PDR - 3D-ROMAP

R Dominick, L Gogley, M Munoz, D Tran

System Overview

- Project goal is to design and implement a robotic ground unit to capture depth and image data.

- The project is intended for research, and targets exploration, such as indoor / outdoor mapping.

- System is divided into three categories:
    - Positional data recording
    - 3D image capture and processing
    - Robotic assembly and control

Top Level Block Diagram

Processor Board - **NVIDIA Jetson**

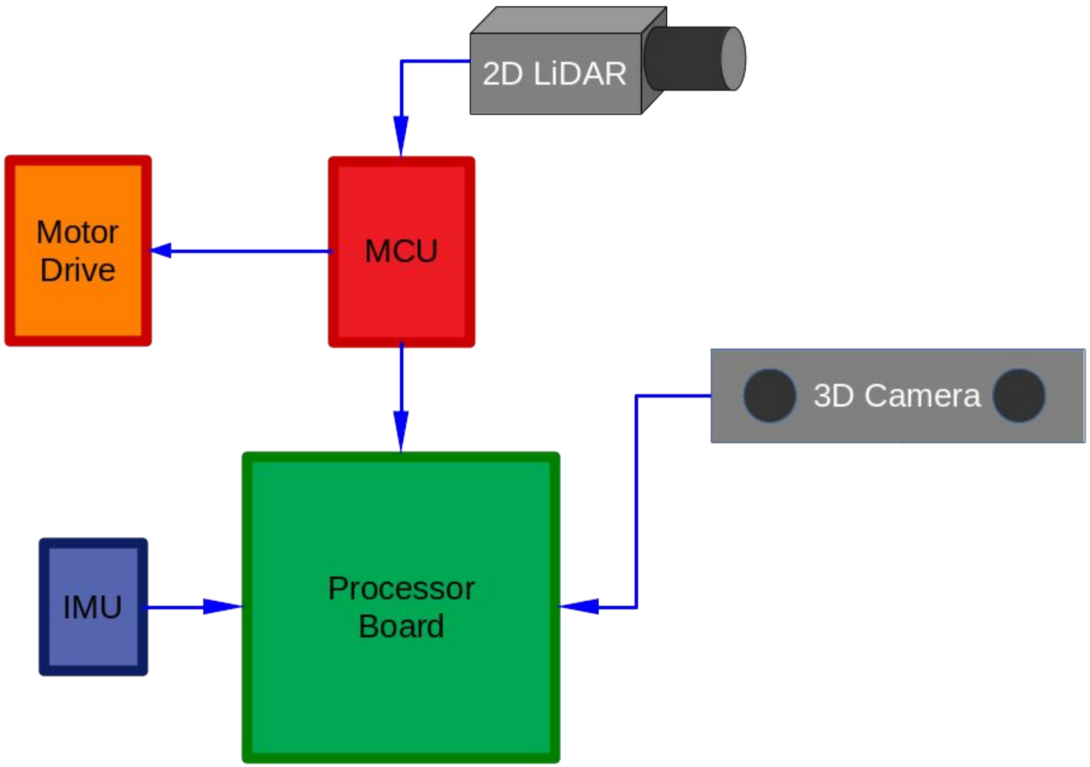
MCU - **Tiva C Launchpad**

3D Camera - **Intel Realsense**

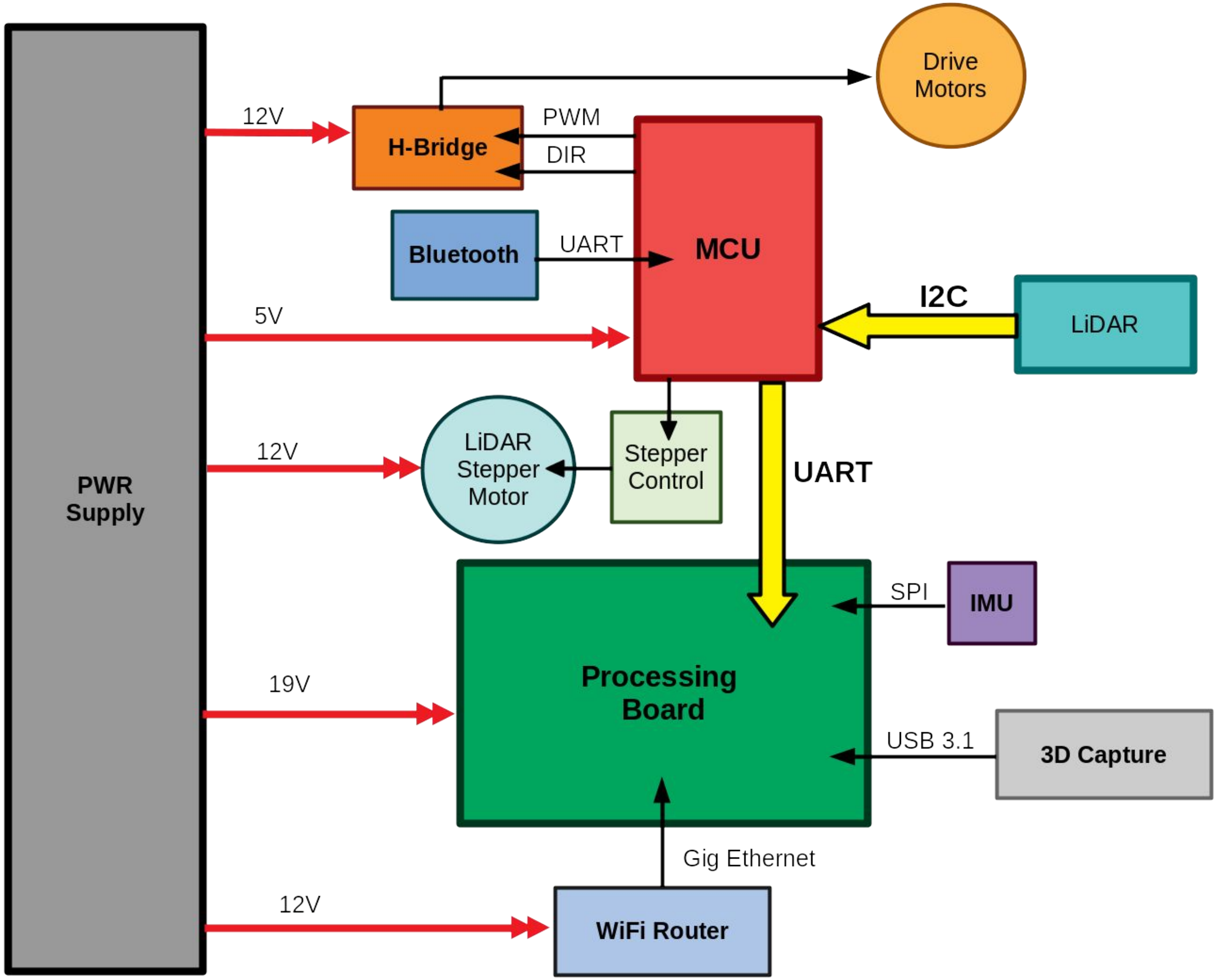
2D LiDAR - **LiDAR Lite v1**

IMU - **9DoF IMU**

Motor Driver - **Stepper & H-Bridge**



Detailed Block Diagram



Requirements

- **The design shall map unknown coordinates through exploration on passable terrain.**
  - Location of choice should not dictate mapping method, i.e. a room vs a field.
- **The design's structural capture shall be capable of producing a visual map.**
  - The scans must produce an image with the desired map.
- **The design shall be capable of 2D SLAM.**
  - SLAM shall be used to handle 2D coordinate recognition, and assist in 3D sampling.
- **The design shall be capable of 3D capture.**
  - 3D capture can be pulled from samples or from real-time/soft real-time capture.
- **The design shall be powered from an independent power supply.**
  - This power source shall provide enough power to explore, at minimum, 20 minutes of capture.

Limitations

- **The design is not confined to low light, visible image capture.**
  - High resolution 3D structural capture is used to compensate for loss of visuals under low light.
- **The design is not confined to autonomous navigation, simply the capability of such.**
