reLive System

*Final Report*

**reImagine Technologies**

David Moreno

Mario Raushel

Terrence Sin

Caleb Wells

Department of Computer Science

Texas A&M University

May 5, 2008

Table of Contents

[Table of Figures iii](#_Toc197604179)

[1 Executive Summary 1](#_Toc197604180)

[1.1 Need, Goal, and Objectives 1](#_Toc197604181)

[1.2 Final Design and Implementation 1](#_Toc197604182)

[1.3 Results 1](#_Toc197604183)

[1.4 Team Management 2](#_Toc197604184)

[2 Project Background 2](#_Toc197604185)

[2.1 Need Statement 2](#_Toc197604186)

[2.2 Goal and Objectives 2](#_Toc197604187)

[2.3 Design Constraints and Feasibility 3](#_Toc197604188)

[2.4 Literature and Technical Survey 3](#_Toc197604189)

[2.5 Evaluation of Alternative Solutions 5](#_Toc197604190)

[3 Final Design 5](#_Toc197604191)

[3.1 System Description 5](#_Toc197604192)

[3.2 Complete module-wise specifications 7](#_Toc197604193)

[3.2.1 Software 7](#_Toc197604194)

[3.2.2 Hardware 10](#_Toc197604195)

[3.3 Approach for Design Validation 11](#_Toc197604196)

[3.3.1 GPS Test 11](#_Toc197604197)

[3.3.2 CMUcam3’s Camera Test 11](#_Toc197604198)

[3.3.3 Philips LPC2106 Microcontroller Test 12](#_Toc197604199)

[3.3.4 System Test 12](#_Toc197604200)

[4 Implementation Notes 12](#_Toc197604201)

[4.1 Hardware 12](#_Toc197604202)

[4.1.1 Preparing components and environment 12](#_Toc197604203)

[4.1.2 GPS NMEA string parsing 14](#_Toc197604204)

[4.1.3 CMUCam3 main program 14](#_Toc197604205)

[4.1.4 Packaging 17](#_Toc197604206)

[4.2 Software 18](#_Toc197604207)

[5 Experimental Results 19](#_Toc197604208)

[5.1 Software Results 19](#_Toc197604209)

[5.1.1 Description of Tests Performed 19](#_Toc197604210)

[5.1.2 Numerical Results of Tests 19](#_Toc197604211)

[5.1.3 Analysis of Results 20](#_Toc197604212)

[5.2 Hardware Results 21](#_Toc197604213)

[5.2.1 Description of Tests Performed 21](#_Toc197604214)

[5.2.2 Numerical Results of Tests 21](#_Toc197604215)

[5.2.3 Analysis of Results 22](#_Toc197604216)

[5.2.4 Hardware Pictures 23](#_Toc197604217)

[6 User’s Manual 23](#_Toc197604218)

[6.1 Software Installation 24](#_Toc197604219)

[6.2 Complete operation Instructions 27](#_Toc197604220)

[6.2.1 Requirements for SD Card Actions 27](#_Toc197604221)

[6.2.2 SD Card Formatting 27](#_Toc197604222)

[6.2.3 Syncing Pictures From SD Card 27](#_Toc197604223)

[6.2.4 Creating the Configuration File for the SD Card 27](#_Toc197604224)

[6.2.5 Uploading the Pictures onto Picasa Web Albums 28](#_Toc197604225)

[6.2.6 Viewing the Pictures 28](#_Toc197604226)

[6.3 Hardware 29](#_Toc197604227)

[6.3.1 Packaging 29](#_Toc197604228)

[6.3.2 Hardware Operation 29](#_Toc197604229)

[6.3.3 Setting Triggers 30](#_Toc197604230)

[6.3.4 Changing GPS Modules 30](#_Toc197604231)

[7 Course Debriefing 31](#_Toc197604232)

[8 Budget 31](#_Toc197604233)

[9 Appendices 33](#_Toc197604234)

# Table of Figures

[Figure 1: System Design 6](#_Toc197604235)

[Figure 2: Concept Software Design 8](#_Toc197604236)

[Figure 3: Final Software Design 9](#_Toc197604237)

[Figure 4: CMUCam3 10](#_Toc197604238)

[Figure 5: SIRFStar III GPS Module 10](#_Toc197604239)

[Figure 6: Hardware Diagram 11](#_Toc197604240)

[Figure 7: Sirfstar III serial Protocol 13](#_Toc197604241)

[Figure 8: Convert degrees formula 14](#_Toc197604242)

