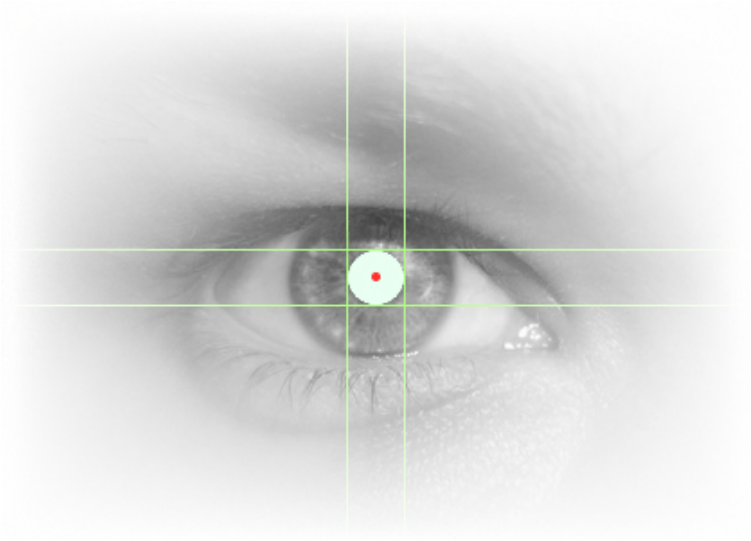
Department of Computer Science & Engineering
The University of Texas at Arlington

Arcitectural Design Specification
CSE 4316: Senior Design II
Fall 2016



Project Iris
Iris

Neil Simmons
Tyler D'Spain
Matthew Zielke
Syed Uddin
Tony McDonald

Revision History

Revision	Date	Author(s)	Description
0.1	12.14.2016	NS\TS\MZ\SU\TM	document creation

CONTENTS

List of Figures

List of Tables

1 Introduction

Project Iris is a software solution that will work in conjunction with an Intel RealSense SR300 camera to provide 3D gaze tracking information to a host computer. Ultimately the target for this solution are developers looking to incorporate 3D gaze tracking into their existing applications, or develop new applications around the added functionality Project Iris can provide. Project Iris will be open source and provided via a GitHub repository to any developer who wishes to use it.

1.1 Purpose and Use

Project Iris will provide 3D gaze tracking information to a host computer in the form of x and y coordinates similar to the way an operating system might provide mouse coordinates to a program. To accommodate this Project Iris will also provide a calibration utility and a screen overlay to indicate where the user is currently looking. This is to be used in conjunction with other software which is outside the scope of Project Iris, but could include mouse emulation, gaming, and efficiency/process improvements for desktop applications. In addition project Iris will also provide a demo program so that users and developers can get a sense of what the camera is seeing.

1.2 Intended Audience

Project Iris is not intended to be a standalone solution, but a tool developers of other software can include to create new and immersive user experiences. Game developers, application developers, and developers of accessibility software are all part of the target audience for Project Iris.

1.3 Features & Functions

The code provided by project Iris will include a calibration utility, an onscreen indicator of where the user is currently looking, and an API that will return the x and y coordinates of the user's current gaze position on the screen. Project Iris will be Open Source software intended for use on a Windows operating system and will require an Intel SR300. Additionally project Iris will provide software that allows users and developers tu run a demo mode program to view the output of the camera's video feeds to get a sense of what the application is doing.

1.4 External Inputs & Outputs

Project Iris will work by using the data gathered from an Intel RealSense SR300 camera mounted as near the horizontal center and vertical top of a computer monitor as possible. The system will expect to be able to reconcile a user's eyes and facial landmarks from a distance of .2 - 1.5 meters. After a calibration procedure this data will be translated into x and y coordinates for use in other applications. In addition to the calibration utility and coordinate outputs via an API, the system will allow the developer to toggle on an indicator to visualize the current gaze position.

1.5 Product Interfaces

Project Iris will provide a calibration utility that will place a dot at different positions on the computer monitor (figure 2) and then use this data to output an estimate of the user's current gaze position relative to the screen. The position will be provided via an API call in the form of a pair of integers representing the estimated x and y coordinates of the user's gaze. In addition, the system will provide a toggle via the API to turn on a translucent dot that will allow for the visualization of the current estimated gaze position (figure 3).

