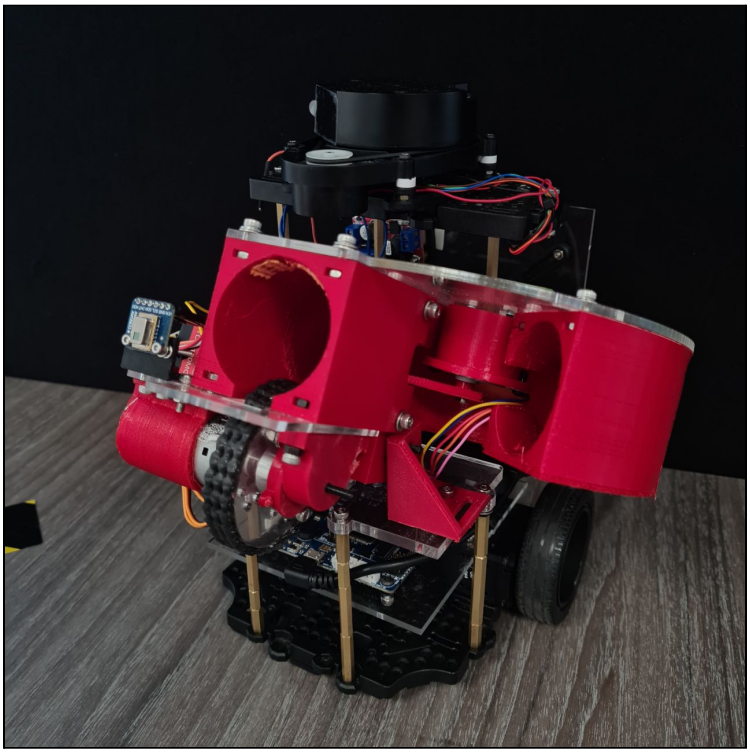
Engineering Documentation
EG2310: Fundamentals of System Design



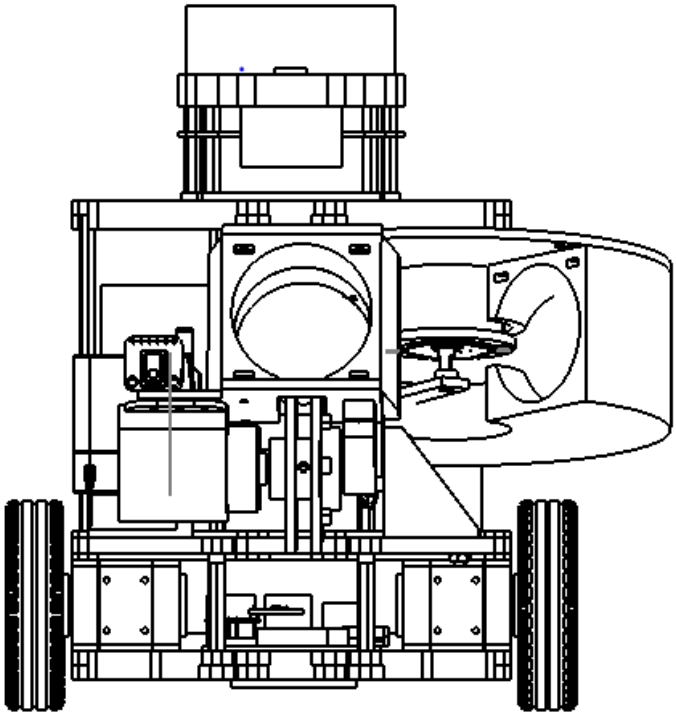
Group 1
Alvin Ben Abraham Alissa Are Mann Oo Lim Jun Yong

1. INTRODUCTION

1.1. Introduction

This documentation details the assembly, pre-, during-, and post-operation procedures, and troubleshooting advice for the Super Automated Machine (SAM).

SAM is capable of autonomously navigating a bounded maze, and identifying and firing at a target emitting infra-red radiation concurrently.



1.2. Safety Precautions

❏ Please be aware of your own safety as you assemble. SAM is not responsible for the result of any accidents caused by the user's negligence.

a) **Read through the manual carefully before assembly.**

b) **Keep a safe distance from SAM during its activation.**

c) **Be careful not to get fingers stuck between SAM's joints and wheel.**

d) **SAM is not waterproof. Do not use it near water.**

e) **Do not use SAM near heat or fire.**

f) **Do not soak or place the Li-Po battery in or near water.**

g) **Do not connect or let the Li-Po battery come in contact with hair pins, clips or any metal objects.**

h) **Do not use excessive force on nuts, bolts, or other parts of SAM.**

i) **Avoid activating SAM on the desk or table to avoid falling.**

1.3. System Technical Specifications

1.3.1. SAM Specification

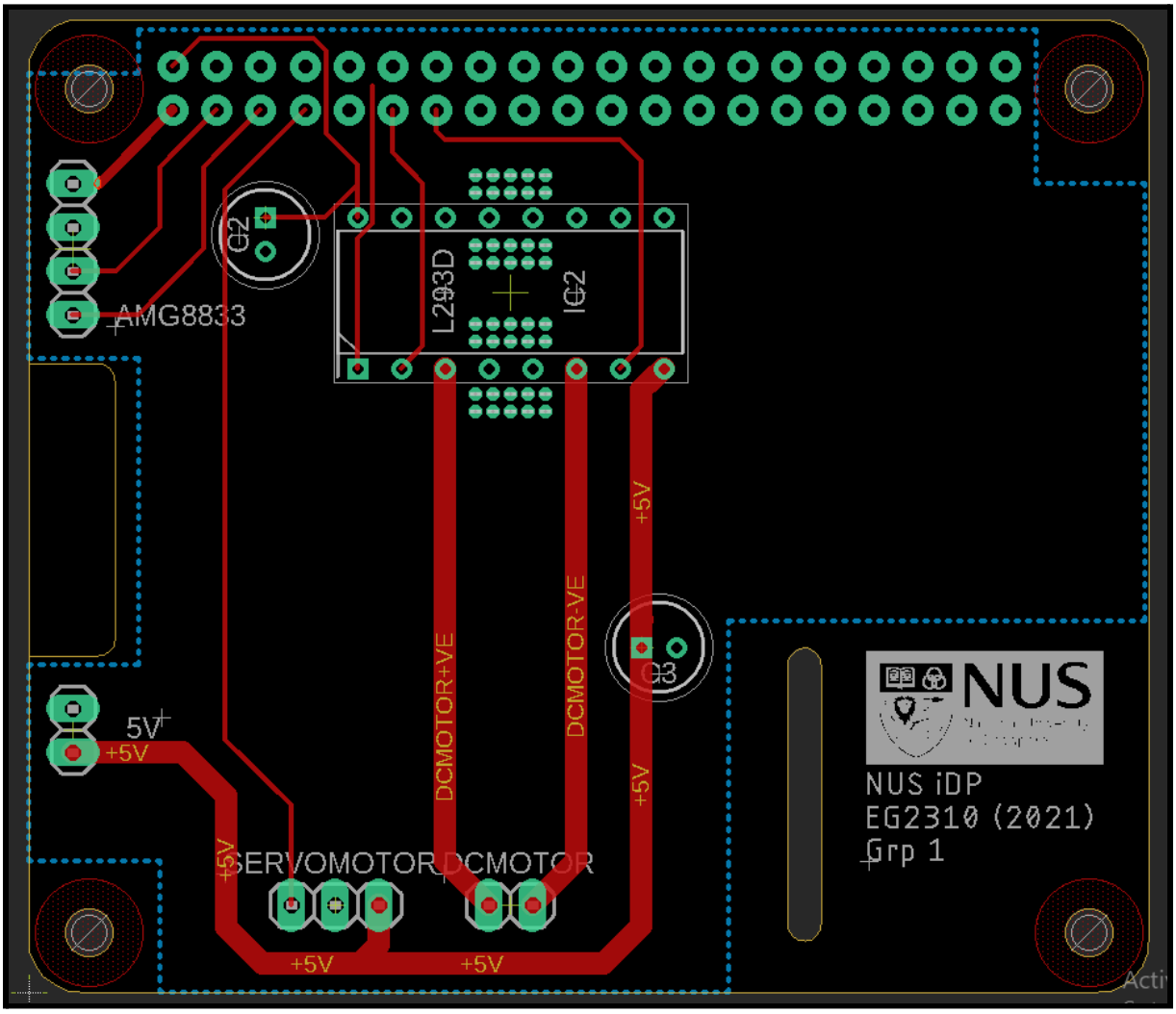
SAM	
Maximum translational velocity:	0.22 m/s
Maximum rotational velocity:	2.84 rad/s (162.72 deg/s)
Size (L x W x H) mm:	230 x 208 x 222
Weight (+ Battery + Firing Mechanism + Camera):	1449.0g
Expected operating time:	1hr
Expected charging time:	2hr 30m
Single Board Computer:	Raspberry Pi 3 Model B+
MCU:	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
Actuator:	XL430-W250
Laser Distance Sensor:	360 Laser Distance Sensor LDS-01
IMU	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 3 Axis

