3D-ROMAP

Interface Design Document

CECS 490

Revision: B

NOTICE

ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND. NOT WITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE ARE PROVIDED “AS IS” WITH ALL FAULTS. I DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL I BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF I HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Rev History

Revision Number	Release Date	Description
A	2018-03-22	Initial release
B	2018-05-22	Preliminary data added

Table of Contents

Index of Tables

Table of Figures

Acronyms and Abbreviations

Abbreviation	Description	Abbreviation	Description
°C	Celsius	Rev	Revision
°F	Fahrenheit	SLAM	Simultaneous Localization and Mapping
bpp	Bits per pixel	TBD	To Be Determined
CCA	Circuit Card Assembly	UART	Universal Asynchronous Receiver Transmitter
COMM	Communication	USB	Universal Serial Bus
CPU	Central Processing Unit		
dB	Decibels		
ESD	Electrostatic Discharge		
FAT	File Allocation Table		
FOV	Field of View		
ft	Feet		
g	Gram		
GPU	Graphics Processing Unit		
GUI	Graphical User Interface		
mA	Milliamps		
I^2C	Inter-Integrated Circuit		
I/O	Input/Output		
ICD	Interface Control Document		
ID	Identification		
IR	Infrared		
km	Kilometer		
LSB	Least Significant Bit		
mm	Millimeter		
ms	Millisecond		
MSB	Most Significant Bit		
NIR	Near Infrared		
OEM	Original Equipment Manufacturer		
PWM	Pulse Width Modulation		

Reference Documentation

The following documents form part of this specification. In the event of a conflict between documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

Document No:

1 Scope

This document describes the design characteristics for the 3D-Robotic Mapping Platform (3D -ROMAP).

2 Project Overview

The following research project documented in this report details a three-dimensional room mapping robot. The robot will be able to enter various rooms and be remotely controlled by the user. While the robot is in the specified environment, it will map the surrounding area in both a two-dimensional and three-dimensional representation. The user will then be able to view the generated maps on a separate remote desktop for observation purposes. The robot will combines various sensors in order to scan the environment around it and will utilize that data to determine its location in the current environment that it is exploring. This research project explores advanced image capturing techniques, navigation of unknown environments, as well as practical applications of electrical and mechanical engineering practices.

3 Project Description

The implementation of our project involves the utilization of an onboard GPU-accelerated embedded computer that will process the data received from both the LiDAR sensor and the Intel Realsense IR Depth Camera. A cost-efficient microcontroller will also be utilized to interface with not only the LiDAR sensor but also the various motors including DC motors and a single stepper motor.

The LiDAR sensor will be mounted on platform that will be rotating at a fixed frequency through the use of a stepper motor. The LiDAR will return a distance value which will be utilized to generate a 2D representation of the environment. In order to accomplish this, the TM4C will send the both the distance value as well as the angle that the stepper motor is at to the Jetson board for processing. The Jetson will then generate a two-dimensional cartesian coordinate plane with those sampled points plotted on it, which will become the two-dimensional representation of the overall environment around the robot. The sensor along with the rotating platform will be mounted at the top of the robot's chassis.

The Intel Realsense camera will be mounted at the front of the robot and will be utilized to extract depth and image data of the surrounding environment in front of it. The robot will capture the data at a fixed rate. This data will be output as a .pcd file format The data must then be filtered in order to reduce the distortion that occurs in the RGB point cloud capture for the user's observability. After the data has been captured as a point cloud, the data will then be stitched with other point cloud captures in post-processing. Due to uncertainty with the data compilation not being able to keep up with real time capture rates, we will not be performing the stitching in real-time. Instead we will stitch the point clouds generated using an open source utility software called Cloud Compare.

The position of the robot in the given environment will be determined through the utilization of SLAM, or simultaneous localization and mapping. The details of the algorithm itself will be elaborated in a further section of this documentation.

These technologies will be powered from a custom-built power supply designed by us. We chose the NVIDIA Jetson TX1 to perform the main computations and instruction executions for various reasons including: its powerful processing capabilities, 256 CUDA core GPU, and the ability to load a flavor of Linux to debug and control the software running. Most of our focus will be distributed towards capturing and processing the data from the sensors, and point cloud generation of our given position in the environment. The 3D representation will be sent using a mounted commercial off the shelf (COTS) router to generate a local WiFi connection for remote desktop connection. A static IP will be assigned to the Jetson, and the remote protocols of interest are VNC, SSH, X2Go, and RDP.

4 Similar Projects

MIT NVIDIA Jetson RACECAR:
Students at MIT developed a "Rapid Autonomous Complex-Environment Competing Ackermann-steering Robot", or RACECAR using the NVIDIA Jetson TX1 to process the data from their sensory suite. The goal was to design and implement various SLAM algorithms that would allow the vehicle to navigate a tunnel fully autonomously without human intervention. Our project will also utilize the NVIDIA Jetson TX1 to process our sensory data and generate the 2D and 3D maps for the user to observe on a remote desktop. This project shows that the hardware has the power and capability to successfully perform the task.

DARPA SLAM Car:
In 2005, DARPA issued the DARPA Grand Challenge, which was a competition where driverless cars would compete on an off-road course to determine which vehicle could autonomously navigate it the fastest. The winning vehicle, named "Stanley", successfully navigated the course the fastest with a time of 6 hours and 54 minutes. The vehicle utilized several technologies including: five LiDAR sensors to generate a 3D map of the environment to aid their GPS's positional sensing capabilities, a video camera to observe driving conditions, and an odometer to calculate how far the vehicle had traveled if the other sensors could not extrapolate enough data to determine its given position. Six low-power computers running Linux operating systems were also used to process the sensor data. Our project will share a few similarities with this design, including the utilization of Linux OS on the NVIDIA Jetson TX1 to process and execute instructions based on the data obtained from our single LIDAR sensor. We will also be utilizing the LIDAR sensor to generate a 2D map rather than a 3D map; instead, we will use an Xbox Kinect to perform the 3D mapping of the environment.

5 Project Significance

For our group, this project will serve as an in-depth introduction to vision systems, high performance embedded computing, power management, and exploration of robot localization techniques. This is intended as a research project in which the concepts listed above are implemented into a single functional design. We would hope that with further implementation, this research would result in possible consumer products such as advanced visual street maps and rescue platforms for victims of natural disasters.

6 Requirements

In order to create a clear end goal for our design, the following requirements describe the necessary elements to satisfy our project of creating a 3D robotic mapping platform:

The design shall map unknown coordinates through exploration on passable terrain

The initial goal of the design was to be able to map an unknown location. The current build of the project is intended to be demonstrated in an office environment, so the qualification of "passable terrain" has been defined as our exploration environment.

The design shall be capable of 2D SLAM

To keep us directed towards the SLAM based side of the research, we have specified that the research should be capable of 2D SLAM with the implementation of the PulsedLight LiDAR Lite. 3D SLAM has been omitted as a requirement to ease completion.

The design shall be capable of 3D capture

Our design needs to be able to capture an environment in 3D in a point cloud format. Our capture must be real time, however the design does not have to process and stitch images together in real time. This condition has been limited to post processing, as we fear data compilation may not be able to keep up with real time capture rates.

The design shall be capable of producing a visual map from 3D capture

Individual 3D point clouds are required to be stitched together to form a completed 3D representation of the environment. Stitching two images together has prove to be successful manually, however the automation of this process is the end goal.

The design shall be powered from an independent power supply

A custom power supply is being designed for this project. While the battery source has yet to be selected, we are expecting a 24V input battery source. The power supply will efficiently step down the voltage to 19V, 12V and 5V using a DC-DC buck converter. The component of interest is the Texas Instruments TPS54531 for high current applications to the 19V and 12V rails, while the TI TPS62177 will provide an efficient low power feed to the 5V rail to the MCU. During operation, the project should be capable of running a minimum of 20 min, which in turn allows for a 20 min room capture.

7 Work Breakdown

Due to the complexity of this endeavour, we wanted to make efficient use of our large group by dividing the work into two teams of two. This ensures that both teams have at least one person to collaborate with, and that no members are excluded from meaningful contribution to the project.