  - The design will be capable of mapping and localization, but the platform may not drive autonomously as functionality of mapping is the first priority.
- **The design is not confined to real-time 3D mapping.**
  - Real-time 3D is sometimes limited to ability of hardware readout, and may be an unrealistic endeavor. 3D may instead be done from sampling and post-processing.
- **The design is not interested in object recognition.**
  - Point clouds will be used to simply display captured “maps”, and points on a 2D plane will only be used to determine the explored area.

Power Source

Runtime = (10 * Ampere Hours) / Load in Watts

- **24V** - Two 12v (7Ah) batteries
- **0.57 Hours** = (10*7) / 122.1

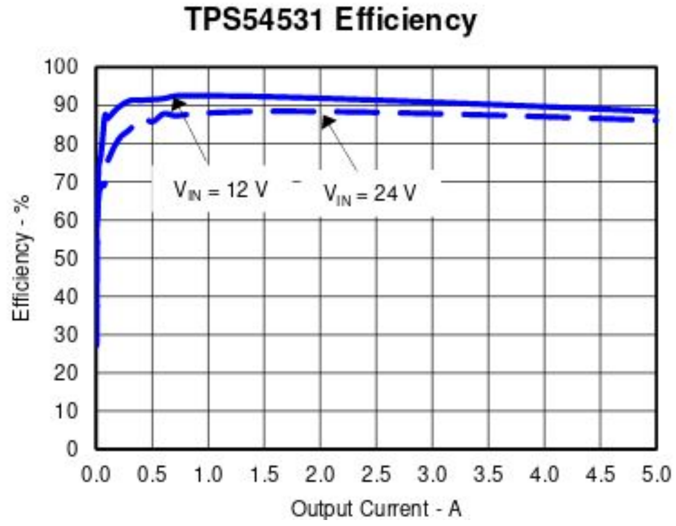
VOLTAGE	CURRENT	PWR 122.1
5	1.22	6.1
12	8.40	100.8
19	0.80	15.2

Voltage Requirements

DC-DC Buck Converter

TI - TPS54531

- 5A continuous current
- High efficiency at light loads
- 570KHz switching frequency



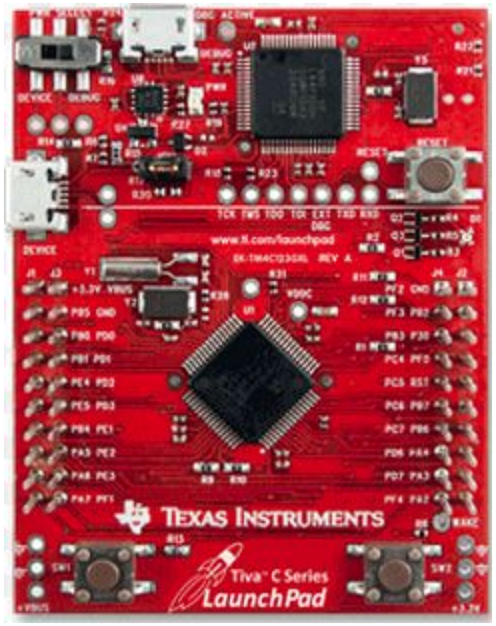
DEVICE (Amps)	5v	12v	19v
TM4C	0.50		
Jetson TX1			0.80
Realsense (TX1 USB)	0.70		
Stepper Motor		0.40	
LiDAR Lite	0.02		
Drive Motors		8.00	

Positional Mapping

Microcontroller

TI Tiva C Launchpad

- TM4C123GH6PM MCU:
  - 80MHz 32-bit ARM Cortex M4
  - 256KB Flash, 32KB SRAM, 2KB EEPROM
  - Two Controller Area Network (CAN) modules
  - USB 2.0 Host/Device/OTG + PHY
  - Dual 12-bit 2MSPS ADCs, motion control PWMs
  - 8 UART, 6 I2C, 4 SPI
- On-board In-Circuit Debug Interface (ICDI)
- USB Micro-B plug to USB-A plug cable



Goal of SLAM

- **Use various sensors to collect data from the environment and provide an accurate estimation of where to robot is in the room at any given time**

- **Keep track of previously maneuvered areas and use this data to create a 2 Dimensional map**

