reLive System

Critical Design Review

reImagine Technologies

David Moreno

Mario Raushel

Terrence Sin

Caleb Wells

Department of Computer Science

Texas A&M University

March 3, 2008

Table of Contents

[1 Introduction 1](#_Toc192232655)

[1.1 Problem Background 1](#_Toc192232656)

[1.2 Needs Statement 1](#_Toc192232657)

[1.3 Goals and Objectives 1](#_Toc192232658)

[1.4 Design Constraints 2](#_Toc192232659)

[1.5 Validation and Testing Procedures 2](#_Toc192232660)

[2 Proposed Design 3](#_Toc192232661)

[2.1 Updates to the proposal 3](#_Toc192232662)

[2.2 System Description 4](#_Toc192232663)

[2.3 Complete module-wise specifications 6](#_Toc192232664)

[3 Project Management 15](#_Toc192232665)

[3.1 Updated implementation schedule 16](#_Toc192232666)

[3.2 Updated validation and testing procedures 16](#_Toc192232667)

[3.2.1 Hardware 16](#_Toc192232668)

[3.2.2 Software 17](#_Toc192232669)

[3.3 Updated division of labor and responsibilities 18](#_Toc192232670)

[4 Preliminary results 19](#_Toc192232671)

# Introduction

## Problem Background

Diaries and logbooks have been around since the beginning of recorded history. Today diaries and logs have extended onto the internet through blogs and social networking websites. In order to create an accurate representation of the day’s events, one has to spend time writing in fine detail, which requires time they may not have to spend. Our system solves this problem by providing a complete photo log of daily events tagged to GPS coordinates, which is then viewable online through Google maps. This is accomplished by having the user wear a combination camera and GPS unit designed to take photographs at designated triggers: time, distance, or face detection. Images can then be transferred from the camera to a user’s computer and eventually the internet before being incorporated into their daily log.

## Needs Statement

There is a need for users to autonomously create and explore daily photo life logs for both a fun and interesting experience.

## Goals and Objectives

The goals for this project is to create a wearable camera system capable of recording GPS data that will automatically record the daily life of the user based on a variety of triggers including time, distance, and face detection.

Here is a list of objectives we have taken into consideration when designing our device:

* The prototype design should cost no more than $500.
* The prototype must be battery powered and capable of logging at least 4 hours without needing to be recharged.
* The camera unit should not cause any harm to the user
* The camera should function well both inside and outdoors
* The GPS should function well in outdoor urban areas and have a decent failover for loss of signal indoors
* The design must be comfortable to wear and lightweight
* The accompanying software must be easy to use and understand

## Design Constraints

The economic constraints of our design are twofold. The initial prototype development has a total budget of $500. To be a viable commercial product, the final design must be both competitively priced and affordable to the consumer while maintaining a profit. In order to meet those specifications, the final consumer product should cost around $400.

As a wearable device, there are several physical constraints that must be met. First and foremost, the device must be lightweight and comfortable to wear for an entire day. As a wearable device, the system must run on batteries and be capable of powering the device for an entire day without being recharged. The placement and charging of the system’s batteries must also be easy to access and replaced as they will have to be replaced or recharged regularly. In addition, access to the removable SD card must also be easy to access as users may want to synchronize pictures on a daily basis.

The design of the accompanying software must also be user friendly and easy to access all of the configuration settings of the camera. Anyone with basic computer experience should have no problem understanding how to use the program without much instruction. For the user to gain full use of the software however, the user must have an active Google account and password. This will enable the user to host their life log online where they can easily share images and accompanying maps to their friends and family.

The system must be usable both inside and outdoors, so the consumer product must be resilient to a variety of weather conditions.

## Validation and Testing Procedures

There are two main components that need to be tested, the hardware and the software. To validate the hardware works we would have to test the GPS unit’s accuracy, the CMUcam3’s camera’s picture quality and responsiveness, and the microcontroller drivers’ logic correctness. To validate the software we must test first of all the logic correctness by running the software with various sample cases. We will test whether the program is reading and writing to the SD card correctly. We must make sure that all functions that interact with the user’s Google account are correct according to the Google API. To completely validate the software we need to test its ability to handle all the files without destroying any of them.

# Proposed Design

## Updates to the proposal

Our design has been completely changed from our initial proposal because after examining our initial proposal, the proposal was less a design than it was a research project. Our new design is now one of a wearable GPS camera (Cmucam3) that logs images throughout the day. The images will be mapped through a software application by their date, time and location. In effect, GPS data will have been incorporated with the video life log idea allowing users to see their pictures both chronologically and geographically.

Since the conception of the idea, there have been numerous changes to our initial design. Most of these changes have occurred in the implementation of the software application as it was difficult to do testing on the hardware design without the camera itself. For our application, we were initially looking into running a Perl script that would create the website’s html. This website would include all the Google maps features and functions we would want implemented.

However, after looking a little more into what resources we had, we found an API for Picasa web albums. Through this, we have implemented many of the features we had first desired through a C# application that interacts directly with a user’s Picasa Web Albums account. The application uploads pictures and retrieves the user’s album information to and from their account. One difficulty presented by using this API is in changing how the map is presented. More work will be done on trying to make this possible and allow the user different ways of viewing their albums and locations.