2 System Overview

Project Iris will consist of four major layers. Tracking, Viewer, Calibration, and the Device.

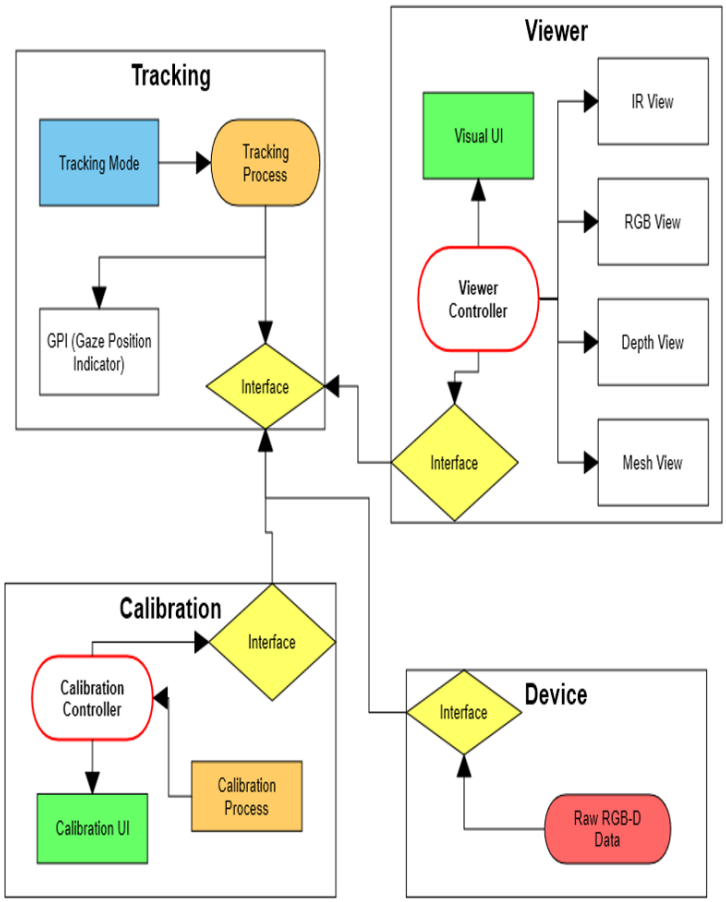


Figure 1: Project Iris architectural layer diagram

2.1 Tracking Description

The tracking layer is the most important layer of the architecture. It is where the most important work will be done for Project Iris. This layer consist of the tracking process, which determines where on the screen the software believes the gaze is located. This layer uses calibration data from the calibration layer, data from the device layer, and receives commands from the interface (i.e. start/stop).

2.2 Viewer Description

The viewer layer provides a GUI for the user, and a way to control Project Iris. It receives camera data trough the tracking layer and can display it for the user.

2.3 Calibration Description

The calibration layer outputs data to the screen with a UI and uses data from the device layer to build a calibration profile of a user. This calibration data is the given to the tracking layer.

2.4 Device Description

The device layer consist of the drivers for the SR300. This layer is the simplest as its interactions with the other layers are purely to send its data to them.

3 SUBSYSTEM DEFINITIONS & DATA FLOW

Project Iris will use a series of interfaces on each of it's layers in order to facilitate communications between them. Each layer will be implemented independently and relie on their stable interfaces to communicate information.

4 Tracking Subsystems

The tracking layer will consist of three subsystems and an interface. These subsystems are the tracking mode, tracking process, and GPI (Gaze Position Indicator).

4.1 Tracking Mode

The tracking mode primarily consist of two states. Either controlling the mouse, or simply returning a set of screen coordinates that indicate the gaze position on the screen.

4.1.1 Assumptions

Project Iris has assumed that windows is the operating system being used, and that it's mousing API will not change significantly.

4.1.2 Responsibilities

It is this layers responsibility to determine if a set of coordinates are made available for API calls, or if the tracking algorithm is actively updating the systems mouse position.

4.1.3 Subsystem Interfaces

Table 2: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Tracking Mode/Tracking Process		Mode

4.2 Tracking Process

The tracking process is the most important subsystem in all of Project Iris. It is where the gaze tracking algorithm will reside and also system updating will occur.