1.3.2. Firing Mechanism Specifications

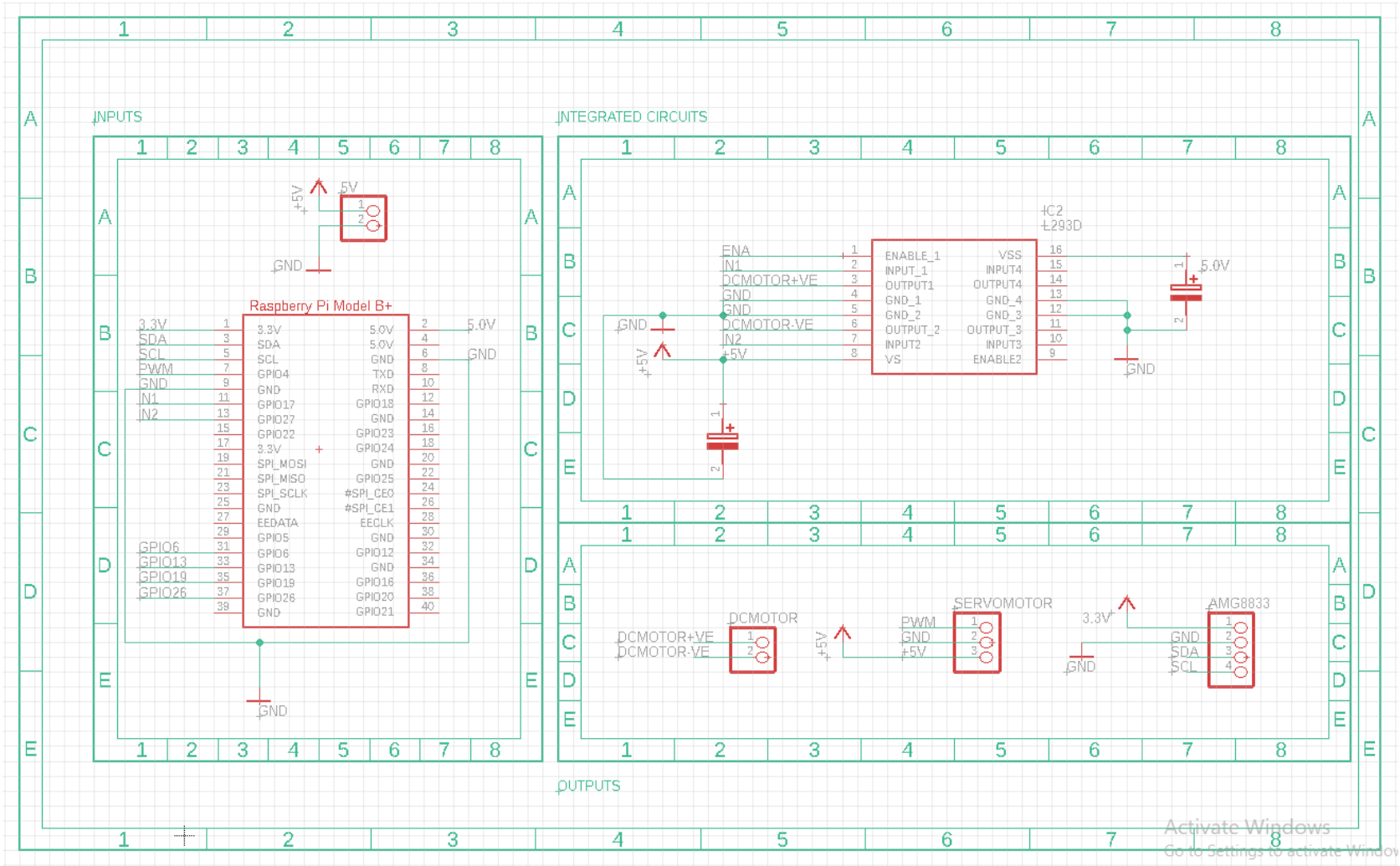
Firing Mechanism	
Maximum flywheel rotational speed:	22356 rpm
Maximum feeding speed:	25.6 rad/s
Maximum load (Payload Carrier):	300g
Maximum Capacity:	5 balls
Angular Boundary (Arm):	0 - 35 degree (0 degree is horizontal to ground)
Actuator:	Feeder 28BYJ-48 Flywheel RS Pro 238-9721 Arm TD-8120MG
Sensor:	AMG8833

1.3.3. Electronic System Architecture

Printed Circuit Board Layout



Printed Circuit Board Schematics



1.3.4. Power Management

The main power supply is a 11.1 V, 2200 mAh Li-Po Battery powering the entire circuit. This battery is connected directly to the OpenCR. From the OpenCR, it is connected to 4 locations:

a) Dynamixels
b) Raspberry Pi (powers AMG8833 camera, LiDAR)
c) Raspberry Pi Hat (powers RS PRO DC Motor, TD8120-MG Servo Motor)
d) ULN2003 Motor Driver (powers 28BYJ-48 Stepper Motor)

As the power requirements are low, we will only be considering the maximum average power consumption of the below components. The power consumption differs across different operating stages, of which there are three:

a) Pre-Target Identification

During this stage, SAM is powering the Dynamixels and the Raspberry Pi only. The power draw is as such:

Component	Voltage/Current (V/A)	Power Consumption (W)
Dynamixels	11.1, 1.3 (max)	14.3
Raspberry Pi	5, 0.4	2.0
AMG8833	3.3, 0.0045	0.001
Total	-	16.3

b) Target Identification

During this stage, SAM is powering all components simultaneously. The power draw is as such:

Component	Voltage/Current (V/A)	Power Consumption (W)
Dynamixels	11.1, 1.3 (max)	14.3
Raspberry Pi	5, 0.4	2.0
AMG8833	3.3, 0.0045	0.001
RS PRO DC Motor	5, 1	5.0
TD8120-MG Servo Motor	5,1	5.0
28-BYJ48 Stepper Motor	5,0.4	2.0
Total	-	28.3

c) Post-Target Identification

During this stage, SAM is powering the Dynamixels and Raspberry Pi only. However, the AMG8833 thermal camera has been deactivated as there is only one target to be identified. The power draw is as such:

Component	Voltage/Current (V/A)	Power Consumption (W)
Dynamixels	11.1, 1.3 (max)	14.3
Raspberry Pi	5, 0.4	2
Total	-	16.3

Overall Consideration:

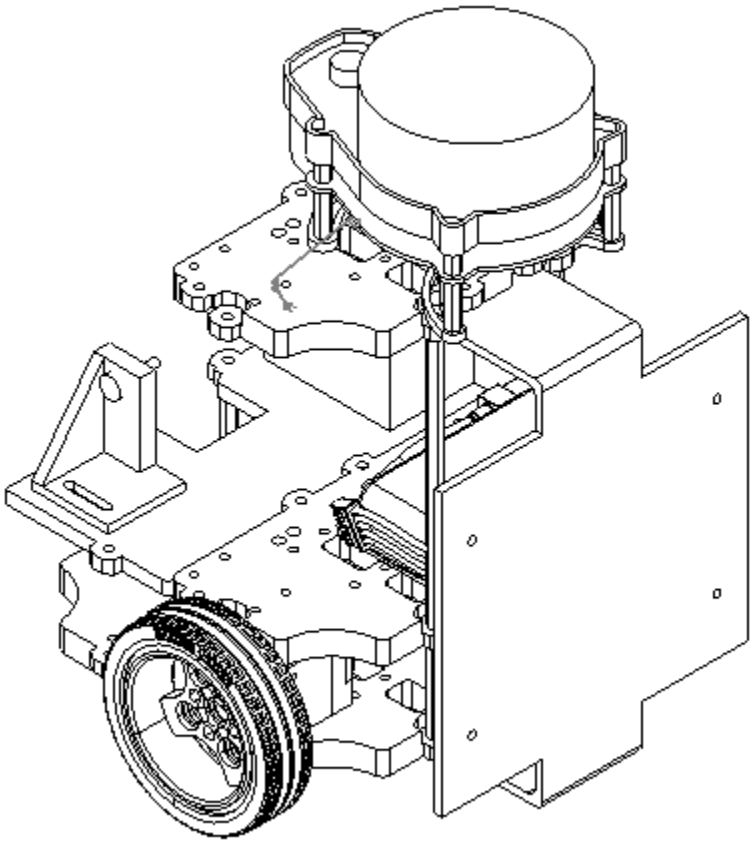
For the majority of operation time, SAM will be in either Pre or Post Target Identification Mode. Considering the capacity of the power supply is 2200 mAh, and the average draw is 16.3 W, SAM should be able to comfortably operate for over 1 hour, and continue on to 1.5 hours if necessary. However, SAM should be charged past 1 hour, to ensure safe operation.