Some other changes include added configuration settings, features, and different ways of sorting albums and directories. On the hardware side of things, one important change has been made. Initially from the specifications, it seemed as if the GPS chip could be powered directly by the Cmucam3’s board. However, once we received the Cmucam3, we found the voltage between the pins we needed to connect to the board to be too high (> 3.3 V). Therefore, we will now be using a voltage regulator.

Because our design is completely different, a new budget will be presented below. The design consists of a programmable camera called the Cmucam3, a GPS chip, an antenna for the chip, and a SD card for storing the pictures taken by the camera.

New Budget

Single Unit

Cmucam3 $ 239.00

Trimble Copernicus GPS Chip $ 74.95

Antenna $ 18.95

SD Memory Card ~ $ 15.00

Card Reader ~ $ 3.00

Enclosure ~ $ 10.00

3.3 Volt Regulator ~ $ 1.00

Battery Pack ~ $ 5.00

Batteries ~ $ 10.00

Total for 1 unit ~ $ 376.90

Accounting for extra costs that may come up, the total cost of our product will be around 400 dollars. This is within our goal of keeping the budget under 500 dollars. A large part of this cost comes from the purchase of the Cmucam3 itself and if bought in bulk, this price would most likely be much lower. For this reason, our product could probably be sold at the price of 400-500 dollars for a reasonable profit.

## System Description

The reLive system will be comprised of two main components that interact in different ways. The user will use a PC to run the application that will program the media card containing the settings the user desires. The user will then wear the camera and will not need to interact with the system. Once the user is finished wearing the system, they will take the media card and insert it into the computer where it will launch the application and allow the user to see a map of their path along with the pictures taken while wearing the device.

The first step will require the user to define the settings to describe how they would like the reLive system to behave. These options will be programmed on the computer with a media card inserted. This media card will be written with a configuration file that will describe the user’s commands. The user will be able to program the triggers for time, distance, or face detection. The user will also be able to specify a beacon and a distance from that beacon to activate the triggers.



Next, user will transfer the media card from the PC to the CMUcam3 and power on the system. The reLive system will then proceed with the commands given to it by the user during the initial step. This step will perform the image processing functions. The user will not need to have any interaction with the system during this phase. The system will capture images after any of the triggers occur.

Once the user is finished logging, the media card will be transferred back to the computer. Once the media card is inserted into the PC, the reLive software will launch, capturing the data from the camera. The media card will be erased (other than the configuration file) and the data will be stored on the PC waiting to be uploaded to the web album.

When the user is ready, they can upload the images to the web album where the pictures will be permanently stored. The user will be able to use these albums to see the maps from the day. Each day the system is run will have its own album. The web album will be able to dynamically create the map from whichever album is chosen. The user will not need to do anything other than ask for the map to be displayed.

The user can also use the software to reprogram the media card and change the settings they would like to use to capture images. This can be done before or after the user uploads the images to the web album.

## Complete module-wise specifications

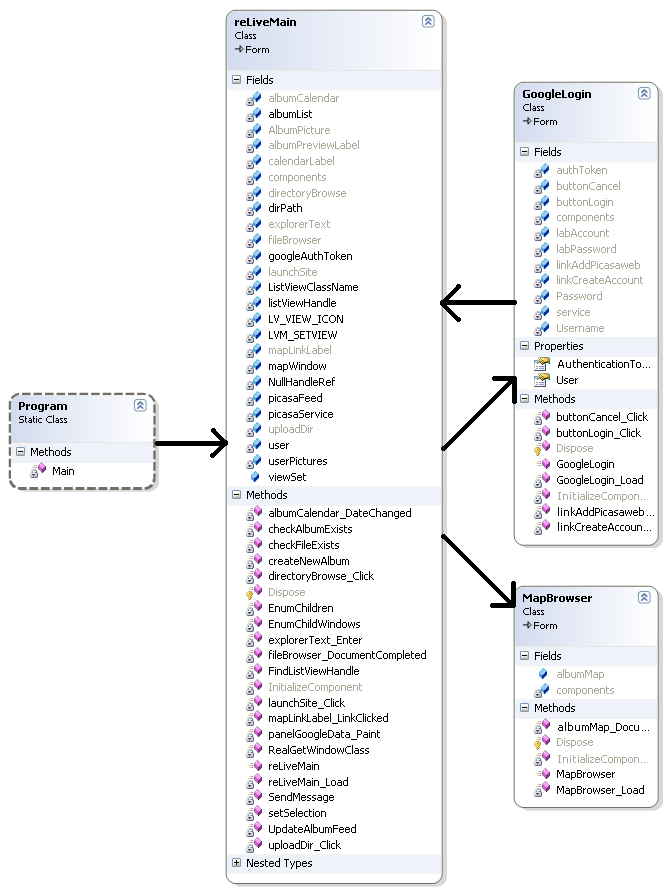
The software will be composed of a graphical user interface written in C# that will allow the user to control many aspects of their camera’s configuration. Camera configuration with be written to a configuration file stored on a SD card and edited as needed by changes in user preferences. Users will also be able to setup and format new compatible SD cards as needed.