4.2.1 Responsibilities

It is this layers responsibility to actively track the users gaze position using data from the device layer and mode information from the mode layer.

4.2.2 Subsystem Interfaces

Table 3: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Tracking Process/Tracking Mode	Mode	
#2	Tracking Process/Interface	Raw Data Settings Changes	Status

4.3 GPI (Gaze Position Indicator)

The GPI subsystem will take the position provided by the tracking process and display a reticle on the screen that indicates position.

4.3.1 SUBSYSTEM INTERFACES

Table 4: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Tracking Process/GPI	Gaze position	

5 Viewer Subsystems

The viewer layer makes up the UI of Project Iris. It will expose settings and raw data to the user to facilitate a better understanding of what is taking place behind the scenes.

5.1 Visual UI

The visual UI is the main component of Project Iris that a user will see. It will allow the user to change settings, calibrate the software, and see the raw data generated by the software. All changes will then go through the controller.

5.1.1 Assumptions

The UI will be built using windows 10 APIs.

5.1.2 Subsystem Interfaces

Table 5: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	UI/Controller	Device Data Software State	User Selections

5.2 Viewer Controller

The controller is where an algorithms or data changes the UI needs to make are accomplished. This will facilitate seperating the Windows UI elements from the rest of Project Iris and should improve the ability of the project to be ported later.

5.2.1 Responsibilities

This subsystem is responsible for updating the Tracking layer, calling the calibration utility, and ensuring important events are relayed to the user.

5.2.2 Subsystem Interfaces

Table 6: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Controller/UI	User Selections	Device Data Software State
#2	Viewer interface	Device Data System Events Software State	Settings Changes

6 Calibration Subsystems

The calibration layer is responsible for the calibration of the software to a specific user. This data is crucial to obtaining acurate gaze tracking information and displaying it for the user/system.

6.1 Calibration UI

The calibration UI displays information to the user so that that the calibration process can calculate accurate data. This is achieved by moving a dot around on the screen for the user to follow with their eyes.

6.1.1 Assumptions

The os of the system the UI is displayed on will be windows 10.

6.1.2 Responsibilities

This subsystem must accurately display the dot on the screen at the proper place and for the appropriate amount of time.

6.1.3 Subsystem Interfaces

Table 7: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Controller/UI	Begin	N/A

6.2 Calibration Process

The calibration process receives camera information via its controller, and information about when the UI subsystem began in order to build accurate calibration information about the user.

6.2.1 Assumptions

The UI always performs its task in the alloted time-frame expected by the calibration process.

6.2.2 Responsibilities

Build accurate calibration data about the user.

6.2.3 Subsystem Interfaces

Table 8: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Calibration Process/Controller	Camera Data Begin	Calibration File

6.3 Subsystem 3

The calibration controller is responsible for coordinating the beginning of the calibration process, passing information to the calibration process, and ensuring the delivery of calibration data back to the rest of the system. This model allows the UI or Process to change independently of each other, all while maintaining a seamless interface for the rest of the system.

6.3.1 Responsibilities

Begin the calibration process in a timely manner and return accurate calibration data to the rest of the system.

6.3.2 Subsystem Interfaces

Table 9: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Controller/interface	Device Data Begin	Calibration File
#2	Controller/UI	N/A	Begin
#3	Controller/Process	Calibration Data	Begin Device Data

7 Device Subsystems

The device layer is the simplest layer in the system, and consist of the drivers for the SR300 and the interface for it.

7.1 Data

The data subsystem only sends its data out to the interface.

