reLive System

*Final Report*

**reImagine Technologies**

David Moreno

Mario Raushel

Terrence Sin

Caleb Wells

Department of Computer Science

Texas A&M University

April 28, 2008

Table of Contents

[2 Table of Figures iii](#_Toc197095730)

[1 Executive Summary 1](#_Toc197095731)

[2 Project Background 1](#_Toc197095732)

[2.1 Need Statement 1](#_Toc197095733)

[2.2 Goal and Objectives 1](#_Toc197095734)

[2.3 Literature and Technical Survey 1](#_Toc197095735)

[2.4 Evaluation of Alternative Solutions 1](#_Toc197095736)

[3 Final Design 1](#_Toc197095737)

[3.1 System Description 1](#_Toc197095738)

[3.2 Complete module-wise specifications 3](#_Toc197095739)

[3.3 Approach for Design Validation 11](#_Toc197095740)

[3.3.1 Copernicus GPS Test 11](#_Toc197095741)

[3.3.2 CMUcam3’s Camera Test 11](#_Toc197095742)

[3.3.3 Philips LPC2106 Microcontroller Test 11](#_Toc197095743)

[3.3.4 System Test 12](#_Toc197095744)

[4 Implementation Notes 12](#_Toc197095745)

[4.1 Hardware 12](#_Toc197095746)

[4.2 Software 14](#_Toc197095747)

[5 Experimental Results 15](#_Toc197095748)

[6 User’s Manual 15](#_Toc197095749)

[6.1 Software 15](#_Toc197095750)

[6.2 Hardware 16](#_Toc197095751)

[6.2.1 Packaging 16](#_Toc197095752)

[6.2.2 Hardware Operation 16](#_Toc197095753)

[6.2.3 Setting Triggers 16](#_Toc197095754)

[6.2.4 Changing GPS Modules 17](#_Toc197095755)

[7 Course Debriefing 17](#_Toc197095756)

[8 Budget 17](#_Toc197095757)

[9 Appendices 18](#_Toc197095758)

# Table of Figures

[Figure 1: System Design 2](#_Toc197095759)

[Figure 2: Concept Software Design 4](#_Toc197095760)

[Figure 3: Final Software Design 5](#_Toc197095761)

[Figure 4: CMUCam3 6](#_Toc197095762)

[Figure 5: SIRFStar III GPS Module 7](#_Toc197095763)

[Figure 6: Serial Communication 8](file:///D:\Documents\Visual%20Studio%202005\483\MISC\Final.docx#_Toc197095764)

[Figure 7: Hardware Diagram 9](#_Toc197095765)

[Figure 8: CMUCam3 program 13](#_Toc197095766)

[Figure 9: CMUCam3 Main method 14](#_Toc197095767)

[Figure 10: Cost of Design 18](#_Toc197095768)

[Figure 11: Cost of Unit 18](#_Toc197095769)