2. GETTING STARTED

2.1. Assembly of SAM

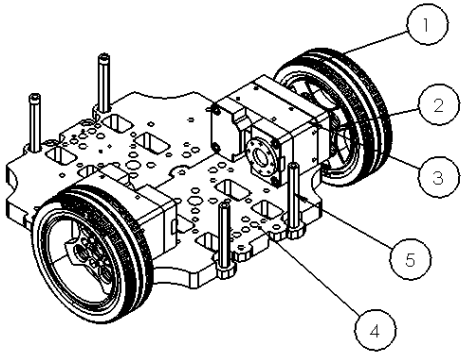
Mechanical Assembly:

Assembly of Chassis



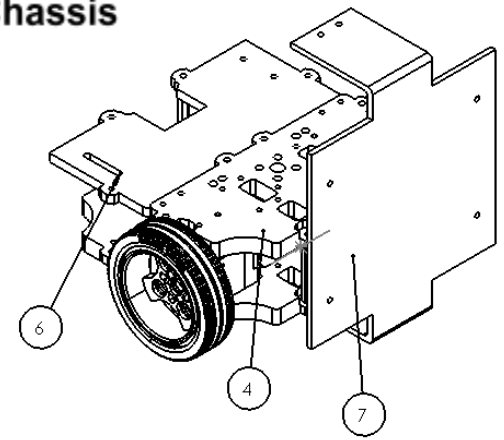
Assembly 1: Chassis

1



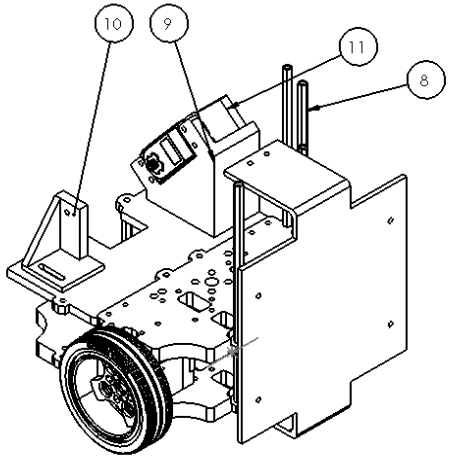
ITEM NO	PART NUMBER	QUANTITY
1	TIRES	2
2	WHEELS	2
3	XM430-W350-R	2
4	WAFFLE PLATE	2
5	BEAM (45)	4

2



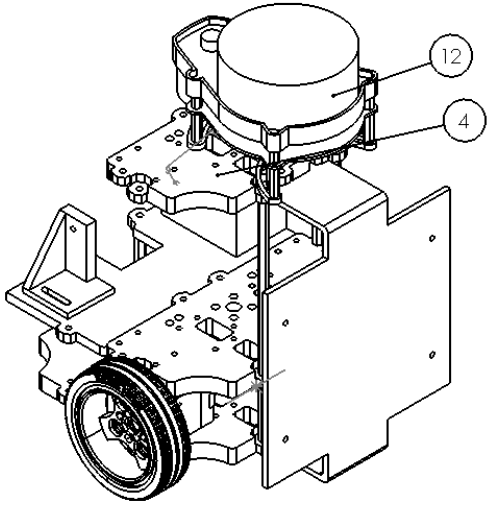
ITEM NO	PART NUMBER	QUANTITY
4	WAFFLE PLATE	1
6	ACRYLIC WAFFLE	1
7	BACK PLATE	1

3



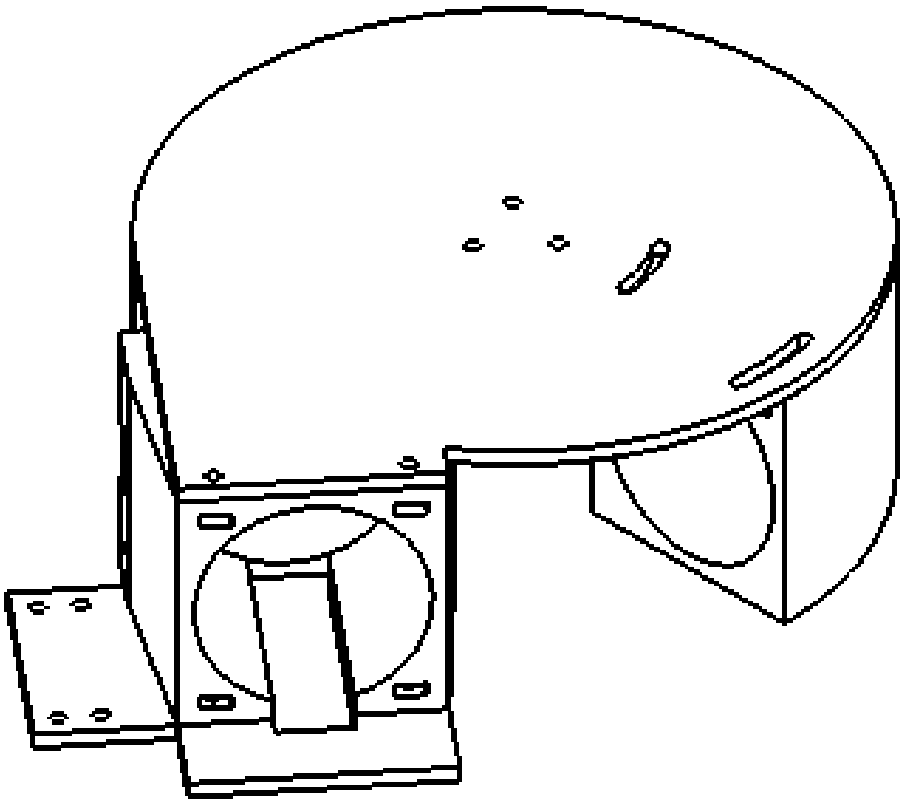
ITEM NO	PART NUMBER	QUANTITY
8	BEAM (100)	4
9	SERVO MOUNT	1
10	SUPPORT	1
11	TD8120-MG	1