The two teams will be assigned one categorical task which will encompass the heart of our design. These categories are: 3D capture and SLAM implementation.

These tasks will be our first priorities for our path to success, as they will create the baseline for our first working prototype. Below lists the two categories, and the requirements for fulfilling the task.

7.1 3D Capture

Intel Realsense Capture

Task to interface with the Intel Realsense D435 stereo vision camera using the official SDK. From this capture we will extract depth and image data from the surroundings to form matrices. Due to the inaccuracies of stereo vision technology, we will be sampling and filtering the data before generating a point cloud of the capture.

3D Point Cloud

Task to associate the IR distance data to image data in a rendered point cloud using the Realsense SDK, and the Point Cloud Library (PCL) platform to generate the 3 dimensional depth structures.

3D Rendering

Task to use 3D rendering software to generate the visual of our stitched result. This task will be accomplished by capturing the point cloud in a supported 3D format such as the Polygon (PLY) format or Point Cloud Data (PCD) format. The post-processing is accomplished with the Open-Sourced utility Cloud Compare, for the point cloud registration and stitching.

7.2 Simultaneous Localization and Mapping (SLAM)

Algorithm

Task to research algorithms to track robot's position as it explores a room. Our preliminary algorithm can be seen in flowchart format in the software section. The algorithm flows as follows: gather positional changes from movement sensors like the odometers and accelerometers, and use this data to create an initial estimate of where the robot is in its environment. Next, gather data from environmental sensors, such as the LiDAR distance sensor, relate it to data previously collected from the environmental sensors, and combine them to create a more accurate position estimation.

Hardware Capture

Task to research and prototype methods of capturing distance data of the surroundings to be used by SLAM. This includes distance measurements from our LiDAR sensor, positional change from motor odometers, angular velocity from our gyroscope, directional acceleration from our accelerometer, or changes in magnetism (essentially a compass).

SLAM Implementation

Task to implement the chosen algorithms with hardware to feed positional data to 3D rendering for completion of 3D mapping robot. Our implementation will be based on Google Cartographer, which incorporates range data (LiDAR Lite V1), IMU data (MPU-9250), and odometry data (wheel encoders) to enable our robot to simultaneously localize and map the environment around it in a two-dimensional implementation. We chose to utilize Google Cartographer because it is open source software and fits well with our selected components.

8 Design Overview & Operation

Our robot design utilizes a microcontroller and an onboard processing computer to handle the various sub-components of the overall system. The microcontroller will handle motor control and will interface with an HC-05 bluetooth module as well as a LiDAR sensor. The onboard computer; the NVIDIA Jetson TX1, will be utilized to interface with our Intel Realsense IR stereo vision camera and will process and generate the 2D and 3D maps for our graphical user interface. The documentation below details each subcomponent in the design and its purpose within the overall system.

8.1 NVIDIA Jetson TX1

An embedded development platform that has the capability of delivering quick processing of complex data as well as delivering the power efficiency needed for visual computing applications. It contains an onboard GPU that contains 256 CUDA cores that is needed to generate the point clouds effectively. This will be used as our main computing module which will process the data obtained from both the LIDAR sensor and the Xbox Kinect to generate both a 2D and 3D map of the environment around the robot. It will also be used to interface directly with the Intel Realsense camera itself to capture the point cloud data and process it. A Linux Operating System will also be loaded onto the platform to provide a suite for debugging the system with ease. The Jetson will also enable the ability for the design to remotely connect with a desktop to view the 2D map as well as the robot's position within that given environment.

8.2 Texas Instruments Tiva C Microcontroller

An embedded microcontroller that contains an I2C communication protocol necessary to interface with the LiDAR Lite sensor as well as hardware pulse-width modulation to control the DC motors. The TM4C was selected mainly due to the familiarity with the development board. The onboard debugging capability will also help with the development process by reducing the amount of time debugging the system. The TM4C will also relay the data captured from the LiDAR sensor to the Jetson board to decrease the computing load on the microcontroller itself. The TM4C will also be utilized to interface with the HC-05 bluetooth module to capture character inputs that will be processed by the TM4C and control the robot's directional steering.

8.3 HC-05 Bluetooth Module

The HC-05 bluetooth module will be utilized to capture data from a bluetooth terminal and transmit it over UART to the TM4C microcontroller. The captured character inputs will be utilized to allow the user to steer and control the direction that the robot navigates in the environment through the use of either an application on a smartphone or a physical controller.

8.4 PulsedLight LiDAR Lite

The LIDAR-Lite v1 sensor module is a LIDAR sensor that captures the distance an object is away from it at a max range of ~40m.The LIDAR sensor itself will be rotating on a platform constantly and will send

the data it generates through the I2C communication protocol to the TM4C. That data that the sensor returns is the distance based on the time of flight. That data will then be relayed to the Jetson where it will be plotted on a two-dimensional cartesian plane as a representation of a point in the environment. The data will also be utilized with the SLAM algorithms to pinpoint the robot's position in the environment on that two-dimensional plane. This sensor was chosen due to its relatively lower cost in comparison to other LIDAR sensors as well as its frequency in generating the data points (100 Hz).

8.4.1 LiDAR Verification

In order to verify that our lidar sensor functions correctly, we utilized an Arduino alongside with the prepackaged software that the manufacturers of the sensor provided for us. The software involved using the I2C communication protocol to capture the data readings from the lidar sensor. The data is returned as a split between the 8 upper bits as well as the 8 lower bits of data and are stored in an array. The contents of the array are then combined and shifted to generate the correct distance value as an integer. The distance is then displayed using the serial terminal on the Arduino IDE. We utilized the demo code and verified that the physical lidar sensor itself functions properly by observing the distance readings that are read through the I2C. We then studied the algorithm and ported the software over using the C language to the TM4C microcontroller. We then utilized an open source library that featured several pre-built functions that allowed us to use the I2C protocol on the TM4C to communicate with the lidar sensor. We declared the TM4C as the master and the sensor as the slave and began reading data from it. Rather than storing the upper and lower bits of data and combining them, we read a different address on the sensor that would return the both the upper and lower bits at once and saved it in a single variable. That variable was then displayed through a UART onto our serial terminal and we compared the distance values that were being captured with the Arduino code's version. Both values were within range of one another, verifying that our port to the TM4C was a success and that the lidar sensor functions correctly.

8.5 Stepper Motor

The 17HS13-0404S1 Stepper Motor will be used to rotate the platform that the LIDAR sensor is mounted on to assist in generating the point cloud for the 2D map representation. Stepper motor will have a 1.8° step angle and will have a step frequency of 200 Hz. This means that our platform will complete one full rotation each second (stepper turns 360° every second). The stepper motor was selected for prototyping purposes as well as familiarity with interfacing with the device with the TM4C.

8.6 Stepper Motor Driver

The A4988 is a stepper motor driver that will be used to control our stepper motor that will rotate the platform that holds the LIDAR sensor. We will be using the driver in full step rotational mode, giving up one full step angle (1.8°) per pulse to the driver's 'step' port. We are generating a 200 Hz, 50 % duty cycle wave and sending it ot the step port, causing the stepper to turn 360° every second. Additionally, the direction port of the driver will receive a steady signal of logical 0, as we will be constantly turning the platform in a single direction. This driver was chosen due to its low cost and compatibility with the TM4C microcontroller. It is compatible with a logic voltage level of 3.3V and is also powered by 3.3V, which is a readily available output on our TM4C.

8.7 DC Motor H-Bridge

The L298N is an H-bridge that will be used to control our DC motors that will provide the robot with movement. This part was also chosen due to its relatively low cost and simplicity in interfacing with the TM4C microcontroller. Because of its low cost, this part acts as a test component before we move into a higher current rated H-Bridge for final implementation of the robot.

