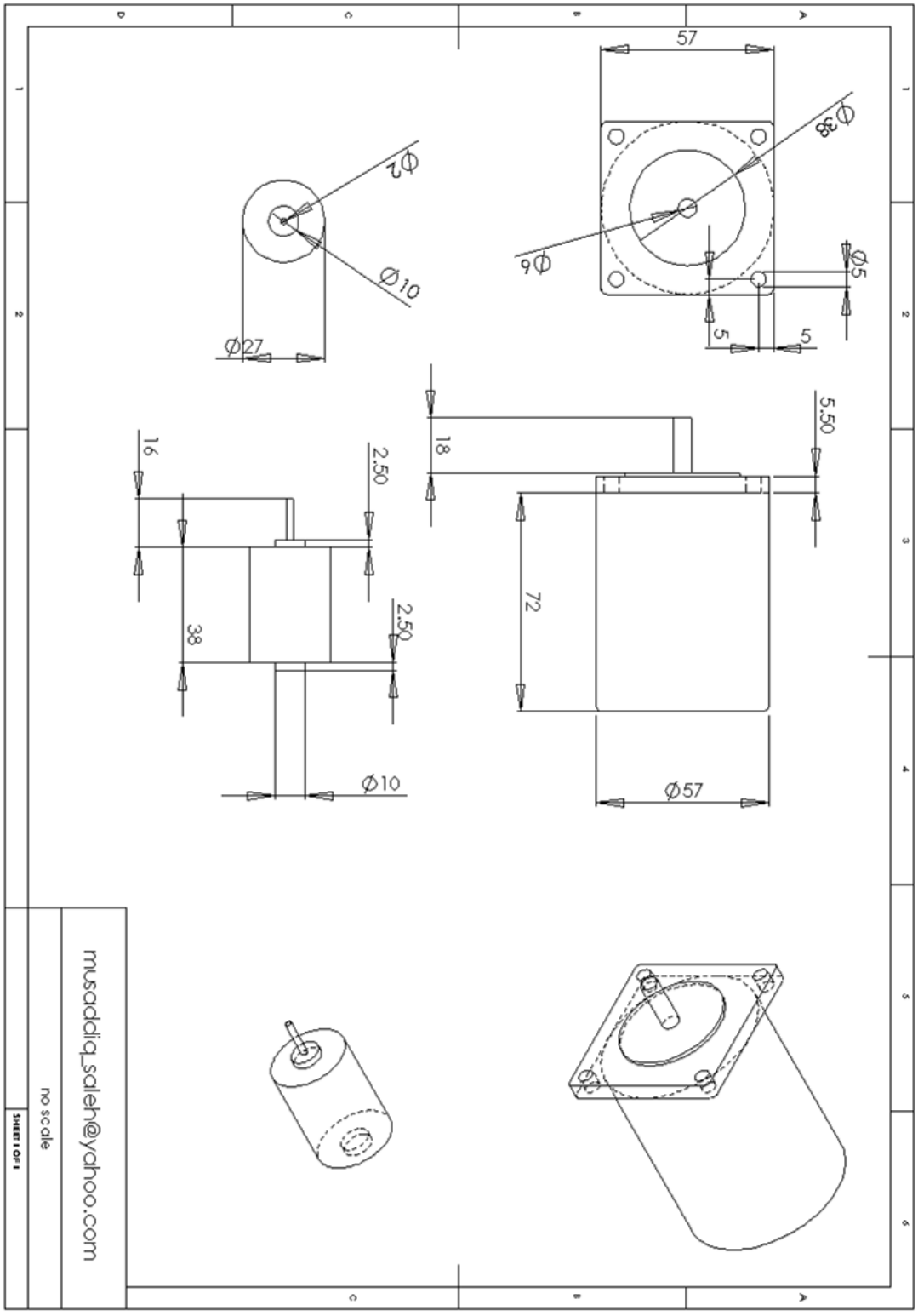
نقشه شماره (4-17) قطعات شماره 2، 1



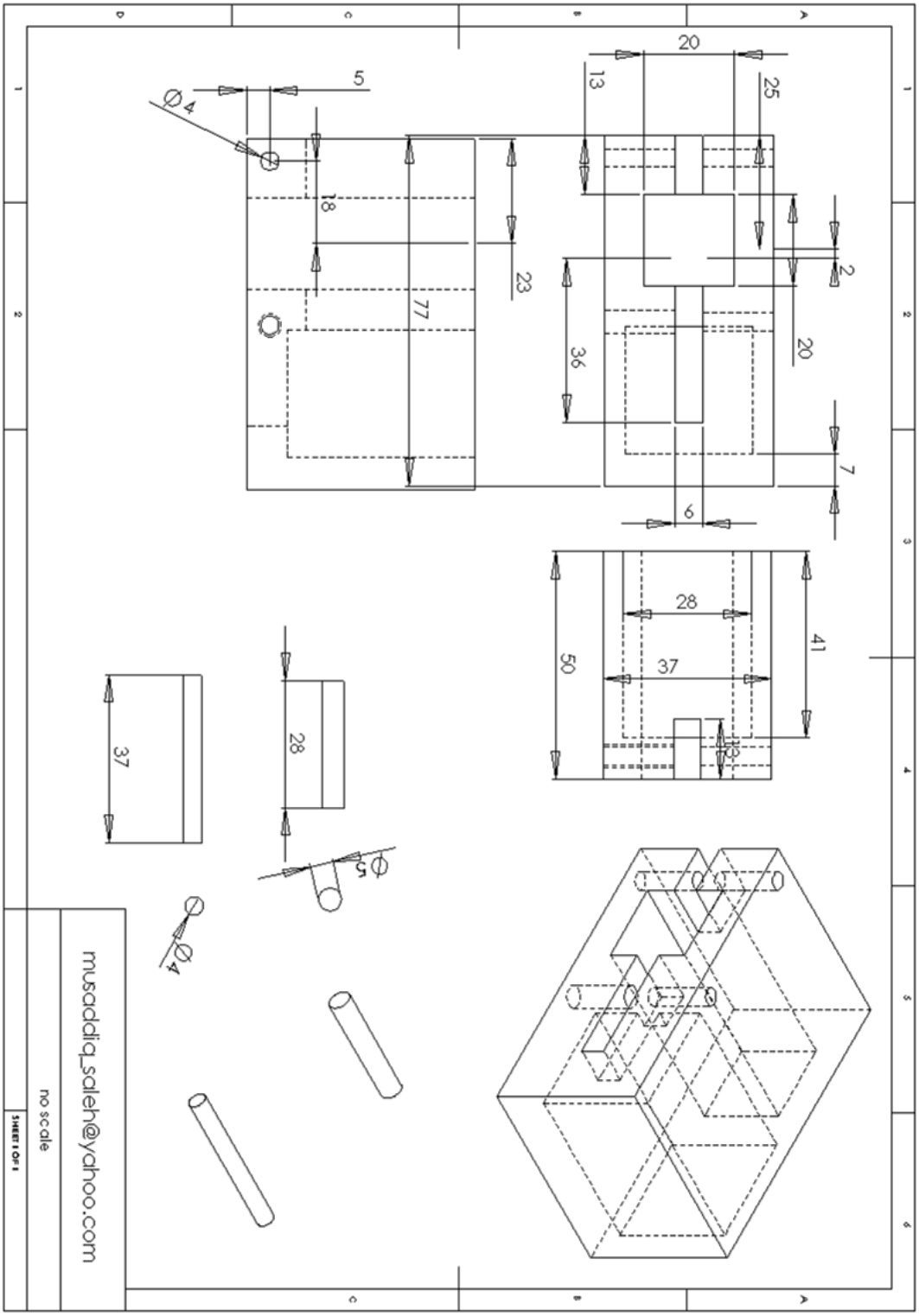
نقشه شماره (4-18) قطعات شماره 6، 5



نقشه شماره (4-19) قطعات شماره 11، 15، 12



نقشه شماره (4-20) قطعات شماره 53، 14- ابعاد استپ موتور و موتورهای DC



نقشه شماره (4-21) قطعات شماره 18، 16، 17

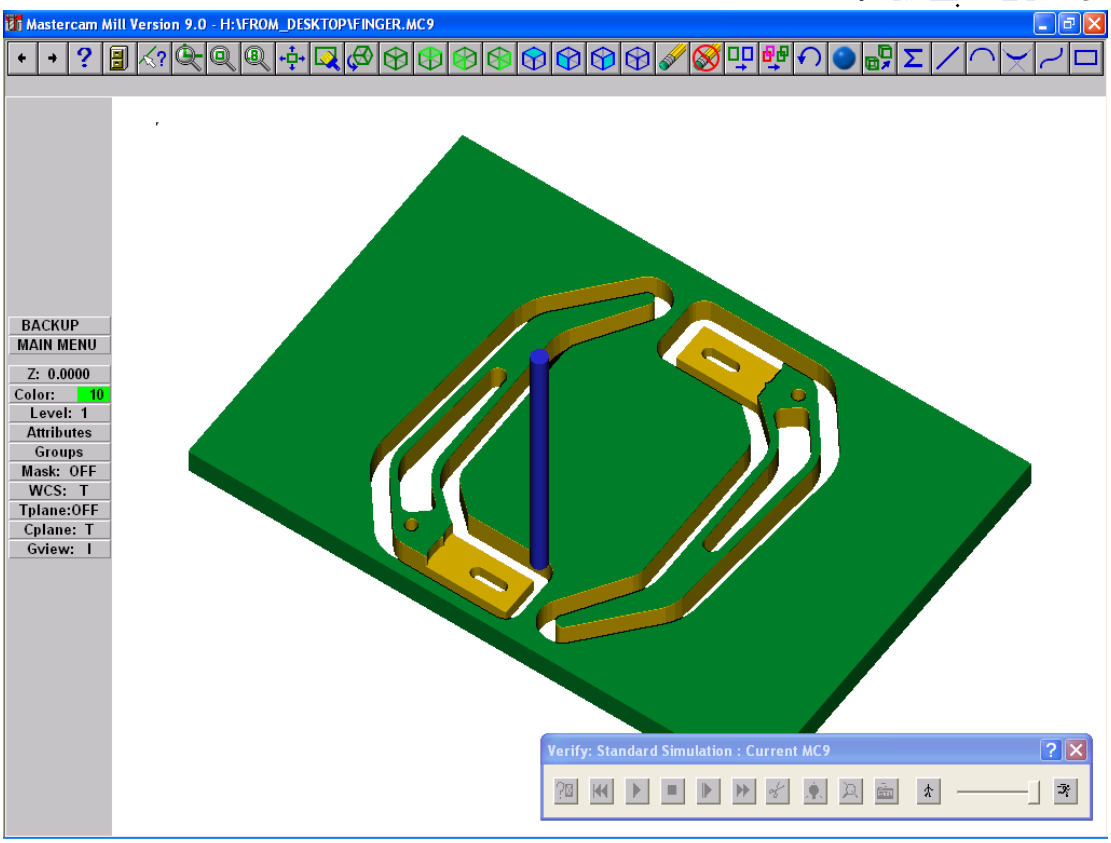
تمام قطعاتي كه نقشه آنها در صفحات قبل‌آمد، داراي فرآيند ساخت ساده‌اي هستند. اما هد انكودر و انگشت بازو ظرافت بيشتري دارند بنابراين براي ساخت آنها از ماشين فرز CNC استفاده شد.

هد انكودر داراي6 قطعه است و انگشت از2 قطعه شبيه هم تشكيل شده است. در صفحات بعد، تصويرهايي ازشبيه سازي صورت گرفته در"MasterCAM 9"براي اين قطعات آورده شده است. پس ازپايان اجراي اين برنامه، قطعات لازم براي مونتاژ 8 هد ويک مجموعه قلاب انگشت بدست مي‌آيد.

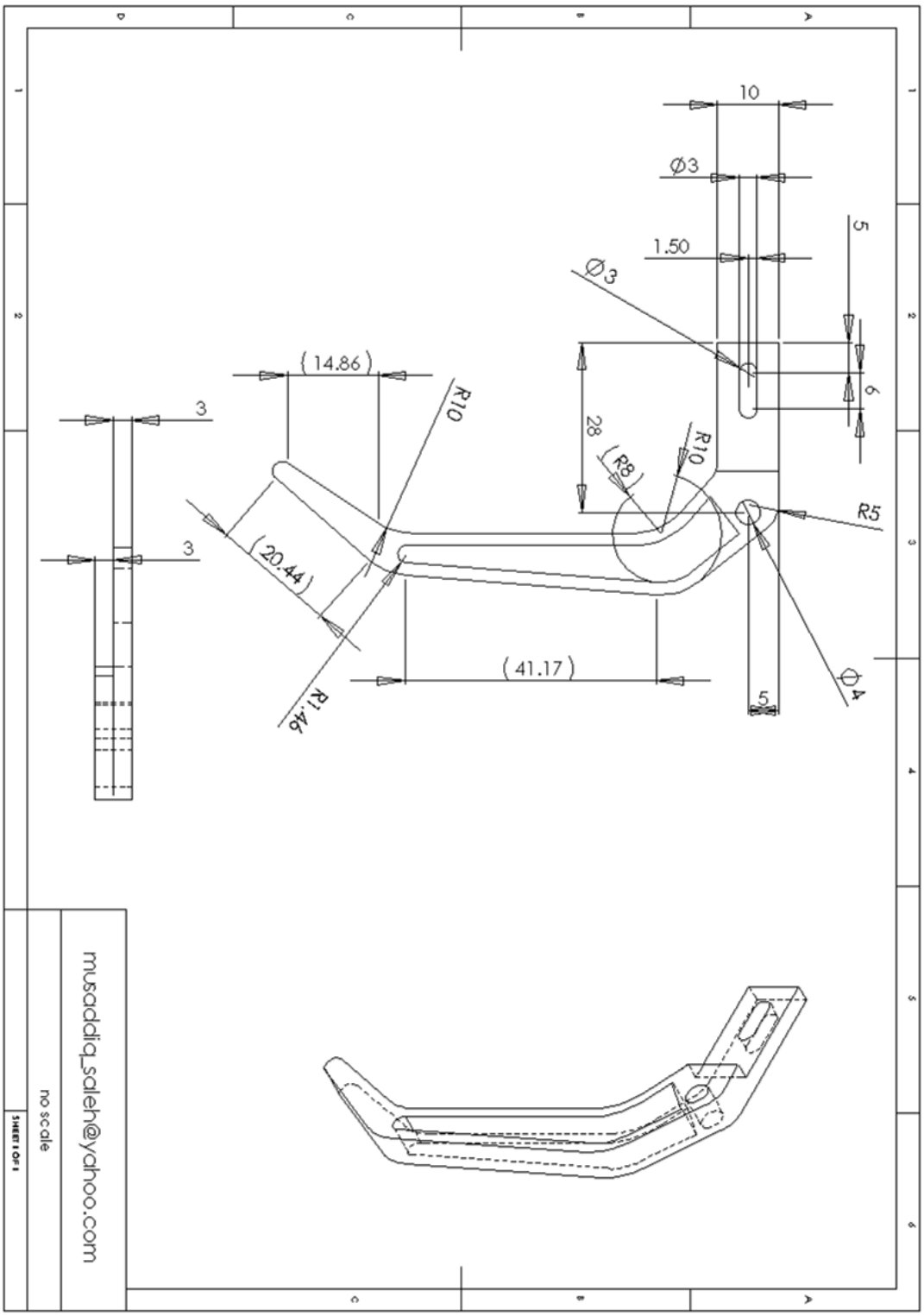
براي‌تهيه Gكدها، ابتدا شكل هد در نرم‌افزار"SolidWork" "طراحي شده، پس از گذر از نرم‌افزار اتوكد وارد نرم افزار"MasterCam"شده است. فايل‌هاي حاوي اينGكدها درشاخه "\CD\MasterCAM files" ديسک فشرده پيوست قراردارد.

ازآنجاكه هرسه ابزارموردااستفاده براي اين قطعات داراي قطركمي بودند (مته1ميليمتر، مته 1/5 ميليمتروتيغه فرزانگشتي1/7) و CNC مورد استفاده نيز داراي تنها يک فشنگي براي ابزارهاي تاقطر1 ميليمتر بود، بنابراين برنامه براي هرابزارجدا نوشته شده است.

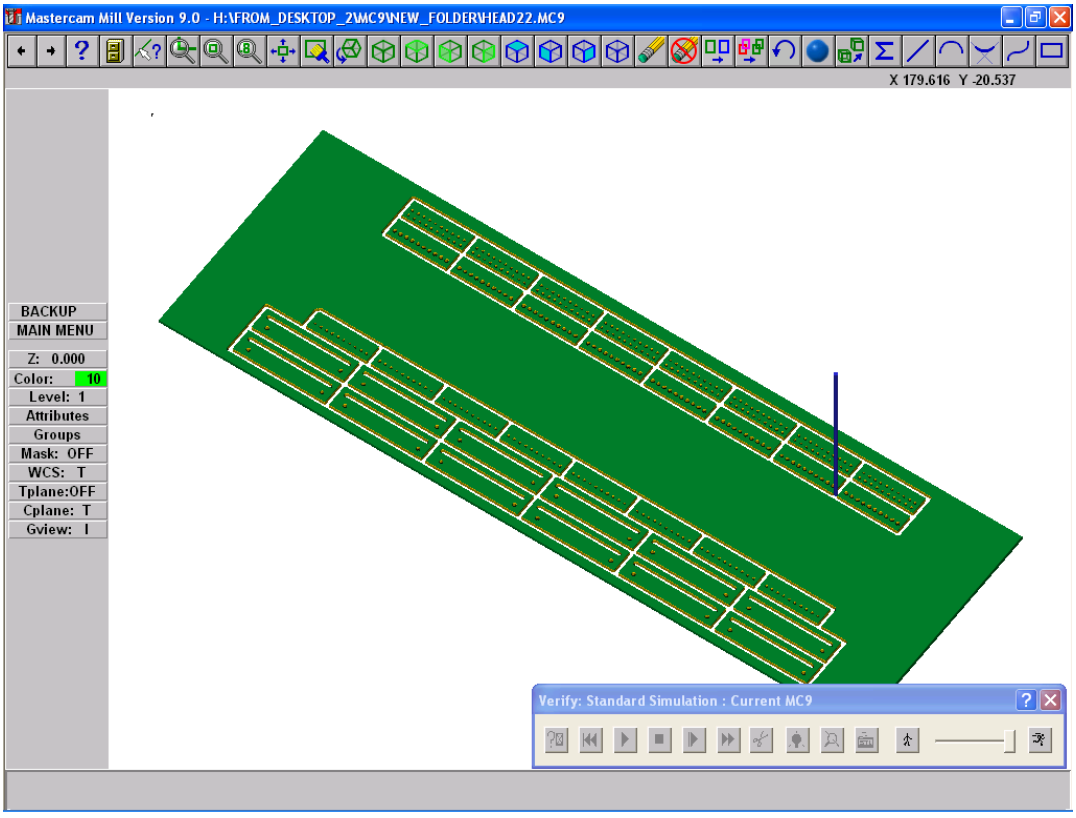
نقشه شماره 4-22 مربوط به جزئيات فكهاى انگشت است. جنس فكها از آلومينيوم است. شيارى كه از بالا تا پايين هر فک كشيده شده است، باعث مى‌شود تا دهانه فک حالت فنرى داشته باشد.



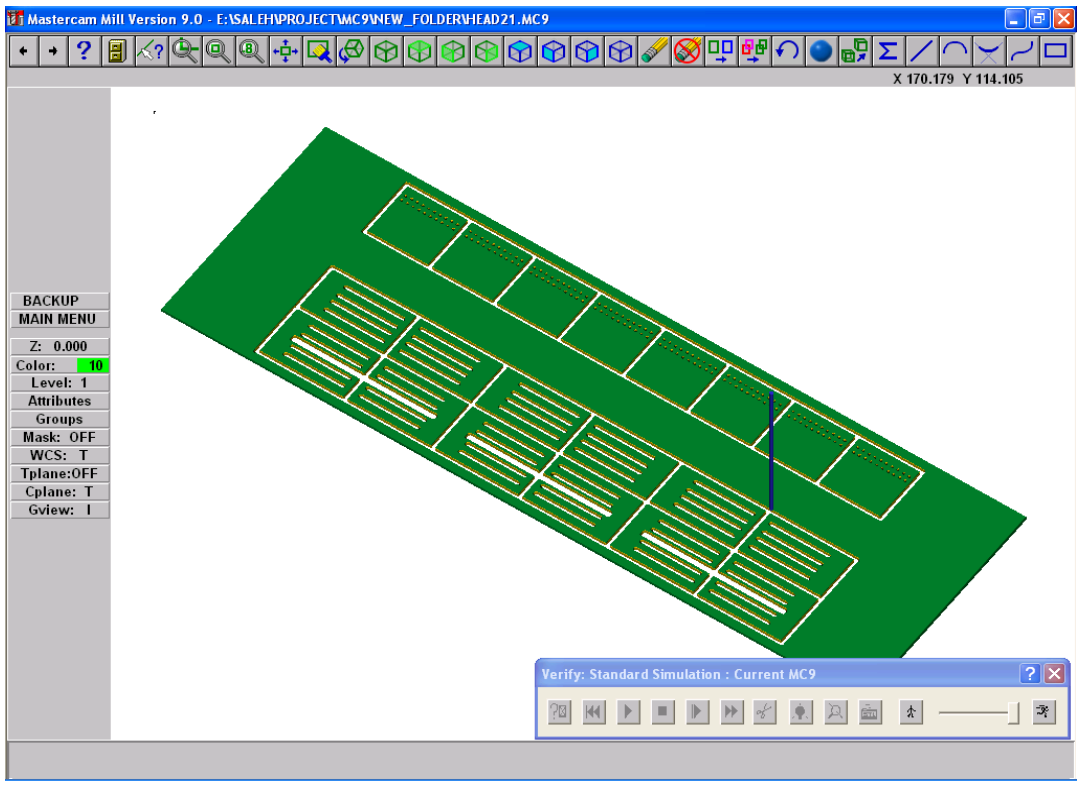
شکل(4-3) تصویر نهائی از شبیه سازی صورت گرفته برای فکهای انگشت