Users will have the following camera options to configure through the software interface:

* Time delay between pictures
* Minimum distance between pictures
* Toggle trigger for facial detection
* Location (GPS) based halo to enable camera use

Once pictures have been synchronized to the user’s computer, they will have the option to edit, tag and remove pictures from their photo log. Tags, descriptions and comments will then be written to the pictures EXIF metadata. After making any desired changes to metadata, users will then have the option of uploading their albums to their Google account. They will then have access to ‘reLive’ their daily experiences from any computer or device with internet access. The resulting daily albums and maps will then be available at http://www.picasaweb.google.com. Users will have an option to launch their generated maps directly from the program.



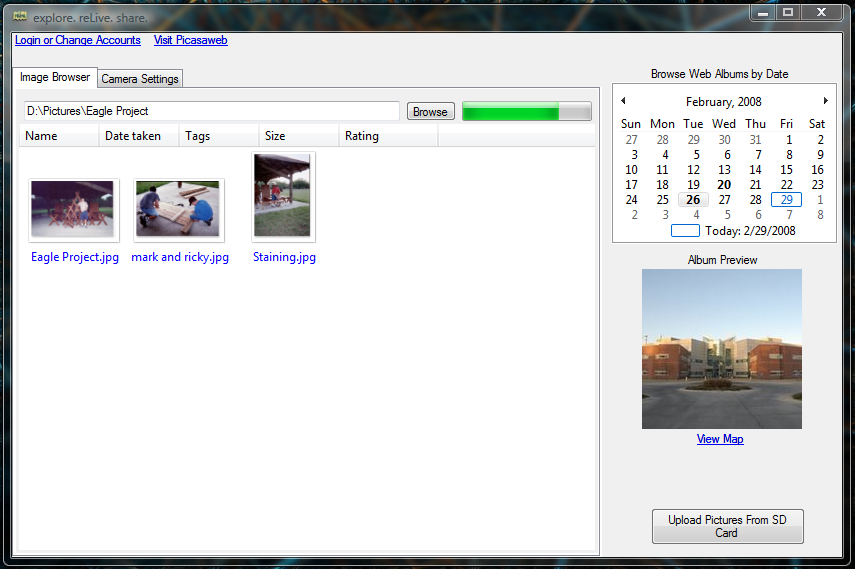


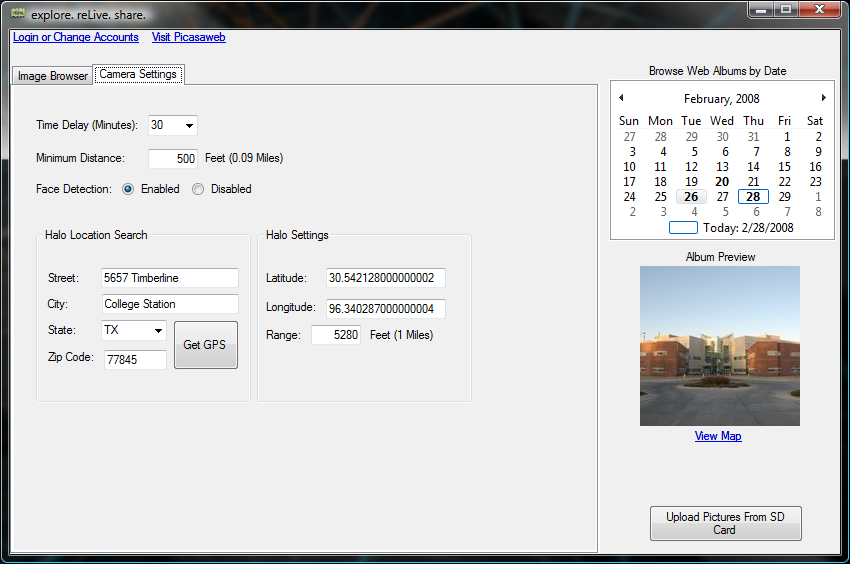
* Main Class – Initialization and start of graphical interface
* reLiveMain - Main graphical user interface for album management
* GoogleLogin - Graphical user interface for Google account login and registration
* MapBrowser - Graphical user interface for viewing GPS based maps of uploaded albums
* Google SDK – Contains necessary API for accessing and updating data with Google services



