18-578 Mechatronics Final Report

Team JollyRoger (Team A, ShipBot)

David Bang
@dbang

Fiona Li
@siliangl

Sara Misra
@saram1

Haowen Shi
@haowensh

Bo Tian
@btian1

May 10 2019

Abstract

Trading across the world has not only facilitated the global economic growth, but also raised huge demand for transportation capability. Shipping remains the most economic bulk transportation method and we have seen potential in autonomously navigating ships. However, human staff is still required and it is not financially feasible to modify all ships for complete electronic control. Thus, we are sponsored by Leidos to create an autonomous robot that can manipulate maritime devices. We named our robot JollyRoger.

In this report, we explain the process of designing, building and testing a robot capable of autonomously operating electromechanical devices that are widely used on ships. The robot uses a 5 degree of freedom arm to manipulate the devices and a 4 wheel chassis to move. The arm consists of HEBI modules and a hybrid end-effector that turns different types of valves and breakers. The robot uses encoders in wheel assembly and time of flight sensors for localization and locomotion. The Mecanum wheels also allow the robot to move in different directions without having to turn. A RealSense camera mounted on chassis and a fisheye camera on end-effector are used for device recognition and end-effector position correction. We also use ROS platform and established wireless communication module to simplify the testing process.

During the test, we find out that the hybrid end-effector is able to locate valves precisely using feedback from the fisheye camera. However, more parameter tuning is required to manipulate breakers reliably.

Table of Contents

1. Project description

There has been great ongoing research in autonomous ships which navigate across long distances without much human intervention. However, human staff is still required to perform tasks like maintenance on the ship, rendering the benefits from the automated driving less useful. Our project addresses this problem by creating a robot which could autonomously operate electromechanical devices. With the addition of what we call JollyRoger, autonomous ships can be self-maintained to a greater degree, reducing the need of onboard human operators, essentially cutting labor cost while reducing risks.

This project provides a proof-of-concept of a mechatronic device capable of operating without modifying the existing human-operated devices . The device should be able to complete the tasks in the confined boundaries of the ship and the short period of allowed time to be a feasible solution to this problem of retrofitting maritime autonomy onto existing ships.

2. Design requirements

Requirement Type	**Design Requirement**
Explicit	Robot must fit within the size constraints of 1.5’ (width) x 1.5’ (depth) x 2.5’ (height)
Explicit	Robot must be to parse and process a mission file designating destination device stations and desired device orientations
Explicit	Robot must be able to manipulate devices including valve and breakers to a desired position (+/- 15 degree tolerance)
Explicit	Robot must be able to traverse any given path in the 3' x 5' testbed space
Explicit	Robot fit under budget constraints of ~$1250 in reimbursable expenses
Explicit	Robot must be built robustly and without construction from prototype kits
Explicit	Robot must be able to safely interact with the environment and will have an emergency shut off button
Implicit	Robot must be able to continuously localize itself with respect to the environment
Implicit	Robot must have the capability to detect the starting orientation of the devices
Implicit	Robot must have the capability to check the position of the devices after manipulation and make additional manipulations

Implicit	Robot must have the computing power to process computer vision algorithms in a time restricted manner
Additional	Robot will have a hybrid end effector with a combination of a granular jammer and a beam to be used on different stations depending on the device requiring manipulation at the station
Additional	Robot will have fully onboard power and remote control through a wireless network
Additional	Robot will be able to move to desired station through the shortest path
Additional	Robot will be able to complete multiple missions as scheduled timed tasks

Table 2.1: Explicit, implicit and additional design requirements for the ShipBot robot system.

3. Functional architecture

3.1 Functional Control Flow

Given a mission file at the initial position, our shipbot will parse the file and maintain a scheduler for missions.

The mission will be divided into tasks where each task is an operation on the device on a single station. The tasks will be sorted for maximum efficiency. For each task, the chassis will localize itself and locomote towards the device's station using the TOF readings and motor encoder reading feedback.

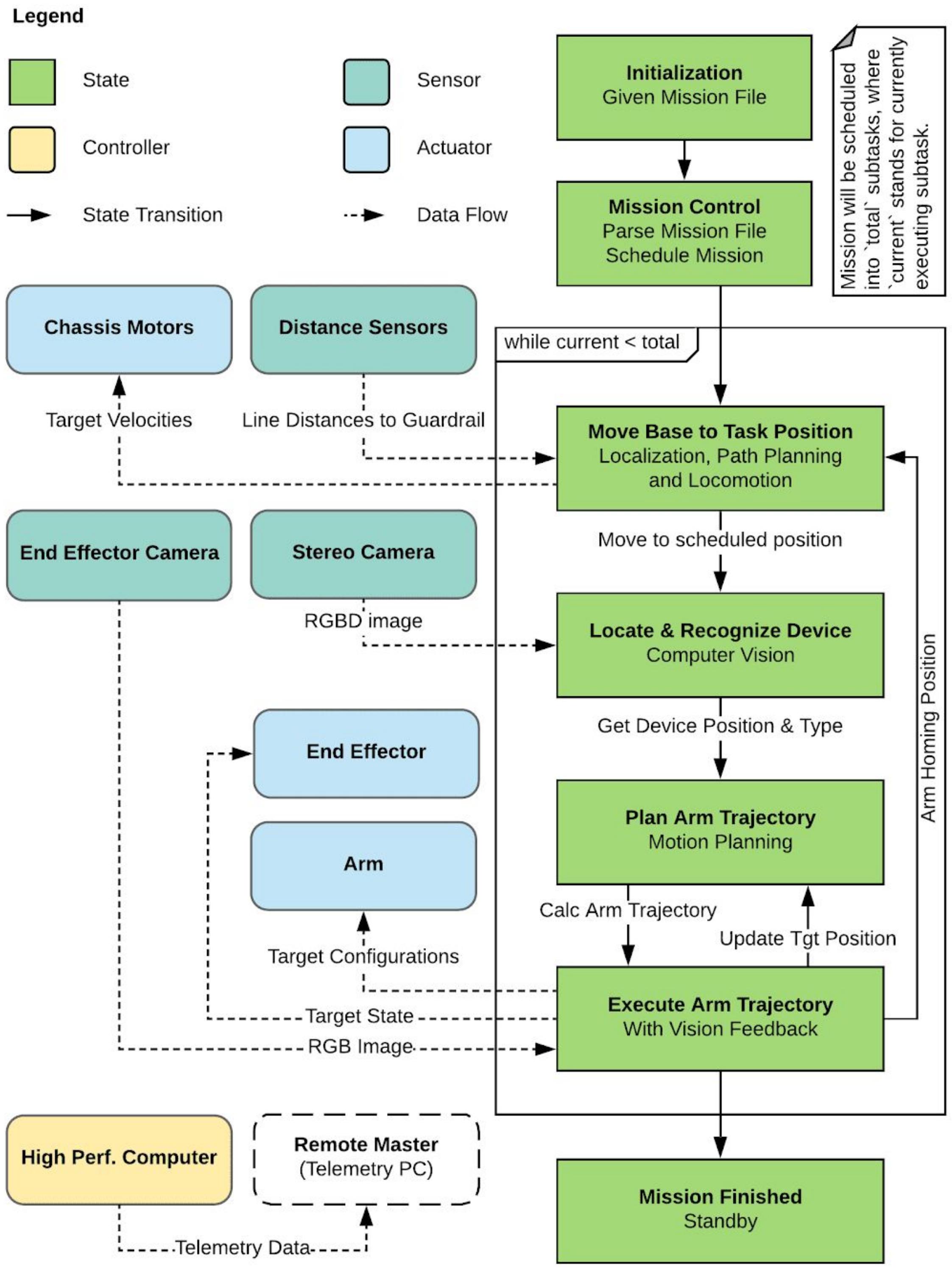
Once the chassis is at the target station, RealSense depth camera will scan the device in front of it and get the transformation + state of the device to be manipulated. The kinematics component will then use this transform to plan out an initial path towards the device. Note that there could be error in this transform and also error in arm's trajectory execution.