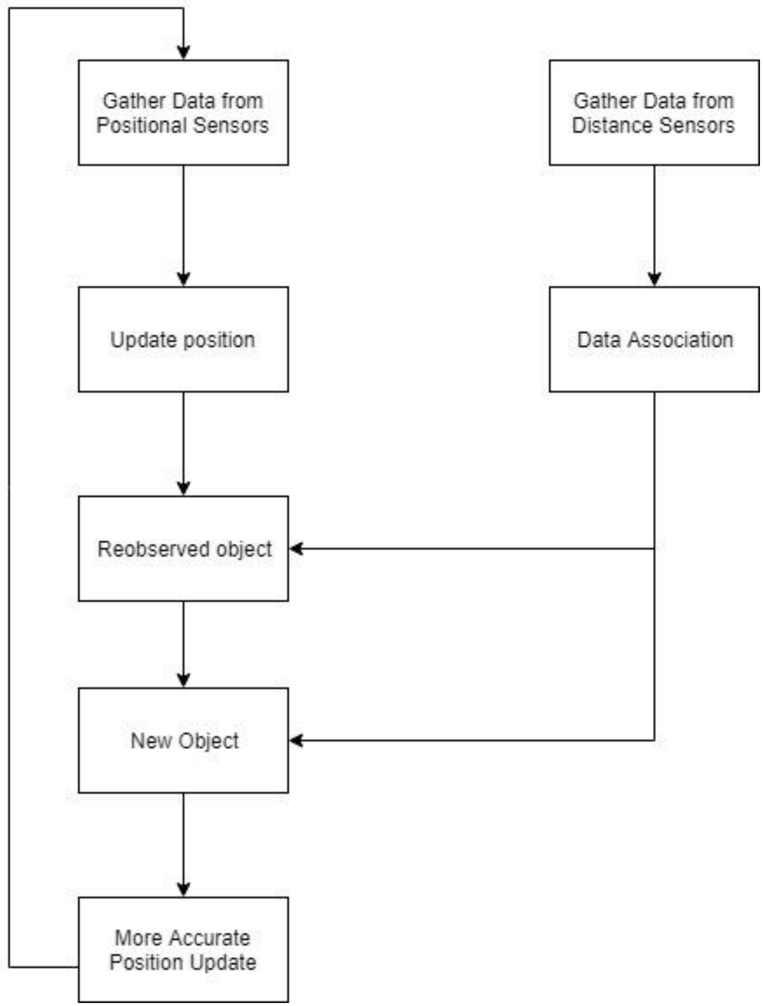
Data Useful to SLAM

- Distance measurement from LiDAR sensor
- Positional change from odometers
- Angular velocity collected from gyro
- Directional acceleration from accelerometer
- Magnetometer essentially serves as a compass

Landmark Extraction

- Landmarks are chosen based on the following criteria:
  - They are easily re-observable
  - Should have enough landmarks so that the robot will never be maneuvering without a visible landmark
  - Should not have so many landmarks the robot has difficulty determining which landmarks have been previously seen
  - Examples include well defined corners, desks, and other objects with well defined edges

Possible SLAM Implementation



- Data association
  - Look at previously seen landmarks and decide if what is currently being scanned matches
  - If matches, give more precise position
  - If no match, add to list of known landmarks

IMU Position Tracking

MPU-9250 - 9DoF IMU

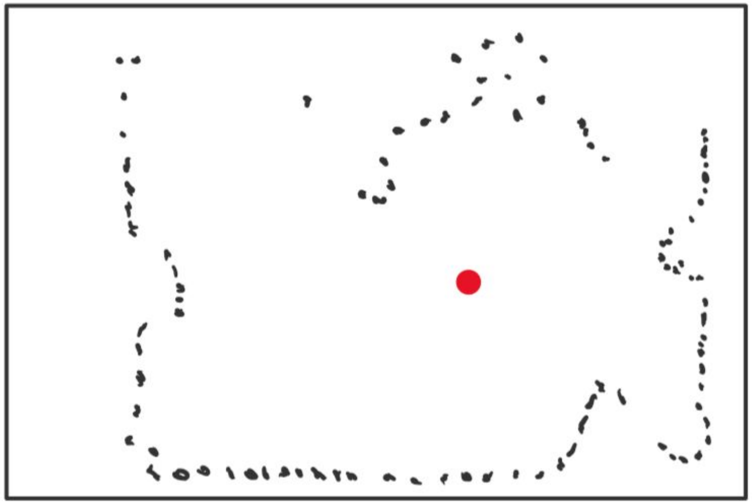
- Accelerometer
  - 3 DoF X, Y, and Z
- Gyro
  - 3 DoF roll, yaw, and pitch
- Magnetometer
  - 3 DoF magnetic field detector

2D Capture

- PulsedLight LiDAR Lite v1
    - Class 1 infrared laser
    - 100Hz sample rate
    - I2C or PWM connection
    - Distance range of 40m
    - Accurate to +/-2.5cm
    - Max current 130mA

LiDAR Sensor - Overview and Purpose

- **The LiDAR Lite v1 Sensor will be utilized to generate a 2D-representation of the environment around the robot**
  - Similar to a top-down of a videogame mini-map
  - Map will contain robot's positional data in the environment
  - Simultaneously map the environment and know its current position in the environment (SLAM)



2D-Mapping Simple Mock-Up

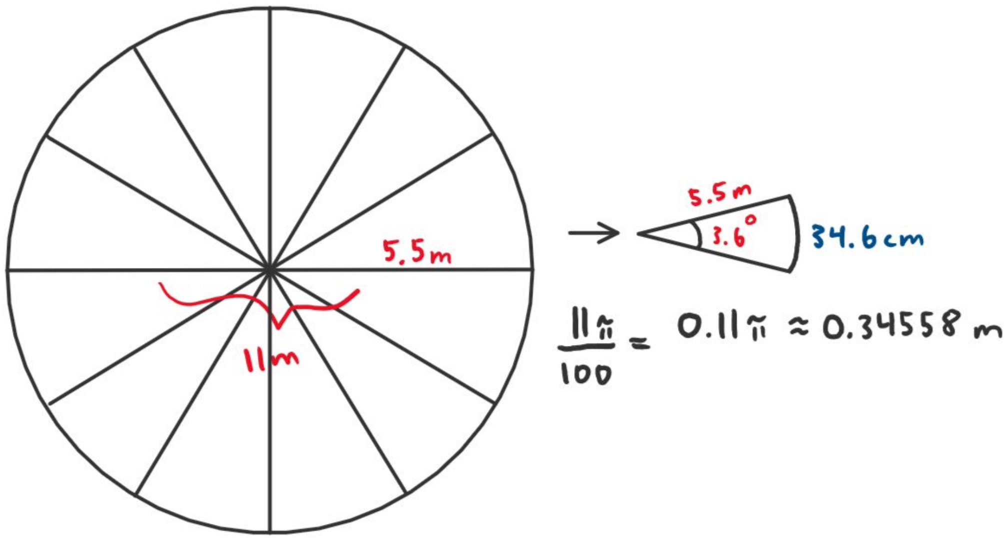
LiDAR Sensor - Capturing Distance Values

- **Interfacing the LiDAR Sensor with the TM4C123 MCU**
  - Utilize the I2C Protocol
  - LiDAR will return a distance value(cm)
  - Feed the data to the NVIDIA Jetson board for processing