# Executive Summary

(text here) - Caleb

# Project Background

## Need Statement

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

## Goal 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

## Literature and Technical Survey

(text here) - Terrence

## Evaluation of Alternative Solutions

(text here) - Terrence

# Final Design

## 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.

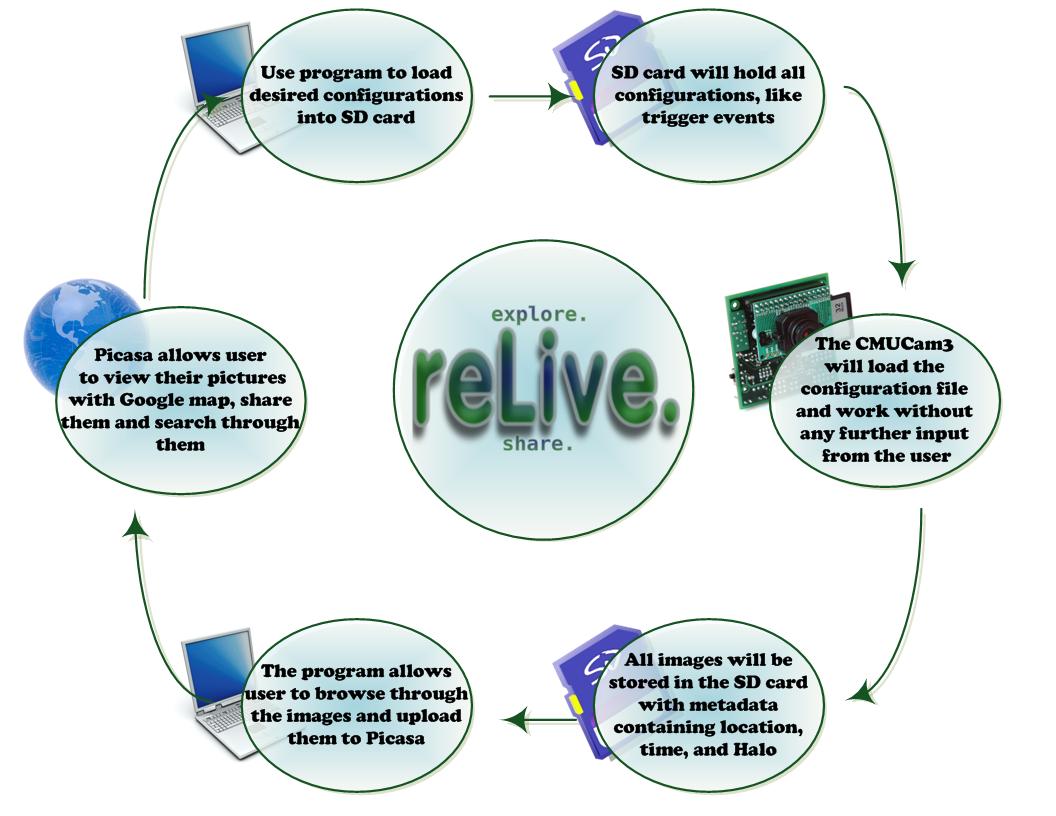


Figure : System Design

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 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.



Figure : Concept Software Design

Early Photoshop concept designs guided early development of the reLive software and can still be seen in our final design. The final build incorporates the conceptual folder and album views, but greatly expands web album management and camera configuration.





Figure : Final 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.