To deal with these potential errors, an extra camera on the end effector feeds the real time target position to the kinematics component and micro adjustments will be calculated and executed by the arm to compensate for the error.

Once the arm end effector is in a good place, it will be actuated towards the device and end effector will either rotate or translate to manipulate the target device.

The aforementioned steps will be repeated until all tasks in a mission is done. Once the current mission is done, ShipBot either stops execution informing user about task completion or continues executing of next mission if there is any.

3.2 Functional Architecture Diagram



3.2.1 Figure : Functional Architecture

4. Design concepts

In this section, localization methods, end effector, wheel type and computer vision system are each analyzed in a trade study chart. The scores for each category are assigned as an integer number between 1 and 5, with 5 being most desired state and 1 being underperformance. The weight is set to reflect a more comprehensive and focused criteria.

4.1 Localization Methods

We choose the TOF sensor as it has a high sample rate and records data in the global frame which we require to localize w.r.t. the testbed's guardrail Note, the frame of reference is the frame of the output data which is either in reference to the world (global) or in reference to the sensor itself (local). Scheme: Global Frame- 5, Local Frame- 3

	Weight	**TOF Sensor**	**Lidar**	**Ultrasound**	**IMU**	**Optical Flow**
Frame of Reference	0.125	5	5	5	1	1
Sample Rate	0.3	5	3	3	5	5
Hardware Costs	0.15	3	1	5	5	5
Computational Cost	0.125	3	1	3	3	5
Range	0.3	5	5	3	1	1
Score	**1**	**4.45**	**3.3**	**3.55**	**3.05**	**3.3**

4.1.1 Table: TOF sensor trade study

4.2 End Effector

The choice of end effector is crucial to the successful completion of our missions. It has to be versatile while reliable. We first came up with three designs, friction pad, solid beam, and granular jammer. However, each of the manipulators lack some manipulativeness in one or more tasks. Thus, we want to create a hybrid end effector that combines the strength of granular jammer and solid beam. We would also carefully design our end effector to avoid feature interference.

	Weight	**Friction Pad**	**Solid Beam**	**Granular Jammer**	**Hybrid (Jammer + Beam)**
Complexity	0.2	5	4	3	2
Handle Manip.	0.25	3	4	5	5
Valve Manip.	0.25	3	2	4	4
Breaker Manip.	0.25	1	4	2	4
Cost	0.05	5	5	3	3
Score	**1**	**3**	**3.55**	**3.5**	**3.8**

4.2.1 Table: End effector trade study

4.3 Wheel Type

The main function of the wheels is to provide an effective and fast means of locomotion for the robot system. Due to the set up of the device stations, the ability to proceed in different directions without turning would minimize our transit time between tasks. The maneuverability of the Mecanum wheel set is outstanding. With a well-implemented control algorithm, the mecanum wheel is the best fit for our application. We focus mainly on the maneuverability of the wheels. Thus, we assign a weight of 0.5 to this criteria.

	Weight	**Skid Steer**	**Mecanum**	**Tread**
Complexity	0.3	3	2	4
Cost	0.2	3	2	4
Maneuverability	0.5	2	5	2
Score	**1**	**2.5**	**3.5**	**3**

4.3.1 Table: Wheel trade study

4.4 Computer Vision System

We want to have it so our Vision System has the resolution and depth perception to correctly detect the position and orientation of the devices. With the constraint of having a materials budget, we decided to go with the Intel RealSense d435 over the Zed Stereo Camera. The lower bound for the depth range for the Kinect and Zed cameras are 0.5 meters, and we will need to see things in the testbed that are closer, so the RealSense with the smallest lower bound provides the optimal solution for our task. In addition, our team members have prior experience working with the Intel RealSense SDK.