4

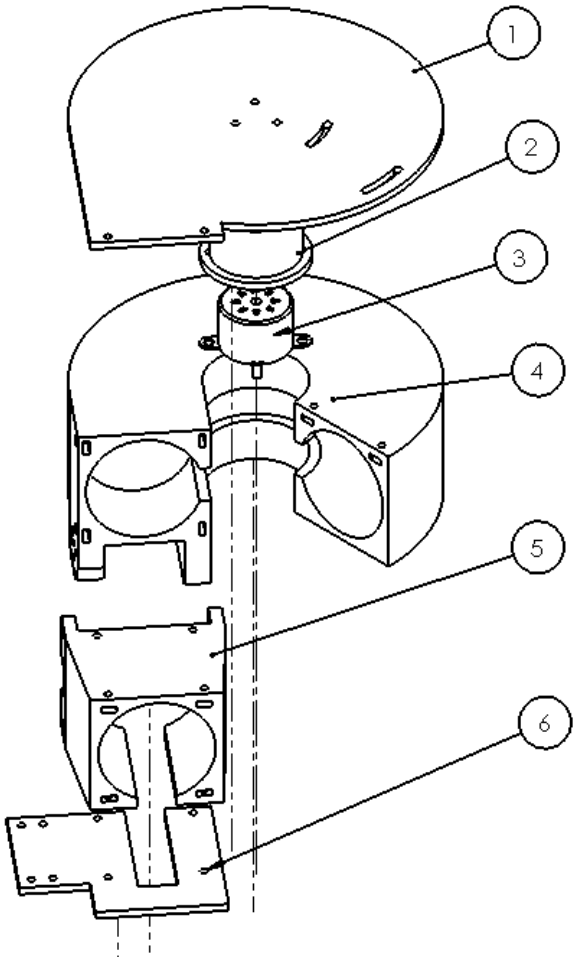


ITEM NO	PART NUMBER	QUANTITY
4	WAFFLE PLATE	1
12	LIDAR	1

Assembly of Payload Carrier

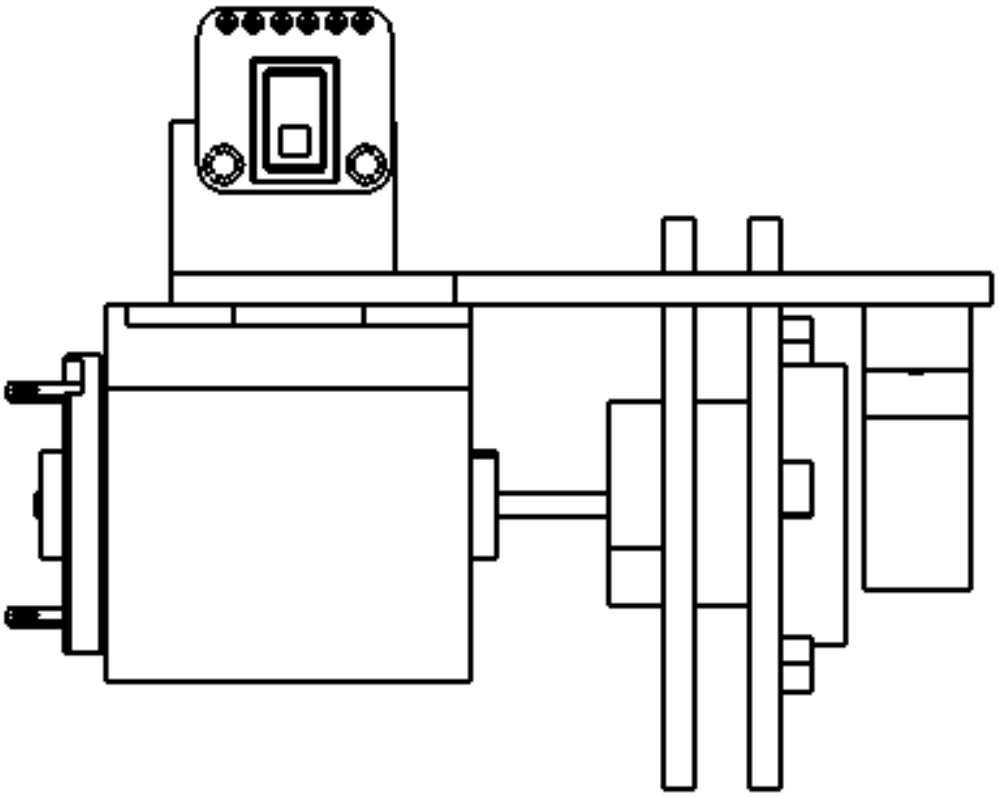


ASSEMBLY 2: PAYLOAD MECHANISM

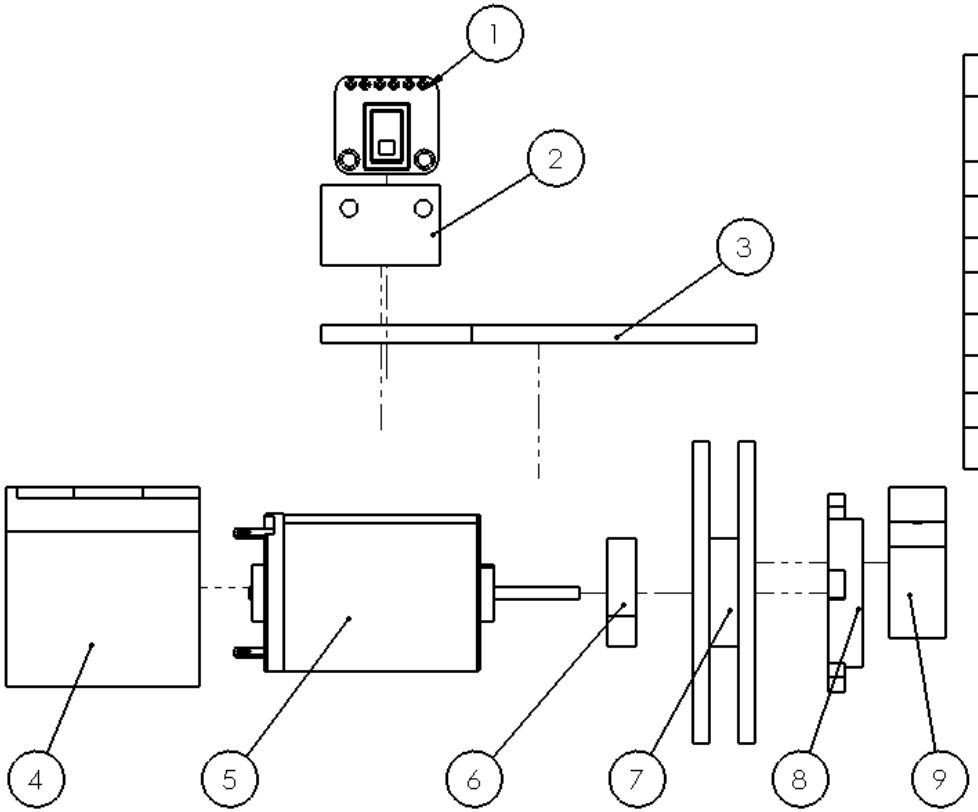


ITEM NO.	PART NUMBER	QTY.
1	Top Plate	1
2	Servo Motor Holder	1
3	28BYJ-48	1
4	Feeder Tube	1
5	Launching Tube	1
6	Acrylic Plate	1

Assembly of Fly Wheel

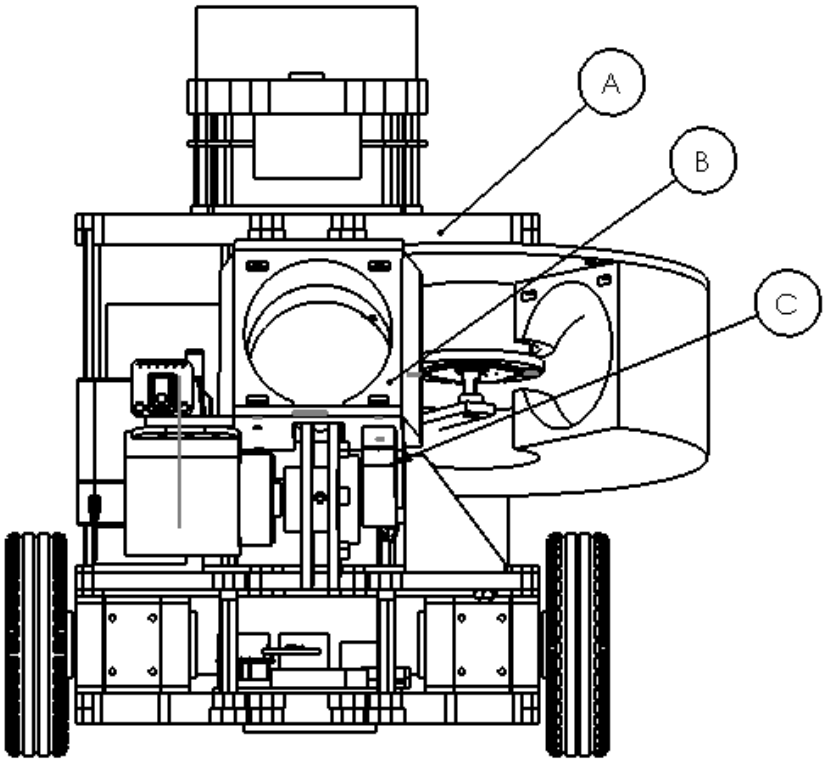


ASSEMBLY 3: FLY WHEEL MECHANISM



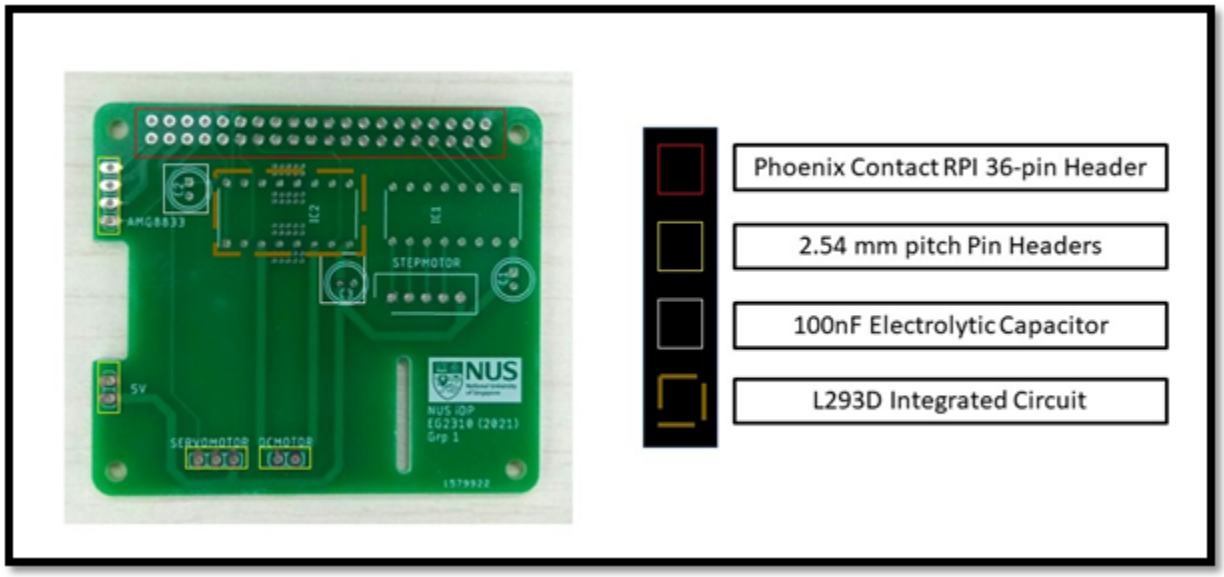
ITEM NO.	PART NUMBER	QTY.
1	AMG8833 IR Camera 8x8	1
2	Camera Mount	1
3	Acrylic Plate	1
4	MotorMount	1
5	RC Pro Motor	1
6	Mounting Hub	1
7	Fly Wheel	1
8	Shaft Connector	1
9	Pillow Bearing	1

