

Group May13-17

Design Document

Senior Design

Group May 13-17

Group Members:

Eric Cheatham, Michael Flagg, Intae Kim, James Sampica, David Turner

Final Version

Table of Contents:

Group May13-17

Introduction

[Executive Summary](#)

[Statement of Purpose](#)

[Definitions](#)

[Operation Environment](#)

[Intended Users](#)

[Intended Uses](#)

Requirements

[Business Requirements](#)

[Technical Requirements](#)

Objectives and Approaches

[Objectives](#)

[Design Approach](#)

[Development Approach](#)

[Validation Approach](#)

Non-Technical Design

[Subsystem Overview](#)

[User Interface module](#)

[Database module](#)

[Server module](#)

[Architectural Diagram](#)

Module Decomposition

[Image Capture Module](#)

[Image Processing Module](#)

[Networking and Communication Module](#)

[Server and Database Module](#)

Standards that Apply

[Portable Operating System Interface](#)

[Java SE Application Design with MVC](#)

Introduction

Executive Summary

Monitoring parking lots at universities like Iowa State is a time and resource intensive task. Parking Division Vehicles are sent out to monitor parking lots and catch parking violations. This costs man hours through staffing as well as vehicle upkeep and maintenance. Many violations do not get ticketed because there is no enforcement present to monitor the lot.

Statement of Purpose

The purpose of this project is to develop a system consisting of a series of centrally monitored image capturing platforms capable of identifying vehicles as they utilize campus parking lots. Our client has three main purposes for this project:

1. Introduce timing and ticketing uniformity into currently existing ticketing system
2. Reduce Department of Public Safety's (DPS's) operating cost by reducing payroll overhead
3. Eventual elimination of parking decals for registered vehicles

Definitions

JDBC – Java framework for database connectivity

MySQL – Database management system to be used for persistent storage

Berkeley Socket - C based API used to establish both TCP and UDP server/client communication

Java Socket – Java class used to establish connectivity between machines

OpenCV – Open source library of functions for real time image processing

Tesseract-OCR – Open source Optical Character Recognition engine

Back-illuminated CMOS – An image sensor design that increases low-light performance

GPU – Graphics Processing Unit. Accelerates image processing functions

Copyleft - Practice of using copyright law to ensure that any licensing restrictions used on a third party library or software applies to any project that uses or extends it.

Lesser General Public Licence Version 3 (LGPL v3) - Copyleft license that ensures that any derivative work is also distributed under the same licence. This licence allows for commercial marketing of a software package that uses libraries that are considered free and open source (FOSS).

System on Chip (SoC)- Integrated circuit that combines the core components of a computer onto one chip.

PIR sensor (PIR)- A sensor that detects the infrared radiation from objects passing in front of it.

Operation Environment

The software portion of our project will developed in C, Java, and OpenCV. The C and OpenCV code will be developed specifically to run on ARM11 based SoCs. All Java code will be used on a centralized server to provide both a data warehousing solution and a web-based client monitoring system.

Intended Users

There are two intended users for our system: Iowa State DPS and anyone parking any vehicle in an Iowa State University controlled parking lot. The vast majority DPS's interaction with the system will be through a Java based web-portal to monitor lot use across campus. Use of the system by vehicle

operators will be limited to having their license plates imaged by our remote devices located at the entrances to the parking lots.

Intended Uses

The vehicle sensor will be placed at the entrances and exits of parking lots on the Iowa State University campus. They will be used to photograph the license plates of passing vehicles and pass license data to a database server. The end user will then be able to access data via the web portal.

Requirements

Business Requirements

- The system will have a total cost of less than \$20,000 USD
- Ability to retrieve data collected through the front end interface
- Ability to determine who gets a ticket and display that information

Technical Requirements

- Solution will be able to make use of existing campus-wide wireless networks
- Solution will have some system for persistent data warehousing

Objectives and Approaches

Objectives

- Capture an image of a license plate using motion detection
- Locate license plate and convert characters to text
- Send information over network to a server
- Check license plate against database
- Determine if car should be ticketed
- Store images and time data
- Display information in user interface

Design Approach

Determine the necessary functions of the device and separate them into modules.

