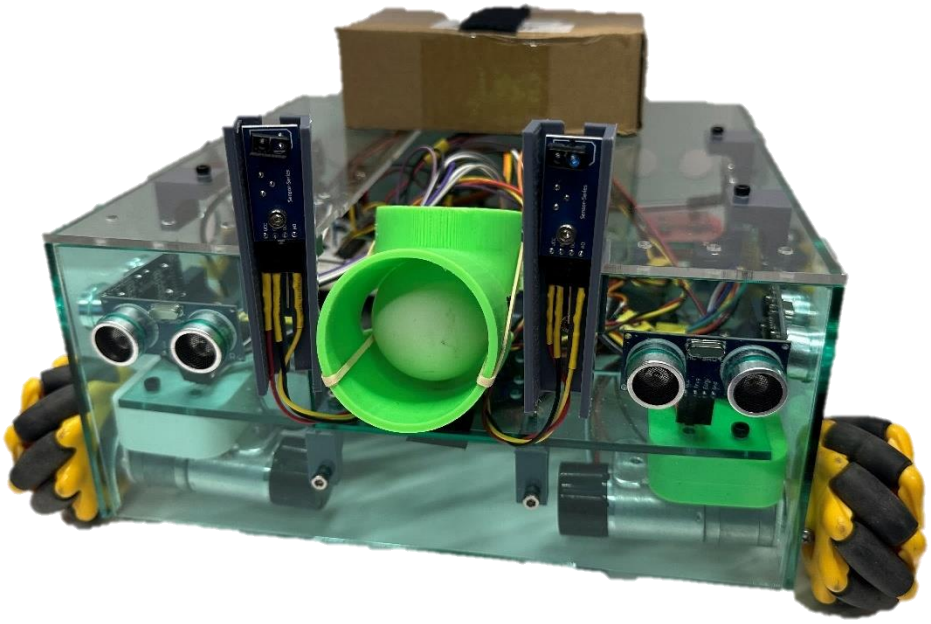
Crab Rave



Saira Billah

Taaseen Jahan

Azam Khan

Tiffany Shum

ME412: Autonomous Mobile Robots

Fall 2024

Submitted: 12/20/24

Prof. Ericson Mar

1. Abstract

Crab Rave (CR) is an autonomous mobile robot that competed in the Fall 2024 Robot Tank Battle competition at the Cooper Union. It features 6 ultrasonic sensors, an IR sensor, and 4 mecanum wheels with corresponding DC motors. It also has a rubber band powered ping pong ball cannon that is controlled by a servo motor. Some notable features are the mecanum wheels, which allow it to translate as well as roll, the suspension cantilevers, and the ping pong ball cannon.

The chassis is made of laser cut acrylic sheets, which are connected using printed parts and small hardware. Custom parts, such as the cannon or the motor mounts and hubs are also 3D printed from PLA. An Arduino mega is used for taking signal inputs and sending motor outputs to the H-Bridge motor drivers.

CR performed okay during the competition. While it completed many achievements, including shooting the opponent's base and returning home, it received many penalties for collisions and emergency teleports.

The complete CAD model of CR can be found using the following link: https://cad.onshape.com/documents/55de810b669197ba1e31ad48/w/23a260eb5b22848c58e4888b/e/6b84b03f303fd490d2508a06up2 | Crab Rave

Table of Contents

2. Introduction

2.1. Objective

CR is an autonomous mobile robot built to compete in The Cooper Union Robot Tank Battle 2024 competition. The main objective for CR is to score points by shooting 400-millimeter ping pong balls at the enemy robot and its base. The robot must navigate and avoid obstacles in the arena while searching for the enemy beacon, an infrared (IR) light source placed in the center of the base. CR is allowed to only carry one ping pong ball at a time but is allowed to reload when docked at its home base. [1]

2.2. Arena

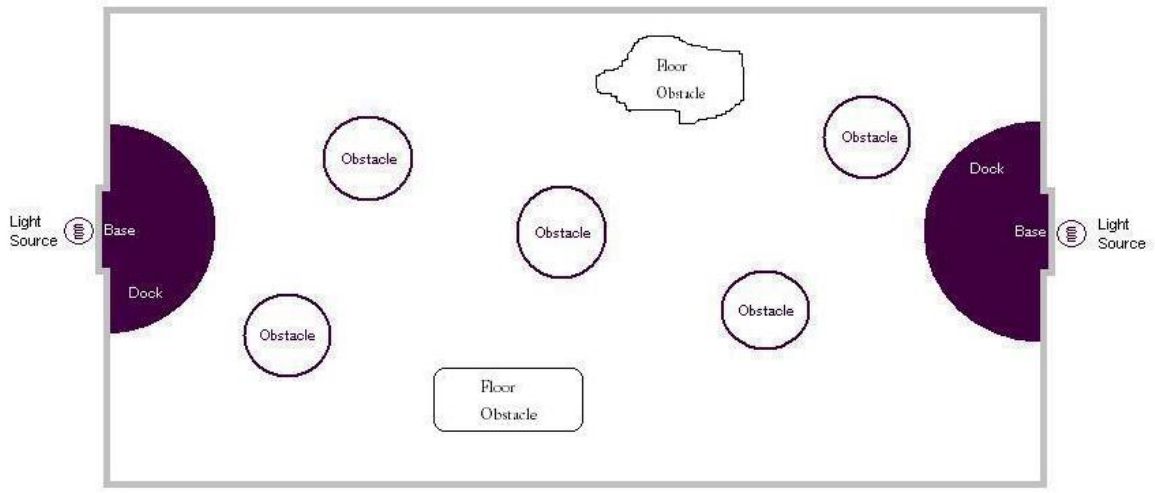


Figure 1. Diagram of the arena [1]

The Arena is a 10’ x 6’ box with a semi-circular dock on either end. Each dock has an IR beacon slightly recessed into the wall 6 inches above the ground. The docks are separated by a playing field that is randomly filled with 1 to 5 obstacles that must be avoided. 2 floor obstacles with a maximum height of 3/8” are placed in the arena that the robot must be able to drive over or avoid.

2.3. Robot Constraints

1. A robot must fit into a box 1 foot long by 1 foot wide by 10 inches high in all modes of operation. There may be no extension of robot parts beyond this space. If a team wishes to place a decorative item that has no effect on the game, then that item will not be counted as part of the robot for size purposes, however, will be counted as part of the robot for scoring (described below) purposes.
2. A robot may not intentionally separate into multiple parts or leave any parts in the arena other than its fired ping pong ball.
3. A robot may not have any parts or devices that are designed to intentionally damage the arena, another robot, or humans. [1]

2.4. General Rules

1. A robot can carry one ball maximum at any given time.
2. A robot is considered "docked" when any part of the robot covers part of its **home** dock from a vertical perspective.
3. A robot may be re-loaded by the human team only when it is docked as described in "Robot Docking and Reloading" below.
4. A ball released by a robot is considered "dead" (i.e. it cannot score any points):
    a. after it ejects from a docked robot (no reward for shooting from home base)
    b. after it touches an object other than the ground (e.g. it may be rolled but may not ricocheted off the wall towards the opponent)
    c. after it touches the releaser itself (e.g. the robot may not "guide" the ball with its own body)

d. after it comes to a complete stop.

e. after it has reached a height of 1 foot from the ground (no shooting over collidable obstacles!)

5. All other times, the released ball is considered “live” (i.e. it can score points).
6. There may be ping pong balls strewn around the arena during the game. Consider this “debris”. There will be an attempt to remove any debris from the arena if it is approximately 4 feet or a greater distance from any fielded robot. A team may remove debris from its home dock to clear the area for their robot to be placed, but they cannot place it back somewhere else. Once dock debris is “disturbed” by the team’s human actions it must be removed for good. [1]

2.5. Scoring

4. Opponent Base Hit (10 pts): When a robot releases a ball that touches its opponent’s base while the ball is live, this is considered an opponent base hit.
5. Docked Robot Hit (10 pts): When a robot releases a ball that touches the docked opponent robot while the ball is live, this is considered a docked robot hit.
6. Fielded Robot Hit (20 pts): When a undocked robot releases a ball that touches the undocked opponent robot while the ball is live, this is considered a fielded robot hit. Get your robots into the field!
7. Robot Self Hit (0 pts): If the live ball touches the robot where the ball originated from, that ball is considered “dead” (no penalty will be given to a robot hit by its own ball by catching up to it, but the robot may not “guide” the ball with its own body, either. See above Game Rules section).

8. Home Base Hit (-10 pts): When a robot releases a ball that touches its own home base while the ball is live, this is considered a home base hit. Be careful not to shoot your own base!
9. Collision (-10 pts. / round): When a robot comes into contact with anything other than a ball, a floor obstacle, or the opponent robot, it is considered a collision. The penalty will only be assessed once per round. No penalties will be assessed for collisions with other robots or caused by robot-robot-contact. In the case of two robots getting entangled with each other or otherwise causing a potentially negative effect through extended physical contact, they can be removed from the arena and reset at their home dock with no penalty. This can be ordered by the referee.
10. Emergency Teleport (-10 pts.): If desired, a team may recover their non-docked robot as if it has docked (as in Robot Docking and Reloading below). [1]

3. Design

3.1. Overview

CR is designed with ultrasonic sensors on most sides that detect obstacles and can react to avoid them by moving in the opposite direction from the detection. CR uses mecanum wheels, a specialized type of wheel that allow for lateral and diagonal translation in addition to regular rolling motion. This allows CR to move laterally until it no longer detects an obstacle before continuing forward. All four of these wheels are driven by 12V DC motors that can spin as fast as 125 RPM.

CR has a front mounted IR sensor to detect targets. It also has a ping pong ball cannon that uses a stretched rubber band held in place with a servo motor. When the IR sensor detects a target the robot releases the servo, which launches the ball.

An Arduino Mega 2560 is used to program CR along with a mini breadboard that handles the power distribution for each of the sensors and motors. CR has multiple states that allow it to respond to different obstacles and situations. As an example, in its default state it moves forward, however it switches to another state when an obstacle is detected to avoid it. This will be explored in depth later in the Programming section.

Table 1. Bill of Materials

Item	Quantity	Provider	Model #
Mecanum Wheels	4	Adafruit	48mmx25mm
DC Encoder Metal Gearmotor 12V High Speed 150RPM	2	Bemonoc	25GA370
12V, 150RPM 30:1 Gear Motor w / Encoder	2	CYTRON	RB-Cyt-82
HC-SR04 Ultrasonic sensor	6	HiLetgo	3-01-0008-A

TCRT5000 Infrared Reflective Sensor	2	HiLetgo	3-01-0485
SG90 Micro servo	1	WWZMDiB	B0BKPL2Y21
Arduino Mega 2560 REV3	1	Arduino	2152366
L293D Quadruple Half-H Drivers	2	BOJACK	BJ-L293
Acrylic	12”x24”x0.125”	Canal Plastics	Transparent Green
Capacitor	2		30pF
12V 2200mAh NiMH battery pack	1	Tectra	RC-AA12V22AA
M3 hardware	48	Qiwuhai	

3.2. Mechanical Design

CR was designed to prioritize navigation and maximize inner storage for its electronics. Thus, the initial design incorporated some form of omni-directional wheels and a box-like design. The purpose of this design was to allow the CR to navigate within a rectangular arena without ever having to return. With side-mounted sonar sensors, CR would only ever move in directions that would put these sensors in direct line of sight with the walls of the arena and obstacles. Furthermore, the front-mounted IR sensors would be aimed towards the enemy beacon and the CR would be able to return home without having to turn.

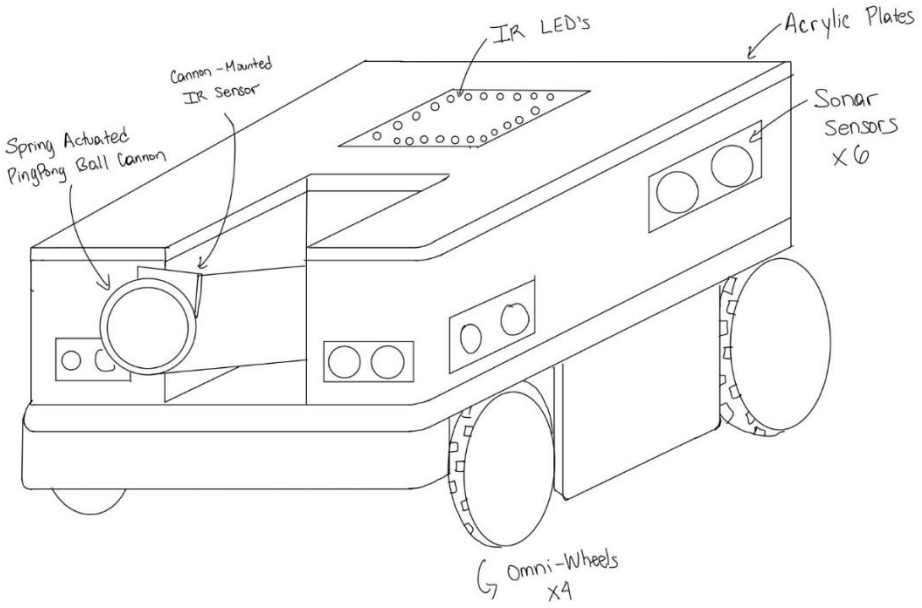


Figure 2. Early sketch of CR design

3.2.1. Frame

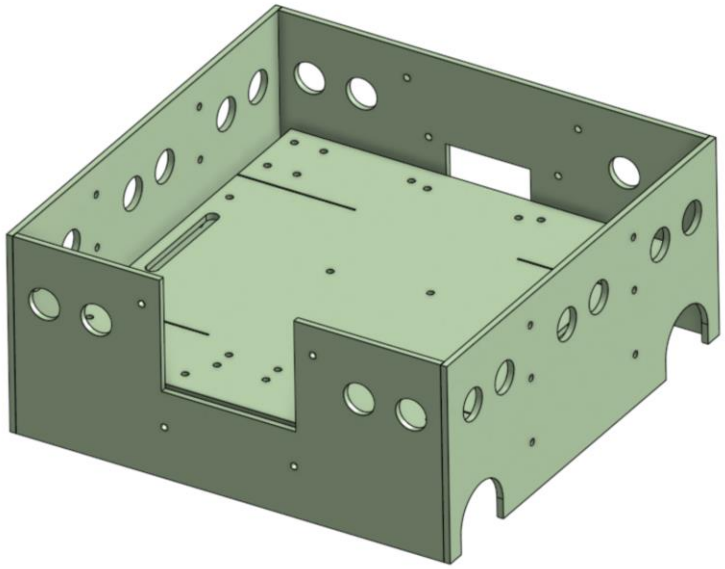
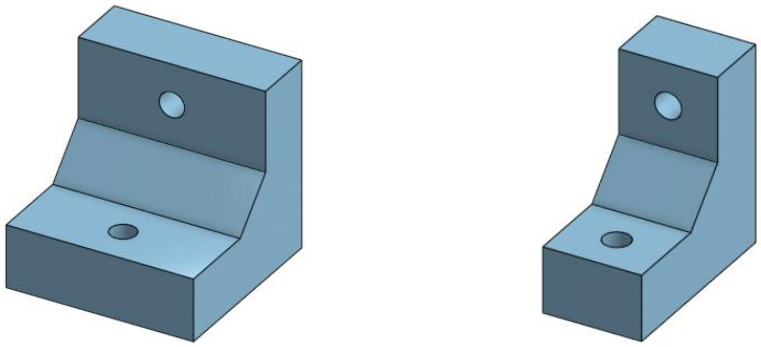


Figure 3. CR's shell

CR's frame is constructed out of 0.125" thick laser cut acrylic sheets. It has a main baseplate with a wall on each side. There are cutouts on each side for ultrasonic sensors, a small cutout on the rear for connecting the Arduino to a computer, and a large cutout at the front for the ping pong ball cannon. Each of these walls has hole cutouts to be held together by 3D printed L-brackets and M3 hardware. The baseplate has M3 mounting holes for constraining the Arduino Mega 2560, motors, and cannon. Slot cutouts on the sides allow wires from the below-mounted motors to route through.



(a) (b)

Figure 4. (a) is the thicker L-bracket used for mounting the lid and sides to the baseplate while (b) is the thinner L-bracket used for mounting the front and rear acrylic plates.

CR has 4 wheels, however 4 wheeled robots can struggle to traverse uneven terrain, as the 4 points of contact are over constrained on a single plane. The solution is to add compliance to the wheels, which allows all 4 wheels to maintain constant contact with the ground. CR does this by adding slits to the baseplate, which means the wheels are now mounted on a cantilever. As the cantilever bends it acts like a suspension system, ensuring that the wheels maintain contact with the ground.

3.2.2. Canon

Figure 5: CR's cannon

The cannon is 3D printed and mounted to the chassis using M3 hardware. It is mounted at a slight angle to fire the ping pong ball slightly above horizontal. A rubber band is stretched through the front hole of the cannon and out the back and held in place by a servo motor. A ping pong ball can then be loaded through the front hole. When the servo rotates, it releases the rubber band, which launches the ping pong ball. Since the servo does not have a flat base, a 3D-printed base adapter was used to hot-glue the servo to the base plate.

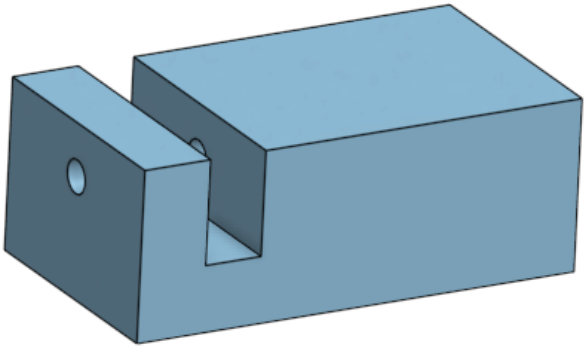
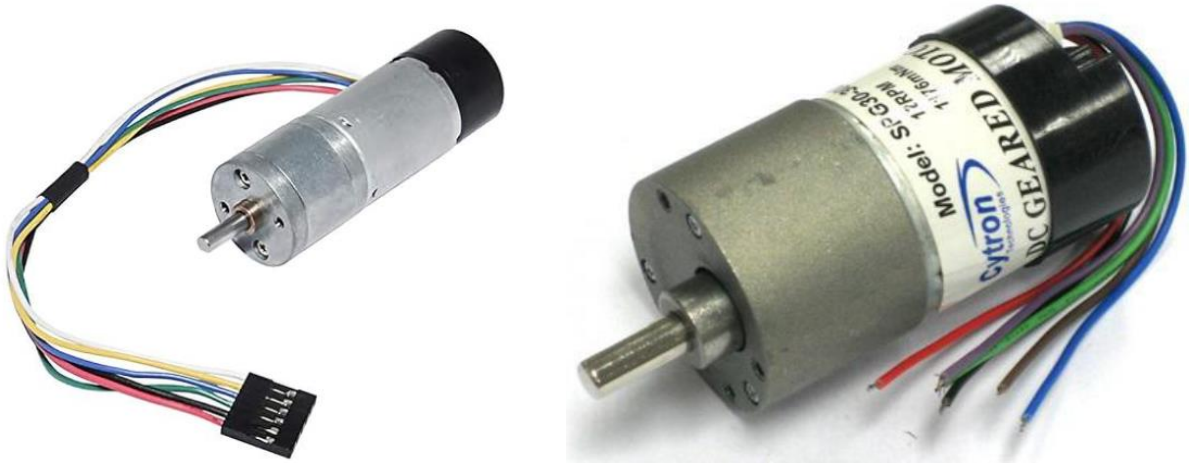


Figure 6. 3D-printed base adapter for mounting the uneven bottom of the servo motor

3.2.3. Motors

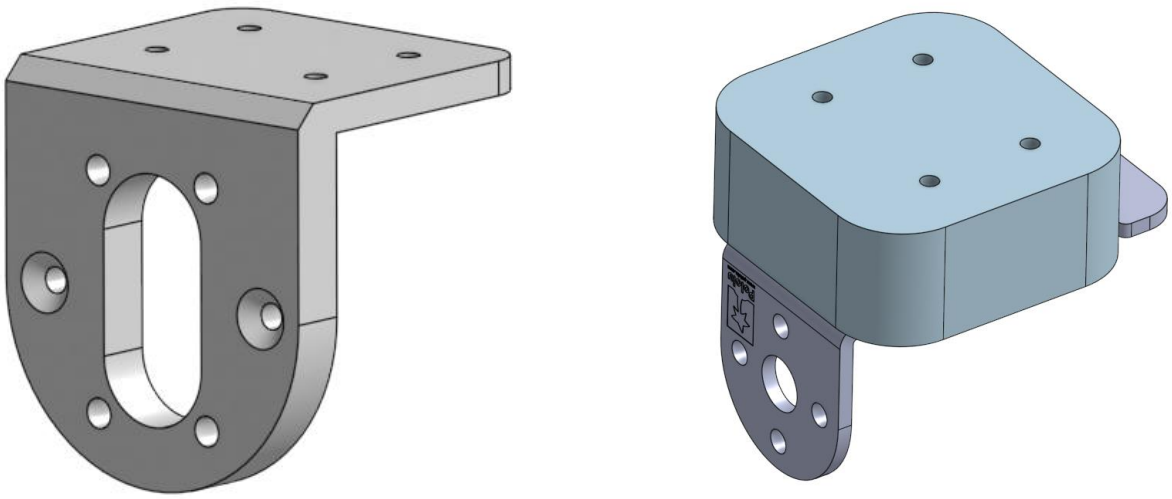


(a) (b)

Figure 7. Front Motor (a) [2] and Rear motor (b) [3]

CR used two different types of motors, which was not a design choice, but rather optimization of parts that were already available. The motors used in the front wheels were Bemonoc 12V 150RPM motors, while the motors used in the rear were CYTRON 12V 150RPM geared motors, with encoders (that were not used). The motors were coupled with the wheels using 3D printed hubs and powered using H-bridge drivers. Since both types of motors were brushed DC motors, they were each connected to two H-bridges with opposite polarities, allowing them to be run both forward and in reverse.

To mount the motors to the chassis, two different types of motor brackets were designed. Using M3 hardware, the brackets are mounted on the face of the motors and allow them to hang from the bottom of the baseplate. To conserve space for the electronics, the motors were mounted closer to the edge of the baseplate, pushing the wheels outboard. Since the front motors are smaller than the rear motors, a different bracket system was needed. Pololu motor brackets designed to fit their 25mm DC motors were found and used to mount the Bemonoc motors. Due to the difference in motor sizes, the spacer was created for the Pololu motor bracket to make CR level with the ground.