8.8 MPU-9250 IMU

The MPU-9250 is an inertial measurement unit that contains 9 degrees of freedom, or 9-axis motion processing. It contains an accelerometer for measuring x,y, and z directional acceleration, a gyroscope for measuring x,y, and z angular velocity, and a magnetometer which is essentially a compass. The MPU-9250 will be utilized in the design to assist in the SLAM algorithm for tracking the robot's position in the environment. Currently testing to see whether we can extract positional data from our visual maps to determine the necessity of utilizing this sensor in the design.

8.9 Intel Realsense Depth Camera

The Intel Realsense is an structured-light, active infrared stereo vision depth camera that will utilized to capture 3D point cloud data rather than the Xbox Kinect in our future iterations. The sensor allows for image overlay as well as the utilization of point clouds for depth representation. We chose to switch to this specific sensor due to the sensor's well-documented development platform: the Realsense SDK. It is a library that allows us to utilize the sensor and capture point cloud data and export it as data to file. It also provides a relatively extensive debugging environment that generates camera log files, frame statistic reports, and firmware test commands that can be sent directly to the camera itself. The camera also allows us to associate RGB values with specific points on the cloud rather than having us overlay the RGB values post-processing. The Realsense camera also has wider viewing angles in comparison to the Kinect which allows for a faster rate of data capture. The RGB camera and IR camera have an FOV of 69.4° and 85.2° respectively.

8.10 Preliminary Chassis

The chassis is constructed from aluminum extrusion. Extrusions are held together via angle brackets and t-slot nuts and an assortment of m3 and m4 sized fittings. Faces of the rover's body are constructed of MDF. All major components are mounted to the chassis with fasteners and 3D printed parts. Expected volume of the rover is estimated to be somewhere around 300x300x300mm. However not all hardware components have been finalized and this may be adjusted in the future.

The following figure details a prototype of the robot chassis:

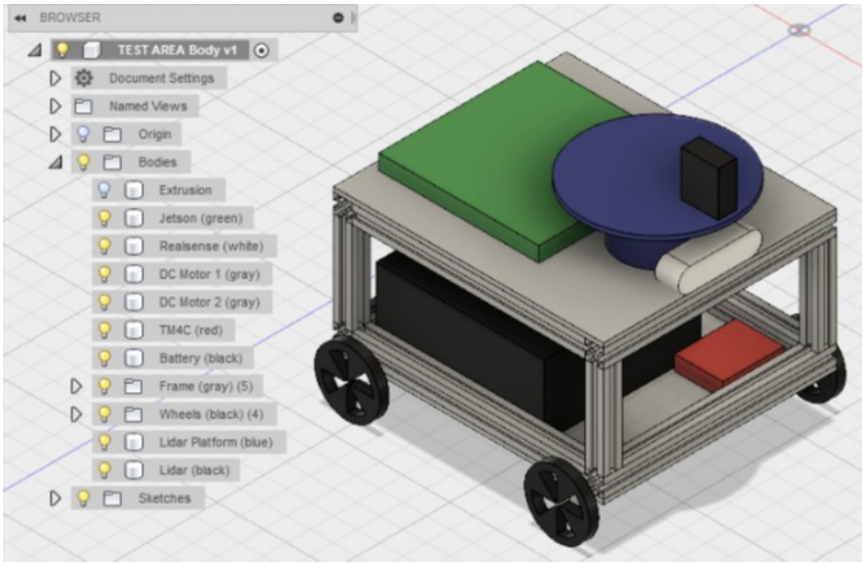

Illustration 1: Preliminary Chassis Concept

9 Preliminary Block Diagram

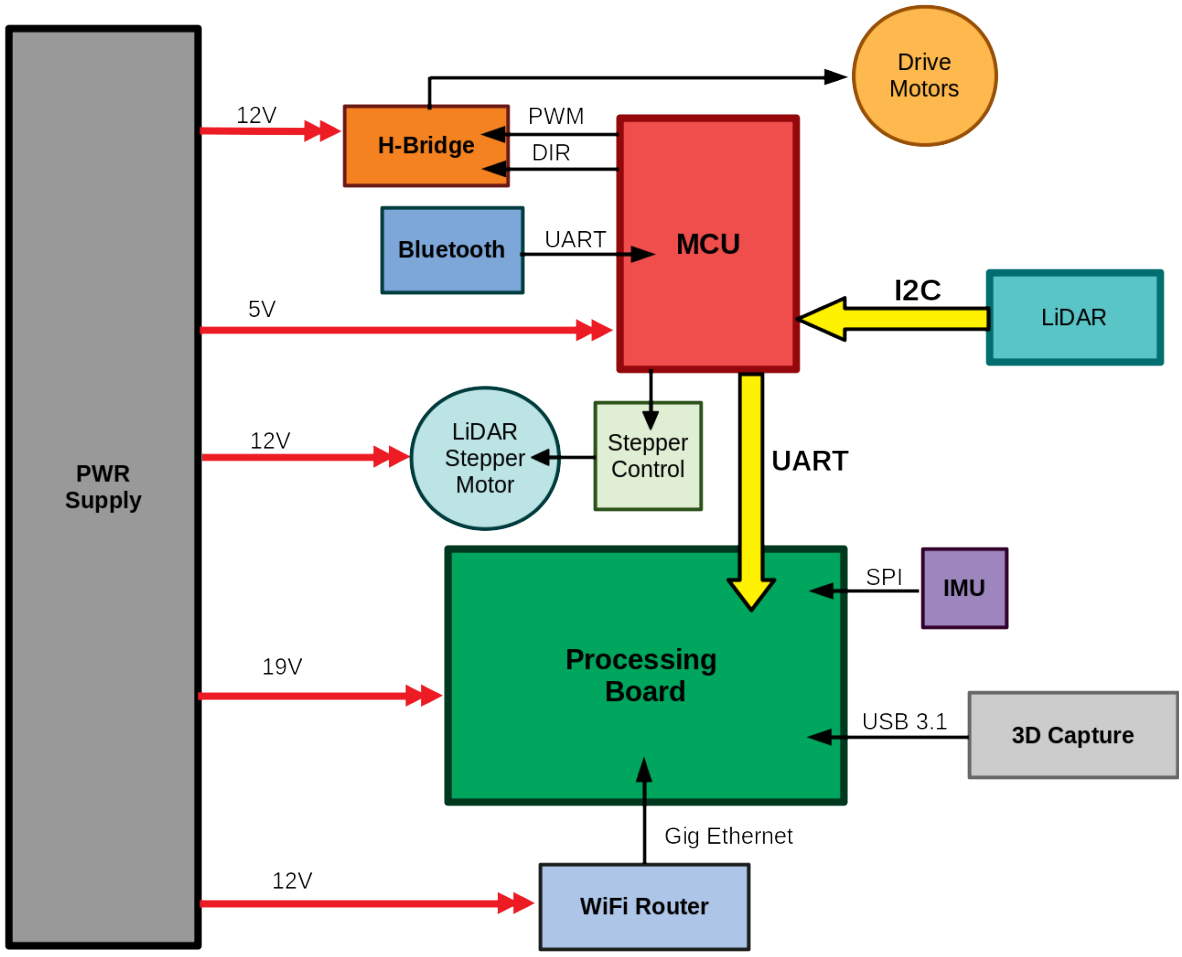

Illustration 2: Top Level Block Diagram

The preliminary Block Diagram shows a top level view of parts used in the design as well as the power and data rails interconnecting each. The power supply is currently estimated to be a 24V source that will be stepped down to 19V, 12V, and 5V for the respective components. Red arrows denote power delivery to the electronics, and black arrows denote communication signals and their direction. The bold yellow arrow denotes communication from the LiDAR to the MCU and finally to the Processing board.

10 Detailed Block Diagrams

10.1 Microcontroller Detailed Block Diagram

The following functional block diagram details the port connections and signals for the TM4C microcontroller portion of the robot design. The diagram also details all ports that will be utilized on the microcontroller itself for various alternative functionalities including UARTs, I2C, and PWM generators.