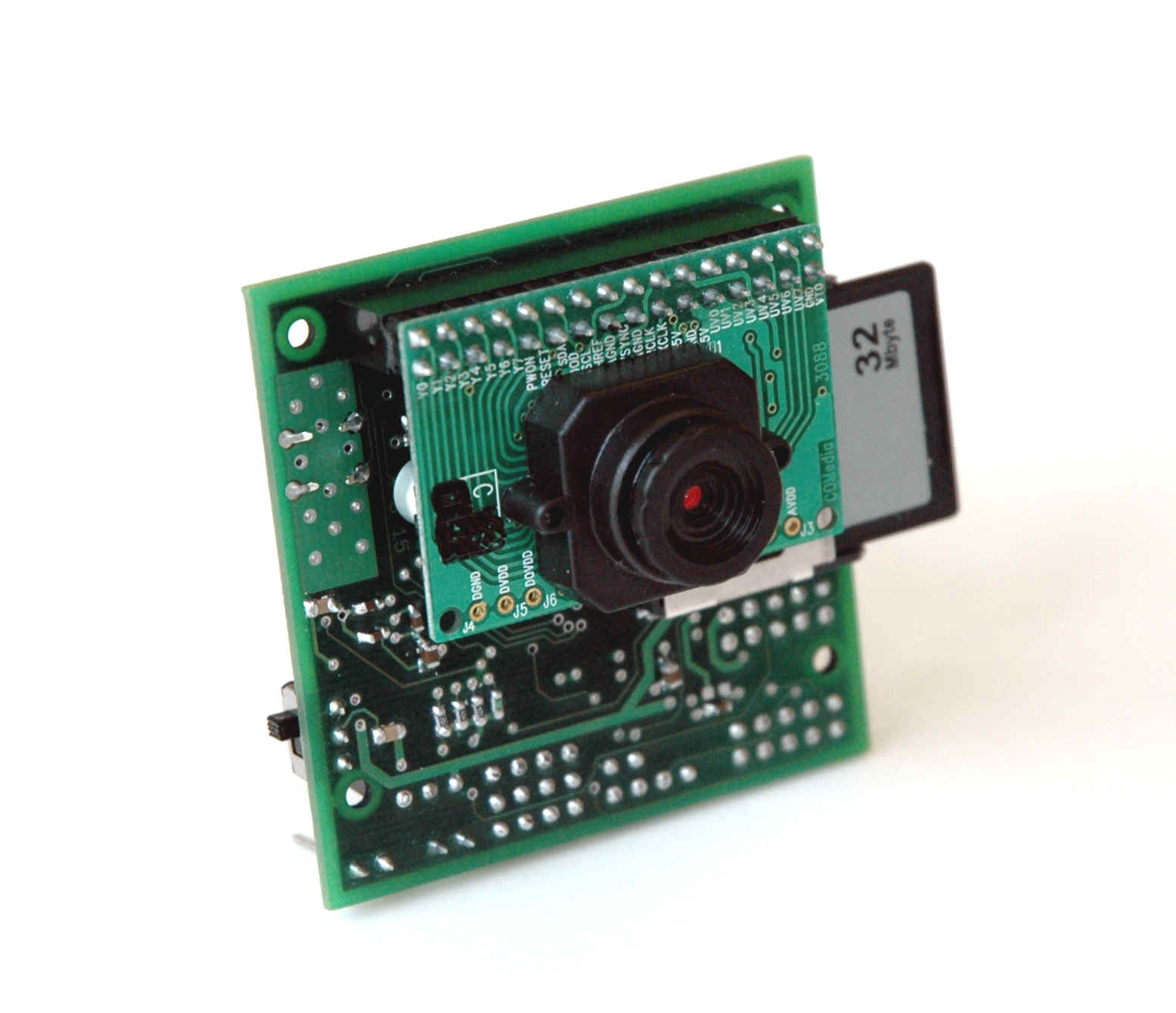


Figure : CMUCam3

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.



Figure : SIRFStar III GPS Module

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.

