Department of Computer Science & Engineering
The University of Texas at Arlington

Detailed Design Specification
CSE 4317: Senior Design II
Spring 2024

MechaMocha
Erbie

Hussain Alkatheri
Christopher DeWitt
Bishal Giri
Alexis Hernandez
Phu Truong

Revision History

Revision	Date	Author(s)	Description
0.1	3.12.2024	HA	document creation
0.2	3.12.2024	HA	System Overview Completed
0.3	3.13.2024	CD	Mobile Layer Rough Draft Completed
0.4	3.20.2024	HG, PT	Command and Control Layer Draft Completed
0.5	3.26.2024	AH	Service Layer Rough Draft Completed
0.6	3.26.2024	AH	Service Layer Final Draft Completed
0.7	3.27.2024	CD	Mobile Layer Final Draft Completed
0.8	3.27.2024	HG, PT	Command and Control Layer Final Draft Completed
1.0	3.27.2024	HA	Final Review and Completion

CONTENTS

List of Figures

List of Tables

1 Introduction

Erbie is a robot that serves Arabic coffee. Directives included in the service are providing cups, pouring coffee, and collecting empty cups. Guests of the service should indicate to the robot threw hand gestures if they would like a cup of coffee, no longer want coffee, want a refill, or don't want any coffee. The robot will also be able to navigate around using a line following strategy to its service area serving one side of the line at a time. An attendant will have to make the coffee for service. The attendant would have access to a front-end service that is running on the robot which would used to monitor its current state. This would include refilling the reservoir, collecting used cups, and adding freshly cleaned cups.

2 System Overview

The system consists of 3 layers each specifiying a specific role. The highest layer would be Command and Control which provides and intermediary for all the other layers.

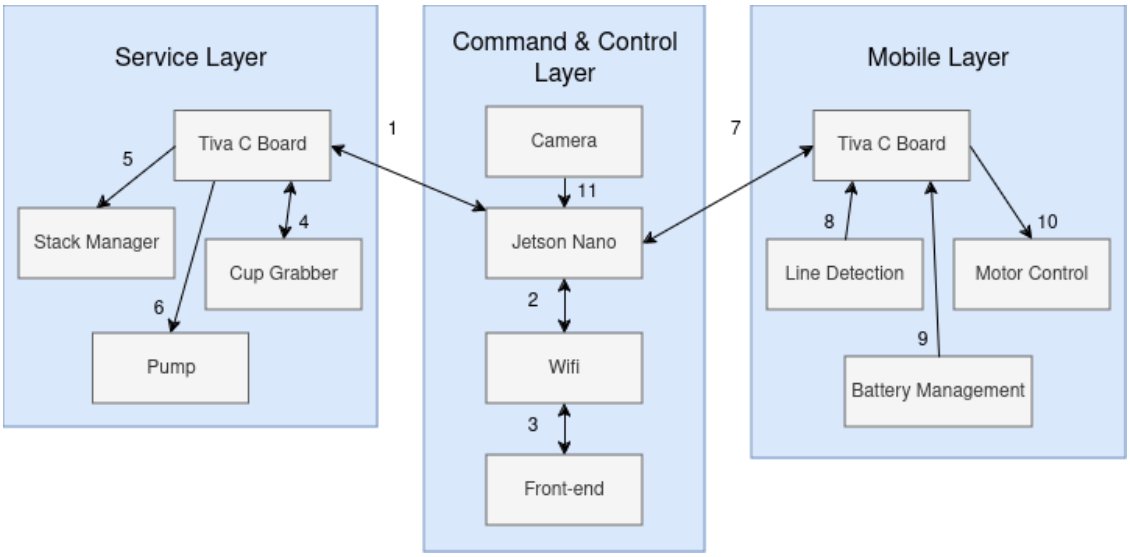

Figure 1: Erbie architectural overview

2.1 Command and Control Layer Description

This layer provides two main purposes. Being the host of the "brain" of the system that functionally commands all layers. And it hosts the external interface for the operator. This interface would display critical information to the operator about the current state of the robot

2.2 Mobile Layer Description

This layer provides two functions. First managing the motors on the robot and translating command from command and control into lower level controls for the motors. It also contains the battery which will be used to provide power to the system. Line following controls will exist at this layer with IR sensors

2.3 Service Layer Description

This layer provides the essential fucntions of the robotic service. This includes controlling the system for cup dispensment, cup collection, and pumping coffee into the cups. It will also include a translation microcontroller which will take command from command and control and respond with its state and complete the tasks given

3 Mobile Layer Subsystems

3.1 Mobile Layer Hardware

The mobile layer consists of the Battery, Motors/Motor Controller, and Sensors subsystems. With the exception of one sonar sensor placed higher on the frame, all Mobile Layer components are attached to the lowest level of the robot.