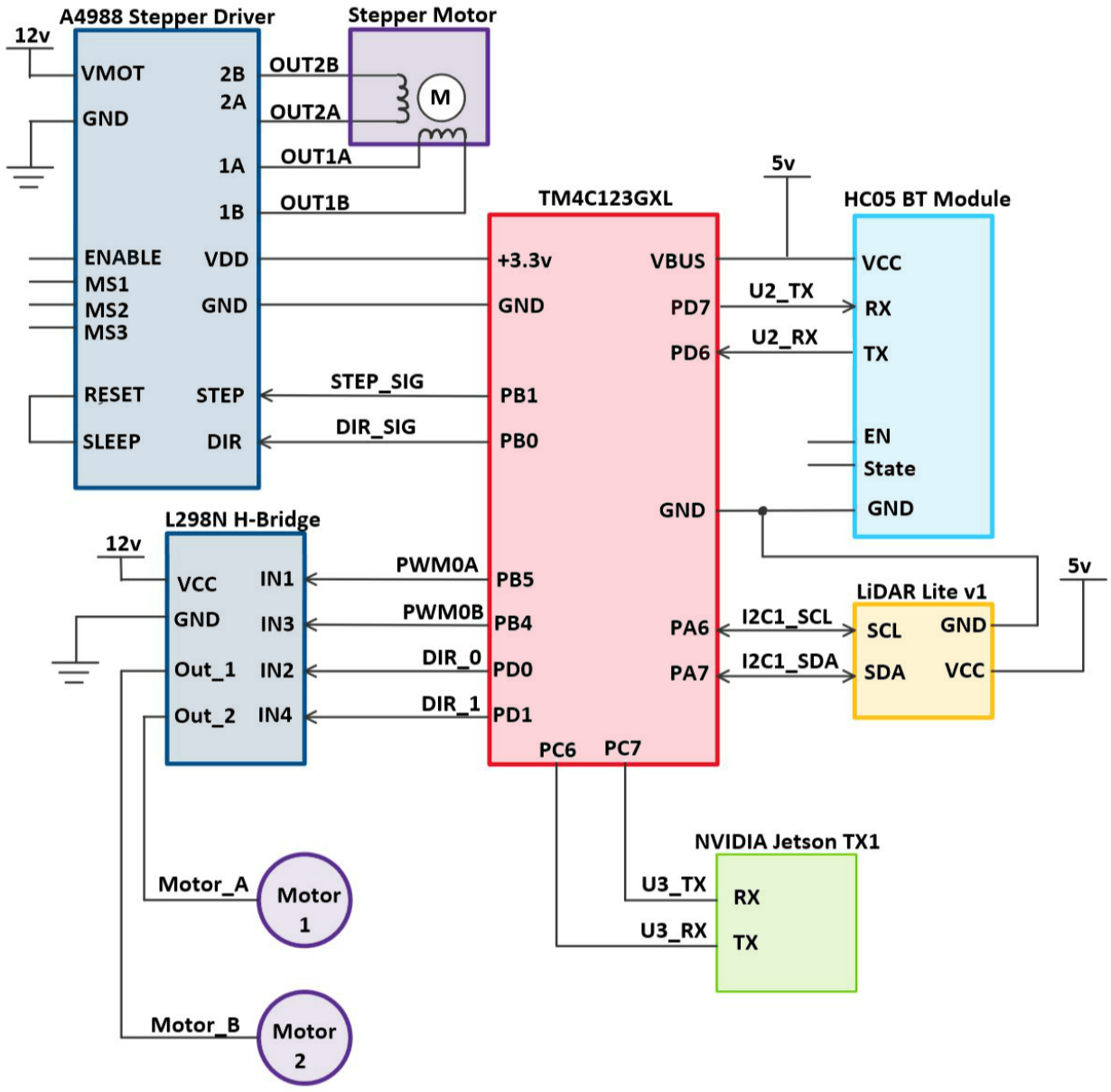

Illustration 3: MCU detailed block diagram

The following charts will detail various aspects of each module including the signal names, descriptions, signal types, voltage and current requirements, and their operation.

TM4C123GXL Description Table

Signal Name	Signal Description	Signal Type	Voltage Level	Current Req.	Operation
+3.3v	Regulated 3.3v from microcontroller	Power Output	3.3v	300mA (Max)	Provides an output of 3.3v to the A4988 stepper motor driver
VBUS	Input to power the microcontroller and output regulated 5v from microcontroller	Power Input/ Output	5v (+/-0.25v)	323mA (Max)	5v input to power the entire microcontroller and outputs 5v to power the HC05 bluetooth module
STEP_SIG (PB1)	GPIO Output	Digital Output	0v to 3.3v	N/A	Provides an output signal to step the stepper motor a certain distance
DIR_SIG (PB0)	GPIO Output	Digital Output	0v to 3.3v	N/A	Provides an output signal to change the direction the stepper motor rotates
PWM0A (PB5)	PWM signal output	Digital Output	0v to 3.3v	N/A	Provides a pulse-width modulated signal to the L298N H-Bridge to control the speed of rotation on DC motors
PWM0B (PB4)	PWM signal output	Digital Output	0v to 3.3v	N/A	Provides a pulse-width modulated signal to the L298N H-Bridge to control the speed of rotation on DC motors
DIR_0 (PD0)	GPIO Output	Digital Output	0v to 3.3v	N/A	Provides an output signal to the L298N H-Bridge to control the directional rotation of the DC Motors
DIR_1	GPIO Output	Digital	0v to 3.3v	N/A	Provides an output signal

(PD1)		Output			to the L298N H-Bridge to control the directional rotation of the DC Motors
U2_TX (PD7)	UART Functionality (TX)	Digital Output	0v to 3.3v	N/A	Transmits characters to the HC05 for configuration purposes
U2_RX (PD6)	UART Functionality (RX)	Digital Input	0v to 3.3v	N/A	Receives characters from the HC05 for steering control
U3_TX (PC7)	UART Functionality (TX)	Digital Output	0v to 3.3v	N/A	Transmits LiDAR distance data to the NVIDIA Jetson TX1 for processing
U3_RX (PC6)	UART Functionality (RX)	Digital Input	0v to 3.3v	N/A	In case we need to relay data from the Jetson to the TM4C
I2C1_SCL (PA6)	I2C Functionality (Clock Signal)	Digital Output	0v to 3.3v	N/A	Provides clock signal for I2C communication between the TM4C and the LiDAR sensor
I2C1_SDA (PA7)	I2C Functionality (Data Line)	Digital Input/ Output	0v to 3.3v	N/A	Data line to transfer data between the TM4C and the LiDAR sensor

HC05 Bluetooth Module Description Table

Signal Name	**Signal Description**	**Signal Type**	**Voltage Level**	**Current Req.**	**Operation**
VCC	Power input to utilize module	Power Input	3.3v to 5v	35mA	Power input of 3.3v to 5v to turn on module
RX	UART Functionality (RX)	Digital Input	0v to 3.3v	35mA	Data input through UART communication for module configuration
TX	UART Functionality (TX)	Digital Output	0v to 3.3v	35mA	Transmits data received from a bluetooth terminal to the TM4C
EN	Power enable to switch to	Power Input	3.3v	35mA	Only connected when configuring settings for

	configuration mode				HC05 module; otherwise unconnected during operation
State	N/A	N/A	N/A	N/A	Unconnected

LiDAR Lite v1 Sensor Description Table

Signal Name	Signal Description	Signal Type	Voltage Level	Current Req.	Operation
VCC	Power input to utilize sensor	Power Input	5v	130mA Max	Power input of 5v to power on sensor
SCL	I2C Functionality (Clock Signal)	Digital Input	0v to 3.3v	130mA	Clock signal input for I2C communication between TM4C and LiDAR sensor
SDA	I2C Functionality (Data Line)	Digital Input/ Output	0v to 3.3v	130mA	Data line to communicate data between TM4C and LiDAR sensor