[Figure 9: CMUCam3 program 16](#_Toc197604243)

[Figure 10: Project Box 17](#_Toc197604244)

[Figure 11: Battery Pack 17](#_Toc197604245)

[Figure 12: Final Product 18](#_Toc197604246)

[Figure 13: Halo Trigger Test run 20](#_Toc197604247)

[Figure 14: Distance Trigger Test Run 21](#_Toc197604248)

[Figure 15: Stationary GPS Test 22](#_Toc197604249)

[Figure 16: Hardware Packaging 23](#_Toc197604250)

[Figure 17: Installation - FIRST STEP 24](#_Toc197604251)

[Figure 18: Installation - SECOND STEP 25](#_Toc197604252)

[Figure 19: Installation - THIRDSTEP 25](#_Toc197604253)

[Figure 20: Installation - FOURTH STEP 26](#_Toc197604254)

[Figure 21: Installation - FIFTH STEP 26](#_Toc197604255)

[Figure 22: CMUcam3 Enclosure 29](#_Toc197604256)

[Figure 23: 5V TTL SERIAL Port Connections 31](#_Toc197604257)

[Figure 24: Servo Port Connections 31](#_Toc197604258)

[Figure 25: Cost of Design 32](#_Toc197604259)

[Figure 26: Cost of Unit 32](#_Toc197604260)

# Executive Summary

## Need, Goal, and Objectives

The reLive system is designed to fill the need for users to autonomously create and explore daily photo life logs. This system can be used as a memory aid or an organization tool that will create a daily log cataloging pictures and storing them online. These albums can be shared with others or simply used to organize daily photos.

The GPS on the reLive system adds a position to the photo life log that can be mapped and each image is tied to a GPS coordinate. This enables the life log to contain positional data and the photographs can be displayed on a map to allow the user to track exactly where they were when the photograph was taken.

## Final Design and Implementation

The user will be required to use the software solution to program an SD media card with the desired settings, insert the SD media card into the hardware, and power on the system. When the user is finished with the system, the hardware should be powered off and the SD media card will contain the images taken during the run. The SD media card can then be transferred to the computer and the pictures can be synced to Google’s Picasa Web Albums as well as stored locally to begin creating a photograph life log.

The software was written in C# and interfaces with the Google API to allow the user to upload the organized images to Picasa Web Albums. The software will show a calendar of dates that have previously been uploaded. The user can view their albums through the software, they can sync from the SD media card, and they can upload new albums.

The hardware is a CMUcam3 and an EM406A GPS encased in a modified project enclosure. The final dimension of the enclosure design is 2.75” x 2.5” x 1.5”. The EM406A GPS outputs the NMEA standard GPS data and has a blinking LED to show when GPS signal has been acquired.

## Results

The reLive system successfully integrated a hardware and software solution to organize images into a photo life log. The hardware is powered by 4 AA batteries and lasts for approximately 15 hours. The hardware is enclosed in just 10.3 cubic inches not including the batteries. The software is integrated with Picasa Web Albums to allow the user to upload albums without having to manually open a browser. To view a map of the photographs, the software will automatically launch the user’s default browser and navigate to a web page with a Google map overlaid with the photographs from the user’s log entry.

The system’s hardware will support a variety of picture taking triggers such as time, distance, inside of a halo, and within a specific schedule. All programming of the hardware will be done with the SD media card which can be formatted and programmed with the software solution. The hardware will not need to be connected to the computer to synchronize pictures.

## Team Management

The team was successful in designing a project that met the constraints and requirements set forth at the beginning of the project. The team set goals for time to finish the project ahead of schedule to account for time to test, debug, and account for other unseen problems. The team ran into several issues, most of which were from hardware defects such as a faulty capacitor on the CMUcam3, a GPS that had a difficult time acquiring signal, and an antenna that became separated from the cord. All of these issues were able to be overcome by planning ahead for these unforeseen complications. Effective teamwork was required to overcome these issues since issues had to be openly discussed and the team needed to be in agreement after meeting to solve the problem.

# 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 goal for this project is to create a wearable camera system capable of automatically recording pictures with GPS data based on a variety of triggers (time, distance, and halo).

The following is a list of objectives that were taken into consideration in the design of the device.