Concept Design of Software

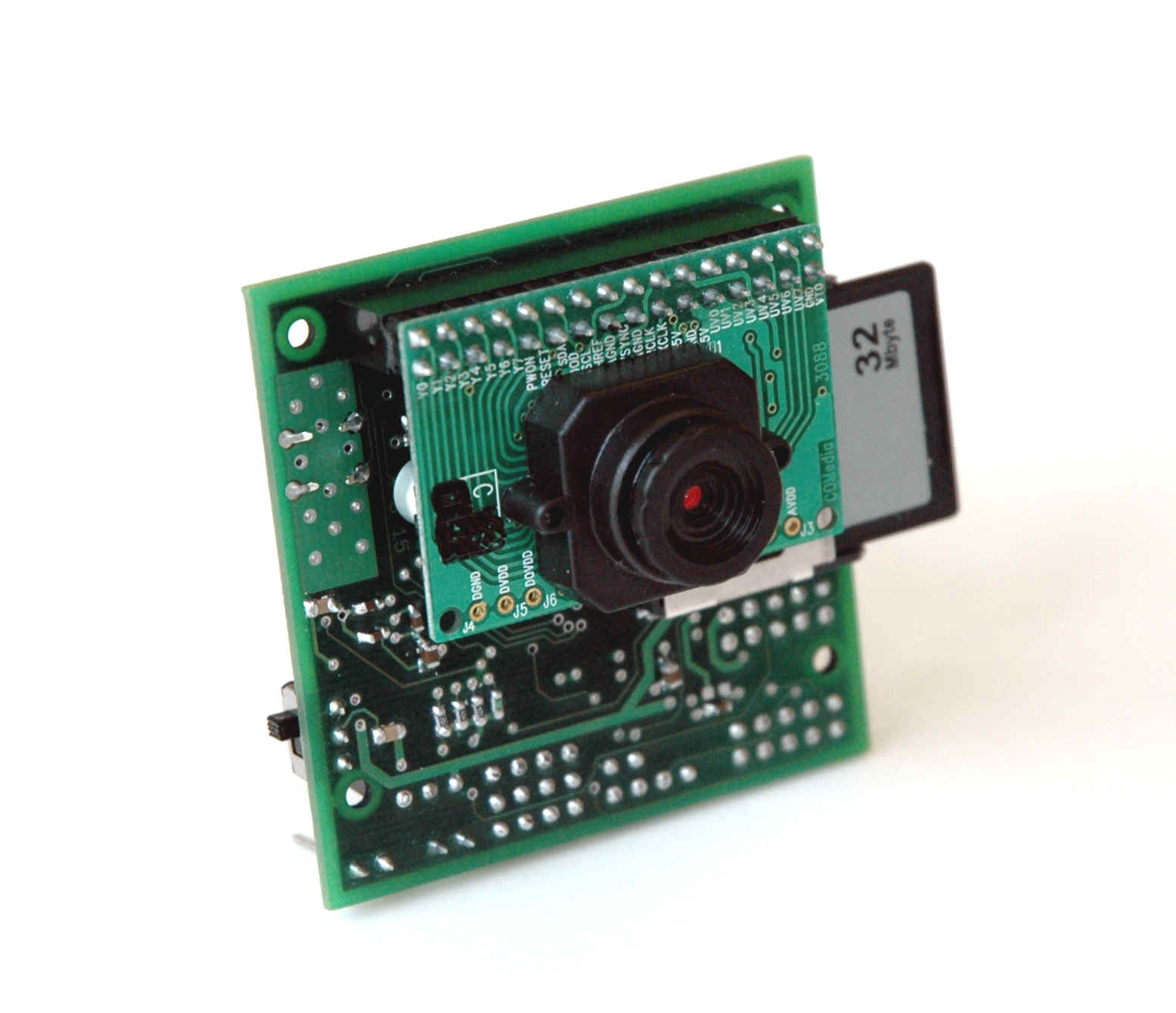
Early Photoshop concept designs have guided the development of the reLive software and can already be seen in early functional design. The current build follows the concept design, but includes added functionality for accessing the users online image storage.





Current Software Design

The reLive system will be run from a bundle of hardware called a CMUcam3. The CMUcam3 has an ARM7TDMI image processor to process the images taken by the Omnivision CMOS sensor. The CMOS sensor has the ability to interface with the OV6620 and the OV7630. The CMUcam3 has a pair of serial ports, one of which is level shifted and one that is not. Along with the serial ports, there are a number of LEDs to indicate status as well as an analog output that will be able to supply power to a second device. This system will be supplied with between 6 and 15 volts of DC power (at least 150 mA). This power will be supplied from batteries that are inside of the enclosure.



This CMUcam3 will connect to a Copernicus GPS module that will allow the CMUcam3 to find its location. The GPS has a horizontal accuracy of less than 2.5 meters 50% of the time and less than 5 meters 90% of the time. When starting cold, this module will take 39.7 seconds to acquire the signal, 35.4 seconds when warm, and 3.1 seconds during a hot start. This module will need to be powered by a 3.3 volt source. The antenna is passive and will be powered by the GPS module. This chip will give the wearer the ability to acquire their location in most outdoor environments.

