Project Infrared

Architecture/Design Document

**Table of Contents**

[1 Introduction 3](#_Toc131876418)

[2 Design Goals 4](#_Toc131876419)

[3 System Behavior 4](#_Toc131876420)

[4 Logical View 5](#_Toc131876421)

[4.1 High-Level Design (Architecture) 5](#_Toc131876422)

[4.2 Mid-Level Design 6](#_Toc131876424)

[4.3 Detailed Class Design 8](#_Toc131876425)

[5 Process View 11](#_Toc131876426)

[6 Development View 12](#_Toc131876427)

[7 Physical View 12](#_Toc131876428)

[8 Use Case View 12](#_Toc131876429)

Change History

**Version:** <1.0>

**Modifier:** <Richard Chan>

**Date:** 06/04/2017

**Description of Change:** Initial release

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

**Architecture and Design**

The purpose of the architecture/design document is to explain the organization of the code. A well-written architecture document will make it easier for new programmers to become familiar with the code.

The architecture/design document should identify major system components and describe their static attributes and dynamic patterns of interaction.

Software architecture and designs are typically expressed with a mix of UML models (class and sequence diagrams being the two most common) and prose. Dataflow diagrams are also helpful for understanding the interaction between components and overall flow of data through the system.

**About this Template**

This template suggests one way of documenting a software system’s architecture/design. You aren’t required to include every section in this template nor all the content in the sections you do include. However, the document you do submit should pass the following checklist:

* Are design objectives clearly stated? For example, if performance is more important than reusability, this should be made clear at the start of the design specification.
* Does the architecture partition the implementation into clearly defined subsystems or modules with well-defined interfaces?
* Does the architecture express in a clear way the main patterns of communication between subsystems and modules?
* Does the architecture satisfy the requirements?
* Is the architecture traceable to requirements?
* Any models created should either be expressed with a well-known modeling language, or if a well-known modeling language isn't used, the syntax and semantics of the symbols that are used should be defined.

This document describes the architecture and design for the Project Infrared application being developed for Richard Chan. The finished product should be able to record infrared signals, such as those produced from TV remote controls, and send recorded signal. The 2x8 LCD display should provide menu options for the following items: Play, record and clear.

Development Estimate:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Start Date | Estimated Hours | Actual Hours | Completion Date |
| Develop LCD menu, including interaction with pushbuttons (debouncing push buttons) | 6/15/17 | 4 | 3 | 6/15/17 |
| Design states and option trigger function | 6/16/17 | 2 | 2 | 6/18/17 |
| Set up Record function to receive infrared signals (Can use code previously developed) | 6/19/17 | 4 | 3 | 6/25/17 |
| Set up Play function to send infrared signals  Note: Might remove infrared LED capability. | 6/19/17 | 3 | 6+ |  |
| Set up Clear function to delete all stored signal data | 6/21/17 | 1 | 1 | 6/25/17 |
| Add feature to save recorded signal to EEPROM (Maintain data after power cycle) | 7/19/17 | 3 | 2 |  |
| Set up cloud IOT account (Maybe Azure) | 8/8/17 | 1 | 1 | 8/8/17 |
| Install ESP8266 into the design eagle diagram. |  | 1 |  |  |
| Finish Record function’s IOT communication part. Encode signal and send to cloud. |  | 5 |  |  |
| Finish Play function’s IOT communication part. Send recorded date and event to cloud. |  | 2 |  |  |
| Finish Record function’s IOT communication part. Send recorded data and event to cloud. |  | 2 |  |  |
|  |  |  |  |  |

Planned Development Milestone:

|  |  |
| --- | --- |
| Date | Achievement |
| 6/14/17 |  |
|  |  |
|  |  |
|  |  |
|  |  |

*The purpose of this document is to describe the architecture and design of the Project Infrared application in a way that addresses the interests and concerns of all major stakeholders. For this application, the major stakeholders are:*

