RME 40002
MECHATRONICS SYSTEM DESIGN
COURSE PROJECT FINAL ASSESSMENT REPORT
GROUP 5

DESIGN AND DEVELOPMENT OF AUTOMATED GOODS PICKING MACHINE

NAME	**STUDENT ID**
JEFF FONG JUN EN	100086314
KAU HUI TING	101212796
CHUAH DING KEN	101218820

Table of Contents

Lists of Figures

Lists of Tables

1. Introduction

The title of this project is design and develop an automated goods picking machine for convenient store. As the project states, the machine has to be autonomous or unmanned control. It able is able to pick up every types of goods repeatedly from current market shelf and left the goods in the shopping basket. As the machine implemented in the convenient store, the size of the machine has to be compact but able to leave some spaces to let the customers pass by and perform in good conditions. With the small sizes and stable output performance, the machine can help the store owner to reduce the human traffic flow and reduce the chances to infect the COVID-19 as well.

The project team consists of 3 members, which are Jeff Fong Jun En, Kau Hui Ting, and Chuah Ding Ken. The project is divided into 3 sections, Mechanical, Electronics, and Programming. The distribution of the tasks among group members are written as below:

Table 1: Task Distribution Among Members

Team Members	Jeff Fong Jun En	Kau Hui Ting	Chuah Ding Ken
Responsibilities	Mechanical design	Programming	Electronics Design

Due to the location issues as two teammates are in Penang, the project fabrication is done in Penang. So, to this case, the mechanical designer also needs to do the programming so the manpower can be utilized. Meanwhile, the team member is doing multitasking.

1.1Problem statement

Back to the normal days when we enter a convenient store, we walked through aisles just to search for and purchase some daily supplies that we would need. Typically, we as consumers tend to make 1-2 grocery trips per week. With the arrival of current era, we would desperately want instant gratification and are increasing pressed for time when it comes to shopping groceries and other daily supply goods. With the COVID-19 pandemic, social distancing becomes the number one important factor when it comes to buying goods. This situation seems more obvious when buying goods inside a convenient store, where customers sometimes need to squeeze between people along the narrow aisle just to look for the items they want. As a result, social distancing becomes more challenging when more customers are looking for the same items in the same aisle at the same time.

In our current Mechatronics Project, a proposal is made to design and develop a goods picking machine which will be used in a convenient store. The entire process of the goods picking machine will be done in automated way, in order to minimize the contact between customers. This proposed project aspires to create a different and unique experience for customers when purchasing goods inside a convenient store.

1.2 Aims and Objectives

The aim of the project is to design and develop a machine that will be able to pick the specific type and amount of goods into the shopping basket. Several minor objectives are set while planning this project:

1. The machine is compact and reliable.
2. The system can receive the input from user through wireless setup.
3. The machine can pick different type of items.
4. Able to pass the items to user after pick.

2. Literature review

This section discusses the type of automated goods picking machines which are available and implemented in the industry sector.

2.1 Walmart Alphabot

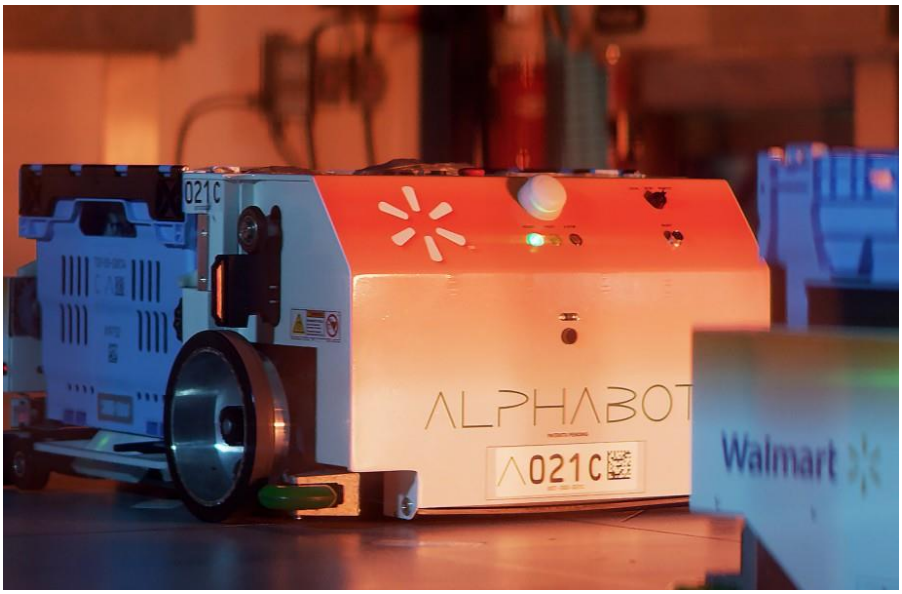


Figure 1: Alphabot used in Walmart

Walmart Alphabot is a cart robot that worked in Walmart warehouses. The robot is small in dimensions, and able to travel between the racks with different groceries. Alphabot is unmanned control robot operated on three axes. Alphabot is able work with real time data sharing, which is a big benefit as it makes the robot more flexible and more intelligent. For example, the refrigerated items, drinks, cans etc. The system operates in a 20000 square feet warehouse. After the machine receives the order and picks out the items, the machine will transport the goods to human workstation to let the workers pack the goods and pass them down to their customers.

2.2 IAM Robotic

Figure 2: IAM Robots used in Convenient Store

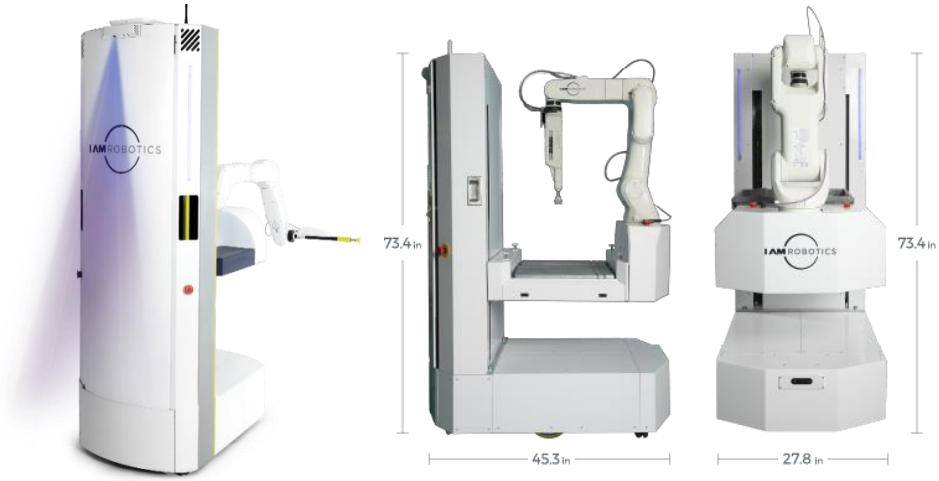


Figure 3: Dimension of the Robot

Swift is a fully automated, battery powered, product pick robot. The machine dimension is around 73.4 inches height, 27.8 inches width and 45.3 inches in length. The robot has 7-axis where provided more features and more probabilities in different and complex environments. Swift has a lift which able to lift the robot arm going up and down as increase the moving scope. By integrated with object detections technologies, Swift able to navigate to the direction and pick different shape of groceries safely and accurately. Yet, swift can pick the groceries from both sides of walk path and the shelves level from ground to two meters height, Other than that, the machine can operate the picking speed same as human level speeds. As the result, the machine is very efficient in progress.

2.3 Magazino

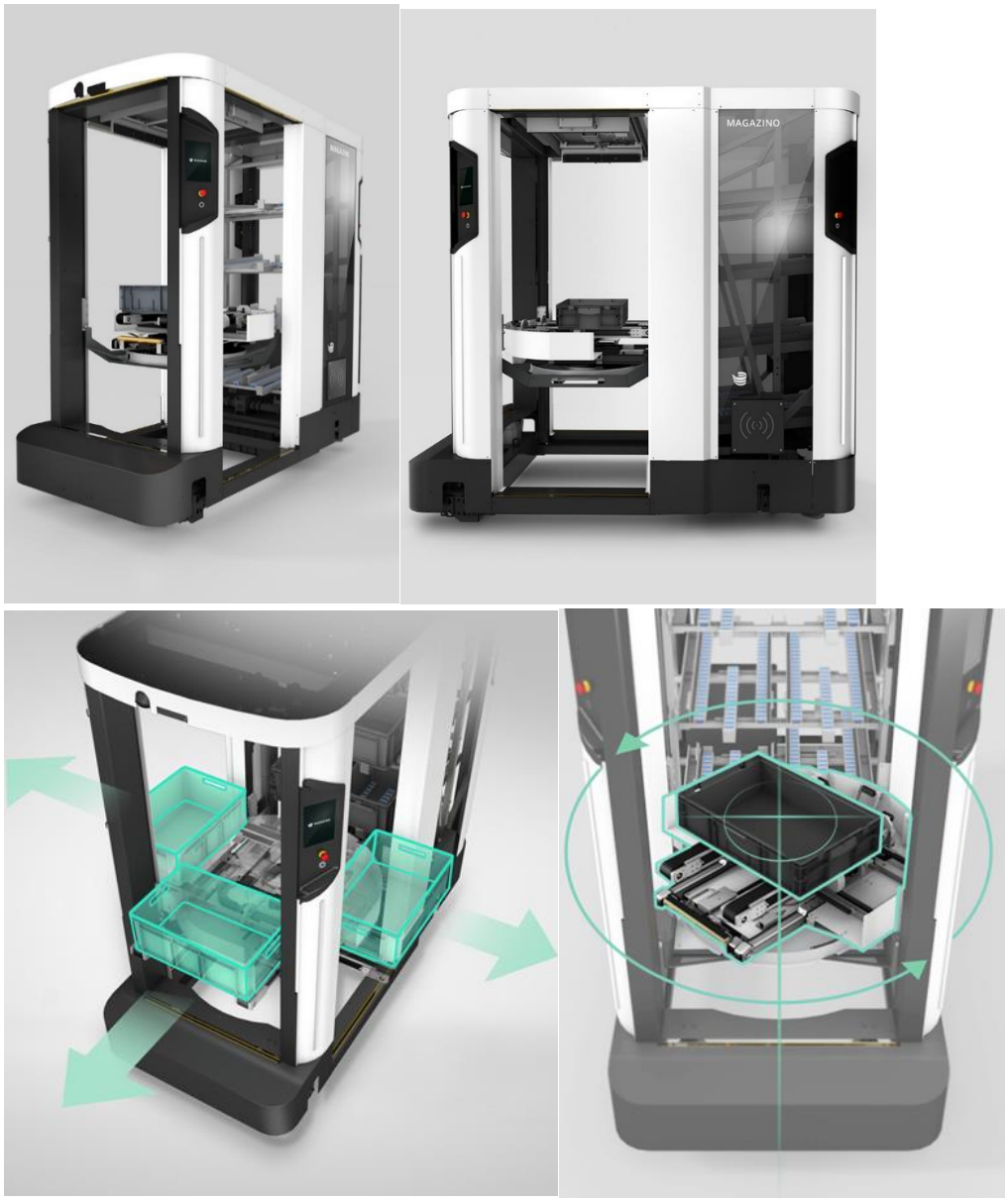


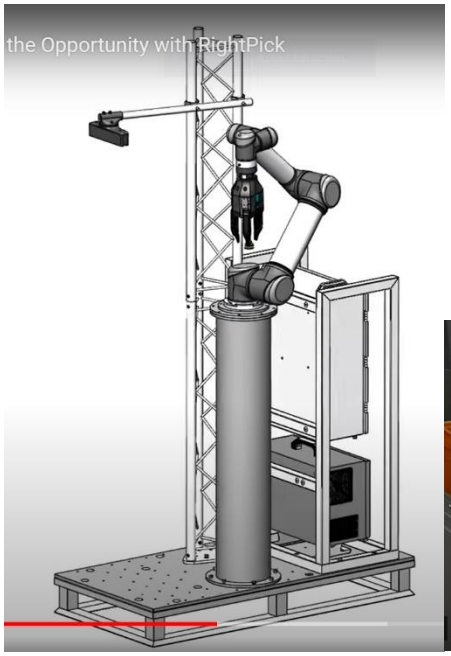
Figure 4: Mechanism of Magazino

This automated robot is developed by MAGAZINO, a robotic company. The product named SOTO 2 that can carry a load up to 20kg and maximum of material size 600 x 400 x 320mm (L x W x H). SOTO 2 is a robot that bring the various materials in the industrial production lines. The robot can be monitoring by software and keep track on real time. SOTO 2 has multiple pick up/unload area and allows turn the materials in 90˚ to 180˚. Other than that, SOTO 2 has variable capacity which depending the material sizes. This feature brings the benefits that utilizes the existing storage spaces and more efficient.

SOTO 2 has the open storage and open gripper pick up area. The design allows the employees to put the materials or unload the material manually if unexpected cases happen. The machine has all dimensional drive wheel which make the robot able access to every corner and move in the tight spaces. SOTO 2 did implement laser scanner at the front and the rear to stop the machine immediately when scanned workers.

2.4 RobotPick

Figure 5: Mechanism of RobotPick



RightHand Robotics (RHR) is robotic company which introduce RightPick, a solution which solve the robot arm items picking process. The design is integrated with industrial level robot arm. The design can pick and place the items of wide ranges. The robot implemented RightPick.AI which is a image processing AI software that drastically improve the robot picking time and accuracy.

2.5 Literature summary

After went through the research and review the idea. The mechanical designer had selected certain features as shown:

1. The machine has X, Y and Z machine movement.
2. Machine can move robot arm up and down, right and left in two axes.
3. The robot arm has one axis to turn and put the items in basket.
4. Basket did not move with robot arm.
5. The design can move parallel with the shelf.

The reason to not make the basket move with robot arm is due to shortage of budget limited the team cannot purchase the high output of motor to lift the basket with robot arm. Other than that, although the multiple axes of robot arm can help the project design be compact and high freedom to pick item at every shelf. To solve the problem, the machine

3. Project management and finance

3.1 Timeline

Table 2: Project Timeline

Week 2	Mechanical Design of Shelf and Draft of Overall System using CAD	Block Diagram and Circuit Design	Design of Flowchart
	Concept Proposal	Literature Review	Flowchart Literature Review
Week 3	Mechanical Parts Design	Choosing of Board (Arduino, Raspberry Pi) + Circuit Design	Modify the Flowchart
	Literature Review		Flowchart Literature Review + Checking with lecturer
Week 4	Mechanical Design + Assembly	Circuit Design (Keypad + LCD Screen)	Testing some code
			Checking with lecturer
Week 5	Mechanical Design Modification + Calculations	Circuit Design (Motor + Servo Motor)	Implement programming code
			Checking with lecturer
Week 6	Mechanical Design Modification + Calculations	Circuit Design and Fabrication	Code debug
	Component Procurement	Mid-Sem Presentation	Checking with lecturer
Week 7	Mechanical sourcing and fabrication	Circuit Fabrication	Code optimization
	GUI Design of software		Checking with lecturer
Week 8	Mechanical sourcing and fabrication	Circuit Fabrication	Implement user interface
			Discuss the GUI with groupmate
Week 9	Mechanical sourcing, fabrication and assembly	Sensor Calibration of the Gripper Arm	User interface calibration
			Discuss the GUI with groupmate

Week 10	Mechanical assembly	Sensor Calibration of other Components	Code Development and Checking
Week 11	Mechanical assembly + Test with electrical components	Sensor Calibration of other Components	Code Development and Checking
Week 12	Mechanical assembly + Test with electrical components	Sensor Calibration of other Components	Code Development and Checking
Week 13	Troubleshooting of the Mechanical Part	Electronic Troubleshooting	Program Troubleshooting
Week 14	Final Report Submission and Presentation	Final Report Submission and Presentation	Final Report Submission and Presentation
Week 15	Reflective Report	Reflective Report	Reflective report

3.2 Cost

Table 3: Estimated Budget for the Project

No.	Item	Price per Item (RM)	Quantity	Total Price (RM)
1	GT2 Idler Pulley Bearing (20T W: 6mm, ID: 5mm teeth)	5	3	15
2	GT2 Pulley Double Head	12	1	12
3	GT2 Pulley 5mm	5	2	10
4	GT2 Pulley 8mm	5	2	10
5	GT2 Open End Timing Belt (6 mm width, pitch 2 mm) 2m	7.5	2	15
6	GT2 Close End Timing Belt (6 mm width, pitch 2 mm) 0.3 m	7.9	1	7.9
7	GT2 Close End Timing Belt (6 mm width, pitch 2 mm) 0.158 m	5.5	1	5.5
8	GT2 Close End Timing Belt (6 mm width, pitch 2 mm) 0.852 m	11.5	1	11.5
9	T8 lead screw(x2) + 5*8 Shaft Coupling (x2) + KP 08 pillow bearing (horizontal bearing)	32.85	2	65.7
10	KFL 08 Pillow Bearing (radial bearing)	3.8	1	3.8
11	L shape bracket (75*75 mm)	5.5	1	5.5
12	L shape bracket (20*20 mm)	1.7	1	1.7
13	Aluminum ruler (30 cm)	2.3	2	4.6

14	Plastic wheel for SPG30/SPG50 (80 mm)	5	6	30
15	3D Printer roller	46.4	1	46.4
16	Aluminum Profile 2020 connector set	1.8	2	3.6
17	Aluminum Profile 2020 (1.85 m)	18/meter	1.85	33.3
18	Aluminum Hollow Bar (1 sq. in) (2.22 m)	23.7/6.1meter	2.22	8.54
19	Aluminum Connector 2-way (1 sq. in)	0.8	8	6.4
20	Socket cap screws – M3*8 (x15) + M3*12 (15) + M5*12 (x30) + M5*25 (x30)	13.05	1	13.05
21	M5*15 (Crosshead Screw) (x20)	2.7	1	2.7
22	Gripper	20.9	1	20.9
23	cardboard box (260*150*180mm)	5	3	15
24	DC motor bracket	5.5	1	5.5
25	line following IR module	2.5	2	5
26	IR analog sensor	19.8	1	19.8
27	5V buck converter/ voltage regulator	0.6	1	0.6
28	Power Supply 12V 10A	22	1	22
29	DC motor 12V 66RPM	27	1	27
30	DC motor driver	2.7	1	2.7
31	Servo motor TD8120MG	45	1	45
32	Arduino UNO prototyping shield R3	14	1	14
33	Veroboard 6*14.5cm	2	1	2
34	ultrasonic sensors	3.2	2	6.4
35	red push button	2.5	1	2.5
36	connectors (housing, pin and header)	11.7	1	11.7
37	Arduino mega	36.5	1	36.5
38	Bluetooth module	14	1	14
39	Stepper Motor Nema 17HS8401	37.9	2	75.8
40	Stepper Motor Driver A4988	5.5	2	11
41	Servo motor MG996R (gripper)	16.5	1	16.5
42	Acrylic customize parts	70	1	70
43	shipping fees	116.78	-	116.78
			Total	817.02

As according to the price list, the budget spent is overall RM 817.02. The prices inexpertly exceeds more than the budget we set. This is happened due to the material we purchased are in local during the project development in pandemic season, where the shipments may delay. This will be the main reason as the team decided to avoid, also due the late of design project building.

4. Methodology

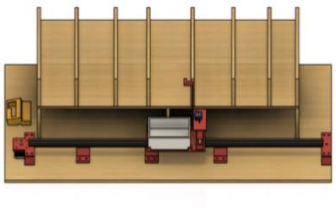
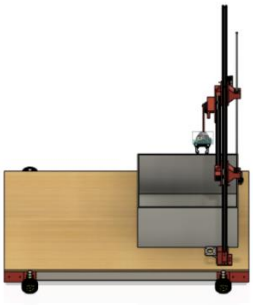
4.1 Mechanical Design

4.1.1 Overview of design

Figure 6: Overview of the Design from Different Angles

4.1.2 Process of mechanical design

Table 4: Design Prototype Timeline

 *Figure 7 1st design*	 *Figure 8 2nd design*	 *Figure 9 3rd design*	 *Figure 10 Final design*

The table above showed the generation of the design change. Before entered the final design, the designs are changed vigorously. By reading the design project, the team are very confused at the design needed. After that, the team group some ideas and proposed solutions.

The proposed solutions that brainstorm:

1. System able to tackle both automated and traditional shopping way.
2. Need to consider the size, shapes and the type packaging of the groceries.
3. User interface - item image will show on the screen and let the customer to pick.
4. Minimize potential safety risk.
5. Types of items in convenience store (shape, size, packaging, material).
6. The item will be gripped by gripper / pushed out by push rod / suction.
7. Groceries will take by robot arm and drop into basket.

The first 3 design is just fulfilling the basic requirement which is “autonomous”. A lot of the propose solutions did not overcome. Yet, after do the researches and consulting the first meeting with lecturer, the concept is improved a lot and come out the final design. To have a picture of the idea, the mechanical designer out by cad software.

4.1.3 Mechanical system design criteria

During the design state, there are certain parts need to take note importantly during the mechanical system.

4.1.4 Calculation

Stepper motor

Table 5: Mass of the Gripper Arm which the stepper motors need to withstand

Mass of component	KG
Stepper motor	0.35
Robot arm	0.3
Aluminum profile 2020	0.53
Aluminum 1in square hollow bar	0.064
Acrylic plate	0.2
3D printed parts	0.074
Weight of groceries	0.6
Total	2.118

Moving speed target

Table 6: Moving speed of the mechanism in x and y-axis

Axis	Driving method	Stepper driver	Homing check	Grabbing item progress (mm/s)	Put item to basket (mm/s)
X-axis	Pulley	A4988	10	80	80
Y-axis	Thread rod	A4988	10	100	80

To choose the correct stepper motor for the design, the torque and RPM of the machine required need to calculate out and compare. For X-axis motor, the design used pulley belt system. Then, the formula provided from Norwegian Creations (Aasvik 2015).

X-axis torque

Mass = robot arm + aluminum 1-inch hollow bar + 3D printed parts + weight of groceries

Table 7: Estimated Torque Developed in the x-axis

Mass = 1.038kg	F = mg	g = 9.81 ms^{-2}
T = rF	Radius of GT2 pulley = 6.35mm	

$$F = 1.038 \times 9.81$$

$$F = 10.62N$$

$$\tau = 0.00635 * 10.62$$

$$\tau = 0.0674437 \text{ Nm}$$

From the stepper datasheet (?), the holding torque of NEMA 17HS8401 is 52 Ncm, which is approximately 0.52Nm.

From the calculation, the torque of x-axis needed is 0.0649Nm. Base on the result, the motor is more than enough to moving the part with more weight.

X-axis step per mm and step per second and RPM

$$Belt\ (\frac{Revolution}{mm}) = \frac{Srev * fm}{p * Nt}$$

S_{rev} is the number of steps per revolution for the motor
f_m is the micro-stepping factor (1, 2, 4, 8 etc.)
p is the pitch (e.g. 2mm)
N_t is the number of teeth on the pulley attached to the motor shaft.

$$Belt\ (\frac{Revolution}{mm}) = \frac{200 * 1}{2 * 20}$$

$$Belt\ \left(\frac{Revolution}{mm}\right) = 5\ step\ per\ mm$$

Yet, this calculated step per millimetre did not proved in testing. Hence, the real outcome will be defined after testing and also the analysis will be discussed.

$$(\max machine\ Speed)\frac{mm}{second} * (Stepper\ Motor)\frac{Step}{mm} = \frac{Step}{Second}(Hz)$$

Targeted speed and frequency

Table 8: Speed of the machine which affects the moving belt speed.

Maximum machine speed (mm/s)	Frequency (fz)	RPM (revolution per minute)
10	50	0.25
30	150	0.75
50	250	1.25
80	400	2

To find the number of turns to move the belt 1mm, the number of steps per revolution is 200m. The factor of micro stepping of the driver is 1. While the pitch of pulley belt is 2mm and the tooth of pulley is 20. As the result, the X-axis motor is move in 5 step/mm. Therefore, to find the suitable frequency for maximum frequency, the second formula had used. Hence, the predicted frequency had used to find the RPM of the motor.