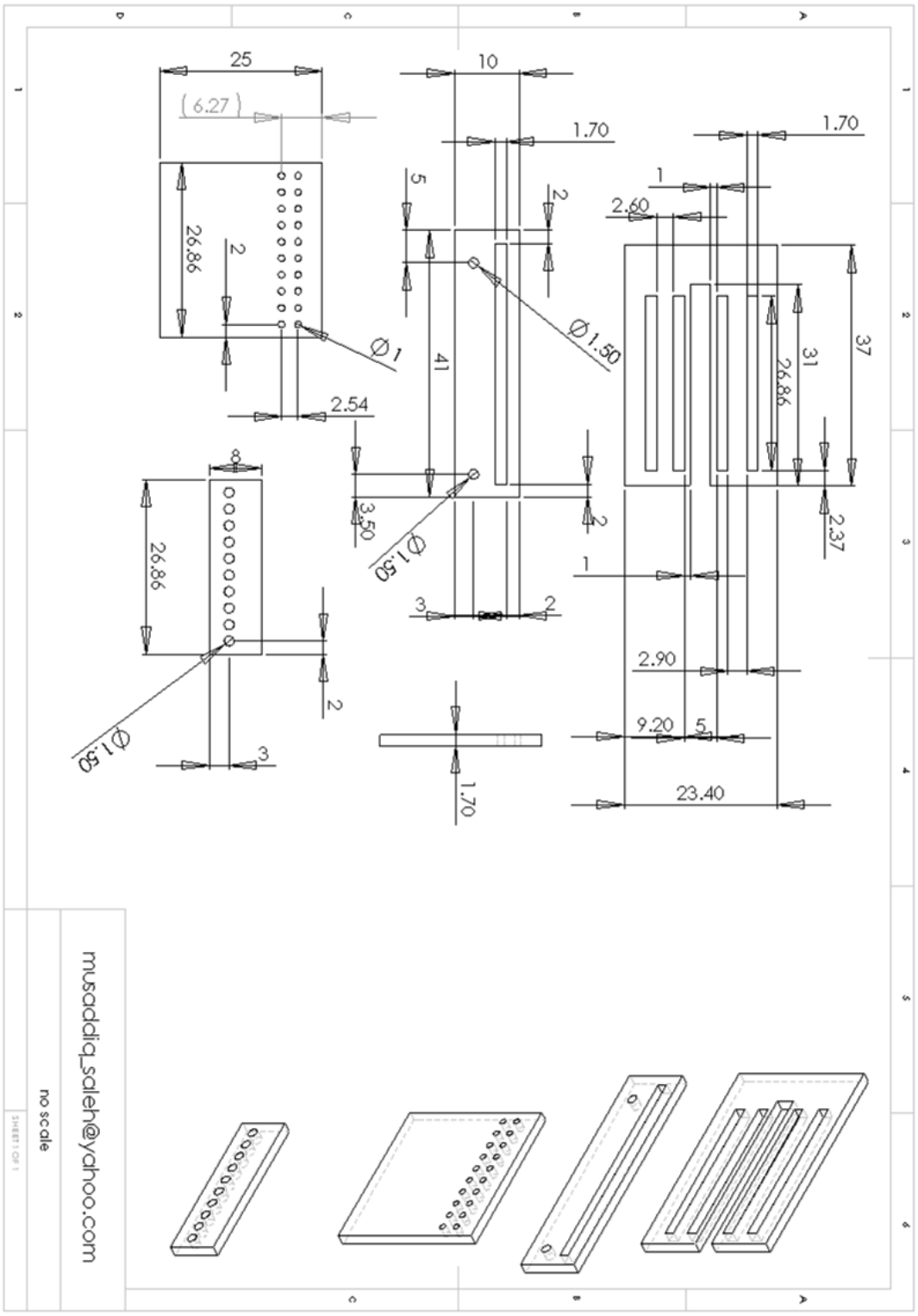
نقشه شماره (4-22) قطعه شماره 19 یا 20



شکل(4-4) تصویر نهائی از شبیه سازی صورت گرفته برای سه قطعه از مجموعه هد انکودر



شکل(4-4) تصویر نهائی از شبیه سازی صورت گرفته برای سه قطعه دیگر از مجموعه هد انکودر



نقشه شماره (4-23) اجزای قطعه 55

نقشه شماره 4-22 مربوط به قطعات تشکیل‌دهنده هد انکودر است. جنس این قطعات فیبر فایبرگلاس دو طرف مس است.

فصل پنجم: مدارهاي الكتريكي

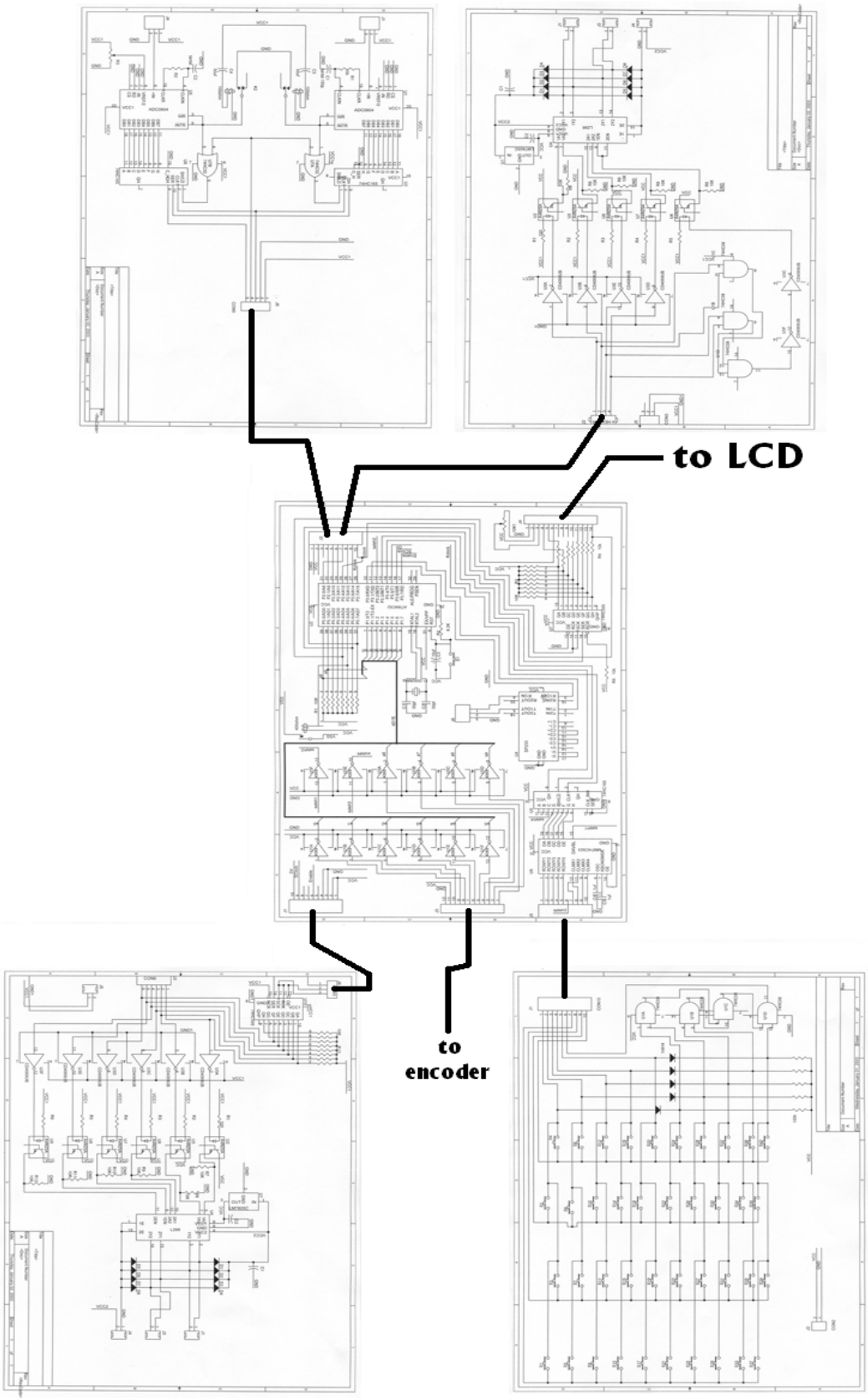
درصفحات بعدي دياگرام كلي و دياگرام جزئي قسمت هاي الكتريكي اين بازو را به همراه شماتيك و PCB[1] آنها مشاهده مي‌كنيد (نرم افزار مورد استفاده Orcad9.2 است). توضيحات لازم مربوط به هر قسمت دركنارآن آورده شده است.
ذكر چند نكته ضروري است:
- در طراحي اين مدارها ارزان بودن، ساده بودن وعملي بودن، نقش اصلي را داشته اند.
-- اين مدارها در ابتدا به صورت جزئي يا به طور كامل بر روي برد آزمايش[2] بسته شده‌اند و پس از حصول اطمينان ازكاراييآنها درداخل نرم‌افزار" Orcad "ترسيم وPCB آنها بدستآمده است. بجز يک مورد كه در زير به آن اشاره می شود، هيچ تفاوتي بين مدار موجود بر روي بردآزمايش و مدارتكميل شده نهايي وجود نداشت.
- درهنگام اتصال LCD برروي بردآزمايش از يک سوكت ISA[3] استفاده شد و مشكلي هم وجود نداشت اما در هنگام اتصال برروي PCB ازكابل پهن 14 تايي و بلند استفاده شدكه باعث شد تا اطلاعات رسيده به LCD ناقص باشد. تغييرطول‌كابل‌و زمانبندي بين پالسها و طول پالسها نيزاثري نداشت. در نهايت استفاده ازمد 4 بيتي براي ارسال اطلاعات به LCD مشكل راحل‌كرد.
اگرPCB ها داراي ابعادكوچكي هستند و قطعات بر روي آنها به صورت فشرده قرار دارند، تنها به اين دليل استكه محفظه‌ايكه اين بردها درآن قرار مي‌گيرند محدود است. در اينجا نيز سعي شده است تا ازحفره‌هاي موجود دربدنه و پايه بازو استفاده شود تا مجبور به اضافه‌كردن محفظه‌اي

---

[1] Printed Circuit Board
[2] Bread Board
[3] Industry Standard Architecture

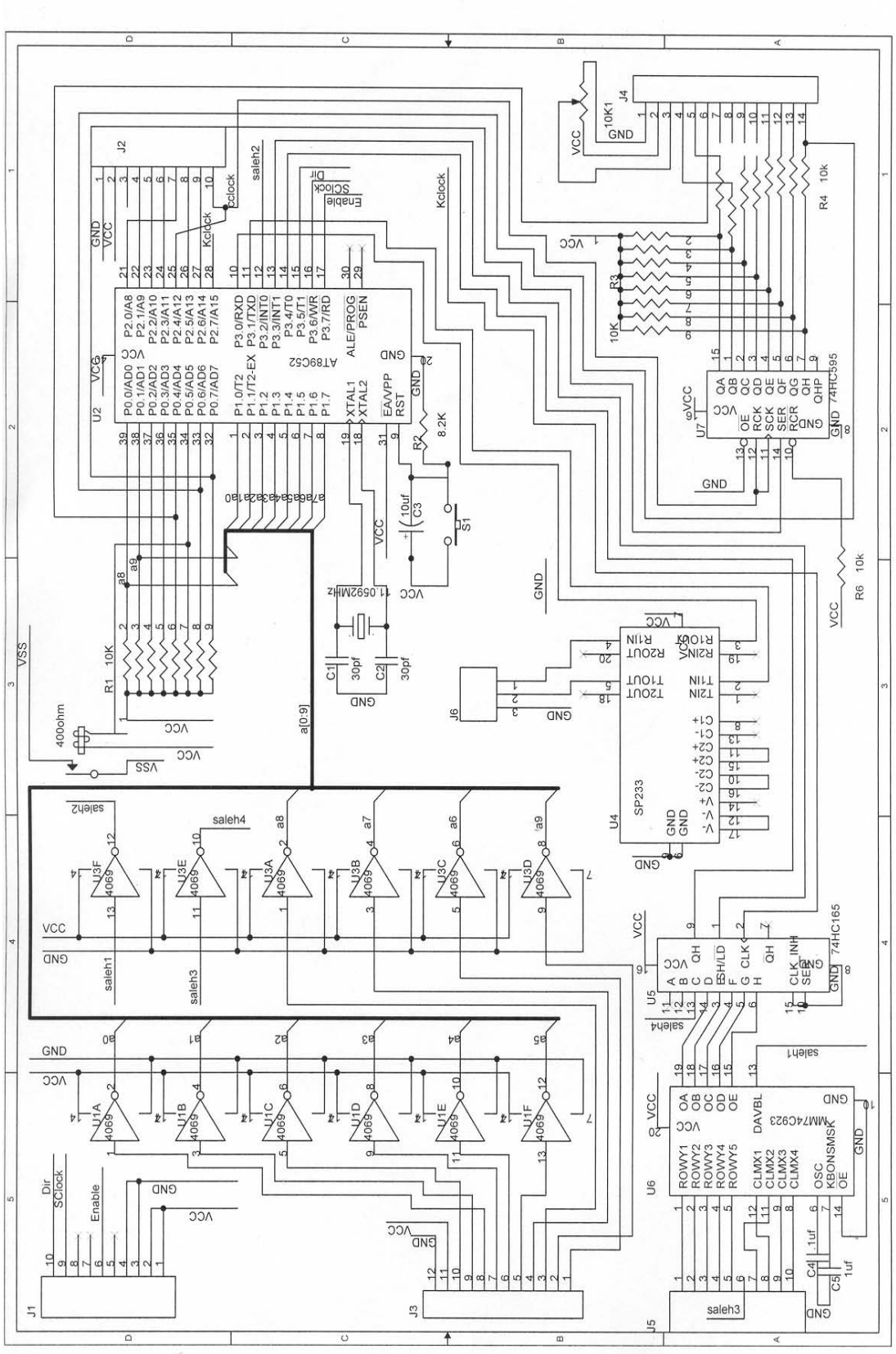
براي نگهداري PCB هانشويم و تناسبات و سادگي بازو حفظ شود.
- در توضيحات مربوط به هر شماتيک تنها به بررسي نقش آي سي ها و قطعات ديگر و در مواردي دلايل استفاده ازآنها مي پردازيم و از تشريح قطعات جلوگيري مي‌کنيم. اين اطلاعات را مي‌توان در برگه اطلاعات اين‌آي‌سي‌ها مطالعه کرد.
- PCB ها دراندازه واقعي خود چاپ نشده اند اما تمام فايل هاي ارکد مربوط به آنها در CD ضميمه اين پايان نامه در شاخه \CD\ORCAD_Project\ قرار دارد.
- اطلاعات مربوط به آی‌سی ها و ساير قطعات استفاده شده برای مدارات الکتريکی در شاخه \CD\ORCAD_Project\data sheet ازديسک فشرده همراه پايان نامه قراردارد.

-

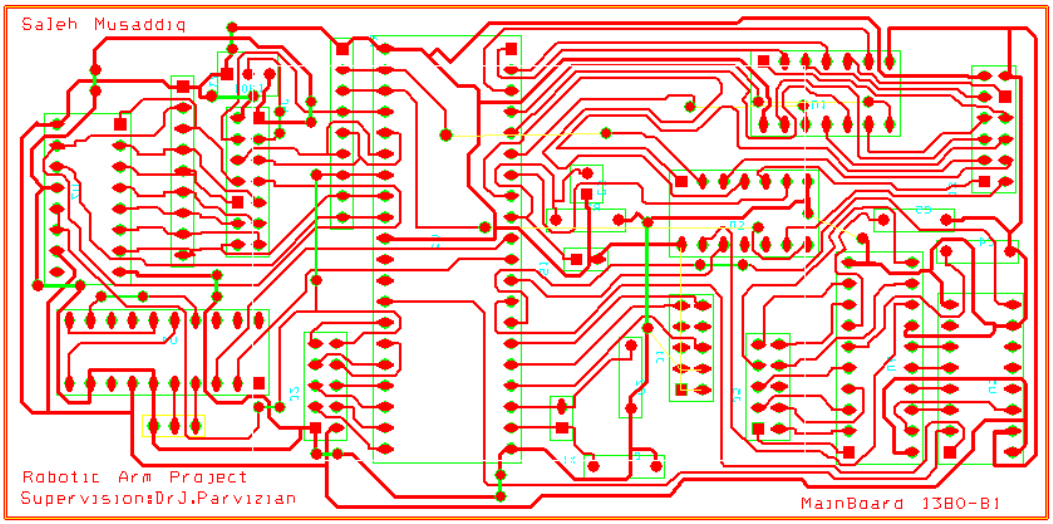


شکل (5-1) دیاگرام کلي بیانگر ارتباط بین بردهای مختلف

برد اصلي:
اين شماتيک مربوط به برد اصلي اين بازو است که ساير بردها به اين برد وصل مي شوند. قلب اين برد يک ميکروکنترلر 89c52 است که وظيفه کنترل بازو را به عهده دارد. اين برد را مي توان به يک مينيکامپيوتر تشبيه کرد که داراي قطعات اصلي زير است:
يک ميکروکنترلر، يک LCD چهار خطي ويک صفحه کليد(برد مربوط به صفحه کليد بعدا تشريح مي شود).
J4کانکتور 14 پايه مربوط به LCD است که اطلاعات به صورت 4 بيتي به آن وارد مي‌شود. براي صرفه‌جوئي در پايه هاي ميکروکنترلر، پايه هاي 11تا 14 که مربوط به ديتا هستند و پايه هاي rw وrs از طريق يک شيفت رجيستر سريال به موازي(74hc595) اطلاعات را بين LCD و ميکروکنترلرجابجا مي‌کنند. پايهen و پايه 14 که براي تست مشغول بودن LCD به کار مي رود به طور مستقيم به ميکروکنترلر وصل شده-اند.
ازطريق j1 ميکروکنترلر با برد مربوط به استپر ارتباط برقرار مي‌کند و آنرا با سرعت و جهت دلخواه به حرکت درمي‌آورد(اين استپر موتور صرفا براي حرکت دوراني بازو استفاده مي شود و با کمي تغيير در برنامه ميکروکنترلر، مي توان آنرا با يک موتور dc تعويض‌کرد).
کانکتور j3 ارتباط بين انکودر و ميکروکنترلر را برقرارمي‌کند. توجه شود که اينورترها بر روي برد اصلي قرار دارند نه بر روي خود هد.
آي سي sp233 براي تطبيق سطح سيگنالهاي TTL/CMOS وRS232 استفاده شده است.
آي سي MM74C923 ،دکودرصفحه کليد است که کليد فشرده شده را تشخيص مي‌دهدوکد مربوط به اين کليد ازطريق شيفت رجيستر 74hc165 به ميکروکنترلر فرستاده مي‌شود.
J5کانکتوري است که ارتباط بين برد صفحه کليد و برد اصلي را تامين مي‌کند.



شکل (2-5) شماتیک برد اصلی

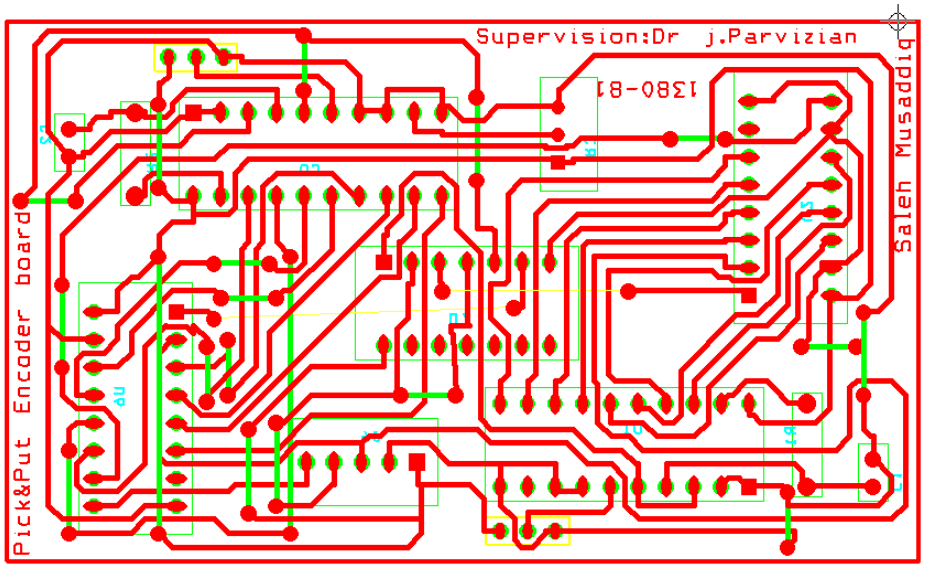


شکل (5-3) برد اصلی

برد مبدل های آنالوگ به دیجیتال :

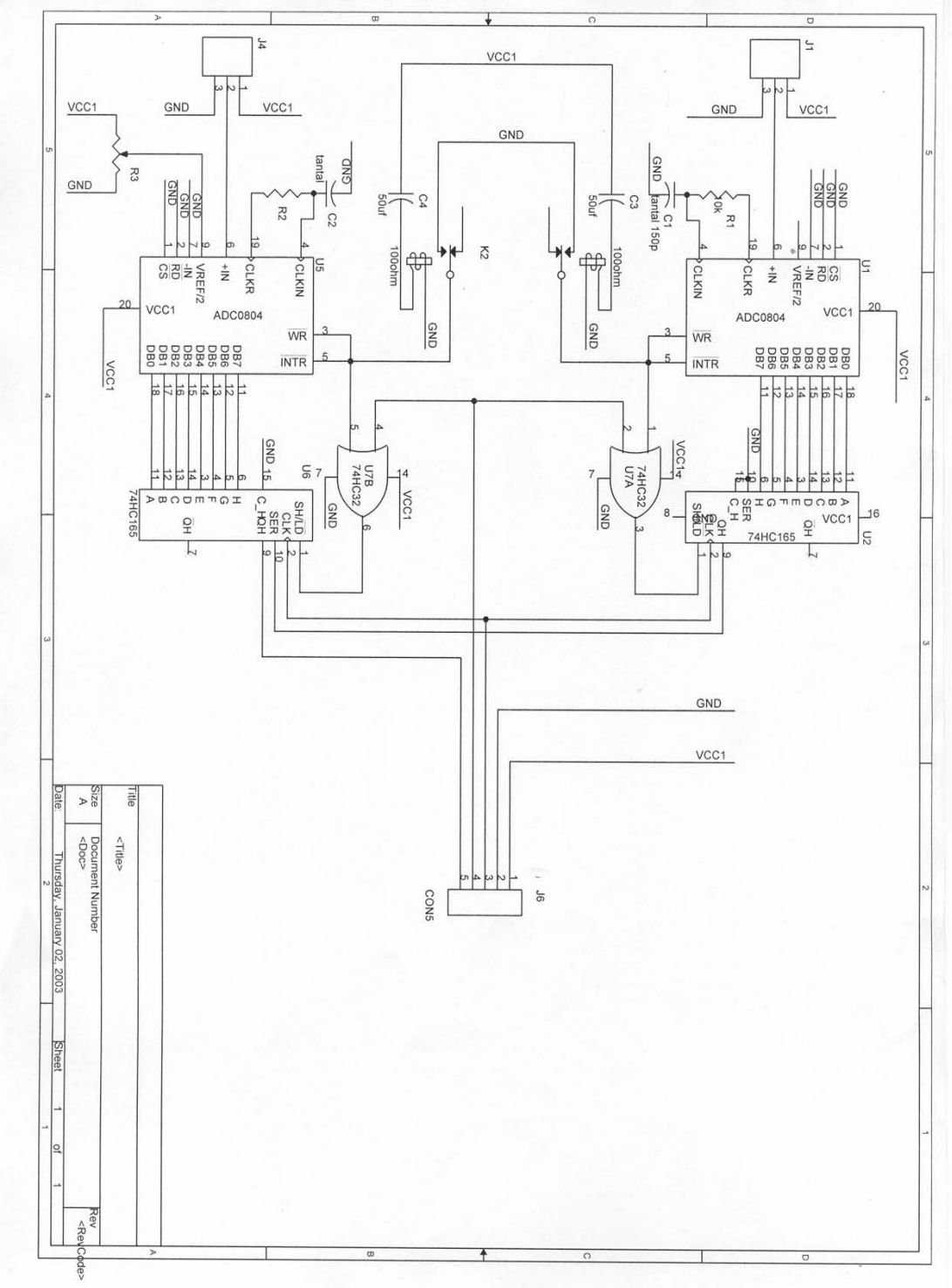
این برد اطلاعات مبدل های آنالوگ به دیجیتال را به صورت سریال به میکروکنترلرمي‌فرستد.