3.2 Mobile Layer Operating System

No operating system is used by the Mobile Layer.

3.3 Mobile Layer Software Dependencies

The Mobile Layer is dependent upon the code that runs the microcontroller in the Motors/Motor Controller subsystem.

3.4 Battery Subsystem

The Battery Subsystem consists of two battery packs and a DC-to-DC converter. The purpose of this susbystem is simply to provide power for the other subsystems and layers.

3.4.1 Battery Subsystem Hardware

Two Tenergy 12V NiMH battery packs provide power to an SD-50A-5 5V output DC-to-DC converter and to other components that require 12V, such as the motors in the Motors/Motor Controller Subsystem. The DC-to-DC converter provides 5V to components that are not 12V-tolerant. The batteries are attached to the robot with hook-and-loop fasteners and quick-disconnect terminal blocks for fast replacement.

3.4.2 Battery Subsystem Operating System

The Battery Subsystem does not rely on an operating system.

3.4.3 Battery Subsystem Software Dependencies

The Battery Subsystem does not directly rely on any software. However, it is worth noting that the voltage of the batteries is monitored by software in the Motor/Motor Controller Subsystem.

3.4.4 Battery Subsystem Programming Languages

The Battery Subsystem does not make use of any programming languages.

3.4.5 Battery Subsystem Data Structures

The Battery Subsystem does not make use of any data structures.

3.4.6 Battery Subsystem Data Processing

The Battery Subsystem does not process any data.

3.5 Motors/Motor Controller Subsystem

The Motors/Motor Controller Subsystem is responsible for moving Erbie. Decision-making (including orders to move, turn, and stop) is handled externally to this layer. The motor controller is responsible only for the two motors in this layer and does not control any other moving components. The controller also monitors the voltage of the Battery Subsystem and communicates with the Command and Control Layer.

3.5.1 Motors/Motor Controller Subsystem Hardware

Two Bringsmart 12V DC motors are mounted to the sides of Erbie, and one 5" diameter acrylic disc wheel is mounted to each motor. The motors are controlled by pulse width modulation (PWM) of their input current. That modulation is sent by a TM4C123GXL microcontroller at 3.3V, is passed to level shifters to bring the signal to 5V, and is then passed to two BTS7960 H-bridges (one per motor) which send the given pulses to the motors at 12V. Each motor is capable of turning its wheel forwards or backwards. The controller periodically connects the Battery Subsystem to voltage dividers and measures the voltage present to determine if the batteries need to be changed.

3.5.2 Motors/Motor Controller Operating System

The Motors/Motor Controller Subsystem does not make use of an operating system.

3.5.3 Motors/Motor Controller Subsystem Software Dependencies

The Motors/Motor Controller Subsystem is reliant on the software installed on the TM4C123GXL microcontroller. Interfacing an external computer with the microcontroller requires the use of software that is capable of using a JTAG interface (such as Texas Instrument's Code Composer Studio), which is not included with Erbie.

3.5.4 Motors/Motor Controller Subsystem Programming Languages

The Motors/Motor Controller Subsystem software is written entirely in C.

3.5.5 Motors/Motor Controller Subsystem Data Structures

The Motors/Motor Controller Subsystem communications are strings stored as null-terminated character arrays. These arrays are short-lived and are overwritten once the strings are sent or acted upon. Signals sent to the level shifters and sonar sensors and received from the IR sensors are digital GPIO signals with no further formatting.

3.5.6 Motors/Motor Controller Subsystem Data Processing

The Motors/Motor Controller Subsystem exchanges data with the Command and Control Layer via UART 8N1 protocol. All data is in the form of strings of ASCII characters which are sent and received one character at a time.