ASSEMBLY OF SAM

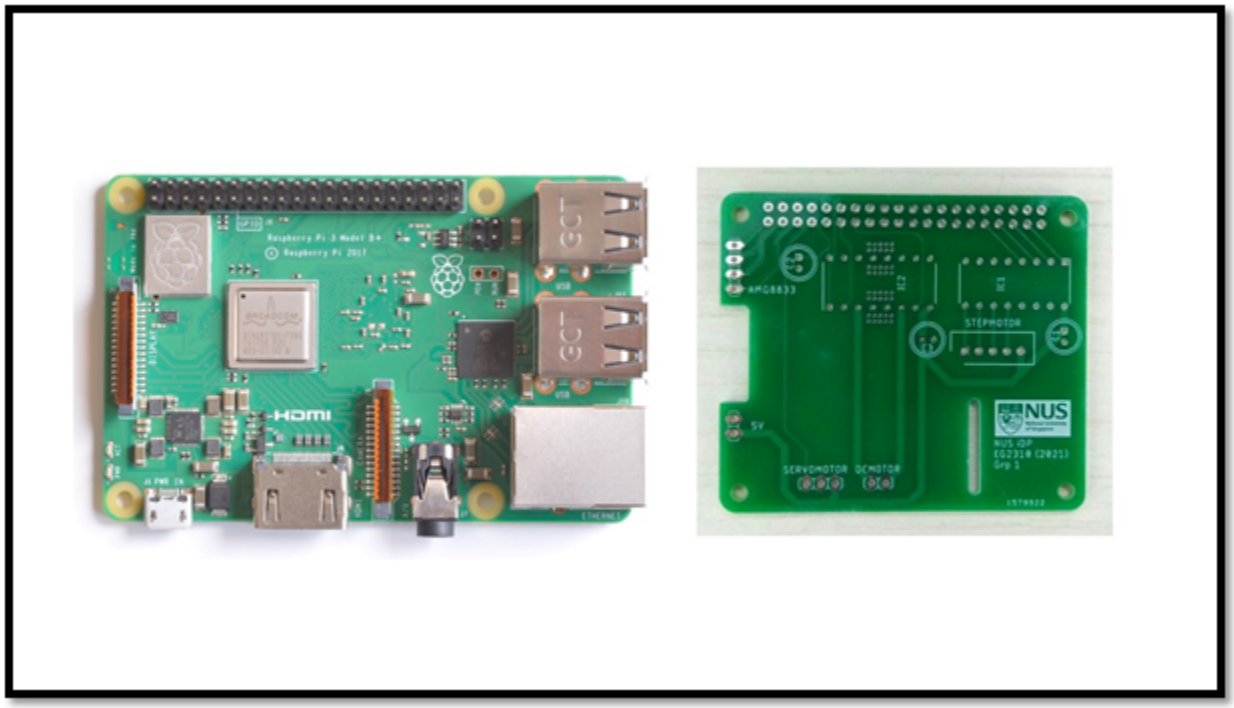


ITEM NO	PART NUMBER
A	CHASSIS
B	FIRING MECHANISM
C	FLY WHEEL

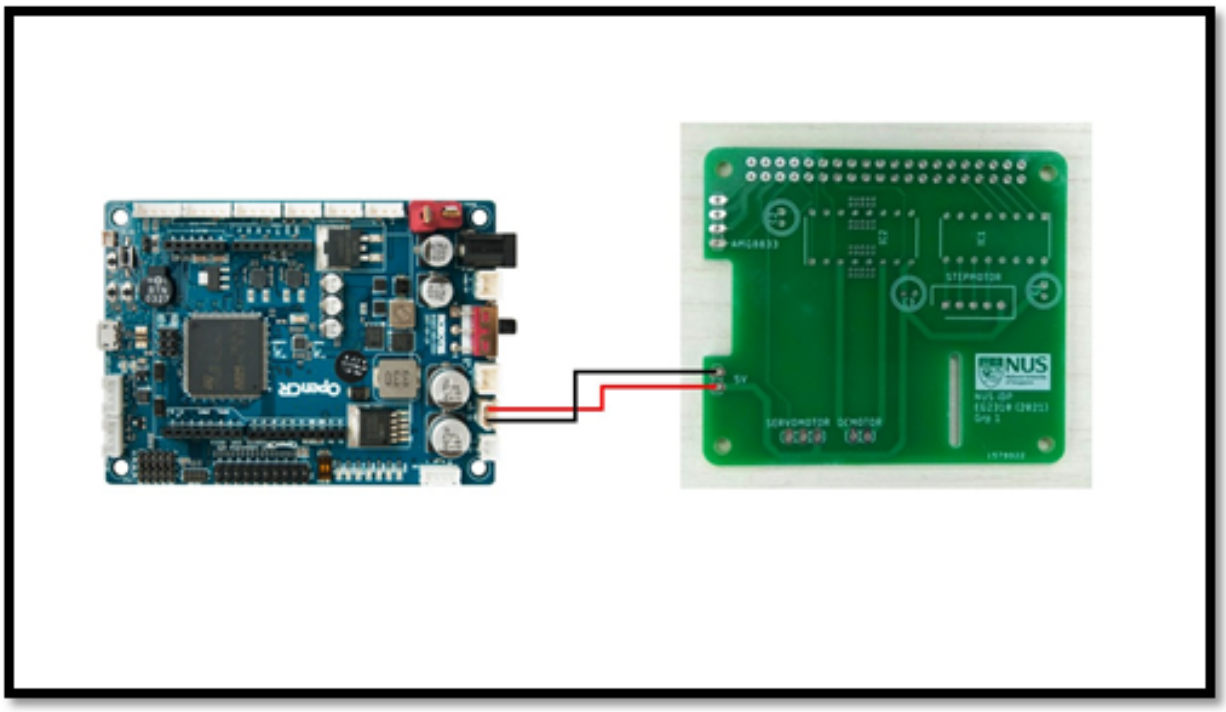
Electronics Assembly:



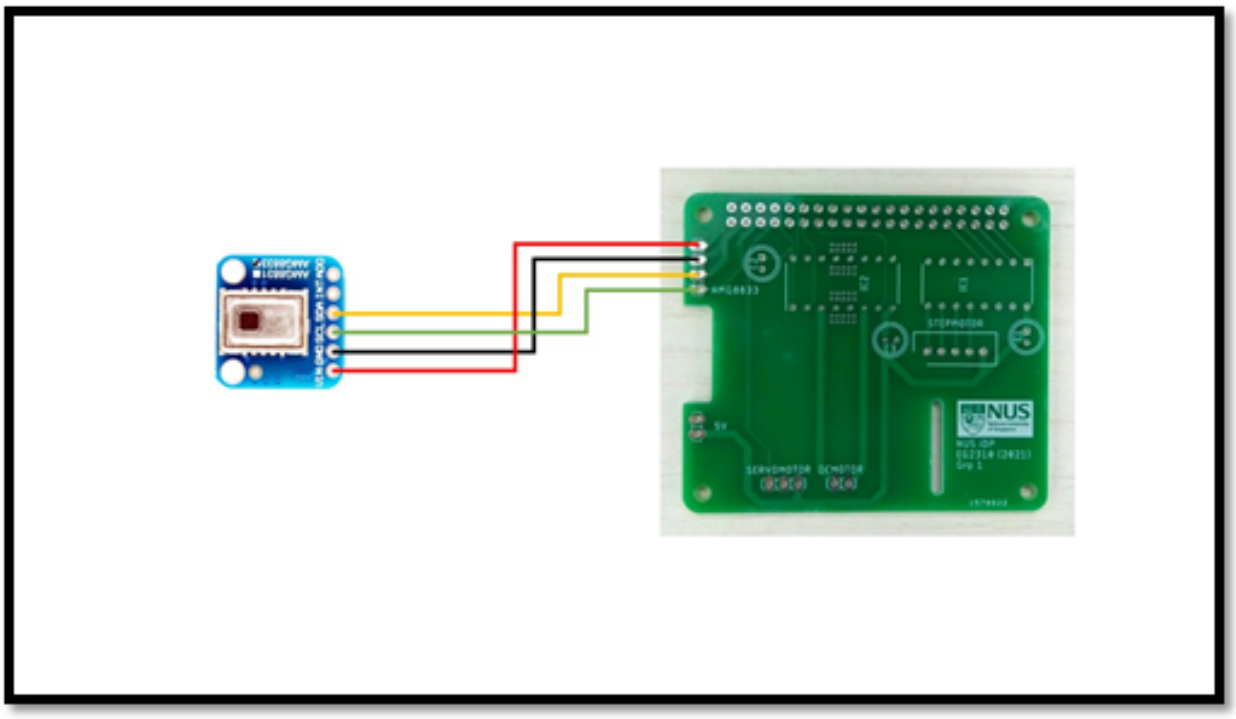
The above parts are soldered onto the circuit board (the Raspberry Pi Hat).