L298N H-Bridge Description Table

Signal Name	Signal Description	Signal Type	Voltage Level	Current Req.	Operation
VCC	Voltage input to power the DC Motors	Power Input	12v	2A	Power input of 12v to power both DC motors
PWM0A (IN1)	Logical Input to H-Bridge	Digital Input	0v to 3.3v	N/A	PWM Signal to logical Input of H-Bridge to control DC Motor 1
PWM0B (IN3)	Logical Input to H-Bridge	Digital Input	0v to 3.3v	N/A	PWM Signal to logical Input of H-Bridge to control DC Motor 2
DIR_0 (IN2)	Logical Input to H-Bridge	Digital Input	0v to 3.3v	N/A	GPIO Output signal to logical Input of H-Bridge to control directional rotation of DC Motor 1
DIR_1 (IN4)	Logical Input to H-Bridge	Digital Input	0v to 3.3v	N/A	GPIO Output signal to logical Input of H-Bridge to

					control directional rotation of DC Motor 2
Motor_A (Out_1)	Voltage Output to DC Motor	Power Output	12v	2A	Power Output to provide rotation for DC Motor 1
Motor_B (Out_2)	Voltage Output to DC Motor	Power Output	12v	N/A	Power Output to provide rotation for DC Motor 2

A4988 Stepper Driver Description Table

Signal Name	**Signal Description**	**Signal Type**	**Voltage Level**	**Current Req.**	**Operation**
VMOT	Voltage input to power the stepper motor	Power Input	12v	4A max	Power input of 12v to power the stepper motor
VDD	Voltage input to power the stepper driver	Power Input	3.3v	8mA	Power input of 3.3v to power the stepper driver
STEP_SIG (STEP)	Step Input to Stepper Driver	Digital Input	0v to 3.3v	N/A	GPIO Output signal from TM4C to control the steps of the motor
DIR_SIG (DIR)	Direction Input to Stepper Driver	Digital Input	0v to 3.3v	N/A	GPIO Output signal from TM4C to control the direction the motor steps in
OUT2B (2B)	Stepper Output to coil of Stepper Motor	Power Output	8v to 35v	N/A	Power Output signal from stepper to coil 1 of stepper to provide rotation
OUT2A (2A)	Stepper Output to coil of Stepper Motor	Power Output	8v to 35v	N/A	Power Output signal from stepper to coil 1 of stepper to provide rotation
OUT1A (1A)	Stepper Output to coil of Stepper Motor	Power Output	8v to 35v	N/A	Power Output signal from stepper to coil 2 of stepper to provide rotation
OUT1B (1B)	Stepper Output to coil of Stepper Motor	Power Output	8v to 35v	N/A	Power Output signal from stepper to coil 2 of stepper to provide rotation

10.2 Microcontroller Detailed Block Diagram

The following functional block diagram details the port connections and signals for the NVIDIA Jetson TX1 processing board portion of the robot design. The diagram details all modules that will be interfacing with the Jetson including the Intel Realsense camera for 3d capture, a router to provide a wifi connection to the Jetson, an IMU, and the TM4C to process the data coming from the microcontroller.

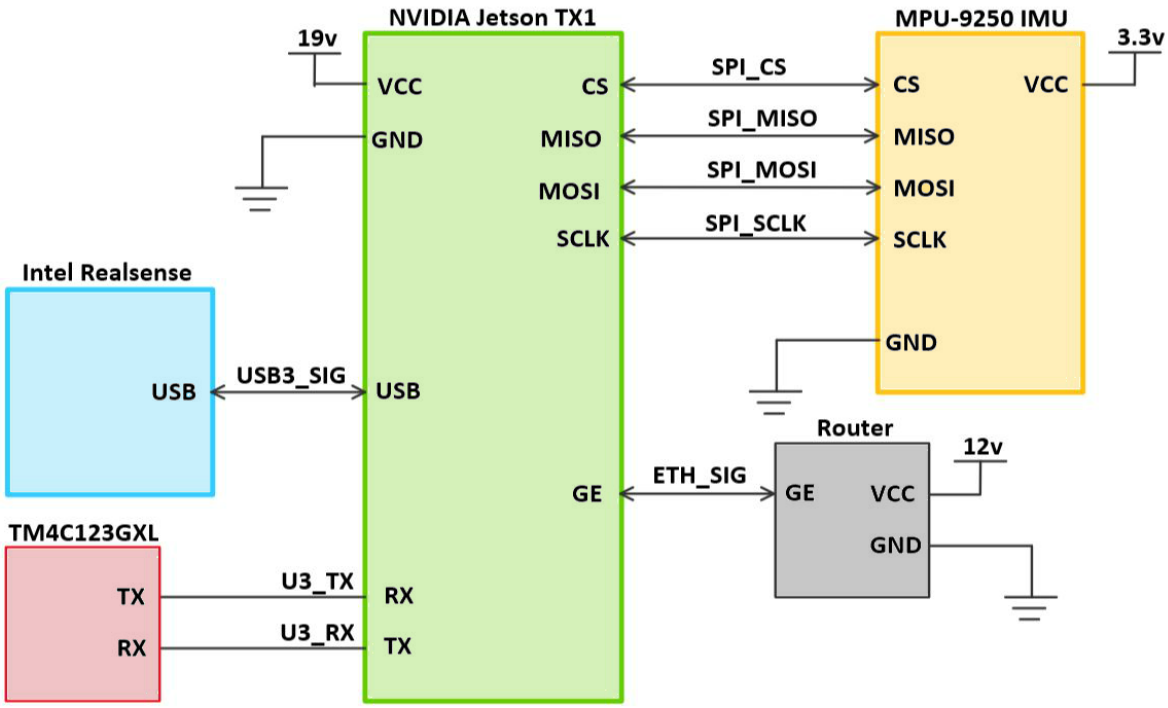

Illustration 4: Processing board detailed block diagram

The following charts will detail various aspects of each module including the signal names, descriptions, signal types, voltage and current requirements, and their operation.

NVIDIA Jetson TX1 Description Table

Signal Name	Signal Description	Signal Type	Voltage Level	Current Req.	Operation
VCC	Input to power the processing board	Power Input	5v-19v	0.8A	Power input of 19v to power the Jetson board
USB	USB 3.1 Port for data communication	Digital Input/ Output	5v	N/A	USB communication to capture data from the Intel Realsense camera to the Jetson for processing
U3_TX (RX)	UART Functionality (RX)	Digital Output	0v to 3.3v	N/A	Receives LiDAR distance data from the TM4C for processing
U3_RX (TX)	UART Functionality (TX)	Digital Input	0v to 3.3v	N/A	In case we need to relay data from the Jetson to the TM4C
CS	Chip Select for SPI utilization	Digital Input	0v to 3.3v	N/A	Port to switch between I2C and SPI utilization on the IMU interface
SPI_MISO (MISO)	Master-In, Slave-Out signal for SPI communication	Digital Input/ Output	0v to 3.3v	N/A	Port to read data from the IMU(Slave) to the Jetson (Master)
SPI_MOSI (MOSI)	Master-Out, Slave-In signal for SPI communication	Digital Input/ Output	0v to 3.3v	N/A	Port to write data from the Jetson(Master) to the IMU(Slave)
SPI_SCLK (Clock Signal)	SPI Functionality (Clock Signal)	Digital Output	0v to 3.3v	N/A	Port outputs a common clock signal between the Jetson and the IMU
GE	Gigabit Ethernet Connection	Digital Input/ Output	N/A	N/A	Port connects to a router using an ethernet cable to provide a wifi connection to the Jetson

MPU-9250 IMU Sensor Description Table

Signal Name	Signal Description	Signal Type	Voltage Level	Current Req.	Operation
VCC	Power input to utilize IMU sensor	Power Input	3.3v	8µA - 3.7mA	Power input of 3.3v to power on sensor. Current requirement dependent on number of axis utilized on IMU
CS	Chip Select for SPI utilization	Digital Input	0v to 3.3v	N/A	Port to switch between I2C and SPI utilization on the IMU interface
SPI_MISO (MISO)	Master-In, Slave-Out signal for SPI communication	Digital Input/ Output	0v to 3.3v	N/A	Port to read data from the IMU(Slave) to the Jetson(Master)
SPI_MOSI (MOSI)	Master-Out, Slave-In signal for SPI communication	Digital Input/ Output	0v to 3.3v	N/A	Port to write data from the Jetson(Master) to the IMU(Slave)
SPI_SCLK (Clock Signal)	SPI Functionality (Clock Signal)	Digital Output	0v to 3.3v	N/A	Port outputs a common clock signal between the Jetson and the IMU