3.6 Sensors Subsystem

The Sensors Subsystem consists of sensors that relay data to the Motors/Motor Controller Subsystem's microcontroller. All sensors in this subsytem operate on 5V logic. No sensors in this subsystem are connected to any other layer.

3.6.1 Sensors Subsystem Hardware

One OPB876N51 optical sensor is mounted adjacent to an optical encoder attached to each wheel. The optical sensor output alternates high and low as the encoders turn. This output is then filtered through Schmitt triggers before being sent to the TM4C123GXL in the Motors/Motor Controller Subsystem.

Two HW201 IR sensors are mounted to the front of the robot. The IR sensors are used to follow reflective tape lines on the floor. They send digital output to the TM4C123GXL in the Motors/Motor Controller Subsystem.

Two HC-SR04 sonar transducers (one near ground level, one higher) are mounted to the center of the robot's forward face. The transducers are regularly triggered by the microcontroller in the Motors/Motor Controller Subsystem. Their outputs are digital signals sent to the same. They are used for collision avoidance.

3.6.2 Sensors Subsystem Operating System

The Sensors Subsystem does not make use of an operating system.

3.6.3 Sensors Subsystem Software Dependencies

The Sensors Subsystem does not directly rely on any software. However, it is worth noting that the output of the sensors is monitored by software in the Motor/Motor Controller Subsystem.

3.6.4 Sensors Subsystem Programming Languages

The Sensors Subsystem does not make use of any programming languages.

3.6.5 Sensors Subsystem Data Structures

The Sensors Subsystem does not make use of any data structures.

3.6.6 Sensors Subsystem Data Processing

All sensors output digital signals that are received by the Motors/Motor Controller's TM4C123GXL GPIO pins.

4 Service Layer Subsystems

4.1 Service Layer Hardware

The service layer utilizes a battery component, motors, pump.

4.2 Service Layer Operating System

No operating system is used by the service layer.

4.3 Layer Software Dependencies

The service layer is dependent on the code implemented by the microcontroller in the service layer subsystem.

4.4 Cup Grabber Subsystem

This subsystem utilizes a servo motor along with two 3D printed grippers that grab a cup from the stack manager.

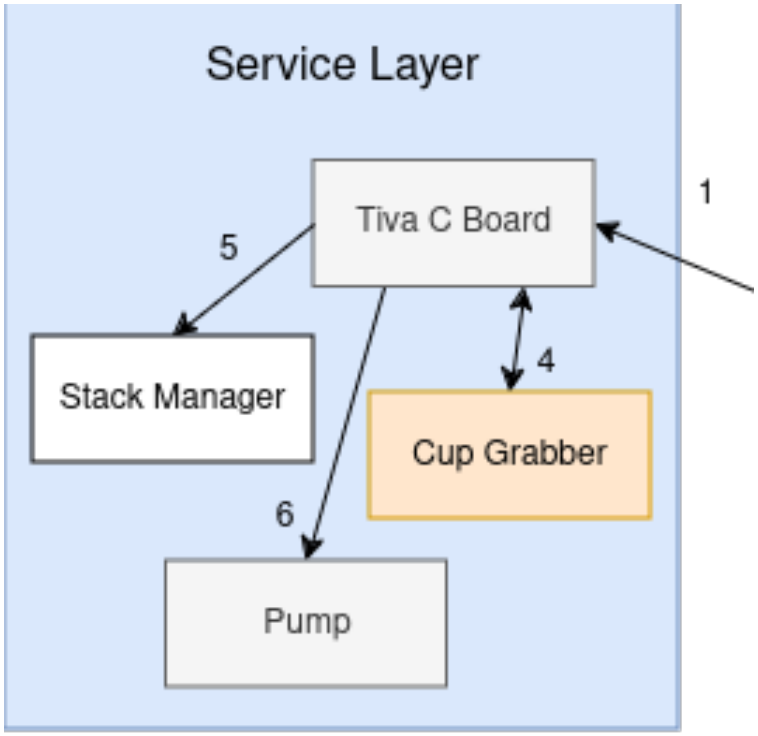

Figure 2: cupgrabber subsystem diagram

4.4.1 Cup Grabber Subsystem Hardware

One 20kg digital servo motor with two grippers attached. The grippers have gears that facilitate opening and closing with the servo motor which is controlled by PWM. The servo motor is connected to TM4C123GXL microcontroller.