7.1.1 Subsystem Interfaces

Table 10: Subsystem interfaces

ID	Description	Inputs	Outputs
#xx	Data/Interface	N/A	Data Stream

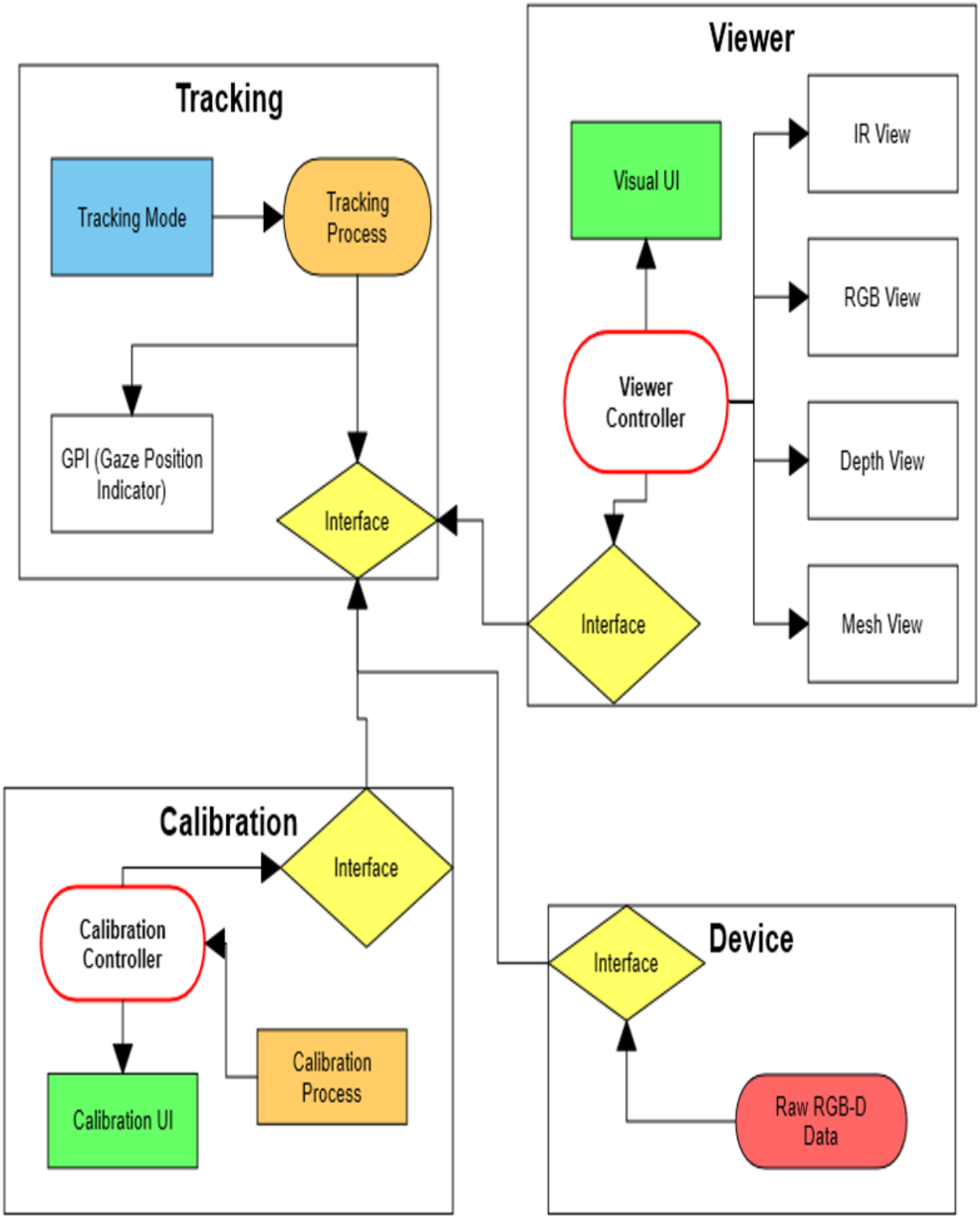


Figure 2: A simple data flow diagram