J1 وj2 به پتانسیومترها خطي دقیق وصل مي‌شوند. مبدل های آنالوگ به دیجیتال درمدآزاد بسته شده اند و با استفاده از ترکیب رله و یک خازن مناسب با روشن شدن ربات براي لحظه کوتاهي پایه هاي 3و5 به سطح پائین‌کشیده مي-شوندتاعملکردصحیح مبدل آنالوگ به دیجیتال‌تضمین شود.



شکل (5-4) برد مبدل های آنالوگ به دیجیتال

باپايين رفتن پايه s/h (پايه4 از j6)، اطلاعات موجود درخروجي مبدل‌هاي آنالوگ به ديجيتال به درون رجيسترها كشيده مي‌شود. حال اگر در همين لحظه كار تبديل تمام شده باشد و اطلاعات جديد در خروجي مبدل‌آنالوگ به ديجيتال قرارگيرد، ممكن است اطلاعات غلط وارد رجيسترشود. براي جلوگيري از اين هم زماني، از يک گيت or استفاده شده است.

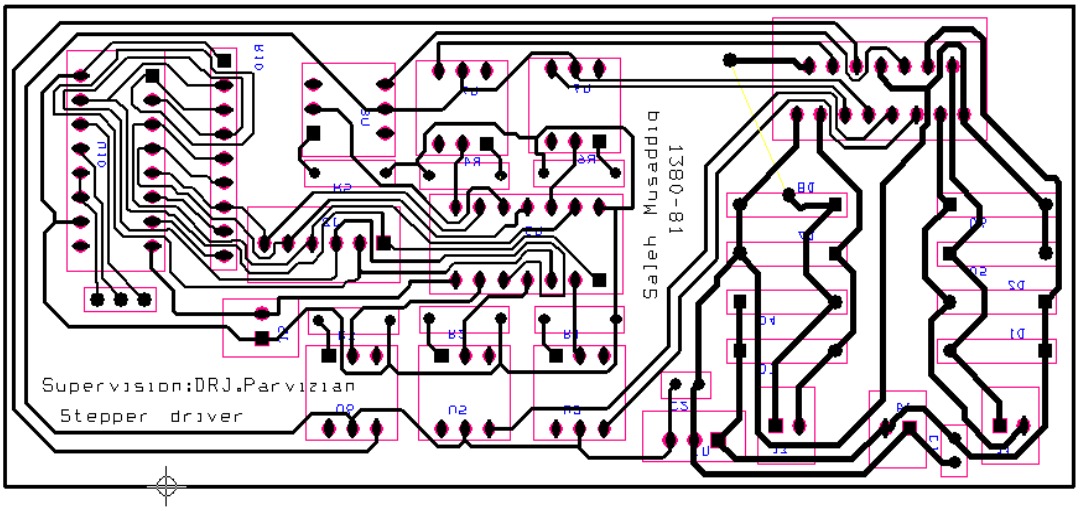


شکل (5-5) شماتیک برد مبدل‌های آنالوگ به دیجیتال

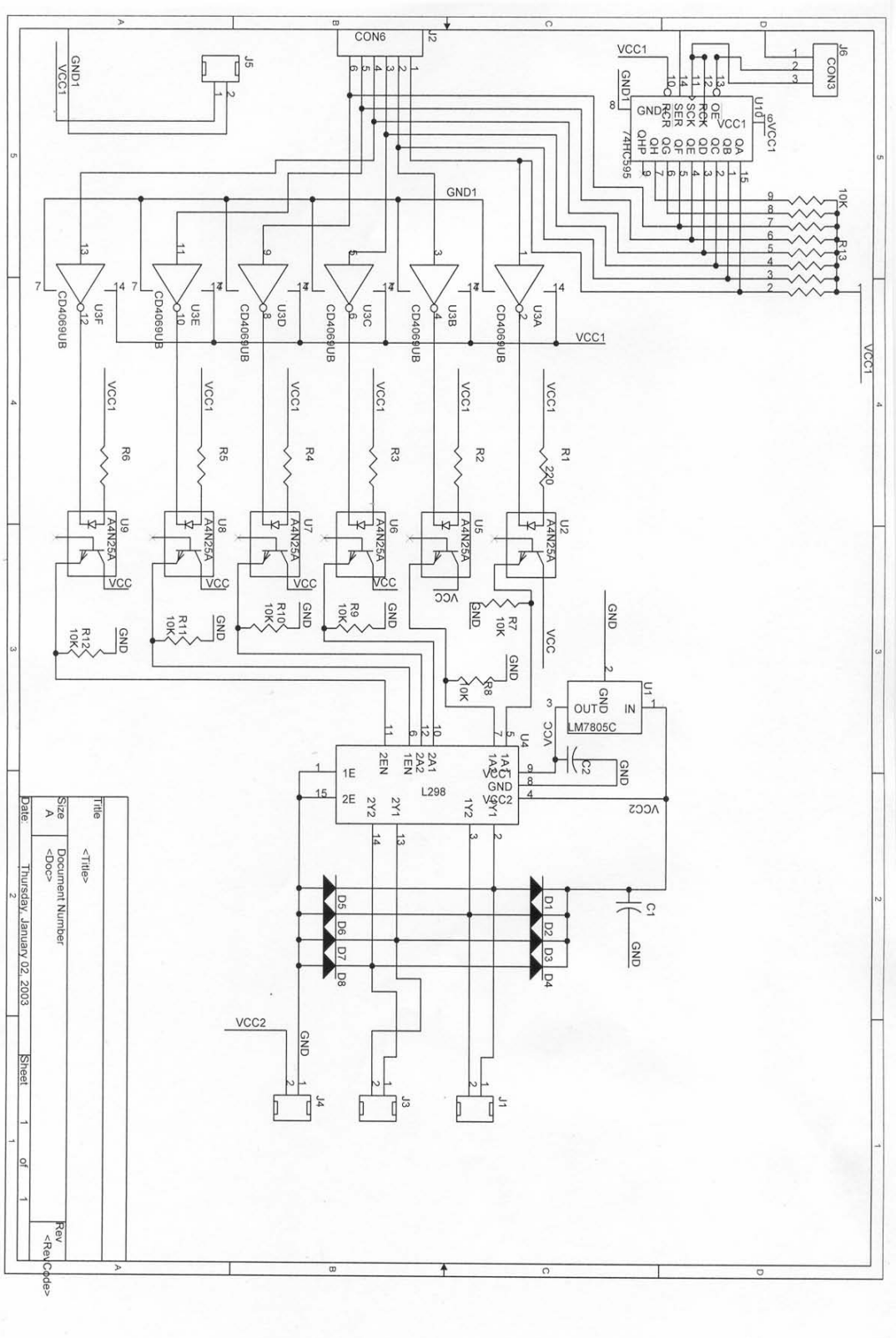
برد محرک استپر:

در این بازو داراي 2 سطح ولتاژ5 ولت و 12 ولت هستیم‌که ولتاژ اخیر براي تغذیه استپ موتور و2 موتور dc استفاده مي‌شود. براي پرهیز از اثرات مخرب اسپایک هاي حاصل از موتورها برروي مدارهاي فرمان ، میکرو و LCD ،

منبع تغذيه 12 ولت به طوركامل از منبع تغذيه 5 ولت جدا شده است و فرمان‌ها توسط اپتوكوپلرهاجابجا مي‌شوند.
J1 وj3 به 2كوئل استپر متصل مي‌شوند. j4 براي تغذيه 12 ولت است و j5 تغذيه 5 ولت را تامين مي‌كند.
سيگنال‌هاي فرمان از طريق j6 به صورت سريال وارد 74hc595 شده و به صورت موازي از اين آي سي خارج مي‌شوند، اين سيگنال‌ها پس‌ازعبورازاپتوكوپلرها باعث‌سوئيچ شدن پل-هاي L298 مي‌شوند .



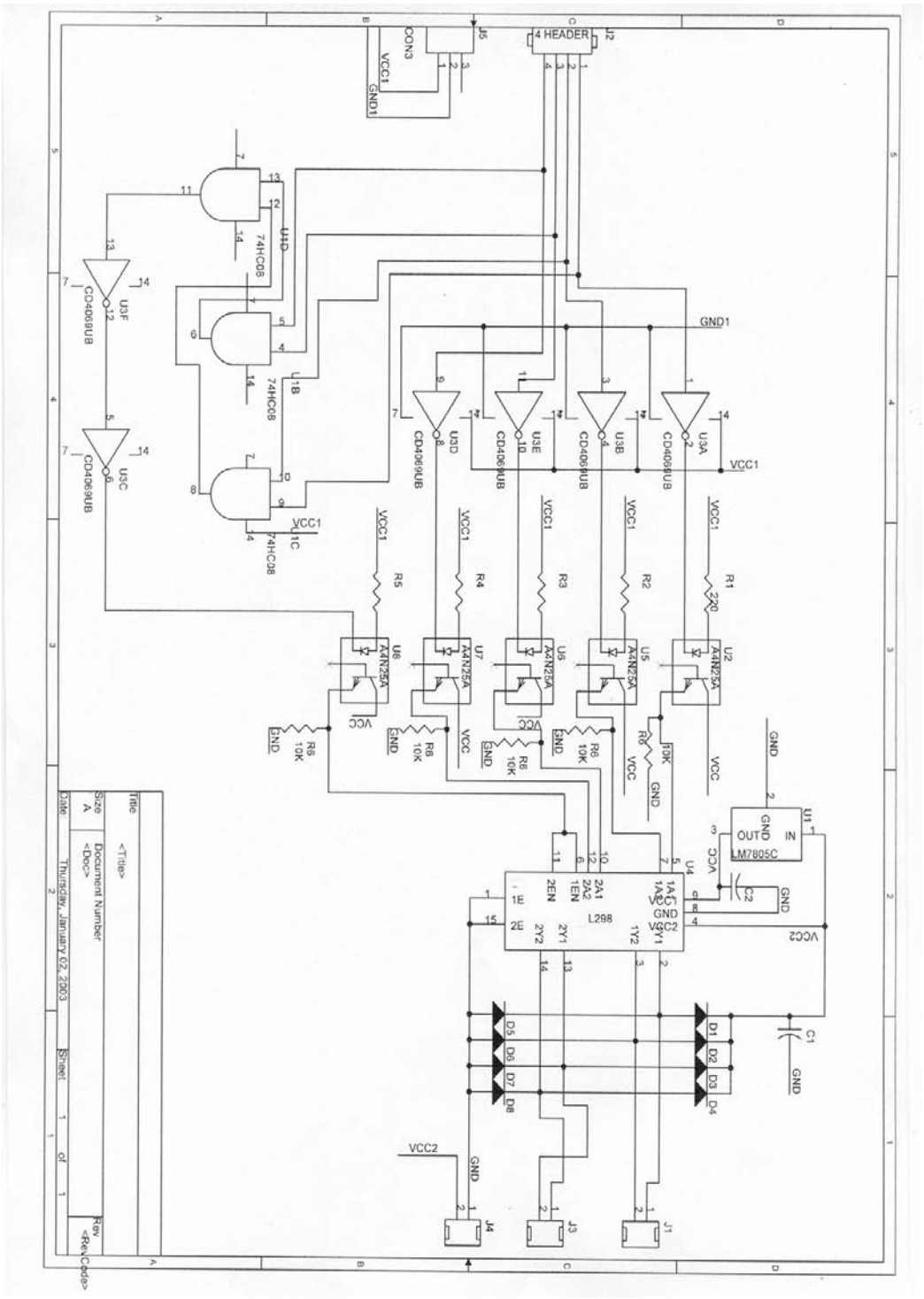
شكل (5-6) برد محرك استپر



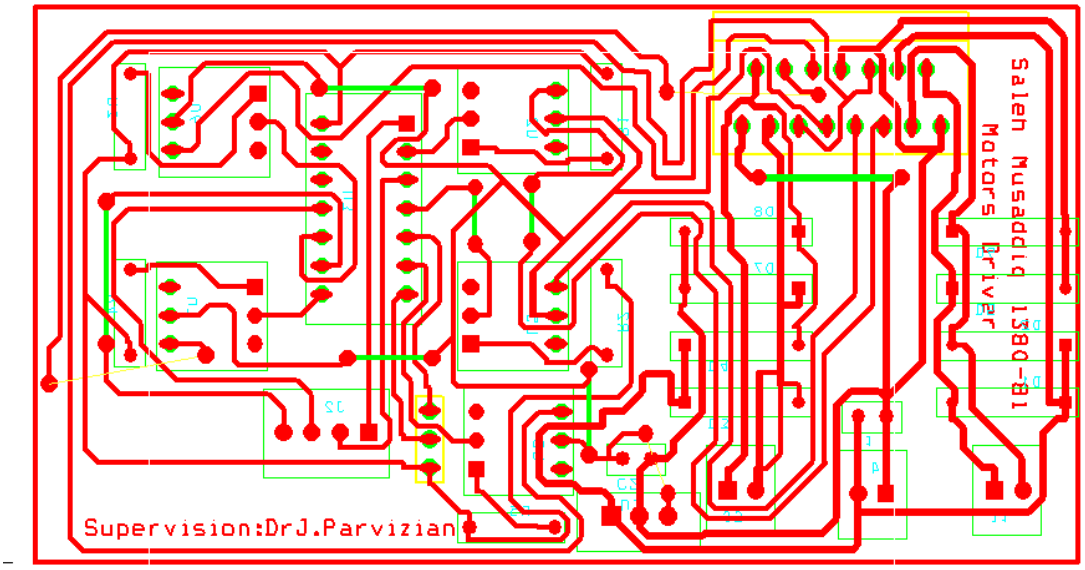
شكل (7-5) شماتيك برد محرك استپر

برد محرك موتورهاي dc:

این برد بعد از اپتوکوپلرهاي، مشابه برد محرک استپراست. ازآنجاکه براي آسيب نديدن آيسي L298 ، بايد قبل ازوصل کردن يا قطع کردن ولتاژ 12 ولت ، پايه en آیسی به سطح پايين آورده شود و اين پايه به طورمستقيم دراختيارميکرونيست، بنابراين از يک مجموعه گيت And وNot استفاده شده است. درنتيجه هرگاه هر 4 سيگنال ورودي در سطح بالا باشند، پايه en درسطح پايين قرارميگيرد.



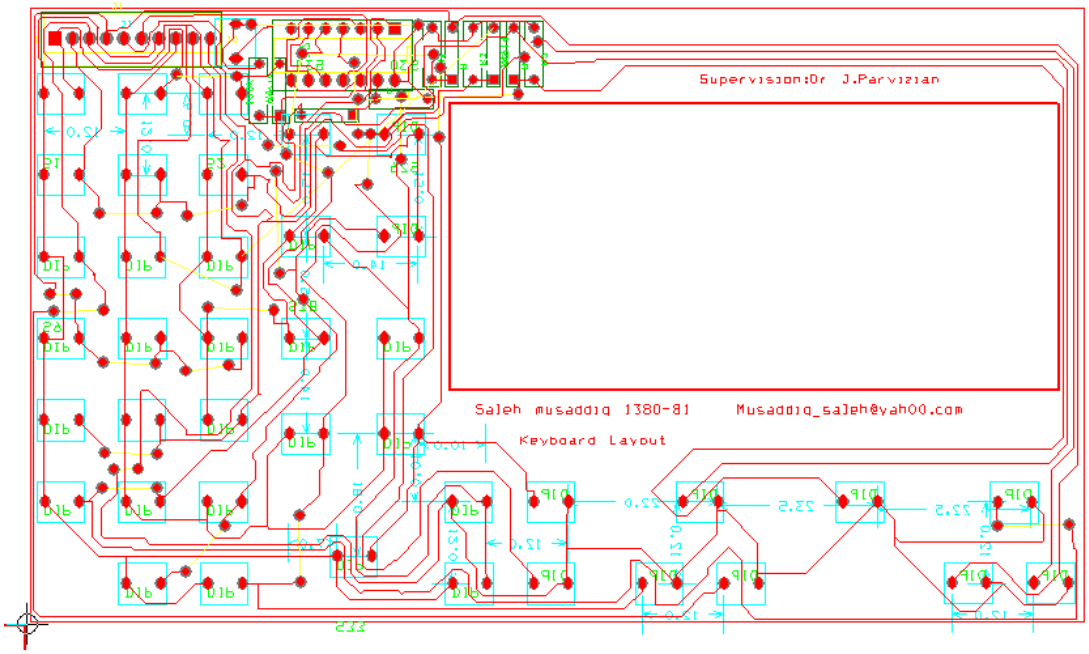
شكل (5-8) شماتيک برد محرک مو تورهاي dc



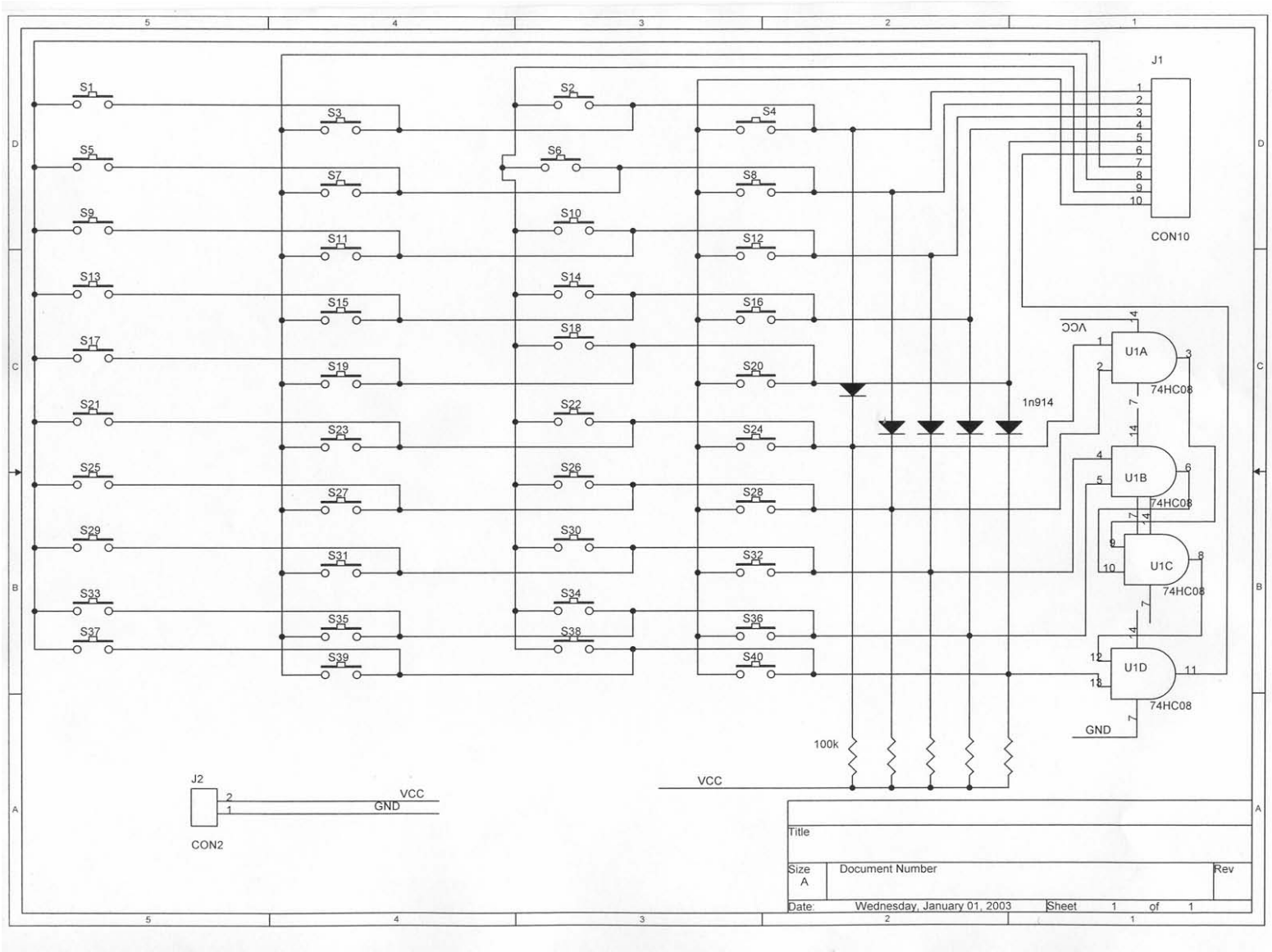
شكل (5-9) برد محرک مو تورهاي dc

برد صفحه كليد:

اين برد يک شبکه از چهل ميکروسوئيچ است. از آنجا که آيسي دکودر صفحه کليد درحالت معمولي داراي ورودي20 کليد است، براي افزايش ورودیها به 40 کليد از 4 گيت And استفاده شده است.



شكل (5-10) برد صفحه كليد

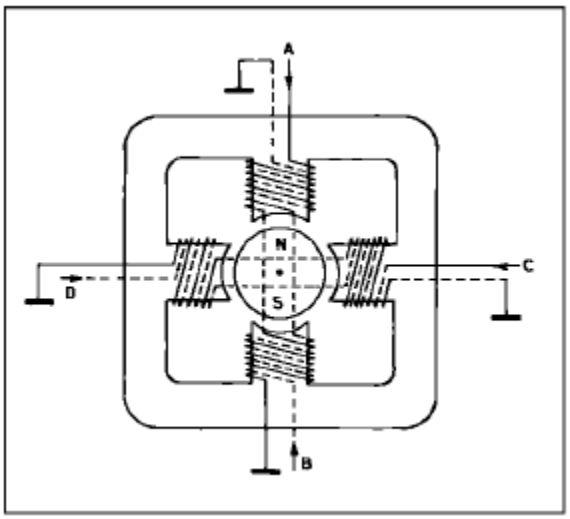

J1
CON10
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
S34
S35
S36
S37
S38
S39
S40
1n914
VCC
U1A
U1B
U1C
U1D
74HC08
GND
100k
VCC
J2
CON2
VCC
GND
Title
Size
A
Document Number
Rev
Date:
Wednesday, January 01, 2003
Sheet
1
of
1

شکل (5-11) شماتیک برد صفحه کلید

فصل ششم: برنامه نويسي ميكروكنترلر

دراين بخشكدهاي اسمبلي موجود در حافظه ميكروكنترلرتشريح ميشود.متن اين برنامه در شاخه \CD\Micro \ program قراردارد. اينكدها (تقريبا 3200 دستور) به70 تابع تقسيم ميشوندكه درادامه منبع[1] برنامه و عملكرد توابع آن آورده ميشود:

قبلازتشريح توابع مربوط به استپرلازم استتا مباني استپ موتورها تشريح شود. موتورهاي پلهاي بر2 نوع هستند: 1- موتورهاي 2 قطبي(bipolar): اين موتورها داراي 2 كوئل هستند و توسط تغيير جهت جريان با يك توالي مناسب در هركوئل، كنترل ميشوند (شكل 6-1).