- **Use the captured distance reading, process the data, and plot the data points on a 2D cartesian plane**
  - The 2D plane will be the representation of the environment around the robot
  - The robot's location will also be represented as a separate point on the map

LiDAR Sensor - Method of Sampling

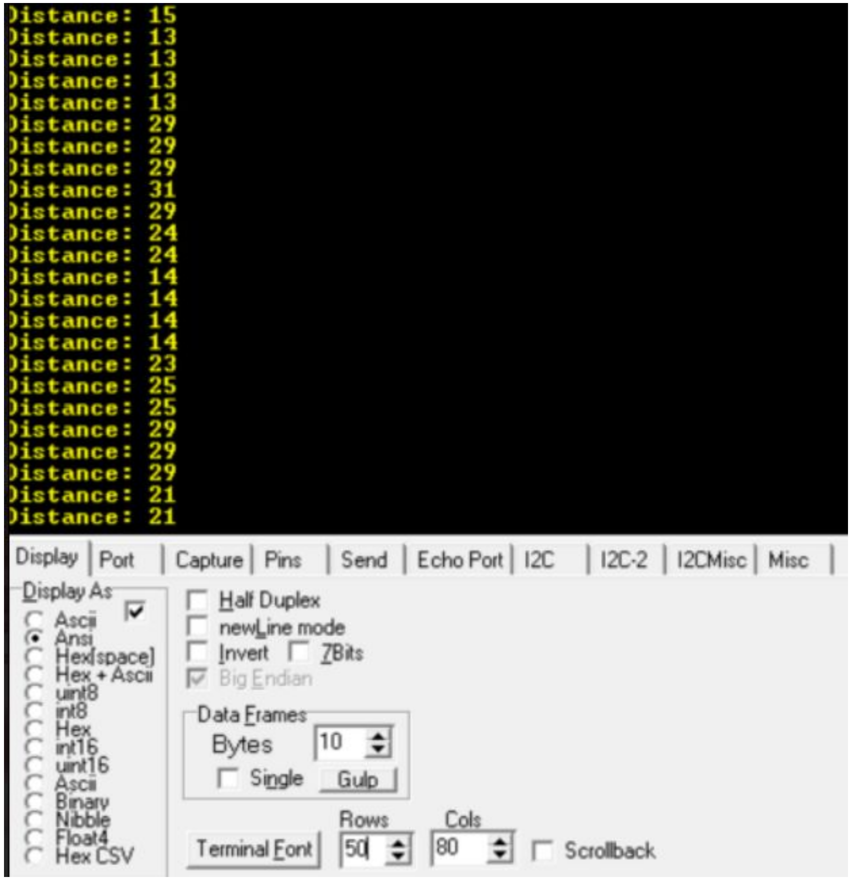
- Rotate the LiDAR on a Stepper Motor
- Assumptions:
  - Rotate the Motor at 1Hz. This will provide 100 sample points in 1 revolution
  - Estimate average radius = 5.5m
  - Circumference = 11π
- $11\pi/100$ samples = <u>34.6 cm</u> between each sample



1 Revolution(1 Hz | 100 Samples)

LiDAR Sensor - Verification

- **The following terminal output showcases the LiDAR sensor successfully interfacing with the TM4C.**
  - **The LiDAR sensor returns data through I2C to the TM4C**
  - **Data is displayed through UART to a serial terminal**