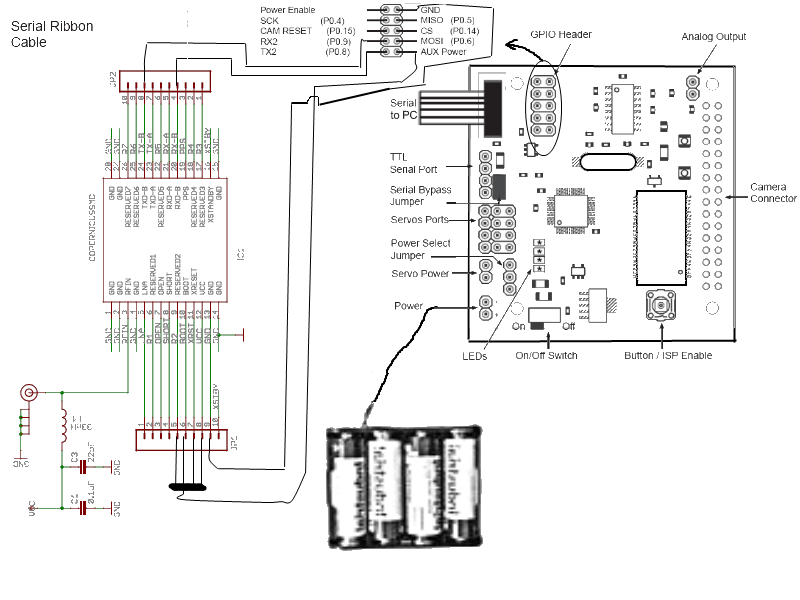


The software solution that will be running on the CMUcam3 will rely on a configuration file that will exist on the media card. For the software to begin execution, the media card must be detected. The CMUcam3 API allows for the detection of a compatible media card if it is formatted to FAT16. Once the media card is found to exist, the configuration file can be read and parsed.

This configuration file will contain information needed to inform the CMUcam3 how to operate. It will contain the user’s preferences that were programmed to the media card using the software solution on the computer. These preferences will include the user’s choice of image triggering, thresholds, and any color or image quality configurations the user chooses to specify.

The CMUcam3’s ARM7TDMI is a Philips LPC2106 microcontroller. The LPC has 2 UARTS for communicating with other devices. The CMUcam3 has made both easily accessible. UART0 has two ways of accessing it, with or without the help of a level shifting chip. There are three ports through which you can access the UARTS. The serial port and the TTL port both connect to UART0, in order to use the serial port the serial bypass jumper must be in place. The serial port is basically used to connect to the PC and it uses the level shifting chip to increase the voltage of the signal from 3.3V to 5V. The TTL port is to connect to TTL devices or other microcontrollers. The bypass jumper must be off to prevent damage to the TTL device. The GPIO Header gives you access to UART1, it does not have any level shifting chip to allow it to communicate to a PC. The GPIO also has many other pins that you could use for general purposes if the memory card is not inserted. We will use UART0 for communicating with the PC for flashing purposes and debugging since we can write code that will send data to the PC as the program runs. UART1 will be used to connect to the Copernicus GPS unit, it does not have any level shifting chip so it operates at 3.3V and because the GPS unit also operate at that voltage we can connect them directly. The GPS will have to be powered through a regulator, however.

The Copernicus GPS has two serial ports which can be configured to output different kind versions of the data at different intervals and baud rates. We only need one so we will make sure that the one we use outputs the data that we need. After opening the serial ports in the CMUcam3 it is easy to access its data. By using the stdio we can easily read from the UART with commands fprintf, and fgetf. The GPS Units can be configured by sending standard NMEA Messages that the GPS can understand. As a default from the factory the GPS Unit will output Trimble’s message interface through serial port A and NMEA through serial port B with the signal characteristics in the following table. For the NMEA message format it will output GGA and VTG messages every second. To change the defaults at startup you can send special messages through Trimble’s message interface, otherwise, what we plan to do, is at startup just send a configuration message. We will configure it to send RMC messages by sending the following string “$PTNLSNM,hhh,xx\*hh” where the h’s are a hex value that will let the GPS unit know what type of message we want, 0100 for just RMC, and the x’s are a decimal value which means the frequency in seconds that we want it to output that message. The h’s after the ‘\*’ are the checksum. The following is a diagram of how the GPS unit will be connected to the CMUcam3.



The string that the GPS will be configured to send will be the GPRMC string. This string is the global positioning recommended minimum sentence. This string will begin by identifying the string type ($GPRMC). This will be identified as being the proper format by using this header. The reLive system will also use the UTC time (HHMMSS.XXX). This data is provided from the GPS satellites and will give an accurate time. The third field identifies the validity of the entire sentence. In the GPRMC sentence, there are two possible values. An ‘A’ in the field indicates a fix is obtained. A ‘V’ indicates invalid data in the string and whether there is data or not, the string is not valid and the position will be assumed to be the last valid GPS position. The latitude and longitude are the next data fields read. They are in the string in degrees and minutes. This value must be converted to only degrees prior to being used. This formula will convert the minutes to degrees and add it to the whole number of degrees that are passed. The date is also passed in the GPRMC string. The date also comes in from the satellite and will be the UTC date. The final value that will be checked is the checksum at the end of the GPRMC string. This checksum will allow the program to verify the data that was transferred.

If the GPS loses the signal, the system will assume that it has stayed at the last known position. If the GPS sends invalid data, it could send an empty string or erroneous data. The string contains a character that can be used to determine if the values are valid or invalid. There is also a checksum in the string that can be used to ensure the transmission was successful. While assuming the last position will halt the picture taking for certain types of triggers, it will enable other triggers to continue capturing images and get a reasonable estimation of position. Once the GPS signal is reacquired, the system will do future processing with the valid data. To let the user know that the GPS data is invalid, the CMUcam3 will have an LED turned on when the GPS data is invalid.

The user will be able to specify which type of triggering they would like to use. When a trigger is enabled, the camera will take a picture, record the latitude, longitude, date, and time of the picture and save this data to the media card. There will be enough information in the GPS file to link a picture to the GPS data so the software on the computer will have the ability to add the GPS data to the JPG file’s EXIF fields.