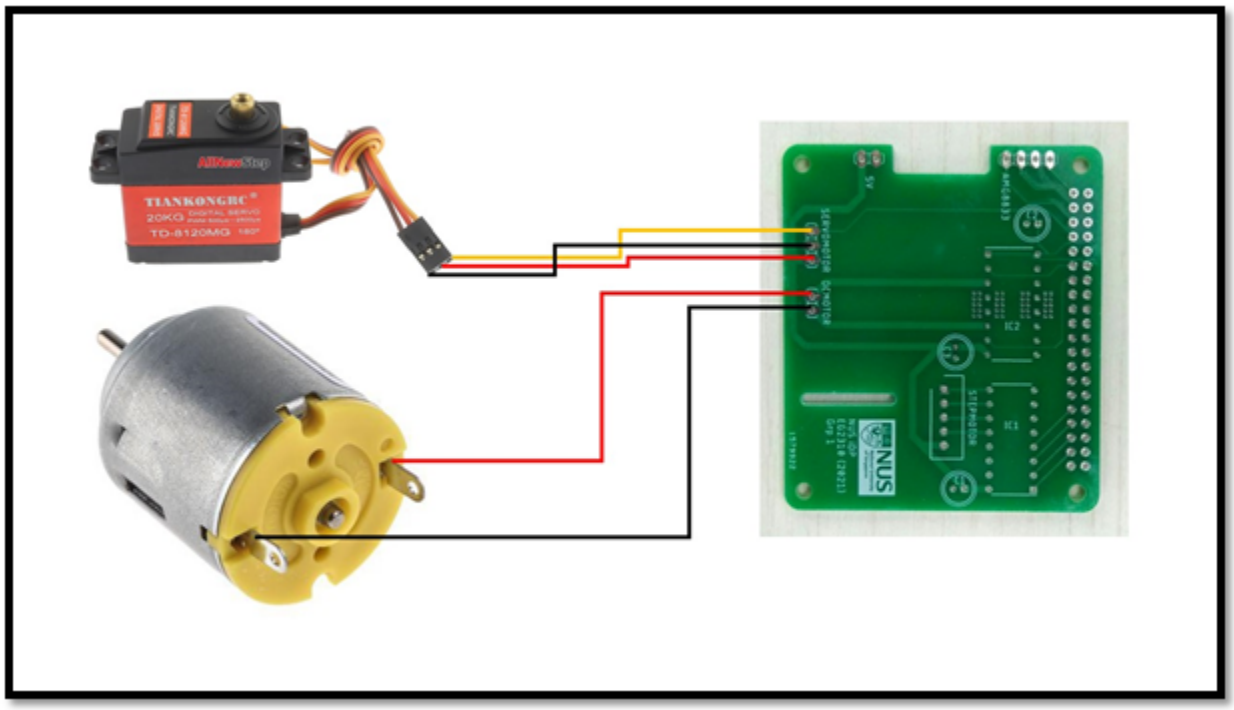
The hat is inserted on top of the Raspberry Pi.



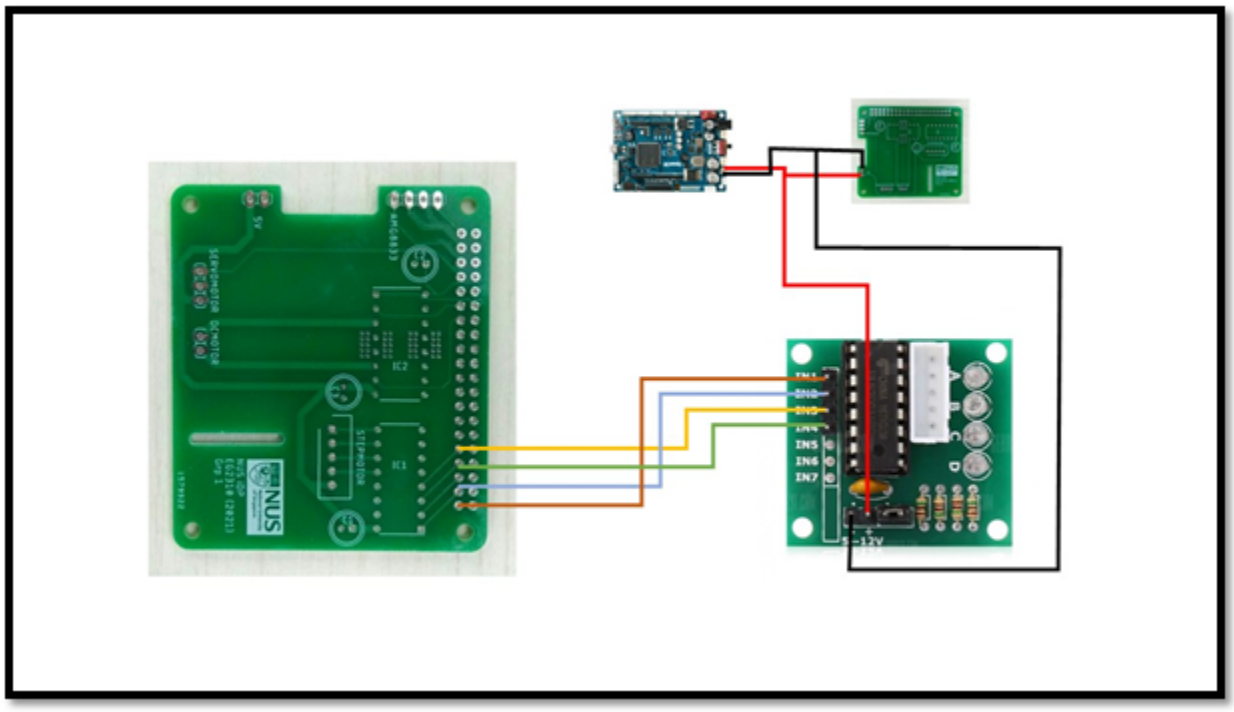
The 5V Supply from the OpenCR is connected to the 5V Input on the Raspberry Pi Hat using two female-female wires. The OpenCR is powered by the 11.1 V, 2200 mAh Li-Po Battery.



The AMG8833 is connected to the Raspberry Pi Hat using 4 female-female wires.



The Servo Motor is connected to the Raspberry Pi Hat using 3 female-male wires. The DC Motor is connected to the Raspberry Pi Hat using 2 female-male wires.



The ULN2003 Stepper Motor Driver is connected to the Raspberry Pi Hat using 4 female-female wires. The 5V Power Supply and GND is split from the OpenCR to the ULN2003 Stepper Motor Driver powering the 28BYJ-48 Stepper Motor.