Figure : Serial Communication

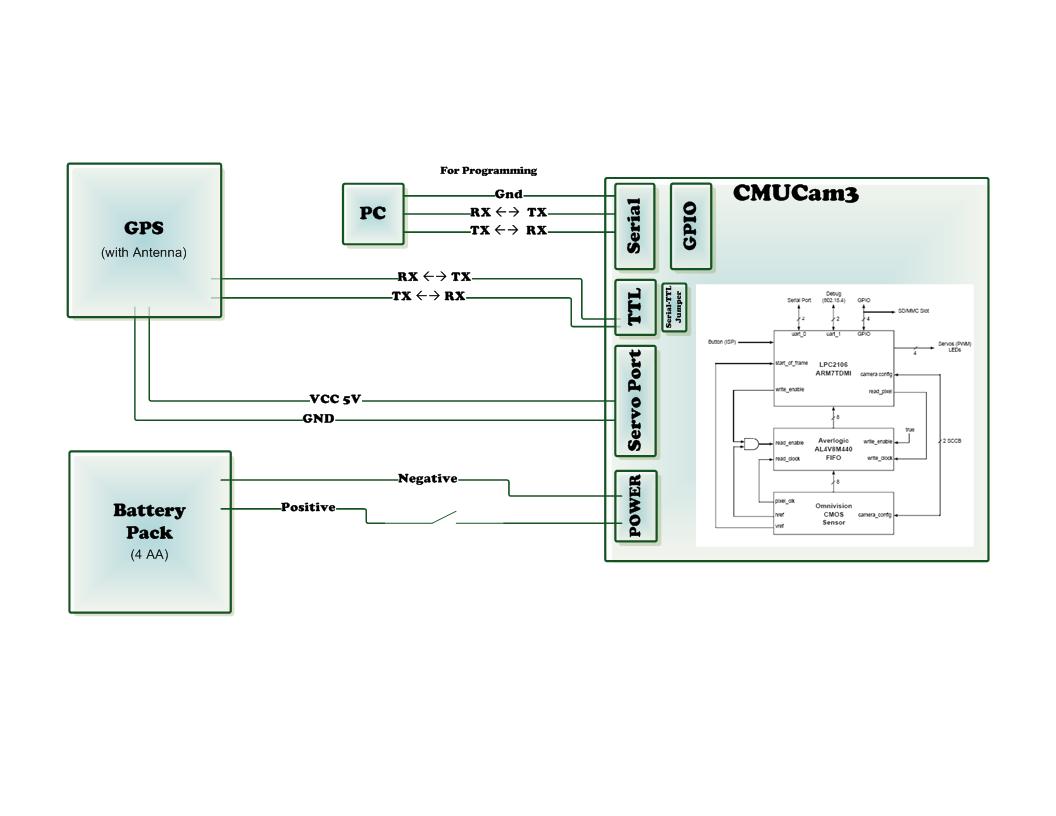


Figure : Hardware Diagram

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.

## Approach for Design Validation

### 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. In doing that we would have also validated the quality of the pictures.

### 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.

### System Test

In order to test the usability of our system, different types of field tests will be held.  To determine the usefulness of the system, the team must attempt to put the system in the situations that we expect in a real world setting. The quality and usefulness of our pictures will be examined when taken indoors, outdoors, while stationary, while moving, while outside and inside halo settings, in adverse conditions, for long and short periods of time, and any other plausible scenarios that the camera may be used in. The team will also test how the system reacts when a GPS signal is lost by unplugging the antenna.

To demonstrate the final product, the team will bring the device into the classroom prepared to demonstrate the time based trigger. Since GPS signal is not likely to be obtained in the classroom, we will have previous runs to demonstrate the different triggers. We will be able to show the software solution by logging into the system, and show the testing runs with the actual pictures taken.