To determine when the images should be taken, the system will be able to be triggered by four separate events. The system will determine when a user specified set of criteria has been met, take an image and record the current GPS location. The reLive system will be able to handle time triggers, distance triggers, and face detection triggers.

Saving images to the CMUcam3 will require a maximum of 128 files per directory due to the FAT16 formatting. The system will limit the number of files in the directory to 128 for long filenames. Since this is the lower bound limit, the team will not allow more than 128 files per directory. The images will be organized first in a day directory followed by an hour subdirectory. Since we are not allowing more than one picture per minute (even with the distance triggering) organization the files in this way will allow the directory to be limited to a maximum of 61 files.

Parsing the GPS string will be done on the CMUcam3. The parsing will search for the commas in the string and check for the validation character. The latitude and longitude will be read from the string and converted from degrees and minutes to just degrees. This will allow the distance from one coordinate to another to be easily calculated.

After reading the configuration file and determining that the system should be a time triggered system, the system will begin by waiting for the GPS to acquire the initial GPS signal. While this signal has not been acquired, the GPS invalid LED will be lit. When triggering based on time, the system will use the onboard real-time clock and determine when a user specified amount of time has elapsed. The system will read the GPS string, parse it, and determine the validity. After that, the CMUcam3 will create the proper directories and save the image and GPS data into the day/hour directory. Once the data has been saved, the system will begin waiting for trigger to occur again.

The second possible trigger is a distance travelled trigger. The user can set the camera to take a picture after a certain amount of distance has been travelled. The distance travelled will be calculated from the point of the last picture taken to the current GPS position. The system will still not allow more than one picture to be taken per minute. This will ensure that the directories do not get over filled and the user is not overwhelmed by pictures due to a GPS that is returning inaccurate data and a user that set the distance too low.

The final trigger that can be run individually is a face detected trigger. The camera will take a picture every minute and run the Viola-Jones face detection algorithm. If a face is discovered, the image will be saved along with its GPS coordinates. This algorithm will return the coordinates of a detected face. If no face is detected, the algorithm will return NULL.

All pictures will be run through the face detection algorithm to determine whether a person exists. If a face exists, the hardware will set a flag that the image has a face detected. This will allow the software on the computer to sort the pictures with a face or without.

The final trigger will be used in combination with other triggers. If a user is inside of a certain threshold of a GPS coordinate, it will activate any of the other triggers. This will enable a wearer to disable the system when the beacon is located outside of the area the user wishes to monitor. When inside of the beacon threshold, the same rules apply for the other triggers and it will be up to the user to specify the separate trigger’s properties.

# Project Management

David Moreno – In charge of hardware and CMUcam3 operations

Mario Raushel – In charge of software interface design

Terrence Sin – In charge of image processing and software imaging functionality

Caleb Wells – Leader, In charge of GPS integration

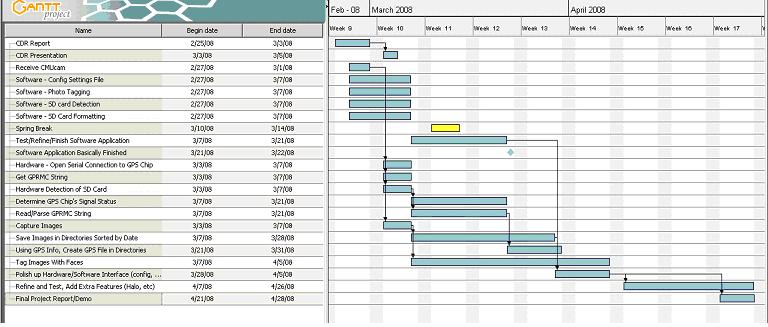
The current situation has been working successfully, the group feels that it has been making solid progress, and has not been held up by not knowing what their responsibilities were. Now that most of the hardware is in, the group will focus on getting more concrete results.

## Updated implementation schedule

The updated schedule targets a completion date of all major features by the first week of April. In order to do this, there is a milestone of finishing major components of the software application by the third week of March, one week after spring break. Hardware components will take a little longer as we have just received the Cmucam3. Once the product is completed, we will consider the implementation of additional features such as the Halo feature and different map views. We will also need to test and polish the project, catching all errors that might occur.

The software part of the schedule includes the tagging of the photos, creating a configuration file for the Cmucam3, detecting a SD card, and formatting the SD card. During the hardware process, much of the software application will also be tested and refined. The hardware part of the process has just started and a lot of work still needs to be done. This includes opening a serial connection, getting the GPRMC string from the GPS chip, parsing it, capturing images, storing them in the SD card by date, and implementing the facial detection algorithm.

The final part of the schedule is the integration of hardware and software. This will be done through the configuration file and the SD card. The configuration file will be generated by the software, stored on the SD card, and detected by the Cmucam3 through the SD card. Both software and hardware sides will automatically detect the SD card and read off of it. The configuration file read by the hardware and the directories of images read by the software.



## Updated validation and testing procedures

To make the relive module a complete success every feature that is promised must work. In order to validate this product we will test its two main components, the hardware and software.