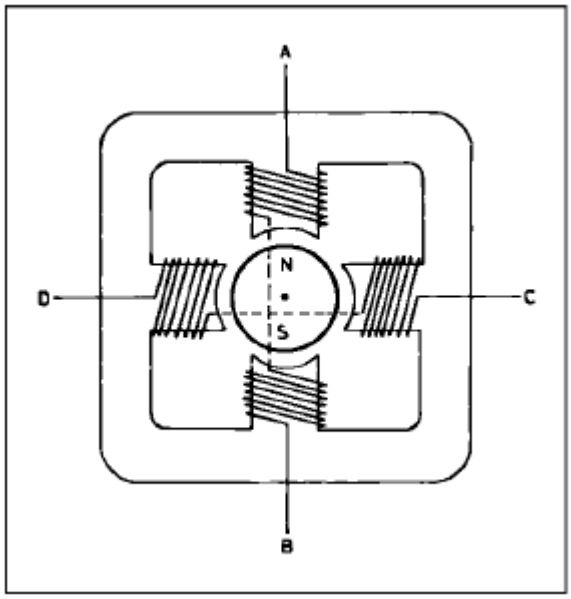


شكل (6-1) موتور2 قطبي

2 موتورهاي تك قطبي (unipolar): اساسكاراين موتورها، مانند موتورهاي 2 قطبي است با اين تفاوتكه سيم پيچ هركوئل آن به صورت زوج است و براي تغيير جهت جريان

[1] source

در هرکوئل، سيم پيچ عوض مي‌شود نه جهت جريان، درنتيجه اين موتورها، بزرگ و سنگين هستند، اما داراي اين مزيت هستند كه مدار فرمان آن ها ساده‌تراست (شكل6 2). امروزه باوجود آي‌سي‌هائي چون L298 مدار فرمان موتورهاي 2 قطبي ساده‌تر شده است.

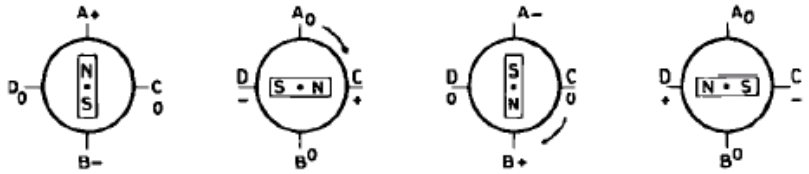


شكل (6-2) موتور تك قطبي

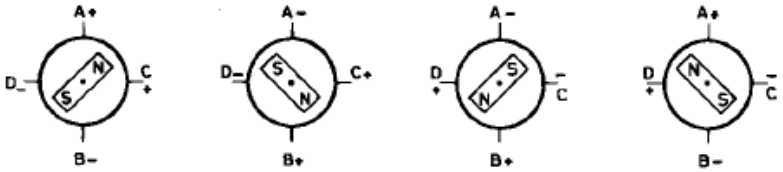
سه روش عمومي براي روشن‌كردن كوئل ها وجود دارد:
1ـ توالي2 فازي(Tow phase on drive): درهرلحظه2 كوئل را روشن مي‌كندكه بيشترين‌گشتاور را دارد و البته باعث تكانهاي شديد در موتور مي‌شود 2ـ توالي موجي(Wave drive): هر لحظه يك كوئل را روشن مي‌كند و حركت نرمي دارد وگشتاورآن كمتر ازتوالي2 فازي است 3ـ توالي نيم‌گام(Half step drive): اين توالي تركيب 2 توالي فوق است و باعث مي‌شود تا موتوردرنصف زاويه چرخش معمولي خود، گام بزند. گشتاوردراين توالي پيوسته كم و زياد مي‌شود . شكل6-3 اين توالي‌ها را نمايش مي دهد.

نرخ گام ها: موتورهاي پله‌اي، تجهيزاتي مكانيكي هستند بنابراين نرخي‌كه پالس‌هاي‌گام اعمال مي‌شوند، اهميت دارد و بايد به‌گونه‌اي باشد كه موتورگام قبلي را كامل-كرده باشد قبل ازآنكه پالس بعدي را دريافت كند. اگر نرخ گام زياد باشد، ممكن است موتور اصلا نچرخد، به جاي حركت كردن بلرزد، اشتباه بگردد (2تا چپ ، يكي راست و...) و يادر جهت مخالف بگردد.

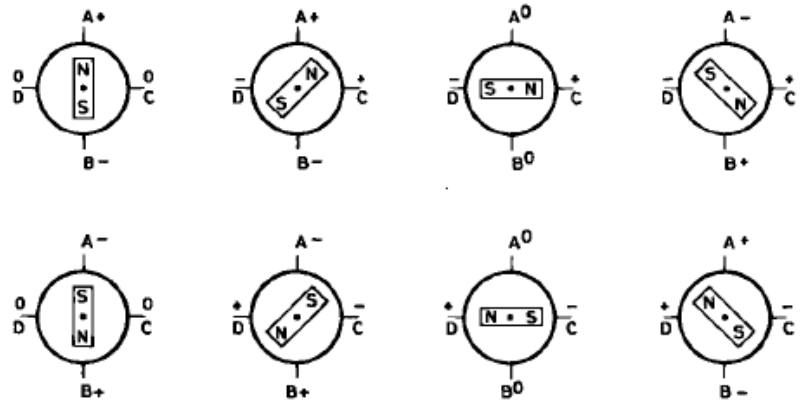
a : Wave drive (one phase on).



b : Two phase on drive.



c : Half step drive.



شکل (3-6) روشهای مختلف گام زدن موتور پله ای

حال به تشریح خطوط برنامه می پردازیم:

از خطوط 1 تا 181 راهنماهای پیش پردازنده قرار دارند. این تعاریف دسته بندي شده اند تا کار با آنها راحتتر باشد. به طور مثال در بخش"macros for encoder"بیت 32 از فضاي آدرس پذیر RAM براي نگهداري یک پرچم به نام"odd_or_even"تخصیص داده شده است. همانطور که بعدا در قسمت توابع مربوط به انکودرخواهیم دید این بیت مشخص-کننده این استکه موقعیت جاري انکودر زوج است یا فرد.

خط1
```
;********************** macros for lcd *******
en                      bit p0.4
rs                      bit acc.1
rw                      bit acc.0
```

```
serial                  bit p0.7        ;for 595
c_clock                 bit p2.4        ;common clock for lcd and ADCes
flag                    bit p0.6
holder                  equ 6ah
;************************** macros for encoder *********
gray_orginal_high       equ 21h
gray_orginal_low        equ 20h
current_high            equ 23h
current_low             equ 22h
current_encoder_high    equ 68h
current_encoder_low     equ 69h
bit_buffer_1            bit 30h
bit_buffer_2            bit 31h
odd_or_even             bit 32h
;************************************ macros for stepper *********
buffer_of_bits          data 24h
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
feed_step               equ 3fh         ;=>1,2,3,4,5,6,7,8,9,10,11  x 10
reduce_for_stop         bit 3ch ;if 1 then yes
;********************** macros for adc **********************
common_clock            bit p2.4
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
etb_direction_flag      equ 87 ;W                                   خط 181
;****************************** IRes **********************
```

در این قسمت، اینتراپت ها به توابع خود راهنمائي مي شوند.

```
org 0000                ; Power up reset vector
        jmp PowerUp
org 003h                ;INT0 int vector
    jmp intrupt0
org 000bh
    jmp Timer0_sir      ; Counter/ Timer 0 int vector
org 002bh
    jmp Timer2_ir       ; Counter/ Timer 2 int vector
org 0023h
        jmp serial_ir   ;serial intrupt vector
;****************************************
```

در اين قسمت مقدار دهي اوليه متغيرها صورت ميگيرد، همچنين نمايشگر راه اندازي شده واز كاربر براي رفتن به نقطه 0و0و0سوال مي شود.
توجه شود كه P3.2 اينتراپت خارجي صفر است كه پايين رفتن آن به معني اين است كه كليدي از صفحه كليد فشرده شده است.

```
org 30h
PowerUp:
initializing:
nop
nop
nop
nop
        setb releh_12_volt
        setb go_up
        setb go_down
        setb open_finger
        setb close_finger
        mov sequence_holder_value,#11011101b
call after_motion
        clr ea
        mov dptr,#seq_even_1
        mov last_seq_holder_high,dph
        mov last_seq_holder_low,dpl
        mov drive_sequence,#02h
        mov tmod,#00100001b                     ;timers mode:timer0(16 bit)
timer1 (8bit auto)
        setb TCON.0                             ;sensivity by blade
        setb p3.2                               ; for INT0
        clr free_or_quick_stop_p
        setb free_or_quick_stop_f
        mov feed_step,#60
        mov feed_mot,#6
        setb permit_change_feed
        mov start_prog_count,#start_prog_cach
        clr record_mode
        mov sp,#0cfh
        setb common_direction
        mov last_freq_holder,#0
        clr reduce_for_stop
        mov at_ret_of_sir,#00
        clr manual
        clr i_am_in_timer2_ir
;**************************************
call setup_lcd
call clear_lcd
call fill_cgram
        clr releh_12_volt
```

```
        mov dptr,#message01
        mov r7,#80h
call display_message
premain02:
call beeb_function
        jb p3.2,$
call get_keyboard
        mov a,keyboard_buffer
    cjne a,#key_ok,premain00
        mov r1,#81h
        mov @r1,#'L'
        inc r1
        mov @r1,#00
        inc r1
        mov @r1,#00
        inc r1
        mov @r1,#'U'
        inc r1
        mov @r1,#01
        inc r1
        mov @r1,#'G'
        inc r1
        mov @r1,#01
        inc r1
        mov @r1,#255
call clear_lcd
call display_frame
call display_main_screen
        mov r1,#80h
call run_program
        jmp premain01
premain00:
    cjne a,#key_cancel,premain02
premain01:
        anl IE,#11111100b     ;deactive keyboard intrupt and timer0
        setb p3.2
;*****************************************
```

خطوط 271 تا 283، چرخه اصلي است كه ميكرو در زمانهاي بيكاري درآن به سر ميبرد و دراين چرخه به بررسي وضعيت صفحه كليد و بازسازي اطلاعات نمايشگر مي‌پردازد.
در اين حلقه چنانچه كليدي از صفحه كليد فشرده شود، دو تابع "get_keyboard" و "main_function" به ترتيب صدا زده مي-شوند تا وظيفه مربوط به آن كليد را انجام دهند.

main_loop: خط 271

```
call refresh_adc_val_function
call clear_lcd
call display_frame
call delay_3
main:
        jnb p3.2,main_loop00
call display_main_screen
     jmp main
main_loop00:
        call get_keyboard
call main_function
        jmp main_loop                                   خط 283
;*********************************
```

از اين قسمت تا انتهاي برنامه، توابع، اينتراپتها ومقاديرجدولي قرار دارند.

تابع زيركليد فشرده شده راتعبيير كرده و عمليات مربوط به آن را انجام مي دهد. قبل از صدا زدن اين تابع، تابع" get_keyboard" صدا زده مي شودكه پس از برگشت از اين تابع كدكليد فشرده شده در آدرس" keyboard_buffer" قرار ميگيرد.

به طور مثال در ابتداي تابع بررسي مي شود كه آيا كليد فشرده شده كليد"باز شدن انگشت" است يا نه، اگر جواب مثبت باشد توابع لازم براي باز شدن انگشت را صدا مي زند. اگر جواب منفي باشد، كليد فشرده شده را با كدكليد بعدي مقايسه مي‌كند واين عمل را ادامه مي دهد تاكليد مربوطه را پيدا كند.

```
main_function:
        mov a,keyboard_buffer
        cjne a,#key_finger_open,main_loop_00
        setb pick_or_put
        mov disired_adc_finger,#255
        setb manual
        orl IE,#10100000b
call pick_at_xx
        anl IE,#11011111b                       ;deactive timer2
        clr manual
ret
main_loop_00:
        cjne a,#key_finger_close,main_loop_01
        setb pick_or_put
        mov disired_adc_finger,#000
        setb manual
        orl IE,#10100000b
call pick_at_xx
        anl IE,#11011111b                       ;deactive timer2
        clr manual
```

```
ret
main_loop_01:
        cjne a,#key_put_up,main_loop_02
        clr pick_or_put
        mov disired_adc_up_down,#000
        setb manual
        orl IE,#10100000b
call put_at_xx
        anl IE,#11011111b                       ;deactive timer2
        clr manual
ret
main_loop_02:
        cjne a,#key_put_down,main_loop_03
        clr pick_or_put
        mov disired_adc_up_down,#127
        setb manual
        orl IE,#10100000b
call put_at_xx
        anl IE,#11011111b                       ;deactive timer2
        clr manual
ret
main_loop_03:
        cjne a,#key_arm_rotate_ccw,main_loop_04
        clr direction_flag
call common_key_arm_rotate_function
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        setb state_of_transfer
        mov a,start_prog_count
        cjne a,#80h,main_loop_1502
        mov r1,#80h
        mov @r1,#255
main_loop_1502:
call key_connection_function
        mov IP,#00000000b           ;default
ret
;*********************************
```

تابع "key_connection_function" :

اين تابعي است كه باوصل شدن به PC، پيوسته حلقه هاي داخلي خود را اجرا مي‌كند تا ارتباط ميكرو بانرم افزاركامپيوتري حفظ شود.

```
key_connection_function:
call get_bytes_from_adces
        mov r0,#first_matrix_1
. . . . . .
```

```
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        mov at_ret_of_sir,#00
        jmp key_connection_function
;*********************************
common_key_arm_rotate_function:
        clr TR0
        orl IE,#10000010b
        clr end_table
        mov last_freq_holder,#0
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        jnb end_table,key_arm_rotate_f02
        clr TR0
        clr reduce_for_stop
        clr TF0
        anl IE,#01111101b                       ;deactive intrupt timer0
        setb beeb
call after_motion
ret
;*********************************
```

با فشردن کلید "Menu" این تابع اجرا شده ومنوئي را بر روي LCD نمايش مي دهد ودر پايان انتخاب کاربر را برمي‌گرداند.

```
key_menu_function:
        jnb record_mode,key_menu_00
        mov dptr,#message24
        mov r7,#80h
call display_message
call delay_3
call delay_3
ret
key_menu_00:
        mov menu_item,#0
call display_menu
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
call display_menu
```

```
        jmp key_menu_0
;*********************************
```
اين‌تابع با فشردن كليد Load اجرا مي‌شود و به ترتيب برنامه هاي موجود در RAM را نمايش مي‌دهد تا كاربريكي ازآنها را براي اجرا انتخاب كند.

```
key_load_function:
        jnb record_mode,key_load_000
        mov dptr,#message24
        mov r7,#80h
call display_message
call delay_3
.........
.........
.........
.........
.........
.........
.........
        call beeb_function
        jmp key_load_3
;*********************************
```
با فشردن كليد" Start" اين تابع اجرا شده‌كه درآن پس از پرسيدن نام برنامه، ميكرو وارد مد ثبت برنامه مي-شود.

```
key_rec_start_function:
        jb record_mode,key_rec_start_end
call verify_function
key_rec_start_end:
ret
;*********************************
verify_function:
verify_00:
        mov dptr,#message22
        mov r7,#80h
call display_message
.........
.........
.........
.........
.........
.........
.........
.........
.........
ret
verify_08:
```

```
call beeb_function
        jmp verify_02
jmp_verify_00:
        jmp verify_00
;*********************************
```

اين تابع وقتي اجرا مي شودكه‌كليد" Rec" بر روي صفحه كليد را فشار دهيم. در اين هنگام چنانچه ميكرودرمدثبت باشد، منوئي را نمايش مي‌دهدكه درپاسخ به آن بايد يكي ازكدهاي G يا U يا L را وارد كنيم. با اين عمل موقعيت جاري مربوط به اين‌كد به انتهاي برنامه چسبانده مي‌شود.

```
key_rec_record_function:
        jnb record_mode,key_rec_record_00
call clear_lcd
        mov dptr,#message23
.........
.........
.........
.........
.........
.........
.........
.........
call key_rec_end_function
end_function:
ret
;*********************************
```

با فشردن كليد" End" اين تابع اجرا شده و ميكرو با ذخيره برنامه ازمدثبت خارج مي‌شود.

```
key_rec_end_function:
        clr record_mode
        mov start_prog_count,prog_count
        mov r1,prog_count
        mov @r1,#255
        mov dptr,#message25
        mov r7,#80h
call display_message
call delay_3
call delay_3
ret
;*********************************
```

اين تابع پس از انتخاب يک آيتم ازمنوصدا زده مي‌شود و باعث مي‌شود تا عمليات مربوط به‌آن انتخاب انجام شود.

```
run_item:
        mov a,menu_item
        cjne a,#1,run_item_00
        mov dptr,#message04
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
call delay_2
ret
run_item02:
        cjne a,#key_cancel,run_item01
jmp run_item07
;*************************************************
```

اين‌تابع راهنگامي‌صدا مي‌زنيم‌كه بخواهيم نامي براي برنامه‌اي انتخاب‌كنيم.

```
enter_name:
        mov dptr,#message12
        mov r7,#80h
call display_message
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
enter_name08:
call beeb_function
        jmp enter_name06
;*************************************************
```

اين‌تابع درهنگام نياز به خواندن يك‌كاراكتراز صفحه-كليد صدا زده مي‌شود.

```
get_alphabet:
        clr TR0
        clr TF0
        mov alphabet_buffer,#55
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
get_alphabet18:
```

```
call beeb_function
        clr TR0
        clr TF0
jmp get_alphabet01
;*****************************************************
```

در هنگام وارد کردن برنامه از طریق‌صفحه کلید، این‌تابع-کلیه عملیات لازم برای‌آن‌که یک‌کد صحیح وارد شود را انجام می دهد.

```
add_action:
call delay_2
call delay_2
        jb p3.2,$
call get_keyboard
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        mov @r1,a
        inc prog_count
        mov r1,prog_count
        mov @r1,0
        inc prog_count
ret
;*****************************************************
```

ازآنجا که زاویه چرخش بین 0 تا 359/5، حرکت عمودي بین 0 تا 127 وحرکت انگشت بین 0 تا 255 تغییر مي‌کند، براي آنکه عدد وارد شده توسط کاربراز این رنج ها خارج نباشد، توابع"get_digit_1" ،"get_digit_2" و"get_digit_3" صدا زده مي-شوند. وظیفه این‌توابع این استکه بسته به‌آن‌که چه‌کدي در حال وارد شدن است، صحت آنرا چک کنند.

```
get_digit_1:
        jb p3.2,$
call get_keyboard
        mov a,keyboard_buffer
        cjne a,#key_delete,get_digit_101
        mov a,#00010000b                        ;mov curser left
call write_2_nibbles_command
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
```

```
. . . . . .
. . . . . .
. . . . . .
        orl a,#30h
call write_2_nibbles_data
ret
;*****************************************************************
get_digit_2:
        jb p3.2,$
call get_keyboard
        mov a,keyboard_buffer
        cjne a,#key_delete,get_digit_201
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
call beeb_function
        jmp get_digit_2
get_digit_216:
        mov buffer_byte_2,r0
        mov a,r0
        orl a,#30h
call write_2_nibbles_data
ret
;*****************************************************************
get_digit_3:
        jb p3.2,$
call get_keyboard
        mov a,keyboard_buffer
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
get_digit_315:
        mov buffer_byte_3,r0
        mov a,r0
        orl a,#30h
call write_2_nibbles_data
ret
;*****************************************************************
get_digit_4:
        jb p3.2,$
call get_keyboard
```

```
        mov a,keyboard_buffer
.........
.........
.........
.........
.........
.........
.........
call beeb_function
jmp get_digit_4
;*****************************************************
  عدد وارد شده از طريق صفحه كليد به فرم اسكي است.
                  اين تابع آنرا به فرم صحيح برمي‌گرداند.

convert_to_binary:
        mov a,buffer_byte_2
        mov b,#10
        mul ab
.........
.........
.........
.........
.........
.........
.........
        mov a,b
        rlc a
ret
e_k_s_not_1:
        mov a,b
ret
;*****************************************************

        با اجراي اين‌تابع، يک برنامه ذخيره شده درحافظه
                                                  اجرا مي‌شود.
run_program:
        mov prog_count,r1
run_program01:
        inc r1
        mov a,@r1
        cjne a,#'U',run_program02
        inc r1
        mov disired_adc_up_down,@r1
        orl IE,#10100001b
        push 1
call put_at_xx
        pop 1
        anl IE,#11011110b                    ;deactive keyboard intrupt and
timer2
```

```
        jmp run_program01
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
call delay_3
call delay_3
ret
;*************************************************************************
get_key:
        jb p3.2,$
call get_keyboard
        mov a,keyboard_buffer
ret
;*************************************************************************
beeb_function:
        clr beeb
call delay_2
        cpl beeb
call delay_2
        cpl beeb
call delay_2
        setb beeb
ret
;*******************************************
display_message:
        mov a,r7
call write_2_nibbles_command
        mov r4,#20
        mov r5,#1
display_message01:
        clr a
        movc a,@a+dptr
        jnz display_message02
ret
display_message02:
call write_2_nibbles_data
        djnz r4,display_message03
        mov r4,#20
        cjne r5,#1,display_message04
        inc r5
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
```

```
. . . . . .
        inc dptr
        jmp display_message01
;*****************************************
display_item:
        mov a,r7        ;set DDRAM
call write_2_nibbles_command
display_item01:
        clr a
        movc a,@a+dptr
        jnz display_item02
ret
display_item02:
call write_2_nibbles_data
        inc dptr
        jmp display_item01
;*****************************************
display_main_screen:
        mov a,#28h
        call write_2_nibbles_command
        mov a,#0fh
        call write_2_nibbles_command
        mov a,#06h
        call write_2_nibbles_command
call encoder
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        clr c
        rrc a
        mov r0,a
        mov a,current_encoder_low
        rrc a
        mov r1,a
        mov odd_or_even,c     ;if odd_or_even=1 then number is odd
call bin_to_bcd
call display_on_lcd
ret
;*************************************
display_frame:
        mov a,#80h
call write_2_nibbles_command
        mov a,#7ch
call write_2_nibbles_data
. . . . . .
. . . . . .
. . . . . .
```

```
. . . . . .
. . . . . .
. . . . . .
call display_row
ret
;*****************************************
display_row:
        mov r4,#3
display_frame01:
        mov r5,#5
display_frame02:
        mov a,#00010100b
call write_2_nibbles_command
        djnz r5,display_frame02
        mov a,#7ch
call write_2_nibbles_data
        djnz r4,display_frame01
ret
;*****************************************
display_menu:
call clear_lcd
        mov dptr,#message07
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        mov r7,#91h
call display_item
ret
;*****************************************
```