(a) (b)

Figure 8. (a) is the rear motor bracket and (b) is the Pololu motor bracket and custom spacer

3.2.4. Wheels

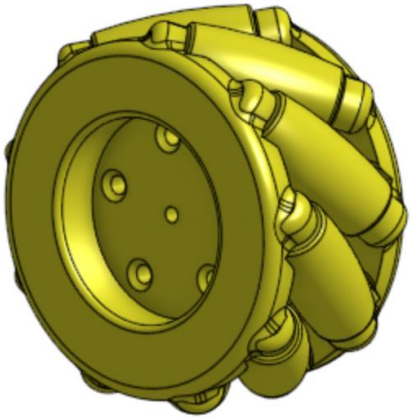


Figure 9. Mecanum wheels

CR used mecanum wheels, a special type of wheel that allows for translational movements of the robot if the wheels are spun in different directions. A simple diagram of this can be seen below.

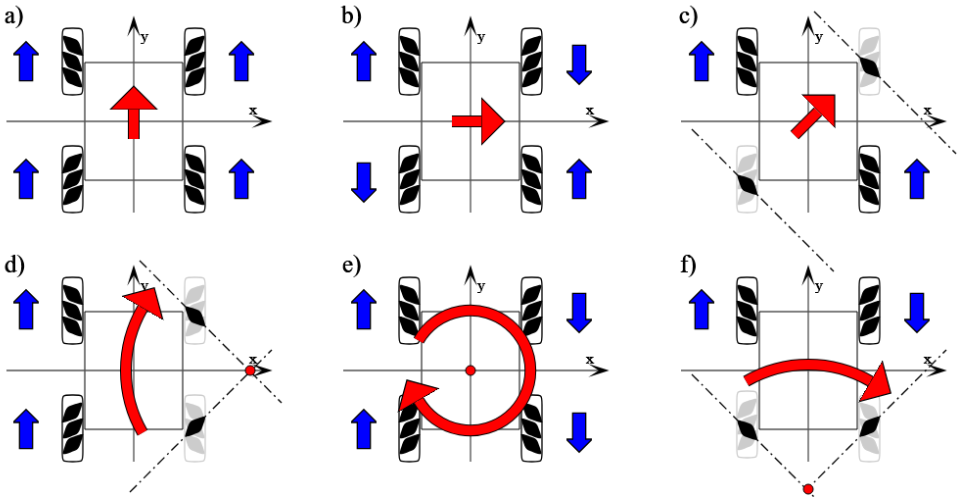


Figure 10. Mecanum wheels guide [4]

These wheels were chosen as they allowed for additional freedom over conventional wheels. CR could traverse the length of the arena, translating itself left or right to avoid obstacles, while staying oriented towards the enemy base the whole time. In practice the wheels did not perform ideally, as they would occasionally lose their grip on the dusty floor, leading to partial rotation. This error would compound over time, leading to the CR getting slowly turned around.

Since no suitable motor hubs that could properly fit to these wheels could be found online, custom hubs were designed and 3D-printed to mount the wheels to the motors. The hubs make use of the center hexagonal hole in the mecanum wheels to constrain the hubs with M3 hardware. On the motor end, the hubs connect to the D-shaft of each motor using a small M3 set screw. Since the motors are different sizes, two motor hubs were designed for each D-shaft size. While the rear motor shaft fit into the hub tightly, the front motor shaft was a looser fit, and thus using the set screw the tighten the shaft against the hub caused the wheel to sit slightly rotated. This also could have caused the slight veering and turning of CR mentioned above.

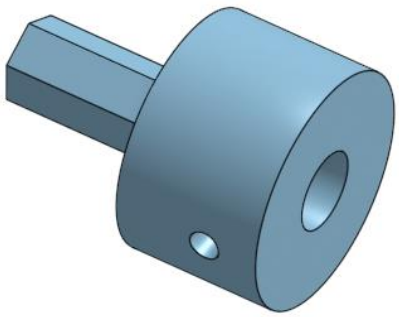


Figure 11. Rear motor hub

3.2.4. IR Sensor Mounts

Since CR has an overall height of only 5 inches and the enemy beacon sits at 6 inches, a mount was needed to aim the IR sensors at this height. The IR mount is modular and holds a slit

gap in the rear so that the IR sensor can be constrained using an M3 screw. While the IR sensors may sit at 6 inches in the CAD, it may be necessary to adjust the height of the sensors in real life to get the best readings from the IR beacon. Thus, the rear slit allows the IR to slide up and down the mount and once the perfect height is found, tightening the M3 screw will hold the IR in place. This mount increases the overall height of the robot to about 5-6” tall so robot beacon for CR can no longer be placed on the lid as shown in the initial sketch. Instead, the robot beacon was attached to a 1-inch-tall box using Velcro that was hot-glued to the lid of CR.

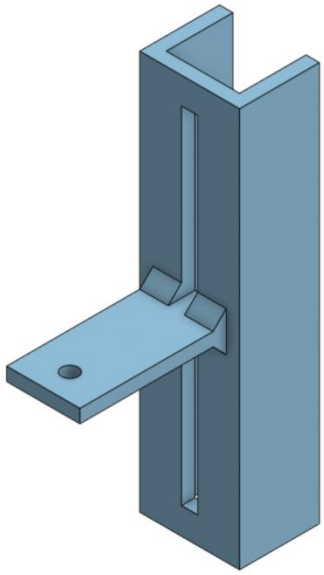


Figure 12. IR Mount

3.3 Electrical Design

3.3.1. Arduino Mega

This robot utilizes an Arduino as a microcontroller due to its simplicity and flexible application to a wide range of circuits. The Mega model was specifically chosen for having a

greater number of output pins, which was necessary for the number of sensor inputs and motor outputs, with each motor using two H-bridges.

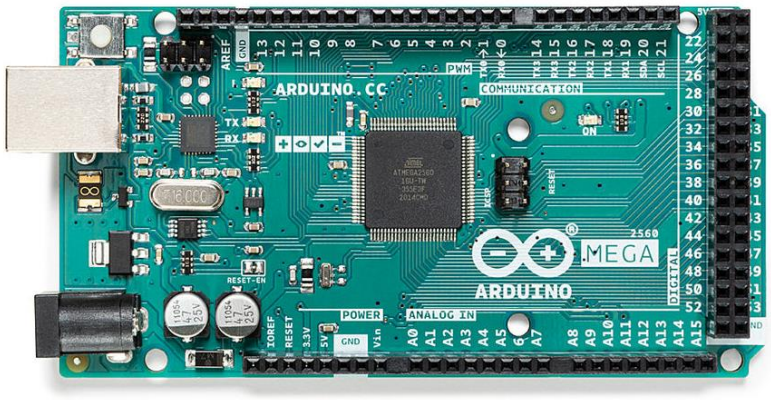


Figure 13. Arduino Mega 2560 Rev3 [5]

3.3.2. Motor Driver

Two L293D quadruple half-H drivers were used to control the direction of the motors. Each chip can drive four motors at once, or two motors at once in both forward and reverse. The chip requires 5V for internal logic, as well as a secondary voltage from 4.5 to 36V, for which 12V from the battery was used. The input pins receive signals from digital pins on the Arduino Mega, and the corresponding output pins are connected to the motor to change the direction of motion for the robot.

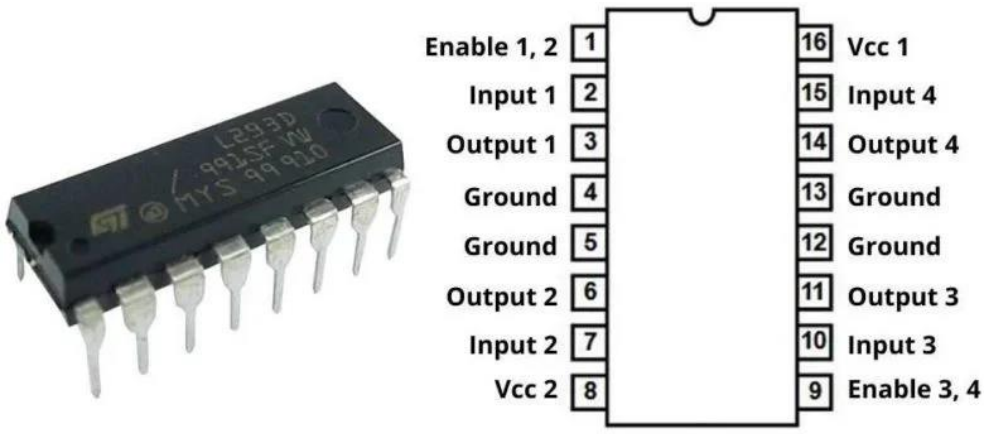


Figure 14. L293D with pinout [6]

3.3.4. HC-SR04 Ultrasonic Sensors

Six HC-SR04 ultrasonic sensors are used for obstacle detection. Since the robot is in most cases programmed to move forward, only going sideways to avoid collisions, these are placed only on the front and sides of the robot (two per side).

Figure 15. Ultrasonic Sensor [7]

3.3.5. TCRT5000 Infrared Reflective Sensor

Infrared reflective (IR) sensors are used in the detection of both the enemy robot's home base and of the enemy robot itself. Both of these entities have an infrared LED beacon that when in range of CR's sensors indicates that the ping pong ball should be fired in order to attempt to secure a hit.

Figure 16. IR Sensor [8]

3.3.6. Micro Servo

A micro servo is used to fire the ping pong ball. When the robot decides to fire, the Arduino sends a signal to turn through the digital pin connected to the servo. This causes the servo to rotate and release the rubber band holding the ball into place.

Figure 17. Servo motor [9]

3.3.7. Battery

A 12V 2200mAh Ni-MH rechargeable battery is used as the robot's main power source. 12V is fed directly to the H-bridges which power the motors. To obtain the lower voltage of 5V required for the sensors and Arduino, a LM7805 voltage regulator is used.

Figure 18. 12V Battery pack [10]

3.3.8. Circuit Wiring

Figure X shows a sample diagram for the wiring, demonstrating how one of each device is wired as well as the driver circuit. The orange power rail represents the 12V from the battery, while the red power rail is the 5V from the voltage regulator. The H-bridges receive both power inputs, and all their enable pins are connected (in green) as the robot moves exclusively in 4-wheel drive. The input pins of each H-bridge receive a high or low signal from digital Arduino pins. In turn, the output pins are connected to the positive and negative terminals of the motor encoder, which also receives 12V. The ultrasonic sensors receive 5V and connect to the digital pins of the Arduino, whereas the IR sensors receive 5V and read out to analog pins. Finally, the micro servo used in firing the ping pong ball receives 5V and is triggered by a digital pin input.

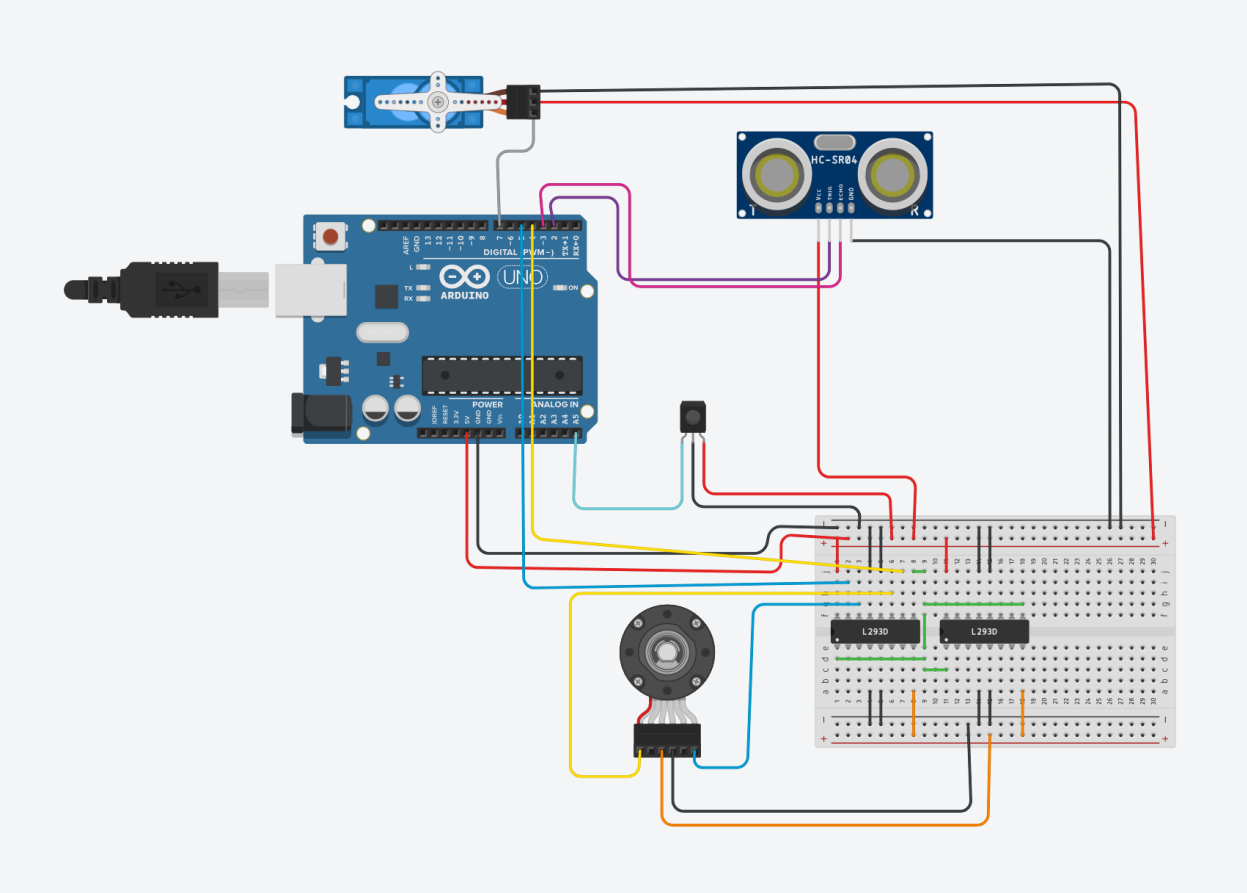


Figure 19. Sample Wiring Schematic. Note that an Arduino Mega, not Uno, was used in the actual robot due to a larger number of pins available.

4. Programming

The robot is controlled using an Arduino Mega 2560, written in C++.

The robot is programmed to only be able to move forward, left, right, and clockwise. This is done using mecanum wheels, which allow the robot to move laterally. The purpose of this is to make the robot easily return to home base without tracking its orientation. Ideally, the robot would always be facing either the opponent's side or its home base, without ever having to rotate to avoid obstacles.

The program is organized using states to account for the conditions of the game. First, the robot must identify either its opponent or its opponent's base while avoiding obstacles. Once the robot has identified a target, it must launch a ping pong ball. Then, the robot must be able to return home while also avoiding obstacles. For the first condition, state 0 was created to allow the robot to traverse around obstacles. State 1 is the behavior of the robot if it is surrounded by obstacles. State 2 occurs when the robot has detected a target. State 3 is when the robot is faced with a wall. Last, states 4 and 5 are to help the robot move side to side when there is a wall in front.

When the program starts, the robot is in state 0. This state is written to help the robot move away from obstacles by moving to the side. For certain cases where there are obstructions on both sides of the robot, it switches to state 1 where the robot will move backwards and to the side until there are no more obstructions. If there is an obstacle in the front of the robot, the robot will assume that the obstacle is a wall, so it will switch to state 3 and keep moving side to side of the wall (states 4 and 5 represent moving one direction until there is an obstacle, and then switching between left and right). The purpose of this is to allow the robot to catch its opponents base using the IR sensors if it has been missed from the first passing. Once the robot identifies the target, it will shoot a ping pong ball and immediately switch to state 2. In state 2, the robot turns 180 degrees and returns to state 0. Now, the robot should be facing its home base, and it will traverse around obstacles in state 0, switching to states 1, 3, 4, and 5 if necessary, until it reaches home. A flow chart of the process can be seen in figure 20 below.

During all of these states, the Arduino was reading information from the ultrasonic and IR sensors to detect any obstacles or targets. There was some calibration required for the IR and ultrasonic sensors to ensure that the robot could identify its targets and effectively avoid obstacles. To do this, testing codes were run to read the values of the sensors as obstacles/ IR LEDs were placed around the robot. These codes can be found in Appendix D.

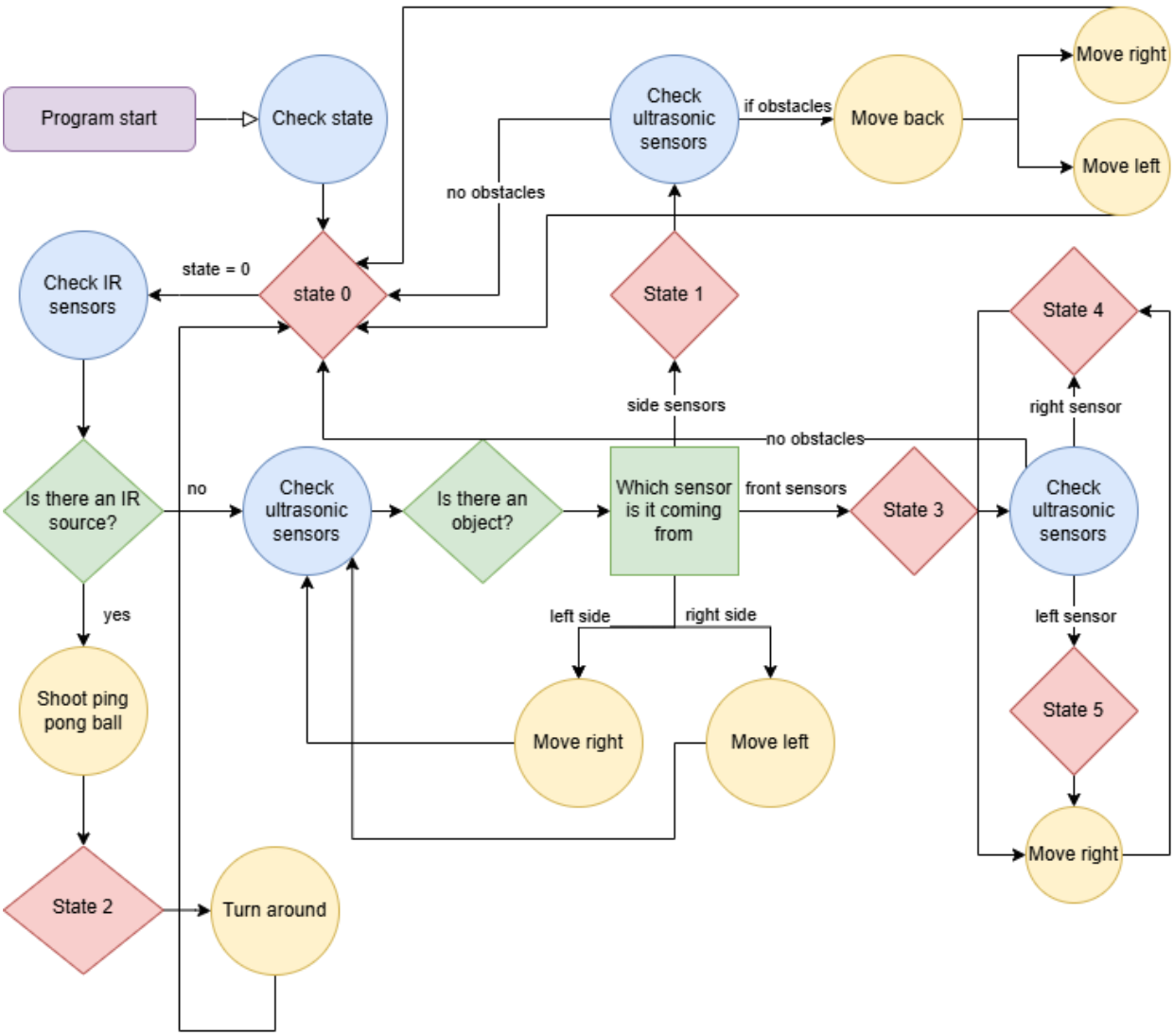


Figure 20. Flow chart of the program

5. Conclusion

CR performed well in the beginning of the competition but through repeated use, the robot suffered from multiple issues that caused it to lose all its matches. In its first match, CR was able to obtain two achievements: shoot the enemy base and return to home base. While at first, CR struggled to leave its obstacle avoidance state when approaching an obstacle that triggered both front ultrasonic sensors, a quick fix to the programming allowed CR to navigate to the enemy base. Once CR identified that it had reached the enemy wall, it strafed left and right thanks to its

mecanum wheels to find the enemy beacon. This is where the issue with the robot turning arose as CR would eventually rotate towards the left as it tried to shuffle left and right. As mentioned before, this was likely due to the front motors and rear motors being different and the loose fitting of the front motor hubs. However, with a few collisions with the enemy wall and side walls, CR was able to find the enemy beacon and successfully shot into the base.

On the return home, CR struggled to effectively avoid the obstacles. It was able to drive over the thinner ground obstacle easily on the way to the enemy base and on the way back. However, since the ultrasonic sensors were placed on the sides of the robot, obstacles sitting diagonal to CR would bypass the ultrasonic sensor range of detection. Thus, the corner of the robot would clip the edge of the obstacle, requiring emergency teleports because CR would spin around the corner of the obstacle and be effectively spun out. This was an issue that carried over to the other rounds as well. If more sensors were added to the corners of CR, this issue would have been mitigated and obstacle avoidance would have been more effective.

Still in the first round, CR had not yet obtained the return home achievement. When it did, it had navigated halfway through the arena and mistakenly shot its cannon. In later matches, CR would do this a lot where if it collided with the side wall head on, the IR sensors would cause the cannon to shoot the ping pong ball. However, even though CR shot its ball at the wall, it was able to turn around and navigate to home base, obtaining the return home achievement. Unfortunately, this is where CR's achievements ended, and its performance started to suffer. It would keep veering right when trying to move in a straight line. In the first match it only scored 10 points but ended with a negative point balance because of the emergency teleports.

Towards the end of the competition, CR struggled with navigating the arena when there were 5 obstacles present. Often, it would get stuck between the ground obstacle and a regular

obstacle. The regular obstacle triggered the robot to move away from it but this caused it to approach the thicker ground obstacle. When strafing left and right, CR failed to climb the thicker ground obstacle. As the last round proceeded, CR's left motor also slowed down much more than the other motors, indicating that one of the pins on the H-bridge may have been burning out. Lastly, CR could have been more successful in the beginning with earning points but the tight space inside the robot made it difficult to load the rubber band and ping pong ball, increasing the amount of time required to load the ball. The IR sensors were also too low to detect enemy robot beacons.