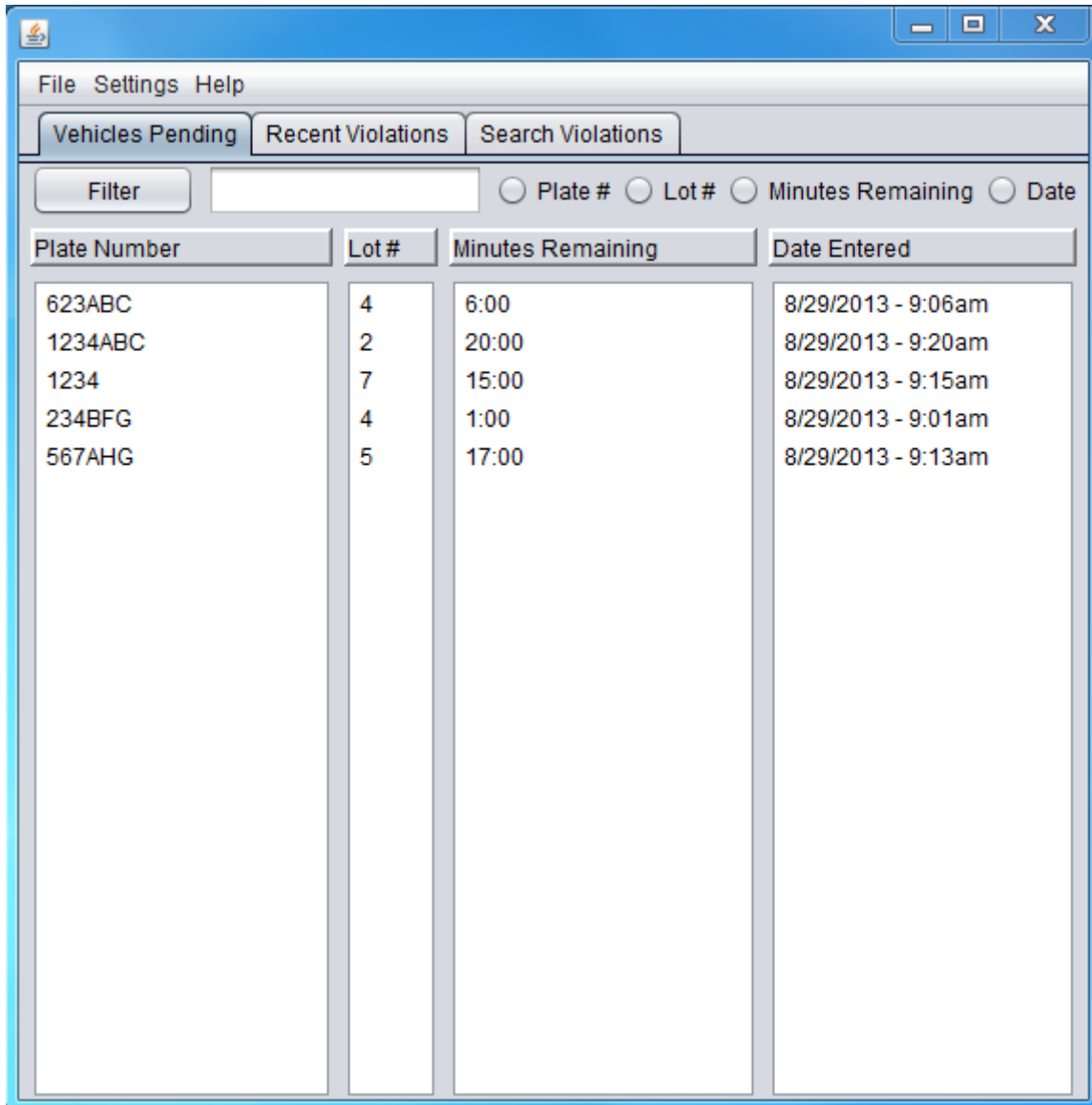
Development Approach

The project modules will be implemented independently but with focus on integration with other relevant modules.

Validation Approach

All major design decisions are to be verified with the client before proceeding. All team members will test their own modules and final testing will be done collaboratively. This final testing will be thorough and attempt to cover edge cases where reasonable and possible.

Non-Technical Design



The screenshot shows a software application window with a blue title bar and standard Windows window controls (minimize, maximize, close). The menu bar includes 'File', 'Settings', and 'Help'. Below the menu bar are three tabs: 'Vehicles Pending' (selected), 'Recent Violations', and 'Search Violations'. A 'Filter' button is followed by an empty text input field. To the right of the input field are four radio buttons: 'Plate #' (selected), 'Lot #', 'Minutes Remaining', and 'Date'. Below this is a table with four columns: 'Plate Number', 'Lot #', 'Minutes Remaining', and 'Date Entered'. The table contains five rows of data.