Intel Realsense Depth Camera Description Table

Signal Name	Signal Description	Signal Type	Voltage Level	Current Req.	Operation
USB	USB3.1 Provides power and communication	Power & Data I/O	5v	N/A	Power input and communication between the Jetson and the Realsense camera

Router Description Table

Signal Name	Signal Description	Signal Type	Voltage Level	Current Req.	Operation
VCC	Power input to utilize router	Power Input	12v	1.5A (Max)	Voltage input of 12v to power on the Router
GE	Gigabit Ethernet Connection	Data Input	N/A	N/A	WiFi connection to the Jetson for remote access

11 Power Analysis

The robot is projected to house two separate battery sources for the motor drive and electronics power supply board to isolate motor noise. Below is a table of estimated voltage requirements, and the estimated current draw in amps. This is used to help select an appropriate voltage regulator per voltage supply. The second table notes the total power draw per voltage source to help determine the appropriate battery chemistry for our device.

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	

Illustration 5: Voltage rails and current

From these results we determined a 24V power source was necessary to meet the 19V requirement for the Jetson, and analyzed our options for runtime and battery chemistry. In order to calculate our total runtime minimum to meet the 20 minute operating requirement, we used the following formula:

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

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

Illustration 6: Power requirement and power draw per channel

Assuming a 7Ah (Amp hour) rated battery, at ideal conditions, our runtime would be:

0.57 Hours = (10 * 7Ah) / 122.1W

With this result we then evaluated the pros and cons of various battery sources. In order to achieve the power draw we necessary for our application, standard alkaline batteries would be a poor choice due to their inability to be recharged, poor discharge characteristics, low current delivery, and low voltage all leading to an unrealistic endeavour of inconsistent discharges, high quantities of batteries, and excessive waste.

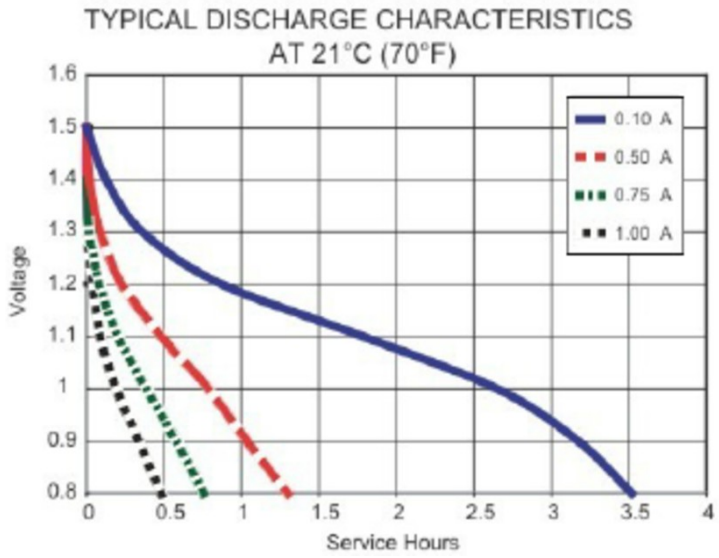

Illustration 7: Alkaline battery discharge

From this evaluation, we could also disregard the NiMH cells due to their low voltage characteristics, and the unreasonable quantity to achieve the energy density we need. In addition, we also chose to disregard Lithium Ion batteries from our selection due to their unsafe nature, lack of 24V sources at high amp hours, inability to safely connect two sources in series, and overall cost to achieve any reasonable results.

After we conducted our analysis of these options, we assumed sealed lead acid (SLA) batteries would yield the best results for our application due to their robust nature, price for performance, recharge capabilities, and endurance for more power aggressive applications.

However, after evaluating the various types of SLAs, we found that due to their weight, and 50% capacity discharge threshold, that the power source would be less than ideal to achieve our runtime. In order to be certain our robot would be capable of supplying the correct voltage rails without damaging our battery source, we would need to purchase batteries with a rating of 10Ah or more to achieve at least 24 minutes. For the price, if we further explore this technology we have explored the options of deep cycle SLA at a higher Amp-hour rating due to their better performance for our application of power usage. Below is the discharge curve for a deep cycle SLA from PowerSonic.

An additional chemistry we have explored is Lithium Iron Phosphate, due to their 70% discharge threshold, an average of 40% lighter than SLAs, and lowered volatile nature over Lithium Ion. The only reason we would not consider this as a solution to our robot, is the cost associated with the power source. At 7Ah we are looking at an average of $130 for a pair vs $30 for a standard SLA pair. Below are the discharge curves of both the SLA and the LiFePO4 batteries.