* The prototype design should cost no more than $500.
* The prototype must be battery powered and capable of logging pictures for at least 12 hours before 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 option for losses in signal.
* The design must be comfortable to wear and lightweight (less than 1 lb.)
* The accompanying software must be easy to understand and use.
* The camera unit must take good quality pictures (at least 320 x 240 resolution)

## Design Constraints and Feasibility

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 at least 12 hours 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 work well outdoors so the consumer product must be resilient to a variety of weather conditions. Originally, an objective of the system was to be able to use it indoors. However, with GPS difficulties, this option has been limited to an extent. The system should still be usable indoors. However, GPS capabilities and triggers will be rendered useless. Another objective that was deemed unnecessary was the face detection trigger. We had planned to implement this trigger because the CMUcam3 seemed to already have this functionality. However, once the space limitations of the CMUcam3 were discovered, it was decided that face detection would likely be impossible to implement without sacrificing too many essential features.

## Literature and Technical Survey

**Ricoh Pro G3 GPS Camera**

*http://www.ricohzone.com/gps/index.html*

Many of the triggers in the ReLive camera system depend largely on the GPS module that has been integrated with the CMUcam. Recently, cameras such as the Ricoh Pro G3 have been released that embed coordinates of where pictures were taken. The ReLive camera system extends this by not only tagging the pictures with coordinates, but also enabling triggers based off of the GPS data received. This means a higher level of interactivity with the GPS chip.

**Lightning Activated Camera Shutter Trigger**

http://www.solorb.com/elect/lightning/index.html

This camera takes pictures whenever it sees a flash of light. This is related to the ReLive camera system as it triggers a picture to be taken on a certain event. Similarly, the camera system has many triggers that can be set by the user. Just as that camera detects flashes of light, the ReLive camera detects distances and time and takes pictures accordingly.

**Multimedia Diary Application**

http://www.freepatentsonline.com/EP1533714A2.html

This patent combines media items such as videos and photos into a calendar view of events. There is a timeline view from which media files may be viewed from. Just like this patent, our application focuses on pictures and organizes them by date. These dates can be selected through a calendar and viewed online by everyone.

**Audio Logger Pro**

http://www.broadcast.co.uk/audiologgerpro-p-55.html

This application continuously records audio 24/7 using the PC and sound card. Sounds from any time can be played back using a playback client. The ReLive system is similar to the Audio Logger Pro in that it also logs media. Instead of photos, the audio logger keeps track of audio. Just as the Audio Logger can retrieve audio, the ReLive system can retrieve photos from any date and time. One difference though, is that while the Audio Logger must stay attached to a computer, the ReLive camera is very portable and can be taken wherever the user goes.

**Sony Smile Shutter**

<http://www.adorama.com/catalog.tpl?op=NewsDesk_Internal&article_num=012808-2>

Sony’s smile shutter shows just how far face detection has come. In Sony’s new W-series Cyber-shot digital cameras, it will be possible to detect and take pictures whenever a person smiles. It can even detect and give priority to children or adults. The ReLive camera will work through certain triggers set through the software program. A feature such as face detection or smile shutter could be easily extended as another feature. In fact, we had planned to do such a thing since a face-detection algorithm was given. However the camera ran out of space for the face-detection code. With a little more space, the ReLive camera could easily extend its current triggers to many more.

## Evaluation of Alternative Solutions

Throughout the implementation process, problems and new ideas were constantly encountered, changing the design of the system. The only constant was that the system would be built around the functionality of the CMUcam3. The first major decision was that of the GPS chip. Initially, we looked closely at the SiRFstar III, SiRFstar II, and Copernicus chipsets. Factors that were taken into consideration were accuracy, startup time, ability indoors and outdoors, and documentation available. The Copernicus chip was chosen mainly for the immense amount of documentation available compared to the other chips. However, after working with the Copernicus, we realized that it was much more difficult to receive GPS signal with the chip than it had originally seemed. Therefore, we were required to switch to a SiRFstar III chip.

One alternative solution that was not explored too deeply was the use of a camera different from the CMUcam3. However, this would most likely require much more work for required triggers to be programmed since the CMUcam3’s main advantage is in its programmability. With it, many triggers can be implemented to control the camera. This gave us a lot of flexibility as we thought up ideas for how we wanted to use the camera. One camera that was seriously considered was the AVRcam, which was a free software/hardware alternative. This was a very good alternative considering the price difference. However, the CMUcam had advantages in more features such as automatic servo control.

On the software side of things, before we used Google’s Picasa service, we had planned to create our own application and website using Google Maps. However, realizing there were services that already did what we had planned to do, we decided that it would be more beneficial to use the Google API to send and receive data to and from a Picasa account. We also looked into using the Flickr service. There was a downside to using the service though. Although Picasa gave us easy access to a great amount of already implemented features, there were times where we wanted to add our own and were unable to because we couldn’t really change Picasa. We also became dependent upon those services provided by Google. There was a point in time where our album viewing could not occur because a certain Google service was down. Without implementing a roundabout way to bypass using that service, a large part of our application would not run.