Overall, CR failed to obtain the remaining achievements and score points. Although it came last in the competition, it shared many similarities with the previous year's champion, such as mecanum wheels and a rectangular frame design. CR has great potential but with a few more ultrasonic sensors, programming tweaks, and stronger motors, it could have been a more formidable robot in the competition.

6. Appendices

Appendix A: CAD Drawings

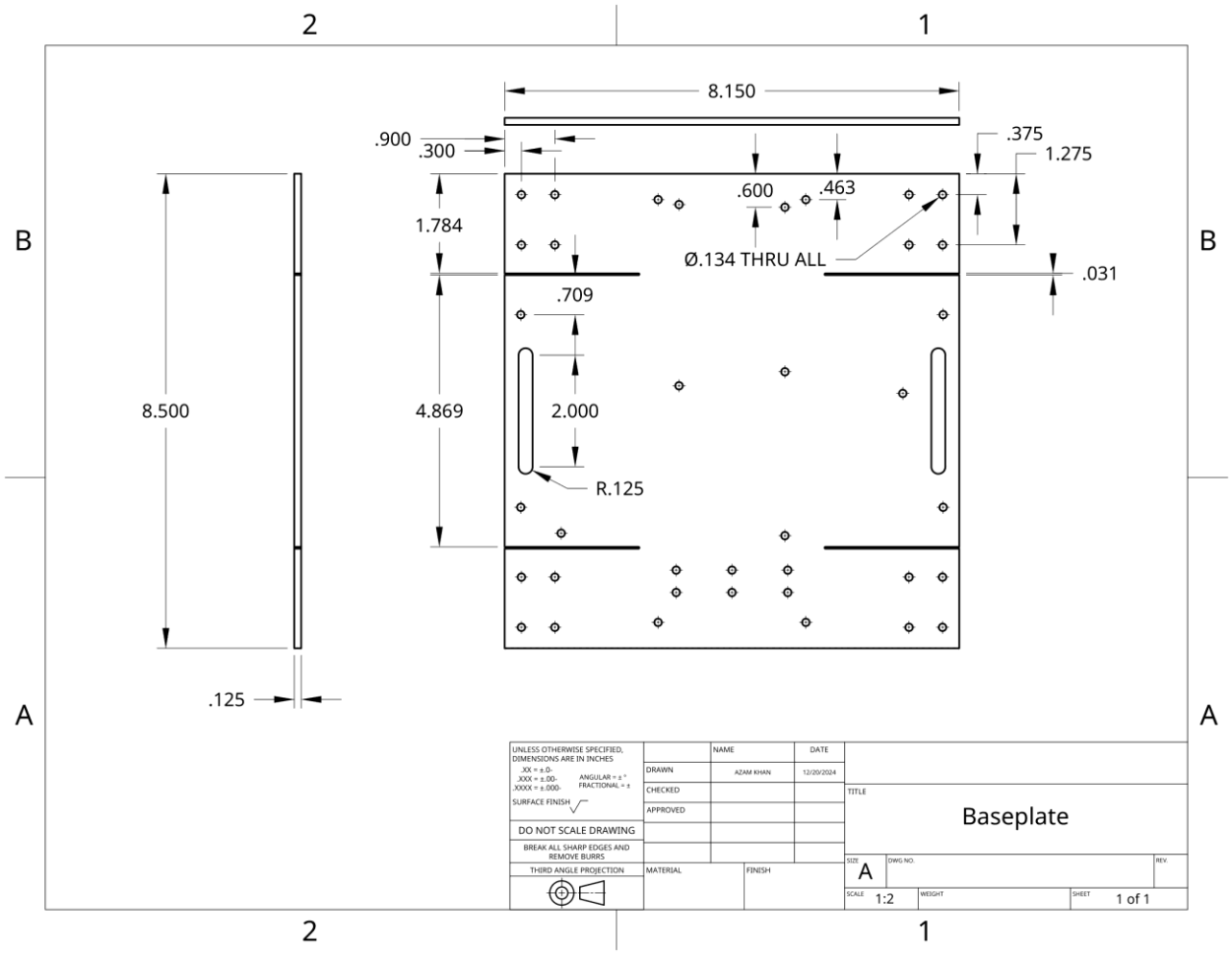


Figure A1. Baseplate Drawing

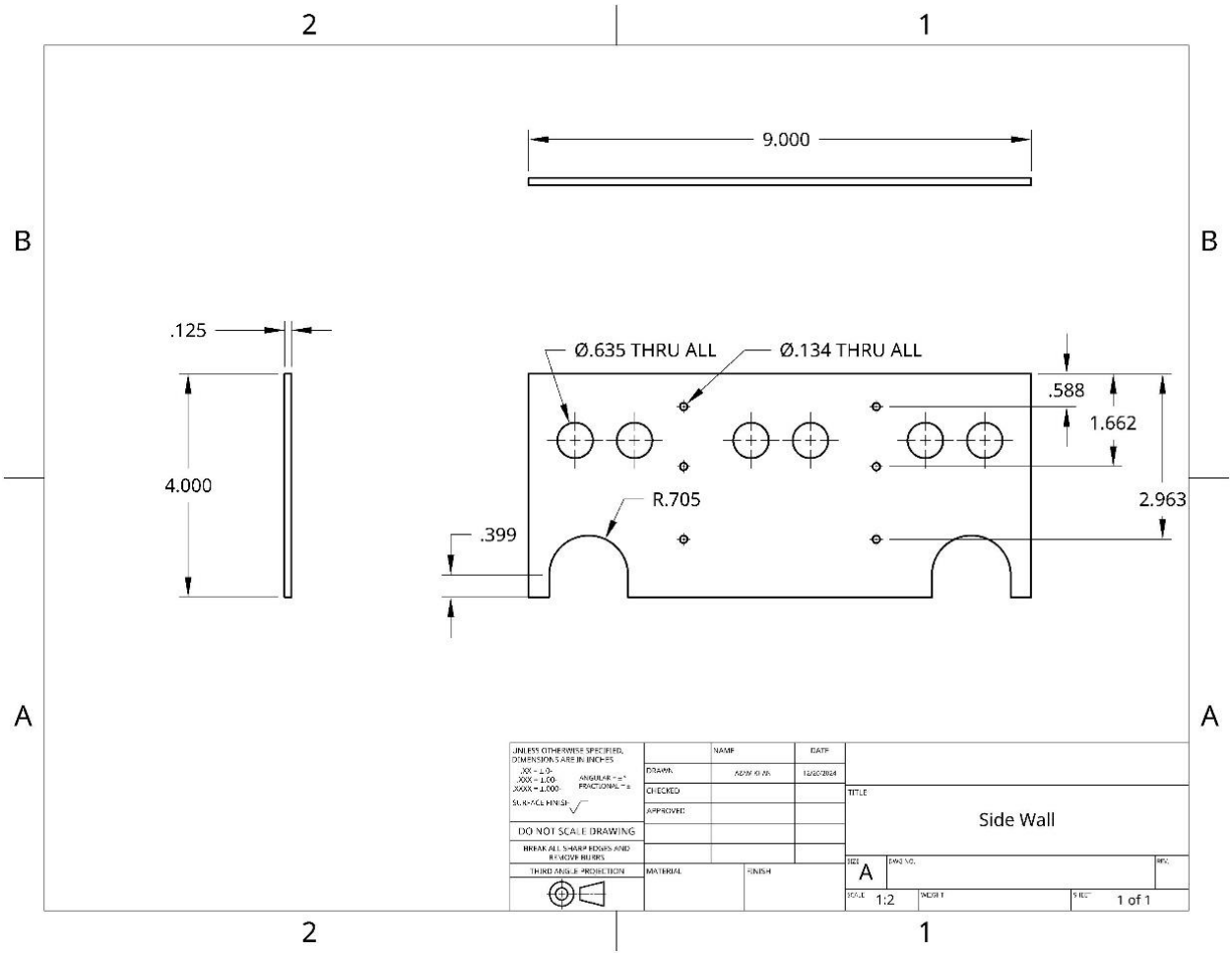


Figure A2. Side Wall Drawing

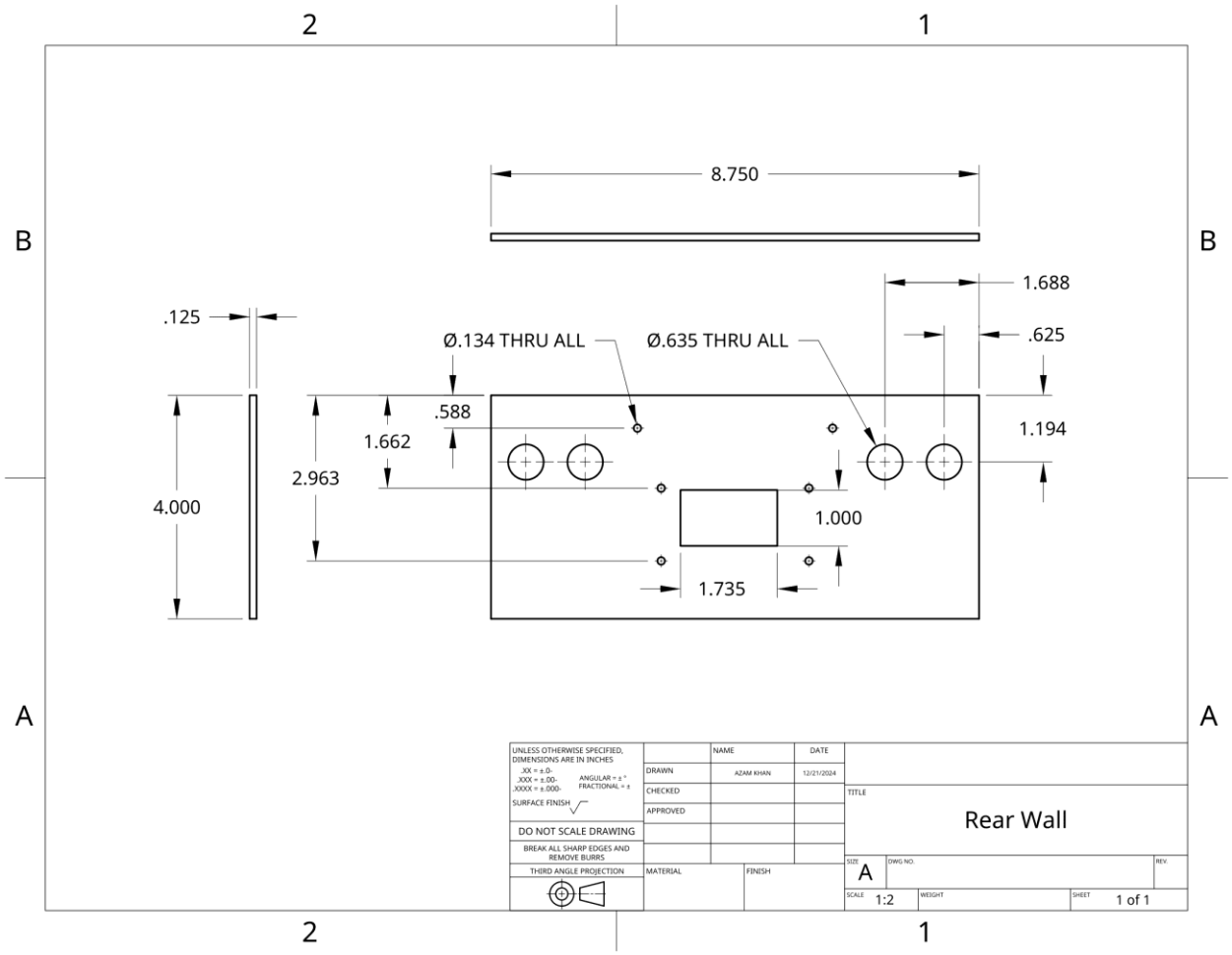


Figure A3. Rear Wall Drawing

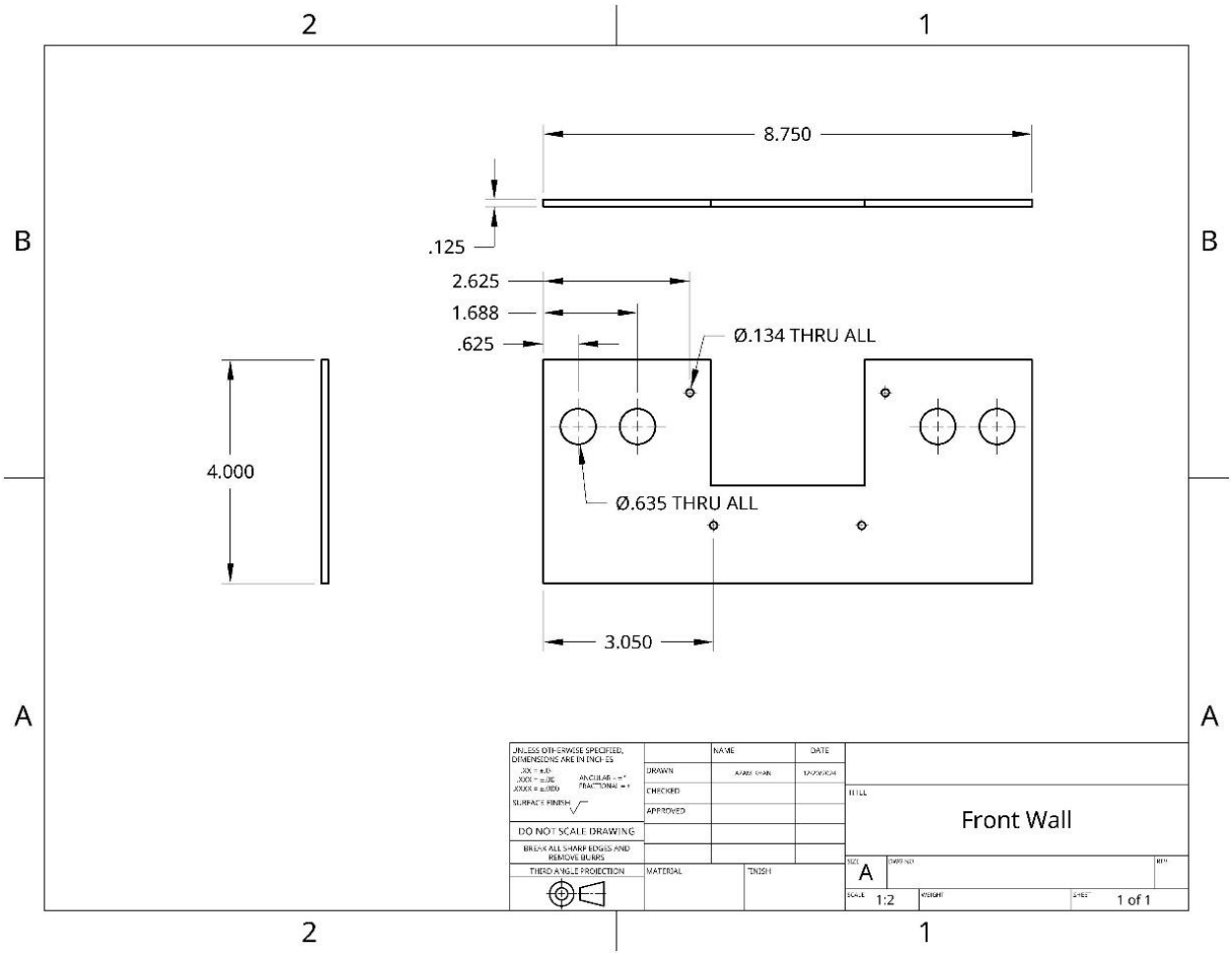


Figure A4. Front Wall Drawing

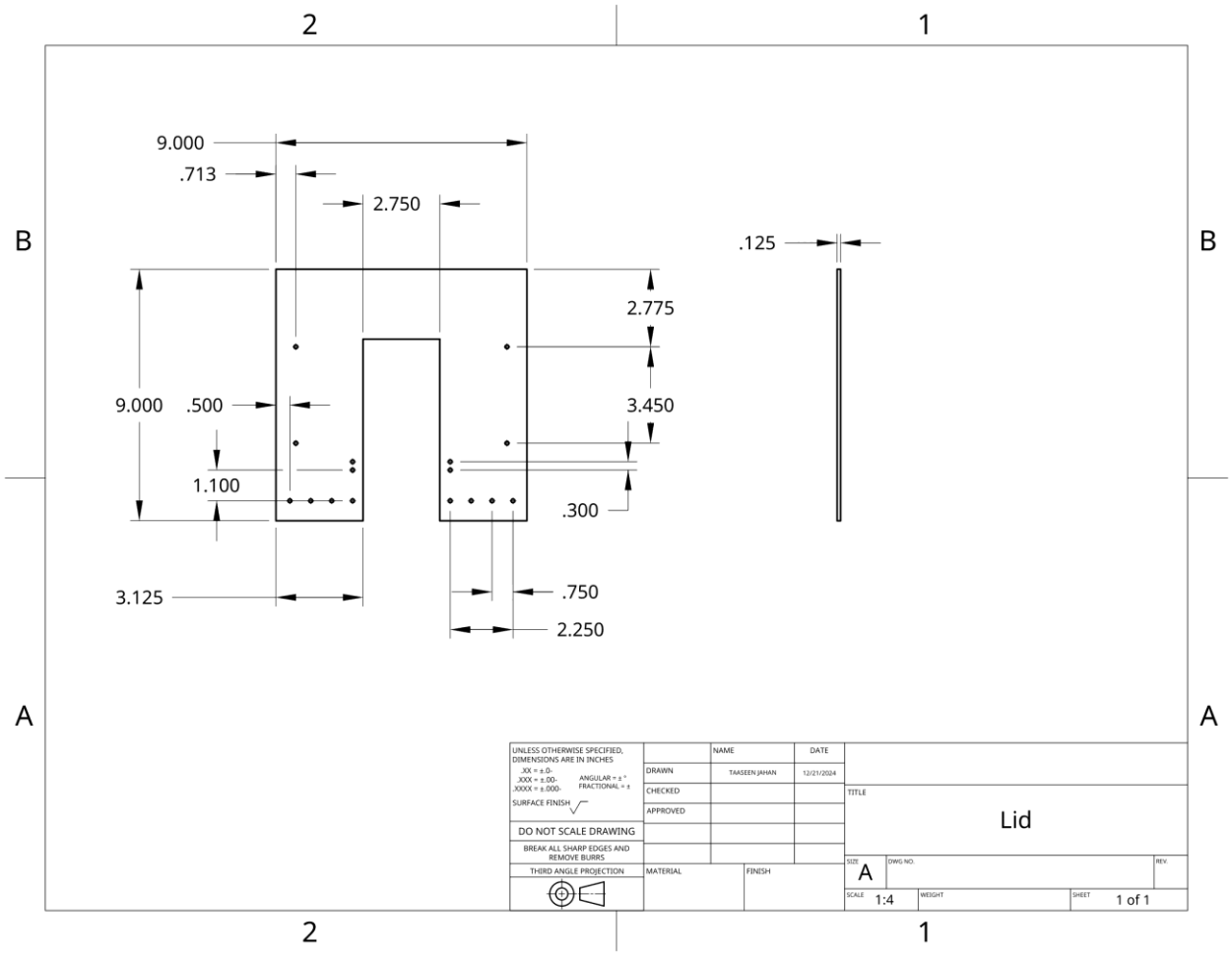


Figure A5. Lid Drawing

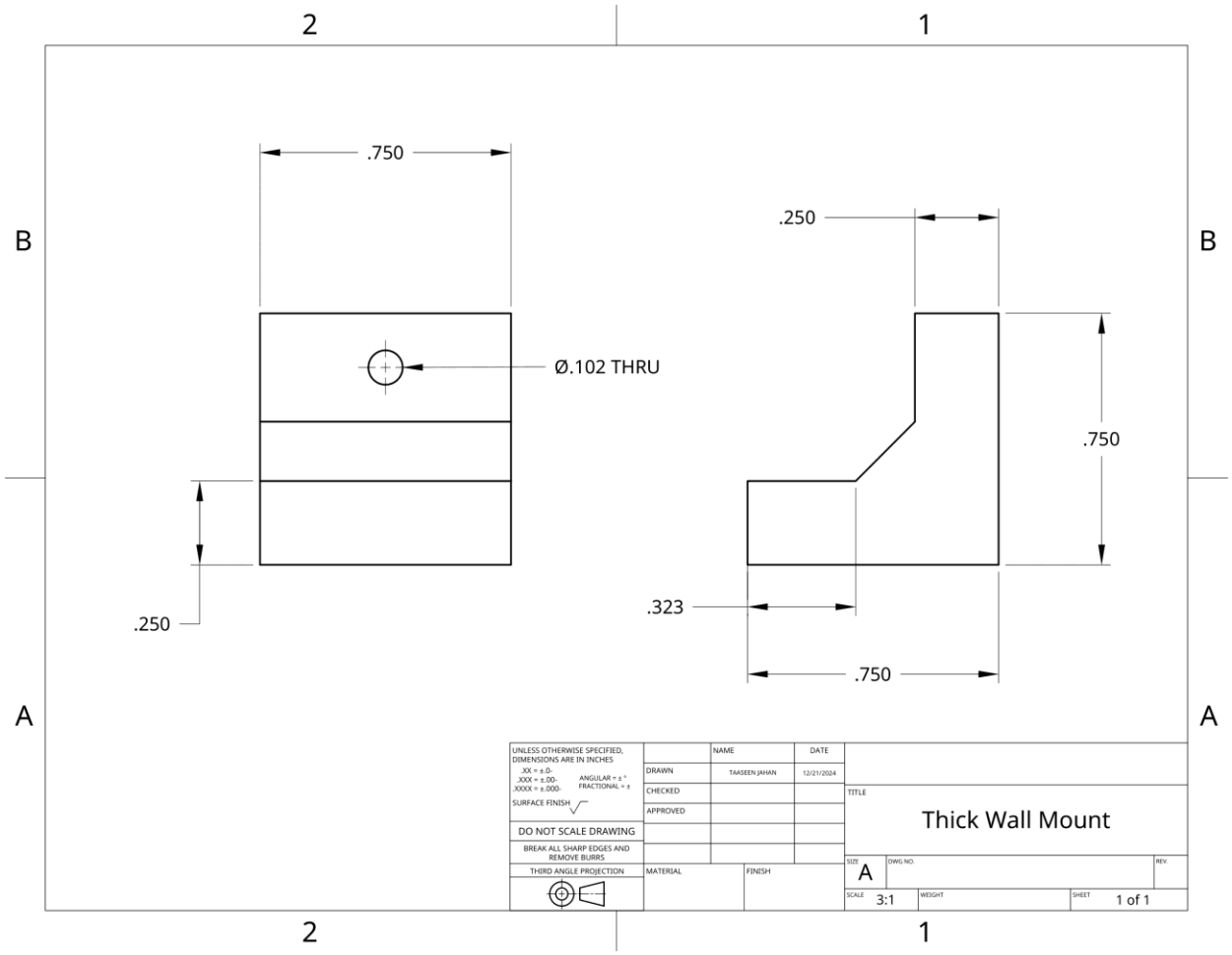


Figure A6. Thick Wall Mount Drawing

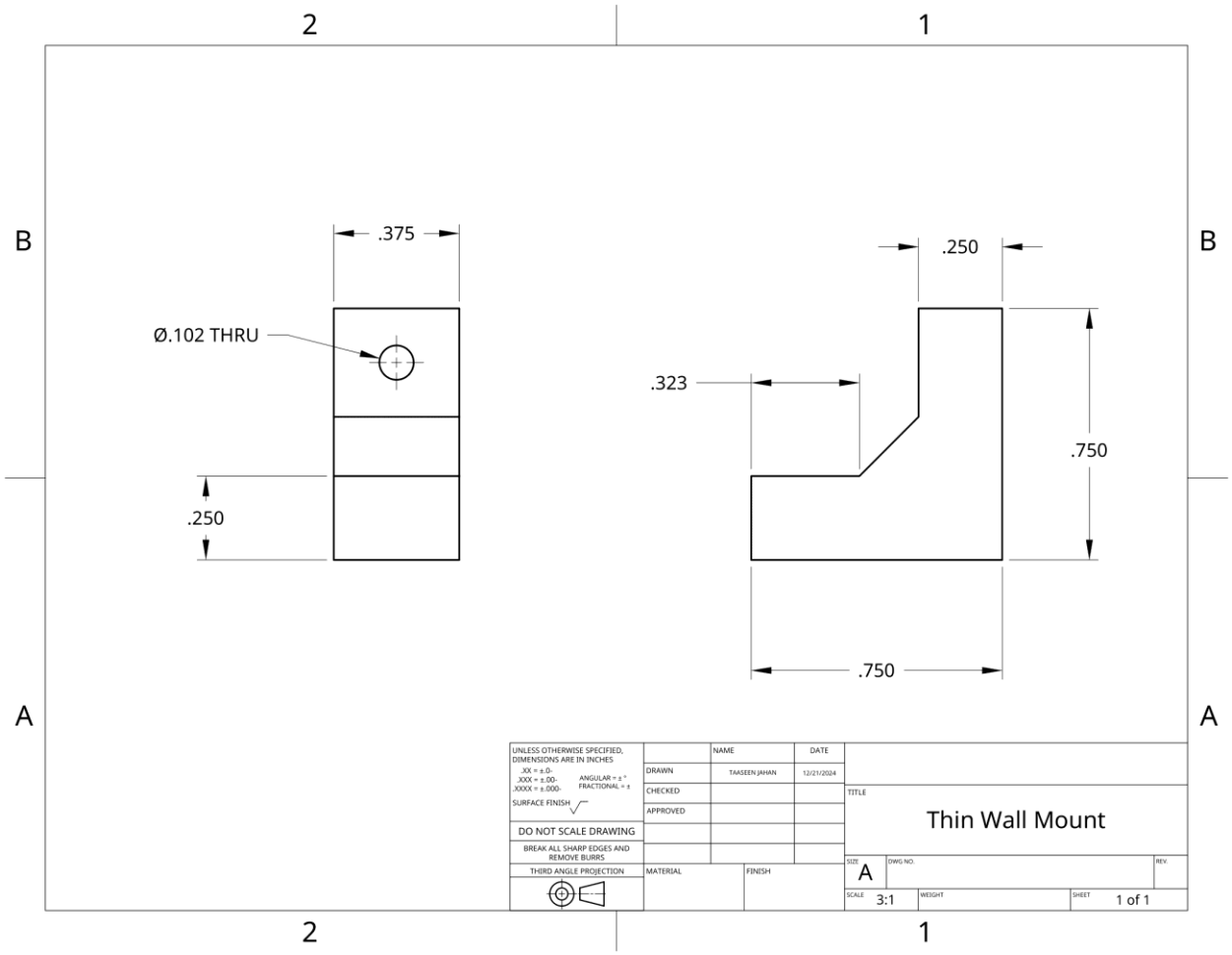


Figure A7. Thin Wall Mount Drawing

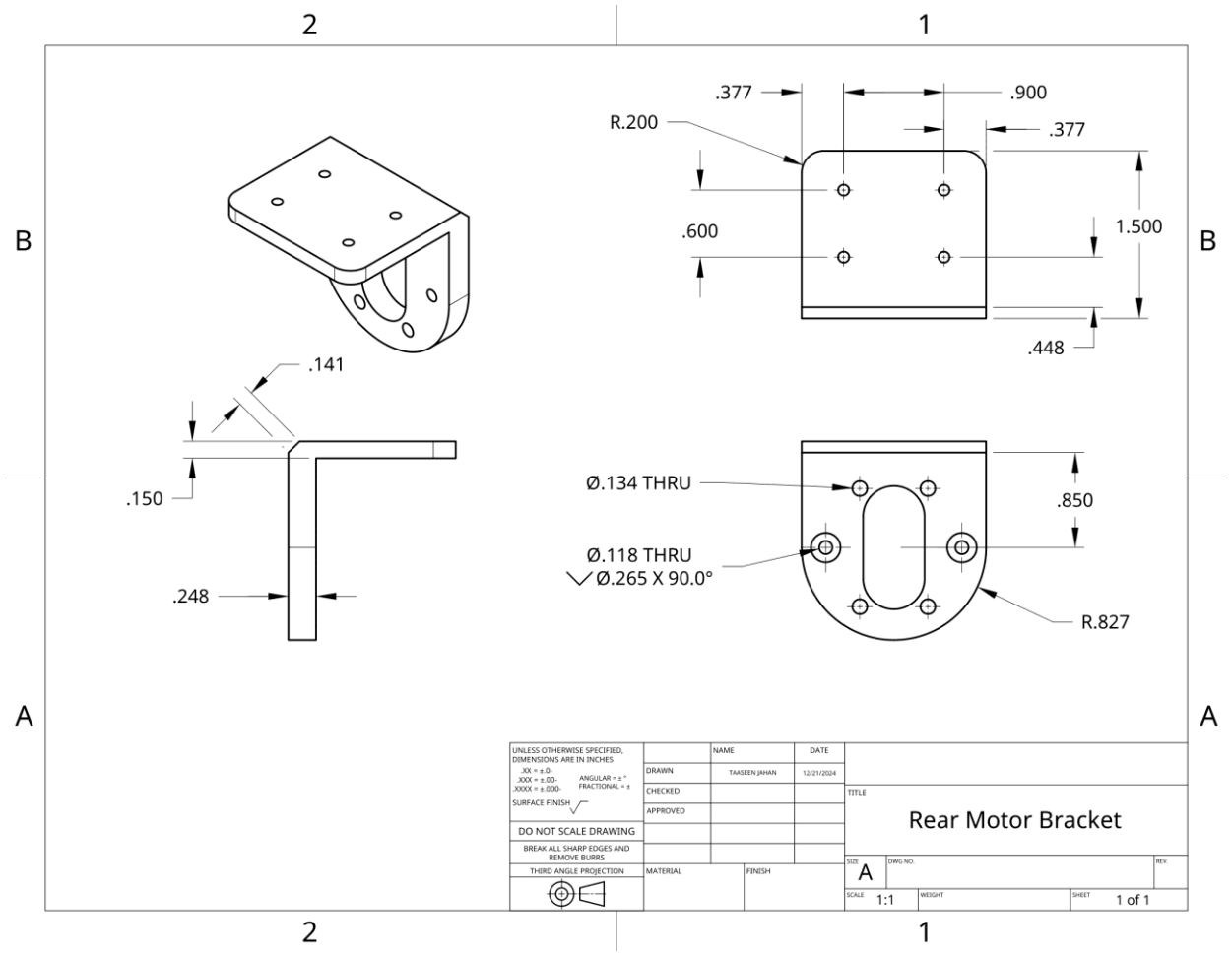


Figure A8. Rear Motor Bracket

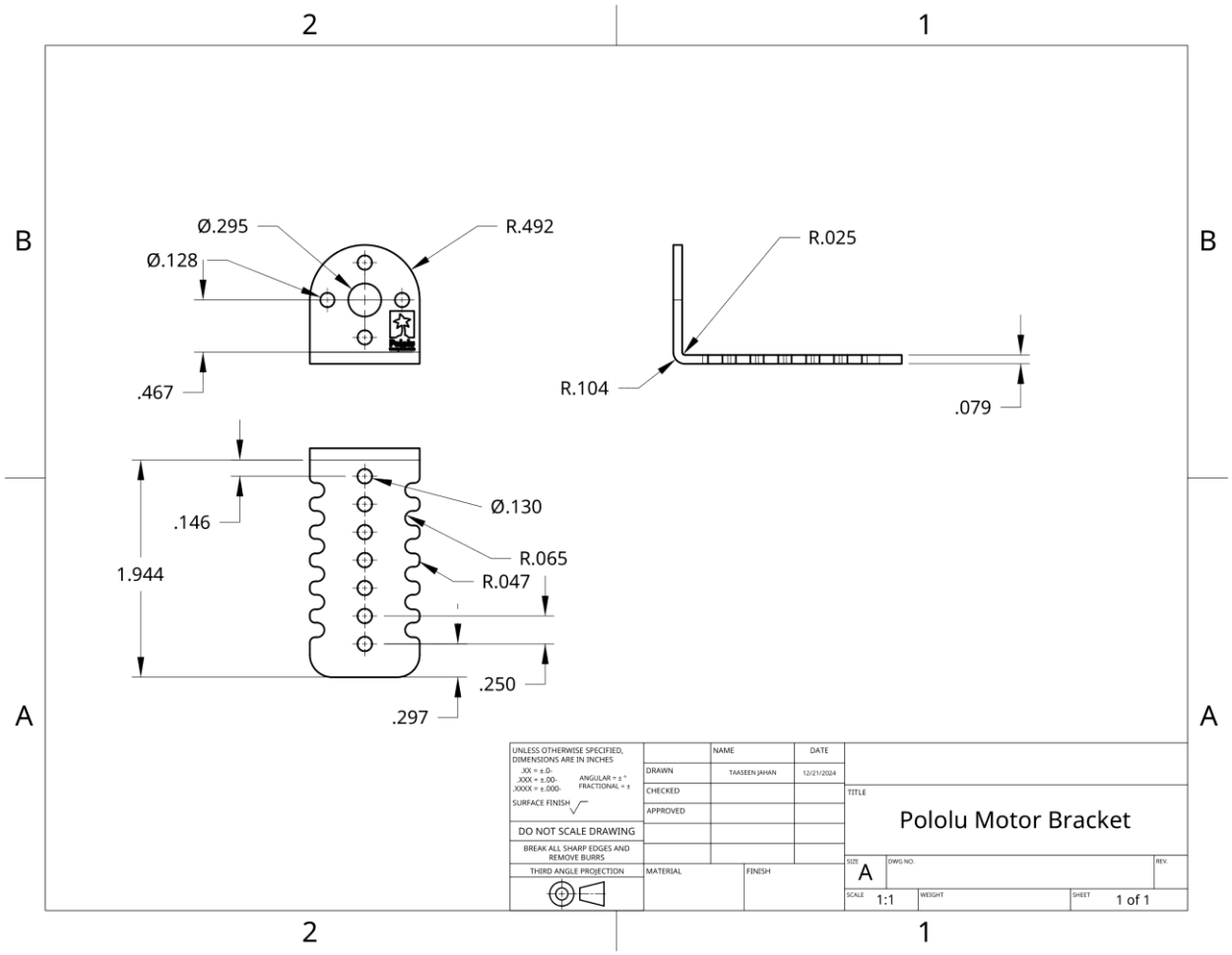


Figure A9. Pololu Motor Bracket for Front Motors

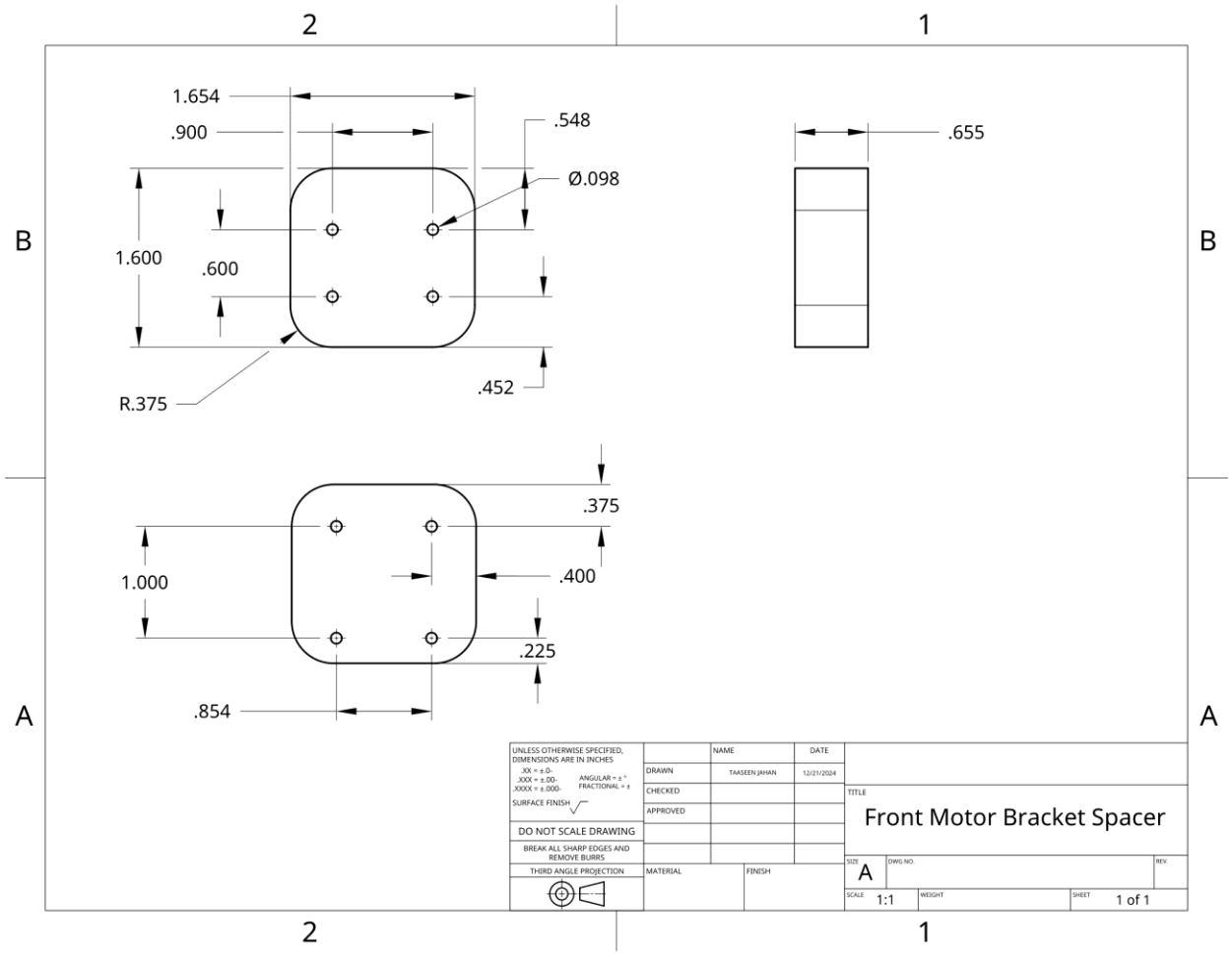


Figure A10. Front Motor Spacer to Keep Robot Level

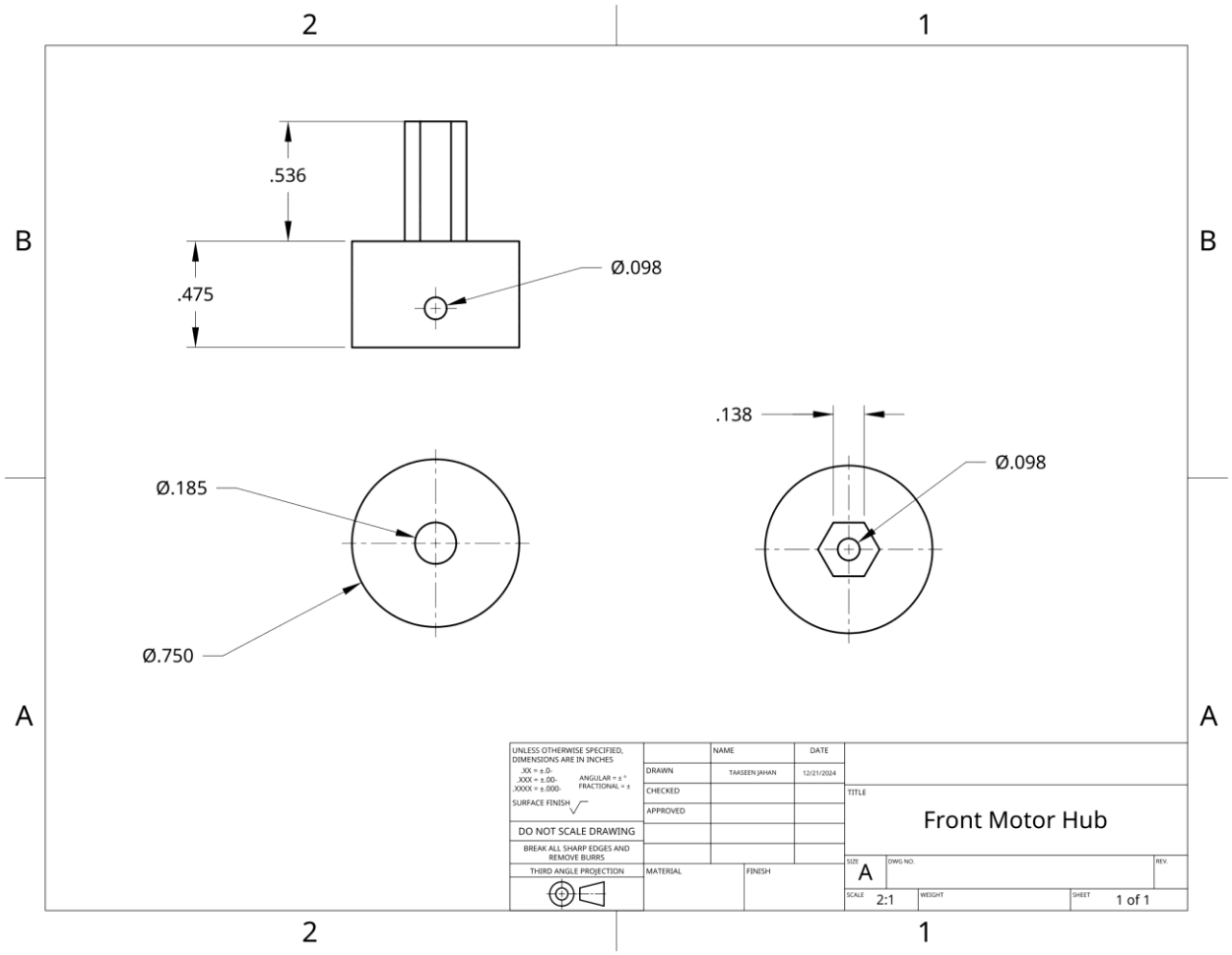


Figure A11. Front Motor Hub Drawing

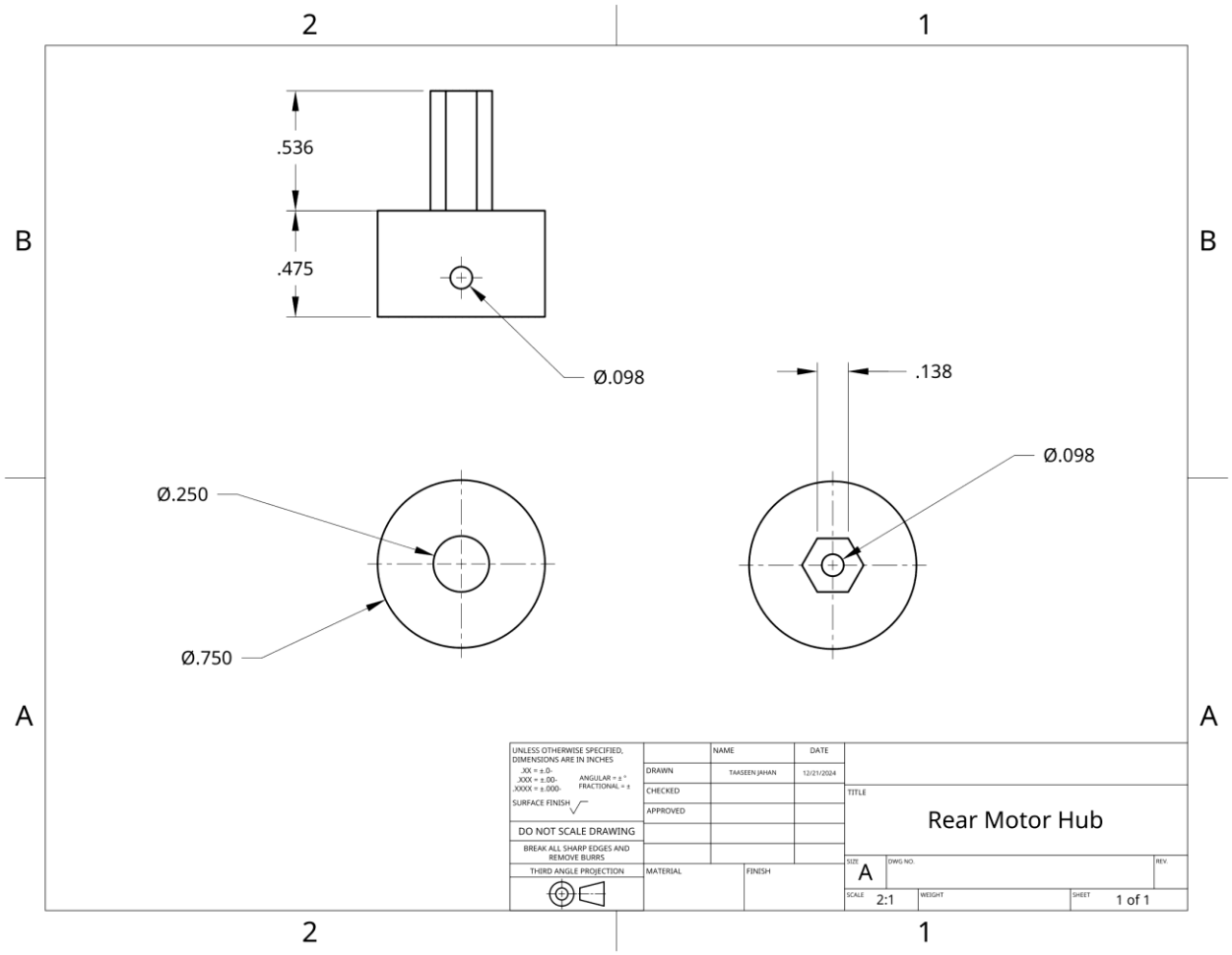


Figure A12. Rear Motor Hub Drawing

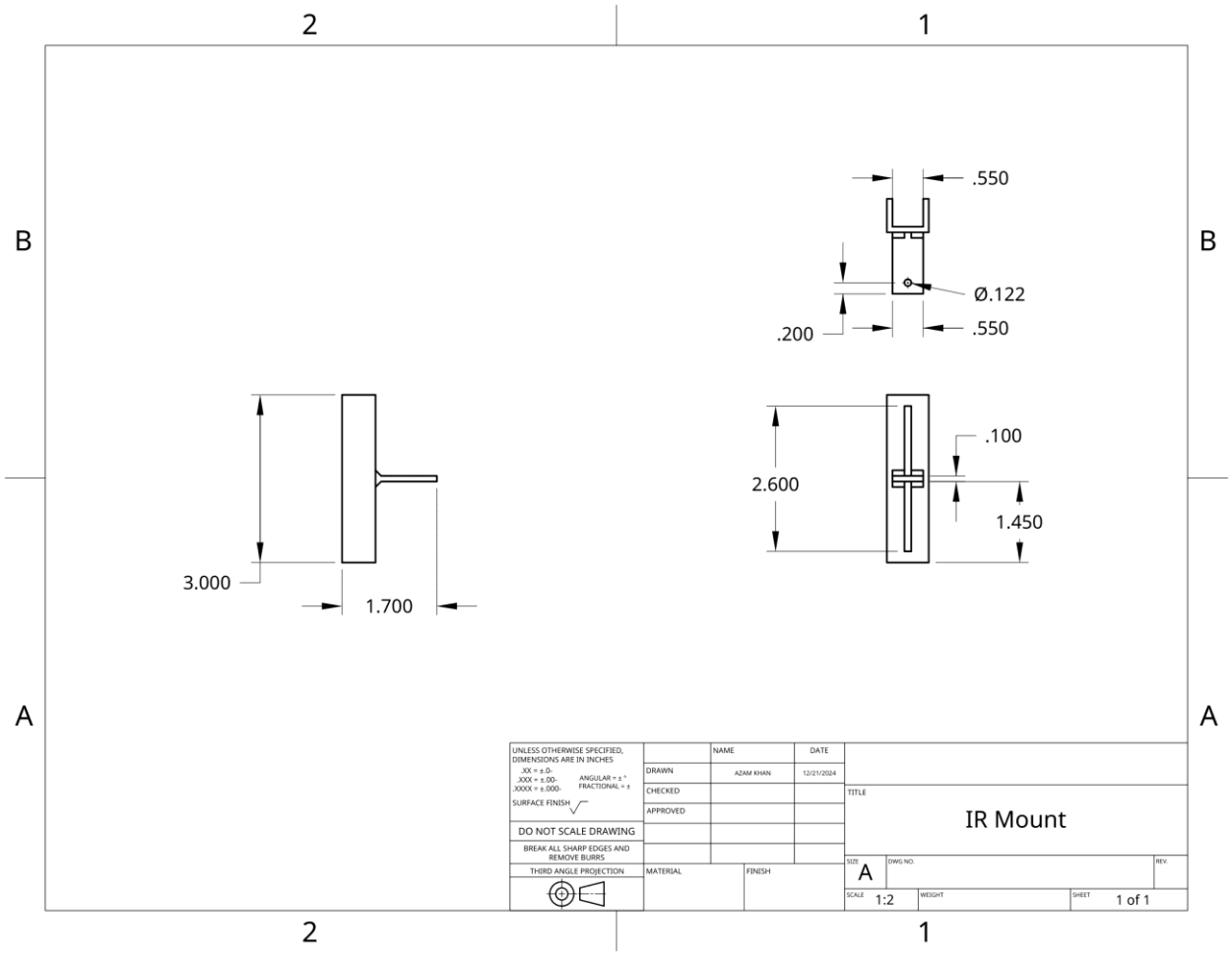


Figure A13. IR Mount Drawing

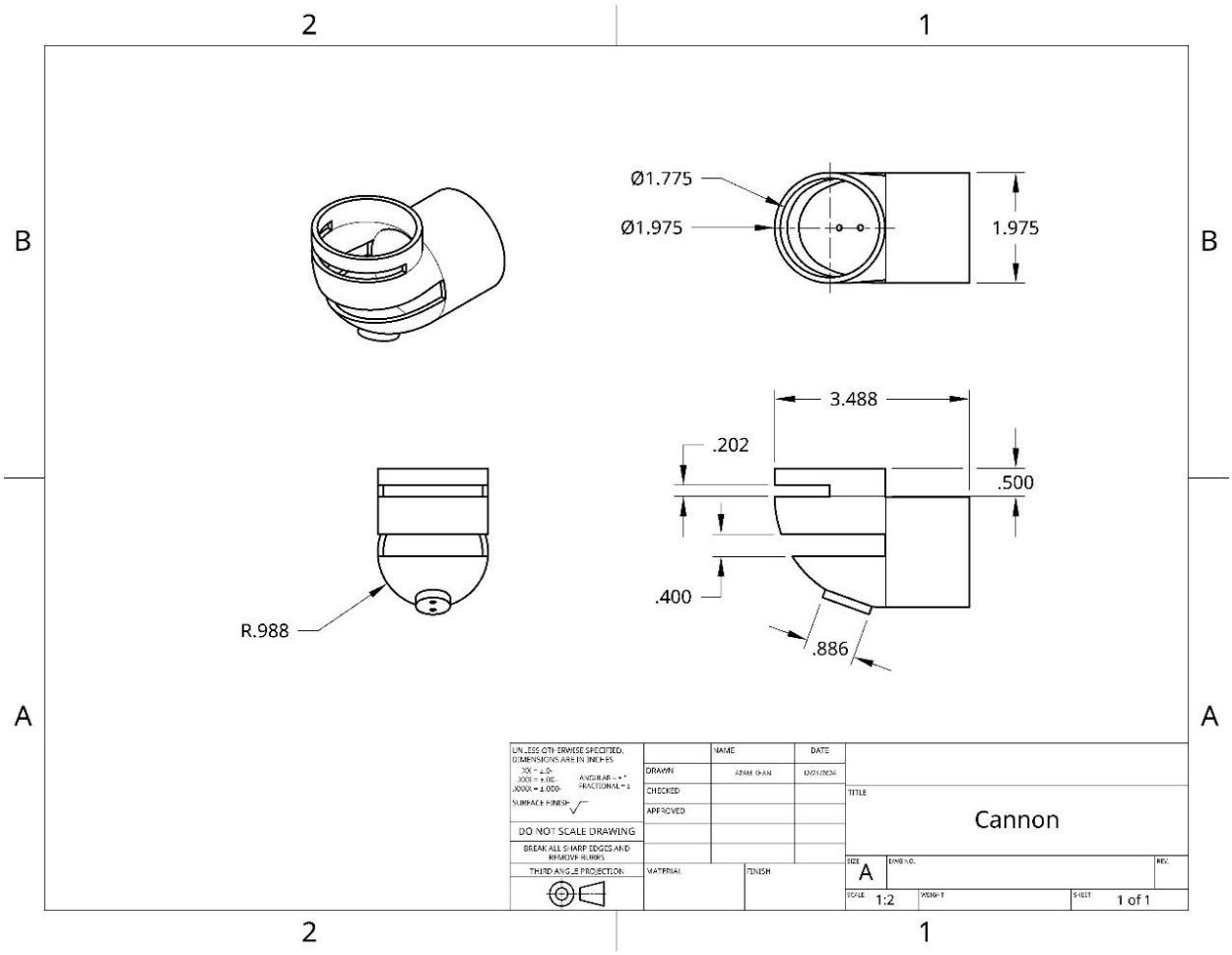


Figure A14. Cannon Drawing

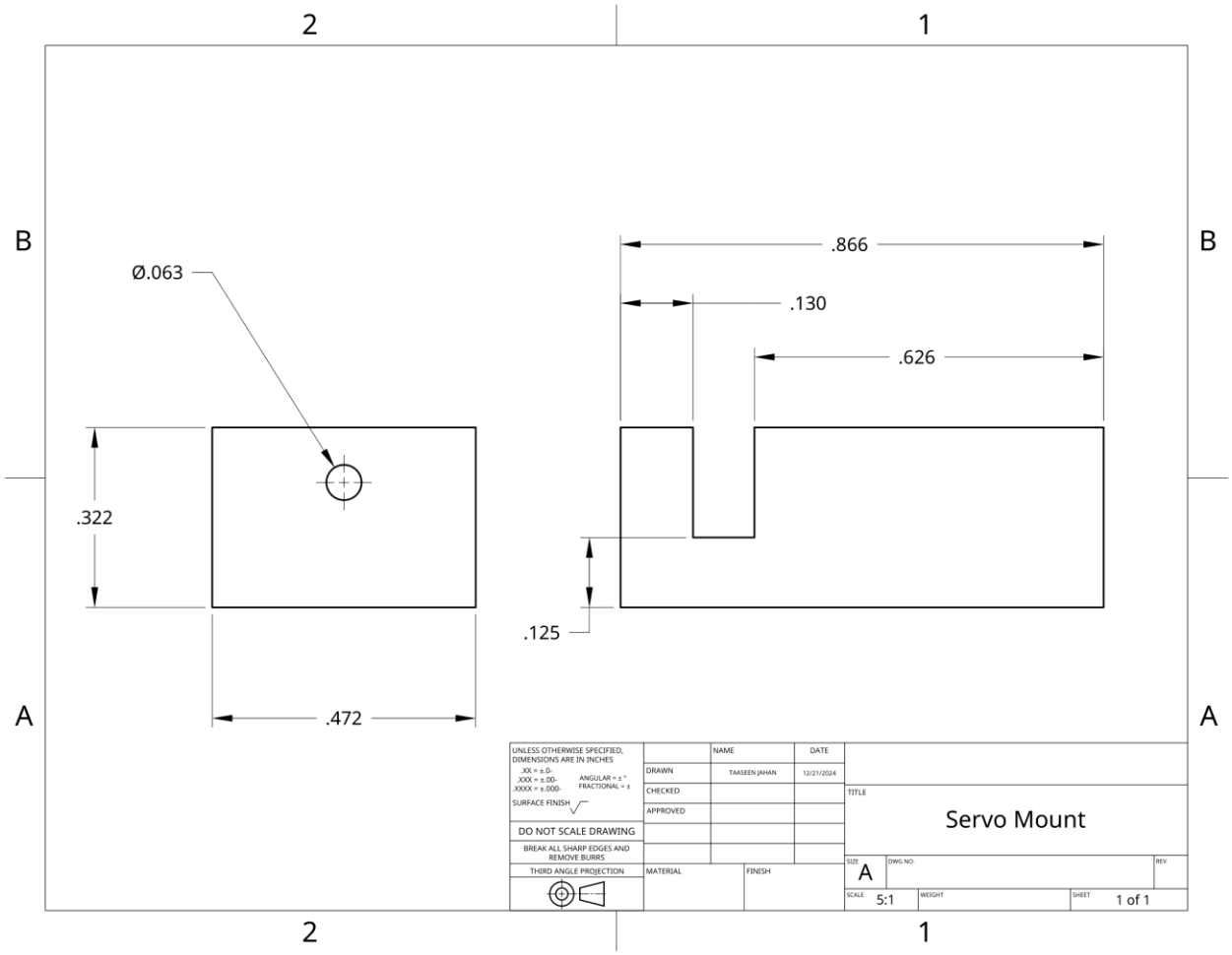


Figure A15. Servo Mount

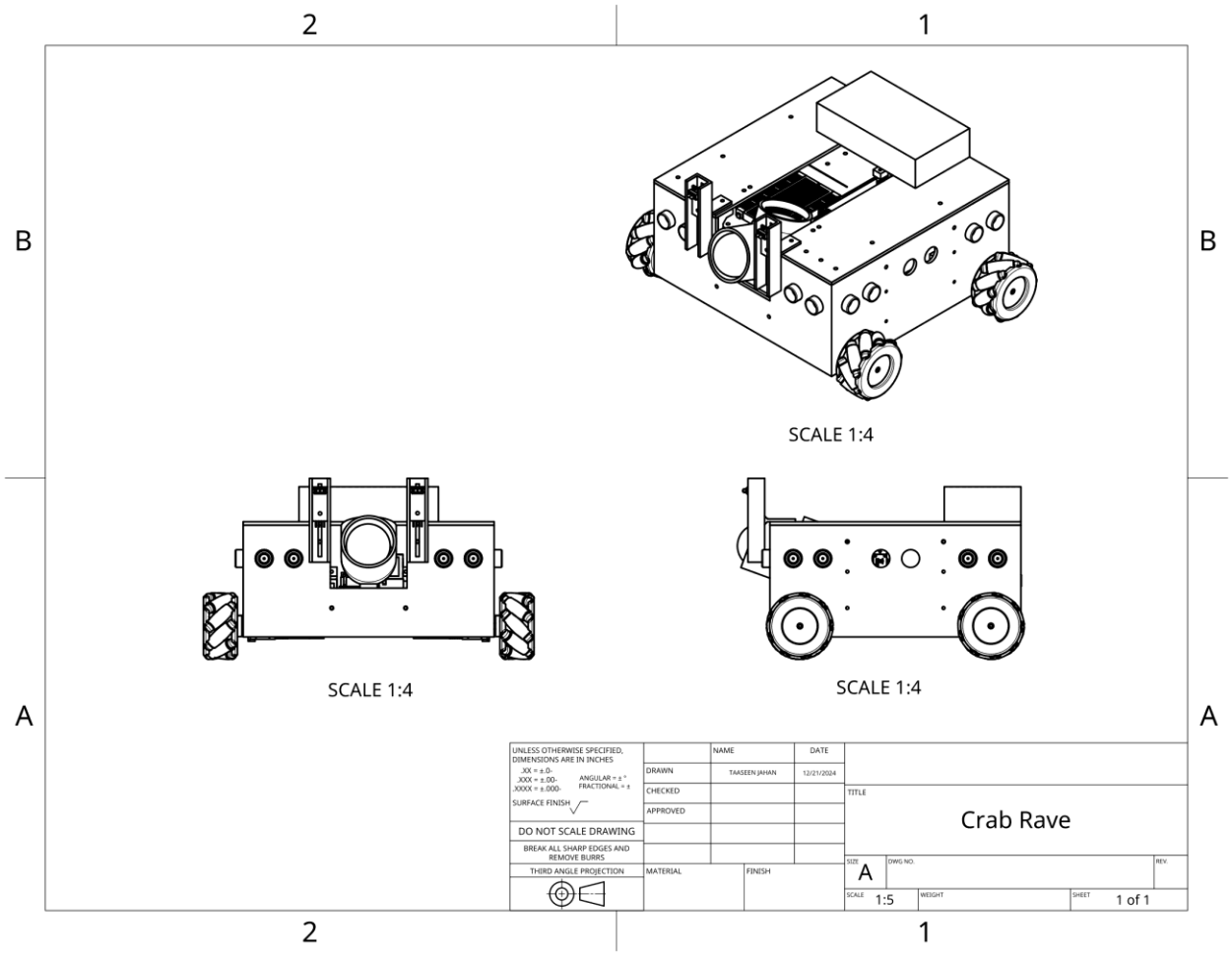


Figure A16. Assembly Drawing of Crab Rave

Appendix B: Datasheet and Specifications of parts

Arduino mega (pp. 45-57)

IR sensor (pp. 58-63)

Ultrasonic sensor (pp. 64-6)

Front motors (pp. 67)

Back motors (pp. 68-69)

Motor driver chip (pp. 70-75)

Servo motor (pp. 76ss)

Arduino mega

Product Reference Manual
SKU: A000067

Description