Y-axis part

$$F = mg$$

$$g = 9.81\, ms^{-2}$$

$$F = 2.118 * 9.81$$

$$F = 20.777N$$

radius of T8 thread rod shaft = 0.008m

$$\tau = 0.1662Nm$$

Y-axis step per mm and step per second and RPM

$$Leadscrew\left(\frac{Revolution}{mm}\right) * \frac{1}{Microstep} * Motor\left(\frac{Step}{Revolution}\right) = \frac{Step}{mm}$$

$$2\left(\frac{Revolution}{mm}\right) * \frac{1}{1} * 200\left(\frac{Step}{Revolution}\right) = 400\frac{Step}{mm}$$

//should get 100, but calculation get 400, result 25

$$(\max machine\ Speed)\frac{mm}{second} * (Stepper\ Motor)\frac{Step}{mm} = \frac{Step}{Second}(Hz)$$

Table 9: Maximum Machine Speed and its frequency in y-axis

Maximum machine speed (mm/s)	Frequency (fz)	RPM (revolution per minute)
10	4000	20
50	20000	100
80	32000	160
100	40000	200

To find the number of turns to move 1mm, the leadscrew revolution per mm has been set to 2 turns to move 1 mm. The factor of micro stepping of the driver is 1 and motor step per revolution is set to 200. As the result, the Y-axis motor is move in 400 step/mm. Therefore, to find the suitable frequency for maximum frequency, the second formula had used. Hence, the predicted frequency had used to find the RPM of the motor.

Servo motor

MG996R

Referred to Electronicos CALDAS, the specification of MG996R has been written in the table below.

Table 10: Electrical Specifications of the MG996R Servo Motor

Operating Voltage	4.8V	6 V
Idle Current	mA	mA
No Load Speed	0.17 sec/60˚	0.14 sec/60˚
Running current	500mA	900mA
Stall torque	9.4 kg cm	11kg cm
Working Voltage Range	4.8-6V	

TD 8120MG

Referred to SunFounder Steam Education (2019), the specification of TD-8120MG has been written in the tale below.

Table 11: Electrical Specifications of the MG996R Servo Motor

Operating Voltage	4.8V	7.2 V
Idle Current	mA	mA
No Load Speed	0.18 sec/60˚	0.14 sec/60˚
No-Load Current	210mA	260mA
Torque	1kgcm	21.8kgcm
Stall Current	2100mA±10%	2700mA±10%
Working Voltage Range	4.8-7.2V	

Weight of robot arm

Table 12: Weight of Robot with different parts joined together

weight of robot arm (from rotating part to gripper, gram)	**g**
Weight of 3d printed parts	35
Servomotor MG996R	55
Gripper	56
Weight of aluminum hollow (200mm)	74
Weight of proximity sensor (assume)	10
Total of estimated weight	230

For MG996R:

$F = mg$	$\tau = rF$

$$F = 0.6kg\ x\ 9.81\ ms^{-2}$$

$$F = 5.886N$$

Radius of gripper = 8cm (approximate)

$$\tau = rF$$

$$\tau = 0.47088Nm$$

The stall torque of TD8120MG is 9.4kgcm which is around 0.92Nm. Assume that the rated torque is 10% lesser than stall torque, which is 0.83Nm.

Table 13: Theoretical torque value vs Practical torque value of Servo Motor TD8120MG

Torque from datasheet	Torque from calculation
0.83Nm	0.47088Nm

For TD8120MG

$F = mg$	$\tau = rF$

$$F = 0.73kg\ x\ 9.81\ ms^{-2}$$

$$F = 8.1423N$$

Length of arm = 21cm (approximate)

$$\tau = rF$$

$$\tau = 1.709Nm$$

The stall torque of TD8120MG is 20.5kgcm which is around 2.011Nm. Assume that the rated torque is 10% lesser than stall torque, which is 1.809Nm.

Table 14: Rated Torque vs Calculated Torque of the Servo Motor TD8120MG

Rated torque	Torque from calculation
1.809Nm	1.709Nm

DC motor

Sum of mass = 2 stepper motor + plywood + sum of weight of profile 2020 + sum of aluminium hollow bar 1inch square + dc motor + sum of pillow bearing

Sum of weight = 5.2kg

The net masses of design are 5.2kg, assume that the design is carried the maximum load around 2kg. Hence, the total mass during machine operation is around 7.2kg.

After that, to find out the acceleration of the machine to move in horizontal, kinematic equation shown in appendixes is used.

$$s = ut + \frac{1}{2}at^2$$

The parameters of the environment which are, the travel distance is 1m, the time taken targeted is 5 seconds.

After substitute the parameter into the equation, the acceleration is 0.08 ms^{-2}. After that, repeat the process with different targeted time in the table. After that, the following equations were used to find the speed of rotation and torque of the DC motor needed to move the machine.

$$Speed\ of\ rotation, n_{in} = \frac{60}{\pi} \times \frac{v_L}{d}$$

$$Torque, M_{in} = \frac{d}{2} \times \frac{F_L}{\eta}$$

Knowing that the diameter of wheel is 80mm, v_L is loaded velocity, F_L is load force and efficiency, η is been assumed 0.6. Moreover, the design used pulley belt system to power the shaft, it can be ignored as there is no gear ratio. Based on the kinematic equation, the load velocity and load force are founded and wrote into the table. Hence, the speed of rotation and torque can be calculated.

Table 15: Kinematic Calculation of the Mechanism

Targeted time (s)	Acceleration	F = ma (N)	Velocity (ms^-1)	Torque (Nm)	RPM
5	0.08	0.576	0.4	0.0384	95.49
8	0.03125	0.225	0.25	0.015	59.68
10	0.02	0.144	0.2	0.0096	47.75
12	0.01388	0.099936	0.16656	0.0066624	39.76

After went through the dc motor calculation, the team decide to use the second calculated data to select the dc motor. There is several reasons as the machine has to move in fast speed in order to finish the picking process quickly. Other than that, the machine needs to work in balanced of torque speed characteristic as enough torque to move the machine forward and backward after picking goods.

DC motor rated data

Model:JGB37-520		Data Sheet							
Voltage		No Load		Load Torque				Stall	
Workable	Rated	Speed	Current	Speed	Current	Torque	Output	Torque	Current
6-15v	12v	960	100	672	200	0.1	0.8	0.4	1.2
6-15V	12V	600	100	480	200	0.15	0.8	0.6	1.2
6-15V	12V	319	100	255	170	0.28	0.8	1.1	1.2
6-15V	12V	200	60	160	150	0.45	0.8	1.8	1.2
6-15V	12V	107	60	85	150	0.84	0.8	3	1.2
6-15V	12V	66	60	52	100	1.35	0.8	5	1.2
6-15V	12V	45	60	36	100	1.9	0.8	7	1.2
6-15V	12V	35	50	28	100	2.5	0.8	10	1.2
6-15V	12V	22	50	18	80	4	0.8	16	1.2
6-15V	12V	12	50	10	80	7.5	0.8	30	1.2
6-15V	12V	7	50	5.6	80	12	0.8	48	1.2

Compare the rated RPM and Torque in datasheet where located in appendix, the suitable dc motor according to reference table is the one operated with 66 RPM at rated voltage 12V and produced 1.35kgcm ~ 0.1324 Nm. We choose this dc motor because the distance travel by design is 1 meter, which is did not required a dc motor with higher rpm to do. After that, the torque of dc motor is enough to carry more weight in the future after the further upgrade is done.

4.1.5 Making procedure

Step 1: Base making

Figure 11: MDF wood plate

Figure 12: 1 inch aluminum hollow bar

Figure 13: Hollow bar connector

Figure 14: Hollow bar cutting

A MDF plate which have the thickness of 12mm and excess sizes been found by teammate. Hence, the size of the MDF plate been cut and sand. After that, to make the frame of the base, the team use 1-inch aluminum hollow bar and rivet them together by the L shape connector.

Step 2: Aluminum Hollow support bar and Aluminum profile 2020 assembly

Figure 15 Front view

Figure 16 Side view

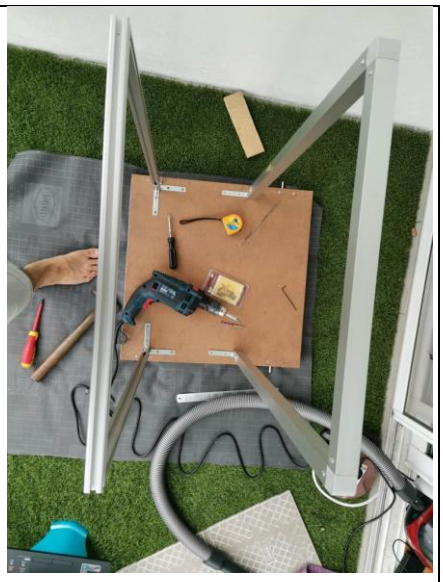


Figure 17 Top view

The precut aluminum profile 2020 is connected with connector set as the shape of U.At the bottom side, 2 of 75*75 mm L-shape bracket been connected by bolts and nuts on the profile. After that, the bracket will screw on the MDF plate by wood screw.

Figure 18 Aluminum profile 2020 connection

Figure 19 Aluminum hollow bar connection

Before screwing the bracket to the wood plate. The distance have to measure. The Aluminum profile 2020 have a distance, 65mm from the front edge, and 20mm from the side edges. After that, the making need to mark to prevent the precision shifted. Therefore, one guy helps to hold the aluminum profile 2020 and acting some forces on it so that the other guy can screw it. After that, the aluminum hollow bar stayed a distance of 290 mm from the aluminum profile. After that, the installation steps been repeat to screw the aluminum hollow bar.

Step 3: Metal bar connetion

Figure 20 Rivet on Aluminum hollow bar

Figure 21 Connected view

After that, the metal bars are ready to connect. At the hollow bar side, the rulers are using rivet method. At the aluminum profile side, the metal bars are use bolts to screw into the profile holes.

Step 4: Y axis motor connection

Figure 22 Stepper motor mount

Figure 23 Stepper motor connected to profile

At the middle stepper motor mounting, it is drilled a 5mm diameter hole, to screw to the profile as shown. After the stepper motor bracket is screwed, the next step is mount the stepper motor to the bracket by screw the M3 screws. After the following step done, the stepper motor should be look like the figure below.

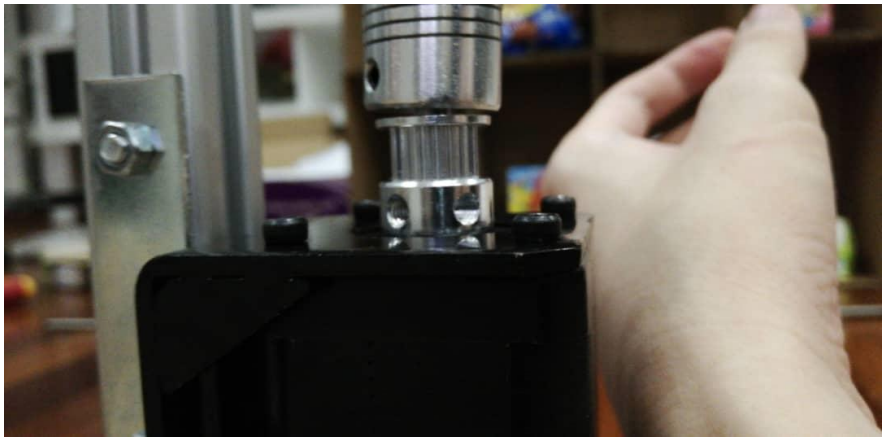


Figure 24 Pulley and coupling on the shaft

Step 5: Driven thread rod and driven pulley connection

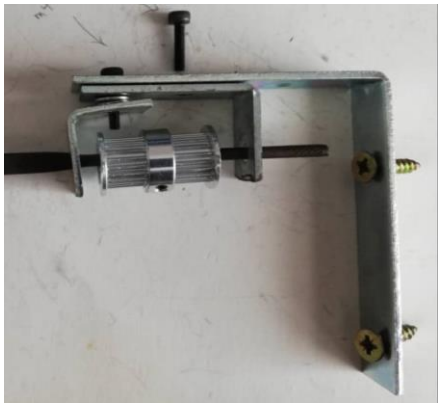


Figure 25 Double head pulley connection view

Due to the limited length of the belt, the alternative that overcome with lecturer while meeting is make an idler driven pulley between two belts. The picture above is the view of idler pulley bracket before fully assembled.

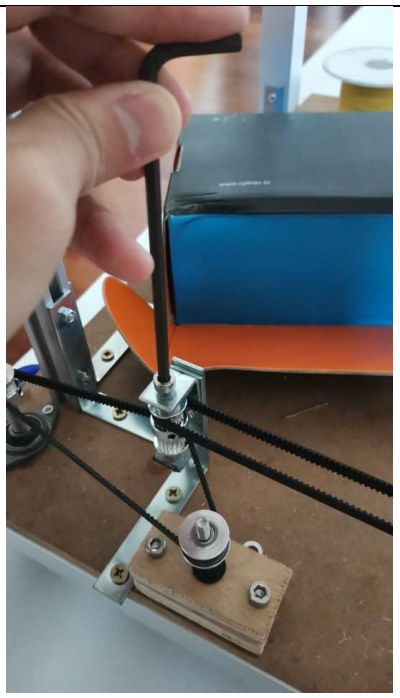


Figure 26 Double head pulley tighten

Figure 27 Driven pulley and belt installation

After the pulley bracket been put between the belts, and yet, another driven pulley is put to make sure the belt is tight enough. After that, at the driven thread rod, a flat pillow bearing is adjusted and directly screw on the MDF board. Before the thread rod lock in the pillow, the pulley and the belt put into thread rod.

Figure 28 Final view of driven belt

After the belt connection done, the final outcome should be look like the figure above.

Step 6: X axis motor assembly

Figure 29 Left side acrylic assemble

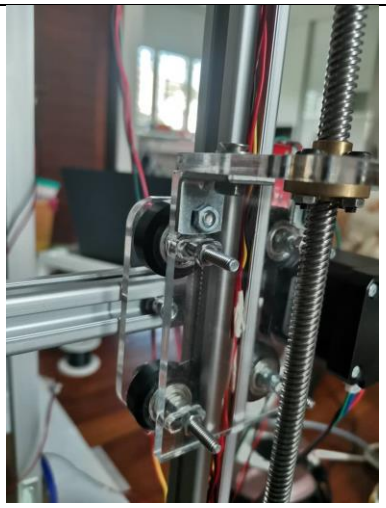


Figure 30 Right side acrylic assemble

At the first point, the acrylic parts are in pieces. The horizontal acrylic parts need to mount particularly and strengthen it with the small L bracket. After that, the rollers screw through the laser engraved hole which the nuts in between the wheels and the acrylic parts. This ensure the rollers is fit in the middle also made the acrylics have a distance from profile 2020. Therefore, the tightness of the rollers can be adjusted at the range of slots.

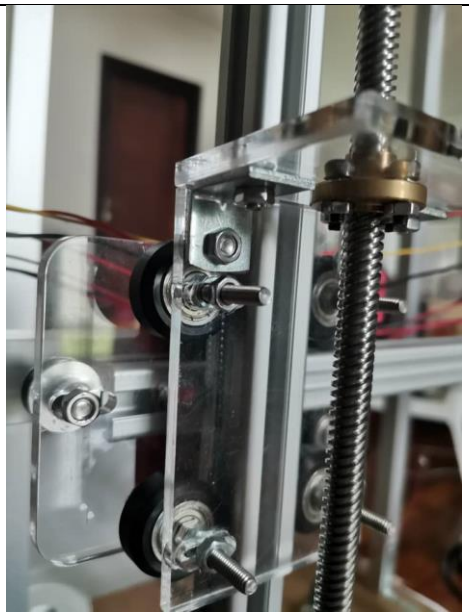


Figure 31 Driven pulley side

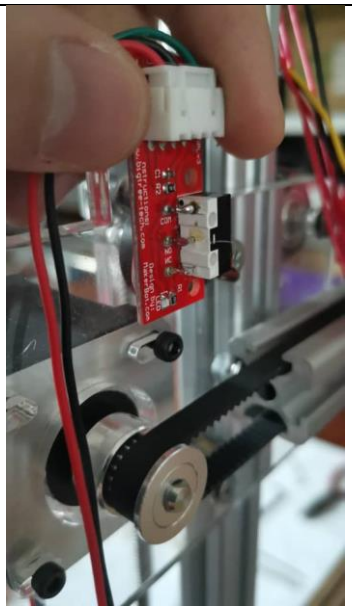


Figure 32 Stepper motor side

After that, the front plate, which is driven pulley side and stepper motor side is connected with a driven pulley and stepper motor respectively. After that, the pulley is screw on the shaft. Moreover, the end stop switch mounted on the stepper motor acrylic by hot glues.

Step 7: Gripper Assembly

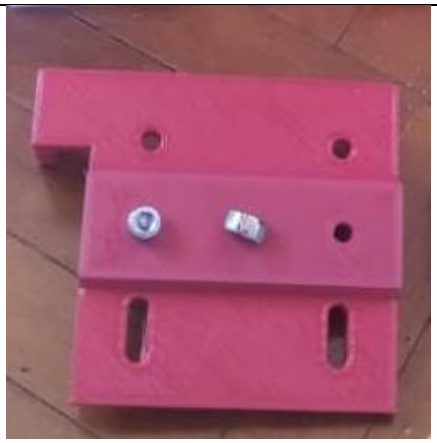


Figure 33 Servomotor mount plate

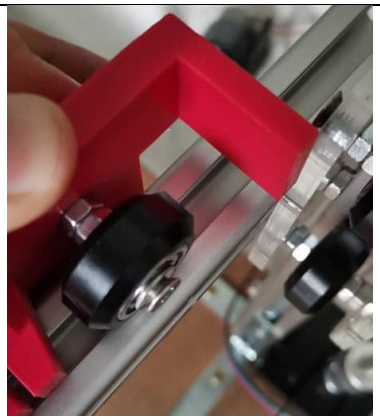


Figure 34 The roller connection

After the X axis done, the next step is connecting the roller to the servo mounting. The roller must at the middle of the aluminum profile. To do that, the way is to screw 2 nuts then lock the wheels. Then repeat the steps as the other wheels.

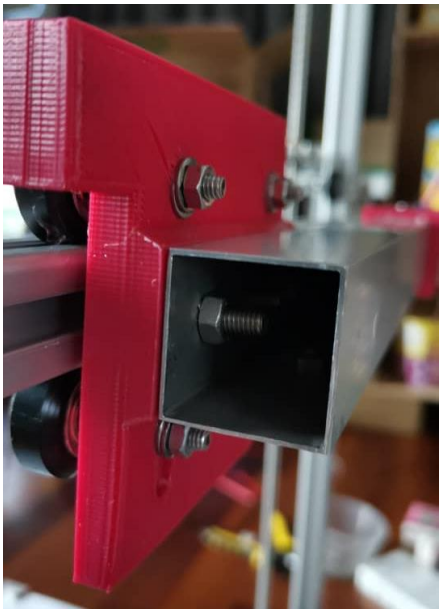


Figure 35 Screw the aluminum hollow bar on servo mount

Before the servo mount install on the X axis profile bar, the aluminum hollow bar is drilled to screw on the servo mount. The 3 screws is put through the servo mount, the nut is locked in the hollow bar.

Figure 36 Pulley belt tighten behind the servo mount

After the hollow bar are install, the servo mount will slide into the profile 2020. Therefore, the belt is pass through the profile 2020 hole. And it will tight at the clip which behind the servo mount.

Figure 37 robot arm first axis installs

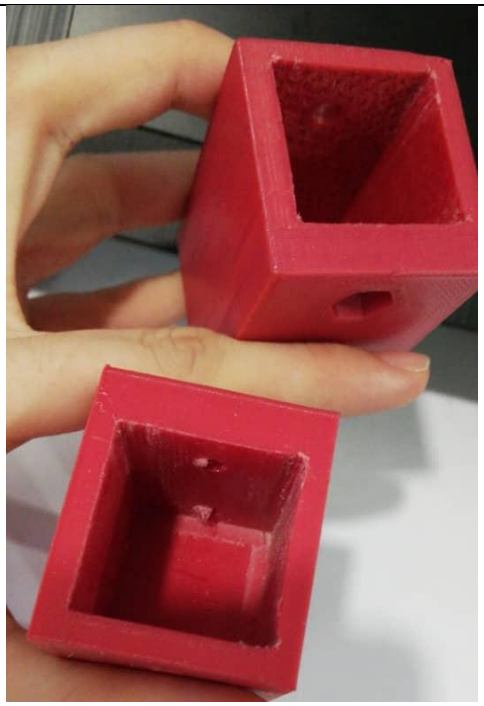


Figure 38 Robot arm connector

At the first axis of the robot arm, the connector is set at the end of the hollow bar. After it is screw, then connect it to the servomotor. After that, the 2 connectors are used for fix the servomotor on the hollow bar.

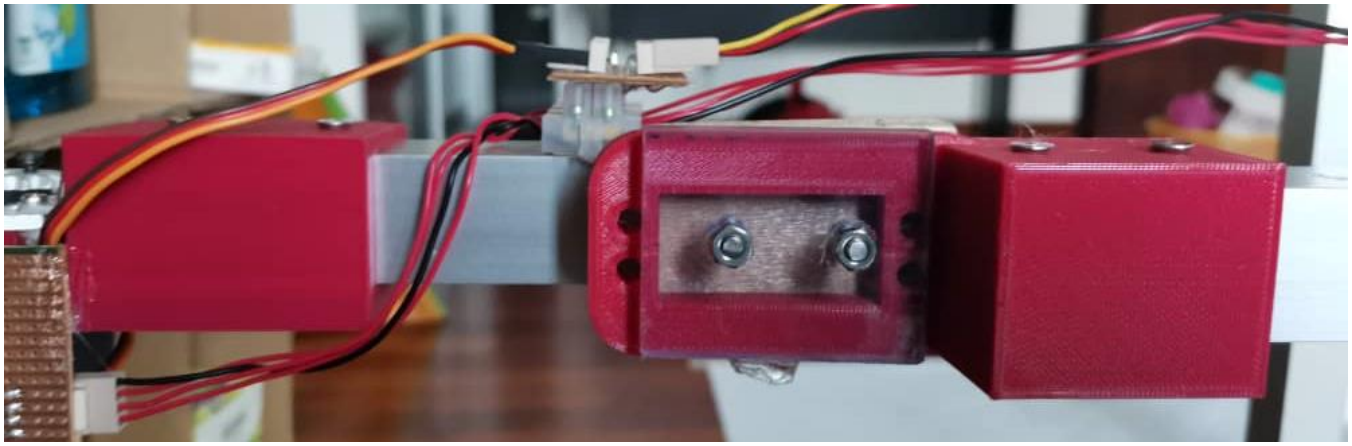


Figure 39 Robot arm

Figure 40 servo gripper modification

The servo motor modification needs to be done as the gripper did not come with the parts expected.

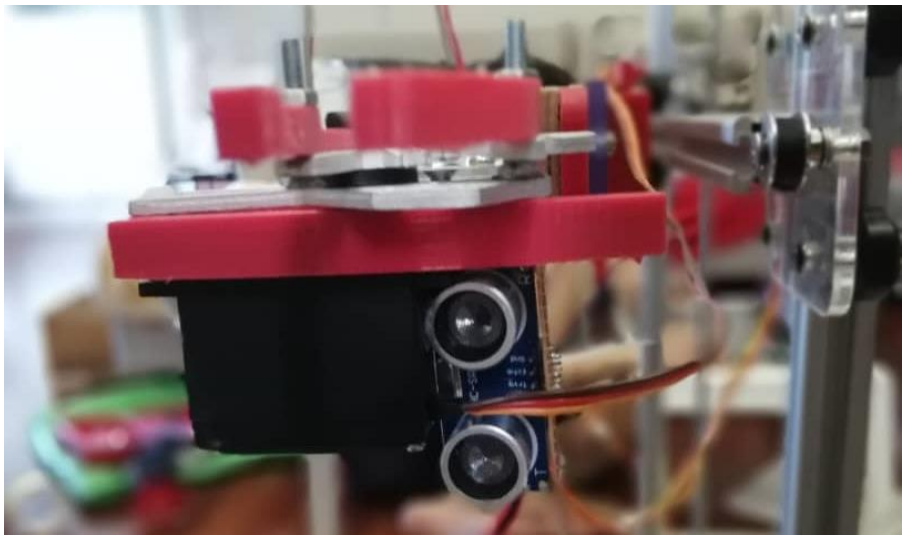


Figure 41 Ultrasonic sensor

At the process of installation, the unknown reason that make the proximity sensor cannot perform what the team expected, hence, the proximity sensor decided to put it aside yet replace the place with one of the ultrasonic sensors. As the ultrasonic sensor size is big, the team agree that put it beside the servomotor. To install it, the hot glue method is used.