Finally, during the enclosure creation process, we explored different types of boxes. Before the final enclosure of our custom-made box, we considered custom acrylic and aluminum project boxes. The aluminum project box had the risk of shorting connections while the custom acrylic was deemed too difficult to create in an aesthetically pleasing way. Our final box is half the size of what it originally was and much closer to what a finished product would be.

# 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 save a configuration file into an SD memory card that the CMUCam3 will use. The SD memory card is the main link between the PC software and the CMUCam3. After inserting the SD card into the CMUCam3 the user will then wear the camera and will not need to further interact with the system. Once the user is finished wearing the system, they will take the SD memory 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 selected on the computer and saved into the SD memory card. The user will be able to program the triggers for time, distance, and time schedule. The user will also be able to specify a beacon and a distance from that beacon to activate the triggers; this will be a halo point.

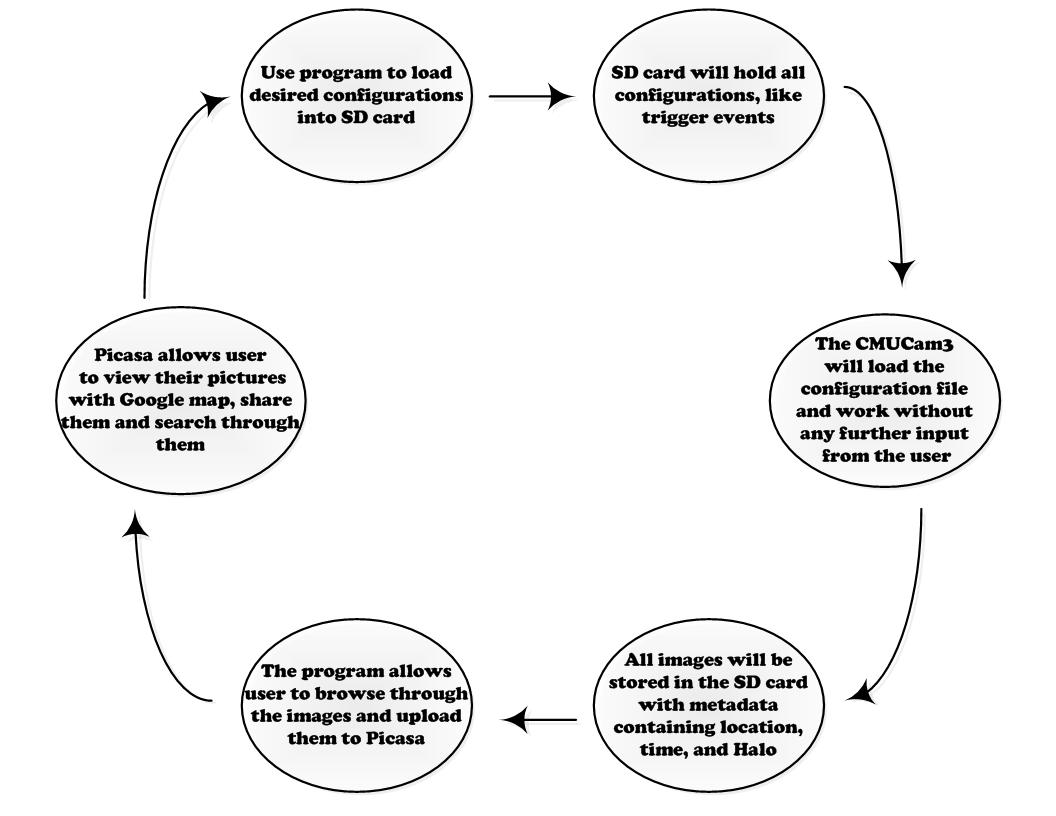


Figure : System Design

Next, user will insert the SD memory card into 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 SD memory card will be transferred back to the computer. With the SD memory card is inserted into the PC, the user will use the reLive software to retrieve the images. The SD memory 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 a Picasa web album. 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. At this point the process will start over again with allowing the user to change configurations settings and reformat the SD card.

## Complete module-wise specifications

### Software

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

### Hardware

The hardware is composed of two main components, the CMUCam3 and the SiRFstar III GPS module. 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, a number of LEDs, and some general input/output pins. This system will be supplied with between 6 and 15 volts of DC power (at least 150 mA). This power will be supplied from four AA rechargeable batteries.