4.4.2 Cup Grabber Subsystem Operating System

The cup grabber does not rely on an operating system.

4.4.3 Cup Grabber Subsystem Software Dependencies

The servo motor relies on software that is installed into the microcontroller that is used for its operation.

4.4.4 Cub Grabber Subsystem Programming Languages

The cup grabber subsystem is reliant entirely in c code.

4.4.5 Cup Grabber Subsystem Data Structures

The cup grabber subsystem does not make use of any data structures.

4.4.6 Cup Grabber Subsystem Data Processing

The servo motor is controlled by PWM. A timer peripheral is used with the microcontroller to create precise time measurements which fire off an interrupt service routine to precisely control the timing of the PWM.

4.5 Stack Manager Subsystem

The stack manager consists of a linear rail that holds a stack of cups. The purpose is simply to dispense a cup when a user ask for a new cup of coffee or to refill the cups with new ones.

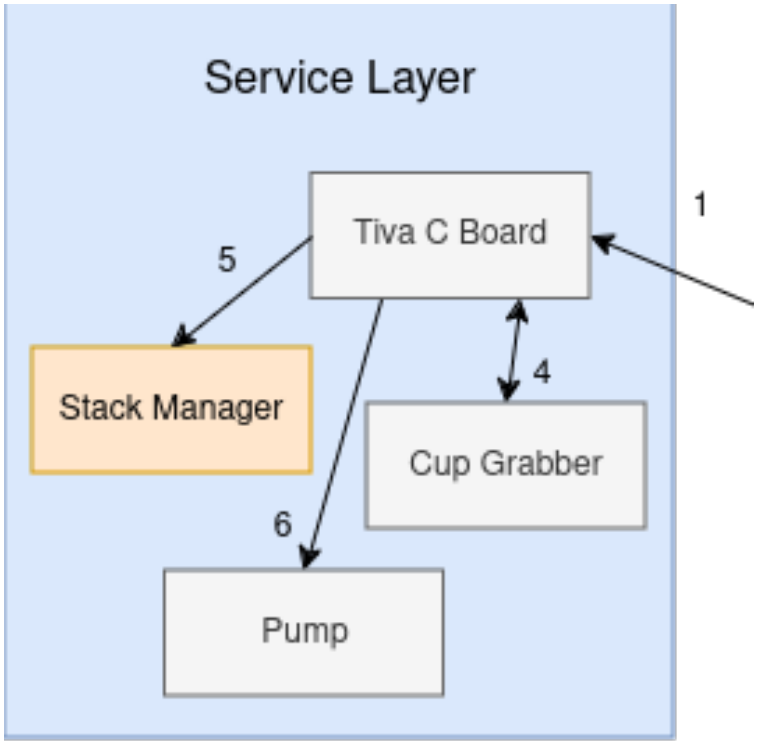

Figure 3: Stack Manager subsystem diagram

4.5.1 Stack Manager Subsystem Hardware

One linear rail along with a stepper motor and a stepper motor driver will be used for this subsystem. Powered by a 12V battery pack.

4.5.2 Stack Manager Subsystem Operating System

The stack manager does not rely on an operating system.

4.5.3 Stack Manager Subsystem Software Dependencies

The stepper motor relies on software that is installed into the microcontroller that is used for its operation.

4.5.4 Stack Manager Subsystem Programming Languages

The stack manager subsystem is reliant entirely in c code.

4.5.5 Stack Manager Subsystem Data Structures

The stack manager subsystem does not make use of any data structures.

4.5.6 Stack Manager Subsystem Data Processing

The stepper motor will be driven by signals sent to the driver with digital GPIO signals that will control the speed and direction of the motor.

4.6 Pump Subsystem

The pump subsystem will consist of a KPHM600 Micro Peristaltic Pump and is responsible for dispensing the coffee from the resevoir into the coffee cups from the stack manager.