Arduino® Mega 2560 Rev3 is an exemplary development board dedicated for building extensive applications as compared to other maker boards by Arduino. The board accommodates the ATmega2560 microcontroller, which operates at a frequency of 16 MHz. The board contains 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a USB connection, a power jack, an ICSP header, and a reset button.

Target Areas

3D Printing, Robotics, Maker

Features

- **ATmega2560 Processor**
  - Up to 16 MIPS Throughput at 16MHz
  - 256k bytes (of which 8k is used for the bootloader)
  - 4k bytes EEPROM
  - 8k bytes Internal SRAM
  - 32 × 8 General Purpose Working Registers
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Four Programmable Serial USART
  - Controller/Peripheral SPI Serial Interface
- **ATmega16U2**
  - Up to 16 MIPS Throughput at 16 MHz
  - 16k bytes ISP Flash Memory
  - 512 bytes EEPROM
  - 512 bytes SRAM
  - USART with SPI master only mode and hardware flow control (RTS/CTS)
  - Master/Slave SPI Serial Interface
- **Sleep Modes**
  - Idle
  - ADC Noise Reduction
  - Power-save
  - Power-down
  - Standby
  - Extended Standby
- **Power**
  - USB Connection
  - External AC/DC Adapter
- **I/O**
  - 54 Digital
  - 16 Analog
  - 15 PWM Output

1 The Board

Mega 2560 Rev3 is a successor board of Arduino Mega, it is dedicated to applications and projects that require large number of input output pins and the use cases which need high processing power. The Mega 2560 Rev3 comes with a much larger set of IOs when we compare it with the traditional Arduino® UNO board considering the form factor of both the boards.

1.1 Application Examples

- **Robotics**: Featuring the high processing capacity, the Mega 2560 Rev3 can handle the extensive robotic applications. It is compatible with the motor controller shield that enables it to control multiple motors at an instance, thus making it perfect of robotic applications. The large number of I/O pins can accommodate many robotic sensors as well.
