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

Architectural Design Specification
CSE 4317: Senior Design II
Fall 2016

Team Epsilon
SCARA Robot Arm

Thomas Zachry
Diane Chin
Shayan Raza
Charlie Pan
Randy Thien Chao

Revision History

Revision	Date	Author(s)	Description
0.1	11.15.2016	DC	document creation
0.2	11.30.2016	DC	complete first draft
0.3	12.02.2016	DC	finalized tables
1.0	12.07.2016	DC	final draft release
1.1	12.09.2016	DC	added final revisions

Contents

List of Figures

List of Tables

1 Introduction

Your introduction should describe your product concept in sufficient detail that the architectural design will be easy to follow. The introduction may include information used in the first sections of your SRS for this purpose. At a minimum, ensure that the product concept, scope and key requirements are described.

1.1 Product Concept

This section describes the purpose, use and intended user audience for the SCARA Robot Arm product. The SCARA Robot Arm is a system that performs typical industrial sorting routines. Users of the SCARA Robot Arm will be able to use the SCARA Robot Arm to efficiently sort various common industrial objects. Through the use of a GUI, the user will also be able to control the arm and view both the camera feed and simulation. This way the user is able to have a product that can automate tasks such as sorting as well as be able to configure and control how it accomplishes such tasks.

1.2 Product Scope

The SCARA Robot Arm consists of the skeleton of the arm, two motors for joint movement, two encoders, a linear actuator that moves the end effector to grip objects, and an air pump. This hardware receives movement commands from the teensy board and the C code on that, which receives data from an ubuntu OS. The GUI and OpenRave are run on ubuntu, with OpenRave handling the kinematics and movement plans for the robot arm and recieving data from the camera to make its plans. The GUI takes this data and shows it on the screen as well as is responsible for sending the data into packets to the teensy board.

1.3 Key Requirements

No.	Name	Description
1	The system shall have the operational design of a SCARA Robot Arm	The design will have the design properties of a SCARA Robot Arm. The Arm will have multiple pivoting joints to aid in movement.
2	The system shall have a operating area of no more than 18 inches by 18 inches.	The operating area of the SCARA Robot ARM will have a operation area of minimum 18 inches by 18 inches.
3	The system shall be a stand alone SCARA Robot Arm with no desktop computer terminal connections for support.	The Robot arm will be designed with no outside user terminal support connections. The arm will have a built in user interface.
4	The system shall have a minimum of 2 segment links for articulation.	The Robot Arm will have at least 2 articulating segments in the arm
5	The system shall utilize available technology for computer vision processes	The Arm will use current available technologies for computer vision processes, i.e. cameras, i.r. sensors, high frequency sound emmiters, laser imagers.
6	The system shall perform current industry accepted sorting processes.	The Robot Arm will perform processes that are currently being utilised in industry, i.e. sorting, object recognition, pick and place.
7	The system shall incorporate current object recognition technologies.	The Robot Arm will use the current methods to perform object recognition in accordance with processes, i.e. openCV, simpleCV
8	The system shall be contained in 1 compact operational package.	The Robot Arm's final structural form will be one complete working package containing all components needed of the Arm.
9	The system shall contain sufficient power supply to feed all components of the SCARA Robot Arm.	The Robot Arm and all its dependent components will be supplied power through on board power supplies within the compact package.

No.	Name	Description
10	The system shall have a repeatability accuracy of minimum .050mm.	The SCARA Robot Arm will have accuracy to repeatedly focus the end effector within a range of .05mm.
11	The system shall have a resolution accuracy of +- 1mm.	The Robot Arm will have a calculating system to acquire objects within 1mm of its actual position.
12	The system shall adhere to all industry standards for machine power/ lockout safety.	The SCARA Robot Arm will adhere to all OSHA 3102 standards covering all power supplying power to machinery and all power lockout processes.
13	The system shall follow all OSHA guidelines for robotic machines.	The SCARA Robot Arm will conform to all OSHA 29 CFR 1910 guidelines for robotic equipment.
14	The system shall allow access for mandatory use period servicing.	The Robot Arm will allow for the ability to have regular interval maintenence performed.
15	The system shall contain interface for technician calibration.	AGO
16	The system shall contain method for system updates delivered through compact disk/usb flash drive.	The Robot Arm shall have the ability for software and firmware updates through either compact disk or through usb flash drive.