Step 8: DC motor assembly

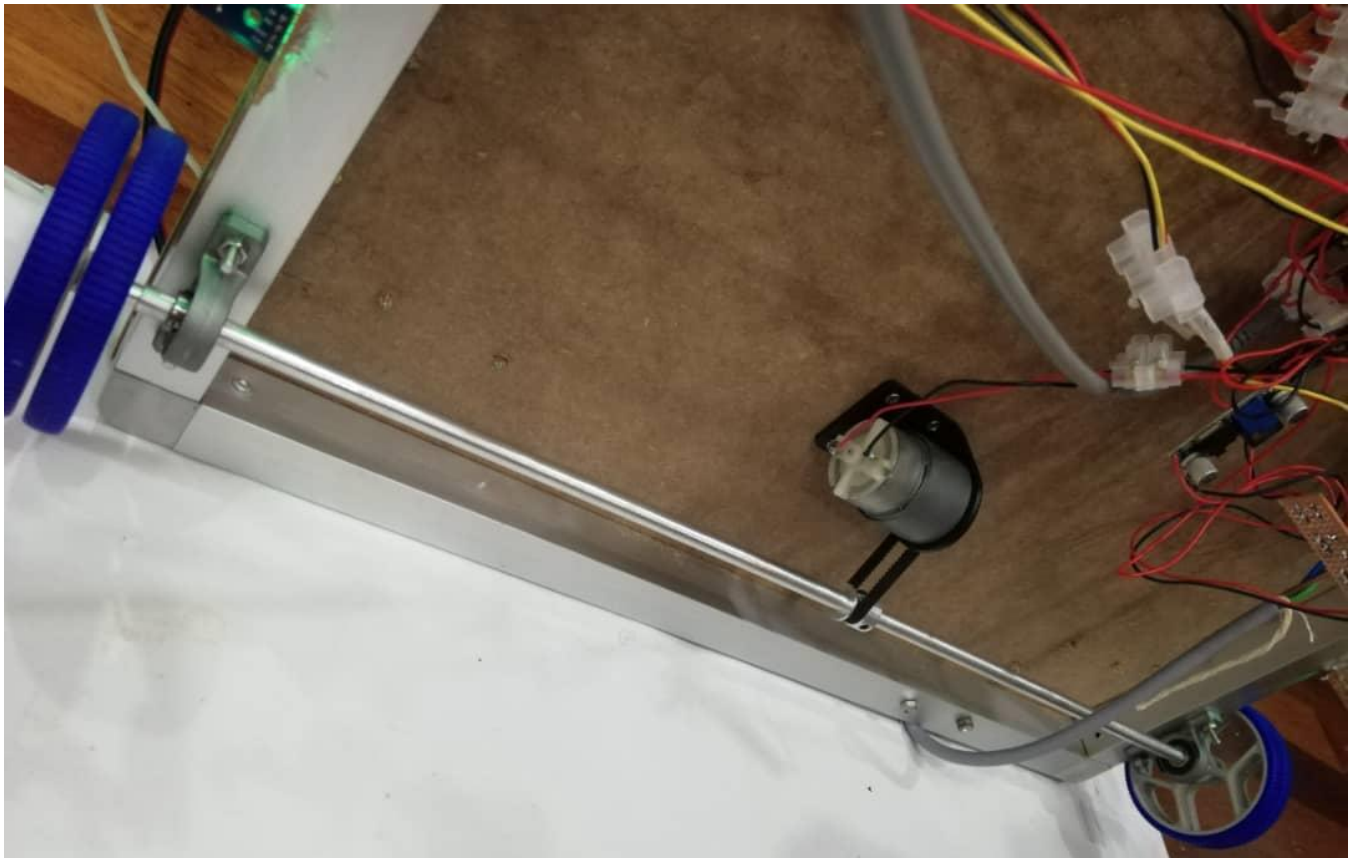


Figure 42 DC motor install

For the DC motor, it is simple to do as screw the mounting first. Hence, the motor is lock on the bracket and connect with pulley and belt to the wheel shaft.

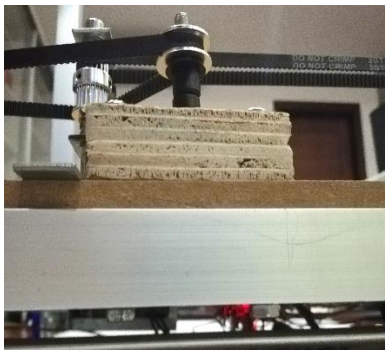
4.2Mechanical Assembly

The team has faced many problems for assembly. One of the problems includes materials sourcing which took very long as price and time to reach are both took into considerations.

There are also lack of considerations during design stage as well as inconvenient excess to some tools and fabrication machines that cause the delay of sourcing, fabricating and assembly. One of the most common problem is about dimensions. Extra fabrications are needed in parts includes hollow bar, acrylics, 3D printed parts, shaft rods, L shaped bars due to incorrect dimensions and missing holes.

Other problems that delays the assembly includes:

Problems	Reason	Solutions
3D printed parts cannot fit into hollow bars and screw dimensions on it changed	shrinking from cooling after 3D printed with high temperature.	Change m4 to m5 sizes screw, nuts and washer.
Gripper assembly with printed parts and servo motor	Insufficient dimension info of gripper, printing orientation mistake	Replace with new printed parts, trial and error with screw, nuts and washer
X-axis acrylic assembly	Missing laser cut holes, incorrect dimension of existing holes	Holes milling, L bracket 20*20 side filing
Base assembly	Hard to align 8 straight holes through MDF board, hollow bar, and bearing for screws	Need 2 person to carefully measure and drill with the help of center punch
Horizontal bearing tightening	Badly design bearings to tighten/ loosen the set screws 	Screw set screws from side way
Y axis belt	Unable to source the exact length needed, circumference of pulleys was not included	Interchange with longer DC motor belt, the extra length is tightened by another pulley

		 *Figure 43 Finding best way and location for the belt with least interference with the basket*
Lead screw can only turn in 1 direction	M4 screw is used as the “shaft” for the double head pulley	M5*55 screw is sourced to hold the m5 bore double head pulley in place, the L bracket 20*20 is also drilled to let the screw pass through
The m5*55 screw is not stable enough to tighten the pulley 	The screw shaft support is too far from the force of the turning pulley	2 wood pieces are stack together with a drill countersunk at the bottom for the screw head while the two sides of the wood are hold down by 2 screws
X axis arm Screwhead blocks the movement of the arm, roller tightening, belt tightening		Cut off the blocking excess part of the 3D parts, drill holes on the acrylic so the roller can tightly fit in its trial of the profile, fit a short section of belt to tighten and prevent slip 

1. Sourcing,
2. Waiting for machining fabrication,
3. Gripper,
4. Machine base,
5. Y axis belt,
6. Tuning angle of X, Y and Z axis parts
7. X axis roller tightening,
8. assembly sequence and ways to calibrate discovering for example: the simplest way for X axis leveling calibration is disconnecting the radial bearing and the thread rod to loosen the belt for tuning,
9. Wire route planning,
10. Exact screw length, washer and nuts are needed for each part so that the moving arm (3D printed part shown below and the servo motor body) does not hit the stationary parts when moving. Trial and errors are done.

Figure 44 Exact combination of screws and washers to let the roller run smoothly to home without blockage

11. Each standing part (thread rods, profile and hollow bars) need to be straight and perpendicular to the base, X axis leveling is made by
12. 4 concurrent pulleys need to be at exact location to be tight but stay parallel with the base at the same time.
13. The location of goods on the rack are also contemplated so that heavier items are at the second row and the machine does not need to carry heavy weight up and down too much for the servo motor to turn the goods down (before redesign of one servo motor).

4.3Electronics and Circuit Design

4.3.1 Electronic Components and Sensors Used

Table 16: List of Electronic Components and Sensors Used

Component	Quantity
12V DC Motor	1
L293D DC Motor Driver	1
12V Stepper Motor	2
A4988 Stepper Driver	2
Servo Motor 5V	1
Red Button Switch	1
Homing Switches	2
Power Supply 12V	1
5V Buck Converter (2A)	1
Arduino Mega Board	1
Strip Boards	2
Prototyping Shield	1
Ultrasonic Sensors SR-04	2
IR Sensors	2
Bluetooth Module HC-05	1

4.3.2 Block Diagram

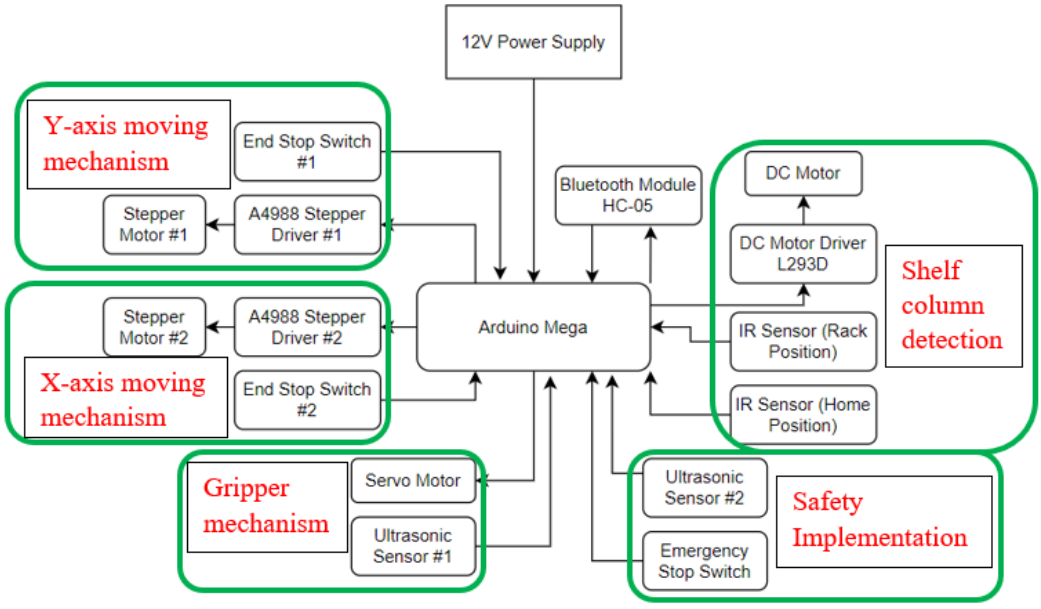


Figure 45: Block Diagram of the Final Design

4.3.3 Components Functions and Connections

L293D Brushed DC Motor Driver

Figure 46: L293D Motor Driver IC

As the voltage rating of the DC motor used is 12 V, thus it cannot be driven directly by the Arduino M. ega Board, or else it will exceed the maximum current sunk by the Arduino board and cause damage to it. As a result, an integrated circuit is required to drive the DC Motor. In this project, L293D is used to isolate the circuit connection between the DC Motor and the Arduino Mega Board as it is able to supply up to a rated current of 1A at the output continuously. Table below shows the specifications of the L293D IC:

Table 17: Current and Voltage Specification of the L293D IC

	MIN	MAX	UNIT
Supply voltage, V_{CC1} (2)		36	V
Output supply voltage, V_{CC2}		36	V
Input voltage, V_I		7	V
Output voltage, V_O	–3	V_{CC2} + 3	V
Peak output current, I_O (nonrepetitive, t ≤ 5 ms): L293	–2	2	A
Peak output current, I_O (nonrepetitive, t ≤ 100 µs): L293D	–1.2	1.2	A
Continuous output current, I_O: L293	–1	1	A
Continuous output current, I_O: L293D	–600	600	mA
Maximum junction temperature, T_J		150	°C
Storage temperature, T_{stg}	–65	150	°C

12V DC Motor

Figure 47: DC Motor used in this project

The DC motor is used to drive the front wheel shaft of the entire system so that the system will move in a straight lines without being accidentally slipped to other angle of direction.

NEMA 17HS8401

The stepper motor used in this project is of model name NEMA 17HS8401, it is a 2-phase hybrid stepper motor. Two stepper motors of same model name are used to move the gripper mechanism in x and y directions.

Figure 48: NEMA 17HS8401

Table below shows the electrical specifications for the stepper motor:

Table 18: Specifications of the Stepper Motor from Datasheet

Electrical Specifications:

Series Model	Step Angle (deg)	Motor Length (mm)	Rated Current (A)	Phase Resistance (ohm)	Phase Inductance (mH)	Holding Torque (N.cm Min)	Detent Torque (N.cm Max)	Rotor Inertia (g.cm²)	Lead Wire (No.)	Motor Weight (g)
17HS2408	1.8	28	0.6	8	10	12	1.6	34	4	150
17HS3401	1.8	34	1.3	2.4	2.8	28	1.6	34	4	220
17HS3410	1.8	34	1.7	1.2	1.8	28	1.6	34	4	220
17HS3430	1.8	34	0.4	30	35	28	1.6	34	4	220
17HS3630	1.8	34	0.4	30	18	21	1.6	34	6	220
17HS3616	1.8	34	0.16	75	40	14	1.6	34	6	220
17HS4401	1.8	40	1.7	1.5	2.8	40	2.2	54	4	280
17HS4402	1.8	40	1.3	2.5	5.0	40	2.2	54	4	280
17HS4602	1.8	40	1.2	3.2	2.8	28	2.2	54	6	280
17HS4630	1.8	40	0.4	30	28	28	2.2	54	6	280
17HS8401	1.8	48	1.7	1.8	3.2	52	2.6	68	4	350
17HS8402	1.8	48	1.3	3.2	5.5	52	2.6	68	4	350
17HS8403	1.8	48	2.3	1.2	1.6	46	2.6	68	4	350
17HS8630	1.8	48	0.4	30	38	34	2.6	68	6	350

Based on the table above, stepper motor NEMA HS8401 has a rated current of 1.7 A. Hence, a stepper motor driver is required to drive the stepper motor.

A4988 Stepper Motor Driver

Figure 49: A4988 Stepper Motor Driver with Heat Sink

The A4988 stepper motor driver is a stepper motor driver which has a maximum input voltage supply of 35 V, which is way more sufficient than the required voltage for the stepper motor NEMA HS8401.

Since the rated current of the stepper motor is 1.7 A as mentioned above, the maximum output current of 2A for the driver is able to drive the stepper motor without damaging the circuit.

Table 19: Specifications of the Stepper Motor Driver A4988 from Datasheet

Characteristic	Symbol	Notes	Rating	Units
Load Supply Voltage	V_{BB}		35	V
Output Current	I_{OUT}		±2	A
Logic Input Voltage	V_{IN}		–0.3 to 5.5	V
Logic Supply Voltage	V_{DD}		–0.3 to 5.5	V
Motor Outputs Voltage			–2.0 to 37	V
Sense Voltage	V_{SENSE}		–0.5 to 0.5	V
Reference Voltage	V_{REF}		5.5	V
Operating Ambient Temperature	T_A	Range S	–20 to 85	°C
Maximum Junction	T_J(max)		150	°C
Storage Temperature	T_{stg}		–55 to 150	°C

Servo Motor TD8120MG

Figure 50: Servo Motor TD8120MG

The servo motor TD8120MG is used to control the swinging position of the gripper arm after grabbing the goods from the shelf. At 4.8 V, it has a no-load current of 210 mA and a stall current of 2.1A.

Servo Motor MG996R

Figure 51: Servo Motor MG996R

The servo motor MG996R is used to control the gripper claw before and after grabbing the goods from the shelf. At 6 V, it has a maximum operating current of 900 mA and a stall current of 2.5 A.

5V Buck Converter 2A

Figure 52: LM2596 DC-DC Buck Converter

The 5V Buck converter model name LM2596 is used to step down the 12 V from the power supply into 5V which in turn drives the servo motor as the maximum current sunk by the Arduino 5V pin is 400 mA. The buck converter module has an input voltage ranging from 4.5 V to 40 V, and an adjustable output voltage from 1.5 to 35 V. The output voltage can be adjusted by turning the potentiometer mounted on the board module.

DC Motor Driver Circuit

After referring to the datasheet and pinout diagram of the L293D, the motor driver circuit is designed, including the DC Motor connecting to the output of the IC.

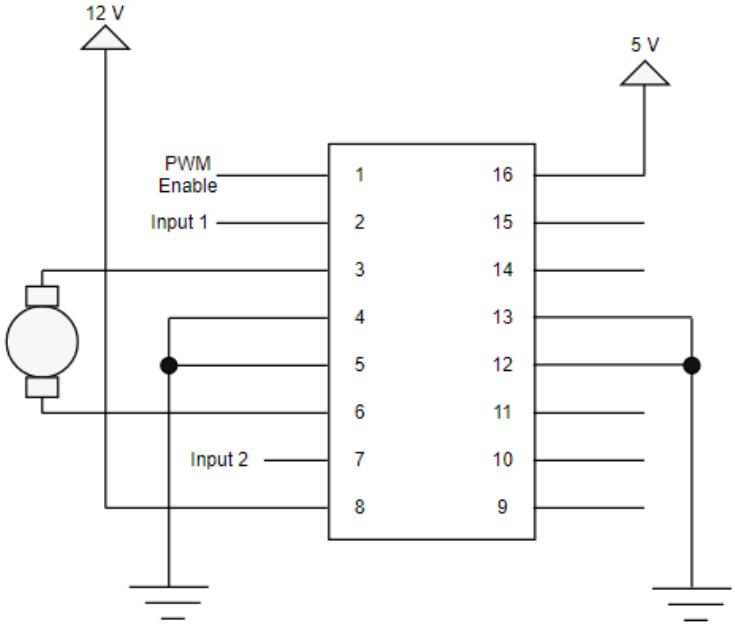


Figure 53: Schematic Diagram of the Motor Driver Circuit

The pinout diagrams is tabulated below:

Table 20: Pinout Connection of the L293 Motor Driver

Pins	Connected to/from
1	PWM Enable pin of Pin 6 from Arduino
2	Motor input 1 from Arduino Pin 10
3, 6	Outputs for the DC Motor
4, 5, 12, 13	Ground
7	Motor input 2 from Arduino Pin 11
8	12 V from the power supply
16	5 V from the Arduino 5V pin

Figure 54: Connection of DC Motor under the Board using Bracket as support

Stepper Motor Driver Circuit

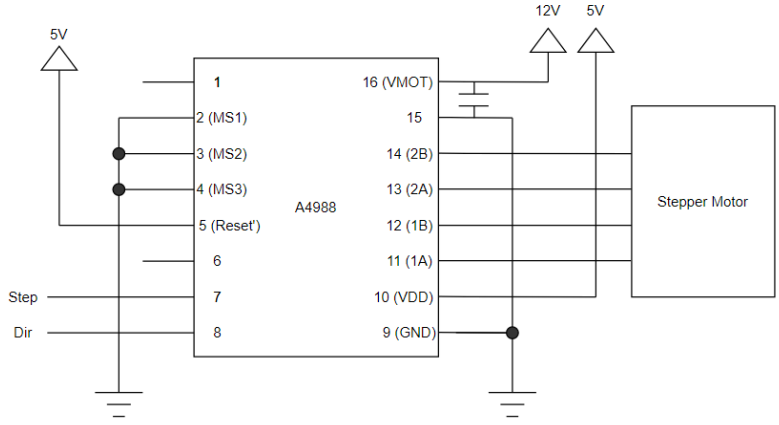


Figure 55: Schematic Diagram of Stepper Motor Driver Circuit

For the stepper motor driver circuit, each driver only requires 2 digital pins from Arduino Mega, which are Step and Dir pins. Step is used to control the step movement of the motor, whereas Dir is used to control the direction of the stepper motor rotation. To allow full step motor operation, all the pins of MS1, MS2, and MS3 are connected to the ground. The reset pin and the VDD pins are both connected to the 5V, which is from the Arduino 5V port.

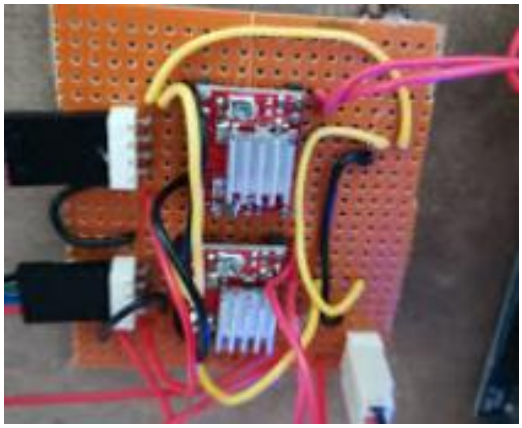


Figure 56: Soldering of both Stepper Motor Drivers on one single Stripped Board

Power Supply and Buck Converter Circuit

The power supply consists of a 3-pin plug which is connected to an external socket, whereas its output V+ and V- are connected to power up the motor drivers which requires a 12 V input voltage.

Figure 57: Installation of Power Supply under the base using screws

The output of the power supply is also connected to the input of the buck converter which the output is supplied to the Vcc pin of the servo motor.

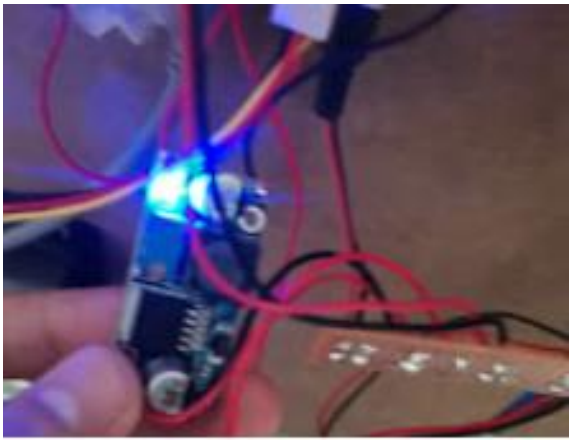


Figure 58: Connection of Buck Converter, with blue light indicates that the device is working properly when an input of 12V is supplied

Ultrasonic Sensor SR-04

Figure 59: Ultrasonic Sensor SR-04 Model

In this project, 2 ultrasonic sensors are used. One of the sensor is attached on top of the platform for detecting human motion, another sensor is attached beside the servo gripper to detect if the item is in front of the gripper.

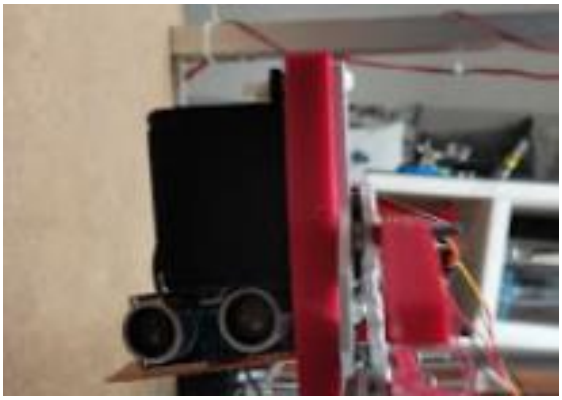


Figure 60: Placement of Ultrasonic Sensor beside the Servo Gripper

IR Sensor Module

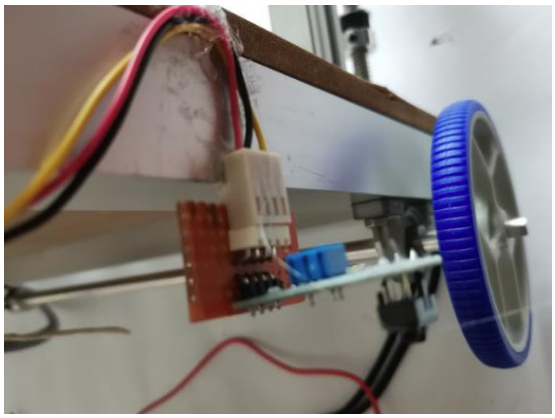


Figure 61: Placement of IR Sensors along the side of the mechanism

2 pair of IR sensor modules are attached on both sides of the mechanism, where one pair is used to detect the rack column position of the mechanism, whereas another pair is used to detect the home position when the robot has finished picking up all the items.