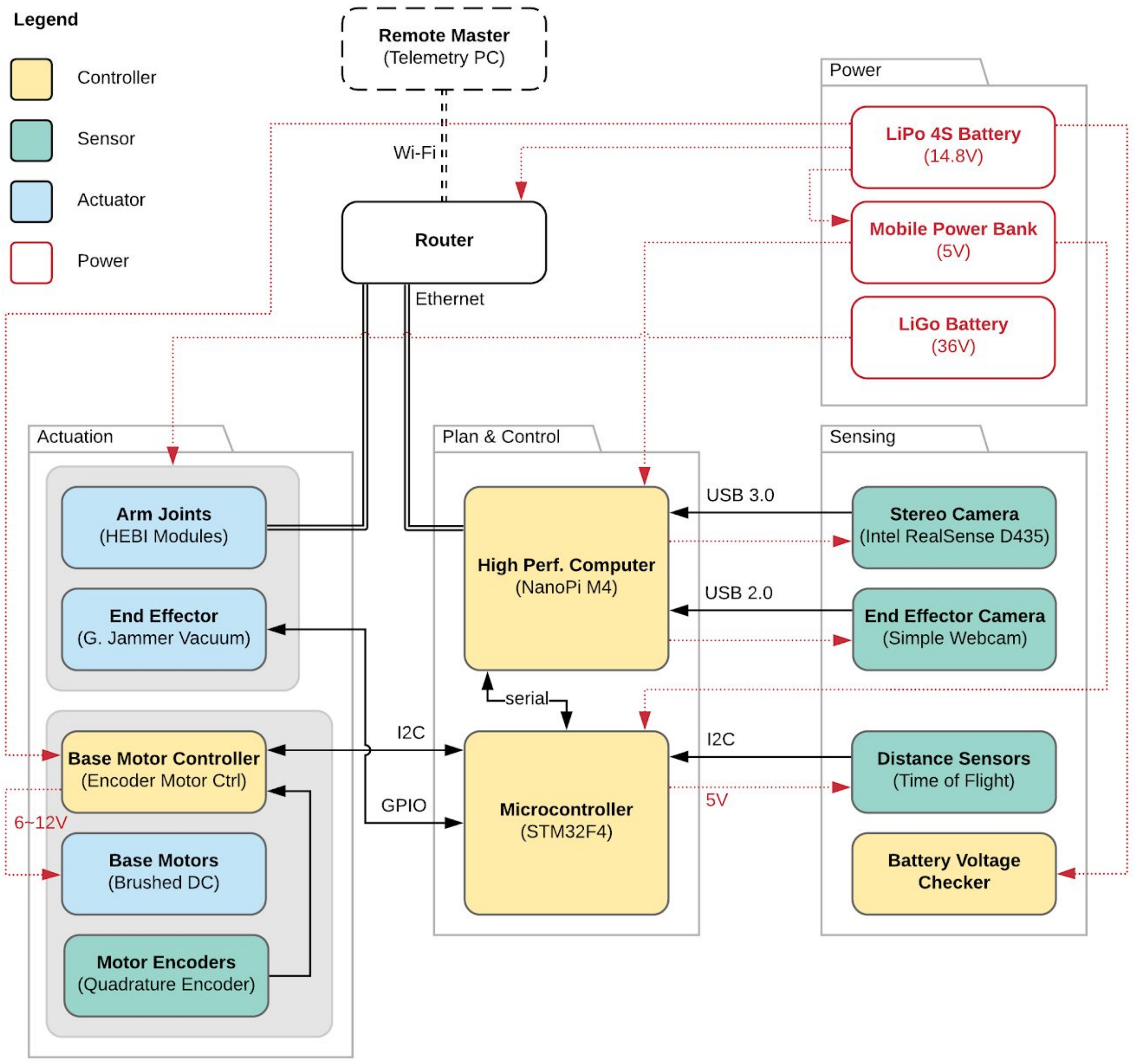
	Weight	**Microsoft Kinect[1]**	**Intel RealSense d435[2]**	**Zed Stereo Camera[3]**
Depth Range	0.3	2 (0.5 - 4.5m)	4 (0.2 - 10m)	3 (0.5 - 20 m)
Field of View	0.2	2 (H-V: 70° x 60°)	4 (H-V-D: 85.2° x 58° x 94°)	5 (H-V-D: 90°x 60° x 110°)
Video Resolution	0.2	3 (1920 x 1080 at 30 fps)	3 (1920 x 1080 at 30 fps)	4 (3840 x 1080 at 30 fps)
Depth Resolution	0.2	2 (512 x 424 at 30 fps)	4 (1280 x 720 at 90 fps)	5 (3840 x 1080 at 30 fps)
Cost	0.1	5 ($45)	3 ($179)	1 ($449)
Score	**1**	**2.5**	**3.9**	**3.8**

4.4.1 Table: Camera trade study

5. Cyber physical architecture

5.1 Electrical and Power Connection Diagram

Our system uses two onboard computers to interface with actuators and sensors. The high performance computer is a NanoPi M4 single board Linux computer that capable of performing heavy computer vision processing tasks. The other computer is a STM32F4 board responsible for controlling low-level components such as motor controller, TOF sensors and end effector. Figure 5.1.1 shows the detailed interfacing and power distribution of our electrical system.