2 System Overview

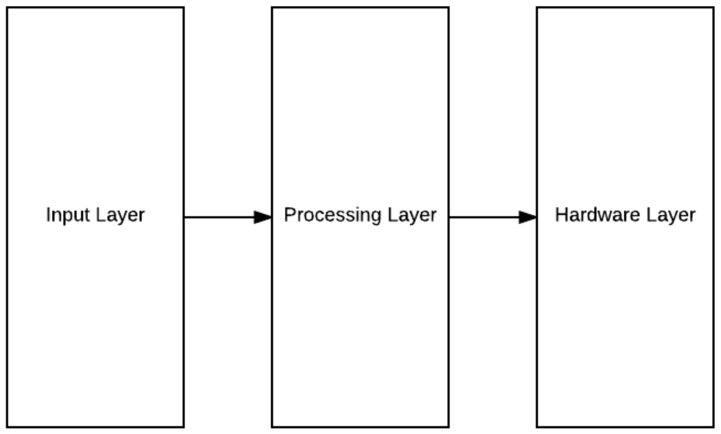

Figure 1: A simple architectural layer diagram

2.1 Input Layer

The input layer is the layer that receives information. It is comprised of the user interface and the camera input and is responsible for communicating commands to the Openrave simulation. From this layer, the user will be able to view the simulation running, the camera feed, motor positions and controls. This layer will be run on the minnow board on Ubuntu

2.2 Processing Layer

The processing layer is run on OpenRave and the C Code on the Teensy. This layer will be what moves the robot arm hardware as well as receive commands and provides simulation data and position data to the application layer. In OpenRave, the processing will calculate the kinematics and optimal movement for the robot arm, and the C Code on the Teensy board is what sends commands to the hardware layer

2.3 Hardware Layer

The hardware layer comprises of the skeletal arm, the motors, the encoders, the linear actuator, and the air pump. This layer will be carrying out the commands and operating the physical process or sorting objects and arm movement, and the pick and place operation. This layer receives the motor step commands as well as the commands for the end effector to pick up and place objects, and returns the encoder positions and linear actuator positions to the application layer.

3 Data Flow

This section illustrates how the various subsystems interact within the SCARA robot arm. Each flow of data is identified with an arrow to the direction of the subsystem it sends data towards, as well as indicates what sort of data is being transfered. The subsections provide a high level overview of the system and the data flows between each of the layers and subsystems as well as describe how individual data elements are used.

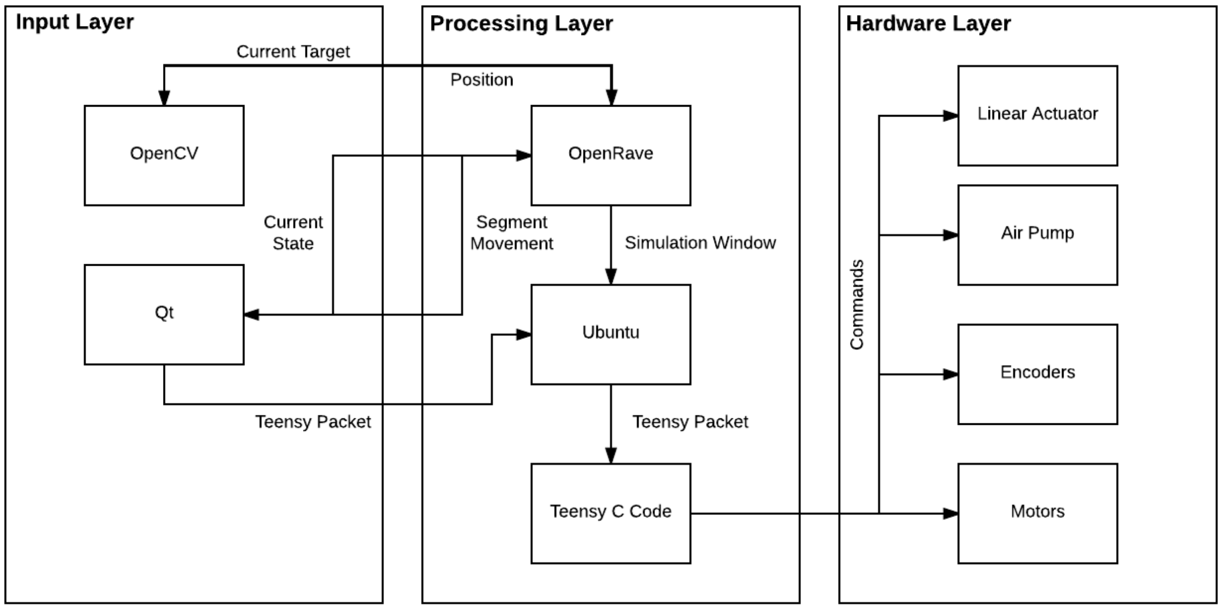

Figure 2: A data flow diagram of the arm

3.1 Data Flow Diagram

The data flow diagram consists of three layers. Data starts at the Input Layer which is inputted through but the GUI application and the camera with OpenCV. The input layer then sends to the Processing Layer which is responsible handling and calculating the input data. Then, the processing layer will send its commands to the Hardware layer, which will then execute these commands physically as output.