Bluetooth Module HC-05

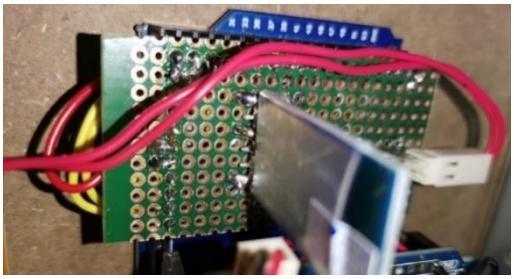


Figure 62: Soldering of Bluetooth Module below the base

The Bluetooth Module HC-05 is used to allow user to purchase their items via mobile app where the items purchased will be sent via Bluetooth transmission to the program. Then the robot will start picking up the items based on the customer's purchase.

End Stop Switches/Homing Switches

In this project, there are 2 end stop switches being used to detect the end position of the moving mechanism in x and y-axis. One of the end stop switch is attached to the top of the platform below the aluminum profile, it is used to detect whether the moving gripper arm has reached the top of the platform. Another one is attached beside the x-axis stepper motor, it is used to detect whether the gripper arm has reached its end position when it reverses back.

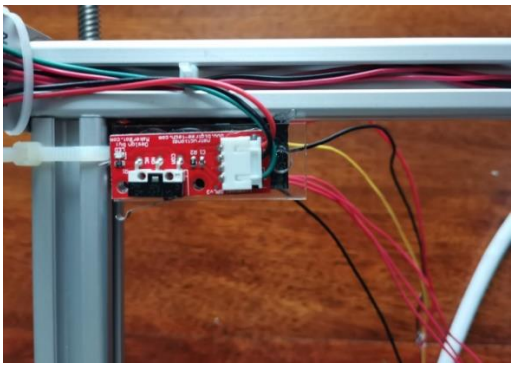


Figure 63: Placement of 1st End Stop Swicth

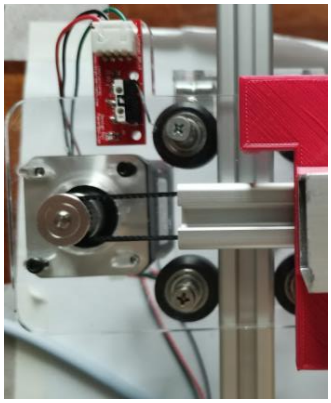


Figure 64: Placement of 2nd End Stop Switch near the x-axis Stepper Motor

4.3.4 Final Schematic Diagram

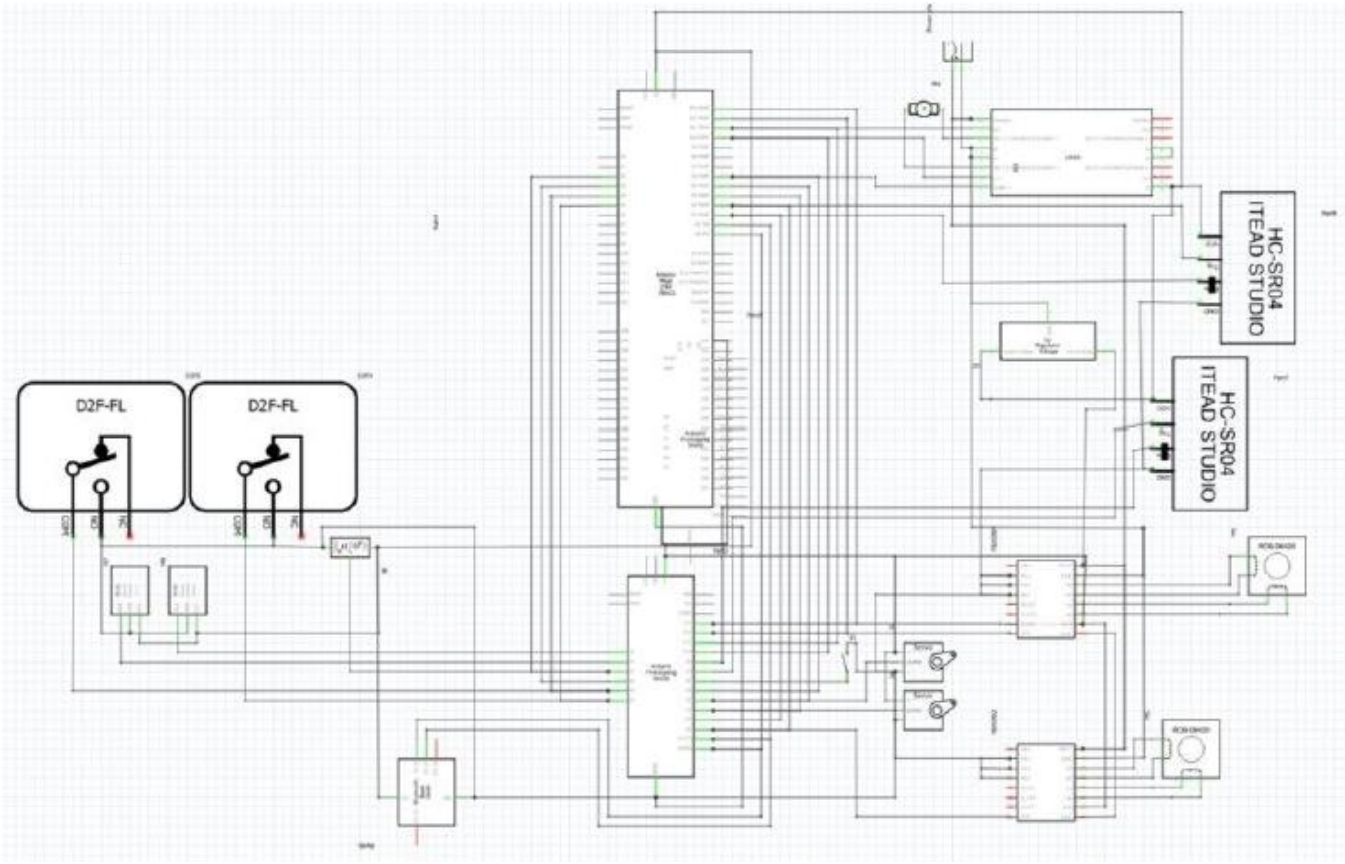


Figure 65: Schematic Diagram of the Mechanism System

Figure above shows the schematic diagram of the circuit, where the output pins from the Arduino are connected to the prototyping shield where all the headers are soldered on it, along with the soldered wires located underneath the board. 2/3/4-way header pins and housings are used for different components with different number of pins.

4.3.5 Wiring and Connectors

Most of the wirings were done on the prototyping shield, stripped boards and also some mini stripped boards to connect components such as end stop switch where the built in cable is not long enough to be connected from the top of the platform to the base.

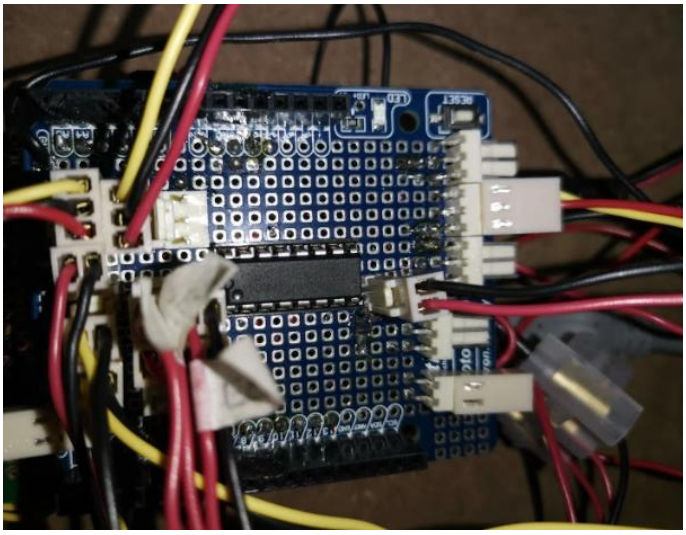


Figure 66: Wiring and Connectors being done on the Prototyping Shield

To prevent confusion, red, yellow, and black wires are used for different types of output. Red wires are used for connecting the 5V/12V voltage from the components, black wires are always used to connect the components to the ground, and yellow wires are used for signal pins and analogue inputs such as IR sensors, end stop switches and servo motors. And most of the large-sized items such as power supply, Arduino Mega board, stripped board for stepper motor connection are placed under the base of the prototype.

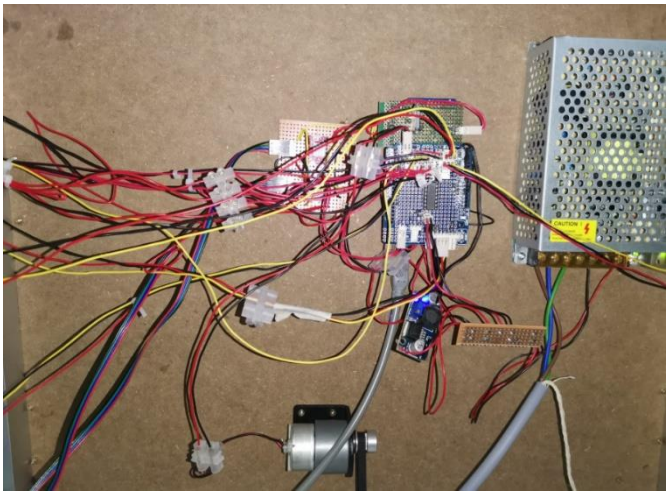


Figure 67: Overall Wiring under the base of the prototype

For the wirings of components on top of the prototype, most of them are tied along the hollow bar for tidier arrangement, whereas some of the wiring connections are placed inside the gap of the aluminum

hollow bar, the connections are done such that it does not interfere with the y-axis movement of the mechanism when the stepper motor rotates, as shown below:

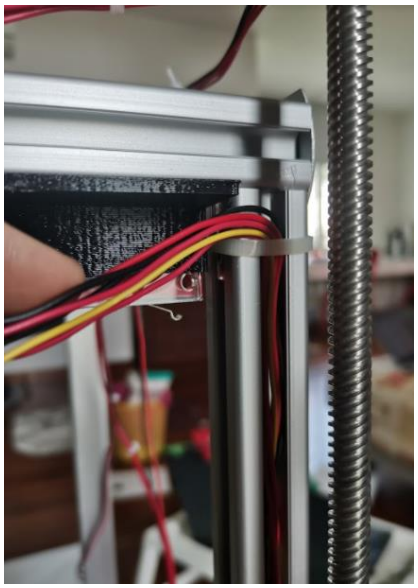


Figure 68: Placement of the Wiring Connection inside the gap of Aluminum Hollow Bar

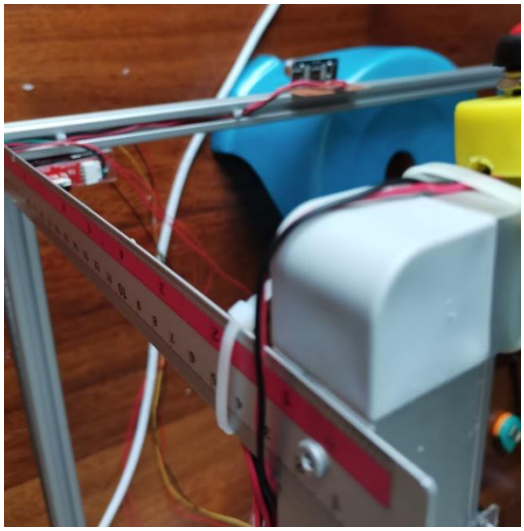


Figure 69: Placement of Wiring along the hollow bar and aluminum profile

In terms of external stripped boards which are used to connect some built-in components which the wires are not long enough to join directly from the top of the prototype until the underneath of base of the prototype.

Connection 1: Connecting Servo Motor to the Prototyping Shield

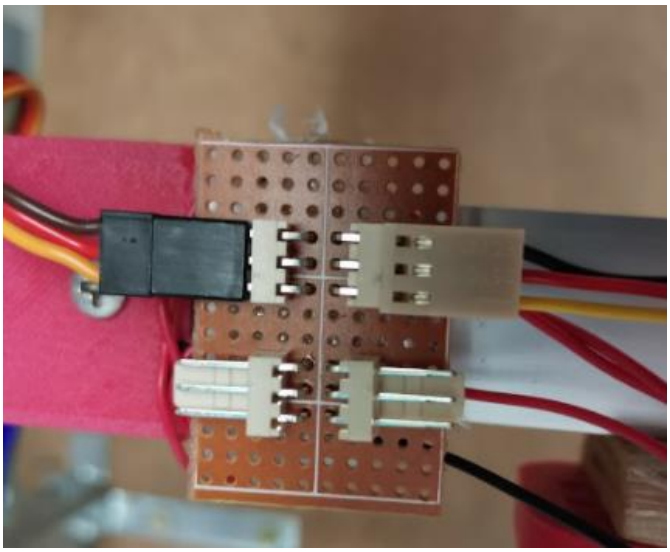


Figure 70: Connecting Servo Motor to the Prototyping Shield using Stripped Board and Connectors

Connection 2: Connecting End Stop Switch to the Prototyping Shield

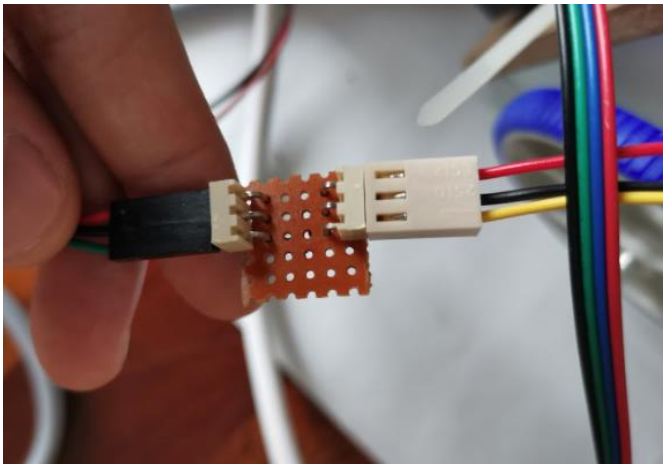


Figure 71: Connecting End Stop Switch to the Prototyping Shield using Stripped Board and Connectors

For the connections above, two header pins are used to connect the components, with the stripped board acts as the platform of connection. Another housing with the crimped wires are connected to the header pins, which is able to connect the components which are far from the prototyping shield.

4.4Programming and Algorithm Implementation

4.4.1 Mobile App Development

Order Screen

Figure 72 Declarations and Initializations of Order Screen


when btn_Eminus .Click
do if get global e ≠ 0
then set global e to get global e - 1
set lbl_quanE . Text to get global e
call updateList
itemName " Yeo's Iced Lemon Tea "
itemQty lbl_quanE . Text
itemPrice 1.50 × lbl_quanE . Text
symbol " 2 "

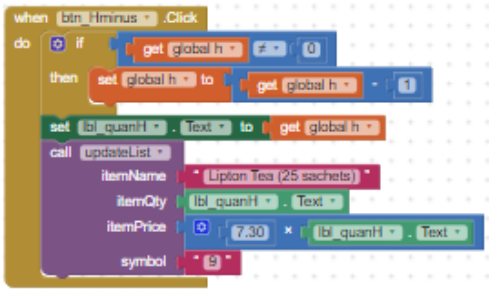

when btn_Eplus .Click
do set global e to get global e + 1
set lbl_quanE . Text to get global e
call updateList
itemName " Yeo's Iced Lemon Tea "
itemQty lbl_quanE . Text
itemPrice 1.50 × lbl_quanE . Text
symbol " 2 "


when btn_Fminus .Click
do if get global f ≠ 0
then set global f to get global f - 1
set lbl_quanF . Text to get global f
call updateList
itemName " 100 plus can drink "
itemQty lbl_quanF . Text
itemPrice 1.80 × lbl_quanF . Text
symbol " 5 "

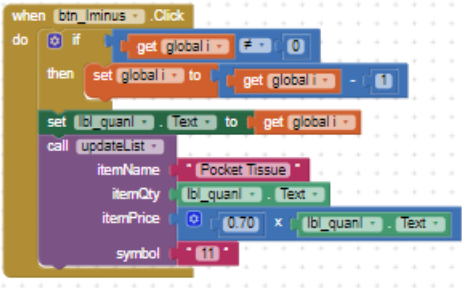

when btn_Fplus .Click
do set global f to get global f + 1
set lbl_quanF . Text to get global f
call updateList
itemName " 100 plus can drink "
itemQty lbl_quanF . Text
itemPrice 1.80 × lbl_quanF . Text
symbol " 5 "


when btn_Gminus .Click
do if get global g ≠ 0
then set global g to get global g - 1
set lbl_quanG . Text to get global g
call updateList
itemName " Munchy's Vegetable Cracker "
itemQty lbl_quanG . Text
itemPrice 0.60 × lbl_quanG . Text
symbol " 7 "


when btn_Gplus .Click
do set global g to get global g + 1
set lbl_quanG . Text to get global g
call updateList
itemName " Munchy's Vegetable Cracker "
itemQty lbl_quanG . Text
itemPrice 0.60 × lbl_quanG . Text
symbol " 7 "


when btn_Hminus .Click
do if get global h ≠ 0
then set global h to get global h - 1
set lbl_quanH . Text to get global h
call updateList
itemName " Lipton Tea (25 sachets) "
itemQty lbl_quanH . Text
itemPrice 7.30 × lbl_quanH . Text
symbol " 9 "


when btn_Hplus .Click
do set global h to get global h + 1
set lbl_quanH . Text to get global h
call updateList
itemName " Lipton Tea (25 sachets) "
itemQty lbl_quanH . Text
itemPrice 7.30 × lbl_quanH . Text
symbol " 9 "


when btn_Iminus .Click
do if get global i ≠ 0
then set global i to get global i - 1
set lbl_quanI . Text to get global i
call updateList
itemName " Pocket Tissue "
itemQty lbl_quanI . Text
itemPrice 0.70 × lbl_quanI . Text
symbol " 11 "

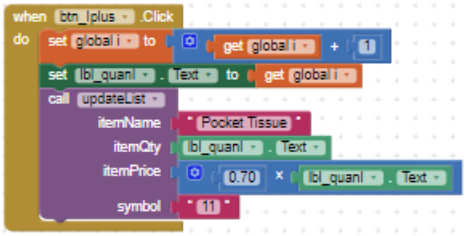

when btn_Iplus .Click
do set global i to get global i + 1
set lbl_quanI . Text to get global i
call updateList
itemName " Pocket Tissue "
itemQty lbl_quanI . Text
itemPrice 0.70 × lbl_quanI . Text
symbol " 11 "

Figure 73 Events for each good (minus or add in quantities, details of item, number symbol in rack)

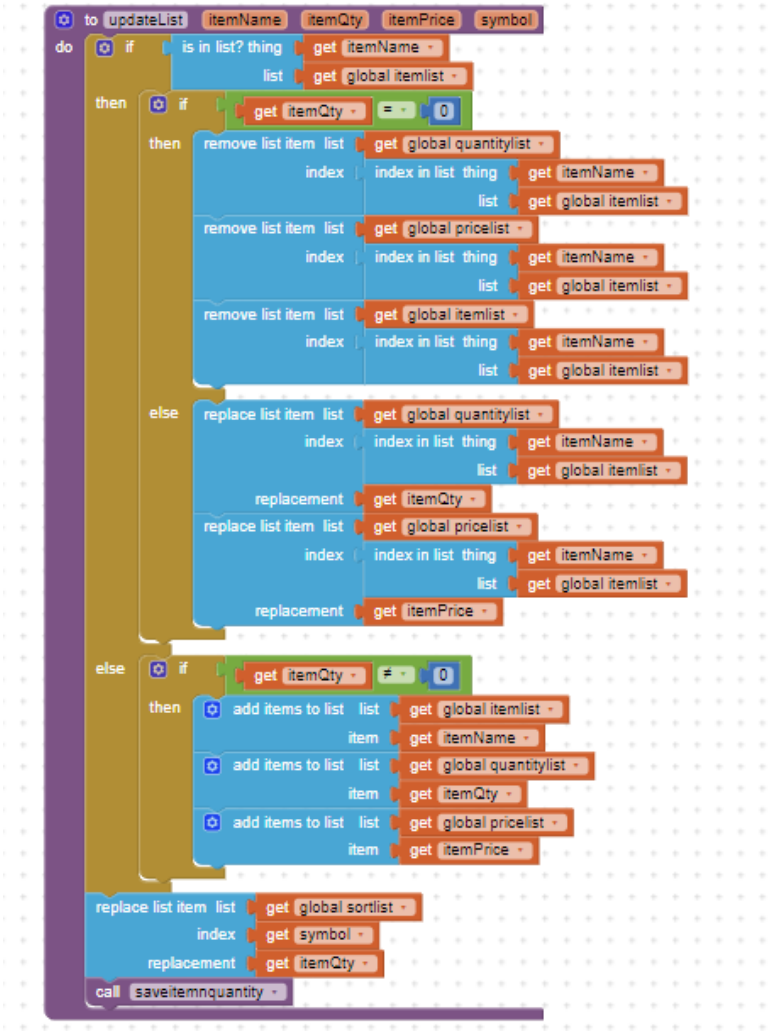


Figure 74 Process to update basket list (add/ remove/ modify)