LiDAR Readings Example

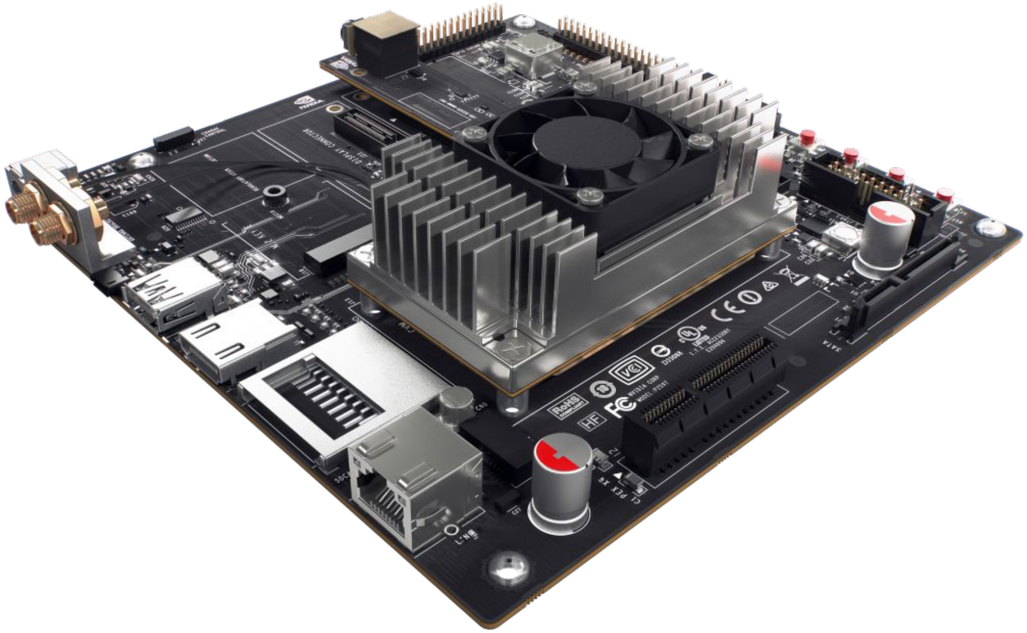
3D Capture

Processing Board

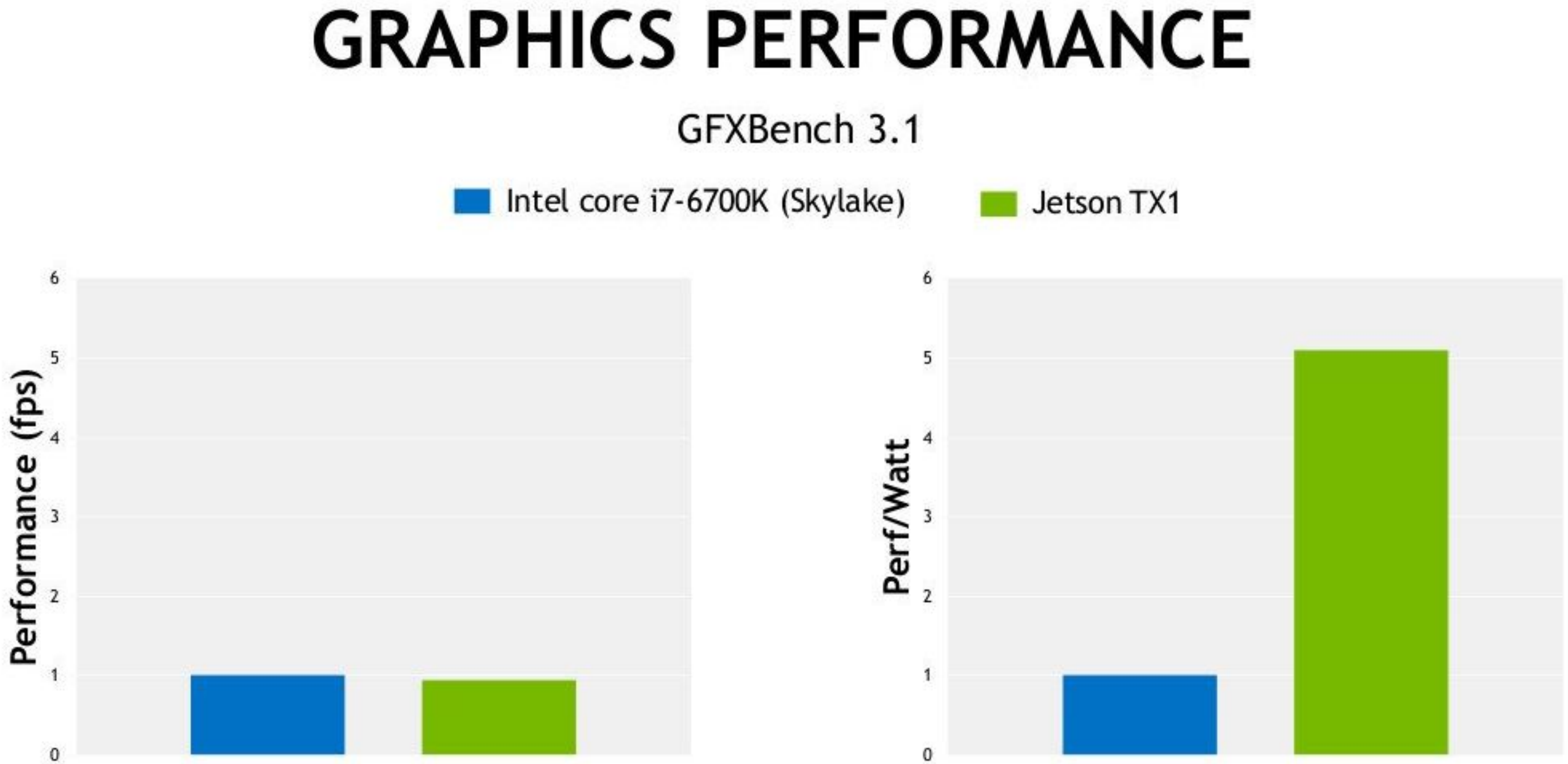
NVIDIA Jetson TX1

- Hardware
  - 64 bit ARM A57
  - 256 CUDA core GPU
  - 4GB RAM
  - 15W power draw

- Peripherals
  - USB 3.0 Port
  - Gigabit Ethernet
  - GPIO header
    - I2C
    - UART
    - SPI
  - SATA port for HDD/SSD

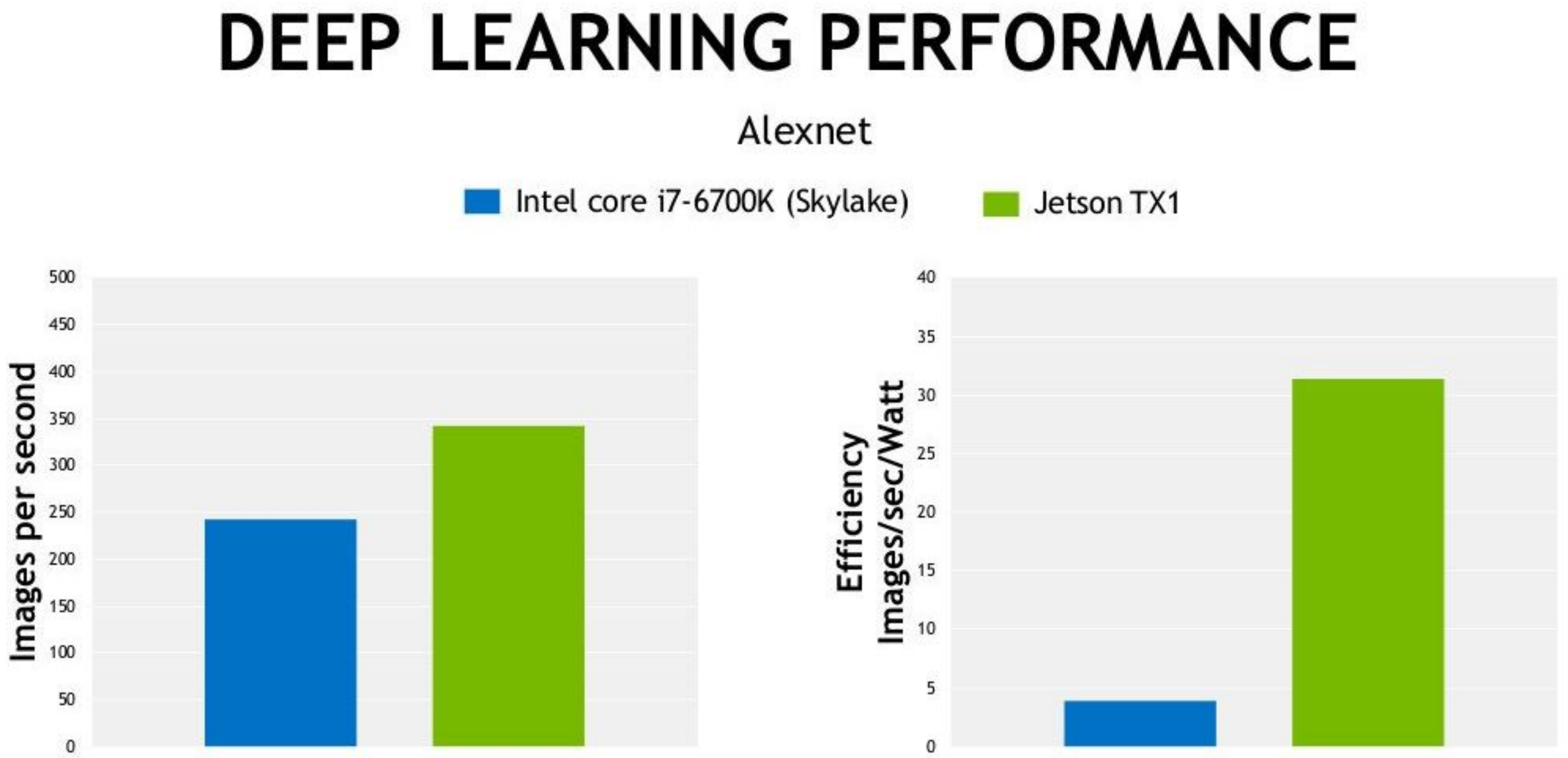


Jetson Power Efficiency



- The graphics performance is even being reported by NVIDIA as being comparable to the i7-6700K's HD Graphics 530 in raw performance or 5x better with performance-per-Watt.