Plate Number	Lot #	Minutes Remaining	Date Entered
623ABC	4	6:00	8/29/2013 - 9:06am
1234ABC	2	20:00	8/29/2013 - 9:20am
1234	7	15:00	8/29/2013 - 9:15am
234BFG	4	1:00	8/29/2013 - 9:01am
567AHG	5	17:00	8/29/2013 - 9:13am

This is the main view tab. The Pending tab would show the user what license plates are pending for a ticket after their grace period has ended. It would contain a list of plates with corresponding lot and time amounts that count down. If the vehicle leaves before the timer expires the plate number will disappear from this list. If the timer expires the plate number will be moved to “Recent Violations”.

“Recent Violations” will contain a list of the most recent violations and display information about that violation such as the plate number, lot, and date. The user can also filter by date and by lot to drill down details. The list size is static in size and when it fills up the least recent violation is dropped off the list.

The “Search Violations” tab allows the user to query the database to get a certain set of violations based upon what they have searched. The list is initially empty and populates when they have entered information and selected a radio button. It is designed to be able to view the data as long term history.

Subsystem Overview

Backend Modules: Front End client (user interface), Server, Database

User Interface module

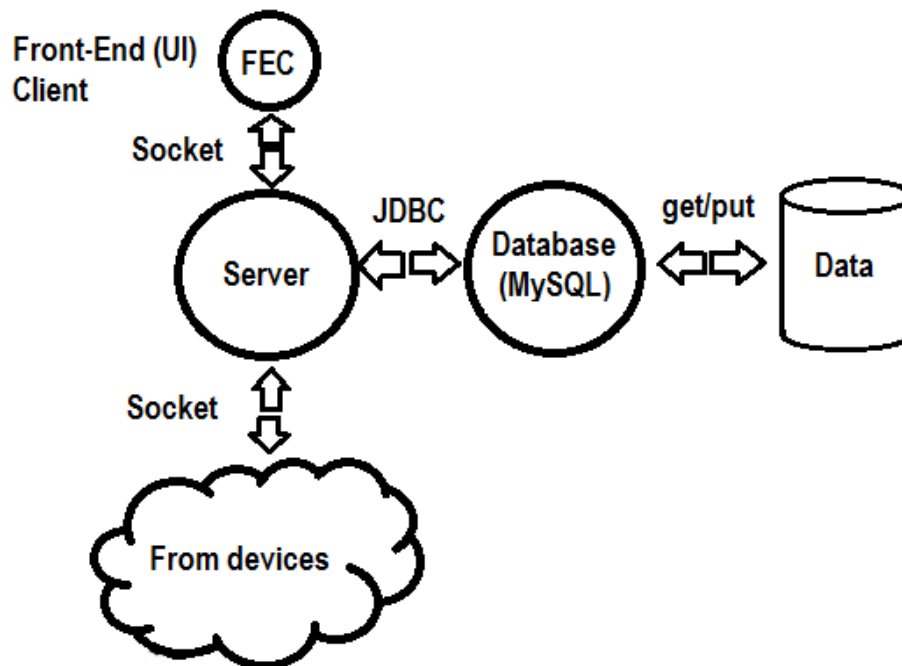
The user interface serves as an information and data outlet for the data collected by the sensors. It makes requests to the server module for information.

Database module

The database module stores and retrieves data stored by the sensors. It will send and retrieve its data to the server.

Server module

The server module acts as a delegate between the user interface module and the database, retrieving data from the database module and sending it to the interface to be displayed. It also acts as a delegate between the database and the sensors, collecting data from the sensors and giving it to the database module. Architectural Diagram



Module Decomposition

This section of the document will provide an overview for each major module of the project. Each module will be discussed in further details in their respective sections of the document.