4 Input Layer Subsystems

In this section, the layer is described in some detail in terms of its specific subsystems. Describe each of the layers and its subsystems in a separate chapter/major subsection of this document. The content of each subsystem description should be similar. Include in this section any special considerations and/or trade-offs considered for the approach you have chosen.

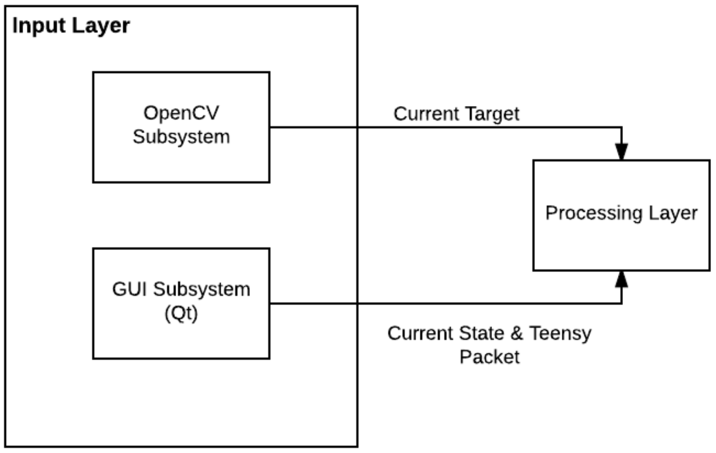

Figure 3: Input Layer subsystem diagram

4.1 GUI Subsystem (Qt)

The purpose of the GUI subsystem is to provide an interface for the user to operate and control the system, as well as see its processes.

4.1.1 Assumptions

The GUI should be able to show the camera feed as well as the OpenRave simulation on the screen, and should be able to receive and send this information

4.1.2 Responsibilities

The GUI subsystem will be responsible for rendering and displaying the user interface, along with the live camera feed and the Openrave simulation. This UI will also generate events to the system when the user interacts with anything interactible on the interface.

4.1.3 Subsystem Interfaces

Table 2: GUI interfaces

ID	Description	Inputs	Outputs
#1	Data gets stored as a data packet to send to the Processing Layer	Event generated by the application	None

4.2 OpenCV Subsystem

The OpenCV Subsystem is what is able to take and show the camera feed.

4.2.1 Assumptions

The OpenCV subsystem will be able to take the camera feed and be able to determine the current target to give to the processing layer.

4.2.2 Responsibilities

The OpenCV subsystem will be responsible for processing the camera feed and accurately pinpointing the target for the arm and send that data to the processing layer.

4.2.3 Subsystem Interfaces

Table 3: OpenCV interfaces

ID	Description	Inputs	Outputs
#1	Data sent from the camera feed	Camera feed	Sent to OpenRave subsystem

5 PROCESSING LAYER SUBSYSTEMS

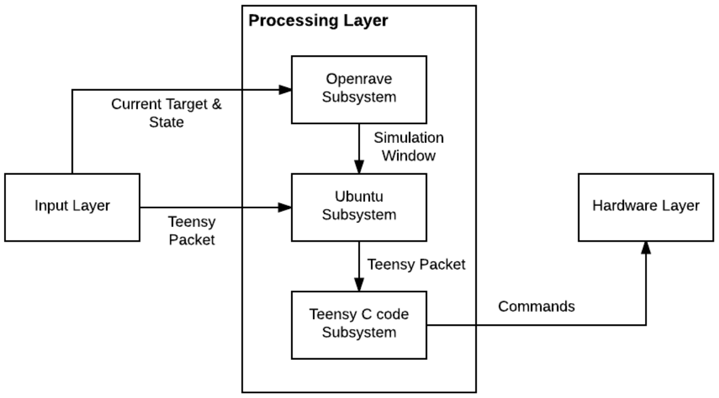

Figure 4: Processing Layer subsystem diagram

5.1 OPENRAVE SUBSYSTEM

The OpenRave Subsystem will simulate the movement of the robot arm and calculate the kinematics to find the optimal path to send to the arm.

5.1.1 ASSUMPTIONS

The OpenRave subsystem will be able to simulate the arm accurately and be able to make accurate calculations

5.1.2 RESPONSIBILITIES

This subsystem is responsible for using the simulation as well as calculate the kinematics needed to find the most accurate path for the robot arm to take, and is also responsible for communicating this data to the system.

5.1.3 OPENRAVE INTERFACES

Table 4: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Data in the form of a packet	Current target and state	None
#2	Simulation Window data	N/A	Simulation window

5.2 Ubuntu Subsystem

The Ubuntu Subsystem is what hosts the GUI and OpenRave simulation and sends data to the Teensy board.