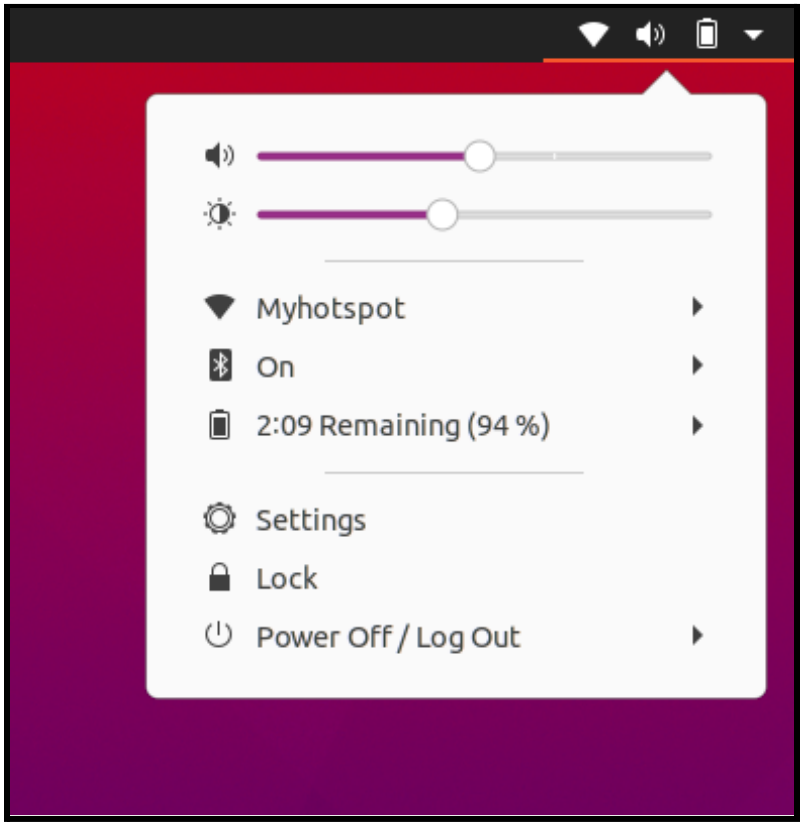
2.2. Powering Up

To power up SAM, simply ensure the battery is connected to the OpenCR and flick the OpenCR on/off switch. All systems will automatically turn on.

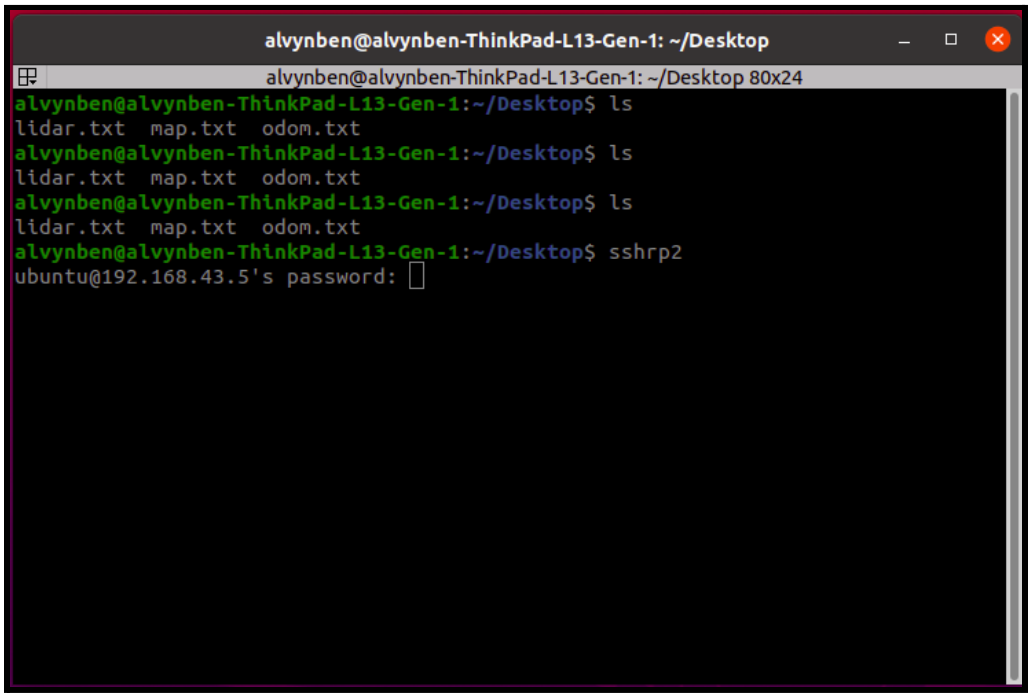
2.3. Connecting to Product

Note: Ensure that you have a way to connect wirelessly to the RPI before proceeding with the next few steps. If you do not, do take a look at the instructions for “Connecting to RPi” in the Software Instructions Folder.

1. Ensure that your laptop is connected to the same hotspot that is connected to the RPi.



2. Connect to the RPi using sshrp2.



2.4. Payload Loading Procedure

Insert the ping-pong ball gently, ensuring that the ball is far enough inside that it is separated from the flywheel. The image below shows how the ball should be inserted into the opening of the launcher.

2.5 Software

Github link for the navigation and targeting code: https://github.com/alvynben/r2auto_nav

Steps to set up the auto_nav package in your laptop:

1. Create a ROS2 package:

```
cd ~/colcon_ws/src
ros2 pkg create --build-type ament_python auto_nav
cd auto_nav/auto_nav
```

2. Move the file in the directory temporarily to the parent directory:

```
mv __init__.py ..
```

3. Now you are ready to clone the GitHub repository to your laptop (make sure you do not miss out on the period at the end):

```
git clone git@github.com:alvynben/r2auto_nav.git .
```

4. Move the file init.py back:

```
mv ../__init__.py .
```

Steps to set up the targeting code in the RPi on SAM:

1. ssh to connect to the RPi
2. Do the following on the RPi:

```
cd ~/turtlebot3_ws/src
ros2 pkg create --build-type ament_python py_pubsub
cd py_pubsub/py_pubsub
wget
https://raw.githubusercontent.com/ros2/examples/master/rclpy/topics/minimal_publish
er/examples_rclpy_minimal_publisher/publisher_member_function.py
cd ..
```

3. Edit the package.xml file in the RPi to insert the highlighted lines below:

```
<export>
        <build_type>ament_python</build_type>
         <exec_depend>rclpy</exec_depend>
         <exec_depend>std_msgs</exec_depend>
</export>
```

4. Edit the setup.py in the RPi to insert the highlighted lines below:

```
entry_points={
        'console_scripts': [
                'talker = py_pubsub.publisher_member_function:main',
                'target = py_pubsub.r2targeting:main',

        ],
},
```

5. Copy the content of the file ‘r2targeting.py’ from the following directory in your laptop:

~/colcon_ws/src/auto_nav/auto_nav

into

the following directory in the RPi on SAM:
~/turtlebot3_ws/src/py_pubsub/py_pubsub

Name the file in the RPi on SAM ‘r2targeting.py’

6. You can now build the package:

```
cd ~/turtlebot3_ws
colcon build --packages-select py_pubsub
```

3. PRODUCT OPERATION

3.1. Pre-Ops Check

3.1.1. Cable Check

Refer to Section 2.1 and ensure the wirings shown are still securely attached. Doing this check after each run is recommended, as severe and unexpected vibrations could dislodge wires or electrical components. You should also ensure no wires are left dangling on the ground as these may damage the wires.

3.1.2. Software Check

Compile the package using the command :

```
cd ~/colcon_ws && colcon build --symlink-install && source ~/.bashrc
```

If there is any compilation error due to having multiple python versions, try to reinstall all the dependencies for the appropriate python version and compile the package again.

After the package has been successfully compiled:

1. Start gazebo using this command:
   ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

2. Start rviz using this command:
   ros2 launch turtlebot3_cartographer cartographer.launch.py use_sim_time:=True

3. Start the Wall Follower code:
   ros2 run auto_nav r2wallfollower

4. Observe SAM's movement in gazebo to ensure that SAM manages to follow the right wall.

Refer to 3.2.2 to try the targeting code and check whether it is working as expected.

3.1.3. Firing Mechanism Check

The Firing Mechanism Check involves checks on the Servo Motor, DC Motor, and Stepper Motor:

Component	Terminal Command	Expected Outcome
TD-8120 MG Servo Motor	*$ sudo python3 test_servo.py**	The servo motor should spin intermittently.
RS Pro DC Motor	*$ sudo python3 test_motor.py**	The DC motor should start spinning.
28BYJ-48 Stepper Motor	*$ sudo python3 test_stepper.py**	The stepper motor should spin through 180 degrees of rotation.
AMG8833 Thermal Camera	*$ sudo python3 amg88xx_simpletest.py**	An 8x8 array of integers roughly representing the temperature in front of the camera should be displayed.

* NOTE: The files above are located in the Software Instructions folder

3.1.4. Calibration

Calibration of AMG8833

To calibrate the AMG8833 to sense the target, it needs to be calibrated so that it recognises the correct temperature.

Inside r2targeting.py (located in Software Instructions), edit the bolded constants based on the calibrations:

```
# constants ---------------------- #
rotatechange = 0.1
speedchange = 0.05
detecting_threshold = 32.0
firing_threshold = 35.0
message_sent = 'Not Detected'
# ------------------------------------ #
```

Change the *detecting_threshold* to the temperature at which it recognises an obstacle as a target. Change the *firing_threshold* to the temperature at which SAM should shoot at the target.