### Hardware

Copernicus GPS Test

Testing the GPS unit will make sure that the location that it sends out is correct. We simply will take the GPS unit to various places, take a picture, and record where we are at. Once we upload the pictures to the PC, we can extract the GPS Data and plug it in to Google Maps if the location that it gives us is what we recorded then that will validate the correctness of the GPS Unit. This all relies on the fact that we were able to connect to the unit with the CMUcam3 and have received valid GPS strings, but more on that later.

CMUcam3’s camera Test

The camera is able to take pictures at two preset resolutions, high and low. Our product should take good quality pictures but it also need to be fast at it. To test this we will write a small program that takes pictures in both resolutions and keeps track of the time it takes to take and store each, this will validate the responsiveness of the system. We will also keep track of the pictures that have faces on them, to validate this feature we will have to take pictures of people’s faces in various environments and see whether or not the CMUcam3 is able to detect the faces. In doing that we would have also validated the quality of the pictures. Good quality pictures would make it easier on the algorithm to detect faces.

Philips LPC2106 microcontroller Test

Here we will mostly test the correctness of the logic of the drivers we write for this microcontroller. We will have to test that first of all the camera can take pictures when we want it to and store them on the SD card. We can validate that easily by pressing the button to take pictures, take out the SD card, load it into a PC, and see if there are image files. Second we want to store the GPS data with the picture, to validate that we have written a test function that will allow us to send a GPS string from the computer through serial. The microcontroller will then parse the string and send to the PC the GPS data through serial, if the string is invalid then it will send invalid. Finally we will test each of the triggers individually by providing the environment in which the trigger will be activated; if it takes the picture the system will be validated.

### Software

SD card management Test

The SD card will hold the key to the interaction between the computer and the CMUcam3. To validate the correctness of the data that the SD card holds we will have to test our programs by saving test data and making sure the desired effect is produced from either end of the system. First we must make sure that the configuration file is correctly written and read into the CMUcam3, this will be done by outputting through serial every step of the microcontroller code. Finally we must test the CMUcam3’s ability to store images into the SD card, we will set up some triggers that will make the CMUcam3 take pictures, if we can see those pictures later on the PC then the SD card is validated.

Google API Test

To make sure that every function that handles the user’s Google account we will test them by inputting most the usual data that may come up through regular use. Our main concern is mishandling the user’s Google information. By following strict guidelines set through the API we should be able to validate our program is secure.

File survival Test

This is will be the easiest to validate. Essentially we need to make sure that our program will not destroy files in the computer, the SD card, or the Google account. To test this we will make sure our program interacts with each and every file that is related to our product in the above mentioned places. If the files are still intact after the end of the program then it is validated.

## Updated division of labor and responsibilities

David Moreno will work with Caleb to finish the hardware design and implementation. The hardware mostly consists of the CMUCam3 and the Copernicus GPS. To start with he will have to get the CMUCam3 and the GPS talking to each other through serial. Concurrently he will also try to get the CMUCam3 to take pictures and store them into the SD card. Upon completion of the above task he will work with Caleb to polish up the overall design.

**David’s Deadlines**

* Get serial communication working – March 7
* Finish SD card management – March 22
* Finish CMUCam3’s programming – April 5
* Refine and Test – April 26

Terence Sin will work with Mario to complete the software application written in C#. This includes implementing the configuration settings, detecting the SD card/formatting it, and tagging the photos. He will also take part in polishing up the application once it has been completed, testing and handling errors that occur. Finally, he will help with interfacing the software and hardware components together once both sides of the project have been completed. Meanwhile, he will also be looking into features involving the face-detection capability of the camera and how those could be added to the project. This task will require research in image processing and understanding how the face-detection feature works on the CMUcam3.

**Terrence’s Deadlines**

* Finish Software App – March 22
* Tag images with Faces – April 5
* Polish Hardware/Software Interface – April 5

Mario Raushel will be in charge of software design and implementation. This includes implementing and designing the initial GUI design, local as well as remote album management, and interaction with Google services through the use of the official C# API. Initially, he was primarily involved with researching various online APIs and services to manage online hosting as well as mapping user data. Upon completion of the client software application, he will work with Terrance to polish up the interface and test the overall design for usability.

**Mario’s Deadlines**

* Finalize camera configuration file – March 7
* Finish Software Application – March 22
* Polish Software Interface – April 5
* Refine and Test – April 26