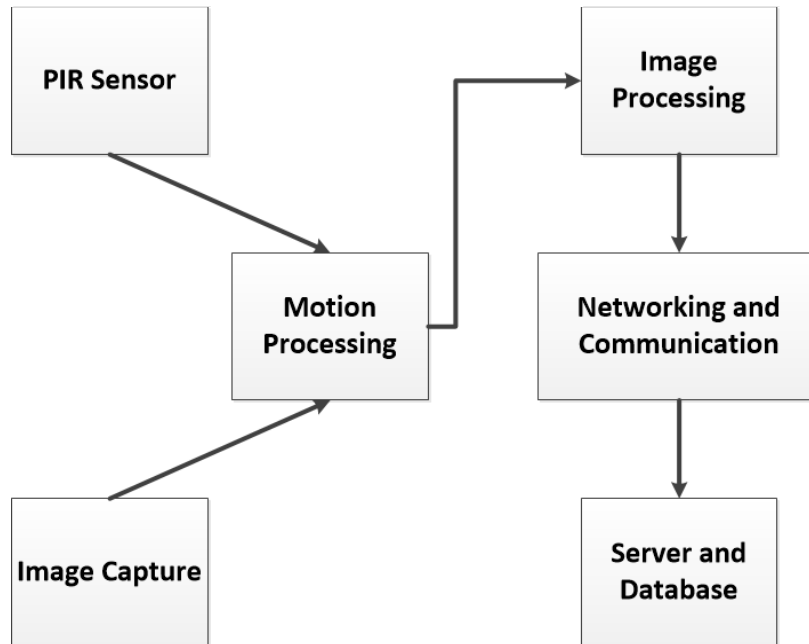


Image Capture Module

This module consists of a 5MP back-illuminated CMOS camera sensor and an API implemented in C. The Raspberry Pi foundation will supply the API. GPU accelerated scaling is used to resize the captured images and video to a size adequate for processing. Image data is forwarded to the motion processing module.

PIR Sensor Module

The PIR sensor is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. A PIR-based motion detector is used to sense movement of people, animals, or other objects. They are commonly used in burglar alarms and automatically-activated lighting systems.

The PIR sensor module is used to detect motion within a short distance in front of the device. If motion is detected, a signal is sent to the motion processing module. This module works in parallel with the Motion Processing Module to attempt to differentiate between automobile traffic and all other forms of motion. In addition, it provides redundancy for situations where the camera fails to detect motion.

Motion Processing Module

The motion processing module is broken into two parts. Motion on the camera is detected by an open-source program named *Motion*. It monitors a low resolution video stream from the image capture

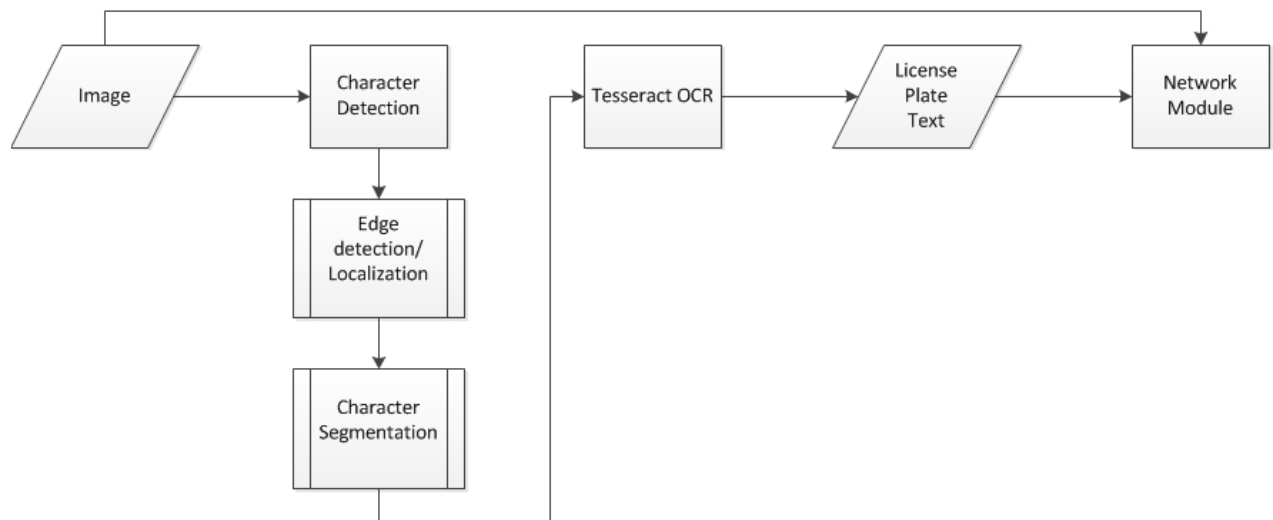
module. It signals custom C++ code to handle motion events near the center of the device's field of view.

The custom code takes inputs from *Motion* and the PIR sensor module. It uses both inputs to determine if a high resolution image should be captured of the passing vehicle. If so, it uses the image capture module's API to capture a high resolution image of the motion event. This image is then forwarded to the image processing module.