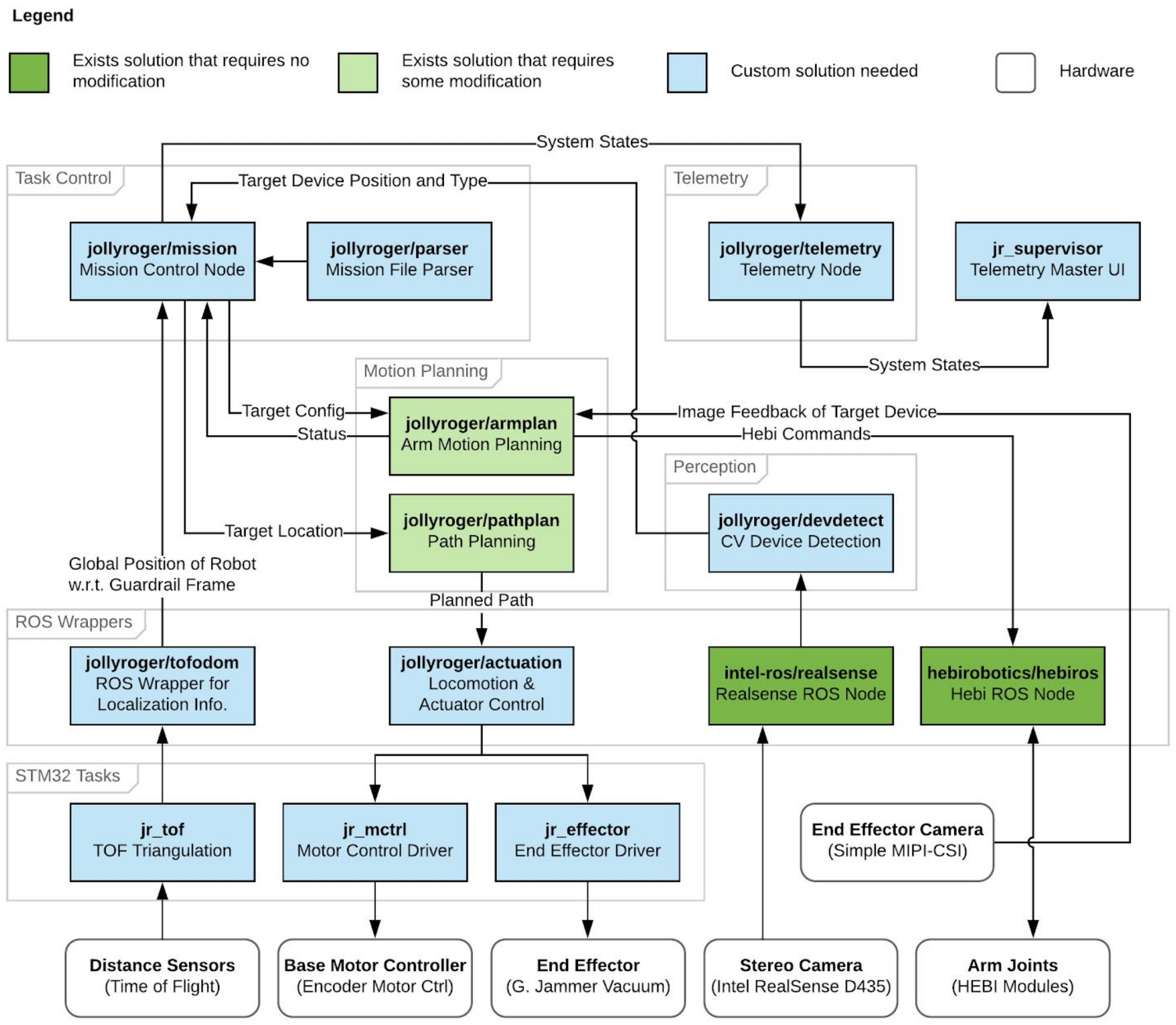
Legend
Controller
Sensor
Actuator
Power
Remote Master (Telemetry PC)
Wi-Fi
Router
Ethernet
Power
LiPo 4S Battery (14.8V)
Mobile Power Bank (5V)
LiGo Battery (36V)
Actuation
Arm Joints (HEBI Modules)
End Effector (G. Jammer Vacuum)
Base Motor Controller (Encoder Motor Ctrl)
6~12V
Base Motors (Brushed DC)
Motor Encoders (Quadrature Encoder)
Plan & Control
High Perf. Computer (NanoPi M4)
serial
Microcontroller (STM32F4)
I2C
GPIO
Sensing
USB 3.0
Stereo Camera (Intel RealSense D435)
USB 2.0
End Effector Camera (Simple Webcam)
I2C
Distance Sensors (Time of Flight)
5V
Battery Voltage Checker

5.1.1 Figure: Electrical and Power Connection Diagram

5.2 Software Architecture

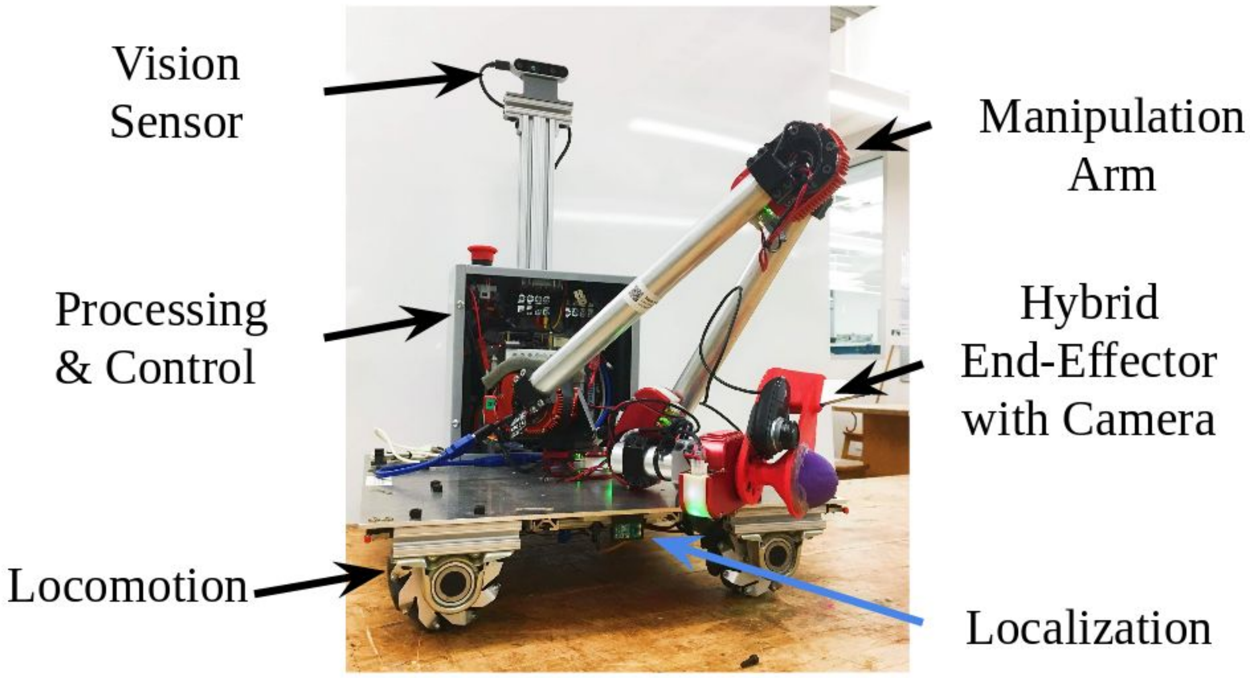
To reduce development time and efforts we decide to make our codebase ROS compliant. This will give us good software compatibility and ease of integration. The microcontroller software stack provides a nice abstraction layer for high level algorithms to work with. For example, time of flight raw data samples are triangulated on STM32 board to provide the global position of robot with respect to the guard rail frame. This offloads computation from onboard computer and also provides a easy interface for obtaining location of the robot.

The figure below (figure 5.2.1) is a detailed interaction map between each module. Note we also highlight amount of work required to get each module up and running.