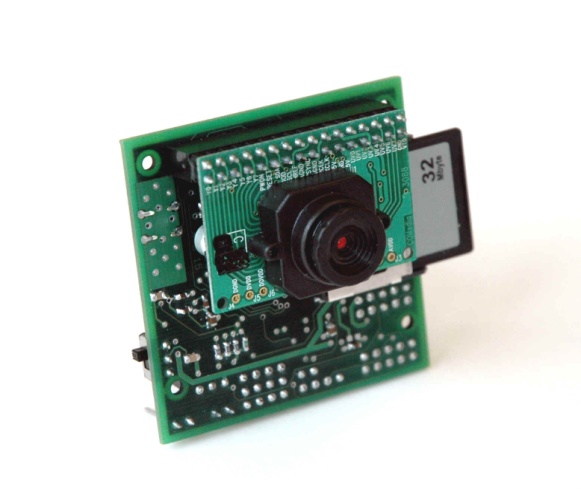


Figure : CMUCam3

This CMUcam3 will connect to a SiRFstar III GPS module with an attached antenna that will allow the CMUcam3 to find its location. The GPS has a horizontal accuracy of less than 10 meters without WAAS and 5 meters with. When starting cold, this module will take an average of 42 seconds to acquire the signal, 38 seconds in a warm start, and 8 seconds during a hot start. This module will need to be powered by a 5 volt source. This chip will give the wearer the ability to acquire their location in most outdoor environments.



Figure : SIRFStar III GPS Module

Previously we were using the Trimble Copernicus GPS module. We decided to change it mostly due to the fact that the Copernicus required the use of a 3.3V voltage regulator circuit and the antenna was separate from the main module. That made the unit much bigger than we wanted and gave rise to complications in packaging it in a small enclosure.