در هنگام اتصال به کامپیوتربا صدازدن این تابع، "نرخ تبادل اطلاعات از راه پرت سریال" پرسیده مي‌شود.

```
display_menu_baud:
call clear_lcd
        mov dptr,#message27
        mov r7,#80h
call display_item
        mov dptr,#message28
        mov r7,#0cch
call display_item
        mov dptr,#message29
        mov r7,#0a0h
. . . . . .
. . . . . .
. . . . . .
. . . . . .
```

```
. . . . . .
. . . . . .
. . . . . .
call display_item
        mov dptr,#message10
        mov r7,#0d1h
call display_item
ret
;*****************************************
display_program:
        mov a,#0c3h
call write_2_nibbles_command
        mov r0,#80h
        mov r5,#15
display_program00:
        mov a,@r0
        push 0
call write_2_nibbles_data
        pop 0
        inc r0
        djnz r5,display_program00
ret
;*************************************************
start_alphabet: db 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q'
                db 'r','s','t','u','v','w','x','y','z'
                db 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q'
                db 'R','S','T','U','V','W','X','Y','Z'
                db '0','1','2','3','4','5','6','7','8','9'
;*************************************************
```

اين تابع تنها يكبار در ابتداي برنامه، براي پيكربندي LCD در مد4 بيتي صدازده مي‌شود.

```
setup_lcd:
        mov a,#38h
call put_for_lcd
        setb en
        clr  en
call delay_lcd
        mov a,#38h
call put_for_lcd
        setb en
        clr  en
call delay_lcd
        mov a,#38h
call put_for_lcd
. . . . . .
. . . . . .
. . . . . .
```

```
. . . . . .
. . . . . .
. . . . . .
call write_2_nibbles_command
ret
;************************************************
reload_lcd:
        mov a,#28h
call write_2_nibbles_command
        mov a,#0dh
call write_2_nibbles_command
ret
;************************************************
clear_lcd:
        mov a,#01h
        call write_2_nibbles_command
ret
;************************************************
```

توابع "write_2_nibbles_command" و"write_2_nibbles_data" :

این دو تابع براي فرستادن دستورات و اطلاعات به LCD صدا زده مي شوندوچونLCD در مد 4 بيتي پيكربندي شده است، ابتدا نيبل پايين و سپس نيبل بالا فرستاده مي شود.

```
write_2_nibbles_command:
        mov holder,a
        clr rs
        clr rw
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        xchd a,@r0
call put_for_lcd
        setb en
        clr  en
call delay_lcd
ret
;************************************************
put_for_lcd:
        mov r7,#9
next_bit:
        rlc a
        mov serial,c
        clr  c_clock
        setb c_clock
        djnz r7,next_bit
        mov c,acc.7
        mov flag,c
```

```
ret
;****************************************
fill_cgram:
        mov a,#40h
call write_2_nibbles_command
        mov dptr,#cgram_data1
fill_cgram01:
        clr a
        movc a,@a+dptr
        cjne a,#99h,fill_cgram02
ret
fill_cgram02:
call write_2_nibbles_data
        inc dptr
        jmp fill_cgram01
;****************************************
delay_lcd:
        mov r7,#03h
delay_lcd1:
        mov r6,#0ffh
        djnz r6,$
        djnz r7,delay_lcd1
ret
;****************************************
cgram_data1:
font6: db 1fh,1fh,03h,03h,03h,1fh,1fh,00h
. . . . . .
. . . . . .
font1: db 00001100b,00001100b,00000000b,00011100b,00010100b,00011100b,00h,00h
        db 99h
;************************************************
message01: db 'ready for test?',0
. . . . . .
. . . . . .
message30: db ' 19200 ',0
;****************************************
```

تابع encoder :

وظيفه اين تابع اين استكه خروجي10بيتي انكودررا بخواند وآنرا در بايت آدرسپذير"gray_orginal_low" ودوبيتكم ارزش بايت "gray_orginal_high" ذخيره كند. سپس2 تابع "gray_to_decimal" و"subb_152" را به ترتيب صدا بزند تا تابع اولي كد فوق را از فرم خاكستری به فرم باينري تبديل كند (صفحه 11 را ببينيد) وتابع بعدي مقدار 152 را ازاين فرم باينريكم كند تا موقعيت فعلي بازو نسبت به نقطه صفر بدست آيد. پس ازآن اينتراپت را ازكار ميانداز وموقعيت فعلي را در بايت هاي "current_low" و"current_high" ذخيره مي كند وسپس اينتراپت را فعال ميكند. به اين دليل

اینتراپت را از کارمي اندازدکه ممکن است پس از ذخیره بایت پایین موقعیت، اینتراپت رخ دهد .حال اگر این اینتراپت از مو قعیتفعلي براي انجام عملي استفاده کند باعث ایجاد خطا میشود. در ادامه موقعیت فعلي در جاي مناسب بر روي LCD نمایش داده میشود.

2 تابع"bin_to_bcd" و"display_on_lcd" براي نمایش موقعیت جاري بر روي LCD صدا زده میشوندکه در جاي خود توضیح داده خواهندشد.

```
;****************************************
encoder:
        mov p1,#0ffh
        orl p0,#00000011b
        mov gray_orginal_low,90h
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
encoder00:
ret
;****************************************
gray_to_decimal:
        mov c,09h
        mov 19h,c
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        mov bit_buffer_1,c
call compar
        mov 10h,c
ret
;****************************
compar:
        setb c
        jb bit_buffer_1,a_is_1
        jb bit_buffer_2,end_compar
        clr c
ret
a_is_1:
        jnb bit_buffer_2,end_compar
        clr c
end_compar:
ret
;****************************
```

```
subb_152:
        anl current_high,#00000011b
        clr c
        mov a,current_low
        subb a,#152d
        mov current_low,a
        mov a,current_high
        subb a,#00h
        mov current_high,a
ret
;*****************************
bin_to_bcd:
        mov a,r1                        ;input r0r1
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        addc a,#02h
        da a
        mov r2,a
        jmp loop
;***********************************
```

: "display_on_lcd"

"write_2_nibbles_data" 2 تابع پركاربرد "write_2_nibbles_command" و استفاده كرده وموقعيت فعلي انكودر را نمايش مي‌دهد. عملكرد اين دو تابع واضح است وتنها ذكر اين نكته لازم است كه LCD درمد4بيتي اطلاعات را دريافت مي كند.

توابع "put_for_lcd" و "SR_for_stepper":

اين دو تابع با زمانبندي مناسب يک بايت را به رجيستر "74hc595" مربوطLCD يا استپر منتقل مي‌كند.

```
;***********************************

display_on_lcd:
        mov a,#0dbh                     ;set DDRAM
call write_2_nibbles_command
        mov a,r2
        anl a,#0fh
        orl a,#30h
call write_2_nibbles_data
        mov a,r3
        swap a
        anl a,#0fh
        orl a,#30h
call write_2_nibbles_data
        mov a,r3
        anl a,#0fh
```

```
        orl a,#30h
call write_2_nibbles_data
        mov a,#2eh
call write_2_nibbles_data
        jnb odd_or_even,_is_even
        mov a,#35h
call write_2_nibbles_data
ret
_is_even:
        mov a,#30h
call write_2_nibbles_data
ret
;*********************************
dec_dptr:
      mov r2,dph
      mov r3,dpl
      mov r4,#00h
      mov r5,#01h
call subb16
      mov dph,r2
      mov dpl,r3
ret
;***********************************************
```

تابع "go_to_xx":

در برنامه نوشته شده براي اين بازو براي آن كه بازو از موقعيت فعلي به موقعيت مطلوب يك حركت دوراني را انجام دهد، بايد ابتدا موقعيت مطلوب را در آدرس هاي"disired_low" و"disired_high" قرار داد (به صورت عددي بين 0 تا 719 كه معادل عددهاي 0 تا 359/5 هستند) و سپس تابع" go_to_xx" را صدا زد. پس از برگشت از اين تابع بازودرموقعيت مطلوب قرار داد. اين تابع، 9 تابع ديگررا براي تكميل عمليات خود صدا مي زند. در اين تابع ابتدا موقعيت جاري و مطلوب باهم مقايسه مي‌شوند، تا اگر در حال حاضر در موقعيت مطلوب هستيم از اين تابع خارج شود در غير اينصورت تابع "direction_function" را صدا مي زندكه دراين تابع جهت حركت بهينه بازو براي رسيدن به نقطه مطلوب مشخص مي‌شود. پس از آن اصلاحي بر روي نقطه مطلوب صورت مي گيرد (تحليل 1).

تحليل1 : براي‌آن كه بازو در نقطه مطلوب متوقف شود همواره نقطه قبل از نقطه مطلوب را در نظر گرفته و به محض‌عبور بازو از اين نقطه استپ موتور را متوقف مي‌كنيم. در نتيجه، توقف در نقطه مطلوب صورت مي‌گيرد. براي به دست آوردن يك نقطه قبل از نقطه مطلوب(آدرس هاي"modify_disired_high" و"modify_disired_low")، چنانچه حركت در جهت

ساعتگرد باشد، يك واحد از نقطه مطلوب كم مي‌كنيم و در غير اينصورت يك واحد به آن اضافه مي‌كنيم.
در حالت ساعتگرد، چنانچه نقطه مطلوب صفر است، كم كردن يك واحد ازآن باعث توليد يك عدد منفي و در نتيجه توليد خطا مي شود حال آن‌كه نقطه قبل ازصفرنقطه 719 (يا 359/5) است كه بايد اصلاح لازم را انجام داد.
در حالت پادساعتگرد مشكل فوق براي نقطه 719 است كه با اضافه كردن يك واحد به آن به 720 تبديل مي شود كه چنين نقطه اي وجود ندارد و بايد آن را به صفر تغيير داد.
در ادامه بازه حركت بدست مي آيد و بسته به اين كه اين فاصله بزرگتر از 30 (معادل 15 درجه) يا كوچكتر از آن باشدكدهاي مختلفي اجرا مي شود.
چنانچه فاصله تا مقصدكوچكتر از 15 باشد نيازي به شتابگيري وكاهش شتاب نيست و مي توان با پالس‌هاي‌آهسته‌اي اين فاصله را پيمود. اين تابع براي اين منظور اينتراپت را ازكار مي اندازد و وارد حلقه‌اي مي‌شودكه اين حلقه توسط تايمر به فاصله زماني منظم تكرار مي شود و چون باهربار اجراي اين حلقه استپر يك گام جابجا مي‌شود بنابراين سرعت تكرارحلقه به‌گونه‌اي تنظيم مي‌شودكه اين جابجايي‌آهسته باشد. باهربار اجراي حلقه توابع" call encoder" و"pulse_sequence" و"SR_for_stepper" و"where_i_am" به ترتيب اجرا مي-شوند. درتابع‌آخري مشخص مي‌شودكه‌آيا به نقطه مطلوب رسيده‌ايم يا نه. پس ازخروج از حلقه تابع" after_motion" براي قطع جريان استپرصدا زده مي شود.
اگر فاصله تا مقصد بيش از 15درجه باشد، كد پيچيده تري بايد اجرا شود به اين صورت كه اين فاصله به سه قسمت تقسيم مي شود و بازو قسمت اول را با سرعت شتابدار، قسمت دوم را با سرعت ثابت و قسمت سوم را باسرعت كاهنده مي پيمايد تا به نقطه مطلوب برسد. توابع"beta_landa_1" و "beta_landa_2"براي پيدا كردن 2 نقطه اي كه بازو را به سه قسمت مي كنند به كار مي روند. ازاينتراپت تايمرصفر براي زمانبندي حركت هاي شتابدار و يكنواخت باستفاده از يك جدول فركانس استفاده مي شود (table).

```
;***********************************************

go_to_xx:
        mov a,disired_high
        cjne a,current_encoder_high,go_to_xx00
        mov a,disired_low
        cjne a,current_encoder_low,go_to_xx00
ret
go_to_xx00:
```

```
call direction_function
        anl tmod,#11110001b          ;changing of timer0 mode to 16bit(mode 1)
        orl tmod,#00000001b
        jnb direction_flag,c_clock_wise
        mov a,disired_high
        cjne a,#00h,_not_blade
        mov a,disired_low
        cjne a,#00h,_not_blade
        mov modify_disired_high,#02h
        mov modify_disired_low,#0cfh
        jmp _after_modify
_not_blade:
        mov r2,disired_high
        mov r3,disired_low
        mov r4,#00h
        mov r5,#01h
call subb16
        mov modify_disired_high,r2
        mov modify_disired_low,r3
        jmp _after_modify
c_clock_wise:
        mov modify_disired_high,disired_high
        mov modify_disired_low,disired_low
_after_modify:
        mov r2,different_high
        mov r3,different_low
        mov r4,#short_dis_high
        mov r5,#short_dis_low
call subb16
        jnc _rane_is_big
        mov tl0,#0ffh
        mov th0,#000h
        anl ie,#11111101b
        clr  tf0
        setb tr0
go_to_xx01:
        jnb tf0,$
        clr tr0
        clr tf0
        mov tl0,#timer0_slow_low
        mov th0,#timer0_slow_high
        setb tr0
call encoder
call pulse_sequence
call SR_for_stepper
        cpl beeb
        mov disired_point_high,disired_high
        mov disired_point_low,disired_low
        mov modi_disired_point_high,modify_disired_high
```

```
        mov modi_disired_point_low,modify_disired_low
call where_i_am
        jnb i_am_at,go_to_xx01
        clr tr0
        orl ie,#00000010b
        setb beeb
call after_motion
ret
_rane_is_big:
        jb direction_flag,direc_1
call beta_landa_2
        jnc landa_is_positive
        mov r2,#02h
        mov r3,#0d0h
        mov r4,peak_high
        mov r5,peak_low
call subb16
        mov peak_high,r2
        mov peak_low,r3
landa_is_positive:
        mov r2,vally_high
        mov r3,vally_low
        mov r4,#02h
        mov r5,#0d0h
call subb16
        jc pulse
        mov vally_high,r2
        mov vally_low,r3
        jmp pulse
direc_1:
call beta_landa_1
        jnc beta_is_positive
        mov r2,#02h
        mov r3,#0d0h
        mov r4,vally_high
        mov r5,vally_low
call subb16
        mov vally_high,r2
        mov vally_low,r3
beta_is_positive:
        mov r2,peak_high
        mov r3,peak_low
        mov r4,#02h
        mov r5,#0d0h
call subb16
        jc pulse
        mov peak_high,r2
        mov peak_low,r3
pulse:
```

```
        clr end_table
        mov last_freq_holder,#0
        clr  tf0
        mov TH0,#0f5h
        mov TL0,#000h
        setb pulse_rate_direction
        setb tr0
        orl ie,#10000010b
go_to_xx02:
call encoder
        mov disired_point_high,peak_high
        mov disired_point_low,peak_low
        mov modi_disired_point_high,peak_high
        mov modi_disired_point_low,peak_low
call where_i_am
        jnb i_am_at,go_to_xx02
go_to_xx03:
call encoder
        mov disired_point_high,vally_high
        mov disired_point_low,vally_low
        mov modi_disired_point_high,vally_high
        mov modi_disired_point_low,vally_low
call where_i_am
        jnb i_am_at,go_to_xx03
        clr ea
        setb reduce_for_stop
        clr pulse_rate_direction
        clr end_table
        setb ea
go_to_xx04:
call encoder
        mov disired_point_high,disired_high
        mov disired_point_low,disired_low
        mov modi_disired_point_high,modify_disired_high
        mov modi_disired_point_low,modify_disired_low
call where_i_am
        jnb i_am_at,go_to_xx04
        anl ie,#01111101b
        clr tr0
        clr reduce_for_stop
        clr tf0
        setb beeb
call after_motion
ret
;*************************************************
where_i_am:
        clr i_am_at
        mov a,disired_point_high
        cjne a,current_encoder_high,where_i_am00
```

```
        mov a,disired_point_low
        cjne a,current_encoder_low,where_i_am00
        setb i_am_at
ret
where_i_am00:
        mov r2,modi_disired_point_high
        mov r3,modi_disired_point_low
        mov r4,current_encoder_high
        mov r5,current_encoder_low
call subb16
        mov rest_dis_high,r2
        mov rest_dis_low,r3
        mov buffer_of_bits.0,c   ;buffer_of_bits.0= first bit of byte 24h
        mov c,direction_flag
        rlc a
        xrl buffer_of_bits,a
        jb buffer_of_bits.0,_no_passed
        mov a,rest_dis_low
        cjne a,#tolerance_arm,_test_fault
_test_fault:
        jnc _no_passed
        mov a,rest_dis_high
        cjne a,#0,_no_passed
        setb i_am_at
_no_passed:
ret
;*********************************
beta_landa_1:
        clr c
        mov a,current_encoder_low
        add a,#peak_val_low
        mov peak_low,a
        mov a,current_encoder_high
        addc a,#peak_val_high
        mov peak_high,a
        mov r2,disired_high
        mov r3,disired_low
        mov r4,#peak_val_high
        mov r5,#peak_val_low
call subb16
        mov vally_high,r2
        mov vally_low,r3
ret
;*********************************
beta_landa_2:
        clr c
        mov a,disired_low
        add a,#peak_val_low
        mov vally_low,a
```

```
        mov a,disired_high
        addc a,#peak_val_high
        mov vally_high,a
        mov r2,current_encoder_high
        mov r3,current_encoder_low
        mov r4,#peak_val_high
        mov r5,#peak_val_low
call subb16
        mov peak_high,r2
        mov peak_low,r3
ret
;*************************************************
direction_function:
        setb direction_flag             ;disired point:disired_high;disired_low
        mov r4,current_encoder_high     ;current
point:current_encoder_high,current_encoder_low
        mov r5,current_encoder_low      ;different:different_high,different_low
        mov r2,disired_high
        mov r3,disired_low
call subb16
        mov different_high,r2
        mov different_low,r3
        mov 0fh,c
        mov r4,#01h
        mov r5,#68h
call subb16
        jc _is_ok
        mov 0eh,c
        mov r4,#01h
        mov r5,#68h
call subb16
        mov different_high,r2
        mov different_low,r3
        mov c,0eh
_is_ok:
        jb 0fh,_is_negative
        jc _dir11
        clr  direction_flag
_dir11:
ret
_is_negative:
        jnc _dir12
        clr direction_flag
_dir12: ret
;*************************************************
pulse_sequence:
        mov dph,last_seq_holder_high
        mov dpl,last_seq_holder_low
. . . . . .
```

```
. . . . . .
. . . . . .
. . . . . .
put_on_port:
        mov last_seq_holder_high,dph
        mov last_seq_holder_low,dpl
        setb c
        rrc a                                   ; a= 1 1 D EN1 EN2 C B A
        mov sequence_holder_value,a
        ret
;*****************************************
SR_for_stepper:
        mov r7,#8
        mov a,sequence_holder_value
next_bit1:
        rlc a
        mov p3.5,c                      ;p3.5=data(serial to 595)
        clr p3.6                        ;p3.6=clock for 595
        setb p3.6
        djnz r7,next_bit1
        clr p3.7
        setb p3.7                       ;p3.7=SC
        clr p3.7
        clr p3.6
ret
;*****************************************
```

با صدازدن اين تابع جريان كوئل هاي استپر قطع مي شود. از اين تابع پس از پايان يافتن هرحركت دوراني استفاده مي شود.

```
after_motion:
        mov a,sequence_holder_value
        anl a,#11100111b
        mov sequence_holder_value,a
        jmp SR_for_stepper
;*****************************************
```

اين تابع براي نمايش يک عدد 3 رقمي بر روي"LCD" است وقبل از صدا زدن اين تابع بايد مكاني كه نمايش داده مي شود رادرR7 قرار داد.

```
display_on_lcd2:
      mov a,r7                  ;set DDRAM
call write_2_nibbles_command
        mov a,r2
      anl a,#0fh
        orl a,#30h
call write_2_nibbles_data
```

```
        mov a,r3
        swap a
        anl a,#0fh
        orl a,#30h
call write_2_nibbles_data
        mov a,r3
        anl a,#0fh
        orl a,#30h
call write_2_nibbles_data
ret
;*********************************************
subb16:
        clr c
        mov a,R3                ;R2R3-R4R5=R2R3
        mov 0,5                 ;mov R0,R5
call subb8                      ;in return if c=1 then result is negative
        mov R3,a
        mov a,R2
        mov 0,4                 ;mov R0,R4
call subb8
        mov R2,a
        jnc _c_is_0
        mov a,R3
        cpl a
        add a,#01h
        mov R3,a
        mov a,R2
        cpl a
        addc a,#00h
        mov R2,a
        setb c
_c_is_0:
ret
;*******************************************
subb8:
        mov R1,a                        ;a-r0=a,c=0 if pos and =1 if neg
        mov bit_buffer_1,c
        subb a,R0                       ;a-R0=a
        mov bit_buffer_2,c
        jnc _result_pos
        cjne  R0,#00h,_result_pos       ;after never c=1
        clr bit_buffer_2
        cjne  R1,#00h,_result_pos
        jnb bit_buffer_1,_result_pos
        setb bit_buffer_2
_result_pos:
        mov c,bit_buffer_2
ret
;*****************************************
```

:"get_bytes_from_adces"

اين تابع 7 نمونه پشت سر هم از"ADC "هارا مي خواند(تعداد نمونه ها رامي توان با تغيير "number_of_sample" كم يا زيادكرد). توسط تابع" delay3 "بيش از 100ميلي ثانيه بين هر نمونه گيري صبرمي‌كندتا"ADC" ها زمان كافي براي اتمام تبديل بعدي داشته باشد. از آنجا كه رجيسترهاي مربوط به "ADC" ها به صورت سري بسته شده اند درهر بار نمونه گيري بايد 16 بيت را خواند كه آنها را به صورت 2 بايت در دو بردار 7 تاييكه براي ذخيره نمونه ها در نظرگرفته شده است ذخيره مي كند (خانه هاي اول اين دو بردار با آدرس هاي"first_matrix_1" و"first_matrix_2" مشخص مي شوند). همواره پس از اين تابع، تابع"correction_function"صدا زده مي شود كه در ادامه توضيح آن خواهد آمد.

```
;****************************************

get_bytes_from_adces:
        setb common_clock
        setb data_from_adces
        clr shift_hold_adces            ;time for load parallel directly to register
        mov r6,#number_of_sample
        mov r0,#first_matrix_1
        mov r1,#first_matrix_2
ff:
        setb shift_hold_adces           ;S/H for ADC board => inhibit input parallel
data
        mov r7,#8
call loop3
        mov @r0,a
        mov r7,#8
call loop3
        mov @r1,a
        clr shift_hold_adces
        inc r0
        inc r1
call delay3
        djnz r6,ff
ret
;************************************************************
loop3:
        mov c,data_from_adces           ; data for ADC board
        rlc a
        clr common_clock                ;common clock for lcd and ADCes
        setb common_clock
        djnz R7,loop3
ret
;************************************************************
```

```
delay_2:
        mov r7,#0ffh
delay_21:
        mov r6,#0ffh
        djnz r6,$
        djnz r7,delay_21
ret
;*************************************************************
delay_3:
        mov r7,#0ffh
delay_31:
        mov r6,#0ffh
        djnz r6,$
        djnz r7,delay_31
ret
;*************************************************************
delay_4:
        mov r7,#80
delay_41:
        mov r6,#0ffh
        djnz r6,$
        djnz r7,delay_41
ret
;*************************************************************
delay3:
        mov r4,#2
lab12:
        mov r5,#0ffh
        djnz r5,$
        djnz r4,lab12
ret
;*************************************************************
```

: "correction_function"

اين تابع يک بردار به اندازه "number_of_sample" را از بزرگ به کوچک مرتب مي‌کند. قبل از صدا زدن اين تابع بايد آدرس دو خانه اول از اين بردار را در رجيسترهاي r0 و r1 قرار داد.