5.2.1 Figure: Electrical and Power Connection Diagram

6. System description and evaluation



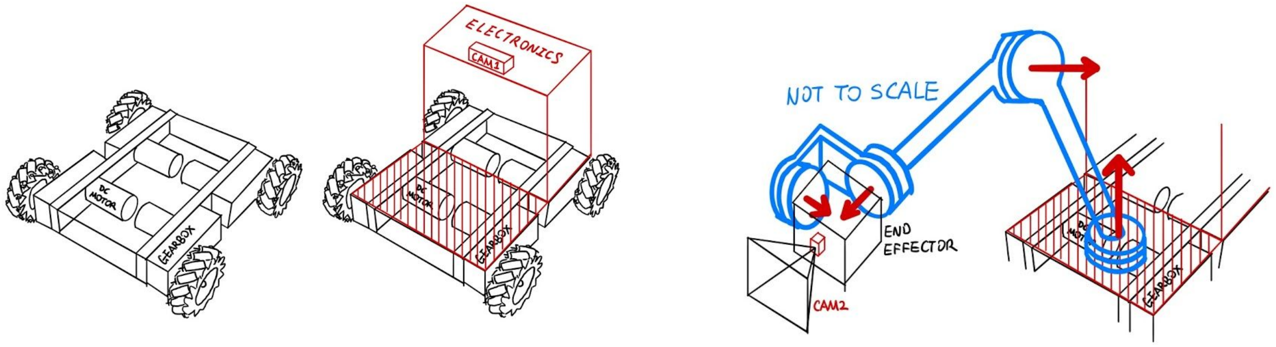
6.1.0 Figure: fully integrated system

6.1 Descriptions/Depictions

6.1.1 Chassis & Arm Subsystem

The Chassis & Arm Subsystem functions to carry the entire system on board and is also responsible for manipulating the end effector. It receives energy from battery in the Compute & Electronics Subsystem to power the movement of the entire robot and commands from the Arm Software & Algorithms Subsystem to reach the target devices. The Chassis & Arm Subsystem should be a stable and agile platform with an accurate and fast responding arm.

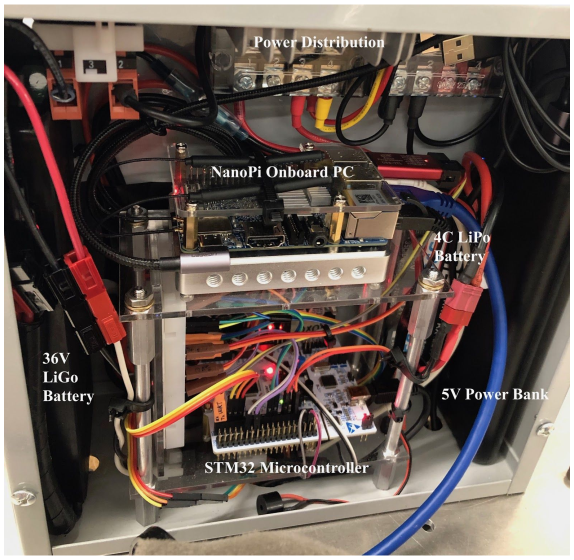
As shown in figure 6.1.1 above, the chassis has a mecanum wheel set and is omnidirectional for quick switching between stations. The electronics box towards the back also acts as a weight balancing block for the heavy arm module up front. The arm comprises of four links in our initial design and the configuration is illustrated above. The end effector is abstracted away as a box but it should contain a rotating granular jammer and a solid beam for operating on different types of electromechanical devices.



6.1.1 Figure: Chassis & Arm sketch

6.1.2 Compute & Power

The Compute & Power Subsystem provides computational and electrical power for the robot to function. The compute part consists of two onboard computers (a high performance NanoPi M4 and an auxiliary STM32 board). The power distribution system includes three separate batteries to provide different power needs from computers, arm and chassis.



6.1.1 Figure: Compute and Power subsystem integration

6.1.3 Vision Subsystem

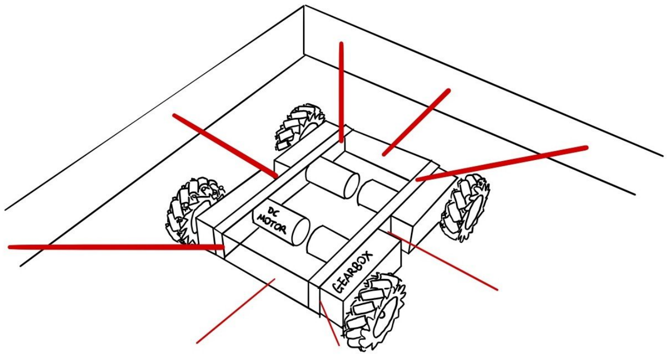
The Vision Subsystem contains a main depth camera for initial recognition and an end effector camera for closed loop control on arm's motion. This RealSense is used to determine the starting orientations of the devices and the distance from the robot arm to the device. The arm will go to an initial position based on the transform from the RealSense. The webcam on the end effector will be used to ensure that the end effector comes in contact with the device and is centered so that manipulation will be accurate.

6.1.4 Arm Software & Algorithms

The Arm Software and Algorithms Systems includes the motion planning, control and kinematics of the robotic arm to manipulate target devices. This includes the decision of strategy taken to change the state of a particular device. Since our arm has 5 degrees of freedom, motion planning and micro adjustments require real IK algorithm instead of simple joint level controls. We use Trac-IK library for calculating joint configurations (IK) and HEBI's built in trajectory API to enable a smooth motion with control over acceleration and velocity.

6.1.5 Locomotion & Localization

The Locomotion and Localization Subsystem is responsible for the movement control on base motors, in which the creation of movement trajectory is aided by the Computer Vision Subsystem, and knowing where the it is and performing moves relies heavily on Compute & Electronics Subsystem. In this project, this subsystem is primarily for realizing robot's coordinates, determining and moving to the mission testbench and performing corresponding tasks.



6.5.1 Figure: Distance sensors for localization

Figure 6.5.1 above shows how we layout a set of 8 distance (time-of-flight) sensors to get robot's position relative to the guard rails. We simplified localization system to utilize only three TOF sensors, with assumption that chassis is always parallel to one of the guide rails.

6.2 Modeling, Analysis, Testing

Chassis & Arm Subsystem

We modeled the entire arm in CAD and tried different arm length combinations to make sure that our arm subsystem can reach all the devices and operate them. We also modeled the locations of wheels on the chassis to make sure we have enough space between all the components. This is especially critical as we could not make changes to the chassis plate after we have it cut and giving the fact we have quite long motors.