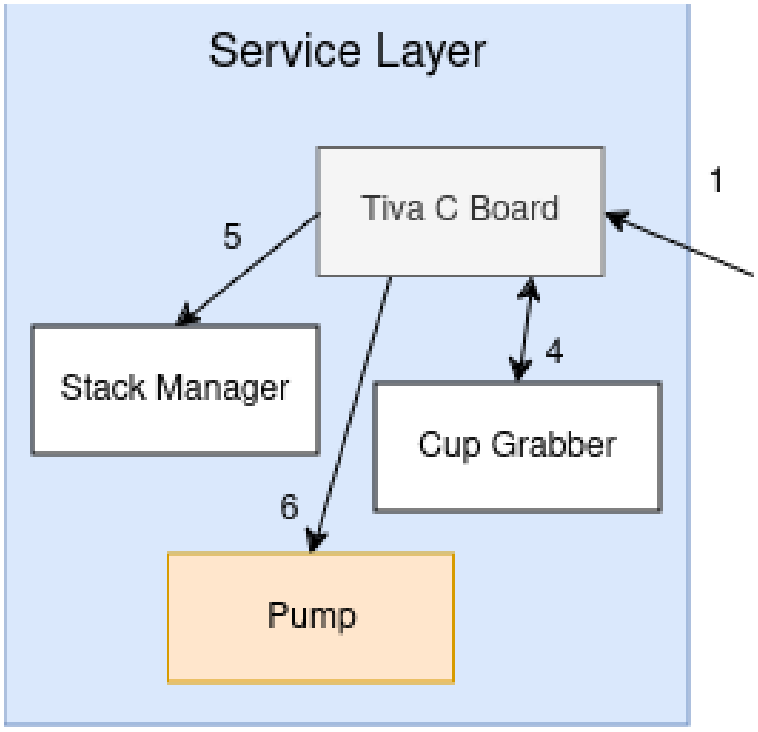

Figure 4: Pump Subsystem diagram

4.6.1 Stack Manager Subsystem Hardware

The pump consists of a DC Brushed motor that is controlled via TMC4 microcontroller and powered by a 12V battery pack source. The implementation uses a 10kohm and 47kohm resistor alongside a FQP20N06L N-channel MOSFET, and a 1N5819 Schottky diode.

4.6.2 Stack Manager Subsystem Operating System

The pump does not rely on an operating system.

4.6.3 Stack Manager Subsystem Software Dependencies

The pump relies on software that is installed into the microcontroller that is used for its operation.

4.6.4 Stack Manager Subsystem Programming Languages

The pump subsystem is reliant entirely in c code.

4.6.5 Stack Manager Subsystem Data Structures

The pump subsystem makes use of strings stored in character arrays formatted as ASCII characters that are sent via UART 8N1 protocol for debugging purposes.

4.6.6 Stack Manager Subsystem Data Processing

The pump will be driven by signals sent to the brushed DC motor using GPIO signals which will be set by a flag depending on the state of the robot.

5 Command and Control Layer Subsystems

5.1 Command and Control Hardware

The command and control layer consists of a Jetson Nano, LEDs, and a Webcam. The Jetson Nano will be responsible to controlling the overall system. It will be switching between different machine states whilst processing different communication input provided by the webcam and LED subsystems and being able to run the programs necessary to process any concurrent data.

5.2 Command and Control Operating System

The Jetson Nano is running on Ubuntu Desktop Version 18.04.06.

5.3 Layer Software Dependencies

The Jetson Nano is dependent on Flask(3.0.2), Mediapipe (0.10.9), and OpenCV (4.9.0) libraries to run the necessary code and all code will all be written in Python (3.9.18).

5.4 Web Camera Subsystem

A web camera will provide the visuals needed for the robot to recognize gestures and faces. When the camera is running and detecting gestures and faces, it's important to have a clear enough visual of the face and hands for the software to be able to recognize.

5.4.1 Subsystem Hardware

A Logitech C920x HD Pro Webcam is utilized and is responsible for video input to the Jetson Nano.

5.4.2 Subsystem Operating System

The Webcam, server and the actives processes all rely on the operating system installed on the Jetson Nano.

5.4.3 Subsystem Software Dependencies

No software is required to use the webcam as it was plug and play.