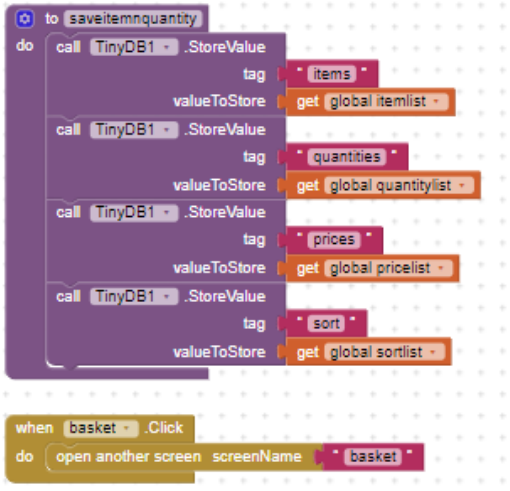


Figure 75 Process to save lists to transfer to another screen

View basket screen

Figure 76 Declarations and Initializations of View Basket Screen

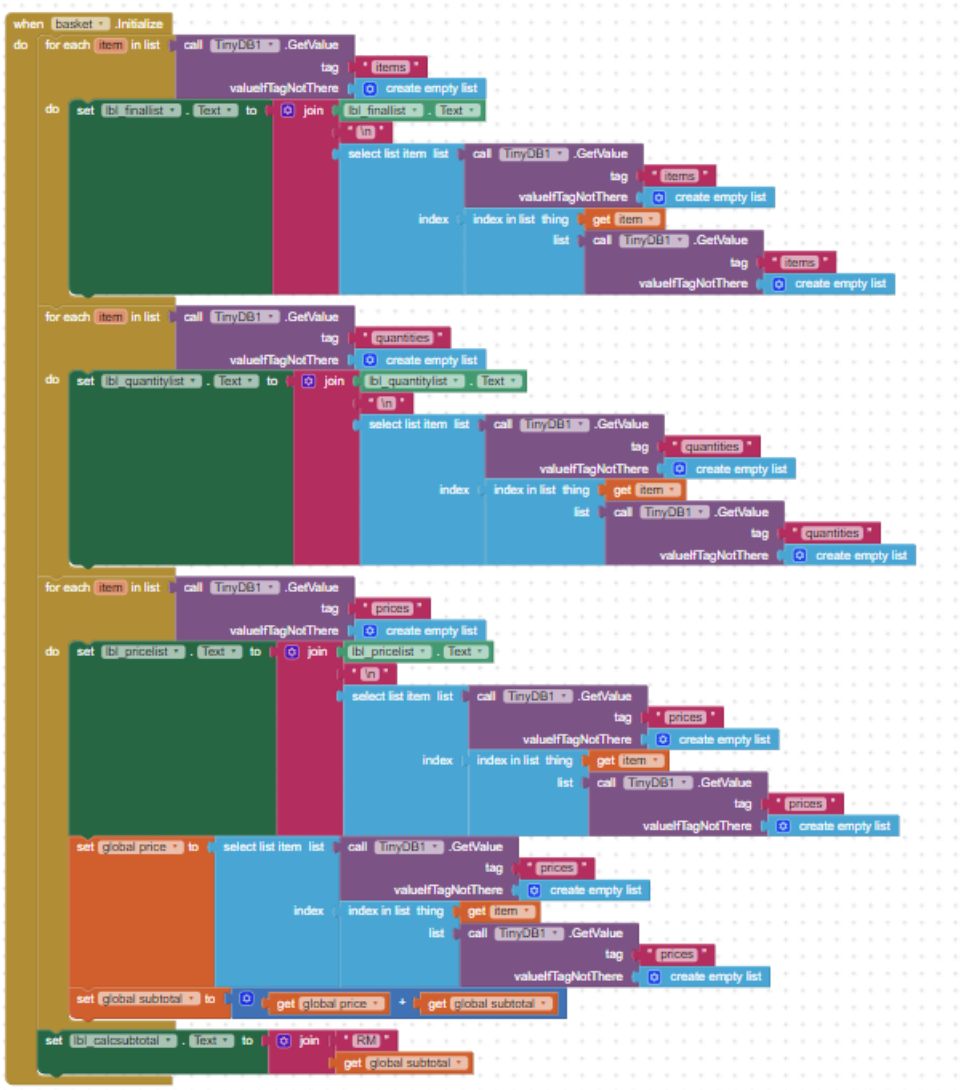


Figure 77 Autofill basket list (shows ordered item names, quantities, price of quantities)

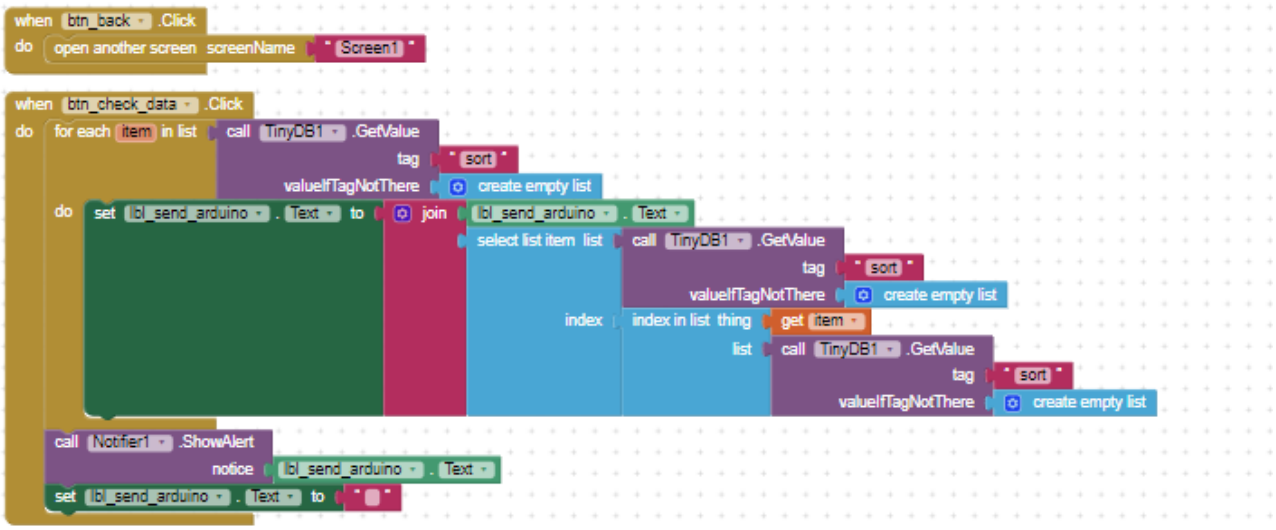


Figure 78 Back button and check data to Arduino small button

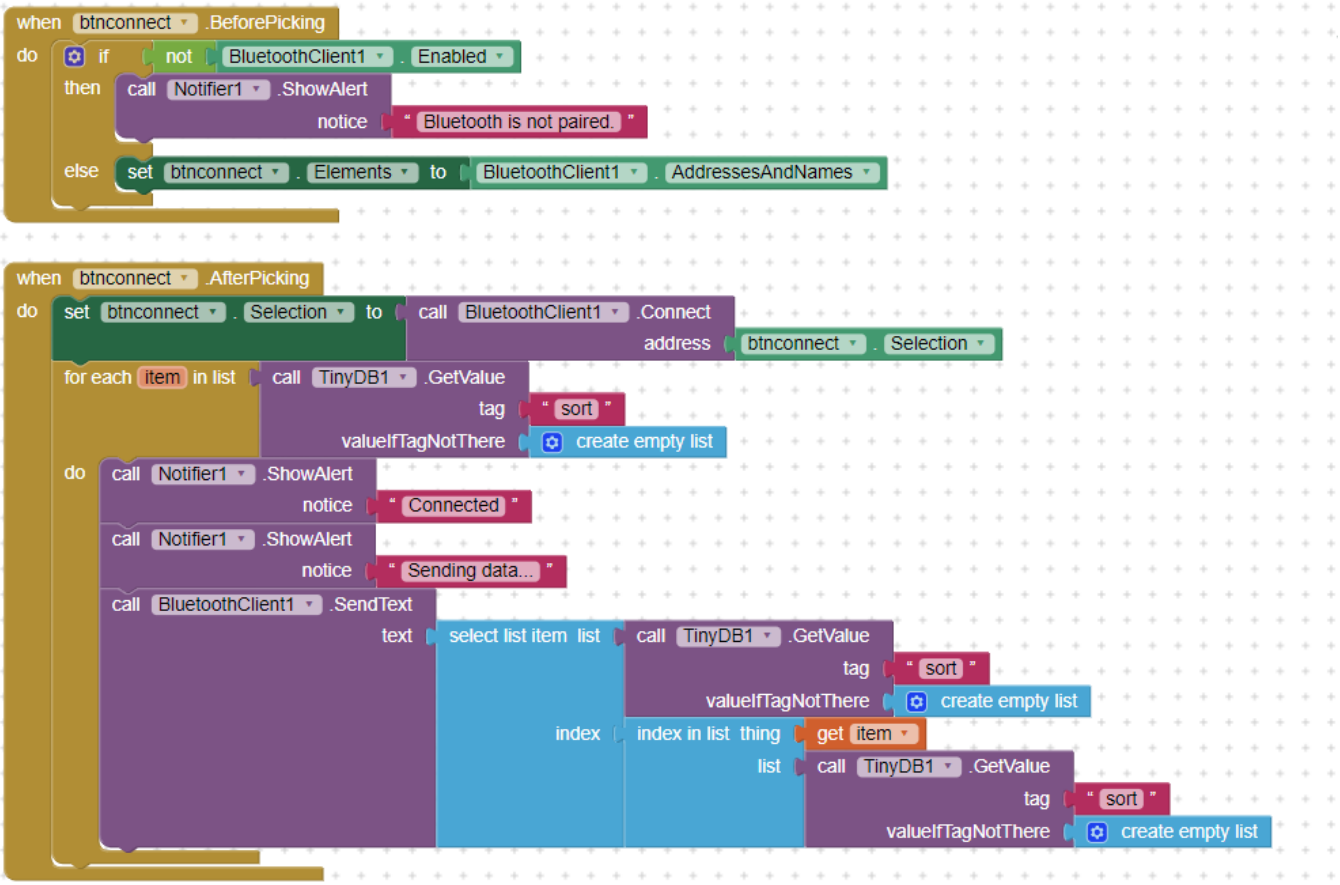
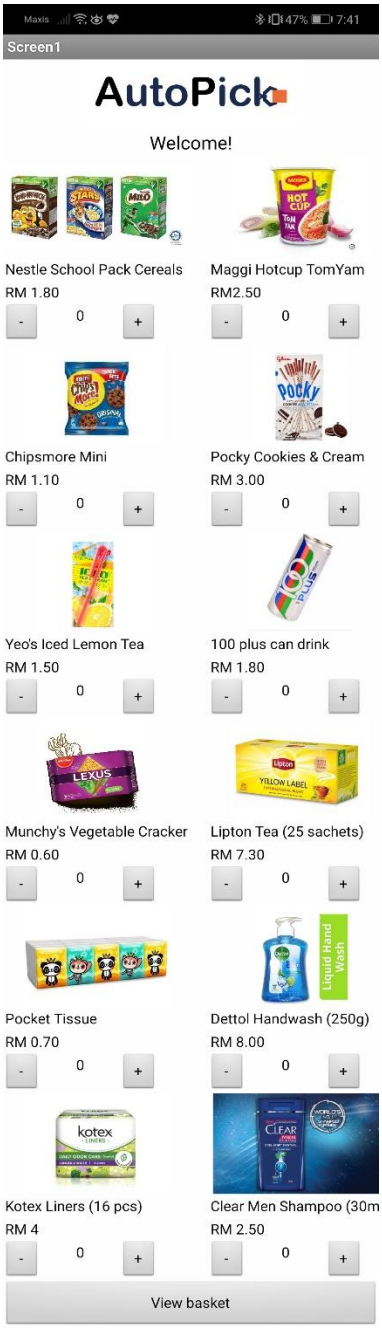


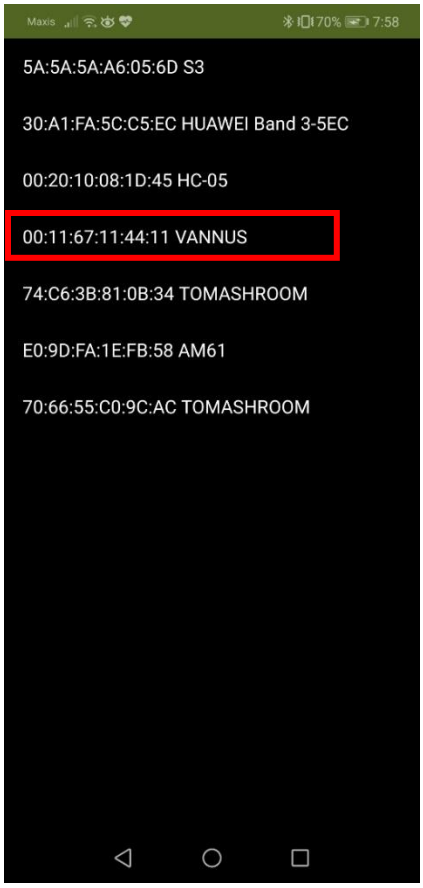
Figure 79 Establish Bluetooth connection and send order info to Arduino

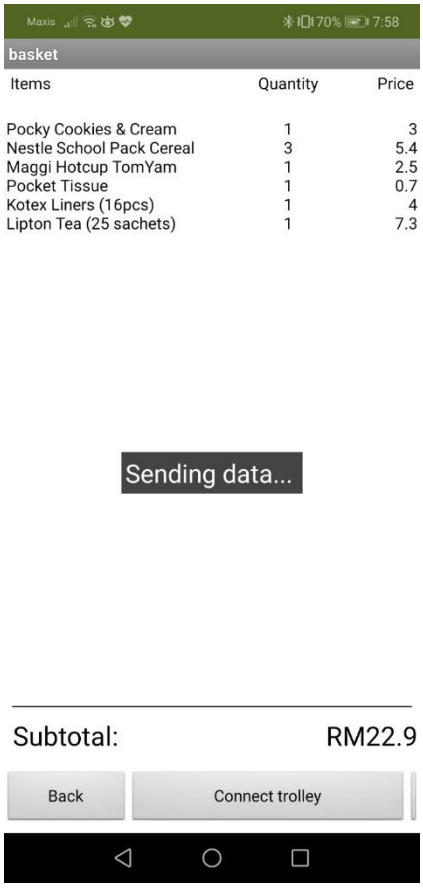
4.4.2 User Manual

The process of the goods picking system starts by customers selecting items from the order screen by clicking the required amount under the displayed goods. Press '+' button to add and '-' button to minus the quantities needed.



When selection is done, press the 'view basket' button to view the order list and total price to confirm the items that will be pick by the machine. Make sure that the Bluetooth of your mobile device is on and paired with HC-05 Bluetooth module on the machine.





Choose *00:20:10:08:1D:45 HC-05* to send the data of your order to the AutoPick machine by clicking 'connect trolley'. The machine will send the goods to the pick-up zone to be retrieve and paid.

4.4.3 Design process

A logo is designed for the system.

Figure 80 Logo of AutoPick

The *order* screen displays the logo together with 12 selling goods. The layout of all controls is arranged with horizontal and vertical blocks for neat display.

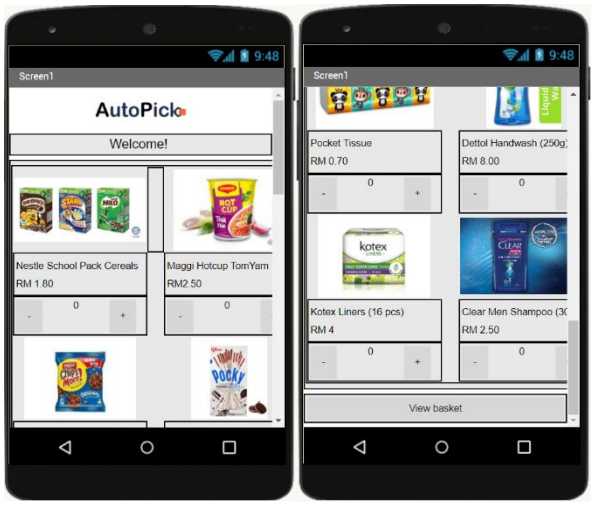


Figure 81 Designer Viewer of order screen

There are 2 buttons for each of the 12 good which stores and updates the details of item *name*, *quantity*, *price of quantity* and *respective number symbol in rack* to a list by calling *updateList* process. The minus button event also prevents negative amount of item happens.

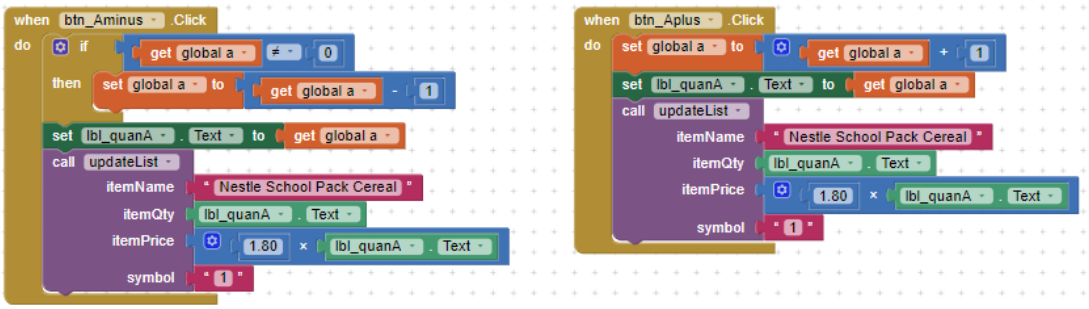


Figure 82 Example of the button controls of a good

updateList process detects updates of each button clicks by checking the presence of chosen item name in the list and also to remove items that changed back to zero when choosing to prevent redundant information in the final order list. Removing or replacing of elements in the lists, the name of the good is used as the unique identifier to retrieve stored index in all lists. Therefore, when removing the entire row of item details from basket, the identifier should be removed the last after using it to retrieve other lists.

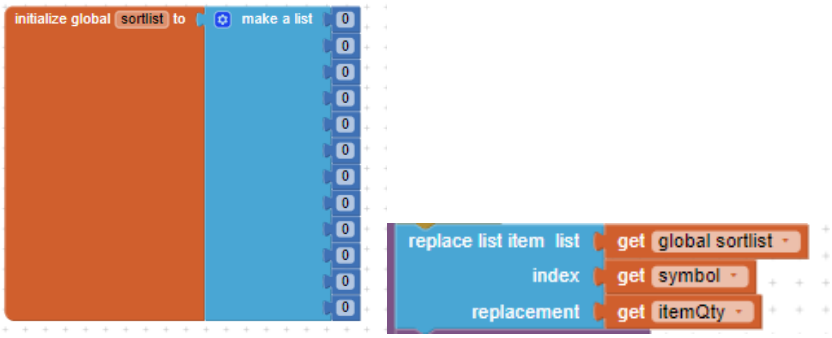


Figure 83 sortlist blocks

The function of these blocks is used to encode the data that will be send to Arduino for processing the movements of the machine. Each elements of the list represent the number symbol of the rack. The quantity of the goods will be updated to the respective number symbol element of the list.

Figure 84 Respective number symbol in rack

Figure 85 TinyDB function

Tiny DB is used to store data persistently so the lists can be transfer to the *basket* screen for display and calculations.

After the view basket is pressed to go the next screen, the data stored in *TinyDB* are retrieved accordingly for each element of *items*, *quantities* and *prices* lists to display in rows. The subtotal price is calculated to show customer by adding up each price element in the price list. A small button is also created to check the encoded data that will be transfer to Arduino.

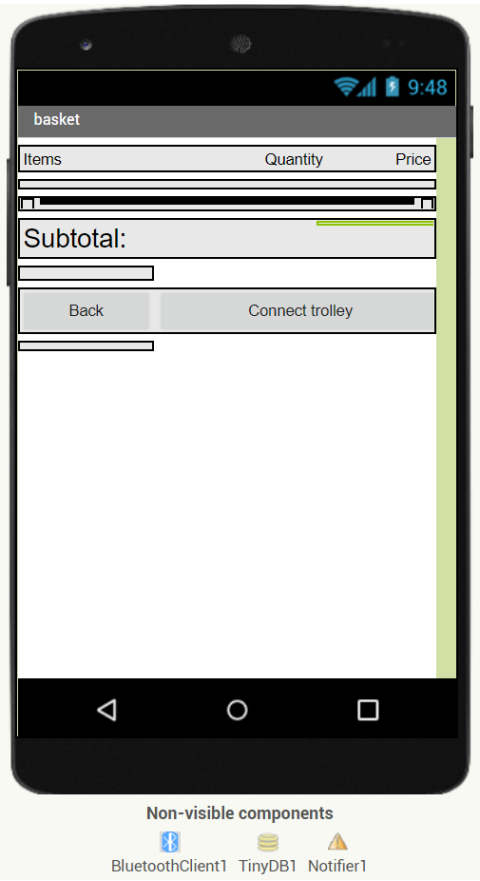
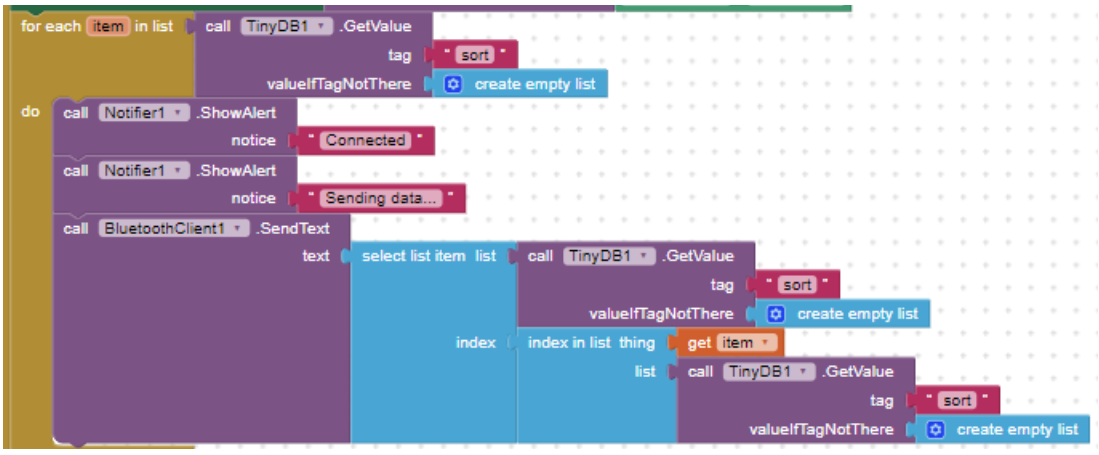


Figure 86 Designer Viewer of view basket screen

ListPicker is used to select available device for Bluetooth connection through *BluetoothClient*. *Notifier* is added to show temporary message about the connection status.

The app will send the encoded data through text/string to Arduino.



4.4.4 Tested Codes

Bluetooth module

app2bt | Arduino 1.8.13

File Edit Sketch Tools Help

app2bt §

```
String quantities;

void setup() {
  Serial.begin(9600);
  Serial1.begin(9600); // Default communication rate of the Bluetooth module
}
void loop()
{
    int i, j;
    int arr[12];
    int y = 0;
    for (i = 0 ; i < 12; i++)
    {
      if(Serial1.available() > 0) // Checks whether data is coming from the serial port
      {
        quantities = Serial1.readString(); // Reads the data from the serial port
        Serial.println(quantities);
        y=1;
      }
    }
    if (y==1)
    {
      for (j = 0 ; j < 12; j++)
      {
         arr[j] = (quantities.charAt(j)-48);
         Serial.println(arr[j]);
      }
      y=0;
    }
}
```

Figure 87: Code Snippet for Testing Bluetooth Connection

The above test program is the implementation of Bluetooth connection which transmits the data from the user app into the Bluetooth module so that the machine is able to pick up the items based on the items ordered by the customer. Initially when the customer selects the items from the shopping basket, upon clicking the checkout function, all the items ordered in that list will be stored into an array, in which the array consists of all the quantities of that item. The quantities of the items can be viewed at the Serial Monitor when the user clicks the checkout function on their app.

Stepper motor #2 & End stop switch #2

endstopstepperY_calibration_ | Arduino 1.8.13

File Edit Sketch Tools Help

endstopstepperY_calibration_

```
// defines pins numbers
int ystepPin =  22; //x axis A0
int ydirPin = 24;  //x axis A1
int ystop = A5 ;   //x axis D3


void setup()
{
  pinMode(ystepPin,OUTPUT);
  pinMode(ydirPin,OUTPUT);
  pinMode(ystop, INPUT_PULLUP);

  Serial.begin(9600);
}

void loop()
{
  int delaytime = 1000;
  digitalWrite(ydirPin,LOW); // HIGH(move down), LOW(move up)

  while(digitalRead(ystop) == HIGH)
  {
    digitalWrite(ystepPin,HIGH);
    delayMicroseconds(delaytime);
    digitalWrite(ystepPin,LOW);
  }
  Serial.println("touch");
}
```

Figure 88: Code Snippet for Stepper Motor #1 and End Stop Switch Testing #1

Stepper motor #1 & End stop switch #1

endstopstepperX | Arduino 1.8.13

File Edit Sketch Tools Help

endstopstepperX

```
// defines pins numbers
int xstepPin =  12; //x axis A0
int xdirPin = 13;  //x axis A1
int xstop = 7 ;   //x axis D3

void setup()
{
  pinMode(xstepPin,OUTPUT);
  pinMode(xdirPin,OUTPUT);
  pinMode(xstop, INPUT_PULLUP);

  Serial.begin(9600);
}

void loop()
{
  int delaytime = 1000;
  digitalWrite(xdirPin,HIGH); // HIGH(move home), LOW(grip)

  while(digitalRead(xstop) == HIGH)
  {
    digitalWrite(xstepPin,HIGH);
    delayMicroseconds(delaytime);
    digitalWrite(xstepPin,LOW);
    delay(0.5);
  }
  Serial.println("touch");
  delay(1000);
}
```

Figure 89: Code Snippet for Stepper Motor #2 and End Stop Switch Testing #2

The code snippets above are the test code for the stepper motor and end stop switch function. These 2 components are tested together as the end stop switch acts as a digital sensor which when the stepper motor reaches the end of the path, it is able to stop with the help of this end stop switch. The inputs of the end stop switches are set as input with internal pull-up resistors in order to prevent the value from floating when the switch is pressed or released. To integrate these 2 components together in the algorithm, a conditional statement is set such that the stepper motor will rotate unless it collides with the end stop switch, which toggles the state of the input from HIGH to LOW.