از آنجا که اين تابع براي مرتب کردن برداري از اطلاعات يک ADC به کار مي رود، بنابر اين بعد از بازگشت از اين تابع خانه چهارم از اين بردار حاوي نمونه اي است که بيشترين تکرار را در بردار داشته است. هرچه تعداد نمونه ها بيشتر باشد احتمال خطا کمتر است و البته زمان بيشتري را از ميکروکنترلر مي‌گيرد. تعداد نمونه 7 تايي مناسب است. از آنجا که محل بردارها در فضاي RAM ثابت است، بنابر اين خانه چهارم داراي آدرس مشخصي است که

آنرا براي 2بردار با نامهاي"mediate_of_f_sample" و "mediate_of_p_sample" مي شناسيم .

```
;*******************************************************
correction_function:                ;input: r0&r1 tow first of list output: sorted list
        mov r6,#number_of_sample
        mov 5,6
        dec r5
again21:
        mov 7,6                             ;mov r7,r6
        dec r7
again11:
        mov a,@r0
        mov 3h,@r1
        cjne a,3h,next
next:
        jnc next1
        mov @r0,3h
        mov @r1,a
next1:
        inc r1
        djnz r7,again11
        inc r0
        mov 1,0
        inc r1
        dec r6
        djnz r5,again21
ret
;*************************************************************
```

توابع" put_at_xx" و" pick_at_xx" :

این توابع موتورهاي DC مربوط به حرکت عمودي و حرکت انگشت را بايك موج PWM با duty cycle برابر با 0/5 به حرکت در مي آورند. تغيير feed باعث تغيير فرکانس اين موج مي شود. در هر لحظه تنها يك موتور را مي توان حرکت داد. بخش عمده کار در اينتراپت تايمر 2 قرار دارد و کدهاي موجود در بدنه اين دو تابع تنها وظيفه نظارت و رديابي نقطه مطلوب را به عهده دارند. توسط متغير "free_or_quick_stop" مي توان مشخص کرد که باقطع جريان موتور ، موتور آزادانه به حرکت خود ادامه دهد(free running) يا سريعا متوقف شود(quick stop) . به محض رسيدن به نزديکي نقطه مطلوب، feed به طور اتوماتيك کم شده و امکان تغيير دستي آن وجود ندارد، مگر اينکه بازو به نقطه مطلوب برسد.

```
;*************************************************************

put_at_xx:
```

```
        mov c,free_or_quick_stop_p
        mov free_or_quick_stop,c
.........
.........
.........
.........
.........
    movc a,@a+dptr
        mov duty_cycle_low,a
jmp_go_ahead:
        jmp go_ahead
;*****************************************************
```

تابع "refresh_adc_val_function" :

با صدا زدن این تابع مقادیر موجود در آدرسهاي "current_adc_finger" و "current_adc_up_down" که معرف موقعیتهاي انگشت و جابجایي عمودي هستند را به لحظه در مي‌آوریم. براي این منظور توابع "get_bytes_from_adces" و "correction_function" صدا زده مي‌شوند. براي جابجایي عمودي عدد جاري پتانسیومتر بر دو تقسیم شده، خارج قسمت آن به عنوان موقعیت جاري ثبت مي‌شود در آخر این موقعیت ها در جاي مناسب بر روي LCD نمایش داده مي شوند.

```
;*****************************************************

refresh_adc_val_function:
call get_bytes_from_adces
        jnb pick_or_put,refresh_adc_put
        mov r0,#first_matrix_1
        mov 1,0
        inc r1
call correction_function
        mov current_adc_finger,mediate_of_f_sample
refresh_adc_put:
        mov r0,#first_matrix_2
        mov 1,0
        inc r1
call correction_function
        mov actual_curr_adc_up_down,mediate_of_p_sample
        mov a,mediate_of_p_sample
        mov b,#2
        div ab
        mov current_adc_up_down,a
end_adc_refresh:
        mov 0,#0
        mov 1,current_adc_finger
call bin_to_bcd
     mov r7,#0d6h
call display_on_lcd2
```

```
        mov 0,#0
        mov 1,current_adc_up_down
call bin_to_bcd
        mov r7,#0e3h
call display_on_lcd2
ret
;*********************************************************
subb8_complete:
        clr c
        mov R1,a                        ;a-r0=a,c=0 if pos and =1 if neg
        subb a,R0                       ;a-R0=a
        mov bit_buffer_2,c
        jnc _result_pos0
        cjne  R0,#00h,_result_pos0    ;after never c=1
        clr bit_buffer_2
_result_pos0:
        jnb bit_buffer_2,c_is_0
        cpl a
        add a,#01h
c_is_0:
        mov c,bit_buffer_2
ret
;*************************************************************
Timer0_sir:
        push acc
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        pop acc
reti
;*********************************************************
Timer2_ir:
        push acc
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        pop acc
reti
;*************************************************************
```

تابع "serial_ir" :

این روتین مربوط به اینتراپت پرت سریال است و وظیفه آن تبادل اطلاعات بانرم افزار کامپیوتري است. براي مثال براي کاهش feed از طریق کامپیوتر اگر کد'A' یا عدد 65 از طریق پرت سریال دریافت شود، اینتراپت رخ داده و میکرو شروع به اجراي این تابع میکند و کد دریافت شده را

باکدهاي مختلف مقايسه مي کند تا هويت آن را تشخيص دهد و بعد از آن که متوجه شد که اين کد مربوط به کاهش 1واحد ازfeed است ، چنانکه feed در حال حاضر برابر با1 نباشد، آنرا يك واحد کم مي کند و از اينتراپت خارج مي شود.

```
serial_ir:
        push acc
        push psw
        push 0
        jb ti,jmp_transfer_complete
        mov a,sbuf
        clr ri
        push b
.........
.........
.........
.........
.........
.........
.........
.........
.........
.........
        clr direction_flag
        jmp end_of_sir
serial_ir_28:
    cjne a,#setb_direction_flag,jmp_end_of_sir
        setb direction_flag
jmp_end_of_sir:
        jmp end_of_sir
;*************************************************************
```

:" intrupt0"

چنانچه اينتراپت فعال باشد با پايين رفتن پايه p3.2 (اينتراپت خارجي صفر)اين تابع اجرا مي شود. در اين تابع ابتدا تابع"get_keyboard" صدازده مي شود تا کدکليد فشرده شده در بافر"keyboard_buffer" قرارگيرد. اگرکليد فشرده شده مربوط به تغييرfeed يا مد استپر باشد آن تغيير اعمال مي شود.

```
;*************************************************************

intrupt0:
    push acc
        push psw
    push 6
        push dph
        push dpl
```

```
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
        pop dph
    pop 6
        pop psw
    pop acc
reti
;*************************************************
```

"get_keyboard" :

"keyboard_buffer"ادر شده فشرده کلید کدآخرین تابع این
قرار مي دهد.

```
;*************************************************

get_keyboard:
        clr p3.3
        nop
        setb p3.3               ;S/H for keyboard
. . . . . .
. . . . . .
        clr p3.3
        mov keyboard_buffer,a
ret
;*************************************************
feed_free:
        db 255,100
. . . . . .
. . . . . .
        db 230,00
;*************************************************
feed_quick:
        db 253,100
. . . . . .
. . . . . .
        db 0ffh
        db 0ffh
;*************************************************
seq_even_1:   db 10111011b          ; 1,D,EN1,EN2,C,B,A,odd/even
. . . . . .
. . . . . .
. . . . . .
seq_odd_4:    db 10100010b
        db 0ffh
        db 0ffh
```

```
;*************************************************
table: db  000h,0fch
       db  037h,0fch
. . . . . .
. . . . . .
. . . . . .
. . . . . .
       db  0e2h,0f7h
;*************************************************
end
```

فصل هفتم: برنامه نويسي رابط گرافيكي

برنامه RobotUI :

شكل هاي7-1، 7-2و7-3، سه تصوير ازرابط گرافيكي اين برنامه را نمايش مي‌دهند. كاركردن با اين برنامه ساده است. در ادامه‌كاركرد دكمه ها وآيتم هاي منوها تشريح مي‌شود:

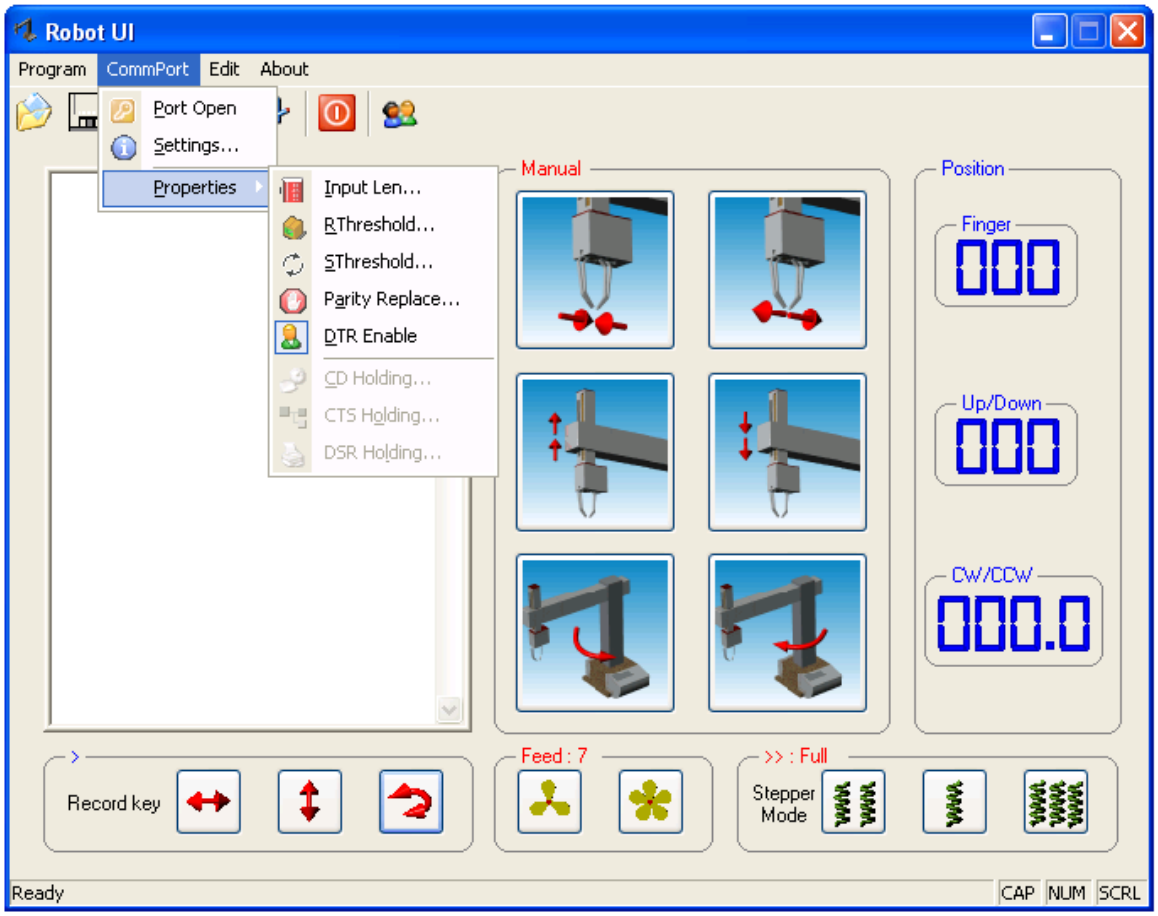
6 دگمه بزرگي كه در گروه"manual" (با رنگ قرمز)قرار دارند براي حركت دستي ربات طراحي شده اندكه تصوير روي هركدام به قدركافي گويا است. سه عدد قرارگرفته دركادر"position" موقعيت‌هاي فعلي هرعضو بازو را نمايش مي-دهند.

با 2دكمه گروه feed مي توان ضريب سرعت راكم و زيادكرد(1 تا 11).

سه دكمه قرارگرفته درگروه" record"، عملكرد مشابه سه دكمه"record"برروي‌صفحه كليد بازو را دارند. با فشردن هردكمه كد وموقعيت جاري مربوط به هردكمه دركادر ويرايش ودر انتهاي برنامه موجود در آن قرار مي گيرد.

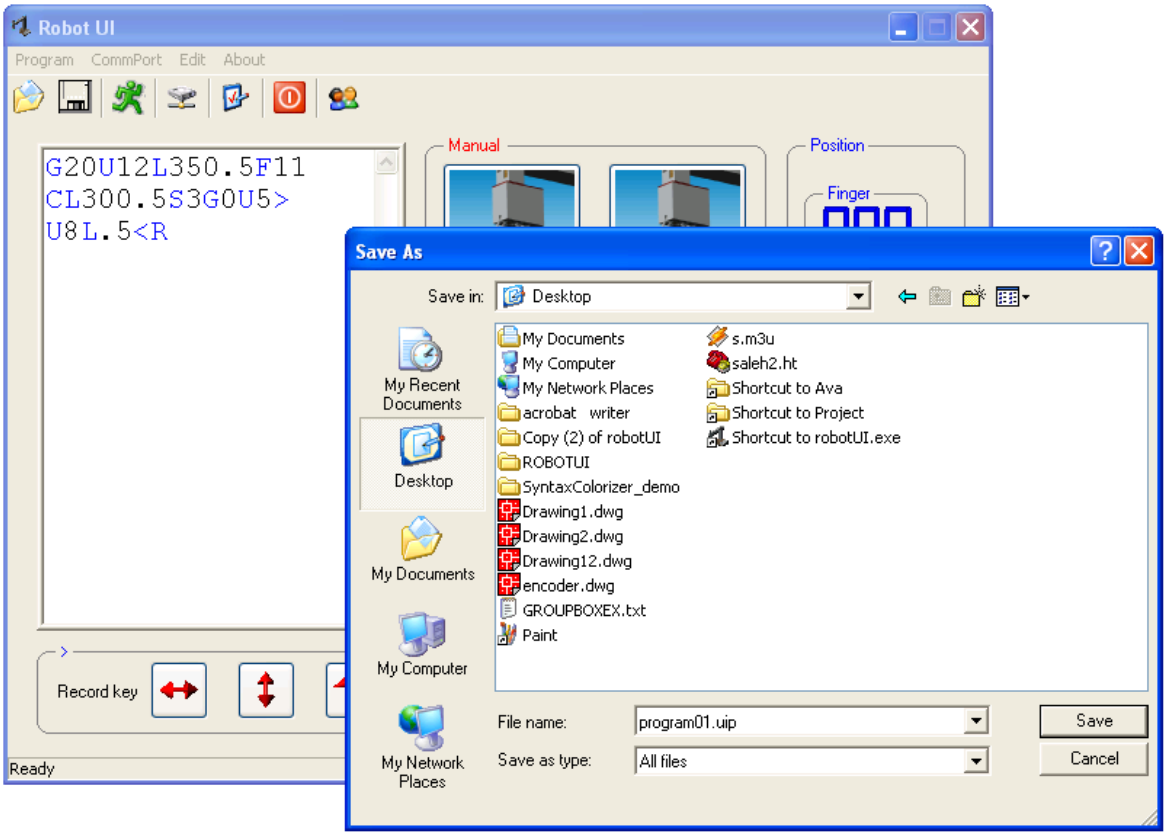
با سه دكمه گروه" stepper mode"مي توان مد حركتي استپر را به"full"يا "wave"يا"half"تغيير داد.

در مد ثبت با راست كليك بر روي هركدام از5 دكمه فوق مي توان‌كد مربوط به آنها رادرجعبه ويرايش وارد كرد(اين قابليت فعلا فعال نيست).



شكل (7-1) رابط گرافيكي

براي اتصال به ربات، پس از آنكه ربات را براي اتصال به pc تنظيم كرديم، در منوي تنظيمات همان تنظيمات را براي برنامه هم انجام مي دهيم وسپس گزينه"Port Open" از منوي"CommPort" را انتخاب مي‌كنيم. پس از اتصال عنوان گزينه به"Port Close" تغييرمي‌كندكه براي قطع ارتباط بايدآنرا انتخاب كنيم.



شكل (2-7) رابط گرافيكي

با فشار دكمه "Download" (دكمه چهارم از سمت راست نوار ابزار) يا آيتم معادل آن در منوي "File"، برنامه هاي موجود درحافظه ميكروكنترلر به انتهاي برنامه موجود درپنجره ويرايش ميچسبدكه ميتوان آنرا بر روي ديسك ذخيره كرد.

براي نوشتن و ويرايش برنامه از پنجره ويرايش استفاده ميكنيم. شيوه نوشتن برنامه مانندنوشتن Gكدبراي يك ماشين CNC است. در زيركدهاي تعريف شده براي اين بازو و عملكرد آنها توضيح داده ميشود:

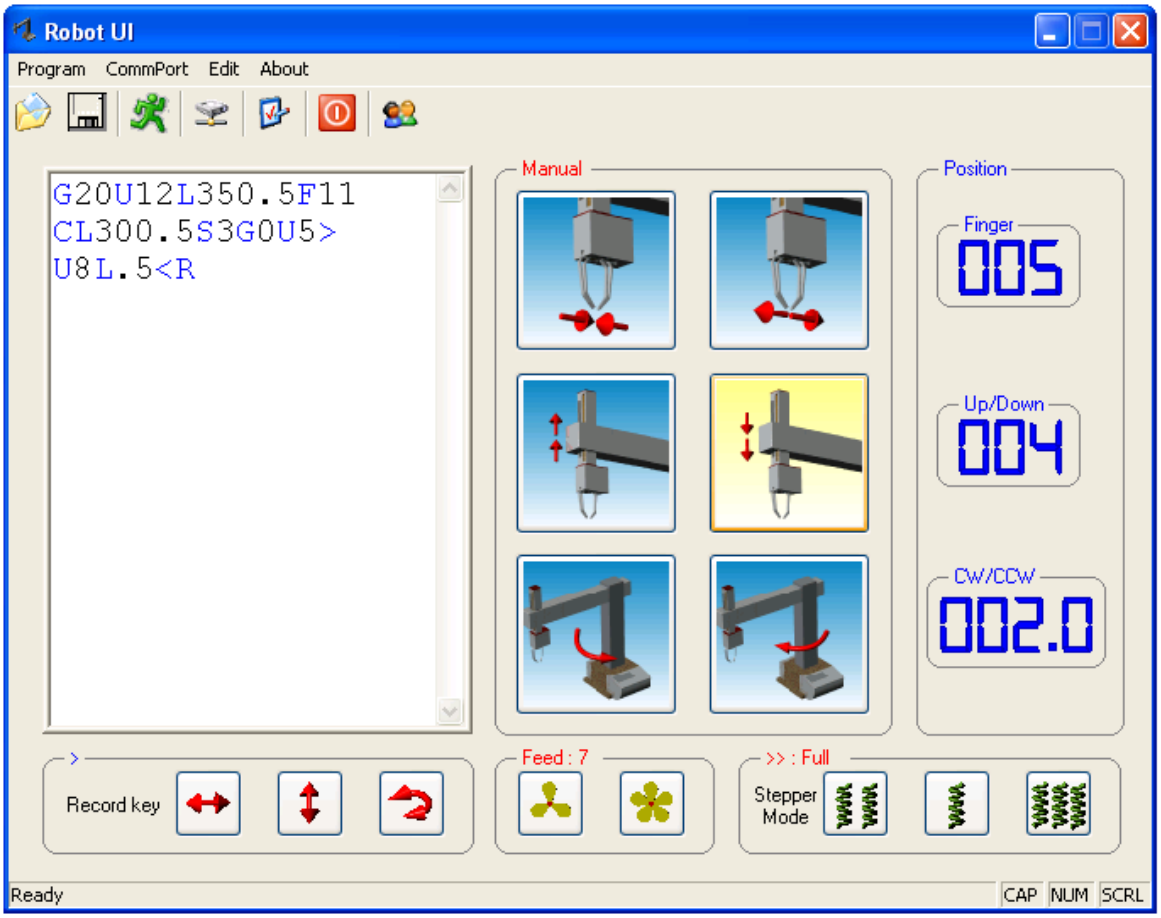
L: زاويه چرخش دوراني را مشخص ميكندو عددي بين 0تا 359/5 ميتواند بعد ازآن قرارگيرد.

G: دهانه انگشت را مشخص مي كند و مي تواند در جلوي خود داراي يكي از مقادير 0تا255 باشد.

U: موقعيت عمودي بازو را تغييرمي دهد ومي توانديكي از مقاديرازتا 127رابپذيرد.

F: ضريب سرعت است ومقدار آن بين 1تا11 تغيير مي كند.

S: مد استپراست و مي تواند 0يا 1 يا 2 باشد.0به معناي مد نيم گام و 1به معناي مدموجي و 2به معناي مد full است.



شكل (3-7) رابط گرافيكي

C : براي انجام همزمان 2 حركت است.يكي از اين حركات چرخش بازو است وحركت ديگر مي تواند حركت انگشت يا حركت عمودي باشد.

R : با رسيدن به اين كد برنامه مجددا از ابتدا اجرا مي شود.

< : با رسيدن به اين علامت فرم pwm به quick stop تغييرمي‌كند.

> : با رسيدن به اين علامت فرم pwm به free running تغييرمي-كند.

T : باعث توقف بازو مي شود .طول اين‌توقف به عدد بعد از اين‌كد درواحدميلي ثانيه است.

بايددر پايان برنامه كاراكتر | را قرار داد. زدن Enter در بين برنامه مجاز نيست.