5.4.4 Subsystem Programming Languages

The webcam is utilized by the OpenCV (4.9.0) to display a live video feed.

5.4.5 Subsystem Data Structures

OpenCV an MediaPipe are responsible for face and gesture recognition. The underlying data structure would be dimensional arrays/ matrices.

5.4.6 Subsystem Data Processing

The webcam processes data through the MediaPipe(0.10.9) and OpenCV(4.9.0) libaries to recognize faces and process gesture inputs. There will be a frame packet that is transmitting data over processes which contain Header, Payload (pixel data) and Footer. We will not be altering with the existing inter process communication.

5.5 LED Subsystem

A set of LEDs will be connected to the GPIO pins of the Jetson Nano.

5.5.1 Subsystem Hardware

A set of LED light diodes are used that are connected to the Jetson Nano's GPIO pins.

5.5.2 Subsystem Operating System

The LEDs relies on the software installed on the Jetson Nano.

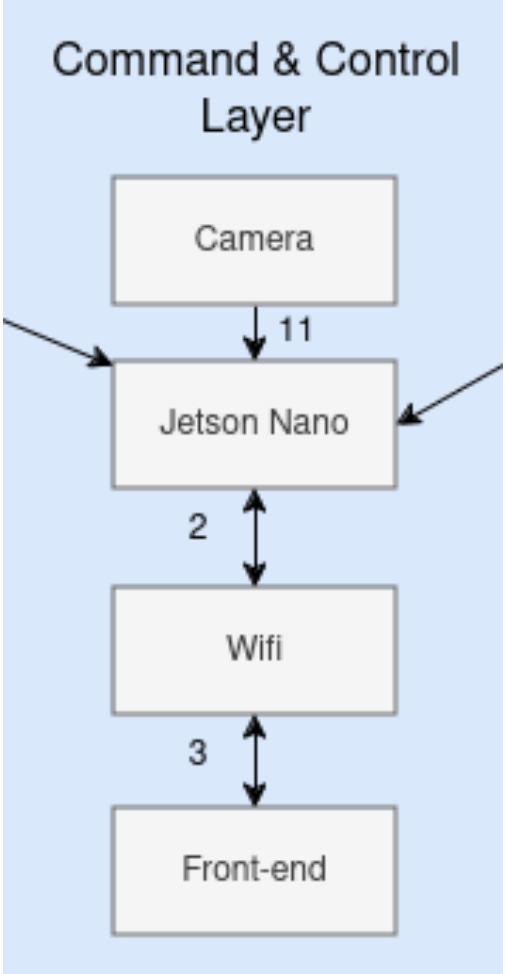

Figure 5: Command and Control subsystem diagram

5.5.3 Subsystem Software Dependencies

The LEDs require the Jetson.GPIO libary to be installed on the Jetson Nano to function properly.

5.5.4 Subsystem Programming Languages

The LED subsystem is dependent upon the code that is being run on the Jetson Nano i.e Python Flask server code which will be responsible for handling states and communicating to other systems accordingly.

5.5.5 Subsystem Data Structures

To handle/ manage the LED subsystem, the data structure for LEDs receiving commands could be a data packet consisting of information regarding the command state (on or off), LED identifier and any additional data.

5.5.6 Subsystem Data Processing

The LEDs do not process any data aside from inputs from the Jetson Nano to turn on/off.

5.6 Front End

The Front end is an HTML, CSS, and Javascript page rendered by the flask server on local-host.

5.6.1 Subsystem Hardware

There is not subsystem hardware except the Jetson Nano running the application process.

5.6.2 Subsystem Operating System

The Front End relies on the software installed on the Jetson Nano.

5.6.3 Subsystem Software Dependencies

It is dependant on the Flask Library to run the server and consequent HTML, CSS, and JavaScript to display on the browser.

5.6.4 Subsystem Programming Languages

The programming Languages present would primarily be Python (3.9.18) for the backend involving data communications and the API Layer. The static fronted template would be developed by HTML, CSS, and JS.

5.6.5 Subsystem Data Structures

There is no inherent data structure utilized. But the different API functions that handle processes triggered by the front end and communication with other subsystems.