- **3D Printing**: Algorithms play a significant role in implementation of 3D printers. Mega 2560 Rev3 has the power to process these complex algorithms required for 3D printing. Additionally, the slight changes to the code is easily possible with the Arduino IDE and thus 3D printing programs can be customized according to user requirements.
- **Wi-Fi**: Integrating wireless functionality enhances the utility of the applications. Mega 2560 Rev3 is compatible with Wi-Fi® shields hence allowing the wireless features for the applications in 3D printing and Robotics.

1.2 Accessories

1.3 Related Products

- Arduino® UNO R3
- Arduino® Nano
- Arduino® Due without headers

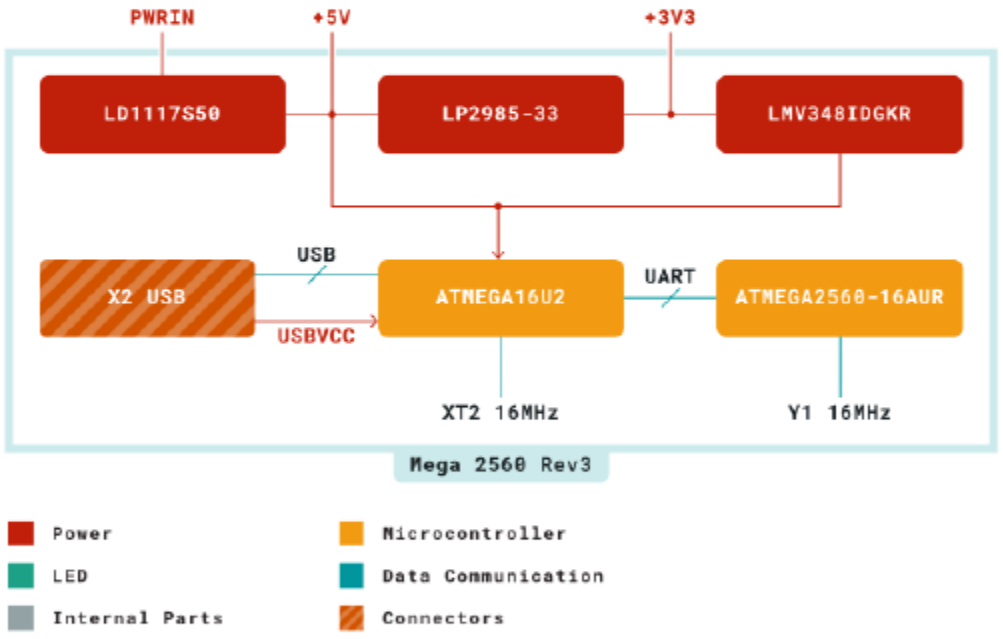
2 Ratings

2.1 Recommended Operating Conditions

Symbol	Description	Min	Typ	Max	Unit
V_{IN}	Input voltage from VIN pad / DC Jack	7	7.0	12	V
V_{USB}	Input voltage from USB connector	4.8	5.0	5.5	V
T_{OP}	Operating Temperature	-40	25	85	°C

3 Functional Overview

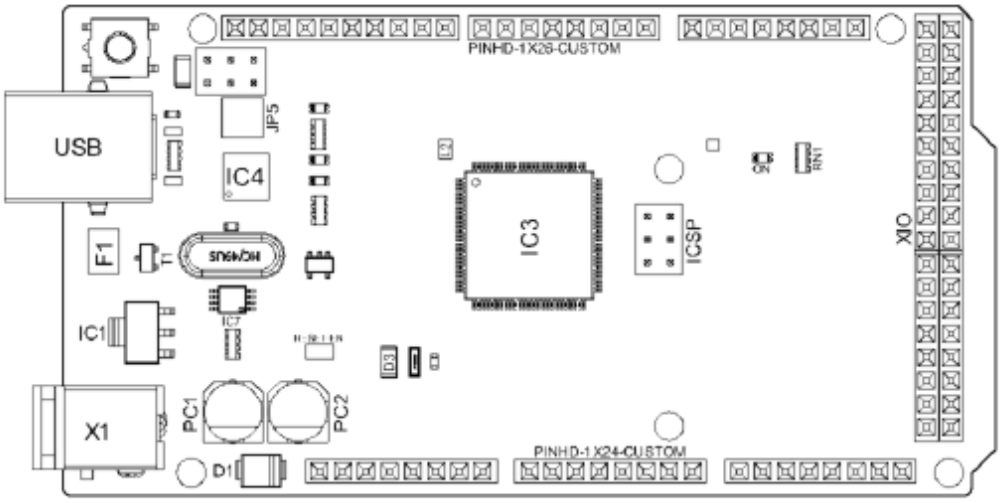
3.1 Block Diagram



Arduino Mega 2560 Rev3 Block Diagram

3.2 Board Topology

Front View



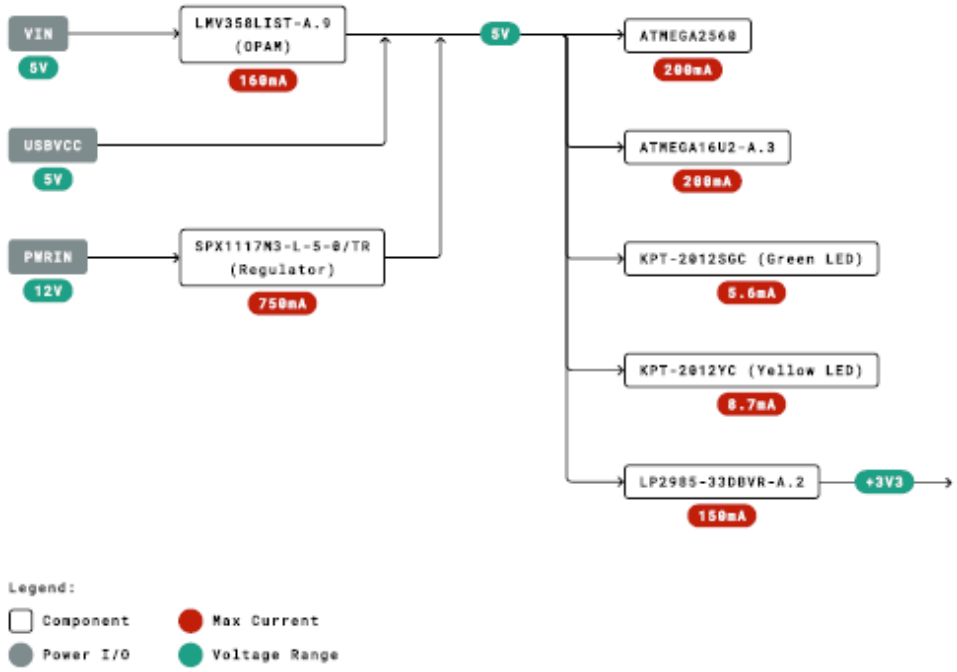
Arduino Mega 2560 Rev3 Top View

Ref.	Description	Ref.	Description
USB	USB B Connector	F1	Chip Capacitor
IC1	5V Linear Regulator	X1	Power Jack Connector
JP5	Plated Holes	IC4	ATmega16U2 chip
PC1	Electrolytic Alumninum Capacitor	PC2	Electrolytic Alumninum Capacitor
D1	General Purpose Rectifier	D3	General Purpose Diode
L2	Fixed Inductor	IC3	ATmega2560 chip
ICSP	Connector Header	ON	Green LED
RN1	Resistor Array	XIO	Connector

3.3 Processor

Primary processor of Mega 2560 Rev3 board is ATmega2560 chip which operates at a frequency of 16 MHz. It accommodates a large number of input and output lines which gives the provision of interfacing many external devices. At the same time the operations and processing is not slowed due to its significantly larger RAM than the other processors. The board also features a USB serial processor ATmega16U2 which acts an interface between the USB input signals and the main processor. This increases the flexibility of interfacing and connecting peripherals to the Mega 2560 Rev3 board.