* *Users and the customer – they want assurances that the architecture will provide for system functionality and exhibit desirable non-functional quality requirements such as usability, reliability, etc.*
* *Developers – they want an architecture that will minimize complexity and development effort.*
* *Project Manager – the project manager is responsible for assigning tasks and coordinating development work. He or she wants an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies. For this to happen, the modules need well-defined interfaces. Also, because most individuals specialize in a particular skill or technology, modules should be designed around specific expertise. For example, all UI logic might be encapsulated in one module. Another might have all business logic.*
* *Maintenance Programmers – they want assurance that the system will be easy to evolve and maintain on into the future.*

The architecture and design for a software system is complex and individual stakeholders often have specialized interests. There is no one diagram or model that can easily express a system’s architecture and design. For this reason, software architecture and design is often presented in terms of multiple views or perspectives [IEEE Std. 1471]. Here the architecture of the Project Infrared application is described from 4 different perspectives [1995 Krutchen]:

1. Logical View – major components, their attributes and operations. This view also includes relationships between components and their interactions. When doing OO design, class diagrams and sequence diagrams are often used to express the logical view.
2. Process View – the threads of control and processes used to execute the operations identified in the logical view.
3. Development View – how system modules map to development organization.
4. Use Case View – the use case view is used to both motivate and validate design activity. At the start of design the requirements define the functional objectives for the design. Use cases are also used to validate suggested designs. It should be possible to walk through a use case scenario and follow the interaction between high-level components. The components should have all the necessary behavior to conceptually execute a use case.

# Design Goals

*There is no absolute measure for distinguishing between good and bad design. The value of a design depends on stakeholder priorities. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vise versa. Therefore, before presenting a design it is good practice to state the design priorities. The design that is offered will be judged according to how well it satisfies the stated priorities.*

The design priorities for the Project Infrared application are:

* The design should minimize complexity and development effort.
* The design should use one DOWN pushbutton for scrolling of the menu (downward, loop back when end is reached) and one SELECT pushbutton to select (a 1-character size arrow icon next to menu title should be the indicator for current menu selection.
* The design should have a LCD display (2x8 characters).
* The design should use red and green LEDs as event indicator.
* The design should use ESP8266 Wi-Fi chip for communication with the cloud.

# System Behavior

*The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review of the expect system behavior in order to set the stage for the architecture description that follows. For a more detailed account of software requirements, see the requirements document.*

SELECT pushbutton:

* Press and release SELECT pushbutton once while selecting **record** option:
  + Blink red LEDs.
  + Start recording on the infrared receiver.
  + After a period of time without detection, complete recording and show solid green LEDs
  + Encode the signal, send to IoT Server
* Press and release SELECT pushbutton once while selecting **play** option:
  + Blink green LEDs
  + Start transmitting infrared signals via infrared LED
  + Record the time of play, send packet to IoT Server
* Press and release SELECT pushbutton once while selecting **clear** option:
  + Blink red LEDs rapidly (2 times/second)
  + Stop red LEDs and show solid green LEDs when finished.
  + Record the time of the clear, send to IoT Server

DOWN pushbutton:

* Press and release DOWN pushbutton once:
  + Select next option. If current option is the last option, select first option.

LCD Display:

* Show two menu options (one for each row) if current selection is not last option. When selecting last option, the last option should be displayed at the first row while leaving second row empty.
* Current selection will always be in the first row.

# Logical View

*The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.*

*In this section the modules of the system are first expressed in terms of high level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.*

The project will use Atmega328p from Arduino as the processing unit. In the main looping function, it will wait for any of the push button to be depressed and then perform appropriate action based on the menu selection.

## High-Level Design (Architecture)

The high-level view or architecture consists of the following major components:

* The **Infrared Receiver** will be receiving signal during signal capture state. Microcontroller should record the signal received and stored.
* The **2 Push Button** enables user to select menu options
* The **ESP8266** allows microcontroller to send data to the cloud
* The **LCD** displays menu options and status messages.

## Mid-Level Design

<Explain and/or show static and dynamic aspects of subsystem components. Probably the most effective way of showing mid-level design is with class and sequence diagrams.>

## Detailed Class Design

<For a few key classes you might want to show associations, attributes and methods.>

# Process View

<Where are the threads of control in the application?>

# Physical View

<Where will major components be physically deployed?>

# Use Case View

<Sketch architecturally significant use cases.>