Jetson Computational Efficiency



- The numbers supplied by NVIDIA report that the Jetson TX1 performance even beats out an Intel Core i7 6700K Skylake CPU when it comes to deep learning.

3D Rendering

- PCL - Point Cloud Library
    - Permissive open-source BSD license
    - Depth capture tools and viewer
    - Allows Point Cloud Registration*
    - Intended for 3D mapping and localization
    - Platform independent

*Alignment and stitching of similar clouds

3D Camera

- Intel Realsense D435
  - Allows for image overlay, with point cloud for depth
  - Structured-Light, Active IR stereo vision.
  - RGB Camera: 1920x1080
  - Depth: 1280x720 @ 90 FPS
  - Range: 0.1m to 10m
  - RGB camera FOV: 69.4° x 42.5° x 77°
  - IR FOV: 85.2° x 58°
  - Indoor/Outdoor

Realsense Software

- Realsense SDK
    - Open-source, cross platform
    - Used for depth and image overlay
    - OpenCV wrapper for image processing
    - Creates Realsense point clouds easily
    - Intended for robotics use
    - Seamless integration with PCL

3D Capture - Point Cloud Data (PCD) Format

- PROS
  - PCL (Point Cloud Library) 3D capture file type
  - Open-Sourced
  - Designed for point clouds
  - Self documenting
  - Stores X,Y,Z and RGB data in ASCII matrix or binary
- CONS
  - Uncommon outside PCL

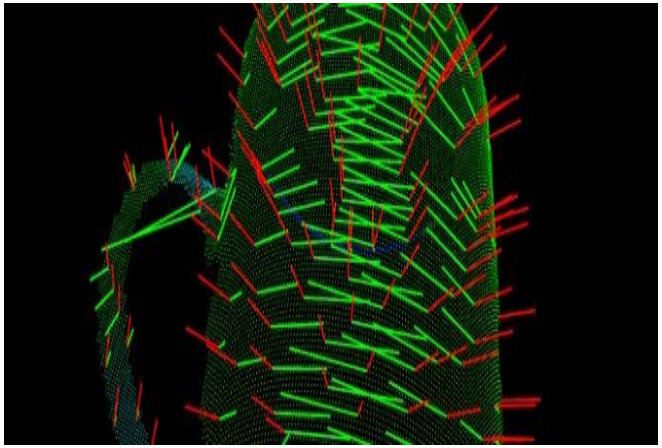
3D Capture - Polygon (PLY) Format

- PROS
  - Polygon 3D capture file
  - Universally supported for 3D viewers
  - Easily converted to PCD, or PCD to PLY
  - Stores X,Y,Z and RGB data in ASCII matrix or binary
- CONS
  - Never intentionally designed for point clouds

Other Notable 3D Capture Formats

Alternate 3D capture file types

- **STL**
  - Pros: Common 3D file type with small file size
  - Cons: No RGB capture
- **OBJ**
  - Pros: Widely accepted file type easily portable to other applications
  - Cons: No RGB capture
- **X3D**
  - Pros: "3D standard for web *[HTML5]*"
  - Cons: Difficulty stitching multiple files



Robot Control / Viewing

Remote Access

- Need to reliably control motor movement wirelessly and from a distance
- Possibly operate device while it is not in direct line of sight of the user (e.g. in a different room)
- Securely connect to device without possible interference from other signals

Bluetooth:

- Bluetooth offers a low power, low cost, fairly secure solution to the motor control
- Operating range varies from under 10m to 100m
- Power consumption ranges from 1mW to 100mW
- HC-05 Bluetooth Module
  - Class 2: 10m operating range, 2.5mW
  - easy to use
  - compatible with many devices, including phones and other physical devices

Remote Viewing

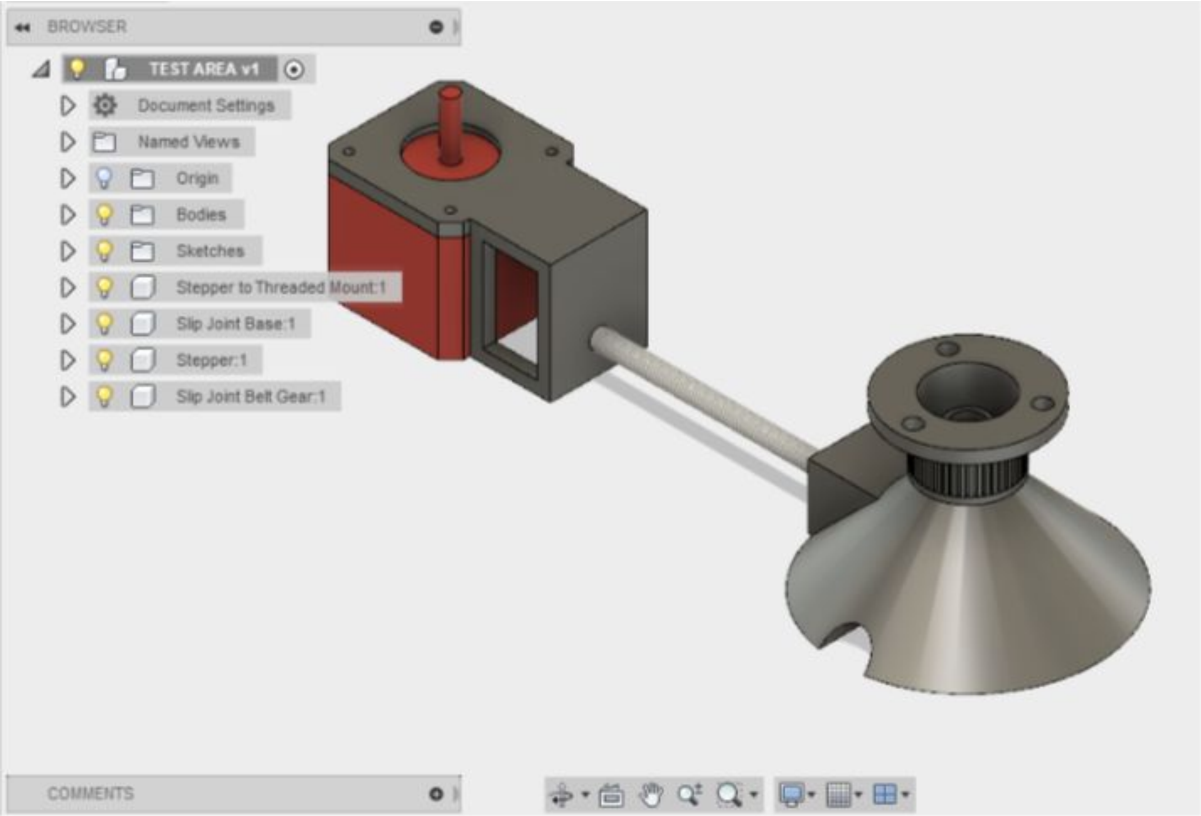
WiFi - 5GHz:

- Avoid interference from 2.4GHz UHF Bluetooth band
- Remotely view the 2D map and 3D
- Accomplished with WiFi router for remote desktop
  - VNC
  - X2Go
  - SSH application to host

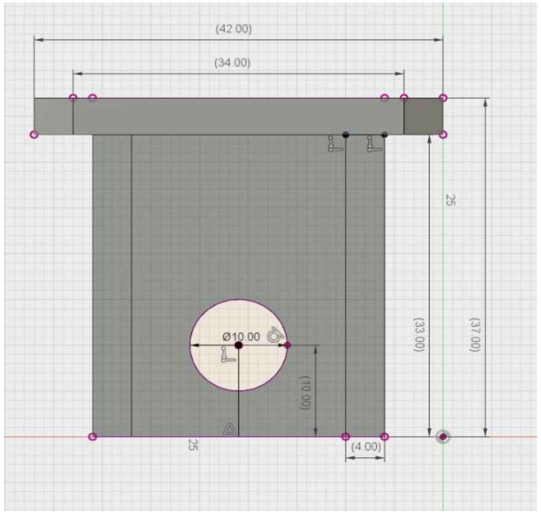
Mechanical Assembly

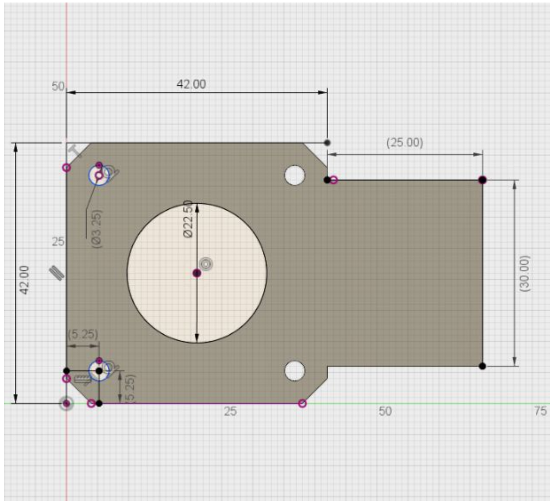
Top Level Motor Assembly

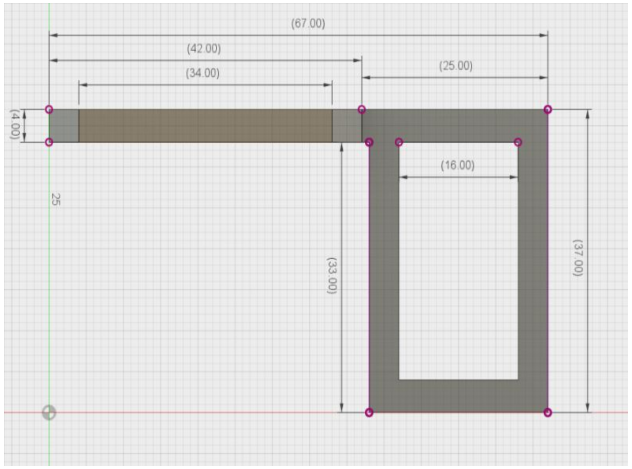
2D capture bay assembly, driven by stepper.