با انتخاب آيتم save در منويfile يا دكمه مربوط به آن در منوي ابزار مي توان برنامه را ذخيره كرد و بطريق مشابه مي توان آنرا توسط گزينه open باز كرد و براي اجرا

آماده نمود. با انتخابگزينه run از منوي file مي توان برنامه موجود در پنجره ويرايش را اجرا نمود.

منبع اين برنامه را مي توان به دو قسمت تقسيم كرد: كدهاي mfc مربوط به كلاس پايه و كدهائي كه براي اين برنامه خاص نوشته شده اند.كدهاي اخير را نيز مي توان به دو قسمت كدهاي مربوط به شكل ظاهري وكدهايي كه در پشت اين ظاهر كارهاي مربوط به ارتباط با ميكرو وساير عمليات كنترل ربات را برعهده دارند، تفكيك كرد.

كدهايي كه شكل ظاهري برنامه را مي سازند خود به دو دسته تقسيم مي شوند: كدهايي كه از پايه بوجود آورده شده اند مثل كلاس مربوط به كادرهاي با لبه گرد و كلاس هايي كه منبع آنها از اينترنت گرفته شده، گسترش داده شده و براي استفاده در برنامه اصلاحاتي بر روي آنها انجام شده است.

در زير شرح كلي از وظايف كلاسها، توابع و يا بخشي ازكد مربوط به آنها راكه توضيح آن به فهم سريع برنامه كمك ميكند، آورده ام. منبع كامل برنامه درCD همراه با اين پايان نامه در شاخه RobotUI آمده است.

كلاس robotUIDlg :

اينكلاس مربوط به پنجره اصلي برنامه استكه اصليترين و مفصلترينكلاس هم است. كليه عمليات گرفتن وفرستادن اطلاعات از و به پرت سريال و ساير عمليات مشابه توسط اينكلاس صورت ميگيرد.

اينكلاس از 8 تايمر براي انجام عمليات خود استفاده مي كندكه وظيفه هرتايمر به شرح زير است:

تايمر يك(delayTimer) : اگرهنگاميكه به صورت دستي بازو را ميچرخانيم همزمان با رها كردن كليد، استپر متوقف شود، به بازو شوك وارد مي شود. براي رفع اين مشكل با رها شدن كليد، اين تايمرآتش شده وfeed نيز به تدريج كم مي شود. در كدمربوط به اين تايمر(در تابع OnTimer) اين تايمرخاموش شده و كد توقف استپر به بازو فرستاده مي شود.

تايمر دو (getposTimer): با شروع برنامه اين تايمر با فاصلههاي زماني 150 ميلي ثانيه فعال مي شود، در كد اين تايمر(در تابع OnTimer) از ميكرو درخواست ارسال موقعيت فعلي بازو مي شود. بنابراين با هربارسرريز كردن اين تايمر، مختصات جاري ربات به برنامه منتقل مي شود.

تايمر سه (checkposTimer): با شروع اجراي G كدمربوط به جابجائي عمودي يا انگشت، اين تايمر نيز فعال شده وپيوسته موقعيت جاري را با موقعيت مطلوب چك مي كند.

تايمر چهار (changeFeedTimer): اگريكي از دكمه‌هاي Feed فشرده شود، اين‌تايمرآتش‌مي‌شود و با رها كردن دكمه، خاموش مي شود. دركد مربوط به اين تايمر، بسته به دكمه فشرده شده، Feed يك واحد كم يا زياد مي شود.

تايمرپنج (tempTimer): اين تايمر با شروع برنامه روشن مي-شودوخاموش نمي‌شود مگرآنكه پرت سريال باز شودو ميكرو به طور صحيح جواب دهد. دركد مربوط به اين تايمر چنانچه ارتباط برقرار باشد،Feed و مد استپردرميكرو و برنامه هماهنگ مي‌شود.
تايمرشش (shortDisTimer) و تايمرهفت (LongDisTimer): اين تايمرها براي چك كردن مختصات مطلوب(مختصاتي كه در هنگام اجراي Gكد، ماييليم به‌آن برسيم) با مختصات جاري به كار مي‌روند وتنها براي حركت چرخشي بازو استفاده مي شوند.
تايمر هشت (programTimer): براي آنكه در هنگام انتقال برنامه‌هاي ذخيره شده در حافظه ميكرو به داخل نرم‌افزار، با اطلاعات مربوط به موقعيت جاري بازو تداخلي صورت نگيرد، از اين تايمراستفاده مي‌شود.

كلاس Colorizer :

- وظيفه اين‌كلاس، تجزيه وتحليل وخطايابي جمله‌اي است كه به عنوان برنامه در پنجره ورود G كدنوشته مي‌شود. اين كلاس داراي سه نسخه از يك تابع به نام"Phrase" است كه اين عمل را انجام مي‌دهد وبسته به نياز يكي را مي توان صدا زد. چون با اجراي توابع اين‌كلاس بر روي يك برنامه، كدها، عددها، توضيحات و خطاها به رنگهاي مختلف ظاهر مي شوند، از اين جهت نام اين كلاس Colorizer گذارده شده است.
تابع عضو"Correction"، وظيفه خطايابي برنامه را به عهده دارد.

كلاسCommCtrl :

اين كلاس يك شئ mscomm32.ocx را رجيسترکرده وسپس با تابع هايGetProperty و SetProperty به توابع و متغيرهاي عضوآن دسترسي پيدا ميكند.
توضيح ضروری: mscomm32.ocx بر روی CD همراه با پايان نامه موجود است. قبل از اجرای از اجرای نرم‌افزار بر روی يک سيستم، بايد اين كنترل را در سيستم عامل رجيستر كنيد.

كلاس SettingsDlg :

اين كلاس مربوط به پنجره‌اي است كه از طريق آن تنظيمات پرت سريال انجام مي شود.

كلاسProgramEdit :

اين كلاس از يك كلاسCRichEditCtrl مشتق شده است و وظيفه آن كنترل عمليات مربوط به جعبه ويرايش برنامه است.

در اينجا براي نمونه، يكي از كلاس هاي برنامه شرح داده مي شود. اين كلاس براي ترسيم يك كادر با لبه‌هاي گرد به كار رفته است. رنگ كادر ورنگ فونت عنوان كادر، قابل تغيير است. تعريف اين كلاس را در زير مشاهده مي كنيد:

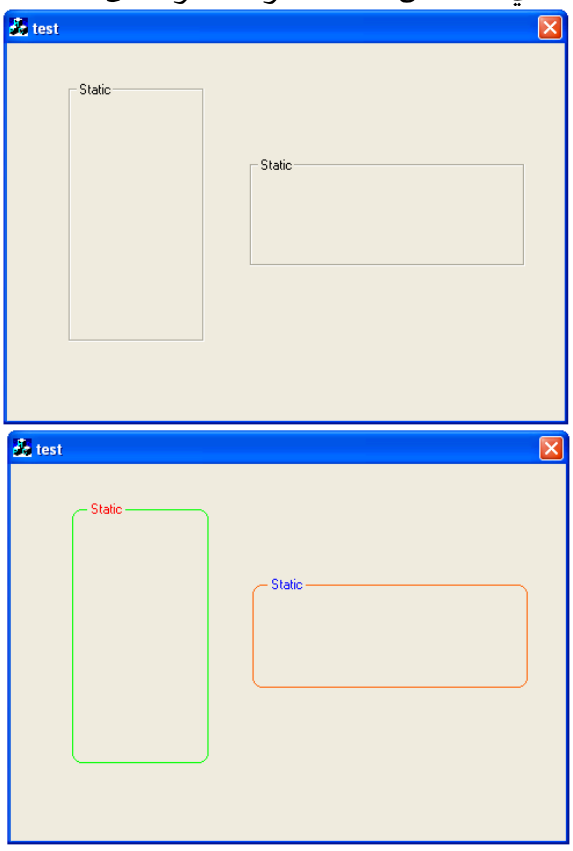
```
_if !defined(AFX_GROUPBOXEX_H__F4365F9B_7CC5_4C4A_A4B2_DF3F6BC27C9D__INCLUDED#
(
_define AFX_GROUPBOXEX_H__F4365F9B_7CC5_4C4A_A4B2_DF3F6BC27C9D__INCLUDED#

if _MSC_VER > 1000#
pragma once#
endif // _MSC_VER > 1000#
GroupBoxEX.h : header file //
//

////////////////////////////////////////////////////////////////////////
CGroupBoxEX window //

class CGroupBoxEX : public CStatic
}
Construction //
:public
;()CGroupBoxEX
;m_boxRect               CRect
;m_txtString             CString
;m_txtColor       COLORREF
; m_ColorBkd      COLORREF
;m_txtFont              CFont*

Attributes //
:public

Operations //
:public

Overrides //
ClassWizard generated virtual function overrides //
(AFX_VIRTUAL(CGroupBoxEX}}//
AFX_VIRTUAL{{//

Implementation //
:public
;( void DesignRect(int,int,int,int,int,CRect,COLORREF
;()virtual ~CGroupBoxEX
,(void    ShowControl(COLORREF m_txtColor = RGB(0,0,255
;((COLORREF m_lineColor = RGB(128,128,128

Generated message map functions //
:protected

(AFX_MSG(CGroupBoxEX}}//
.NOTE - the ClassWizard will add and remove member functions here //
AFX_MSG{{//

()DECLARE_MESSAGE_MAP
;{

////////////////////////////////////////////////////////////////////////
```

```
{{AFX_INSERT_LOCATION}}//
.Microsoft Visual C++ will insert additional declarations immediately before the previous line //

endif #
(_// !defined(AFX_GROUPBOXEX_H__F4365F9B_7CC5_4C4A_A4B2_DF3F6BC27C9D__INCLUDED

_if !defined(AFX_GROUPBOXEX_H__F4365F9B_7CC5_4C4A_A4B2_DF3F6BC27C9D__INCLUDED#
(
_define AFX_GROUPBOXEX_H__F4365F9B_7CC5_4C4A_A4B2_DF3F6BC27C9D__INCLUDED#

if _MSC_VER > 1000#
pragma once#
endif // _MSC_VER > 1000#
GroupBoxEX.h : header file //
//

/////////////////////////////////////////////////////////////////////////////
CGroupBoxEX window //

class CGroupBoxEX : public CStatic
}
Construction //
:public
;()CGroupBoxEX
;m_boxRect                CRect
;m_txtString              CString
;m_txtColor       COLORREF
; m_ColorBkd       COLORREF
;m_txtFont              CFont*

Attributes //
:public

Operations //
:public

Overrides //
ClassWizard generated virtual function overrides //
(AFX_VIRTUAL(CGroupBoxEX}}//
AFX_VIRTUAL{{//

Implementation //
:public
;( void DesignRect(int,int,int,int,int,CRect,COLORREF
;()virtual ~CGroupBoxEX
,(void    ShowControl(COLORREF m_txtColor = RGB(0,0,255
;((COLORREF m_lineColor = RGB(128,128,128

Generated message map functions //
:protected

(AFX_MSG(CGroupBoxEX}}//
.NOTE - the ClassWizard will add and remove member functions here //
AFX_MSG{{//

()DECLARE_MESSAGE_MAP
;{

/////////////////////////////////////////////////////////////////////////////
```

{{AFX_INSERT_LOCATION}}//
.Microsoft Visual C++ will insert additional declarations immediately before the previous line //

endif #
(_// !defined(AFX_GROUPBOXEX_H__F4365F9B_7CC5_4C4A_A4B2_DF3F6BC27C9D__INCLUDED

نام اين كلاسCGroupBoxEX است و از كلاسCStatic مشتق شده است. براي استفاده از اين كلاس، توسط كنترل هاي CStatic محل-كادرها را بر روي پنجره مشخص مي‌كنيم. آنگاه اين‌كادرها را شئ ازكلاس CGroupBoxEX تعريف مي‌كنيم. وتابع ShowControl را بر روي اين اشيا در تابع OnPaint پنجره مادر، صدا مي زنيم. پارامتر اول اين تابع رنگ متن عنوان كادر و پارامتر دوم رنگ خود كادر است. نبايد فراموش كرد كه گزينه Visible ازجعبه خواص‌كنترل ها، تيك نخورده باشد.
نمونه اي ازكنترل هاي استاتيك معمولي را درشكل 7-4 وكنترل-هاي مشتق شده را درشكل 7-5 مي‌بينيد.



شكل (7-4) كنترل هاي استاتيك معمولي
شكل (7-5) كنترل هاي استاتيك مشتق شده

درزيرجزئيات مربوط به اين‌كلاس‌آورده شده است:
اين كلاس داراي 5 متغيرعضو و2 تابع عضو است.
متغيرm_boxRect براي نگهداري مستطيل كادر استفاده مي‌شود.
متغير m_txtString براي نگهداري جمله عنوان‌كادر به‌كارمي رود. 2متغير m_txtColor و m_ColorBkd از نوع COLORREF بوده و

رنگ متن وپس زمینه‌کادر را نگهداري مي‌کنند. متغیر m_txtFont اشاره‌گر به شئ ازکلاس CFont بوده و معرف مشخصات فونت عنوان‌کادر است.

تابع عضو ShowControl براي نمایش‌کادر به‌کار مي رود و پارامتر اول این تابع رنگ متن عنوان کادر و پارامتردوم رنگ خودکادر است. در این تابع، ابتدا اشاره‌گري به DC پنجره اصلي بدست مي‌آید. سپس مستطیل‌کادر، بدست‌آمده و مختصات آن از مختصات desktop به مختصات پنجره اصلي برنامه تبدیل مي‌شود. پس ازآن نقطه شروع وانتهاي متن عنوان کادر محاسبه مي‌شود و سپس قلم وقلم موي مناسب با توجه به رنگ هاي انتخابي بارگذاري مي‌شود. در ادامه با صدا زدن تابع هاي DesignRect و DrawText کادرترسیم شده وعنوان آن در جاي خود نوشته مي‌شود. در تابع DesignRect از توابع AngleArc و LineTo براي ترسیم کادر با لبه گرد استفاده مي شود.

```
(void CGroupBoxEX::ShowControl(COLORREF m_txtColor,COLORREF m_lineColor
}
;()CWnd* pWnd = AfxGetMainWnd
;()CDC* pDC = pWnd->GetDC
store the box rectangle //
;(GetWindowRect(&m_boxRect
;(MapWindowPoints(pWnd,&m_boxRect
;(ScreenToClient(&m_boxRect

;(GetWindowText(m_txtString
add a blank on each side of text //
; CString sText
;(sText.Format(" %s ", m_txtString
;CPoint ptStart, ptEnd
;(CSize seText0 = pDC->GetTextExtent(sText
;(CSize seText = pDC->GetTextExtent(sText
;ptStart.x = m_boxRect.left + 8
;ptEnd.x = ptStart.x + seText.cx
;ptStart.y = ptEnd.y = m_boxRect.top + seText0.cy / 2 - 1
m_ColorBkd = ::GetSysColor(COLOR_3DFACE); // Initializing background color to the system
.face color
;CBrush mybrush
;(mybrush.CreateSolidBrush(m_ColorBkd
;(CPen myPen(PS_SOLID,1,m_lineColor
;(pDC->SelectObject(myPen
;(pDC->SelectObject(mybrush
DesignRect( m_boxRect.left,ptStart.y-1,m_boxRect.right,m_boxRect.bottom,8,CRect(ptStart,
;(ptEnd), m_lineColor

;(CFont *oldFont = pDC->SelectObject(m_txtFont
;CSxLogFont lf0
delete current font //
(if(m_txtFont
;delete m_txtFont
create new font on heap //
;m_txtFont = new CFont
;(m_txtFont->CreatePointFontIndirect(&lf0
set color and drawing mode //
;()COLORREF oldColor = pDC->GetTextColor
;(pDC->SetTextColor(m_txtColor
```

```
;(pDC->SetBkMode(TRANSPARENT          //
;(pDC->SetBkColor(m_ColorBkd
get top of text box //
;ptStart.y -= 3
;ptStart.x += 5
,(pDC->DrawText(sText, CRect(ptStart, ptEnd
;(DT_VCENTER | DT_LEFT | DT_SINGLELINE | DT_NOCLIP

;(pWnd->ReleaseDC(pDC
{
```

در پیوست 2 اطلاعات مربوط به نحوه استفاده از نرم افزار برای نوشتن و اجرای یک برنامه آورده شده است.

فصل هشتم: جمع بندی و پیشنهادات

این پایان‌نامه تلاش برای توسعه یک محیط نرم‌افزاری برای یک مجموعه CIM بود که برای نمایش قابلیت‌های آن سخت‌افزاری ساده، طراحی و ساخته شد. این نرم‌افزار به دو بخش شامل نرم‌افزار میکروکنترلر و نرم‌افزار کامپیوتری تقسیم می‌شود. موارد زیر را می‌توان به عنوان زمینه‌های توسعه آتی در این دو بخش، ذکر کرد:

میکروکنترلر: برنامه‌ای که از طریق صفحه کلید بازو نوشته می‌شود، تنها سه G کد از ده تا را می‌پذیرد. برنامه را تا پذیرفتن تمام G کدها باید توسعه داد. امکان ویرایش برنامه را نیز باید فعال کرد.

RobotUI: با یک کامپیوتر چند ربات را می‌توان به طور همزمان کنترل کرد. تنها لازم است برای هر ربات یک نسخه از نرم‌افزار اجرا شود. عامل محدودکننده تعداد پرت‌های سریال کامپیوتر و سرعت پردازش آن است. این نرم‌افزار را باید گسترش داد تا یک نسخه از آن بتواند چند ربات را به طور همزمان کنترل کند.

توسعه سخت‌افزاری:

انکودر: نمونه ساخته شده بسیار بزرگ است که دلیل آن محدودیت اندازه لدهای فرستنده و گیرنده است. باید با باریک کردن اشعه مادون قرمز، حجم انکودر را کاهش داد و دقت آن را بالا برد.

حسگرها: این بازو فاقد هرگونه حسگر است و به همین دلیل یک درجه آزادی برای انگشت در نظر گرفته شده است تا بازو بتواند اجسام با ابعاد مختلف را بگیرد. با اضافه کردن حسگرهای فشار می‌توان حرکات انگشت را سریع‌تر کرد.

قسمت الکتریکی: برای منبع تغذیه بازو از ترکیب ترانس و رگولاتور استفاده شده، که ساده ترین و سریع‌ترین روش ممکن است. باید یک منبع تغذیه سوئیچینگ کوچک برای ربات طراحی شود.

مکانیک بازو: بازوی ساخته شده دارای قابلیت اطمینان پایینی است و باید تحلیل‌های مکانیکی برای اجزای سازنده بازو صورت گیرد.

یکی از قابلیت‌های بالقوه این بازو، درجه آزادی مربوط به حرکت شعاعی است (حرکت دور شدن ونزدیک شدن انگشت به بازو). چون این حرکت مشابه حرکت عمودی است تنها لازم است تا با تغییر مختصری در برنامه‌ها و قسمت مکانیکی بازو، این قابلیت را فعال کرد. از آنجا که فضای کافی در اختیار است، به راحتی می‌توان نمونه خطی انکودر ساخته شده را با دقت بسیار بالاتر برای این حرکت استفاده کرد.

منابع

-Muhammad Ali Mazidi, Janice Gillispie Mazidi ,8051 microcontroller and embedded systems, Prentice Hall, Upper Saddle River, N.J, c2000

-Richard M. Jones, Introduction to MFC Programming with Visual C++, Prentice Hall , December 22, 1999

- Myke Predko,Programming Robot Controllers,McGraw-Hill,New York,2003

مكنزي، آي . اسكات، ميكروكنترلر ۸۰۵۱ ، بنياد فرهنگي -ناغاني/ خراسان، ۱۳۷۷

- لي فور، رابرت، برنامه نويسي شئ گرا با C++، انتشارات سيماي دانش، تهران ، 1378

منابع اينترنتي براي برنامه نويسي ويژوال:

http://www.experts-exchange.com
http://www.codeproject.com/
http://devcentral.iftech.com/default.php
http://directory.google.com/Top/Computers/Programming/Languages/C++/Class_Libraries/MFC/
http://www.2000shareware.com/

منابع اينترنتي براي برنامه نويسي ميكروكنترلر:

http://www.8052.com/
http://www.myke.com
http://www.repairfaq.org/
http://www.technologicalarts.com
http://www.worldservo.com

منابع اینترنتي متفرقه:

http://www.qsl.net/oe5jfl/encoder.htm
http://www.mathtools.net/Applications/Data_Acquisition/index.html
http://chaokhun.kmitl.ac.th
http://www.howstuffworks.com/index.htm
http://www.kmitl.ac.th/
http://www.seattlerobotics.org
http://ee.cleversoul.com/
http://emulazione.multiplayer.it/
http://users.auth.gr/~mixos/projects/circuitslab.htm
http://www.hans-w.com
http://members.shaw.ca/cncstuff/home.html
http://cstep.luberth.com/
http://home.iae.nl/users/pouweha/index.shtml

forum:
http://www.eio.com

پيوست 1

نحوه كار با ربات ازطريق صفحه كليد

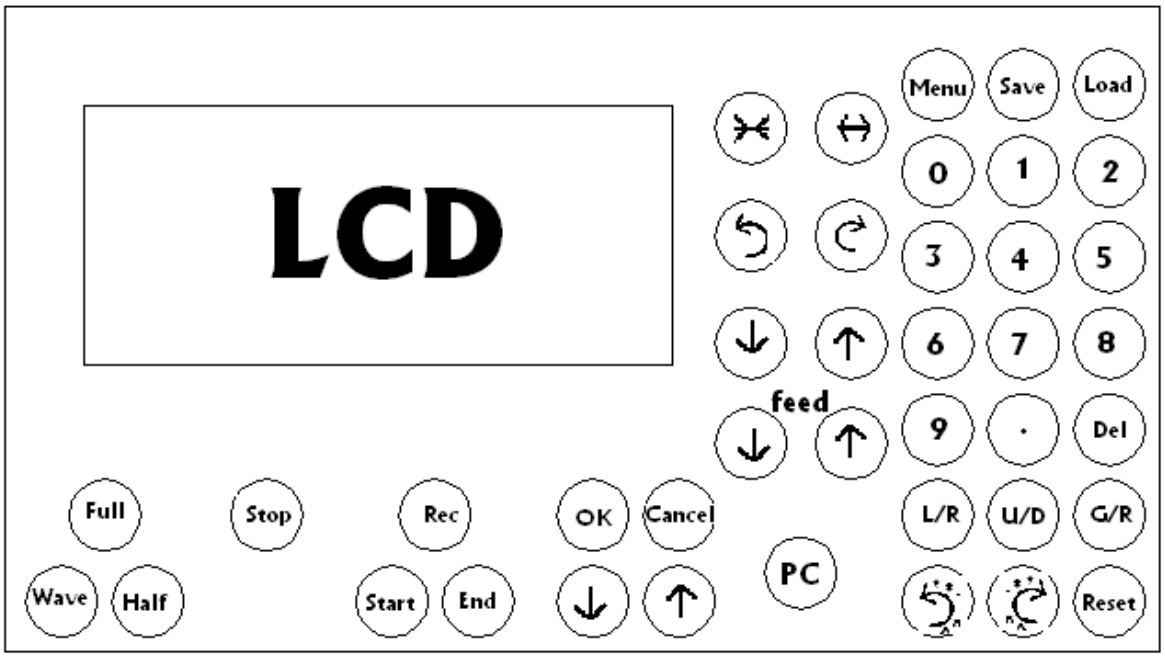
توضيحات مربوط به صفحه كليد در فصل دوم، آمده است. در اينجا به دو روش برنامه‌اي را در حافظه بازو وارد مي‌كنيم كه با اجراي آن، ربات مكعبي را از محلي برداشته و در محل ديگري به زمين مي‌گذارد.