Image Processing Module

The image processing module is implemented in C++ and consists of two distinct parts, character detection and character recognition. First, the module takes a license plate image from the image capture module and uses functions from the OpenCV library to detect edges in the image. Contour detection is applied to localize possible license plates. An algorithm is then run on the possible plates to detect and segment characters.

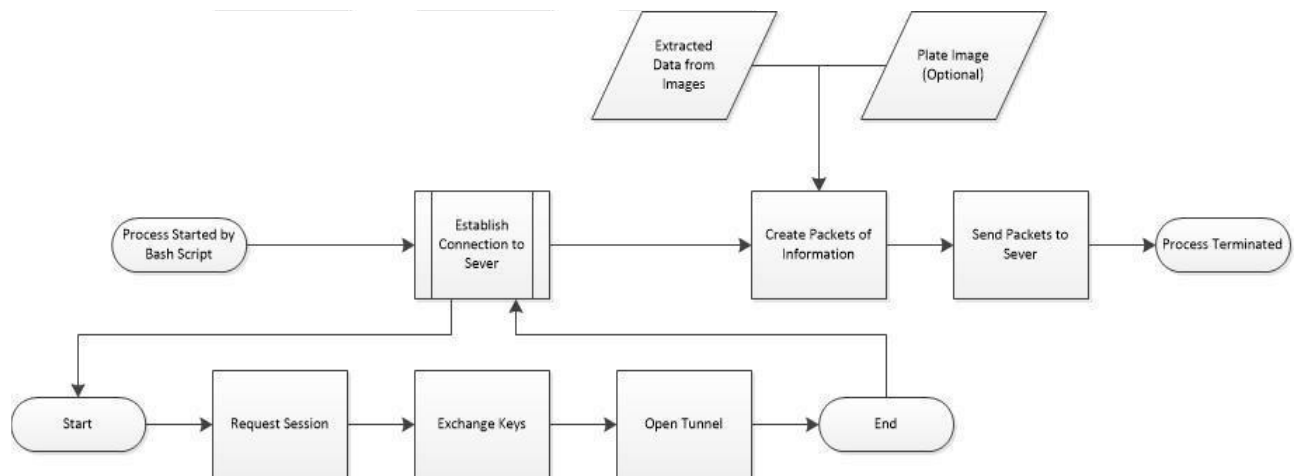
After the characters have been detected they are sent to the character recognition function, which uses the Tesseract OCR engine to translate these characters from images to plain text. Finally, the license plate text is sent along with the original image to the networking module, for transfer to the server.



Networking and Communication Module

This module is implemented in C using BSD Sockets to establish a connection between the remote devices and the central server. The module uses the GnuTLS library to secure communication between the devices and the server using SSL .

As inputs, this model takes an image from the OCR module, and the information extracted from that image, to be split. From the inputs, this module creates a series of packets. The first of which contains the device name and information while the subsequent ones contain the stored image.



Server and Database Module

The database portion of the SDM module stores and retrieves data stored by the sensors. It will send and retrieve its data to the server portion. The server portion of the SDM acts as a delegate between the user interface and the database, retrieving data from the database module and sending it to the interface to be displayed.

Standards that Apply

Portable Operating System Interface

(<http://www.opengroup.org/austin/papers/backgrounder.html>)

The Portable Operating System Interface (POSIX) standard was created to ensure portability of code between operating systems. Code portability is achieved by defining a clear set of services that every POSIX compliant systems are expected to provide. POSIX acts as an umbrella for a large number of IEEE standards that define operability across multiple platforms.

Java SE Application Design with MVC

(<http://www.oracle.com/technetwork/java/codeconv-138413.html>)

(<http://www.oracle.com/technetwork/articles/javase/mvc-136693.html>)

The Java coding standard will mean it would be easy to read and for somebody else to pick it up and modify it. The MVC standard means that we can easily change views or models depending upon what need arises in the future. MVC also makes the code much more modular so it's more maintainable.

Software License Compatibility Standards

(<http://www.gnu.org/licenses/lgpl-3.0.html>)

(<http://www.gnu.org/licenses/license-list.html>)

Most mainstream software packages are protected by very specific software licenses. These licenses are designed to protect the intellectual property of the developers of the product. To ensure that any future development in this and any project remains the property of the developing entity, a series of general use licenses were created. Due to the targeted nature of many of the existing licenses, there exist many incompatible licenses. To be able to market a product and still comply with software licensing laws, the Lesser General Public License was created. This license specifies proper use of software libraries secured by any potentially conflicting licenses.