Motor drive support assembly

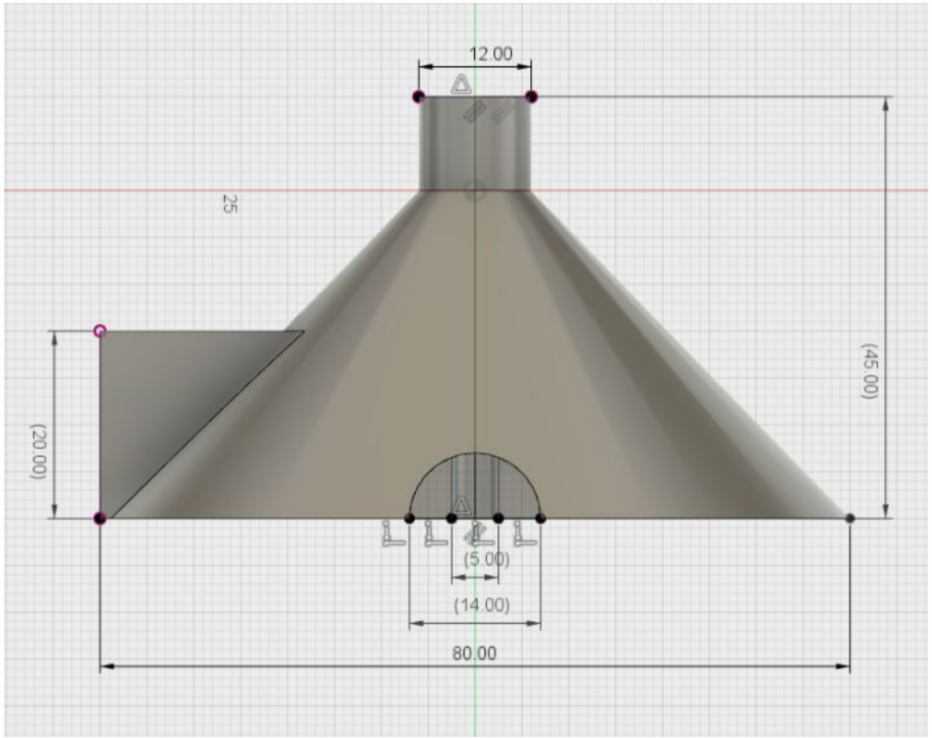


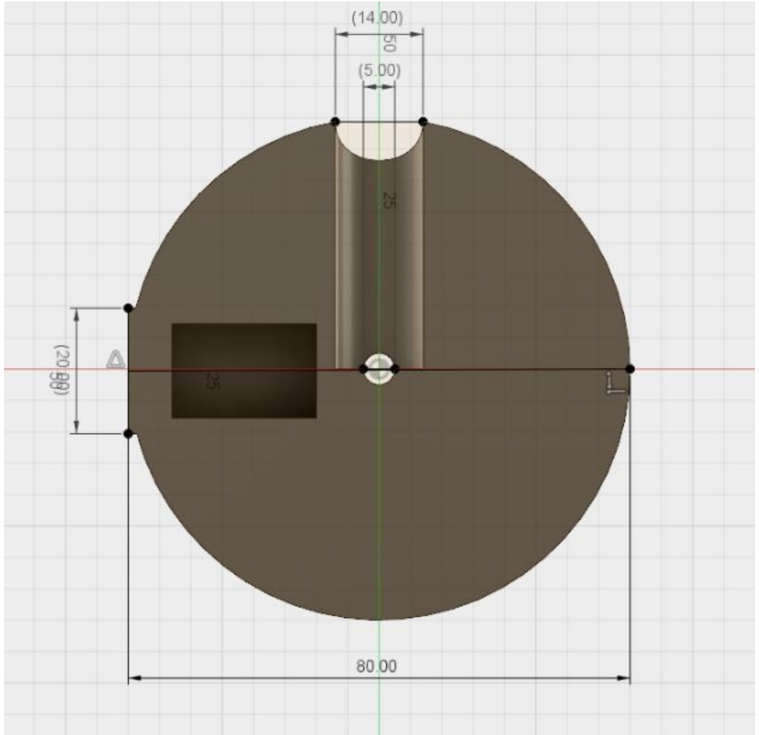




Motor Platter Assembly

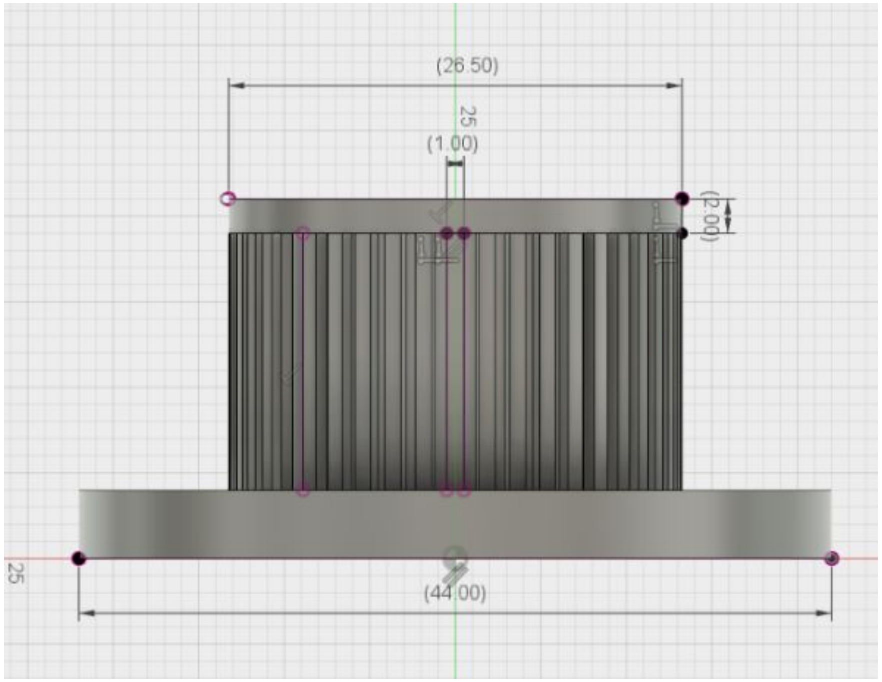
Platter support assembly

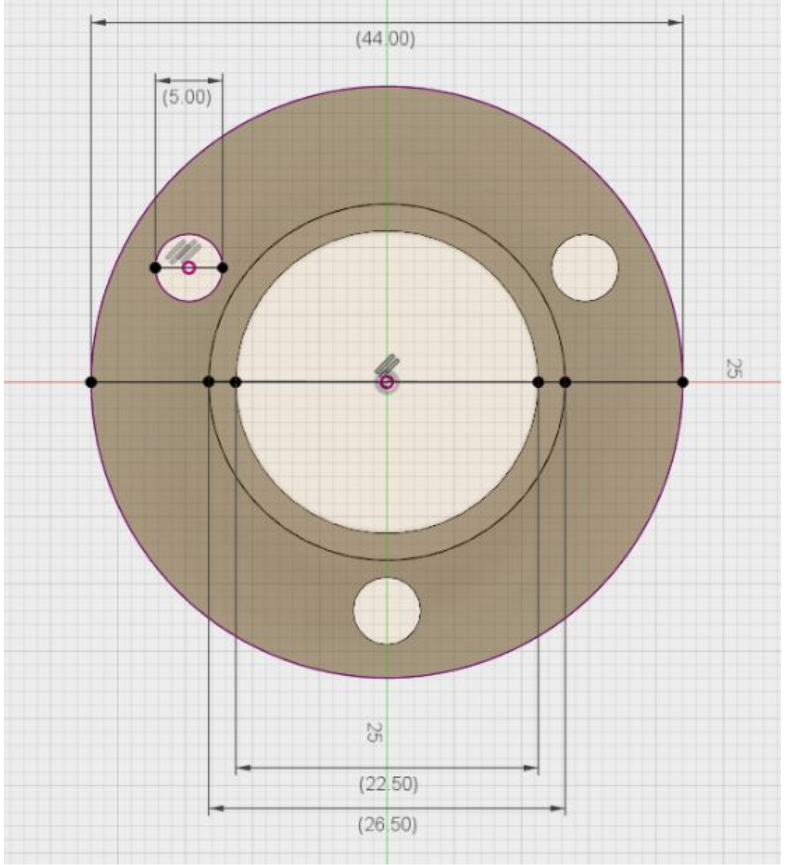




Platter Belt Ring Assembly

Gear Belt Platter Attachment





Final Complete Assembly Concept