3.4 Power Tree



Power Tree

4 Board Operation

4.1 Getting Started - IDE

If you want to program your Mega 2560 Rev3 while offline you need to install the Arduino Desktop IDE **[1]** To connect the Mega 2560 Rev3 to your computer, you'll need a Type-B USB cable. This also provides power to the board, as indicated by the LED.

4.2 Getting Started - Arduino Cloud Editor

All Arduino boards, including this one, work out-of-the-box on the Arduino Cloud Editor **[2]**, by just installing a simple plugin.

The Arduino Cloud Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow **[3]** to start coding on the browser and upload your sketches onto your board.

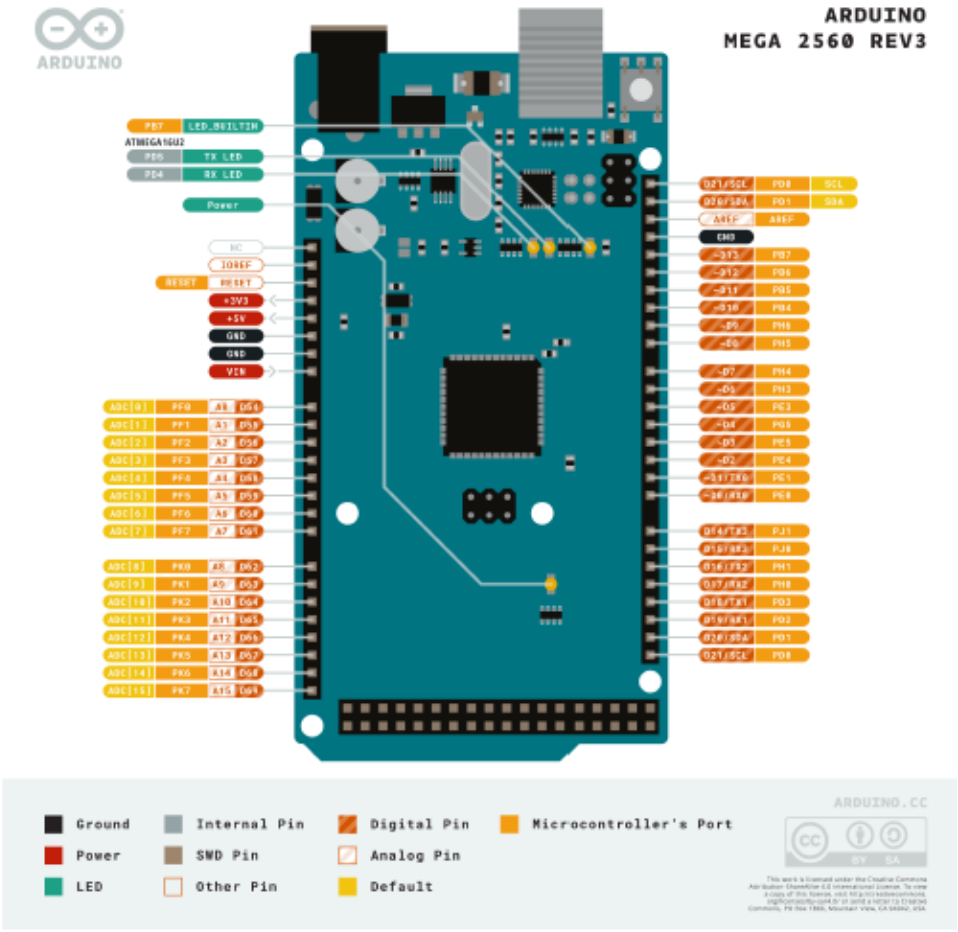
4.3 Sample Sketches

Sample sketches for the Mega 2560 Rev3 can be found either in the "Examples" menu in the Arduino IDE or under the "Documentation" menu on the Arduino website [4].

4.4 Online Resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on Arduino Project Hub **[5]**, the Arduino Library Reference **[6]** and the online store **[7]** where you will be able to complement your board with sensors, actuators and more.

5 Connector Pinouts



Arduino Mega 2560 Rev3 Pinout

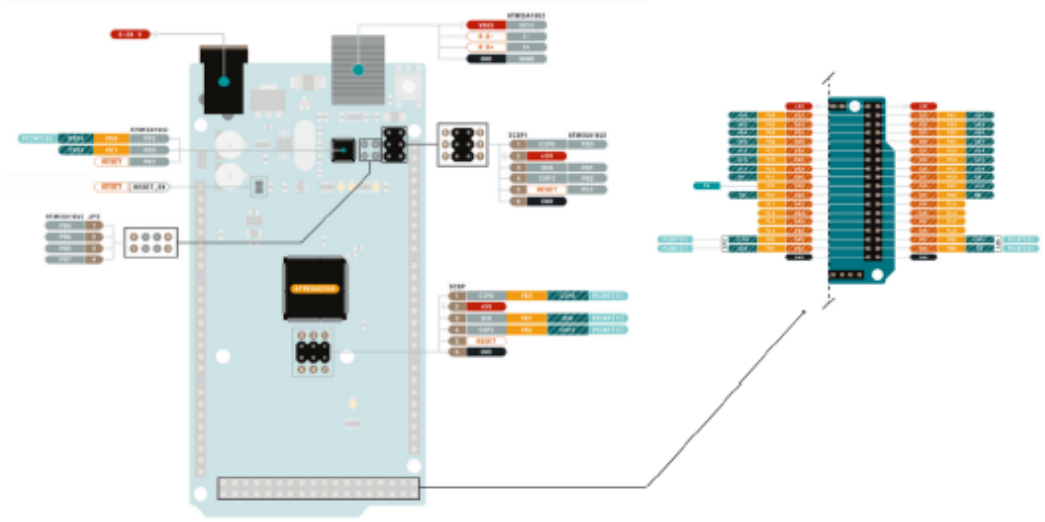
5.1 Analog

Pin	Function	Type	Description
1	NC	NC	Not Connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog	Analog input 0 /GPIO
10	A1	Analog	Analog input 1 /GPIO
11	A2	Analog	Analog input 2 /GPIO
12	A3	Analog	Analog input 3 /GPIO
13	A4	Analog	Analog input 4 /GPIO
14	A5	Analog	Analog input 5 /GPIO
15	A6	Analog	Analog input 6 /GPIO
16	A7	Analog	Analog input 7 /GPIO
17	A8	Analog	Analog input 8 /GPIO
18	A9	Analog	Analog input 9 /GPIO
19	A10	Analog	Analog input 10 /GPIO
20	A11	Analog	Analog input 11 /GPIO
21	A12	Analog	Analog input 12 /GPIO
22	A13	Analog	Analog input 13 /GPIO
23	A14	Analog	Analog input 14 /GPIO
24	A15	Analog	Analog input 15 /GPIO

5.2 Digital

Pin	Function	Type	Description
1	D21/SCL	Digital Input/I2C	Digital input 21/I2C Dataline
2	D20/SDA	Digital Input/I2C	Digital input 20/I2C Dataline
3	AREF	Digital	Analog Reference Voltage
4	GND	Power	Ground
5	D13	Digital/GPIO	Digital input 13/GPIO
6	D12	Digital/GPIO	Digital input 12/GPIO
7	D11	Digital/GPIO	Digital input 11/GPIO
8	D10	Digital/GPIO	Digital input 10/GPIO
9	D9	Digital/GPIO	Digital input 9/GPIO
10	D8	Digital/GPIO	Digital input 8/GPIO
11	D7	Digital/GPIO	Digital input 7/GPIO
12	D6	Digital/GPIO	Digital input 6/GPIO
13	D5	Digital/GPIO	Digital input 5/GPIO
14	D4	Digital/GPIO	Digital input 4/GPIO

Pin	Function	Type	Description
15	D3	Digital/GPIO	Digital input 3/GPIO
16	D2	Digital/GPIO	Digital input 2/GPIO
17	D1/TX0	Digital/GPIO	Digital input 1 /GPIO
18	D0/Tx1	Digital/GPIO	Digital input 0 /GPIO
19	D14	Digital/GPIO	Digital input 14 /GPIO
20	D15	Digital/GPIO	Digital input 15 /GPIO
21	D16	Digital/GPIO	Digital input 16 /GPIO
22	D17	Digital/GPIO	Digital input 17 /GPIO
23	D18	Digital/GPIO	Digital input 18 /GPIO
24	D19	Digital/GPIO	Digital input 19 /GPIO
25	D20	Digital/GPIO	Digital input 20 /GPIO
26	D21	Digital/GPIO	Digital input 21 /GPIO



Arduino Mega 2560 Rev3 Pinout

5.3 ATMEGA16U2 JP5

Pin	Function	Type	Description
1	PB4	Internal	Serial Wire Debug
2	PB6	Internal	Serial Wire Debug
3	PB5	Internal	Serial Wire Debug
4	PB7	Internal	Serial Wire Debug

5.4 ATMEGA16U2 ICSP1

Pin	Function	Type	Description
1	CIPO	Internal	Controller In Peripheral Out
2	+5V	Internal	Power Supply of 5V
3	SCK	Internal	Serial Clock
4	COPI	Internal	Controller Out Peripheral In
5	RESET	Internal	Reset
6	GND	Internal	Ground

5.5 Digital Pins D22 - D53 LHS

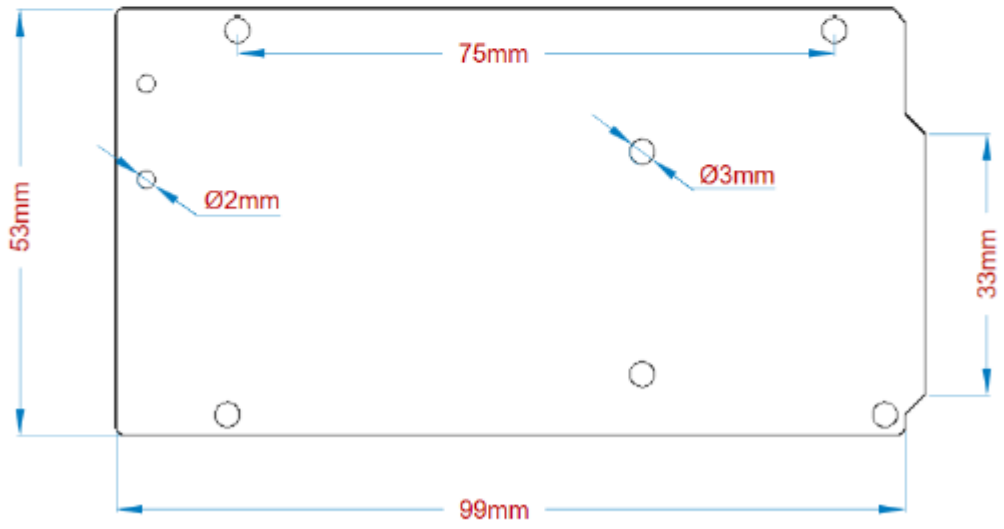
Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D22	Digital	Digital input 22/GPIO
3	D24	Digital	Digital input 24/GPIO
4	D26	Digital	Digital input 26/GPIO
5	D28	Digital	Digital input 28/GPIO
6	D30	Digital	Digital input 30/GPIO
7	D32	Digital	Digital input 32/GPIO
8	D34	Digital	Digital input 34/GPIO
9	D36	Digital	Digital input 36/GPIO
10	D38	Digital	Digital input 38/GPIO
11	D40	Digital	Digital input 40/GPIO
12	D42	Digital	Digital input 42/GPIO
13	D44	Digital	Digital input 44/GPIO
14	D46	Digital	Digital input 46/GPIO
15	D48	Digital	Digital input 48/GPIO
16	D50	Digital	Digital input 50/GPIO
17	D52	Digital	Digital input 52/GPIO
18	GND	Power	Ground

5.6 Digital Pins D22 - D53 RHS

Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D23	Digital	Digital input 23/GPIO
3	D25	Digital	Digital input 25/GPIO
4	D27	Digital	Digital input 27/GPIO
5	D29	Digital	Digital input 29/GPIO
6	D31	Digital	Digital input 31/GPIO
7	D33	Digital	Digital input 33/GPIO
8	D35	Digital	Digital input 35/GPIO
9	D37	Digital	Digital input 37/GPIO
10	D39	Digital	Digital input 39/GPIO
11	D41	Digital	Digital input 41/GPIO
12	D43	Digital	Digital input 43/GPIO
13	D45	Digital	Digital input 45/GPIO
14	D47	Digital	Digital input 47/GPIO
15	D49	Digital	Digital input 49/GPIO
16	D51	Digital	Digital input 51/GPIO
17	D53	Digital	Digital input 53/GPIO
18	GND	Power	Ground

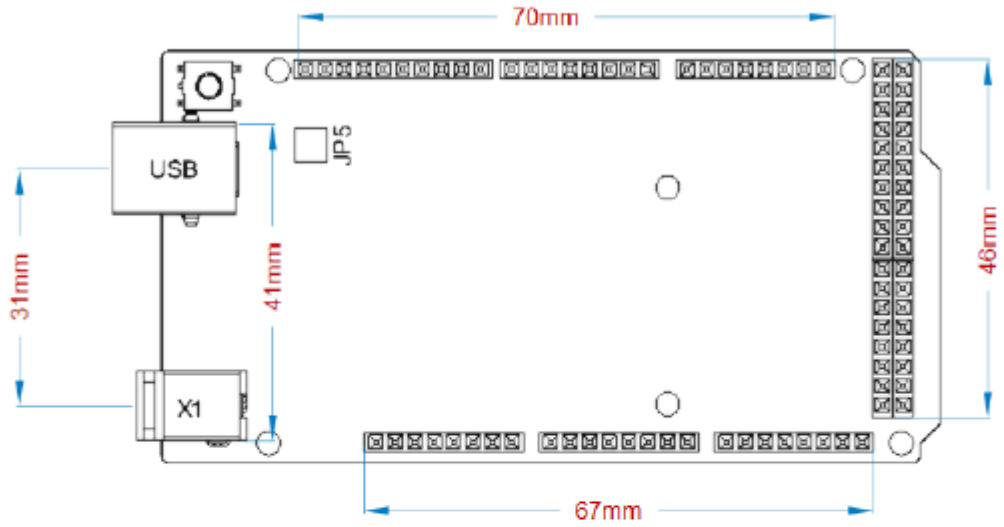
6 Mechanical Information

6.1 Board Outline



Arduino Mega 2560 Rev3 Outline

6.2 Board Mount Holes



Arduino Mega 2560 Rev3 Mount Holes

Certifications

7 Declaration of Conformity CE DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

IR sensor

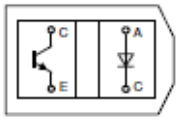
VISHAY.

TCRT5000, TCRT5000L

Vishay Semiconductors

Reflective Optical Sensor with Transistor Output

19156_2



Top view

19156_1

FEATURES

- Package type: leaded
- Detector type: phototransistor
- Dimensions (L x W x H in mm): 10.2 x 5.8 x 7
- Peak operating distance: 2.5 mm
- Operating range within > 20 % relative collector current: 0.2 mm to 15 mm
- Typical output current under test: I_C = 1 mA
- Daylight blocking filter
- Emitter wavelength: 950 nm
- Lead (Pb)-free soldering released
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

RoHS COMPLIANT

DESCRIPTION

The TCRT5000 and TCRT5000L are reflective sensors which include an infrared emitter and phototransistor in a leaded package which blocks visible light. The package includes two mounting clips. TCRT5000L is the long lead version.

APPLICATIONS

- Position sensor for shaft encoder
- Detection of reflective material such as paper, IBM cards, magnetic tapes etc.
- Limit switch for mechanical motions in VCR
- General purpose - wherever the space is limited

PRODUCT SUMMARY

PART NUMBER	DISTANCE FOR MAXIMUM CTR_{rel} (1) (mm)	DISTANCE RANGE FOR RELATIVE I_{out} > 20 % (mm)	TYPICAL OUTPUT CURRENT UNDER TEST (2) (mA)	DAYLIGHT BLOCKING FILTER INTEGRATED
TCRT5000	2.5	0.2 to 15	1	Yes
TCRT5000L	2.5	0.2 to 15	1	Yes

Notes

(1) CTR: current transfere ratio, I_{out}/I_{in}

(2) Conditions like in table basic charactristics/sensors

ORDERING INFORMATION

ORDERING CODE	PACKAGING	VOLUME (1)	REMARKS
TCRT5000	Tube	MOQ: 4500 pcs, 50 pcs/tube	3.5 mm lead length
TCRT5000L	Tube	MOQ: 2400 pcs, 48 pcs/tube	15 mm lead length

Note

(1) MOQ: minimum order quantity

ABSOLUTE MAXIMUM RATINGS (1)

PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
INPUT (EMITTER)				
Reverse voltage		V_R	5	V
Forward current		I_F	60	mA
Forward surge current	$t_p \leq 10$ µs	I_{FSM}	3	A
Power dissipation	$T_{amb} \leq 25$ °C	P_V	100	mW
Junction temperature		T_j	100	°C

ABSOLUTE MAXIMUM RATINGS (1)				
PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
OUTPUT (DETECTOR)				
Collector emitter voltage		V_{CEO}	70	V
Emitter collector voltage		V_{ECO}	5	V
Collector current		I_C	100	mA
Power dissipation	$T_{amb} \leq 55$ °C	P_V	100	mW
Junction temperature		T_j	100	°C
SENSOR				
Total power dissipation	$T_{amb} \leq 25$ °C	P_{tot}	200	mW
Ambient temperature range		T_{amb}	- 25 to + 85	°C
Storage temperature range		T_{stg}	- 25 to + 100	°C
Soldering temperature	2 mm from case, $t \leq 10$ s	T_{sd}	260	°C

Note
(1) T_{amb} = 25 °C, unless otherwise specified

ABSOLUTE MAXIMUM RATINGS

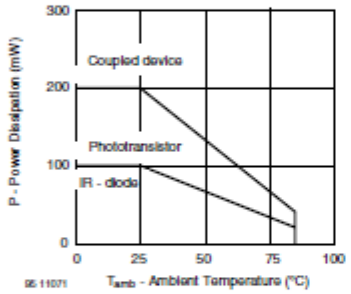


Fig. 1 - Power Dissipation Limit vs. Ambient Temperature

BASIC CHARACTERISTICS (1)						
PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
INPUT (EMITTER)						
Forward voltage	I_F = 60 mA	V_F		1.25	1.5	V
Junction capacitance	V_R = 0 V, f = 1 MHz	C_j		17		pF
Radiant intensity	I_F = 60 mA, t_p = 20 ms	I_e			21	mW/sr
Peak wavelength	I_F = 100 mA	λ_p	940			nm
Virtual source diameter	Method: 63 % encircled energy	d		2.1		mm
OUTPUT (DETECTOR)						
Collector emitter voltage	I_C = 1 mA	V_{CEO}	70			V
Emitter collector voltage	I_e = 100 µA	V_{ECO}	7			V
Collector dark current	V_{CE} = 20 V, I_F = 0 A, E = 0 lx	I_{CEO}		10	200	nA
SENSOR						
Collector current	V_{CE} = 5 V, I_F = 10 mA, D = 12 mm	I_C (2) (3)	0.5	1	2.1	mA
Collector emitter saturation voltage	I_F = 10 mA, I_C = 0.1 mA, D = 12 mm	V_{CEsat} (2) (3)			0.4	V

Note
(1) T_{amb} = 25 °C, unless otherwise specified
(2) See figure 3
(3) Test surface: mirror (Mfr. Spindler a. Hoyer, Part No. 340005)