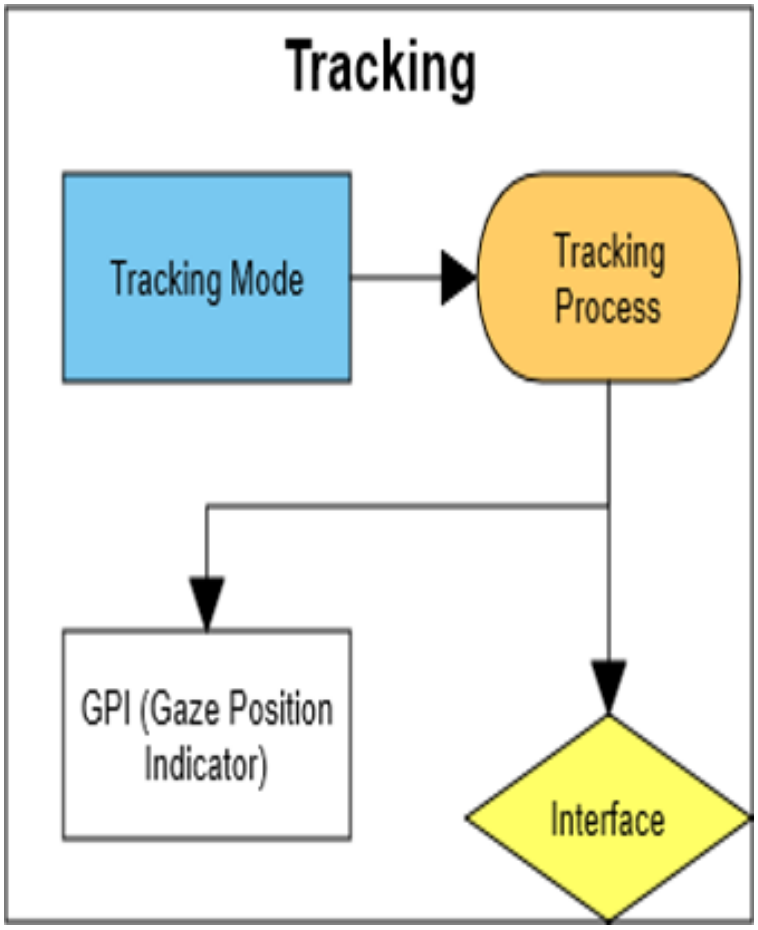


Figure 3: Example subsystem description diagram

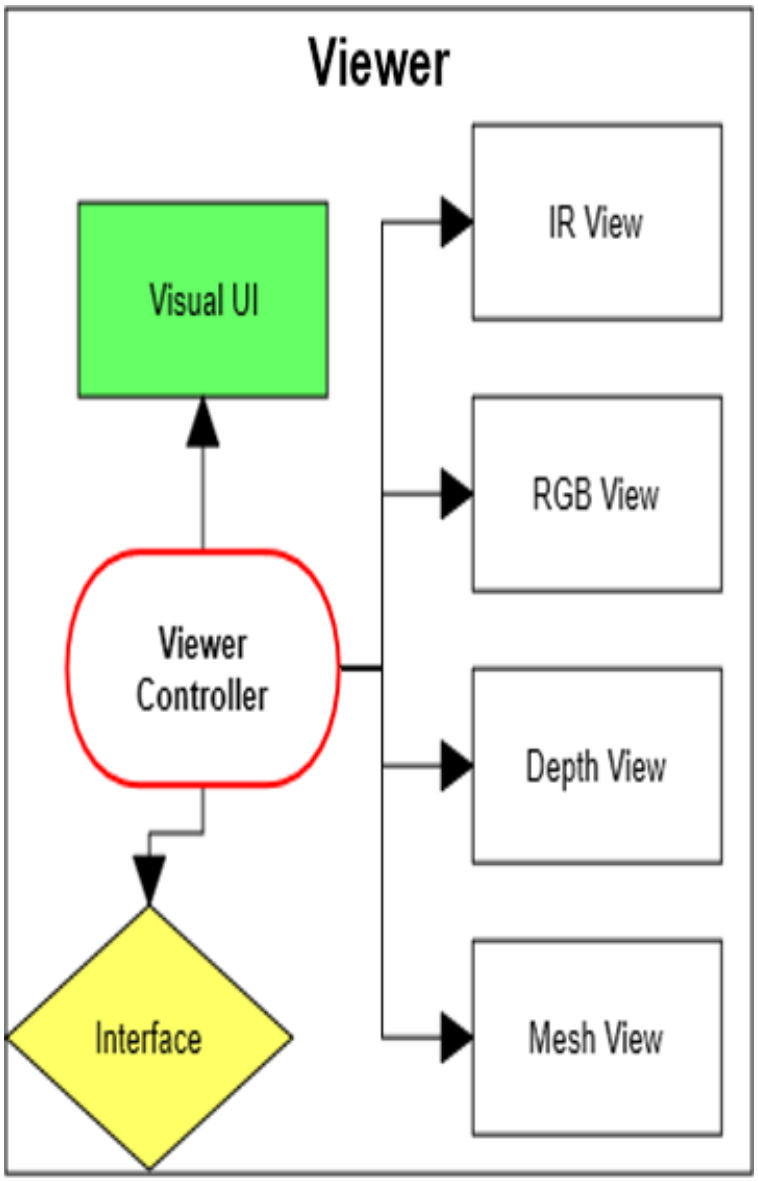


Figure 4: Example subsystem description diagram