# Implementation Notes

## Hardware

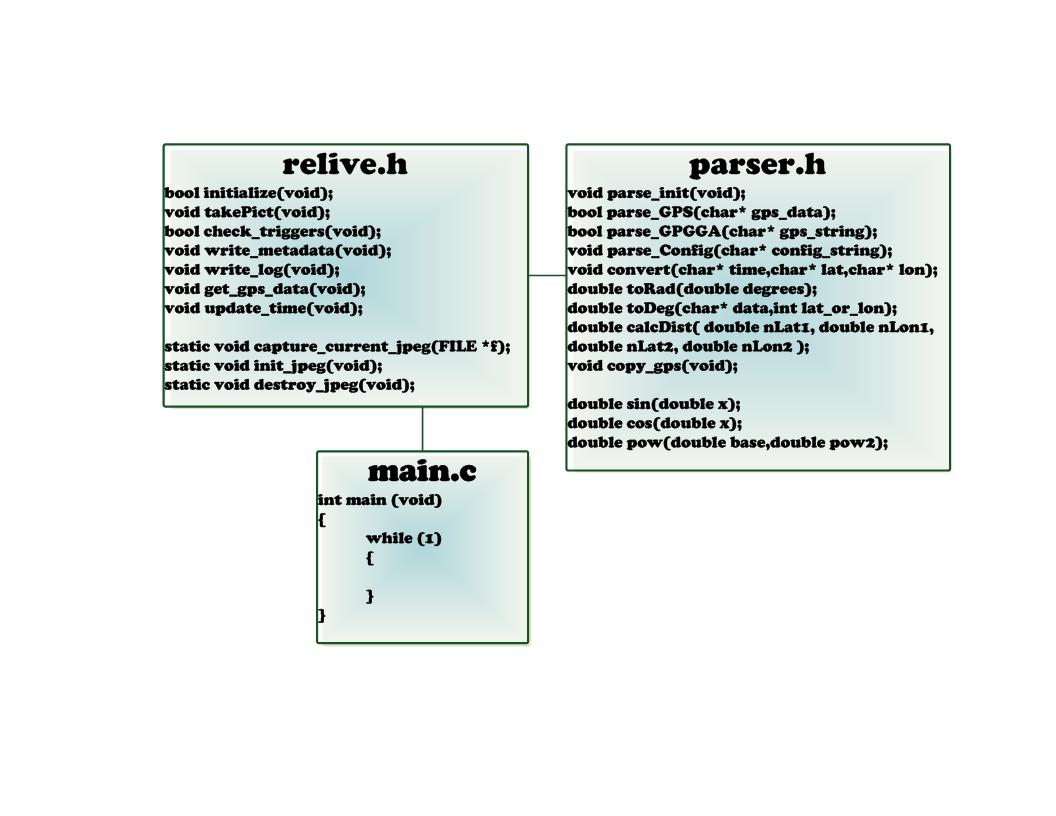


Figure : CMUCam3 program

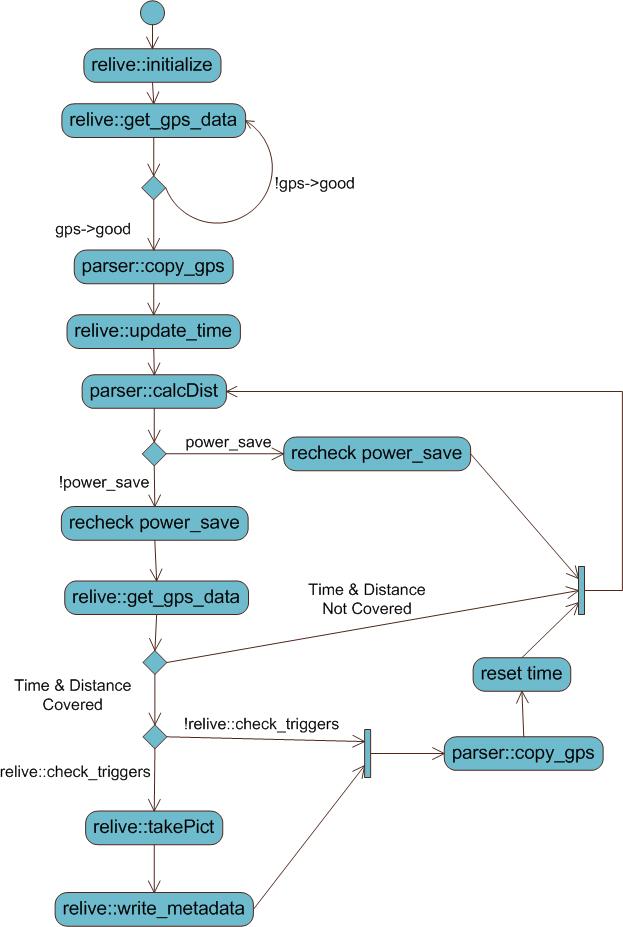


Figure : CMUCam3 Main method

## Software

The client program is written in C# based on the .net 2.0 Framework. The GUI based application allows users to synchronize data from the camera to their preferred computer before uploading it to Google. In order to upload the images to Google and subsequently generate an associated map, the metadata containing latitude, longitude, and timestamps must be read from a file and processed. Once a day’s worth of images has been uploading to a web album, the user can then view and share their information from anywhere via the internet.

Local image browsing on the host computer utilizes the same folder view implemented by windows explorer to keep a strong external consistency within the interface. The second main feature of the software, camera configuration, is kept on a secondary tab hidden away until it is needed. From the configuration panel, users can select different combinations of the available capture triggers, which include minimum time, minimum distance, time based scheduler, and location based halo. Once configured, settings are saved to the SD card for later parsing by the hardware.

Web album browsing through the application is controlled through a windows calendar form. Album dates are downloaded from Google and automatically highlighted as available dates on the calendar, which then allow users to select from available dates. Once a date has been selected, a thumbnail preview of the album is presented as well as additional options to view the entire album or simply just the map of the album online.