Compute & Power

We modelled how to place all the electronics in the e-box including batteries, power convertors, emergency buttons, microcontroller, computer, motor controllers. As there are so many components, we first modeled it by drawing rough placement ideas on the paper, later when all the physical components arrived, we modeled by trying out layout in the box. Finally we modeled in CAD to save more space and to finalize the placement.

Vision Subsystem

We modeled the RealSense placement in CAD and manually checked the field of view from the mounted position to ensure that it could view the devices. We tested the placement of the webcam and ensured that we could view the devices from close position. We tested and calibrated the webcam closed loop control with the end effector position adjustments, in order to make the end effector correctly touched the center of the valves, so that we can correctly manipulate them.

Arm Software & Algorithms

We modeled the arm in simulation to test our motion planning. We used rVIZ through ROS, which helped us with fail-fast testing on both forward kinematics and inverse kinematics design and debugging. We also completed PID tuning to mainly reduce the error due to effect of gravity on the arm, as well as ensuring smooth, consistent motion.

Locomotion & Localization

We modeled the chassis in simulation to test localization strategies. We used VREP for localization which improved our developing speed by testing the software without the hardware chassis existence.

For TOF sensor testing, we made cables of different lengths and tested each with connections of different number of TOFs. We found that the bottleneck was the I2C wire length so in the final design, we used the shortest configuration for wires and TOF connections. For localization testing, we placed the robot onto the test bed in order to collect data for ideal TOF distance data readings for each station.

6.3 Performance Evaluation

6.3.1 Stations and runtime

The test for the final performance evaluation

Locomotion to Station			Manipulation of Station		
Station	Time	Accuracy	Device	Time	Accuracy
A	3.5s	99%	V1 Horizontal	34s	99%

B	3.5s	99%	V1 Vertical	34s	98%
C	3.5s	99%	V2	34s	99%
D	3.5s	99%	V3 Horizontal	34s	60%
E	3.5s	99%	V3 Vertical	34s	50%
G	3.5s	75%	Breaker Box A	34s	0%
H	3.5s	99%	Breaker Box B	34s	0%

6.3.2 Coolness Factor

- **Accepting multiple missions in a single mission file**
  We can take in a single file that contains multiple missions and make sure all the tasks in the first mission will be executed before the second mission.
- **Adjusting the order in which devices are visited to improve mission performance**
  We can take in a mission file and sort the tasks in the mission based on the alphabetical order so that the robot will execute tasks by going through the entire test bed one and only once from A to H.
- **User telemetry**
  The user of our shipbot could view its decisions, actions and states and even abort operations with a command line interface and also a view of end effector camera remotely on a host PC.
- **Fully onboard battery**
  We have three separate power sources to power compute, chassis and arm systems individually. We fully studied the power requirements of each subsystem and made sure we fully spec out the batteries for uncompromising performance under any reasonable workload conditions.
- **Technical sophistication**
  In terms of hardware, we built a robust chassis using strong mecanum wheels, 8020 sets and aluminum sheets. The manipulation system is an ambitious 5-DOF HEBI module arm and gives the system vast workspace while being very flexible.
  In terms of software, the low level embedded controller is rocking RTOS with clearly designed threads responsible for different hardware components. The timing and interaction between tasks is carefully designed to prevent priority inversions and deadlocks. The communication between low level controller and onboard PC is done with a custom defined packing protocol with CRC checks to ensure maximum reliability. The high level control code is mostly compliant with ROS to ease development. The critical software components such as arm kinematics is written in C++ to achieve maximum performance.

6.3.3 Failure

- **Manual interventions:**
  We had two manual interventions during the final demos and the competition. Both of them are caused by localization and locomotion.
  - The first time during the final demo was due to the I2C and UART poor wire connection. When the robot was already in the correct position, the message of it being in the correct position was not delivered successfully so it got stuck in the test bed. Right before the competition and public presentation, we detected the issue and fixed it.
  - The second time during the competition was due to the limit switch touching the test bed rail guard by accident which made the robot turned and running away. We should have a more robust software logic to handle such case.

- **End effector closed loop vision camera failure:**
  In the final demo encore, where we were trying to incorporate the end effector closed loop control, we discovered that the camera connection was lost and caused our pipeline to fault, requiring a system restart. We later discovered that the connection issue was due to a too long and not so well insulated USB extension cable. We replaced it and resolved the issue.

6.4 Strong/Weak Points

The strong and weak points will be evaluated based on each subsystem in the table below:

Subsystem	Strong/ Weak	
Chassis & Arm	Strong	Arm of 5 HEBI modules provided 5 degrees of freedom in total for arm manipulation.
	Strong	8020 along with an aluminum board to make our robot stable and strong to hold.
	Strong	All the electronics are well organized in the electronic box.
	Strong	Mecanum wheels provide 360 degrees of freedom when moving inside the test bed.
	Weak	The arm has too many degrees of freedom than needed to perform the required tasks. Thus, provided us a lot of extra work to test and adjust in order to perform missions.
	Weak	The end effector design is also complicated because both of the stick and the granular jammer require a lot of accuracy which required

		more software development than we expected.
Compute & Power	Strong	STM32 is used which is more powerful than Arduino.
	Strong	A real time operating system is implemented on STM32, which provided us a lot of freedom to schedule and control tasks on chassis, TOF and motors.
	Strong	NanoPi is used which is a more powerful computer than Raspberry Pi.
	Strong	ROS is used on the NanoPi for better subsystem integration.
	Weak	Dealing with real time operating system, UART communication between STM32 and NanoPi took way longer than we expected as both of them are more complicated hardware to work with.
Vision Subsystem	Strong	Both of the device detection with RealSense and end effector closed loop device detection and hand correction with webcam are useful and accurate enough to allow the end effector to reach the desired device.
	Weak	The webcam placement of the end effector could have been better - the current placement has both x and y offset to the end effector so manual offsetting and lots of experiments are required to make sure hand adjustments are accurate.
Arm Software & Algorithms	Strong	Three stages are created, which are arm execution that move arm to the approximate position, arm correction that calculated based on the webcam on the end effector and hand execution which performs the actual task on the device.
	Strong	PID closed loop controlled HEBI modules for the arm.
	Weak	The exact software running on 64 bit linux behaves differently from it running on 32 bit linux. We needed to bridge kinematics node to external laptop to get around this issue.
Locomotion & Localization	Strong	Feedback loop of station detection is implemented, result in extremely accurate localization.
	Strong	PID is well tuned for locomotion controls, which makes the move and turn of the robot accurate and desirable.
	Strong	Make use of three TOF sensors for localization.
	Strong	Applied Kalman Filter on TOF sensor data readings to stabilize

		hardware data output for better software controls.
	Weak	Did not make use of all 6 TOF we purchases due to I2C wire length constraints which took us a lot of time to debug, change plan and fix the problems.