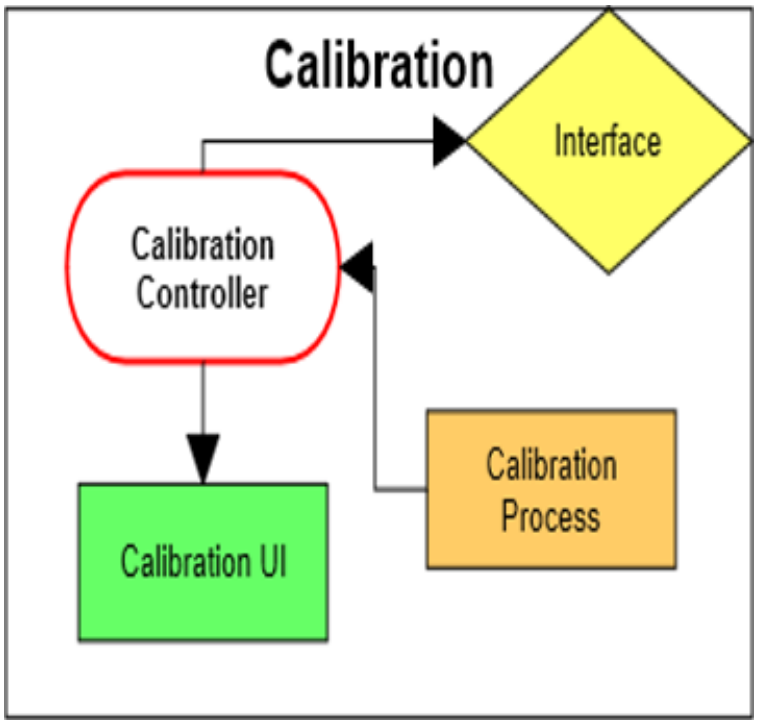


Figure 5: Example subsystem description diagram

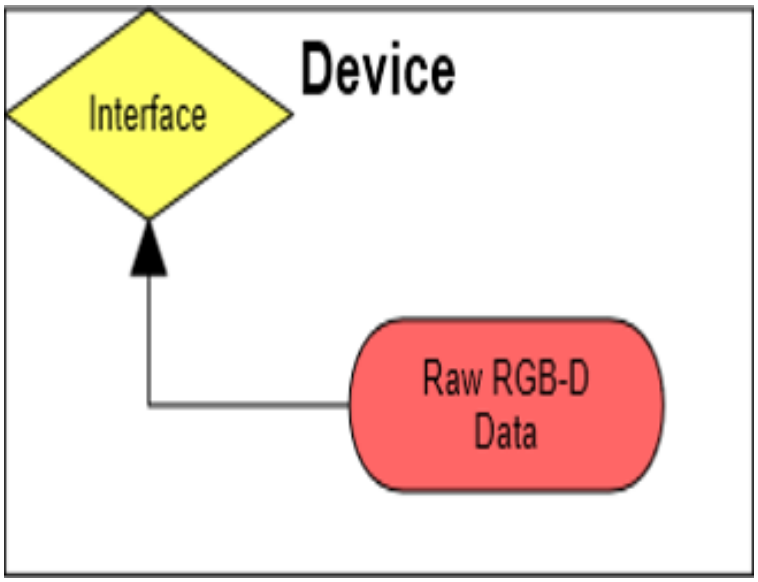


Figure 6: Example subsystem description diagram