Red push button

Figure 90: Code Snippet for the Emergency Stop Switch Implementation and Testing

The above program is the testing code for the emergency button switch. When the button is pressed, the stepper motors and other moving components will stop functioning until it is pressed again.

IR sensors

IR_interval | Arduino 1.8.13

File Edit Sketch Tools Help

IR_interval §

```
#define ir_sensor A1
void setup()
{
  pinMode(ir_sensor,INPUT);
  Serial.begin(9600);
}

int old_a = 0;

void loop()
{
  int a;
  int old_a;
   a = analogRead(ir_sensor);
   if (a - old_a >= 500) //not in white range
   {
      // black is detected
      Serial.println(a);
      Serial.println(" black");
      delay(500);
   }
    else
   {
       // white
       Serial.println(a);
       Serial.println(" white");
       old_a = a;
       delay(500);
   }
}
```

Figure 91: Code Snippet for IR Sensor Testing

The code above shows the testing and calibrating of IR sensor. Both IR sensors are connected to the analogue pins of the Arduino Mega board. If the value detected from the sensor which is fed into the Arduino Mega board produces a value less than 500, the sensor will detect that it is in the white region, else the sensor will detect a black tape attached on the floor. For more accurate reading, the IR sensor must not be placed too high from the ground as the difference between detecting black and white region will be less significant.

DC Motor

DCmotor | Arduino 1.8.13

File Edit Sketch Tools Help

DCmotor §

```
#define en_pwm 6                    // dc motor l293d pwm
#define in_1 10                     // dc motor input pin 1
#define in_2 11                     // dc motor input pin 1

void setup(){
  pinMode(en_pwm, OUTPUT);
  pinMode(in_1, OUTPUT);
  pinMode(in_2, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  int i;
  for (i=0; i<50; i++)
  {
    //forward
    digitalWrite(in_1,HIGH);
    digitalWrite(in_2,LOW);
    //backward
    //digitalWrite(in_1,LOW) ;
    //digitalWrite(in_2,HIGH);

    analogWrite(en_pwm,255) ;
    Serial.println("running");
  }
  digitalWrite(in_1,LOW); // HIGH
  digitalWrite(in_2,LOW); //HIGH
  Serial.println("pause");
  delay(1000);
}
```

Figure 92: Code Snippet for DC Motor Rotation

The above code explains the testing and implementation of DC Motor rotation. Three pins from the Arduino Mega Board are used to implement the algorithm, 2 of the inputs are digital pins which is fed into the input of the motor driver L293D, and another PWM input is used to control the smoothness and speed of the DC motor rotation. To enable full speed of the motor, the PWM value must be set to 255, in which the motor rotates without stopping in between the intervals.

Ultrasonic sensors

ultrasonic_sensors | Arduino 1.8.13

File Edit Sketch Tools Help

ultrasonic_sensors

```
#define echoPin 3 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 2 //attach pin D3 Arduino to pin Trig of HC-SR04
#define echoPin2 9
#define trigPin2 8

// defines variables
long duration; // variable for the duration of sound wave travel
int distance;  // variable for the distance measurement
long duration2;
int distance2;

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT);  // Sets the echoPin as an INPUT
  pinMode(trigPin2, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin2, INPUT);  // Sets the echoPin as an INPUT
  Serial.begin(9600);        // Serial Communication is starting with 9600 of baudrate speed
  Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
  Serial.println("with Arduino UNO R3");
}
void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
```

```
  // Clears the trigPin condition
  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration2 = pulseIn(echoPin2, HIGH);
  // Calculating the distance
  distance2 = duration2 * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor

  Serial.print("Distances: ");
  Serial.print(distance);
  Serial.println(" cm");

  Serial.print(distance2);
  Serial.println(" cm");
  delay(500);

  if (distance < 50){
    Serial.print("Don't stand too close");
  }

  if (distance2 < 10){
    Serial.print("Item detected");
  }
}
```

Figure 93: Code Snippet for Ultrasonic Sensors Implementation

The code above shows the testing and implementation of ultrasonic sensors in the prototype. One ultrasonic sensor will be used to detect the human motion for safety purpose. Another ultrasonic sensor will be glued to the side of the gripper to detect if there is an item present in front of it. For the human motion detection, a 50 cm threshold is set such that any distance detected less than 50 cm will temporarily put all the operations on hold, until the distance becomes greater than 50 again. For the item detection, if the distance between the sensor and the item placed on the rack is less than 10cm, the mechanism will stop for a short while before moving forward to get the item.

Servo Motor

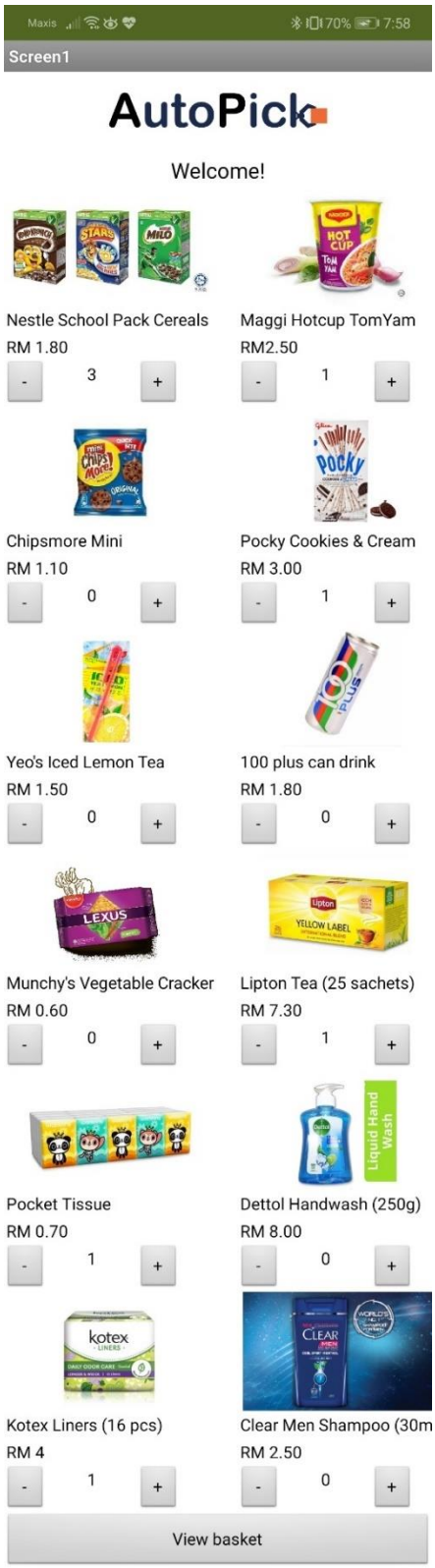
```
#define SERVO_PIN         5  // Any pin on the Arduino or Gertboard will work.

void setup()
{
  pinMode(SERVO_PIN, OUTPUT);
}

int lenMicroSecondsOfPeriod = 25 * 1000; // 25 milliseconds (ms)
int lenMicroSecondsOfPulse = 1 * 1000; // 1 ms is 0 degrees
int first = 1.3 * 1000; //0.5ms is 0 degrees in HS-422 servo
int end = 3 * 1000;
int increment = 0.01 * 1000;

void loop()
{
    int current = 0;
    for(current = end; current >first; current-=increment){
         digitalWrite(SERVO_PIN, HIGH);

         // Delay for the length of the pulse
         delayMicroseconds(current);

         // Turn the voltage low for the remainder of the pulse
         digitalWrite(SERVO_PIN, LOW);

         // Delay this loop for the remainder of the period so we don't
         // send the next signal too soon or too late
         delayMicroseconds(lenMicroSecondsOfPeriod - current);
    }
}
```

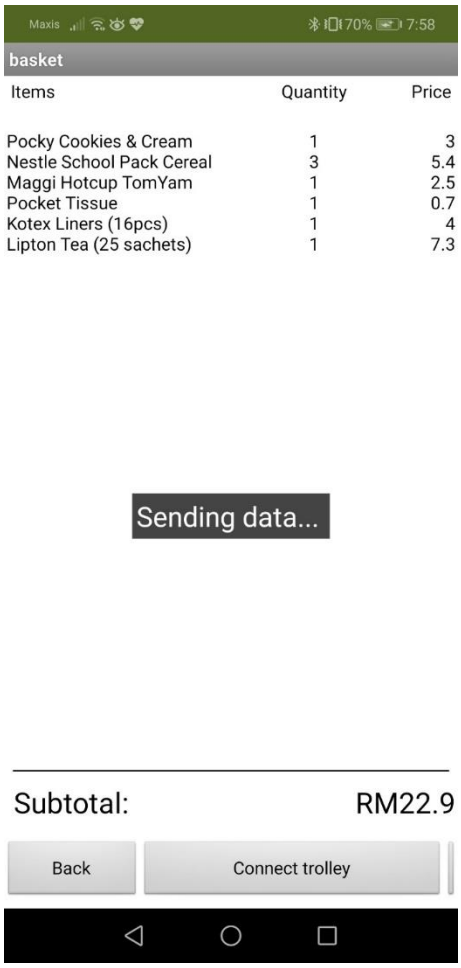
Figure 94: Code Snippet for Testing Servo Motor

The above code shows the testing and implementation of servo motor. Since the servo motor signal pin 5 is powered by PWM signal. Hence, calibration of the time intervals need to be set in order for the servo gripper to open and close in a way that it does not damage the inner side of the shelf when opening, and it is able to grip the item firmly when closing.

4.4.5 Results

Bluetooth with the Graphic User Interface (GUI)



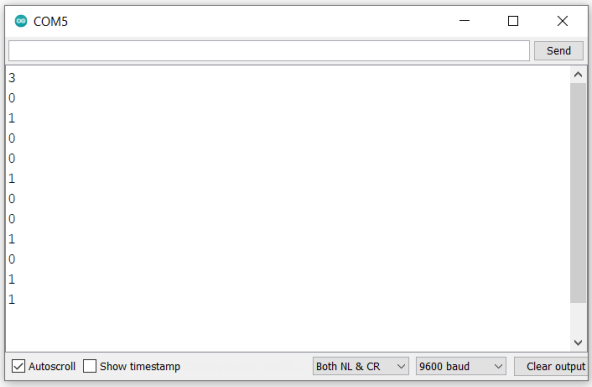


app2bt | Arduino 1.8.13

File Edit Sketch Tools Help

app2bt

```
String quantities;

void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
}
void loop()
{
    int i, j;
    int arr[12];
    int y = 0;
    for (i = 0 ; i < 12; i++)
    {
      if(Serial1.available() > 0) // Checks whether data is coming from the serial port
      {
        quantities = Serial1.readString(); // Reads the data from the serial port
        //Serial.println(quantities);
        y=1;
      }
    }
    if (y==1)
    {
      for (j = 0 ; j < 12; j++)
      {
         arr[j] = (quantities.charAt(j)-48);
         Serial.println(arr[j]);
      }
      y=0;
    }
}
```



5 Result and discussion

5.1 The advantages of the outcome

Go through the project, many kinds of thing happen during the project development. For mechanical side, the Y axis belt did not have the perfect length in the market. Which forced the team used a double head pulley to connect both different lengths together. So that both thread rod can turn together. Other than that, as the X axis side, there is no suitable materials that able to connect the X stepper and aluminum profile 2020 together. So, the team purchase the acrylic custom parts for it, as the design idea can easily to achieve through the laser cutting. Moreover, the team did use the 3D printing methods to fabricate certain parts as well. This bring some benefits as no need to fabricate the handy parts.

At the electronic side, there are a few troubleshooting tasks which is used to check the electrical faulty of the circuit board and the prototyping shield:

Troubleshooting Task 1: Check Connectivity of the Circuit using Digital Multimeter Connectivity Function

In some cases, there is a possibility that the internal wires is broken or there is a short circuit occurs within the stripped board or prototyping shield. Hence, digital multimeter is used to check these faulties.

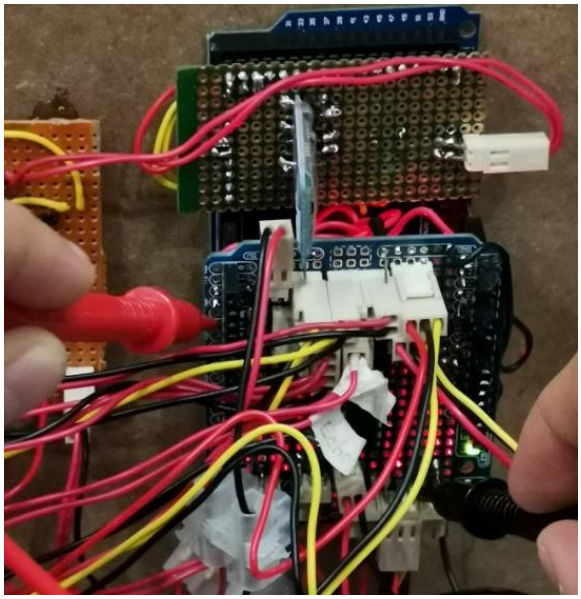


Figure 95: Checking of Circuit Connectivity using Digital Multimeter

Based on the testing done so far, the following faulties were detected during the troubleshooting process:

- Short-Circuit occurs at the stripped board which connects the end stop switch, which causes the Arduino Mega board to cut off its power after the connector housing is plugged into the header pin.
- There is a break-off connection occurs at one of the sensor, the main cause of the break-off is due to the detachment of one of the wires connected to the ground.

To solve the faulties mentioned above, a newly cut and stripped wire is use to reconnect the broken connection. After that, the wired connection is secured by soldering it with the equipment.

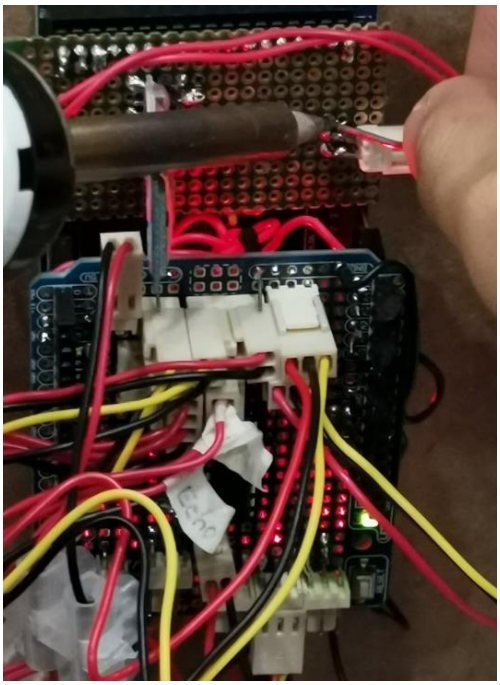


Figure 96: Soldering of new wire/connection to connect the designated section

5.2 The shortcoming of the outcome

The short coming on the mechanical as shown following:

1. The Y axis belt across the MDF wood base and which make the customer feel unconformable to pick out the groceries.
2. Acrylic parts are easy to laser cut but it cannot handle too much forces, also it is too expensive once the parts are broken and need to replace it again.
3. 3D printing method consume a lot of the time which cannot have a high quantity in fabricating the parts in the same time consuming as other methods.

The short coming on the electrical as shown following:

5.3The improvement opportunity

For the mechanical improving opportunity:

1. Reduce the distance between two profiles and use two stepper motor to lift the X axis together.
2. Using the sheet metal as the fabricating method to make the parts. As the sheet metal market is mature and commonly used in the market.
3. Instead of using the 3D printing method, the sheet metal may also can replace the 3D printed parts as well. The other option to improve the design is welding the metal plate together.

For the electrical improving opportunity:

1. Improving of the wiring connection so that the probability of overtight connection can be reduced. For example, the length of the wires need to be calculated accurately, with an additional length of 5-10 cm, so that the wiring can be connected properly without the risk of the wires being overtightened and break off in the middle of picking process.
2. More proper planning of the prototyping shield in order to prevent the wiring underneath the circuit from becoming too messy.

For programming improving opportunity:

1. Implement Out of Stock function, so when a customer purchases an item which its quantity exceeds the available of items placed in the shelf, it will send a message informing that the particular item is low on quantity/out of stock.
2. Implement maximum number of items can be purchased, so that the orders are not comprising of heavy loads which may damage the basket/platform.

6 Conclusion and future works

As the conclusion of this project, the mechanical design was built as expected. The dimension did not offset too much. The main reason is that the team cannot demo the project due to the messy circuit board soldering, which makes the circuit board to be short circuited frequently, thus wasting too much time on solving it. Second, due to the mechanical design failure at the Y axis driven belt, the double head pulley was not considered properly, therefore the nuts kept coming up and causing frictions which blocks the double head pulley to turn smoothly. After that, due to the late project development, the programmer has not much time to program the machine work and calibrate with the machine. Even the machine code can work, due to the pinout unknown short circuit, as result, many machine functions cannot show properly. Certain parts will move even the code is not related to the part.

For the future work of the project, as the project cannot perform the function at the end of the semester. The first work to do is improve the mechanical part of the project wisely to eliminate the bad effects brought. Therefore, the wire routing will be included in the timeline of the project, to make it to not look too messy and the potential caused the machine or people get shocked. Other than that, the extra coverage will be added as well. Hence, the groceries that picked will not lost before it passed to the buyer. Moreover, the next steps are replaced or repurchased the broken motors or parts.

For the electronic side, the very first thing is purchasing a brand-new prototyping shield and the component. Then remake the circuit diagram that shows where the connection goes and solid soldering, so no wire loss happened and slowed progress. Also, the pinout diagram has to be shared with the teammates hence the progress will boost up by a lot. The microcontroller board needs to fully understood before the designing the circuit so that more problems can be avoided.

Reference

[1] CNC.xyz,2010?, Stepper motor calculation, viewed 27 October 2020 <https://wiki.cnc.xyz/Stepper_Motor_Calculations>.

[2] Oriental motor,2010?, Motor Sizing, viewed 27 October 2020, <https://www.orientalmotor.com/technology/motor-sizing-calculations.html >.

[3] Aasvik. M, 2015 July, Tutorial Calibrating Stepper Motor Machines with Belts and Pulleys, Norwegiancreations, viewed 27 October 2020, <https://www.norwegiancreations.com/2015/07/tutorial-calibrating-stepper-motor-machines-with-belts-and-pulleys/ >.

[4] Schneider Electric, 2017 January, How do I calculate the RPM's of a stepping motor? , viewed 27 October 2020, < https://www.se.com/ww/en/faqs/FA337686/>.

[5] DOMOTIRCX Knowledge Center, 2010?, Mechanica- GT2 Tandriem en Pulley, viewed 26 October 2020, <http://domoticx.com/mechanica-gt2-tandriem-en-pulley/>.

[6] MotionKing, 2010? , 17HS Stepper Motor, viewed 27 October 2020, < http://www.motionking.com/show_products_detail.asp?ID=8&fenlei_ID=2 >.

[7] Electronicos CALDAS, 2010?, MG996R, viewed 27 October 2020, <https://www.electronicoscaldas.com/datasheet/MG996R_Tower-Pro.pdf > .

[8] SunFounder Steam Education, 2019 September, TD-8120MG Servomotor Datasheet, viewed 26 October < http://wiki.sunfounder.cc/images/9/9a/TD-8120MG_Digital_Servo.pdf >

[9] Braun, J, 2018, 'Fromulae Handbook', maxon academic, pg23, viewed 20 October 2020, <https://maxon.blaetterkatalog.ch/b9991/catalog/index.html?data=b9991/b999145&lang=e#22 >.

[10] TJ Stallbaumer, Jan 2020, WalMart, viewed by ,26 September 2020, <https://corporate.walmart.com/newsroom/2020/01/08/how-walmarts-alphabot-is-helping-to-revolutionize-online-grocery-pickup-and-delivery >.

[11] SOTO2, n.d., MAGAZINO, viewed by, 26 September 2020, <https://www.magazino.eu/products/soto/?lang=en>.

[12] Swift, n.d., IAMROBOTICS, viewed by 26 September 2020, < https://www.iamrobotics.com/ >.

[13] RightPick, n.d., RIGHTHAND Robotic, viewed 26 September 2020, <https://www.roboticstomorrow.com/article/2018/10/piece-picking-robots/12618/ >.

Appendix

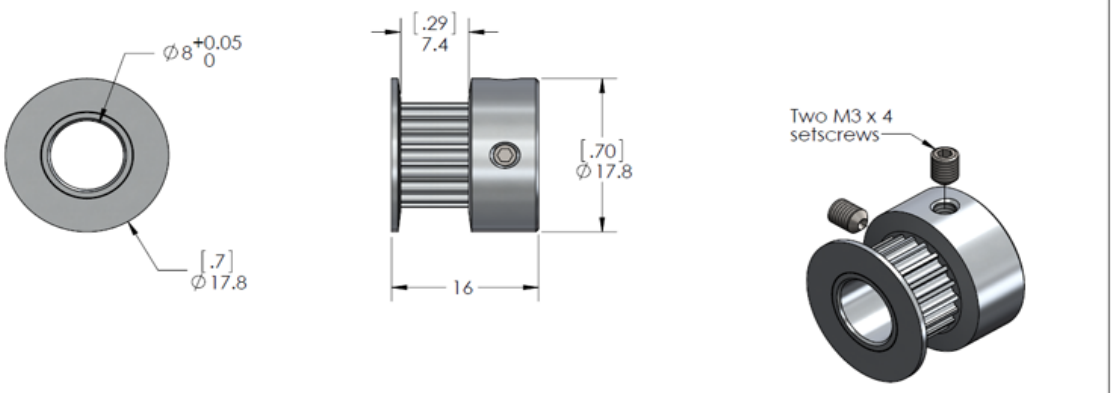
Appendix A: GT2 Pulley Datasheet

Reference Table

Model:JGB37-520		Data Sheet							
Voltage		No Load		Load Torque				Stall	
Workable	Rated	Speed	Current	Speed	Current	Torque	Output	Torque	Current
Range	Volt.V	rpm	ma	rpm	ma	kg.cm	W	kg.cm	A
3-9V	6v	1280	300	896	700	0.07	1	0.28	2
3-9V	6V	800	300	640	700	0.1	1	0.4	2
3-9V	6V	425	300	340	700	0.18	1	0.72	2
3-9V	6V	266	200	212	500	0.3	1	1.2	2
3-9V	6V	142	200	113	400	0.56	1	2.2	2
3-9V	6V	88	100	70	400	0.9	1	3.6	2
3-9V	6V	61	100	48	350	1.3	1	5.2	2
3-9V	6V	47	100	37	300	1.68	1	6.7	2
3-9V	6V	29	100	23	200	2.7	1	10	2
3-9V	6V	16	100	12	150	5	1	20	2
3-9V	6V	10	100	8	150	8	1	32	2
6-15v	12v	960	100	672	200	0.1	0.8	0.4	1.2
6-15V	12V	600	100	480	200	0.15	0.8	0.6	1.2
6-15V	12V	319	100	255	170	0.28	0.8	1.1	1.2
6-15V	12V	200	60	160	150	0.45	0.8	1.8	1.2
6-15V	12V	107	60	85	150	0.84	0.8	3	1.2
6-15V	12V	66	60	52	100	1.35	0.8	5	1.2
6-15V	12V	45	60	36	100	1.9	0.8	7	1.2
6-15V	12V	35	50	28	100	2.5	0.8	10	1.2
6-15V	12V	22	50	18	80	4	0.8	16	1.2
6-15V	12V	12	50	10	80	7.5	0.8	30	1.2
6-15V	12V	7	50	5.6	80	12	0.8	48	1.2
12-30v	24v	960	100	672	150	0.15	0.7	0.6	0.8
12-30V	24V	600	100	480	150	0.2	0.7	0.8	0.8
12-30V	24V	319	100	255	150	0.37	0.7	0.15	0.8
12-30V	24V	200	50	160	120	0.6	0.7	2.4	0.8
12-30V	24V	107	50	85	120	1	0.7	4	0.8
12-30V	24V	66	50	52	100	1.8	0.7	7	0.8
12-30V	24V	45	50	36	80	2.5	0.7	10	0.8
12-30V	24V	35	30	28	60	3.3	0.7	13	0.8
12-30V	24V	22	30	18	50	5	0.7	20	0.8
12-30V	24V	12	30	10	40	10	0.7	40	0.8
12-30V	24V	7	30	5.6	40	16	0.7	64	0.8

Specifications	
For belt type / pitch	GT2 / 2mm pitch
Number of Teeth	20
Pitch Diameter	12.7mm / 0.5"
Outside Diameter	12.2mm / 0.48"

Calculated steps per mm for firmware	
1.8° stepper motor, full step	5
1.8° stepper motor, 1/8 step	40
1.8° stepper motor, 1/16 step	80
1.8° stepper motor, 1/32 step	160



DIMENSIONS ARE IN MILLIMETERS [INCH]
TOLERANCES: ANGULAR: ± 1° .X ± 0.2 .XX ± 0.02 .XXX ± 0.005
THIRD ANGLE PROJECTION
DATE --
MATERIAL --
WEIGHT 5.4g
FINISH --

RP One Labs
http://www.rp-one.com
+1 (917) 391-2322
info@rp-one.com
The work is licensed under GPLv3 licence
Drawings - CC BY-SA 3.0 license

TITLE: GT2 20T Aluminum Pulley 8mm Bore

SIZE	DWG. NO.	PART NO.	REV
A	13082102	GT208	1.0

SCALE: 2:1 | DO NOT SCALE DRAWING | SHEET 1 OF 1

Figure 97: GT2 Pulley datasheet

Appendix B: L293D Datasheet

L293x Quadruple Half-H Drivers

1 Features

- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- High-Noise-Immunity Inputs
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

2 Applications

- Stepper Motor Drivers
- DC Motor Drivers
- Latching Relay Drivers

3 Description

The L293 and L293D devices are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN.