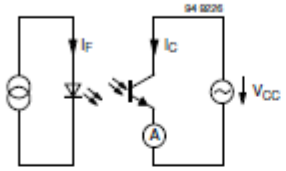


Fig. 2 - Test Circuit

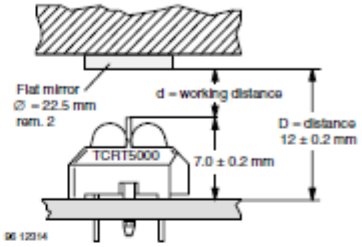


Fig. 3 - Test Circuit

BASIC CHARACTERISTICS

T_{amb} = 25 °C, unless otherwise specified

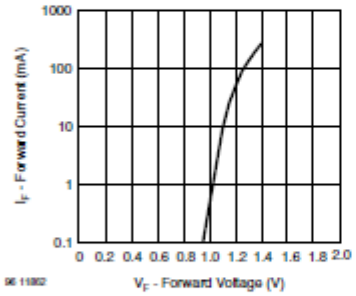


Fig. 4 - Forward Current vs. Forward Voltage

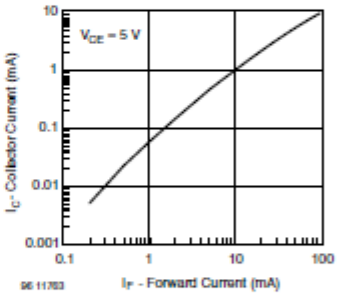


Fig. 6 - Collector Current vs. Forward Current

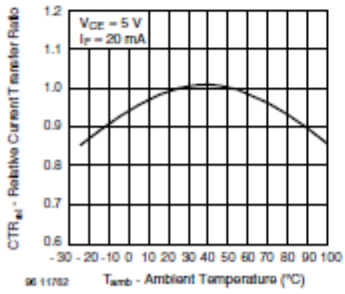


Fig. 5 - Relative Current Transfer Ratio vs. Ambient Temperature

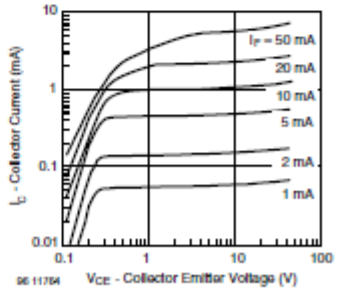


Fig. 7 - Collector Emitter Saturation Voltage vs. Collector Current

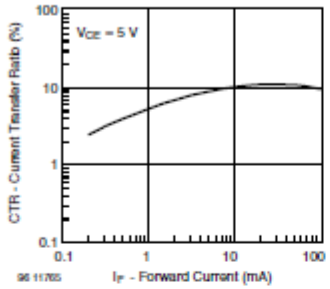


Fig. 8 - Current Transfer Ratio vs. Forward Current

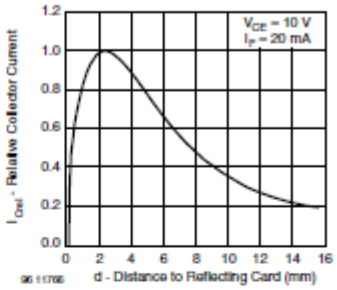
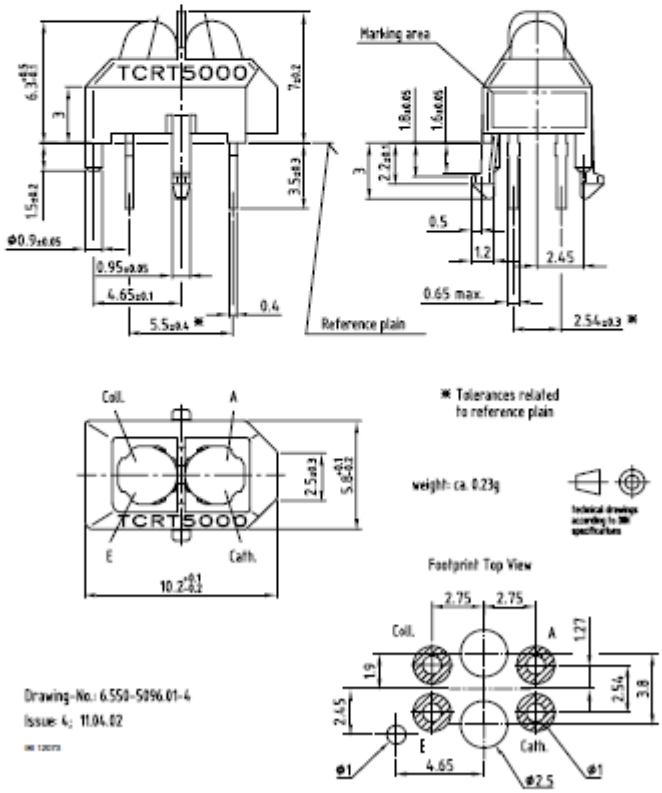
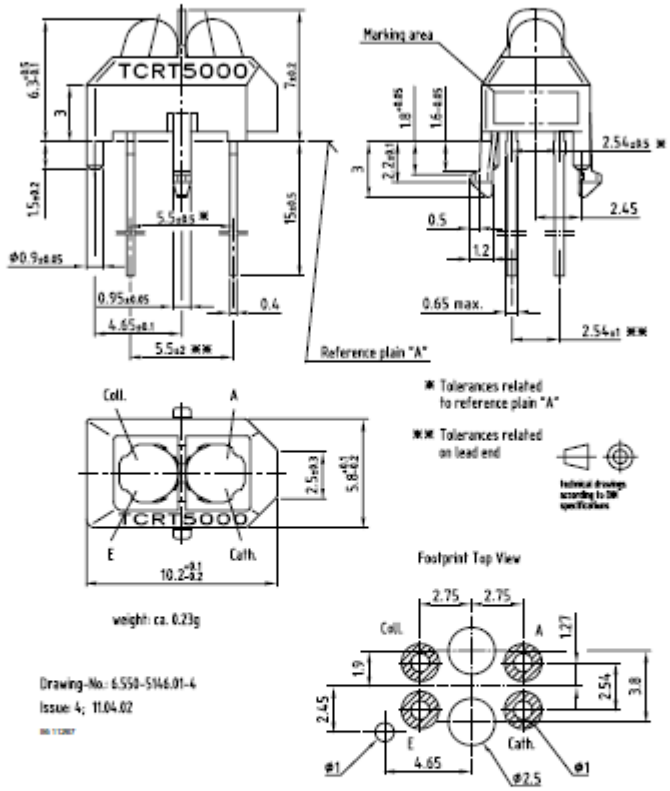


Fig. 9 - Relative Collector Current vs. Distance

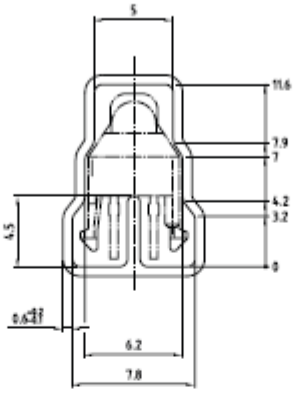
PACKAGE DIMENSIONS in millimeters, **TCRT5000**



PACKAGE DIMENSIONS in millimeters, **TCRT5000L**



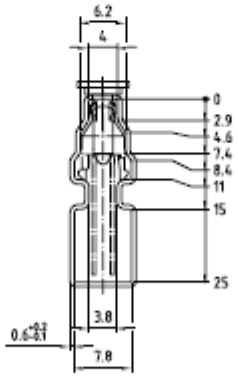
TUBE DIMENSIONS in millimeters, **TCRT5000**



With rubber stopper
Tolerance: ±0.5mm
Length: 575±1mm

Drawing-No.: 9.700-5139.01-4
Issue: 1; 10.05.00

TUBE DIMENSIONS in millimeters, **TCRT5000L**



With stopper pins
Tolerance: ±0.5mm
Length: 575±1mm

Drawing-No.: 9.700-5178.01-4
Issue: 1; 25.02.00

Ultrasonic sensor

Ultrasonic Ranging Module HC - SR04

□ Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time×velocity of sound (340M/S) / 2,

□ Wire connecting direct as following:

5V Supply

Trigger Pulse Input

Echo Pulse Output

0V Ground

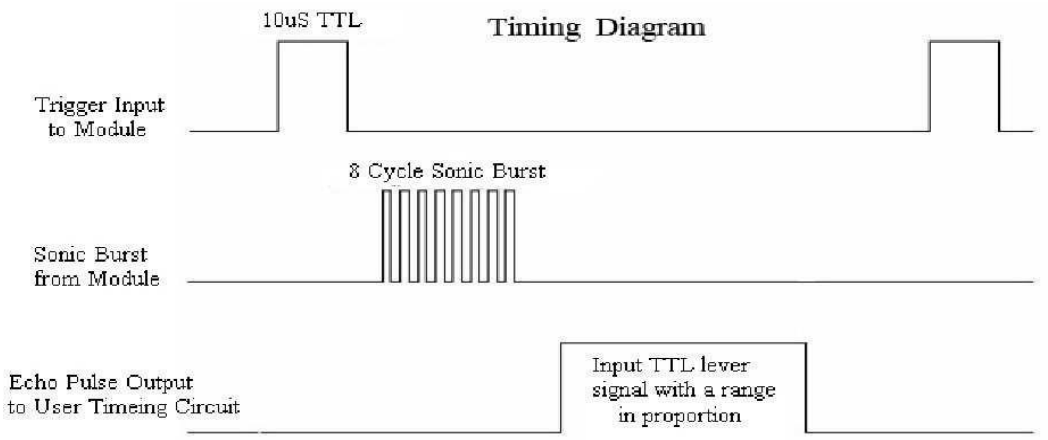
Electric Parameter

Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



□ Attention:

□ The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.

□ When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.Elecfreaks.com

Front Motor

Characteristics:

- **The rear of the motor is equipped with a dual channel Holzer effect encoder, AB dual output, single circuit per cycle pulse 12PPR, double down the road, a total output of 48CPR. Phase difference 90 degrees.**
- **The two-channel Hall effect encoder is used to sense the rotation of a magnetic disk on a rear protrusion of the motor shaft. The quadrature encoder provides a resolution of 48counts per revolution of the motor shaft.**
- **To compute the counts per revolution of the gearbox output, multiply the gear ratio by 48.**

Sweet tips:

1. **1.Be sure to keep the wiring connection correct before Working;**
2. **2.Be sure to keep the motor stopped before change the CCW/CW direction; Or the motor will be burned!!!**

Rear Motor

SP 30 Series

TG-38127000 Series

Cytron Technologies

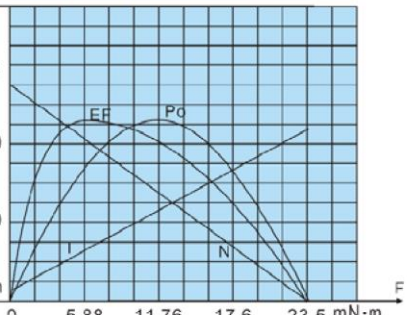
12VDC STANDARD DC GEARED MOTOR

Output Power: 15W(APPROX) Weight: 180~200g(APPROX)

Typical applications: Labal printers、auto shutter、welding machines、water meter IC card、grill、oven、cleaning machine、garbage disposers、household appliances、slot machines、money detector、automatic actuator、coffee machine、towel disposal、lighting、coin refund devices、peristaltic pump.

MOTOR CHARACTERISTICS

Po	EF	I	N
6	60	3.0	7500
4	40	2.0	5000
2	20	1.0	2500
W	%	A	r/min

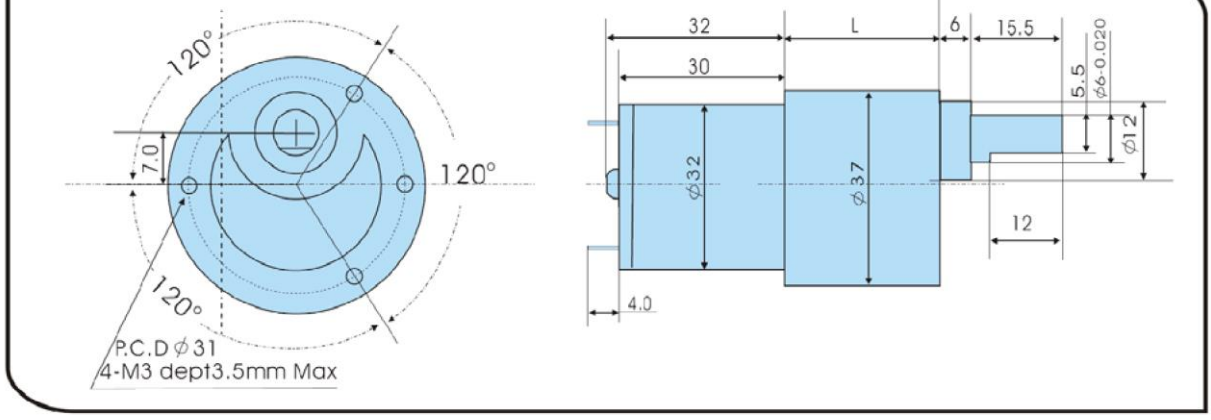


I:current N:speed
Po:output power F:Torque
EF:efficiency

MOTOR TORQUE/SPEED/CURRENT

Rated voltage	12VDC
No load speed	7000r/min
No load current	70mA
Rated torque	60gf • cm 5.88mN • m
Rated current	410mA
Rated speed	5200r/min
Stall torque	240gf • cm 23.5mN • m
Stall current	1.8A

APPEARANCE SIZE



Order Option							
Order Code	Input Voltage	Rated		Weight (g)	Power (w)	Diameter (mm)	L (mm)
		Speed (RPM)	Torque (mN.m)				
SPG10-30K	6	440	29.4	10	-	12	24
SPG10-150K	6	85	107.9	10	-	12	24
SPG10-298K	6	45	176.5	10	-	12	24
SPG20-50K	12	130	58.8	60	0.6	27.2	-
SPG30-20K	12	185	78.4	160	1.1	37	22
SPG30-30K	12	103	127.4	160	1.1	37	22
SPG30-60K	12	58	254.8	160	1.1	37	25
SPG30-150K	12	26	588	160	1.1	37	27
SPG30-200K	12	17	784	160	1.1	37	27
SPG30-300K	12	12	1176	160	1.1	37	27
SPG50-20K	12	170	196	300	3.4	37	23
SPG50-60K	12	56	588	300	3.4	37	26
SPG50-100K	12	34	980	300	3.4	37	26
SPG50-180K	12	17	1960	300	3.4	37	28

Motor driver chip

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.

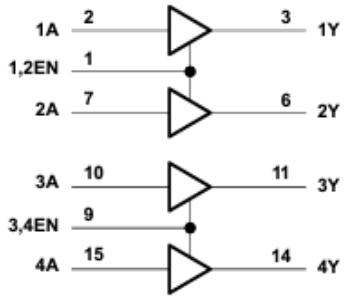
The L293 and L293D are characterized for operation from 0°C to 70°C.

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

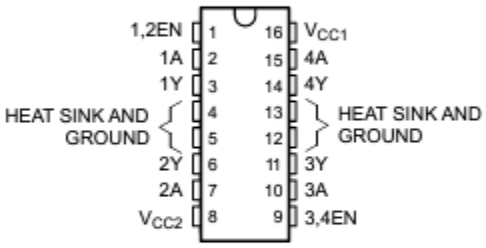
(1) For all available packages, see the orderable addendum at the end of the data sheet.

Logic Diagram



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

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report, SPRA953.
(2) The package thermal impedance is calculated in accordance with JESD 51-7.

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

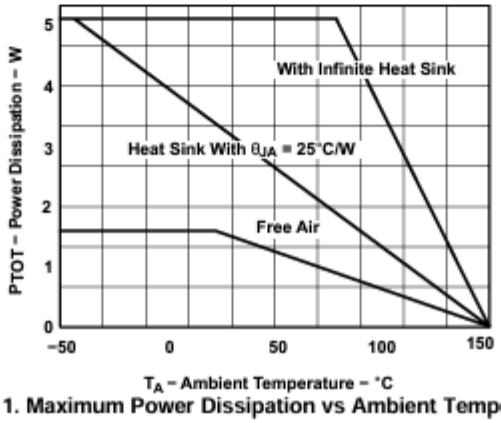
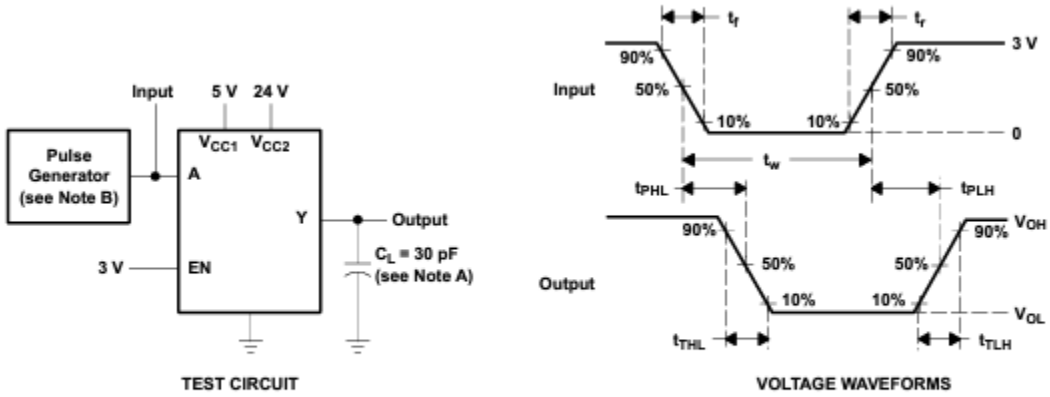


Figure 1. Maximum Power Dissipation vs Ambient Temperature

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

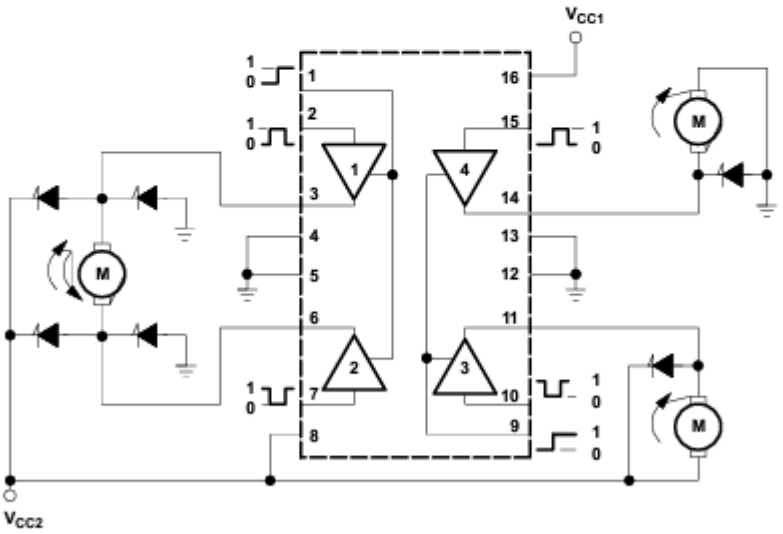
8.1 Overview

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



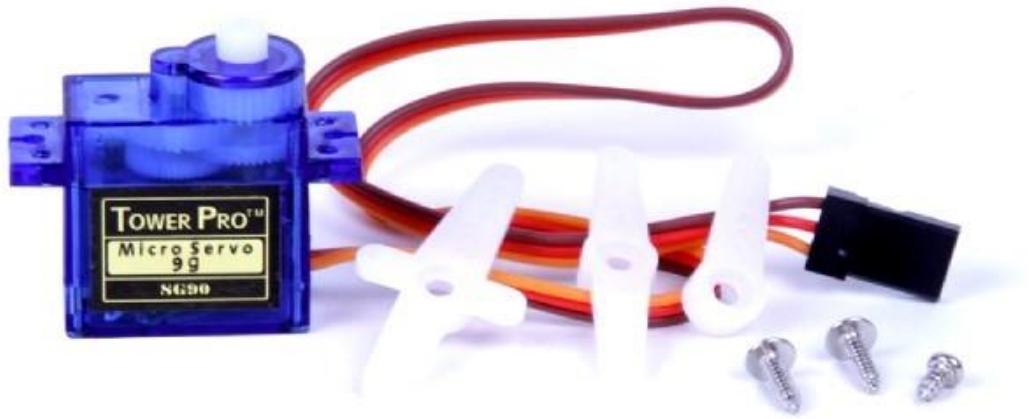
Output diodes are internal in L293D.

8.3 Feature Description

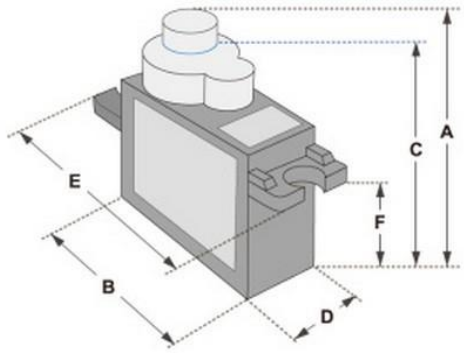
The L293x has TTL-compatible inputs and high voltage outputs for inductive load driving. Current outputs can get up to 2 A using the L293.