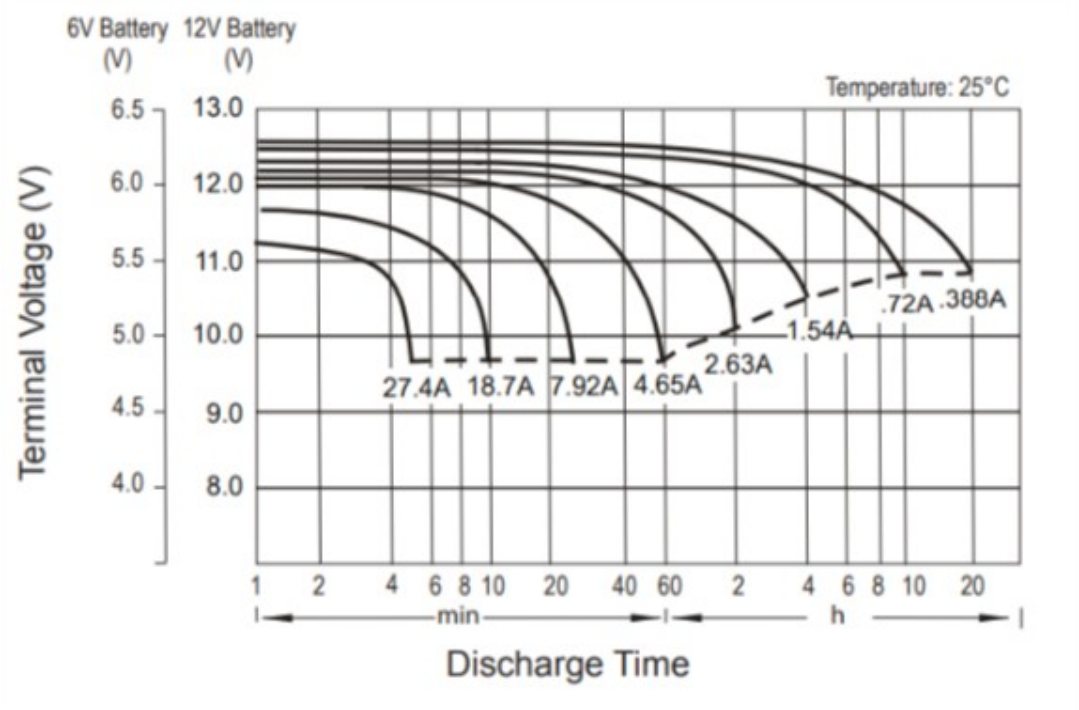

Illustration 8: Deep cycle sealed lead acid discharge

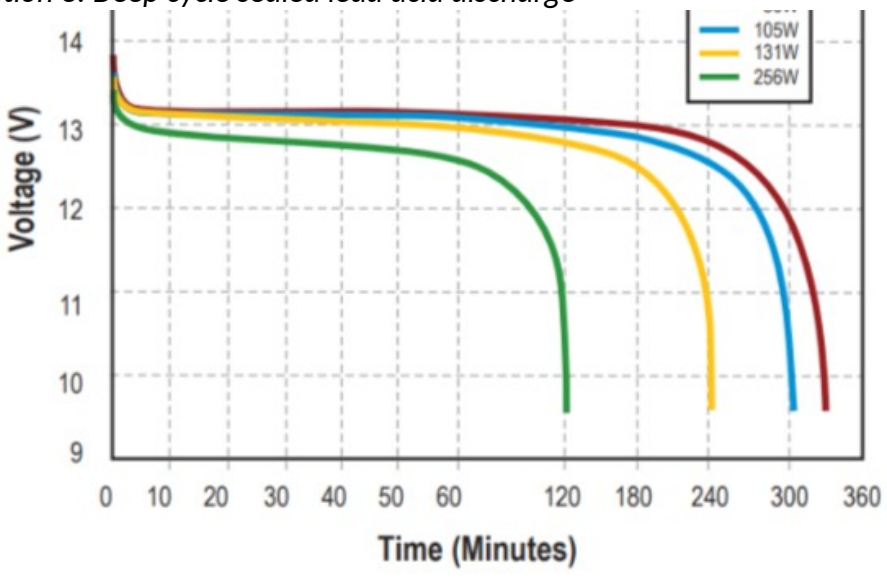

Illustration 9: LiFePO4 runtime discharge

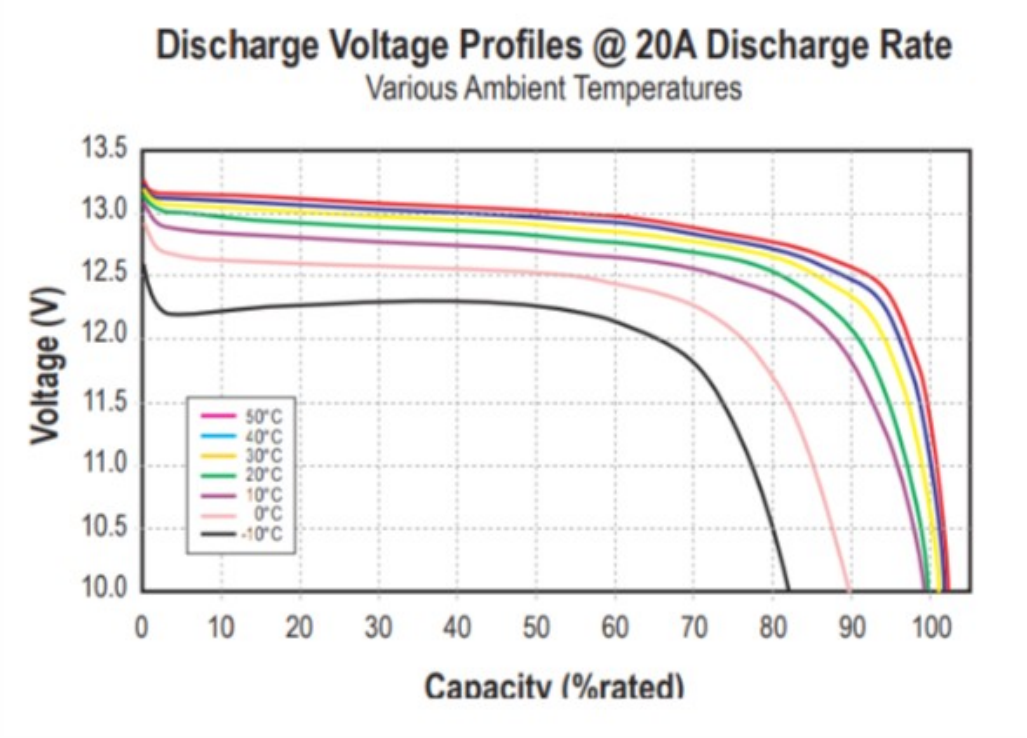

Illustration 10: LiFePO4 capacity discharge

12 Hardware Design

12.1 Texas Instruments - TPS54531

High efficiency, high current DC-DC buck converter used to source 19V and 12V electronics in our design. The max current allowed from this regulator is 5A, which is perfect for our 2A motors, and max current draw of 4A of our Jetson. This chip is also inexpensive for our design, and in a compact SMT package.

The formula used to calculate the minimum inductance for our requirements is as follows:

$$L_{min} = \frac{V_{out}\left(Vin_{MAX} - V_{out}\right)}{Vin_{MAX} * K_{IND} * I_{out} * f_{sw}}$$

From this equation we get 12uH minimum inductance for our requirements, using a Vout of 12V and 19V, a Vin (max) of 30V, an Iout of 5A, a switching frequency of 570KHz, and a constant K of 0.2 for higher ESR coefficient.

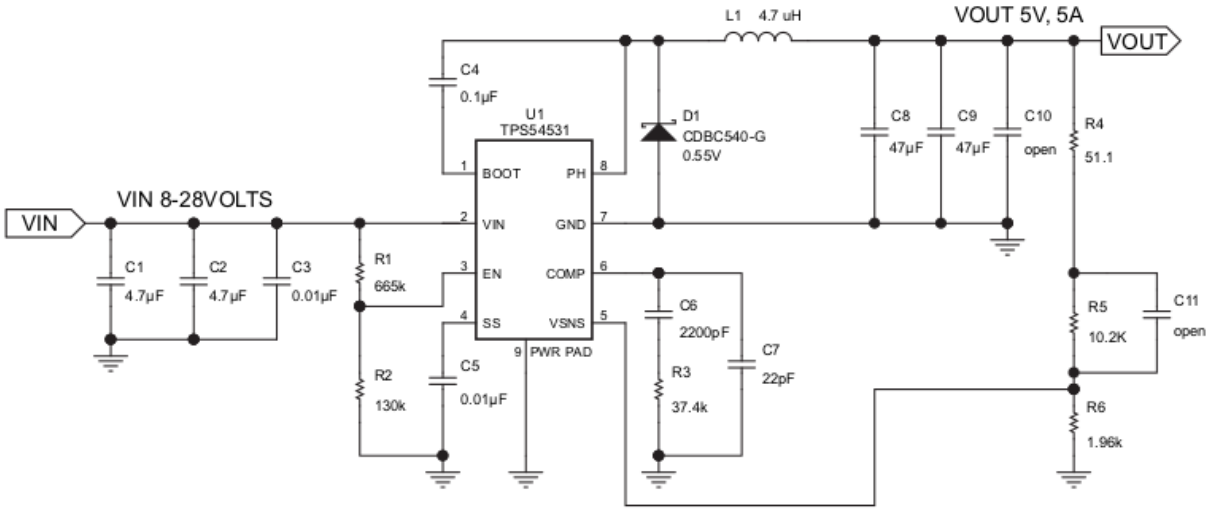

Illustration 11: 19V and 12V DC-DC Converter

To achieve the voltages desired we used the following formula:

$$R6 = \frac{10K(0.8V)}{Vout - 0.8V} = \frac{R5(Vref)}{Vout - Vref}$$

For 19V we achieved the R6 voltage divider value of 440, and for 12V we achieved the resistor value of 714. See schematic for circuit implementation.

12.2 Texas Instruments - TPS62175

High efficiency at low current DC-DC buck converter used to source 5V to our MCU. The max current allowed from this regulator is 500mA, and is specifically tuned for MCU usage. This chip is also inexpensive for our design, and in a compact SMT package.