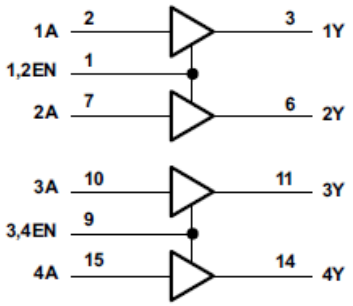
The L293 and L293D are characterized for operation from 0°C to 70°C.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
L293NE	PDIP (16)	19.80 mm × 6.35 mm
L293DNE	PDIP (16)	19.80 mm × 6.35 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Logic Diagram



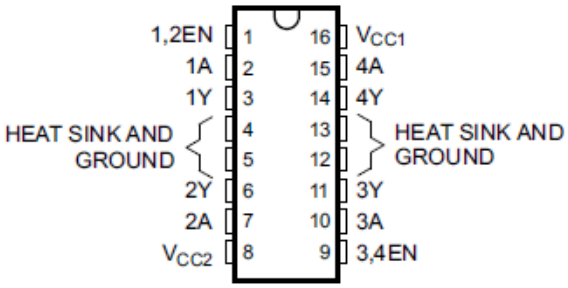
4 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision C (November 2004) to Revision D	Page
• Removed *Ordering Information* table	1
• Added *ESD Ratings* and *Thermal Information* tables, *Feature Description* section, *Device Functional Modes*, *Application and Implementation* section, *Power Supply Recommendations* section, *Layout* section, *Device and Documentation Support* section, and *Mechanical, Packaging, and Orderable Information* section.	1

5 Pin Configuration and Functions

NE Package
16-Pin PDIP
Top View



Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to printed-circuit-board ground plane with multiple solid vias
V_{CC1}	16	—	5-V supply for internal logic translation
V_{CC2}	8	—	Power VCC for drivers 4.5 V to 36 V

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

	MIN	MAX	UNIT
Supply voltage, V_{CC1}(2)		36	V
Output supply voltage, V_{CC2}		36	V
Input voltage, V_I		7	V
Output voltage, V_O	–3	V_{CC2} + 3	V
Peak output current, I_O (nonrepetitive, t ≤ 5 ms): L293	–2	2	A
Peak output current, I_O (nonrepetitive, t ≤ 100 µs): L293D	–1.2	1.2	A
Continuous output current, I_O: L293	–1	1	A
Continuous output current, I_O: L293D	–600	600	mA
Maximum junction temperature, T_J		150	°C
Storage temperature, T_{stg}	–65	150	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values are with respect to the network ground terminal.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2000	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101(2)	±1000	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

			MIN	NOM	MAX	UNIT
	Supply voltage	V_{CC1}	4.5		7	V
		V_{CC2}	V_{CC1}		36	
V_{IH}	High-level input voltage	V_{CC1} ≤ 7 V	2.3		V_{CC1}	V
		V_{CC1} ≥ 7 V	2.3		7	V
V_{IL}	Low-level output voltage		–0.3(1)		1.5	V
T_A	Operating free-air temperature		0		70	°C

(1) The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

6.4 Thermal Information

THERMAL METRIC(1)		L293, L293D NE (PDIP) 16 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance (2)	36.4	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	22.5	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	16.5	°C/W
ψ_{JT}	Junction-to-top characterization parameter	7.1	°C/W
ψ_{JB}	Junction-to-board characterization parameter	16.3	°C/W

6.5 Electrical Characteristics

over operating free-air temperature range (unless otherwise noted)

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		L293: I_{OH} = −1 A		V_{CC2} – 1.8	V_{CC2} – 1.4		V
			L293D: I_{OH} = − 0.6 A					
V_{OL}	Low-level output voltage		L293: I_{OL} = 1 A			1.2	1.8	V
			L293D: I_{OL} = 0.6 A					
V_{OKH}	High-level output clamp voltage		L293D: I_{OK} = –0.6 A			V_{CC2} + 1.3		V
V_{OKL}	Low-level output clamp voltage		L293D: I_{OK} = 0.6 A			1.3		V
I_{IH}	High-level input current	A	V_I = 7 V			0.2	100	μA
		EN				0.2	10	
I_{IL}	Low-level input current	A	V_I = 0			–3	–10	μA
		EN				–2	–100	
I_{CC1}	Logic supply current		I_O = 0	All outputs at high level		13	22	mA
				All outputs at low level		35	60	
				All outputs at high impedance		8	24	
I_{CC2}	Output supply current		I_O = 0	All outputs at high level		14	24	mA
				All outputs at low level		2	6	
				All outputs at high impedance		2	4	

6.6 Switching Characteristics

over operating free-air temperature range (unless otherwise noted) V_{CC1} = 5 V, V_{CC2} = 24 V, T_A = 25°C

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output from A input	L293NE, L293DNE	C_L = 30 pF, See Figure 2		800		ns
		L293DWP, L293N L293DN			750		
t_{PHL}	Propagation delay time, high-to-low-level output from A input	L293NE, L293DNE			400		ns
		L293DWP, L293N L293DN			200		
t_{TLH}	Transition time, low-to-high-level output	L293NE, L293DNE			300		ns
		L293DWP, L293N L293DN			100		
t_{THL}	Transition time, high-to-low-level output	L293NE, L293DNE			300		ns
		L293DWP, L293N L293DN			350		

6.7 Typical Characteristics

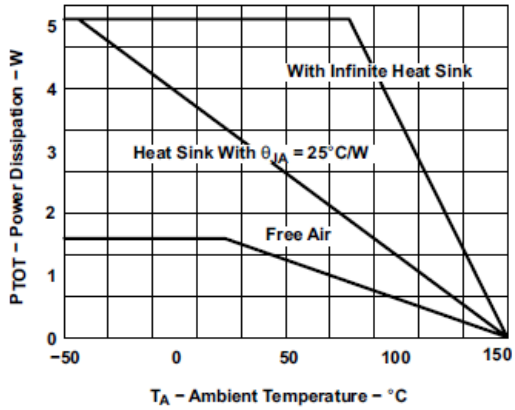
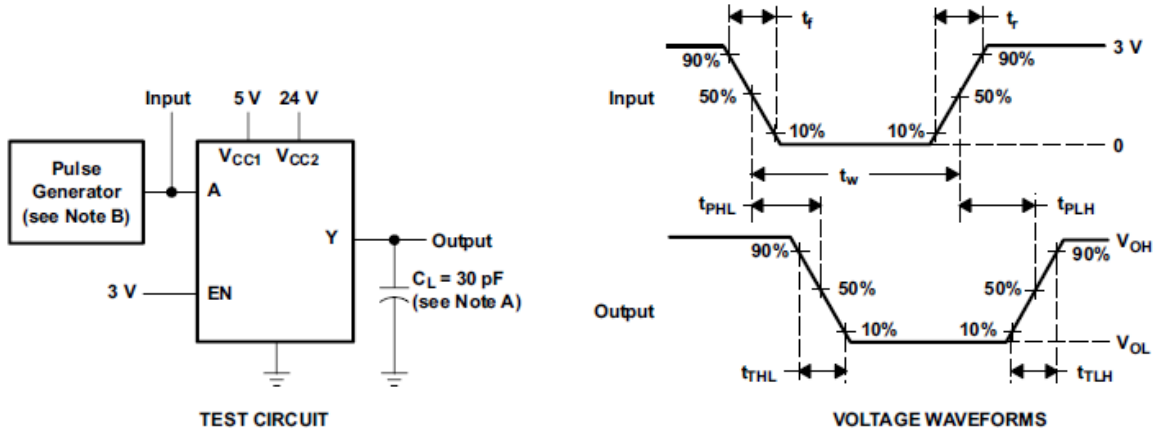


Figure 1. Maximum Power Dissipation vs Ambient Temperature

7 Parameter Measurement Information



NOTES: A. C_L includes probe and jig capacitance.
B. The pulse generator has the following characteristics: $t_r \leq 10$ ns, $t_f \leq 10$ ns, $t_w = 10$ µs, PRR = 5 kHz, $Z_O = 50$ Ω.

Figure 2. Test Circuit and Voltage Waveforms

8 Detailed Description

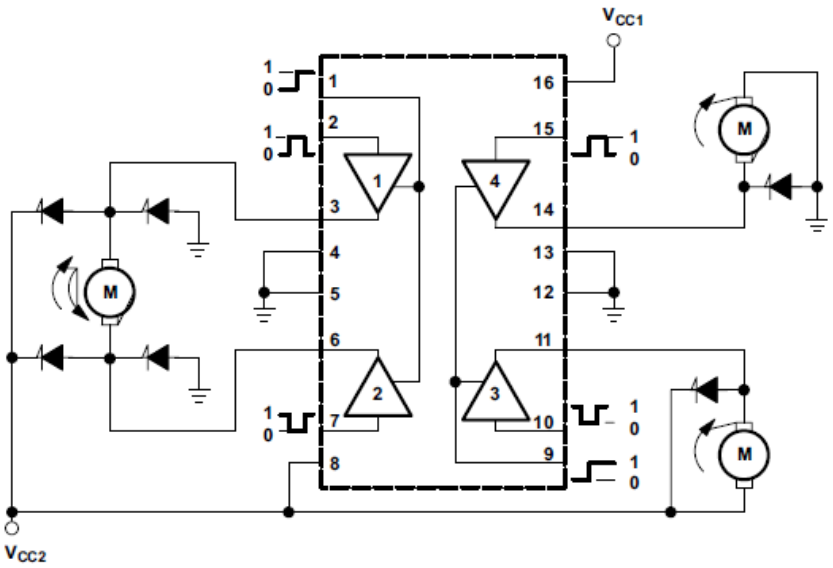
8.1 Overview

The L293 and L293D are quadruple high-current half-H drivers. These devices are designed to drive a wide array of inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current and high-voltage loads. All inputs are TTL compatible and tolerant up to 7 V.

Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression. On the L293D, these diodes are integrated to reduce system complexity and overall system size. A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation. The L293 and L293D are characterized for operation from 0°C to 70°C.

8.2 Functional Block Diagram



Output diodes are internal in L293D.

8.3 Feature Description

The L293x has TTL-compatible inputs and high voltage outputs for inductive load driving. Current outputs can get up to 2 A using the L293.

8.4 Device Functional Modes

Table 1 lists the fuctional modes of the L293x.

Table 1. Function Table (Each Driver)(1)

INPUTS (2)		OUTPUT (Y)
A	EN	
H	H	H
L	H	L
X	L	Z

(1) H = high level, L = low level, X = irrelevant, Z = high impedance (off)
(2) In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.

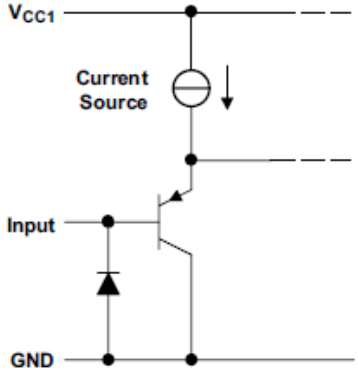


Figure 3. Schematic of Inputs for the L293x

Appendix C: NEMA-17HS8401 Datasheet

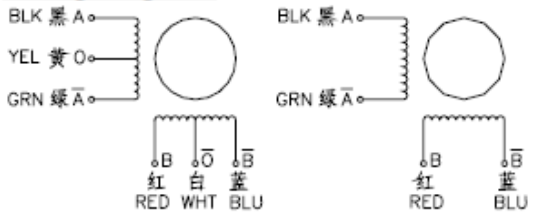
MotionKing

HB Stepper Motor Catalog

MotionKing (China) Motor Industry Co., Ltd.

2 Phase Hybrid Stepper Motor
17HS series-Size 42mm(1.8 degree)

Wiring Diagram:



UNI-POLAR(6 LEADS) BI-POLAR(4LEADS)

Electrical Specifications:

Series Model	Step Angle (deg)	Motor Length (mm)	Rated Current (A)	Phase Resistance (ohm)	Phase Inductance (mH)	Holding Torque (N.cm Min)	Detent Torque (N.cm Max)	Rotor Inertia (g.cm²)	Lead Wire (No.)	Motor Weight (g)
17HS2408	1.8	28	0.6	8	10	12	1.6	34	4	150
17HS3401	1.8	34	1.3	2.4	2.8	28	1.6	34	4	220
17HS3410	1.8	34	1.7	1.2	1.8	28	1.6	34	4	220
17HS3430	1.8	34	0.4	30	35	28	1.6	34	4	220
17HS3630	1.8	34	0.4	30	18	21	1.6	34	6	220
17HS3616	1.8	34	0.16	75	40	14	1.6	34	6	220
17HS4401	1.8	40	1.7	1.5	2.8	40	2.2	54	4	280
17HS4402	1.8	40	1.3	2.5	5.0	40	2.2	54	4	280
17HS4602	1.8	40	1.2	3.2	2.8	28	2.2	54	6	280
17HS4630	1.8	40	0.4	30	28	28	2.2	54	6	280
17HS8401	1.8	48	1.7	1.8	3.2	52	2.6	68	4	350
17HS8402	1.8	48	1.3	3.2	5.5	52	2.6	68	4	350
17HS8403	1.8	48	2.3	1.2	1.6	46	2.6	68	4	350
17HS8630	1.8	48	0.4	30	38	34	2.6	68	6	350

*Note: We can manufacture products according to customer's requirements.

Appendix D: A4988 Datasheet

Allegro MicroSystems, LLC

A4988

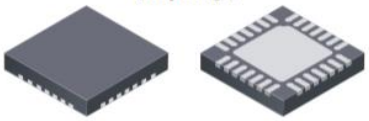
DMOS Microstepping Driver with Translator And Overcurrent Protection

Features and Benefits

- Low $R_{DS(ON)}$ outputs
- Automatic current decay mode detection/selection
- Mixed and Slow current decay modes
- Synchronous rectification for low power dissipation
- Internal UVLO
- Crossover-current protection
- 3.3 and 5 V compatible logic supply
- Thermal shutdown circuitry
- Short-to-ground protection
- Shorted load protection
- Five selectable step modes: full, $^1/_2$, $^1/_4$, $^1/_8$, and $^1/_{16}$

Package:

28-contact QFN
with exposed thermal pad
5 mm × 5 mm × 0.90 mm
(ET package)



Approximate size

Description

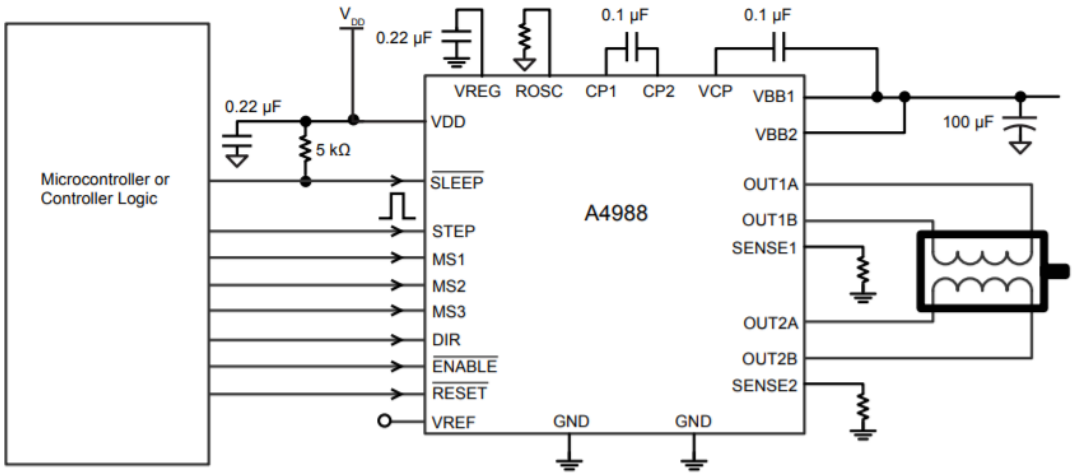
The A4988 is a complete microstepping motor driver with built-in translator for easy operation. It is designed to operate bipolar stepper motors in full-, half-, quarter-, eighth-, and sixteenth-step modes, with an output drive capacity of up to 35 V and ±2 A. The A4988 includes a fixed off-time current regulator which has the ability to operate in Slow or Mixed decay modes.

The translator is the key to the easy implementation of the A4988. Simply inputting one pulse on the STEP input drives the motor one microstep. There are no phase sequence tables, high frequency control lines, or complex interfaces to program. The A4988 interface is an ideal fit for applications where a complex microprocessor is unavailable or is overburdened.

During stepping operation, the chopping control in the A4988 automatically selects the current decay mode, Slow or Mixed. In Mixed decay mode, the device is set initially to a fast decay for a proportion of the fixed off-time, then to a slow decay for the remainder of the off-time. Mixed decay current control results in reduced audible motor noise, increased step accuracy, and reduced power dissipation.

Continued on the next page…

Typical Application Diagram



Description (continued)

Internal synchronous rectification control circuitry is provided to improve power dissipation during PWM operation. Internal circuit protection includes: thermal shutdown with hysteresis, undervoltage lockout (UVLO), and crossover-current protection. Special power-on sequencing is not required.

The A4988 is supplied in a surface mount QFN package (ES), 5 mm × 5 mm, with a nominal overall package height of 0.90 mm and an exposed pad for enhanced thermal dissipation. It is lead (Pb) free (suffix –T), with 100% matte tin plated leadframes.

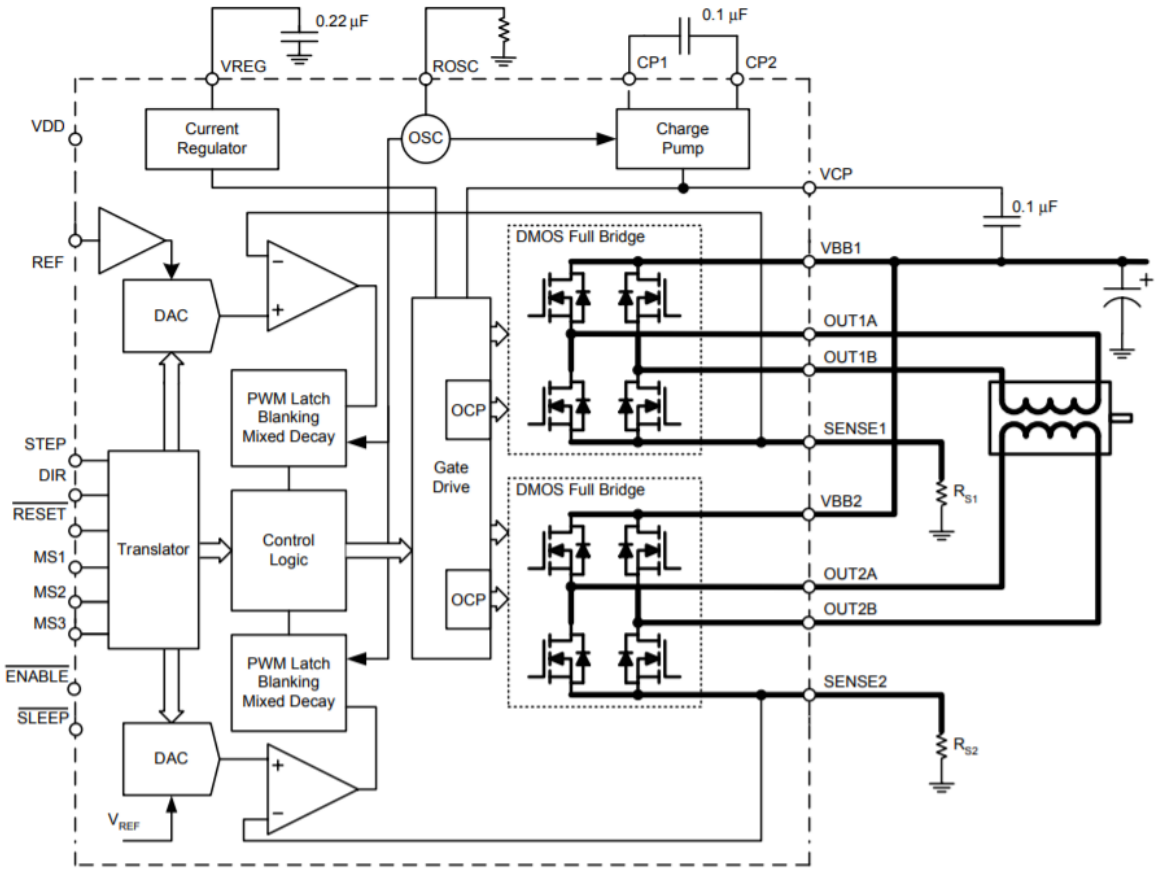
Selection Guide

Part Number	Package	Packing
A4988SETTR-T	28-contact QFN with exposed thermal pad	1500 pieces per 7-in. reel

Absolute Maximum Ratings

Characteristic	Symbol	Notes	Rating	Units
Load Supply Voltage	V_{BB}		35	V
Output Current	I_{OUT}		±2	A
Logic Input Voltage	V_{IN}		–0.3 to 5.5	V
Logic Supply Voltage	V_{DD}		–0.3 to 5.5	V
Motor Outputs Voltage			–2.0 to 37	V
Sense Voltage	V_{SENSE}		–0.5 to 0.5	V
Reference Voltage	V_{REF}		5.5	V
Operating Ambient Temperature	T_A	Range S	–20 to 85	°C
Maximum Junction	$T_J(max)$		150	°C
Storage Temperature	T_{stg}		–55 to 150	°C