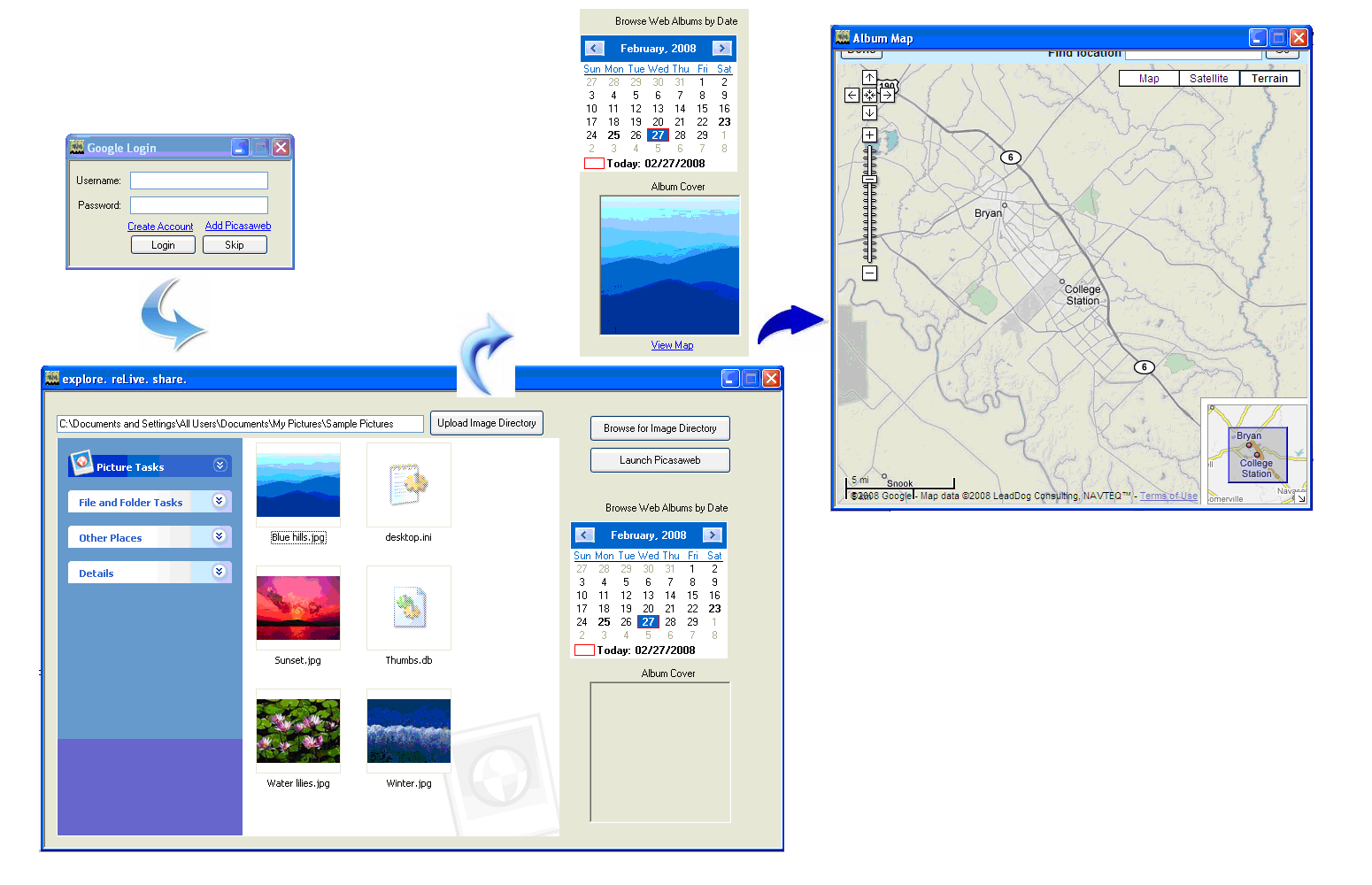
Caleb Wells will be in charge of integrating the GPS and the CMUcam3 to allow it to transmit data as well as parse the data. The GPS will need to be connected to the CMUcam3 with a voltage regulator to keep the voltage at 3.3V instead of the 5V that is output from the CMUcam3. Once the GPS is able to be connected, Caleb will test the parser that was written for the GPRMC string as well as saving the information to a located on the memory chip contained by the CMUcam3. Also, the CMUcam3 will need to be programmed to have the ability to calculate the distances between two GPS points. Caleb will use the solution he created and port it to the CMUcam3 to implement this solution. After that has been completed, the GPS file that was written will need to be integrated into the software and tested with the final version of the software.

**Caleb’s Deadlines**

* Create circuit to get correct voltage – March 7
* Solder connectors and find a way to safely connect the GPS module and CMUcam3 – March 7
* Test Parser and distance finder. Make adjustments – March 20
* Create GPS data file (along with parser for file) – March 29
* Integrate with software solution – April 7
* Refine and Test – April 26

# Preliminary results

Because we just received the Cmucam3, we have been unable to test and refine the code written for the hardware aspect of our design. However, this has also caused us to focus much of our efforts toward the development of the software application.

The application’s process is shown below

The first step window that comes up is the Google Login window. This is where the user logs in to his Picasa Web Albums account. There are also options of creating an account, adding Picasa web, and skipping the login stage. A user skipping the login stage may only desire to change configuration settings on his camera. However, he will not be able to access and browse his albums that have already been uploaded. The other two links merely link to the signup and Picasa web pages.

Once a user logs in, the main page of the application shows up. Currently, only the Picasa album interaction is implemented. The user will select a directory containing the images he wishes to upload. The default for this is the ‘my pictures/reLive’ directory or whatever directory our camera will be uploading into. The user can also browse his past albums by clicking on the bolded dates on the calendar. The album cover of those albums will be shown. Clicking view map will then popup the map of the photos with pictures showing up at their respective positions.