5.2.1 Assumptions

The Ubuntu subsystem will be able to adequately host the application and be able to communicate with the Teensy board

5.2.2 Responsibilities

This subsystem is responsible for hosting the application and simulation while communicating with the Teensy board through a USB connection.

5.2.3 Subsystem Interfaces

Table 5: Ubuntu interfaces

ID	Description	Inputs	Outputs
#1	Data in the form of a packet	Teensy Packet (from the GUI)	Teensy Packet (to the Teensy)
#2	Simulation Window data	N/A	Simulation window display

5.3 Teensy C-Code Subsystem

The Teensy C Code subsystem has the code to send commands to the hardware.

5.3.1 Assumptions

The Teensy C Code subsystem will be coded to send commands to operate the hardware accurately.

5.3.2 Responsibilities

This subsystem is responsible for receiving the packet data and using it to communicate with and send commands to the hardware subsystems to operate them.

5.3.3 Subsystem Interfaces

Table 6: Teensy interfaces

ID	Description	Inputs	Outputs
#1	Data in the form of a packet	Teensy packet data	None
#2	Command data	N/A	Commands

6 Hardware Layer Subsystems

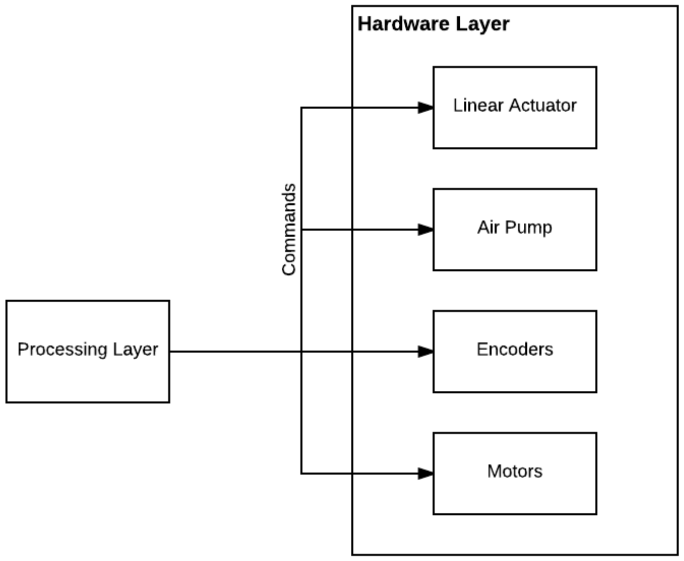

Figure 5: Hardware Layer subsystem diagram

6.1 Linear Actuator Subsystem

The linear actuator is what moves the end effector down and up in order to grip an object

6.1.1 Assumptions

The subsystem will be able to receive commands in order to know when to operate its movement

6.1.2 Responsibilities

This subsystem is responsible for receiving its commands and making its movements when it should in a timely manner.

6.1.3 Subsystem Interfaces

Table 7: Linear Actuator interfaces

ID	Description	Inputs	Outputs
#1	Command Data from Teensy	Commands	Actuator movement

6.2 Motor Step Movement Subsystem

The Stepper Motors are what moves the joints of the robot arm.

6.2.1 Assumptions

The subsystem will be able to receive commands in order to know when to operate its movement

6.2.2 Responsibilities

This subsystem is responsible for receiving its commands and making its movements when it should in a timely manner.

6.2.3 Subsystem Interfaces

Table 8: Motor interfaces

ID	Description	Inputs	Outputs
#1	Command Data from Teensy	Commands	Motor movement

6.3 Encoder Position Subsystem

The Encoder subsystem should be able to take the angle of the arm joints and communicate the angle positions in order to provide motion control.

6.3.1 Assumptions

The subsystem will be able to receive commands in order to know when to operate its control

6.3.2 Responsibilities

This subsystem is responsible for receiving its commands and controling the movement accurately.

6.3.3 Subsystem Interfaces

Table 9: Encoder interfaces

ID	Description	Inputs	Outputs
#1	Command Data from Teensy	Commands	Encoder Position

6.4 Air Pump Subsystem

This subsystem receives commands to operate the vacuum air pump which manipulates the gripper and allows the gripper to hold onto objects.

6.4.1 Assumptions

The subsystem will be able to receive commands in order to know when to operate vacuum.

6.4.2 Responsibilities

This subsystem is responsible for receiving its commands and operating when it should in a timely manner.

6.4.3 Subsystem Interfaces

Table 10: Air Pump interfaces

ID	Description	Inputs	Outputs
#1	Command Data from Teensy	Commands	Air pump operation