Interaction with Google services has been implemented with the use of the Google Data API for .net applications. The following is a breakdown of the various classes used to create the software application:

* 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 activation
* SelectDrive – Graphical user interface to control the proper selection of SD cards
* Google Data API – Contains necessary API for accessing and updating data with Google services

# Experimental Results

(text here) – Caleb & Terrence

# User’s Manual

This manual has been created with the expectation that the user of the system has a basic knowledge of computers. The user is expected to know how to use the input devices of a computer (such as the mouse and keyboard) as well as the understanding of how to properly insert and remove a SD media card from their computer. Since the software will format any removable device when requested, the user should also be able to distinguish between the removable devices connected to their computer. The user is also expected to be using Windows XP or Windows Vista as the product has not been tested on other versions of Windows. The user will also need an active internet connection to be able to upload their pictures to their Picasa web account. The user should be able to maintain an active internet connection and create an account with the Picasa service.

## Software

* Software Installation
* Complete Operation Instructions

## Hardware

### Packaging

The hardware is installed in the reLive system has been enclosed in half of a Radio Shack project enclosure. This enclosure contains the CMUcam3 as well as the connections for the battery pack and the GPS.

<Insert Picture of Enclosure>

### Hardware Operation

The operation of the reLive system should require no tweaking from the user after the user programs the SD media card with the software solution. All programming will be done using the SD media card. The user will be required to have the media card in the reLive system during operation. The software solution will program the SD media card to correctly control the reLive hardware.

The reLive system must be able to acquire GPS signal before it will operate properly. The first signal will be obtained in approximately 45 seconds in while in an area clear of obstructions such as tall buildings, trees, or power lines. After the first GPS signal has been acquired, the system will begin taking pictures as the triggers indicate. If the GPS signal is lost, the previous GPS position will be assumed when taking the next pictures so the reLive system will work in urban areas and inside buildings.

### Setting Triggers

The triggers that are programmed by the software to the media card control the operation of the reLive hardware. These triggers will control when a picture is taken and how the system will act when no GPS signal is present.

The **time trigger** is the basic trigger that will always be enabled. The minimum time trigger is 90 seconds. This trigger will be controlled off of the clock located on the CMUcam3 so it does not depend on a GPS signal.

The **distance trigger** can be disabled by setting the minimum distance to 0 meters. This trigger can be set to any value from 0 meters to 10,000 kilometers. If this trigger is not disabled, a GPS signal will be required to activate this trigger. If a GPS signal is not acquired, the distance cannot be calculated.

The **halo trigger** will allow the user to specify a distance from a specific GPS location as the only area that pictures will be taken. If the GPS is last acquired inside of the halo, pictures will continue to be taken until GPS has been acquired outside of the halo. This trigger can be disabled with a checkbox in the software solution.

The **scheduler trigger** will use the real time clock from the GPS to determine if a picture should be taken. This trigger will only take pictures between the hours specified in the software solution. Like the halo trigger, this option can be disabled through the software solution.

### Changing GPS Modules

While the GPS and CMUcam3 come packaged together in the reLive system, the GPS can be replaced by any GPS that meets the following requirements:

* Powered with 5 V
* Serial output
* Outputs NMEA “GPGGA” string
* No other pins required for normal operation

A new GPS needs to be connected to the pins shown below

<Insert Picture of CMUcam3 Connections>

# Course Debriefing

(text here) - All

# Budget

For the design of this prototype, our overall spending was allocated $500. This total includes not only the parts used in the design, but also any additional costs such as shipping and handling required to order them. Parts were purchased either online or at local electronics shops, which included Radioshack and Midstate Electronics. The following is a compilation of the total cost for the entire design.



Figure : Cost of Design

Although the total cost of design is high, the overall cost of a prototype is reduced by only requiring components that were used in the final product. The primary reductions in cost for the final product are associated with switching to the EM-406A SiRF III GPS Unit, which already has an integrated antenna and greater power than the Trimble Copernicus model purchased initially.



Figure : Cost of Unit

The final cost of a single unit is at a reasonable price point for anyone interested in logging their daily activities.

# Appendices