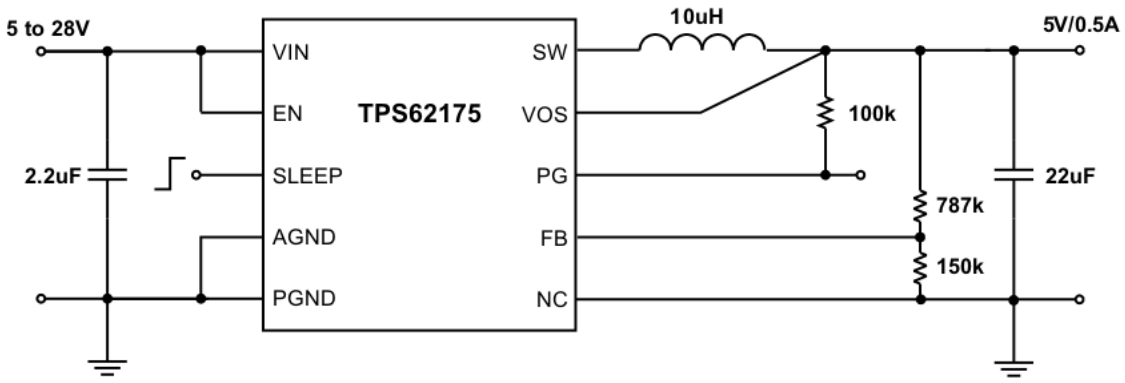

Illustration 12: 5V DC-DC Converter

13 Software Design

The following software flow chart details the overview that our SLAM algorithm will take to provide our system with an accurate estimate position of where it is in the room. During the LiDAR Reading stage, our Jetson will be reading the most recent samples from the LiDAR sensor, which will be send into the landmark extraction stage where the system will detect objects in the environment such as walls, chairs, furniture, etc. The Jetson will then compare the objects that are currently being detected and compare it to objects that have been detected in the past. During the re-observation and new observations stage, the algorithm will determine if a given object has been seen in the past. If the object has not been seen before, it will be marked as a new object and added to the list of recognized objects that are used in the data association step. However, if the object has been seen before, during the re-observation stage the Jetson will combine the distance data from the LiDAR and the current estimate position to create a new, more accurate estimated position. Lastly, as the wheels keep turning the odometry data is updated and the cycle repeats.

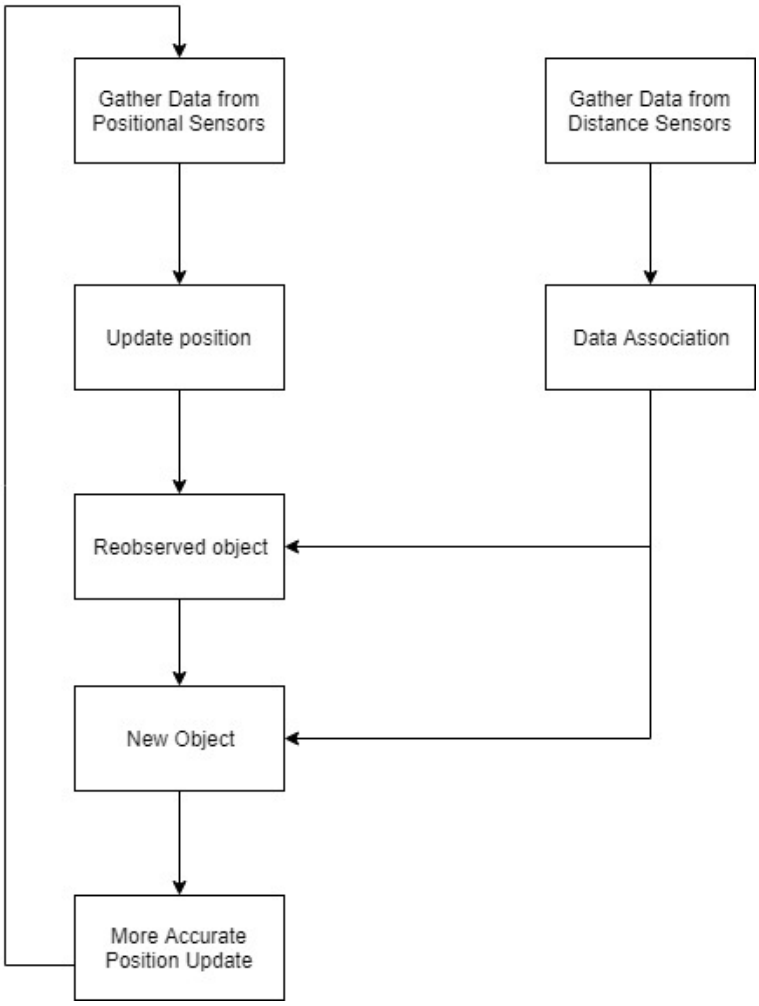

Illustration 13: SLAM Software Flow

The software that we plan to utilize for the environment mapping portion of the design involves capturing data for both the 2D and 3D map rendering. The 2D portion of the design involves interfacing with the LiDAR sensor through I2C, capturing the distance values it returns, and plotting it on a two-dimensional cartesian plane. In order to generate the plane, we must utilize both the distance the sensor detects as well as the angle that the sensor is facing. This angle is given by the stepper motor's orientation. Every sampled point is then plotted on the plane, where eventually the plane itself becomes the 2D representation of the environment around it.

The 3D portion of the design involves programming the camera to capture a frame of data, or point cloud, and output the data as a file format at a fixed rate. The file format ideally should be a .pcd type format in order for us to utilize the point cloud library and stitch multiple clouds together. These stitches allow us to generate a 3d representation of the environment around the robot.

14 Referenced External Documentation

[1]P. Henry, M. Krainin, E. Herbst, X. Ren and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments", *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647-663, 2012.

[2]K. Khoshelham and S. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications", *Sensors*, vol. 12, no. 2, pp. 1437-1454, 2012.

[3]N. Namitha, S. Vaitheeswaran, V. Jayasree and M. Bharat, "Point Cloud Mapping Measurements Using Kinect RGB-D Sensor and Kinect Fusion for Visual Odometry", *Procedia Computer Science*, vol. 89, pp. 209-212, 2016.

[4]P. Tarkowski, I. Malujda, K. Talaśka, M. Kukla and J. Górecki, "Adjustment of the Distance of Objects to the Microsoft Kinect Device Fitted with Nyko Zoom Attachment Used in a Three-axis Manipulator", *Procedia Engineering*, vol. 177, pp. 387-392, 2017.

[5]M. Chang and Z. Kang, "AN INDOOR SLAM METHOD BASED ON KINECT AND MULTI-FEATUREEXTENDED INFORMATION FILTER", *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. -27, pp. 327-332, 2017.

[6]C. Bellés and F. Pla, "A Kinect-Based System for 3D Reconstruction of Sewer Manholes", *Computer-Aided Civil and Infrastructure Engineering*, vol. 30, no. 11, pp. 906-917, 2014.

[7]L. Lu, R. Tao and G. Jin, "A Low-Cost 3D Local Mapping and Localization Method", *Applied Mechanics and Materials*, vol. 556-562, pp. 4089-4092, 2014.

[8]L. Zhang, P. Shen, G. Zhu, W. Wei and H. Song, "A Fast Robot Identification and Mapping Algorithm Based on Kinect Sensor", *Sensors*, vol. 15, no. 8, pp. 19937-19967, 2015.

[9]K. Kamarudin, S. Mamduh, A. Shakaff and A. Zakaria, "Performance Analysis of the Microsoft Kinect Sensor for 2D Simultaneous Localization and Mapping (SLAM) Techniques", *Sensors*, vol. 14, no. 12, pp. 23365-23387, 2014.

[10]T. Taketomi, H. Uchiyama and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016", *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, 2017.

[11]"Documentation - Point Cloud Library (PCL)", *Pointclouds.org*, 2018. [Online]. Available: http://www.pointclouds.org/documentation/. [Accessed: 07- Apr- 2018].

[12]"OpenGL Programming - Wikibooks, open books for an open world", *En.wikibooks.org*, 2018. [Online]. Available: https://en.wikibooks.org/wiki/OpenGL_Programming. [Accessed: 07- Apr- 2018].

[13]*Ocw.mit.edu*, 2018. [Online]. Available: https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf. [Accessed: 07- Apr- 2018].

[14]"rgbdslam - ROS Wiki", *Wiki.ros.org*, 2018. [Online]. Available: http://wiki.ros.org/rgbdslam. [Accessed: 07- Apr- 2018].

[15]"Stanley (vehicle)", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Stanley_(vehicle). [Accessed: 07- Apr- 2018].

[16]"OpenSLAM.org", *Openslam.org*, 2018. [Online]. Available: http://openslam.org/. [Accessed: 07- Apr- 2018].