7. Project management

7.1 Schedule: Available at Schedule

Task ID	Task Title	Task Owner
1	Design Proposal	All
1.1	Meeting 0: Project overview and sub-system assignment	All
1.2	Project Abstraction and Design requirements	Sara, Bo, David
1.3	Meeting 1: Project control flow, Mechanical Design, Functional and Cyber physical architecture	Haowen, Fiona
1.4	Cyber physical architecture and Battery management	Haowen, Fiona
1.5	Trade analysis on sensors, end effector, wheels and system	Sara, Bo, David, Haowen
1.6	Meeting 2: Working on the proposal together: 1) Filling out trade analysis 2) Creating and linking time and budget management chart 3) Adding diagram and description for Functional and Cber 4) Applying proposal standards and formats	All
2	Mock-Up Demonstration	
2.1	CAD Model Parts Sketch and Engineering Drawings	Bo
2.2	Base Structure Design, Arm Link Design and Rigid Finger Design on the End Effector	All
3	Sensors Lab	
3.1	Base Structure Design and Manufacturing	All
3.2	Arm Link Design and Manufacturing	All
4	Microcontroller DC, RC Servo and Stepper Motor Lab	
4.1	Wheel, Motor and Controller Integration	Bo
4.2	Monocular Camera Integration End Effector and RS Camera Integration and on the Base	All
4.3	HEBI Module Integration on Arm Joints	Haowen, Sara
4.4	Base Motor and Base Motor Encoder Integration	Haowen
4.5	TOF Sensors Integration and Calibration	Fiona, Haowen
5	System Demo #1	
5.1	RS Camera, Monocular, TOF Calibration	Sara, Haowen
6	System Demo #2	
6.1	Main Depth Camera CV System Design	David
6.2	Arm Software System Design	Sara, Haowen
6.3	Locomotion and Localization Software System Design	Haowen, Sara

7	Mid-semester Presentation	All
8	Peer Evaluation #1	All
9	System Demo #3	All
9.1	Subsystem Independent Development And Testing	All
9.2	Chassis Integration	Bo
10	System Demo #4	All
10.1	Communication between STM32 and NanoPi	Haowen, Fiona
11	System Demo #5	All
11.1	TOF sensor debugging and integration	Fiona, Haowen
12	System Demo #6	All
12.1	Localization Finalization	All
12.2	Locomotion Testing and Tuning	Haowen, Fiona
13	System Demo #7	All
13.1	Arm Implementation and Integration	Sara, Haowen
13.2	Final Subsystems Integration	All
14	Final System Demo	All
14.1	Testing and Fine Tuning	All
15	Final System Demo Encore	All
15.1	End Effector Webcam Closed Loop Control Implementation And Integration	David, Haowen
15.2	Testing and Fine Tuning	All
16	Public Presentation	All
17	Final Report	All
18	Peer Evaluation #2	All
19	Lab Clean-up	All

7.1.1 Table: Schedule with detailed work breakdown

7.2 Budget: Available at Budget

Date (mm/dd/yyyy)	Description	Vender#	Unit Price	Qty	Total Price
2/5/2019	STM32 Development Board	511-NUCLEO-F411RE	$13.00	1	$13.00
2/5/2019	TOF Sensors	511-VL53L1X-SATEL	$18.50	1	$55.50
2/5/2019	Intel Realsense D435	607-82635AWGDVKPRQ	$199.00	1	$208.79
2/7/2019	DC Motor Encoder Driver	RB-Mab-181	$49.99	1	$99.98
2/7/2019	6mm Set Screw Hubs	RB-Nex-17	$9.40	1	$57.33
2/12/2019	Mecanum Wheels	RB-Nex-13	$134.00	set of 4	$153.77
2/20/2019	BUD Industries JB-3960 Steel NEMA 1 Junction Box	JB-3960	$24.07	1	$24.07
2/20/2019	3M Pro Grade Sandpaper, 9 X 11-Inches, 220 Grit	26220CP-5-G	$15.95	20/pack	$15.95

2/20/2019	6061 Aluminum Sheet 0.19" Thick, 18" x 18"	89015K284	$72.12	18" x 18"	$72.12
2/20/2019	Male-Female Threaded Hex Standoff Aluminum, 3/8" Hex Size, 1" Long, 10-32 Thread Size	93505A178	$1.26	1	$15.12
2/20/2019	Female Threaded Round Standoff Aluminum, 1/4" OD, 3" Long, 10-32 Thread Size	93330A544	$1.69	1	$20.28
2/25/2019	Thermal Pad 145 x 145 x 1.0 mm	ACTPD00005A	$14.99	1	$14.99
3/5/2019	Low-Profile Mounted Sealed Steel Ball Bearing	5913K63	$11.78	1	$47.12
3/7/2019	8020 fastener Part No. 3495	3495	$0.44	1	$19.80
3/7/2019	8020 corner bracket Part No. 4108	4108	$2.75	1	$41.25
3/7/2019	8020 corner bracket Part No. 4115	4115	$4.05	1	$79.61
3/26/2019	NeveRest 60 12V, 105RPM, 593 oz-in Gearmotor w/ Encoder	RB-And-169	$28.00	1	$131.82
3/27/2019	8020 fastener Part No. 3416	3416	$0.40	1	$42.61
04/08/2019	Logitech C270	c270	$19.99	1	$19.99
4/30/2019	URBEST Hinge Roller Lever Micro Switches		$8.29	10pk	$8.29
				Total	**$1,141.39**

7.3 Risk Management

7.3.1 Probability and Impact Matrix

9	PROBABILITY	**9**	**27**	**45**	**63**	**81**
7		**7**	**21**	**35**	**49**	**63**
5		**5**	**15**	**25**	**35**	**45**
3		**3**	**9**	**15**	**21**	**27**
1		**1**	**3**	**5**	**7**	**9**
Low Med High		IMPACT				
		1	3	5	7	9