3.2. Laptop Controls

3.2.1. Starting Navigation and Mapping & Targeting and Firing

Specify the initial_direction inside the file r2wallfollower.py. This will determine the direction of the initial move of SAM during which it will find the appropriate wall to follow. SAM will follow the wall on its right.

```
initial_direction = "Front" # "Front", "Left", "Right", "Back"
```

Steps to start the Navigation and Mapping & Targeting and Firing of SAM:

1. Type 'sshrp2' in your laptop's terminal to connect your laptop to SAM's RPi
2. In SAM's RPi, start rosbu by typing 'rosbu'
3. On your laptop, start rslam by typing:
   ros2 launch turtlebot3_cartographer cartographer.launch.py
4. In SAM's RPi, start the targeting code by typing:
   ros2 run py_pubsub target
5. On your laptop, start the navigation code by typing:
   ros2 run auto_nav r2wallfollower

4. BATTERY CHARGING

4.1. SAM's Li-Po Battery

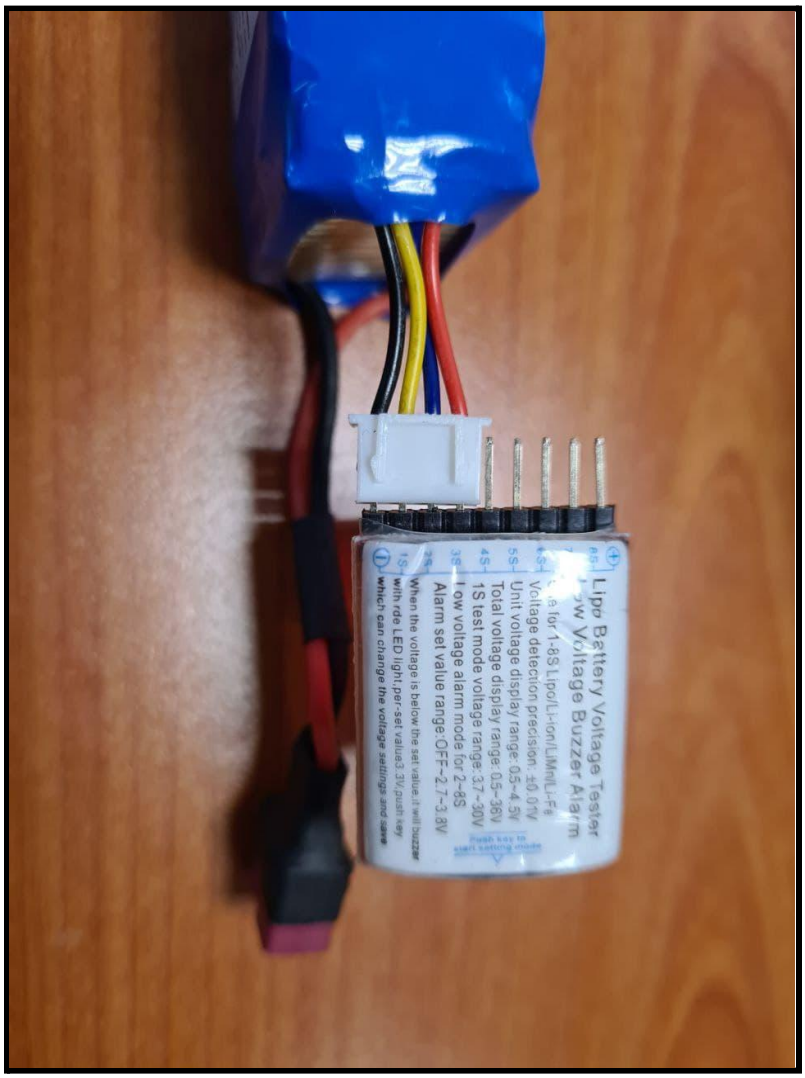
4.1.1. Charging Battery

1) Connect the Li-Po battery's charging head into the charger's port as shown in the photo below.

2) Connect the charger to a wall outlet using the black cable as shown below. If the red LED light turns on, the Li-Po battery is being charged. When the battery is fully charged, the green LED light will turn on.

4.1.2. Checking Battery Level

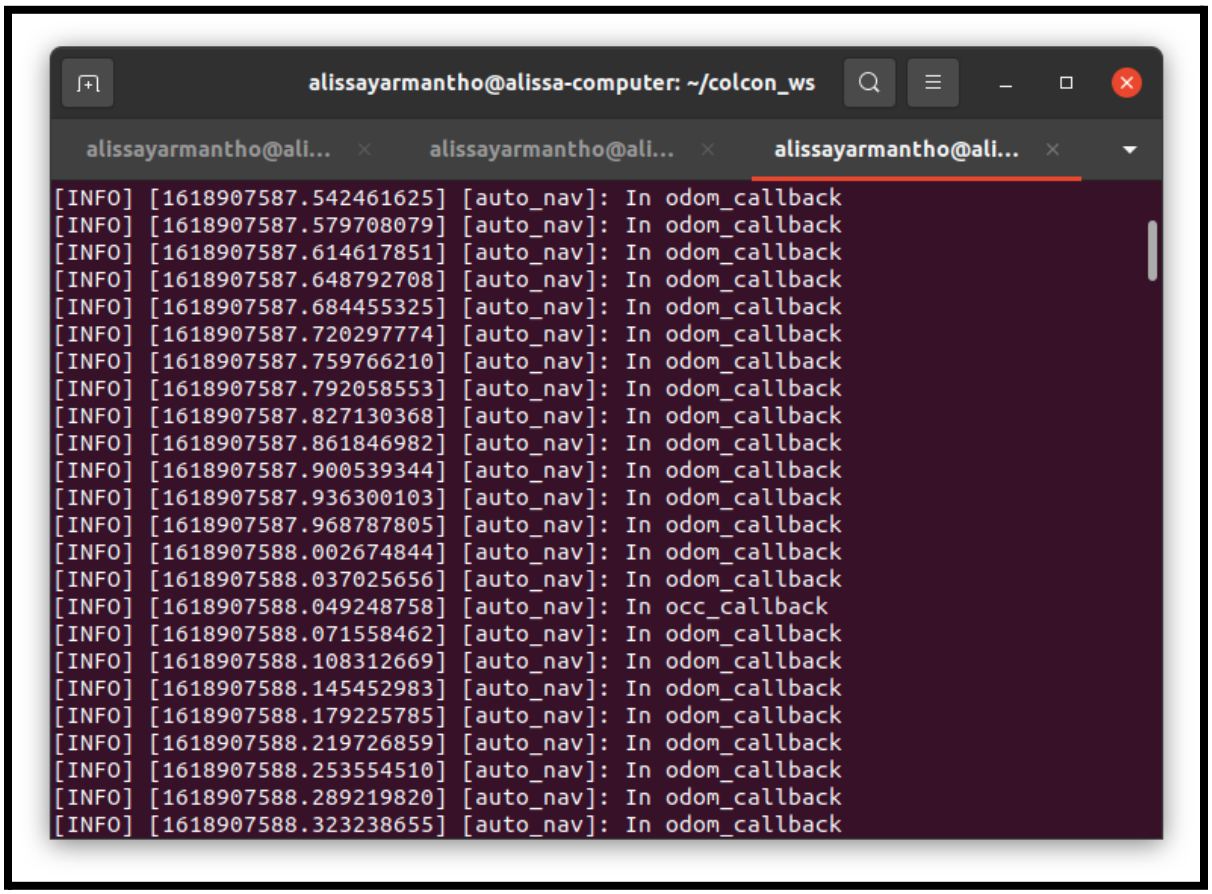
The battery voltage level can be checked using the Li-Po Battery Voltage Tester. Connect the battery to the tester starting from the negative pin as shown below. The tester will indicate the voltage level of the battery.



5. TROUBLESHOOTING

5.1. Error Messages

5.1.1. Navigation and Mapping



Common troubleshooting steps to solve infinite loop without ever going to the wall following logic error:

1. Stop the code with Ctrl + c and run it again.
2. Check if the lidar is connected properly to the raspberry pi.
3. Restart rosbu.
4. Turn SAM off and turn SAM on. Then restart rosbu.

5.1.2. Targeting and Firing

Common troubleshooting steps to solve the problem of not finding the target / target is found but SAM does not fire:

1. Stop the code with Ctrl + c and run it again.
2. Check target temperature calibration for target detection and firing threshold. Then run targeting code again.
3. Check all motor connections and try running the code again.

6. MORE INFORMATION

SAM's AutoNav Repo	SAM's Video Demo

First Edition: April 20, 2021
Revision 4: April 23, 2021

Published by: EG2310 Group 1
Address: Engineering Design & Innovation Centre
Block E2A #04-05
5 Engineering Drive 2
Singapore 117579