Functional Block Diagram



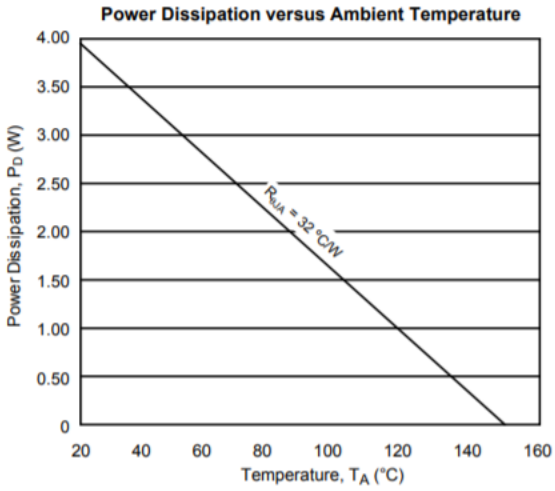
ELECTRICAL CHARACTERISTICS[1] at T_A = 25°C, V_{BB} = 35 V (unless otherwise noted)

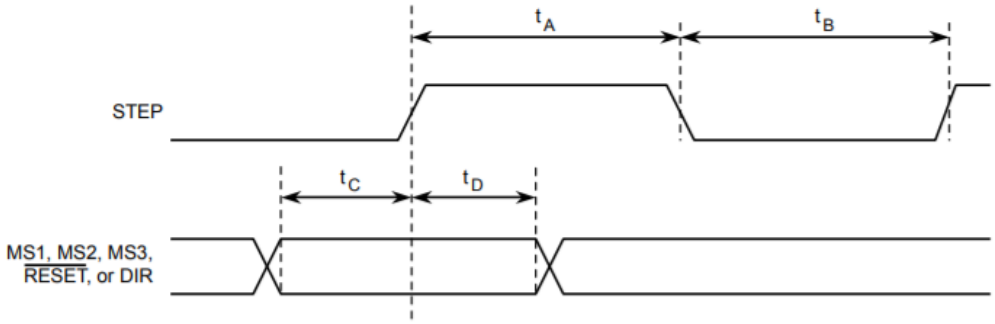
Characteristics	Symbol	Test Conditions	Min.	Typ.[2]	Max.	Units
Output Drivers						
Load Supply Voltage Range	V_{BB}	Operating	8	–	35	V
Logic Supply Voltage Range	V_{DD}	Operating	3.0	–	5.5	V
Output On Resistance	R_{DSON}	Source Driver, I_{OUT} = –1.5 A	–	320	430	mΩ
		Sink Driver, I_{OUT} = 1.5 A	–	320	430	mΩ
Body Diode Forward Voltage	V_F	Source Diode, I_F = –1.5 A	–	–	1.2	V
		Sink Diode, I_F = 1.5 A	–	–	1.2	V
Motor Supply Current	I_{BB}	f_{PWM} < 50 kHz	–	–	4	mA
		Operating, outputs disabled	–	–	2	mA
Logic Supply Current	I_{DD}	f_{PWM} < 50 kHz	–	–	8	mA
		Outputs off	–	–	5	mA
Control Logic						
Logic Input Voltage	$V_{IN(1)}$		V_{DD}×0.7	–	–	V
	$V_{IN(0)}$		–	–	V_{DD}×0.3	V
Logic Input Current	$I_{IN(1)}$	V_{IN} = V_{DD}×0.7	–20	<1.0	20	µA
	$I_{IN(0)}$	V_{IN} = V_{DD}×0.3	–20	<1.0	20	µA
Microstep Select	R_{MS1}	MS1 pin	–	100	–	kΩ
	R_{MS2}	MS2 pin	–	50	–	kΩ
	R_{MS3}	MS3 pin	–	100	–	kΩ
Logic Input Hysteresis	$V_{HYS(IN)}$	As a % of V_{DD}	5	11	19	%
Blank Time	t_{BLANK}		0.7	1	1.3	µs
Fixed Off-Time	t_{OFF}	OSC = VDD or GND	20	30	40	µs
		R_{OSC} = 25 kΩ	23	30	37	µs
Reference Input Voltage Range	V_{REF}		0	–	4	V
Reference Input Current	I_{REF}		–3	0	3	µA
Current Trip-Level Error[3]	err_I	V_{REF} = 2 V, %$I_{TripMAX}$ = 38.27%	–	–	±15	%
		V_{REF} = 2 V, %$I_{TripMAX}$ = 70.71%	–	–	±5	%
		V_{REF} = 2 V, %$I_{TripMAX}$ = 100.00%	–	–	±5	%
Crossover Dead Time	t_{DT}		100	475	800	ns
Protection						
Overcurrent Protection Threshold[4]	I_{OCPST}		2.1	–	–	A
Thermal Shutdown Temperature	T_{TSD}		–	165	–	°C
Thermal Shutdown Hysteresis	T_{TSDHYS}		–	15	–	°C
VDD Undervoltage Lockout	V_{DDUVLO}	V_{DD} rising	2.7	2.8	2.9	V
VDD Undervoltage Hysteresis	$V_{DDUVLOHYS}$		–	90	–	mV

THERMAL CHARACTERISTICS

Characteristic	Symbol	Test Conditions*	Value	Units
Package Thermal Resistance	$R_{\theta JA}$	Four-layer PCB, based on JEDEC standard	32	ºC/W

*Additional thermal information available on Allegro Web site.





Time Duration	Symbol	Typ.	Unit
STEP minimum, HIGH pulse width	t_A	1	µs
STEP minimum, LOW pulse width	t_B	1	µs
Setup time, input change to STEP	t_C	200	ns
Hold time, input change to STEP	t_D	200	ns

Figure 1: Logic Interface Timing Diagram

Table 1: Microstepping Resolution Truth Table

MS1	MS2	MS3	Microstep Resolution	Excitation Mode
L	L	L	Full Step	2 Phase
H	L	L	Half Step	1-2 Phase
L	H	L	Quarter Step	W1-2 Phase
H	H	L	Eighth Step	2W1-2 Phase
H	H	H	Sixteenth Step	4W1-2 Phase

Appendix E: Servo Motor TD8120MG Datasheet

Applicably Environmental Condition

Storage Temperature Range	-30°C ~ 80°C
Temperature Range	-25°C ~ 70°C

Standard Test Environment

Temperature Range	-25°C ~ 70°C
Humidity Range	65%±10%

Mechanical Specification

Size	40.0*20.5*40.5mm
Weight	56g±5%
Gear Type	5 Metal
Limit Angle	360°
Bearing	2BB
Horn Gear Spline	25T 5.9mm
Horn Type	Plastic, POM
Case	Engineering Plastics(PBT)+Aluminum Alloy
Connector Wire	320mm±5mm
Motor	Carbon Brush Motor
Water Resistance	No

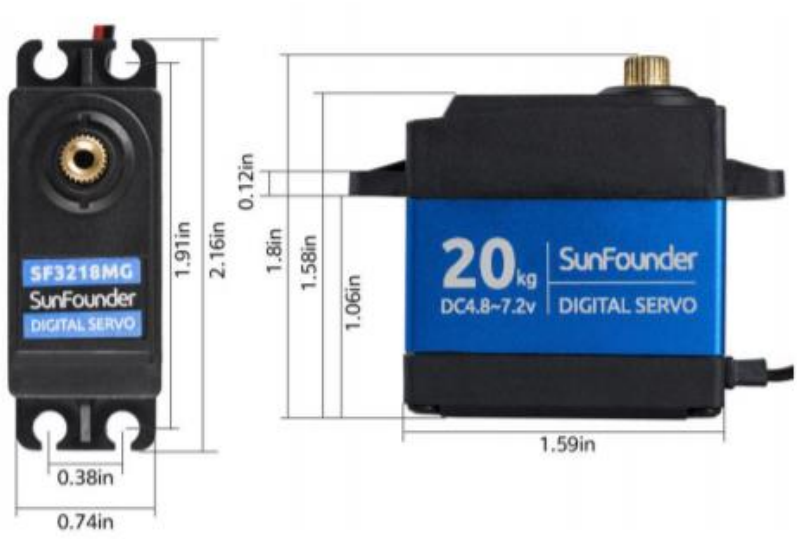
Electrical Specification

Operating Voltage	4.8V	7.2V
Idle Current	mA	mA
No Load Speed	0.18sec/60°	0.14sec/60°
No-load Current;	210mA	260mA
Stall Torque	20.5kg.cm	22.8kg.cm
Stall Current	2100mA±10%	2700mA±10%
Working Voltage Range	4.8-7.2V	

Control Specification

Command Signal	Pulse Width Modification
Amplifier Type	Digital Controller
Pulse Width Range	500～2500usec
Neutral Position	1500usec
Running Degree	270°±3°(when 500～2500usec)
Dead Band Width	5 usec
Rotating Direction	Counterclockwise (when 1000～2000usec)

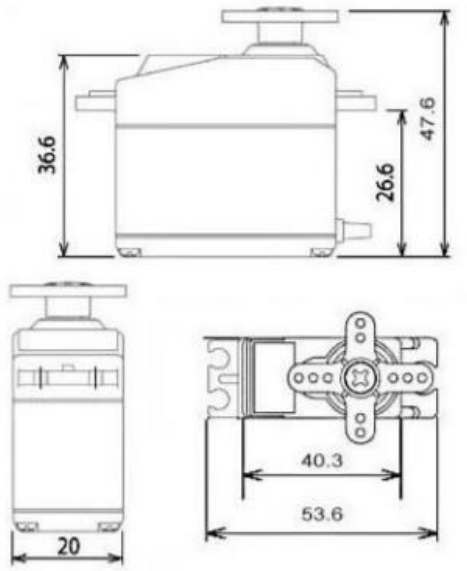
Dimensions



Internal Structure

Appendix F: Servo Motor MG996R

MG996R High Torque
Metal Gear Dual Ball Bearing Servo



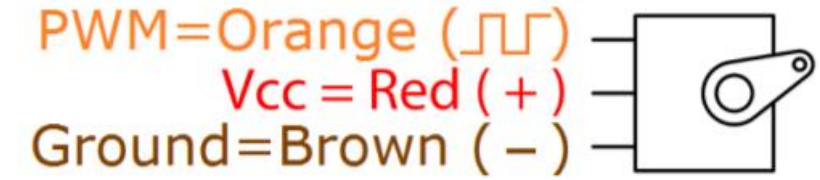
This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

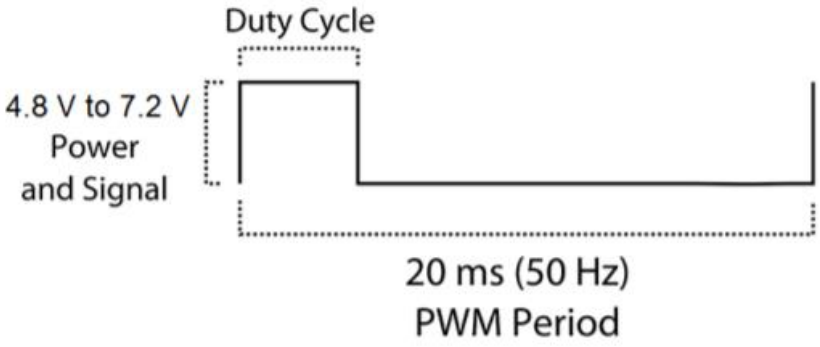
This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC





Appendix G: Source Code

```
//Pin declaration
#define proximity A2            // proximity sensor
#define echoPin 9               // ultrasonic echo pin
#define trigPin 8               // ultrasoni trigger pin
#define servo_grip 5            // pwm servo gripper
#define xstepPin 12             // x axis motor step pin
#define xdirPin 13              // x axis motor direction pin
#define xswitch 7               // x axis endstop switch
#define zstepPin 22             // z axis step pin
#define zdirPin 24              // z axis direction pin
#define zswitch A5              // z endstop switch
#define emergency_button A4     // emergency button
#define en_pwm 6                // dc motor l293d pwm
#define in_1 10                 // dc motor input pin 1
#define in_2 11                 // dc motor input pin 1
#define IRsensor A1             // IR sensor input pin
//#define IRsensor2 A0
int dummy;                      // dummy
int IRsensor_interval = 700;    // IR sensor preset value
int y_current_pos = 1;          // dc motor posiiton
String quantities;              // variable to store the value received through Bluetooth

void setup() {
  Serial.begin(9600);
  Serial1.begin(9600); // Default communication rate of the Bluetooth module

  // Input pin setup
  pinMode(proximity,INPUT);
  pinMode(xswitch,INPUT_PULLUP);
  pinMode(zswitch,INPUT_PULLUP);
  pinMode(emergency_button,INPUT_PULLUP);
  pinMode(echoPin, INPUT);
  pinMode(IRsensor,INPUT);

  // Output pin setup
  pinMode(servo_grip, OUTPUT);
  pinMode(xstepPin,OUTPUT);
```

```
  pinMode(xdirPin,OUTPUT);
  pinMode(zdirPin,OUTPUT);
  pinMode(zstepPin,OUTPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(en_pwm,OUTPUT) ;
  pinMode(in_1,OUTPUT) ;
  pinMode(in_2,OUTPUT) ;

}


// Bluetooth function
int bluetooth(int arr[12])
{
   int i;
   int j;

   int y = 0;
   for (i = 0 ; i < 12; i++)
   {
    if(Serial1.available() > 0) // Checks whether data is coming from the serial port
    {
      quantities = Serial1.readString(); // Reads the data from the serial port
      Serial.print("quantities");
      Serial.println(quantities);
      y=1;
    }
   }
   if (y==1)
   {
    for (j = 0 ; j < 12; j++)
    {
       arr[j] = (quantities.charAt(j)-48);
       Serial.println(arr[j]);
    }
    y=0;
   }
}
```

```
// x axis motor control
// comand HIGH will move to home, vice versa. travel is travel distance in mm
int xaxis(int xcommand, int travel)
{
  // steppin = 4.8 step/mm
  // run until proximity sensor send feedback
  int delaytime = 1000;
  double xtravel = 4.8 * travel;
  digitalWrite(xdirPin, xcommand); // Enables the motor to move in a particular direction
  for (int x = 0; x < xtravel; x++)
  {
    digitalWrite(xstepPin, HIGH);
    delayMicroseconds(delaytime);
    digitalWrite(xstepPin, LOW);
    delayMicroseconds(delaytime);
  }
  delay(1000); // One second delay
}

// z axis motor control
// comand HIGH will move to home, vice versa. travel is travel distance in mm
int zaxis(int zcommand, int travel)
{
  // steppin = 4.8 step/mm
  // run until proximity sensor send feedback
  int delaytime = 800;
  double zstep = 25 * travel; //4.8
  digitalWrite(zdirPin, zcommand); // Enables the motor to move in a particular direction
  for (int x = 0; x < zstep; x++)
  {
    digitalWrite(zstepPin, HIGH);
    delayMicroseconds(delaytime);
    digitalWrite(zstepPin, LOW);
    delayMicroseconds(delaytime);
  }
  delay(1000); // One second delay
}
```

```
// X axis homing function
// X motor will turn until the endstop switch clicked
void xhome()
{
  int delaytime = 1000;
  int tick = digitalRead(xswitch);
  digitalWrite(xdirPin, HIGH);
  //do while loop to keep stepping the switch
  for(int x = 0; (x < 10000000 && digitalRead(xswitch) == HIGH); x++)
  {
     digitalWrite(xstepPin,HIGH);
     delayMicroseconds(delaytime);
     digitalWrite(xstepPin,LOW);
     delayMicroseconds(delaytime);
    }
  delay(1000); // One second delay
 }

// Z axis homing function
// Z motor will turn until the endstop switch clicked
void zhome()
{
  int delaytime = 800;
  int tick = digitalRead(zswitch);
  digitalWrite(zdirPin, LOW); //direction
  //do while loop to keep stepping the switch
  for(int x = 0; (x < 10000000 && digitalRead(zswitch) == HIGH); x++)
  {
     digitalWrite(zstepPin,HIGH);
     delayMicroseconds(delaytime);
     digitalWrite(zstepPin,LOW);
     delayMicroseconds(delaytime);
  }
  delay(1000); // One second delay
 }

// DC motor turn left in directon
```

```
// go to rack
void DC_move_left (){
  while (analogRead(IRsensor) <= IRsensor_interval)
  {
    digitalWrite(in_1,HIGH) ;
    digitalWrite(in_2,LOW) ;
    analogWrite(en_pwm,255) ;
  }
  digitalWrite(in_1,LOW); // HIGH
  digitalWrite(in_2, LOW); //HIGH
  delay(1000);
}

// DC motor turn right in directon
// go to customer
void DC_move_right()
{
  while (analogRead(IRsensor) <= IRsensor_interval)
  {
    digitalWrite(in_1,LOW) ;
    digitalWrite(in_2,HIGH);
    delay(3000) ;
  }
  digitalWrite(in_1,LOW) ;
  digitalWrite(in_2,HIGH);
  delay(1000);
}

// PWM pulse are been calculate and used
// Servomotor function
// use this function is to avoid the Servo.h file affect the pinout
void servoPulse (int angle, int state)
{
  int pwm = (angle*11) + 500;      // Convert angle to microseconds
  digitalWrite(servo_grip, state); //HIGH OPEN LOW CLOSE
  delayMicroseconds(pwm);
  delay(250);                      // Refresh cycle of servo
}
```

```
// Ultrasonic sensor
  int ultrasonic_sensor(){
  int duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  return distance;
}

//Grip motion function will run after the dc motor funciton (y axis) and stepper motor (z axis) finished
// The x axis stepper motor will run until the ultrasonic sensed value, 300mm.
// Therefore the servomotor will open and go gorward abit, to grip the item
void grip_motion(){
  int delaytime = 3000; //adjust speed
  xaxis(LOW,300);
  delay(1000);
  for (int angle = 60; angle <= 50; angle += 1){servoPulse(angle,HIGH);} //Open Grip
  for(int x = 0; (x < 50 && ultrasonic_sensor()>=4); x++){
      digitalWrite(xstepPin,HIGH);
      delayMicroseconds(delaytime);
      digitalWrite(xstepPin,LOW);
      delayMicroseconds(delaytime);
    }
  delay(10);
  for (int angle = 50; angle <= 60; angle += 1){servoPulse(angle,LOW);} //Grip IN
  delay(1000);
  xhome();
  //xaxis(HIGH,480); //LOW OUT // HIGH IN
  delay(1000);
  for (int angle = 60; angle <= 50; angle += 1){servoPulse(angle,HIGH);} //Open Grip
```

```
}

// the function that move y axis (dc motor) and z axis (z stepper)
int pos_give(int y_target, int z_target)
{
  if(z_target == 1)
  {
    zaxis(HIGH, 50);
  }
  else if(z_target == 2)
  {
    zaxis(HIGH, 300);
  }
  else if(z_target == 3)
  {
    zaxis(HIGH, 520);
  }

  if(y_target == '1')
  {
    if (y_current_pos == '2')
    {
       DC_move_right();
       y_current_pos -= 1;
    }
     if(y_current_pos == '3')
     {
       DC_move_right();
       y_current_pos -= 1;
       DC_move_right();
       y_current_pos -= 1;
     }
  }

  else if(y_target == '2')
  {
    if (y_current_pos == '1')
    {
```

```
    DC_move_left();
    y_current_pos += 1;
  }

  else if(y_current_pos == '3')
  {
    DC_move_right();
    y_current_pos -= 1;
  }

  else if(y_current_pos == '4')
  {
    DC_move_right();
    y_current_pos -= 1;
    DC_move_right();
    y_current_pos -= 1;

  }
}

else if(y_target == '3')
{
  if (y_current_pos == '1')
  {
    DC_move_left();
    y_current_pos += 1;
    DC_move_left();
    y_current_pos += 1;
  }

  else if(y_current_pos == '2')
  {
    DC_move_left();
    y_current_pos += 1;
  }

  else if(y_current_pos == '4')
  {
```

```
      DC_move_right();
      y_current_pos -= 1;
    }
  }

  else if(y_target == '4')
  {
    if (y_current_pos == '1')
    {
      DC_move_left();
      y_current_pos += 1;
      DC_move_left();
      y_current_pos += 1;
      DC_move_left();
      y_current_pos += 1;
    }

    else if(y_current_pos == '2')
    {
      DC_move_left();
      y_current_pos += 1;
      DC_move_left();
      y_current_pos += 1;
    }

    else if(y_current_pos == '3')
    {
      DC_move_left();
      y_current_pos += 1;
    }
  }
}

int decoder(int item[12])
{
  int i;
  int current_item;
  //int current_item_number;
```

```
//int last_item_number;
//int times_of_array_shifted;
//int number_of_turns_in_between_each_height;
//int eq;

for(i= 0; i< 12; i++)
{
  current_item = item[i];
}

// first column
if ((item[1]||item[2]||item[3]) != 0 )
{
  for(int i = 0; i <= item[1]; i ++)
  {
    if(item[1]!= 0)
    {
   pos_give(1, 1);
    }
  }

  for(int i = 0; i <= item[2]; i ++)
  {
    if(item[2]!= 0)
    {
   pos_give(1, 2);
    }
  }

  for(int i = 0; i <= item[3]; i ++)
  {
    if(item[3]!= 0)
    {
   pos_give(1, 3);
    }
  }
}
```

```
else if ((item[4]||item[5]||item[6]) != 0 )
{
  for(int i = 0; i <= item[4]; i ++)
  {
    if(item[4]!= 0)
    {
   pos_give(2, 1);
    }
  }

  for(int i = 0; i <= item[5]; i ++)
  {
    if(item[5]!= 0)
    {
   pos_give(2, 2);
    }
  }

  for(int i = 0; i <= item[6]; i ++)
  {
    if(item[6]!= 0)
    {
   pos_give(1, 3);
    }
  }
}
else if ((item[7]||item[8]||item[9]) != 0 )
{
   for(int i = 0; i <= item[7]; i ++)
  {
    if(item[7]!= 0)
    {
   pos_give(3, 1);
    }
  }

  for(int i = 0; i <= item[8]; i ++)
  {
```

```
    if(item[8]!= 0)
    {
   pos_give(3, 2);
    }
   }

  for(int i = 0; i <= item[9]; i ++)
  {
    if(item[9]!= 0)
    {
   pos_give(2, 3);
    }
  }
}

else if ((item[10]||item[11]||item[12]) != 0 )
{
   for(int i = 0; i <= item[10]; i ++)
  {
    if(item[10]!= 0)
    {
   pos_give(4, 1);
    }
  }

  for(int i = 0; i <= item[11]; i ++)
  {
    if(item[11]!= 0)
    {
   pos_give(4, 2);
    }
  }

  for(int i = 0; i <= item[12]; i ++)
  {
    if(item[12]!= 0)
    {
   pos_give(4, 3);
```

```
    }
   }
  }
}

// Main function
// The machine stop when the emergency button press, else will work after received the order
//
void loop() {
   int y_target = 0;
   int z_target = 0;
   /*
   while (digitalRead(emergency_button)== LOW)
   {
    bluetooth();
    while (true)
    {
     //the serial received certain value
     //User type row value
     if (Serial.available() > 0)
     {
      Serial.print("Input your Y: ");
      y_target = Serial.read();
      if (y_target != 0)
      {
       Serial.print("I received: ");
       Serial.println(y_target);
       break;
      }
     }
    }
  }
  while (true) {
  delay(3000);
  // user type column value
  if (Serial.available() > 0) {
   Serial.print("Input your Z: ");
   z_target = Serial.read();
```

```
    if (z_target!= 0)
    {
      Serial.print("I received: ");
      Serial.println(z_target);
      break;
    }
  }
}*/
int arr[12];
 while (digitalRead(emergency_button)== HIGH)  //emergency button
 {
   while(true)
   {
     xhome();
     delay(1000);
     zhome();
     bluetooth(*arr);
     Serial1.println(arr[1]);
   }
 }

 int item[12];
    delay(2000);
    decoder(*item);
    Serial1.println(item[1]);
    delay(1000);
    grip_motion();
    delay(1000);

}
```