روش اول:

دكمه Start رافشاردهيد. ربات نام برنامه را درخواست مي‌كند. يك كلمه سه حرفي بايد وارد شود. براي اين منظورمي‌توان ازكليدهاي عددي يا 2 كليدقرارگرفته درسمت چپ كليد Reset استفاده كرد. براي استفاده ازاين 2 كليد، يكي ازآنهارافشاردهيدونگه داريد. با اين كار،حروف الفباي كوچك و بزرگ و اعداد 0 تا 9 به ترتيب درمحل كرسرظاهرمي-شوند. تفاوت 2كليدتنهادرجهت نمايش حروف واعداد است. بارسيدن به عدد ياحرفموردنظركليد را رها كنيد. چنانچه از كد موردنظر عبوركرديد، باكليد ديگر مي‌توانيد به آن كد برگرديد.

پس از واردكردن سه حرف براي نام برنامه، كليدOK رافشاردهيد، ازاين لحظه به بعد، ربات درمد ثبت قراردارد. ربات راباكليدهاي حركت دستي، به يك نقطه اوليه مثل (0، 0، 0) ببريد (باشروع اجراي برنامه، ربات در هرموقعيتي كه باشد، ابتدا خودرا به اين نقطه مي-رساند). موقعيت جاري راثبت كنيد، براي اين منظور دكمه

Recرا فشار دهيد. صفحه اي ظاهرمي‌شودكه ازشما مي‌پرسدكدام يك از سه موقعيت، ثبت شود. يكي از كليدهاي L/R,U/D,G/R را فشار دهيد، براي دو موقعيت باقيمانده نيز مراحل بالا را تكراركنيد تا هر سه موقعيت جاري ثبت شود.



شكل پ(1-1) صفحه كليد متصل به بازو

قطعه رادرموقعيت ابتدائي خودقراردهيدو با ربات آنرا بگيريد اما قبل ازبلندكردن قطعه، موقعيت‌ها را ثبت كنيد. توجه كنيدكه دراين حالت ترتيب ثبت موقعيت‌ها اهميت دارد و بايدبه ترتيب، U/D, L/R وG/R راثبت كرد. سپس قطعه را تا ارتفاع مناسب بلندكرده واين موقعيت را نيزثبت كنيد. طبيعي است كه دراين حالت تنها موقعيت U/D بايد ثبت شود، چون دوموقعيت ديگر، تغييرنكرده اند.

قطعه رابه محل موردنظربرده وآن را به زمين بگذاريد. پس از بازشدن انگشت ورهائي قطعه، موقعيت جاري راثبت كنيد. بازهم توجه شودكه موقعيت‌ها به ترتيبU/D, L/R وG/R ثبت شوند. درآخربازو را به محل استراحت! (محلي كه پس ازاتمام برنامه، بازو درآن قرارمي‌گيرد) برده وآن موقعيت رانيزثبت كنيد.

عمليات ثبت تمام است. دكمه End رافشاردهيدتابرنامه به طورموقت درحافظه ذخيره شود.

روش دوم:

دراين روش مختصات رابه صورت دستي واردمي‌كنيد. كليد Menu رافشاردهيد. صفحه اي ظاهر مي‌شودكه گزينه اوّل آن براي واردكردن يك برنامه وگزينه دوّم آن براي ويرايش يك برنامه موجوداست. گزينه دوّم درحال حاضرفعّال نيست. گزينه اول را انتخاب كنيد. نام برنامه درخواست مي‌شود، مانندروش اول يك اسم سه حرفي براي برنامه انتخاب

كنيدودكمه OK رافشاردهيد. صفحه اي ظاهرمي‌شودكه مي‌توانيد برنامه را درآن واردكنيد.
براي واردكردن يك موقعيت، ابتداكليدمربوط به آن (يكي ازكليدهايU/D, L/R يا G/R) رافشارداده وسپس مقدارآن را واردكنيد. براي حركت چرخشي بايدعددي بين0 تا 359/5 واردشود. مميز به طورخودكاردرمحل خودظاهرمي‌شودوبعدازآن تنهالازم است0 يا 5 واردشود. براي حركت عمودي، عددي بين0 تا 127 وبراي انگشت عددي بين 0 تا 255 رابايد واردكرد. بعدازآن كه همه برنامه راواردكرديد، دكمه Save را فشاردهيدتا برنامه به طورموقت درحافظه ميكرو ذخيره شود.
توجه: تعدادارقام مهم است به طورمثال به جاي عدد27 بايدعدد027راواردكرد.

اجراي برنامه:

بازدن دكمه Load نام اولين برنامه‌اي كه درحافظه ميكروواردشده است، برصفحه ظاهرمي‌شود. براي انتخاب برنامه بعدي، دكمهCancel رافشاردهيد، پس ازرسيدن به نام برنامه موردنظردكمهOK رابراي اجراي آن فشاردهيد. براي خارج شدن ازحالت اجرا، بدون اجراكردن برنامه‌اي، دكمه Stop رافشاردهيد.
درحين اجرايياقبل ازآن مي‌توانFeed راتغييرداد. درحين اجرافشاردادن ونگه داشتن كليدهاي Feed تنهايك واحد، آن راتغييرمي‌دهد. بنابراين براي تغييرمجددآن، بايددكمه را رها كرد ودوباره فشارداد. درسايرحالات فشاردادن ونگه داشتن يكي ازكليدهايFeed باعث تغييرپيوسته آن مي‌شود. مداستپرهم مانند Feed، درحين اجرايابعدازآن قابل تغييراست.
توجه: باقطع منبع تغذيه بازو، برنامه‌ها ازبين مي‌روند، بنابراين بايدبهPC وصل شد وآنها را برروي ديسك ذخيره كرد. چگونگي اين عمل درپيوست 2 تشريح شده است.

پيوست 2

نحوه كاربانرم افزار RobotUI

براي اتصال روبات به نرم‌افزار، در روي صفحه كليد ربات دكمه PC را فشار دهيد. ليستي ظاهر مي‌شودكه درآن نرخ تبادل اطلاعات پرت سريال را وارد مي‌كنيم، پس از انتخاب يك گزينه مناسب كليد OK را فشار دهيد. از اين مرحله به بعد ربات تنها از طريق نرم‌افزار قابل كنترل است.

حال به سراغ نرم‌افزار RobotUI مي‌رويم. پس از اجراي نرم‌افزار از منوي CommPort گزينه Settings را انتخاب كنيد. جعبه محاوره‌اي ظاهرمي‌شودكه درآن نرخ تبادل اطلاعات و پرتي كه ربات ازطريق آن به كامپيوتر وصل شده است را وارد مي‌كنيد. پس از اين كارOK را فشار دهيد.

از منوي CommPort گزينه PortOpen را انتخاب كنيد. در اين لحظه اگر تنظيمات صحيح باشد، موقعيت جاري ربات در كادر Position منعكس مي‌شود و عنوان پنجره نرم‌افزار و ضعيت اتصال به بازو را نمايش مي‌دهد.

- عملكرد دكمه‌ها، آيتم‌هاي منوها، نوار ابزار و همچنين مفهومG كدهادرفصل هفتم تشريح شده است. براي روشن شدن قابليت هاي نرم‌افزار، يك برنامه كوتاه درزيرتشريح مي شود. اين برنامه را مي‌توان درپنجره ويرايش واردكرد و با فشار دكمهRun از نوار ابزار، آنرا اجرا نمود.

F9S1<CG8L300.5>F11U107G5U10L0U50G10U10T20000CL30U100U120G7U20|

عمليات زيرنتيجه اجراي اين برنامه است:

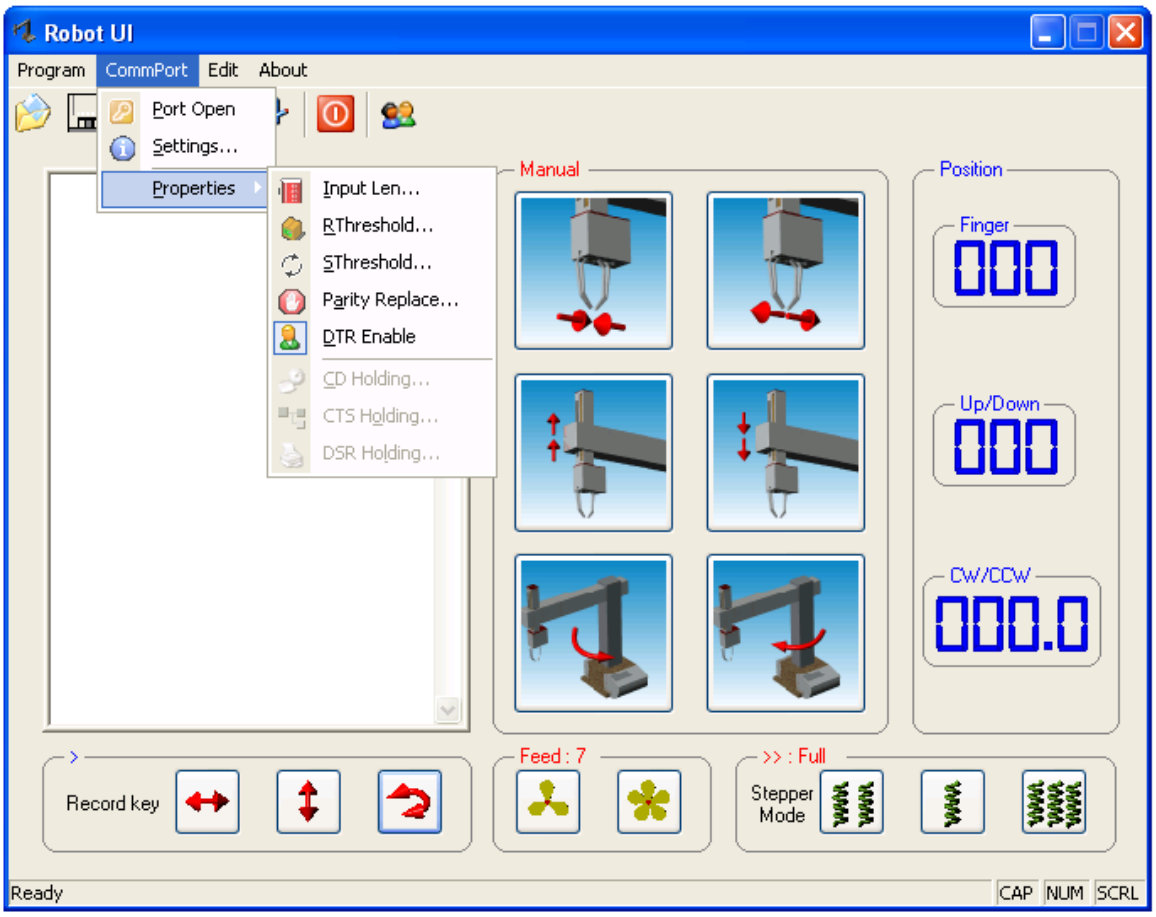
>F9S1: پس ازاجراي اين سه كد، feed به 9 تغييركرده و مد استپربه Wave تبديل مي‌شود. پس ازاجراي كد

> ، موج PWM محرك موتورها، به شكلquick-stop اعمال مي‌شود.

CG8L300.5: دهانه انگشت تغييرمي‌كندتابه موقعيت8 برسد. همزمان با آن بازومي‌چرخدتادر موقعيت 300/5 درجه قرارگيرد.

F11>: فرم موجPWM رابهfree-running تغييرداده وfeed رانيزدربيشترين مقدارخودقرارمي‌دهد.

U107G5U10L0U50G10U10T20000: بازو حرکت عمودي کرده و قطعه اي را مي‌گيرد وسپس آن را بلند مي‌کند و مي‌چرخد تا به موقعيت 0 درجه برسد. سپس قطعه را پايين آورده و رها مي‌کند و به بالا برمي‌گردد و2 ثانيه صبرمي‌کند.



شکل پ(2-1) رابط گرافيکي

و به همين ترتيب پس ازسپري شدن 2ثانيه، بازو سايرکدها را اجرا مي‌کند تا به کاراکتر| برسدکه به معناي پايان برنامه است.

براي انتقال برنامه‌هاي موجود درحافظه ميکرو به روي ديسک کامپيوتر، دکمه DownLoad را کليک کنيد. پس از چند لحظه تمام برنامه‌ها، پشت سرهم به انتهاي برنامه موجود درپنجره ويرايش اضافه مي‌شوند، که مي‌توانيد آنها را ذخيره کنيد.

کاربرددکمه Correction برروي نوارابزار، خطاگيري برنامه موجود درپنجره ويرايش است. فعلاً اين قابليت فعال نيست.

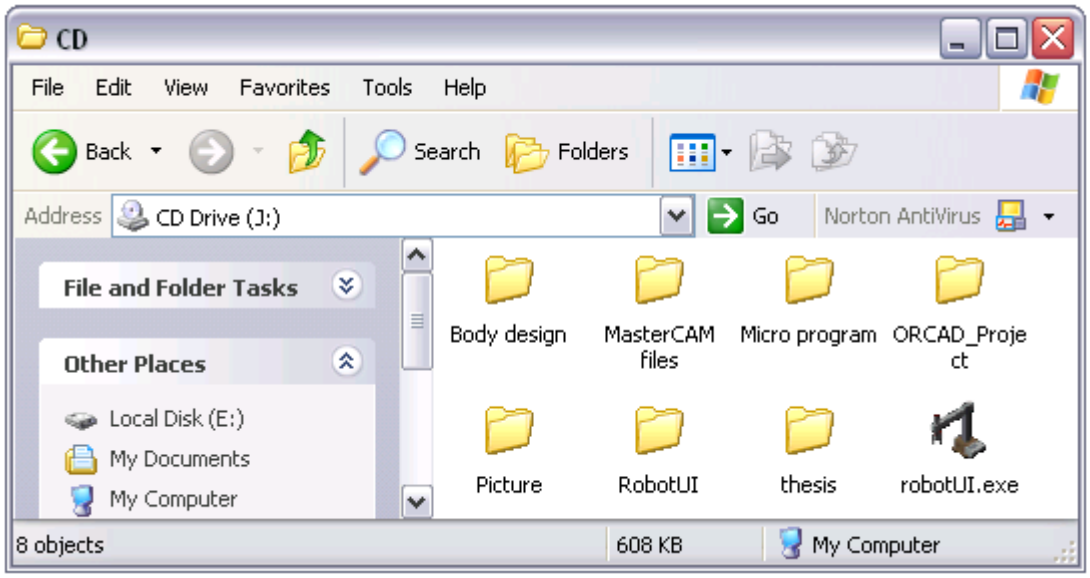
پس ازاتصال ربات به PC، بجزکليد Reset صفحه کليد ربات، کليد ديگري فعّال نيست. براي خارج شدن از وضعيت اتصال، تنهاراه کليک کردن دکمه disconnect بر روي نوار ابزار است.

توجه: اگر پس ازکليک گزينه PortOpen، اتصال برقرارنشد يا خطائي روي داد، به اين دليل است که يا نرخ تبادل

اطلاعات در2طرف یکسان نیست و یا ربات دروضعیت متصل به PC قرارندارد.

پیوست 3: محتوای دیسک فشرده
وچند تصویر از نمونه ساخته شده

محتوای دیسک را در شکل زیر مشاهده می‌کنید:



شکل پ(3-1) محتوای دیسک فشرده

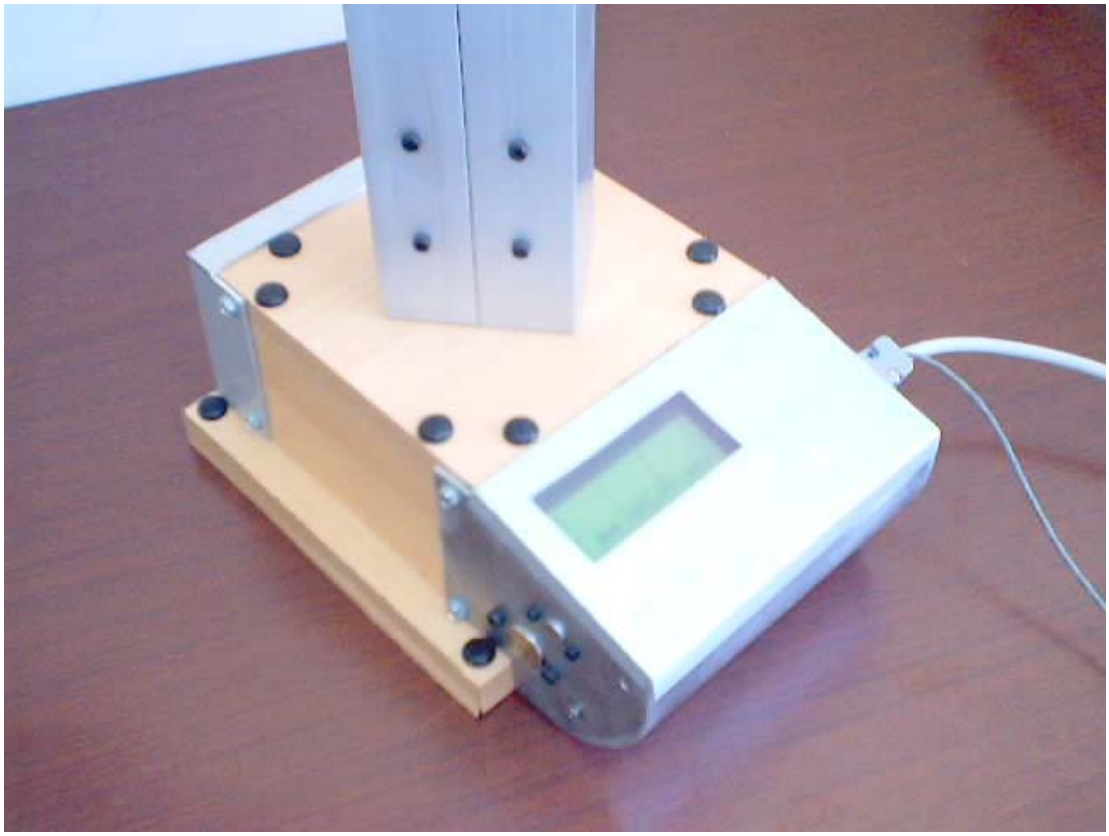
Body design: حاوی فایل های SolidWorks مربوط به مکانیک بازو.
MasterCAM files: فایل های MasterCAM مربوط به انگشت و هد انکودر.
Micro program: فایل های ROBOT.HEX و ROBOT.ASM مربوط به میکروکنترلر.
ORCAD_Project: حاوی فایل های ORCAD مربوط به الکترونیک بازو.
RobotUI: حاوی فایل های VC++6 مربوط به نرم افزار بازو.
thesis: فایل های Word مربوط به این پایان نامه.
robotUI.exe: نرم افزار روبوت.

چند تصویر از نمونه ساخته شده:

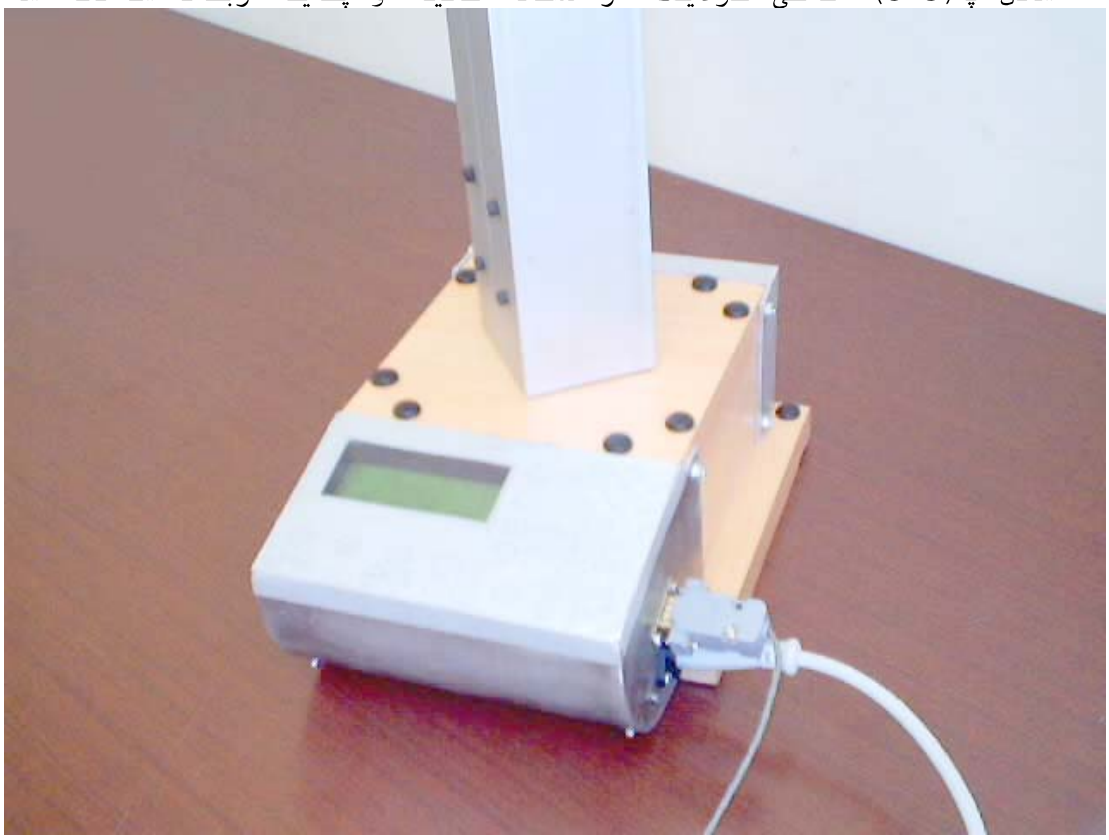
شکل پ(3-2) تصویری از نمونه ساخته شده

شکل پ(3-3) تصویری دیگر از نمونه ساخته شده

شکل پ(3-4) تصویری از نمونه ساخته شده



شکل پ(3-5) نمائی نزدیک از صفحه کلید و پایه ربات ساخته شده



شکل پ(3-6) زاویه دیگری از صفحه کلید و پایه ربات ساخته شده

شکل پ(3-7) نمائی نزدیک از انگشت ربات ساخته شده