7.3.1 Figure: Probability and Impact Matrix

7.3.2 Risk Response Strategy

Qualitative Risk Score (X)	**Risk Response Strategy**		**Quantitative Risk Analysis Required?**
	Threat	**Opportunity**	
X < 10	Accept	Accept	No
10 ≤ X < 36	Mitigate	Enhance	Yes
36 ≤ X	Avoid	Exploit	Yes

7.3.2 Table: Risk Response Strategy

7.3.3 Risk Analysis and Response

Failure	Impact	Probability	Risk	Response	Description	Response Cost	Residual Risk
Motor malfunction	8	4	Not able to control the robot	Mitigate	Replace motors and check controllers/wires	Motor cost	Motor malfunction
Structure failure	9	2	Not able to hold the system as a whole	Mitigate	Use thicker plate and check the fasteners	Material and machining cost	Higher weight
End Effector Inaccuracy	7	5	Not being able to move the device to required state	Mitigate	Re-calibrate the arm and vision sensor. May have to redesign arm	Machining the arm/calibrating	Damage to the arm
Recognition Failure	9	3	Unable to recognize and change device state	Mitigate	Re-tune parameters/ use another recognition algorithm	Software change and tuning cost	Failure still occurs
Localization Inaccuracy	7	4	Not able to move to station / align the robot to the station properly	Mitigate	Calibrate sensors and change their configuration to assist in better localization	Calibrating and sensor cost	Failure still occurs/cost of more sensors and weight
Computing Latency	9	2	Not able to process CV algorithms in timely manner and execute mission in time constraints	Mitigate	Upgrade computing hardware. Optimize CV algorithms	Software change and hardware cost	Failure to complete mission in time
Battery Malfunction	9	1	Not able to power components of the robot. Potential battery hazard	Accept	Ensure robust circuitry. Upgrade power management board	Spare Batteries and easy to replace dock	The battery could catch on fire again

7.3.3 Table: Risk Analysis and Response

8. Conclusions

8.1 Lessons Learned

<u>Planning and scheduling</u>

One thing we did not do great on this semester is planning. Our shipbot implementation is incredibly complex and involves multiple major subsystems such as localization, locomotion, kinematics, computer vision and more. Since the system is so complex, we needed a lot of prior planning to make sure we have enough time for system integration. However, we did not realize the importance of planning and left little time for integration & testing. This left us in a bad situation because sometimes bugs could not be found until everything is put together and we discovered a lot of problems in our system in the last two weeks. To fix everything we needed to make compromises and sometimes major changes to already implemented subsystems which take a lot of time.

<u>Mechanical intelligence</u>

In this project, we also realized the importance of solving problems with mechanical solutions instead of software. For our ship bot we decided to go with a 5-DOF arm which was easy to construct using HEBI modules. The complexity of controlling a 5-DOF arm and its inherent precision problems gave us a hard time writing smart heuristics and workarounds which would not have been needed if we made the mechanical platform simpler to control, e.g. using XYZ gantry plus a simple manipulator.

Make things work first before trying to perfect them

For this project our ambition led us to make many design choices which complicate our system development a lot. For example, for interfacing with low level hardware, we could have used more accessible platforms such as Arduino, but we sticked with STM32 which is a slightly more professional platform but much harder to deal with. Putting RTOS on STM32 and making it work with a bunch of sensors / actuators plus dealing with communication has a big learning curve. In addition to that, we tried to make things even more perfect and professional by adding CRC check in serial communication, using packed data format instead of using simple ASCII strings. Another mistake we made is to try to implement particle filter for multiple sensors to achieve all-time localization. This required research level efforts and took us a lot of time to try figure out, but just to eventually give up. Many small design decisions like these complicate our system and devour our precious time to actually develop a working shio bot system. Through this project we realized the importance not to be an idealist and try to make things work before trying to make them perfect.

8.2 Future Work

The robot has had all its components installed and subsystems functioning. However, there is still future work to be done to demonstrate its full potential. Two major aspects would be computer vision robustness and arm system accuracy fine tuning.

Better computer vision algorithm would improve our accuracy in detecting device types and operating them. We are currently getting device types from the mission file. However, one of our original coolness factors is to recognize the devices automatically. We also implemented an algorithm that captures the device position relative to the RealSense camera and then transmit the information to arm subsystem for preliminary device localization. We would want to fully test and tune the algorithm to provide accurate translation matrix to the control system under different circumstances. This would allow our robot to operate in different lighting conditions and increase its reliability. The camera on the end-effector is used to make sure we can localize the device precisely. We would adjust the algorithm to make it compatible to all devices we need to operate on. This would provide us reliable close-loop control on the end-effector and make sure we can manipulate the devices to desired state.

We would also like to improve our arm subsystem if we are allowed more time to work on the robot. The arm subsystem is responsible of manipulating the devices under our command. The accuracy of its execution is essential in fulfilling the goal of our robot. We use HEBI modules to command the 5 degree of freedom arm. Due to the weight of the arm itself, we need to do gravity compensation when commanding the arm to reach certain position. We could improve our implementation on this to improve our arm accuracy. We also uses inverse kinematics in planning the arm trajectory. This can sometimes cause problem as we do not have collision detection. While adding a collision detection algorithm could be a solution, we would like to just improve on how the trajectory is planned and executed. This is because we have a relatively simple and fixed working space with known positions we need to reach. If we have the work mentioned above finished, we would improve both our devices detection and arm execution, thus advance the overall functionality of the robot.

9. References

[1] https://www.researchgate.net/figure/Technical-specifications-of-the-Kinect-v2_tbl1_321048476

[2] https://click.intel.com/intelr-realsensetm-depth-camera-d435.html

[3] https://www.stereolabs.com/zed/one

[4]https://towardsdatascience.com/kalman-filter-an-algorithm-for-making-sense-from-the-insights-of-various-sensors-fused-together-ddf67597f35e

[5] http://wiki.ros.org/trac_ik

10. Appendix: Code and Documentation

Github Repositories: Contains all the code we have written for this project

[1] https://github.com/cmu-jollyroger/jr-ros

[2] https://github.com/cmu-jollyroger/jr-stm32-rtos

[3] https://github.com/cmu-jollyroger/arm-control

[4] https://github.com/cmu-jollyroger/jr-stm32-firmware

[5] https://github.com/cmu-jollyroger/jr-base-locomotion

[6] https://github.com/cmu-jollyroger/jr-sim

Website: All the documentation of the project is on our website

https://sites.google.com/view/cmu-jollyroger/home