Servo motor

SERVO MOTOR SG90

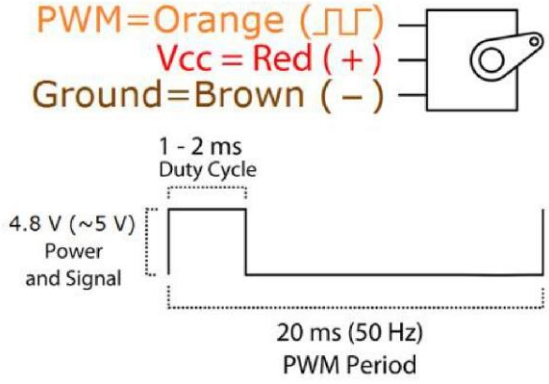


Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns



Dimensions & Specifications
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

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



Appendix C: Program Codes

Main Code

```
/*
The following code employs state machines to control the navigation of an autonomous mobile robot
via its ultrsonic sensors and IR sensors.
*/

#include <Servo.h>

Servo servo;

const int IRSensorR = A0; //ir sensor for beacon (6 inches tall) - right side
const int IRSensorL = A1; //left side

// GLOBAL VARIABLES
const int thresh = 22; // Ultrasonic Distance Threshold Variable
const int threshf = 25; //front ultrasonic threshold
const int irthreshold = 500;
int state = 0;        // State Machine Variable
const int speed = 190; //motor speeds

// PIN ASSIGNMENT
// // Ultrasonic Sensors Pins
const int echoLS2 = 22; // Rear Left Side Echo
const int trigLS2 = 23; // Rear Left Side Trigger
const int echoLS1 = 24; // Front Left Side Echo
const int trigLS1 = 25; // Front Left Side Trigger
```

```
const int echoFL = 26;  // Front Left Echo
const int trigFL = 27;  // Front Left Trigger
const int echoFR = 28;  // Front Right Echo
const int trigFR = 29;  // Front Right Trigger
const int echoRS1 = 30; // Front Right Side Echo
const int trigRS1 = 31; // Front Right Side Trigger
const int echoRS2 = 32; // Rear Right Side Echo
const int trigRS2 = 33; // Rear Right Side Trigger

// // Motor Pins
const int motorRLpin1 = 52; // Rear Left Motor A
const int motorRLpin2 = 50; // Rear Left Motor B
const int motorFRpin1 = 43; // Front Right Motor A
const int motorFRpin2 = 45; // Front Right Motor B
const int motorFLpin1 = 44; // Front Left Motor A
const int motorFLpin2 = 42; // Front Right Motor B
const int motorRRpin1 = 51; // Rear Right Motor A
const int motorRRpin2 = 53; // Rear Right Motor B
const int enfr = 3;        // Motor Speed Pin, speed fr = speed fl,rl,rr


// Global variable to track the timer for turning
unsigned long turnStartTime = 0;    // Stores the start time of the turn
bool isTurning = false;             // Tracks if the robot is currently turning
const unsigned long turnDuration = 1700; // Time (in milliseconds) to complete a 180-degree turn (adjust as needed)

void turnAround() {
```

```
    if (!isTurning) { // Start the turn
        isTurning = true;
        turnStartTime = millis(); // Record the start time
        clockwise();              // Begin turning
    }

    if (isTurning) {
        // Check if the turn duration has elapsed
        if (millis() - turnStartTime >= turnDuration) {
            stopMove();           // Stop the robot after turning
            isTurning = false;    // Reset the turning state
        }
    }
}

//states of robot during game.
//state 0 means the robot has not identified the target or base yet
//state 1 means the robot is stuck in place because of side restriction
//state 2 means the robot is turning around
//state 3 means the robot is moving left and right to find the IR beacon
//state 4 and 5 make the robot move left or right if there is a wall infront of it

// STATE BEHAVIOR FUNCTIONS

int state0behavior(int distanceFL = 0, int distanceFR = 0, int distanceRS1 = 0, int distanceRS2 =
0, int distanceLS1 = 0, int distanceLS2 = 0, int irsensorStatusR=0, int irsensorStatusL=0) {
  Serial.println("state 0");
```

```
  if ((irsensorStatusR < irthreshold) || (irsensorStatusL < irthreshold)){
    stopMove();
    shootpp();
    state = 2;
  }
  if ((distanceFL < threshf) && (distanceFR < threshf)) {
    state = 3;
  }


  else if (((distanceRS1 < thresh) || (distanceRS2 < thresh)) && ((distanceLS1 < thresh) ||
(distanceLS2 < thresh))) {
    state = 1;
  }
  else if (distanceFL < threshf){
    moveRight();
    //delay(250);
    }
  else if (distanceFR < threshf){
    moveLeft();
    //delay(250);
  }
  else if ((distanceRS1 < thresh) || (distanceRS2 < thresh)){
    moveLeft();
  }
  else if ((distanceLS1 < thresh) || (distanceLS2 < thresh)){
    moveRight();
  }
  else {
```

```
    moveForward();
  }
}


int state1behavior(int distanceFL = 0, int distanceFR = 0, int distanceRS1 = 0, int distanceRS2 =
0, int distanceLS1 = 0, int distanceLS2 = 0) {
  Serial.println("state 1");
  if ((distanceFL > threshf) && (distanceFR > threshf)) {
    moveForward();
    state = 0;
  }
  else {
    moveBackward();
    //delay(1000);
    if ((distanceRS1 < thresh) || (distanceRS2 < thresh)){
      moveLeft();
    }
    else if ((distanceLS1 < thresh) || (distanceLS2 < thresh)){
      moveRight();
    }
  }
}


int state2behavior(int distanceFL = 0, int distanceFR = 0, int distanceRS1 = 0, int distanceRS2 =
0, int distanceLS1 = 0, int distanceLS2 = 0) {
  Serial.println("state 2");
  turnAround();
```

```
  if (!isTurning){
    state = 0;
  }
}

int state3behavior(int distanceFL = 0, int distanceFR = 0, int distanceRS1 = 0, int distanceRS2 =
0, int distanceLS1 = 0, int distanceLS2 = 0, int irsensorStatusR=0, int irsensorStatusL=0) {
  Serial.println("state 3");
  bool left = false;
  bool right = false;

  if (distanceLS1 < thresh) {
    left = true;
    right = false;

  }
  else if (distanceRS1 < thresh) {
    right = true;
    left = false;
  }
  else if ((distanceFL > threshf) && (distanceFR > threshf)){
    state = 0;
  }

  if (left == true){
    //moveRight();
    state = 5;
  }
```

```
  else if (right == true){
    //moveLeft();
    state = 4;
  }
  else{
    state = 4;
  }

  if ((irsensorStatusR < irthreshold) || (irsensorStatusL < irthreshold)){
    stopMove();
    shootpp();
    state = 2;
  }
}

int state4behavior(int distanceFL = 0, int distanceFR = 0, int distanceRS1 = 0, int distanceRS2 =
0, int distanceLS1 = 0, int distanceLS2 = 0, int irsensorStatusR=0, int irsensorStatusL=0)
{
  if(distanceLS1 > thresh)
  {
    moveLeft();
  }
  else if ((distanceFL > threshf) && (distanceFR > threshf)){
    state = 0;
  }
  else if(distanceRS1 > thresh)
  {
    state = 5;
```

```
  }
  if ((irsensorStatusR < irthreshold) || (irsensorStatusL < irthreshold)){
    stopMove();
    shootpp();
    state = 2;
  }
}


int state5behavior(int distanceFL = 0, int distanceFR = 0, int distanceRS1 = 0, int distanceRS2 =
0, int distanceLS1 = 0, int distanceLS2 = 0, int irsensorStatusR=0, int irsensorStatusL=0)
{
  if(distanceRS1 > thresh)
  {
    moveRight();
  }
  else if ((distanceFL > threshf) && (distanceFR > threshf)){
    state = 0;
  }
  else if(distanceLS1 > thresh)
  {
    state = 4;
  }
  if ((irsensorStatusR < irthreshold) || (irsensorStatusL < irthreshold)){
    stopMove();
    shootpp();
    state = 2;
  }
}
```

```
void setup() {
  // Serial
  Serial.begin(115200);
  // Set Pin Modes
  Serial.print("ARDUINO RESET");
  pinMode(IRSensorR, INPUT);
  pinMode(IRSensorL, INPUT);
  pinMode(motorRLpin1, OUTPUT);
  pinMode(motorRLpin2, OUTPUT);
  pinMode(motorFRpin1, OUTPUT);
  pinMode(motorFRpin2, OUTPUT);
  pinMode(motorFLpin1, OUTPUT);
  pinMode(motorFLpin2, OUTPUT);
  pinMode(motorRRpin1, OUTPUT);
  pinMode(motorRRpin2, OUTPUT);
  pinMode(enfr, OUTPUT);          // Same speed set for all four motors
  pinMode(trigLS2, OUTPUT);
  pinMode(echoLS2, INPUT);
  pinMode(trigLS1, OUTPUT);
  pinMode(echoLS1, INPUT);
  pinMode(trigFL, OUTPUT);
  pinMode(echoFL, INPUT);
  pinMode(trigFR, OUTPUT);
  pinMode(echoFR, INPUT);
  pinMode(trigRS1, OUTPUT);
  pinMode(echoRS1, INPUT);
```

```
  pinMode(trigRS2, OUTPUT);
  pinMode(echoRS2, INPUT);
  servo.attach(9);
  servo.write(0);
}

void loop() {
  int irsensorStatusR = analogRead(IRSensorR);
  int irsensorStatusL = analogRead(IRSensorL);
  int distanceLS2 = getDistance(trigLS2, echoLS2);
  int distanceLS1 = getDistance(trigLS1, echoLS1);
  int distanceFL = getDistance(trigFL, echoFL);
  int distanceFR = getDistance(trigFR, echoFR);
  int distanceRS1 = getDistance(trigRS1, echoRS1);
  int distanceRS2 = getDistance(trigRS2, echoRS2);

  analogWrite(enfr, speed);          // Set Motor Speed
  Serial.print("R: ");
  Serial.print(irsensorStatusR);
  Serial.print("L: ");
  Serial.println(irsensorStatusL);

  //logic: if the ir sensor detects an ir led upto a certain range, it will stop moving entirely.
otherwise, it will be constantly moving
  if (state==0){

    state0behavior(distanceFL, distanceFR, distanceRS1, distanceRS2, distanceLS1, distanceLS2,
irsensorStatusR, irsensorStatusL);
```

```
    }
  if (state==1){
    state1behavior(distanceFL, distanceFR, distanceRS1, distanceRS2, distanceLS1,
distanceLS2);
    }

  if (state==2){
    state2behavior(distanceFL, distanceFR, distanceRS1, distanceRS2, distanceLS1,
distanceLS2);
    }
  if (state==3){
    state3behavior(distanceFL, distanceFR, distanceRS1, distanceRS2, distanceLS1, distanceLS2,
irsensorStatusR, irsensorStatusL);
  }
  if (state==4){
    state4behavior(distanceFL, distanceFR, distanceRS1, distanceRS2, distanceLS1, distanceLS2,
irsensorStatusR, irsensorStatusL);
  }
  if (state==5){
    state5behavior(distanceFL, distanceFR, distanceRS1, distanceRS2, distanceLS1, distanceLS2,
irsensorStatusR, irsensorStatusL);
  }
  }

// Functions made to simplify code

void shootpp(){ //shoot ping pong
  //servo stuff to move
  servo.write(120);
```

```
}

int stopMove(){
   digitalWrite(motorFRpin1, LOW);
   digitalWrite(motorFRpin2, LOW);
   digitalWrite(motorFLpin1, LOW);
   digitalWrite(motorFLpin2, LOW);
   digitalWrite(motorRRpin1, LOW);
   digitalWrite(motorRRpin2, LOW);
   digitalWrite(motorRLpin1, LOW);
   digitalWrite(motorRLpin2, LOW);
}

int moveForward(){ //ALL MOTORS MOVE FORWARD
   digitalWrite(motorFRpin1, HIGH);
   digitalWrite(motorFRpin2, LOW);
   digitalWrite(motorFLpin1, HIGH);
   digitalWrite(motorFLpin2, LOW);
   digitalWrite(motorRRpin1, HIGH);
   digitalWrite(motorRRpin2, LOW);
   digitalWrite(motorRLpin1, HIGH);
   digitalWrite(motorRLpin2, LOW);
   delay(50);
}

int moveRight(){
   digitalWrite(motorFRpin1, LOW);
```

```
    digitalWrite(motorFRpin2, HIGH);
    digitalWrite(motorFLpin1, HIGH);
    digitalWrite(motorFLpin2, LOW);
    digitalWrite(motorRRpin1, HIGH);
    digitalWrite(motorRRpin2, LOW);
    digitalWrite(motorRLpin1, LOW);
    digitalWrite(motorRLpin2, HIGH);
    delay(50);
}

int moveLeft(){
    digitalWrite(motorFRpin1, HIGH);
    digitalWrite(motorFRpin2, LOW);
    digitalWrite(motorFLpin1, LOW);
    digitalWrite(motorFLpin2, HIGH);
    digitalWrite(motorRRpin1, LOW);
    digitalWrite(motorRRpin2, HIGH);
    digitalWrite(motorRLpin1, HIGH);
    digitalWrite(motorRLpin2, LOW);
    delay(50);
}

int moveBackward(){
    digitalWrite(motorFRpin1, LOW);
    digitalWrite(motorFRpin2, HIGH);
    digitalWrite(motorFLpin1, LOW);
    digitalWrite(motorFLpin2, HIGH);
```

```
    digitalWrite(motorRRpin1, LOW);
    digitalWrite(motorRRpin2, HIGH);
    digitalWrite(motorRLpin1, LOW);
    digitalWrite(motorRLpin2, HIGH);
}

int clockwise(){
    digitalWrite(motorFRpin1, LOW);
    digitalWrite(motorFRpin2, HIGH);
    digitalWrite(motorFLpin1, HIGH);
    digitalWrite(motorFLpin2, LOW);
    digitalWrite(motorRRpin1, LOW);
    digitalWrite(motorRRpin2, HIGH);
    digitalWrite(motorRLpin1, HIGH);
    digitalWrite(motorRLpin2, LOW);
}
int cclockwise(){
    digitalWrite(motorFRpin1, HIGH);
    digitalWrite(motorFRpin2, LOW);
    digitalWrite(motorFLpin1, LOW);
    digitalWrite(motorFLpin2, HIGH);
    digitalWrite(motorRRpin1, HIGH);
    digitalWrite(motorRRpin2, LOW);
    digitalWrite(motorRLpin1, LOW);
    digitalWrite(motorRLpin2, HIGH);
}
int getDistance(int triggerPin, int echoPin) { //gets distance of ultrasonic sensors
```

```
  long duration;
  int distance;

  // Send a 10-microsecond pulse to trigger the sensor
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);

  // Read the echo pin, which returns the pulse width in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  distance = duration * 0.034 / 2;

  return distance;
}
```

Ultrasonic Testing Code

```
const int echoLS2 = 22; // Rear Left Side Echo
const int trigLS2 = 23; // Rear Left Side Trigger
const int echoLS1 = 24; // Front Left Side Echo
const int trigLS1 = 25; // Front Left Side Trigger
const int echoFL = 26;  // Front Left Echo
const int trigFL = 27;  // Front Left Trigger
const int echoFR = 28;  // Front Right Echo
const int trigFR = 29;  // Front Right Trigger
const int echoRS1 = 30; // Front Right Side Echo
```

```
const int trigRS1 = 31; // Front Right Side Trigger
const int echoRS2 = 32; // Rear Right Side Echo
const int trigRS2 = 33; // Rear Right Side Trigger



void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(trigLS2, OUTPUT);
  pinMode(echoLS2, INPUT);
  pinMode(trigLS1, OUTPUT);
  pinMode(echoLS1, INPUT);
  pinMode(trigFL, OUTPUT);
  pinMode(echoFL, INPUT);
  pinMode(trigFR, OUTPUT);
  pinMode(echoFR, INPUT);
  pinMode(trigRS1, OUTPUT);
  pinMode(echoRS1, INPUT);
  pinMode(trigRS2, OUTPUT);
  pinMode(echoRS2, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  int distanceLS2 = getDistance(trigLS2, echoLS2);
  int distanceLS1 = getDistance(trigLS1, echoLS1);
```

```
  int distanceFL = getDistance(trigFL, echoFL);
  int distanceFR = getDistance(trigFR, echoFR);
  int distanceRS1 = getDistance(trigRS1, echoRS1);
  int distanceRS2 = getDistance(trigRS2, echoRS2);
  Serial.print("LS2: ");
  Serial.print(distanceLS2);
  Serial.print(" | LS1: ");
  Serial.print(distanceLS1);
  Serial.print(" | FL: ");
  Serial.print(distanceFL);
  Serial.print(" | FR: ");
  Serial.print(distanceFR);
  Serial.print(" | RS1: ");
  Serial.print(distanceRS1);
  Serial.print(" | RS1: ");
  Serial.println(distanceRS2);
}

int getDistance(int triggerPin, int echoPin) {
  long duration;
  int distance;

  // Send a 10-microsecond pulse to trigger the sensor
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
```

```
  digitalWrite(triggerPin, LOW);

  // Read the echo pin, which returns the pulse width in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance in centimeters
  distance = duration * 0.034 / 2;

  return distance;
}
```

IR Testing Code

```
int irsensorL = A0;
int irsensorR = A1;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(irsensorL, INPUT);
  pinMode(irsensorR, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  int sensorStatusL = analogRead(irsensorL);
  int sensorStatusR = analogRead(irsensorR);
  Serial.print(sensorStatusL);
  Serial.print(" | ");
  Serial.println(sensorStatusR);
```

```
}
```

7. References

[1] Autonomous Mobile Robots 2024, http://faculty.cooper.edu/mar/Welcome.html

[2] "25GA370 DC Encoder Metal Gearmotor 12V High Speed 150RPM Gear Motor with Two-Channel Hall Effect Encoder for DIY Parts", Amazon, Amazon.com: BEMONOC 25GA370 DC Encoder Metal Gearmotor 12V High Speed 150RPM Gear Motor with Two-Channel Hall Effect Encoder for DIY Parts : Industrial & Scientific

[3] "12V, 150RPM 30:1 Gear Motor w / Encoder", Robotshop, https://eu.robotshop.com/products/12v-150rpm-301-gear-motor-w-encoder

[4] Mecanum Wheel, Wikipedia. https://en.wikipedia.org/wiki/Mecanum_wheel#/media/File:Mecanum_wheel_control_principle.svg

[5] Arduino Mega 2560 rev3. (n.d.). Arduino Official Store. https://store.arduino.cc/products/arduino-mega-2560-rev3

[6] "L293d H-Bridge Motor Driver IC: What Is, Working, Pin Diagram", Discrete, https://discrete.co.in/blog/l293d-h-bridge-motor-driver-ic-working-diagram

[7] "HiLetgo 5pcs HC-SR04 Ultrasonic Module Distance Sensor", Amazon, https://www.amazon.com/HiLetgo-HC-SR04-Ultrasonic-Distance-MEGA2560/dp/B00E87VXH0?crid=2K78VSU19EQR9&dib=eyJ2IjoiMSJ9.QDSY4r8kdI-Maqb5OGqDzvuZ-EfbuACMS2UCZICmB0_67fr_F5rD0nxGnyvOO647AQ0wyuIYDFKm0pbe6hl5UhkSGZmskf_urZnodWbecnMu_s0Iod1JUacucsC7zRfGyiNCZkf5_z7fbJgRuWqPO1HtdZZjsA916D8gw8nWD_O8ybvO-FTilHu81rSy8j7YdbM2DbTrZp0qKpoMT5_DHLDdXuZykLltgNV3ZliB_4M.bEc6Y2CHJp613YsuinYo2iJtPz0WRIhRZ82kZ0Sllnk&dib_tag=se&keywords=ultrasonic+sensor&qid=1734714941&sprefix=ultrasonic+sensor%2Caps%2C123&sr=8-11

[8] HiLetGo 10pcs TCRT5000 Infrared Reflective Sensor IR Photoelectric Switch Barrier Line Obstacle Avoidance Module Tracing Sensor Tracing Module for Arduino Smart Car Robot: Amazon.com: Industrial & Scientific. (n.d.). https://www.amazon.com/HiLetgo-Channel-Tracing-Sensor-Detection/dp/B00LZV1V10?crid=10WIX7Y9DCUZK&dib=eyJ2IjoiMSJ9.uBiXlLxMkdrrYltHQAPYp4pRPiJv62HcKI7dkBLoYVZWi6HHn93eGoV4AIz7U3tYrO6UitgzfXiw5FPagutyirL-f7Orfr6Vzpb5aOxzDaEstoQEBvTF9TLAbHIlDTTf6MC3amX-W0y0kA7plsaMIPo2ERXN1LosYSVV-PlsmiIRlUJakBrvYHO6VcDEZpRkOCv7AAMxaCV4sAfTQl_IV9n_l2CaL38RETwYIkPwCPk.33354ayG4alPGBV3atHbAZaJkMWzQ4Y29tWbtikNRuI&dib_tag=se&keywords=tcrt5000+sensor&qid=1734712317&sprefix=tcrt5000+%2Caps%2C102&sr=8-1

[9] “SG90 9G Micro Servo Metal Geared Motor”, Amazon, https://www.amazon.com/Smraza-Helicopter-Airplane-Control-Arduino/dp/B07L2SF3R4?crid=1VT3IQ1L0EX56&dib=eyJ2IjoiMSJ9.w0Xv0V9ezw2OMZIJmWpQPkzp83NSGXHa6nsKq5ydQWbmZoG1zWija-JF4VT5Co2hII3t_7d8cD75DJRAag0SSCVX1fZRcJ3t-KHUp3Gp6kNQ6hQLMAwbh3lgMdL9Mkt4ZozyWxl3kV9IkO4AyB8VjYuoNXsoQxx3TOPYRtSy2FxZ5ay1kb29gpSuyNmlcnscY3yl--4dfpctKX4d6yiSo7-yYxB0sgHWm-3Aj_Va1NSRQRh-wrkuT0bsnG6MpDrLFlDNyiJfbv4jL4fZLsXE7ZgPZ9ATapFVNLkmdYjEC2g2ar6obUvBxtRPwhQV3tRSYJBIFQYooZeWf4qLl2HTt3yM1bnMAK4MMpn15lR4nCKzHQkKqMOZZ45opl7y43rqHSlDmvOJ71uTwXB85juS59FtUC4xtQCexjEalGVGpRgfm1n14OwL0lWmleeMgeCn.fU_Lbu0icyuJFzNID03Uq04MzETwyuKtzsMmRecKd58&dib_tag=se&keywords=servo%2Bmotor&qid=1734716865&sprefix=servo%2Bmotor%2Caps%2C149&sr=8-7&th=1

[10] " Tectra 12V RC Battery with Bare Leads, 2200mAh High Capacity Ni-MH 10 Cells Rechargeable Battery Pack for RC Car, RC Boat, RC Robot, RC Airplane, DIY and More”, Amazon, https://www.amazon.com/Battery-Tectra-Capacity-Rechargeable-Airplane/dp/B09P4WWFFM/ref=sr_1_4?crid=2I1FIOUZPBOR5&dib=eyJ2IjoiMSJ9.xUaSI4tptptXcRQF9sRXeaspQ1qwjCLygMxlm04zuwz6x7J7V-ZdyBB3Fx2LyT11A6lHTMDOxrjq8EImou9i2vFDpd2c7LNogEjO_fFgvExsJxx8rqp-EYwkxNXvXcpGHllKJCL_0P1A-_xeLEIGLS8RzjqzAyDkzU8Th4HLwg_AYD6rKwM5l1nZe_pnkRfcQV6cTqB-sR2AF4VvP3g-eRbCXIbxCls9aNvld_po86q-rmXyP3bb3o0DCSP1EWhVpehUodfsXOEwE_daBf7u67hJlc2EpO7D86io5BF_E1lo.EUrbLOEi_2OMPXO-EN6UBiK4j8Ip921dcqictKhOMDE&dib_tag=se&keywords=12v+nimh&qid=1710180964&sprefix=12v+nimh%2Caps%2C101&sr=8-4