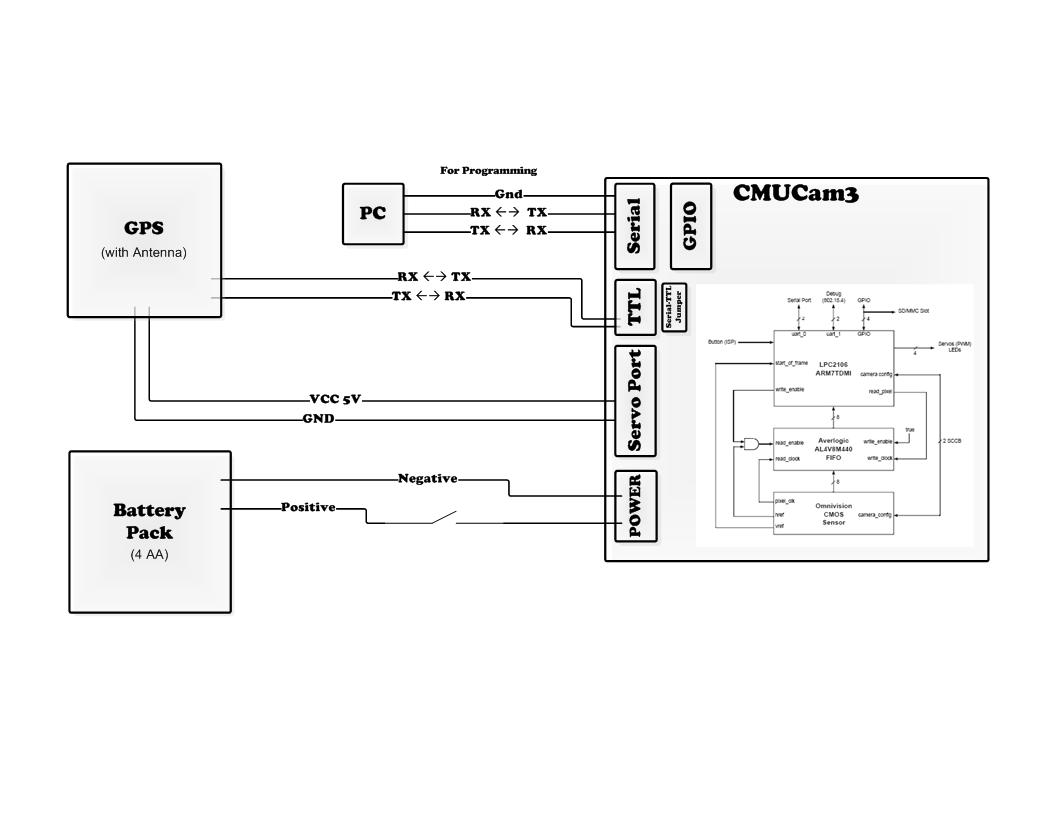


Figure : Hardware Diagram

The CMUCam3 will be programmed through the serial port with a PC. The GPS module will connect to the TTL serial port. Both ports are connected to the same UART in the microcontroller so in order to activate the serial a jumper must be placed on the pins next to the TTL. The GPS will be connected to a 5V regulated voltage supply in order to power it. Both the CMUCam3 and the GPS will be powered by 4 AA rechargeable batteries placed in a battery pack with a switch.

## Approach for Design Validation

### 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

### Preparing components and environment

The CMUCam3 API requires a Linux environment; there is a choice to use Cygwin which is a Linux-like environment for Windows. We have found that setting up Linux for programming was more reliable. The CMUCam3 community has very comprehensive instructions on setting up and getting started. The API consists of multiple libraries that allow programming in C and abstracting the hardware layer. Projects for the CMUCam3 are located in a specific folder, which is where we added our project folder. For the relive program we simplified it to 5 files: parser.h, parser.c, relive.h, relive.c, and main.c. The parser files are dedicated to parsing the GPS strings and the configuration file. Also it will have the functions required to calculate the distance between two points. The relive files are dedicated to all other functionality like SD card management, and communicating with the GPS. The main file will have the main loop that the program runs.

The GPS module has a default setting of outputting the NMEA messages at 9600 Baud but we need it to output at 4800 baud. With a TTL to USB converter and the SiRFstar demo software we are able to configure the GPS to output at the rate we want it to and the messages we want, only GPGGA. The GPS module has a battery that will allow it to save this configuration permanently.



Figure : Sirfstar III serial Protocol

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, a MAX232, to increase the voltage of the signal from levels 0V – 5V to +/-15V. 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. UART1 does not support TTL devices that run at 5V so we will also have to use UART0 to communicate to the GPS Unit. For debugging we will have to keep a log in the SD memory card that we can later look at.

After opening the serial ports in the CMUcam3 it is easy to access its data. By using the stdio we can easily read and write from the UART with commands printf, and scanf. The GPS Units can be configured by sending standard NMEA Messages that the GPS can understand. We have already configured the GPS like we want to so no need to use them.

### GPS NMEA string parsing

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.

The string that the GPS will be configured to send will be the GPGGA string. This string is the Global Positioning System Fix Data. This string will begin by identifying the string type ($GPGGA). 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 sixth field identifies the validity of the entire sentence. In the GPGGA sentence, there are three possible values. A ‘1’ in the field indicates a GPS fix is obtained. A ‘0’ indicates invalid data. A ‘2’ in the field indicates a DGPS fix which is more precise than a regular GPS fix. The latitude, fields 2 & 3, and longitude, fields 4 & 5, in the string are in degrees and minutes. This value must be converted to only degrees prior to being used.



Figure : Convert degrees formula

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.

### CMUCam3 main program

The software solution that will be running on the CMUcam3 will rely on a configuration file that will exist on the SD memory card. For the software to begin execution of the main loop, the SD memory card must be detected. If the memory card is not detected it will try loop until it does detect it. The CMUcam3 API allows for the detection of a compatible SD memory card if it is formatted to FAT16. Once the SD memory 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 trigger preferences that were programmed to the SD memory card using the software solution on the computer.

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 GPS module will have an LED that will stop blinking.

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, and time of the picture and save this data to the SD memory card. If it was taken in a halo it will also record that information. There will be enough information in the metadata file to link a picture to the GPS data so the software on the computer will have the ability to tag the pictures.

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 a time trigger, distance trigger, halo trigger, and a schedule trigger.

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. This won’t be much of a problem considering that the minimum time delay between pictures is 1 minute 30 seconds. That will limit the system to only have 40 pictures per hour directory. The images will be organized first in a day directory followed by an hour subdirectory. Currently the CMUCam3 doesn’t have the ability to create directories so the hour subdirectories will be created by the software program.

After reading the configuration file the system will begin by waiting for the GPS to acquire the initial GPS signal. While this signal has not been acquired, an LED on the CMUCam3 will be off and the one on the GPS module will be steady lit. When the first GPS signal is acquired that position will be saved as the current and previous GPS fix. That will conclude the initialization stage, now it will start the main loop.

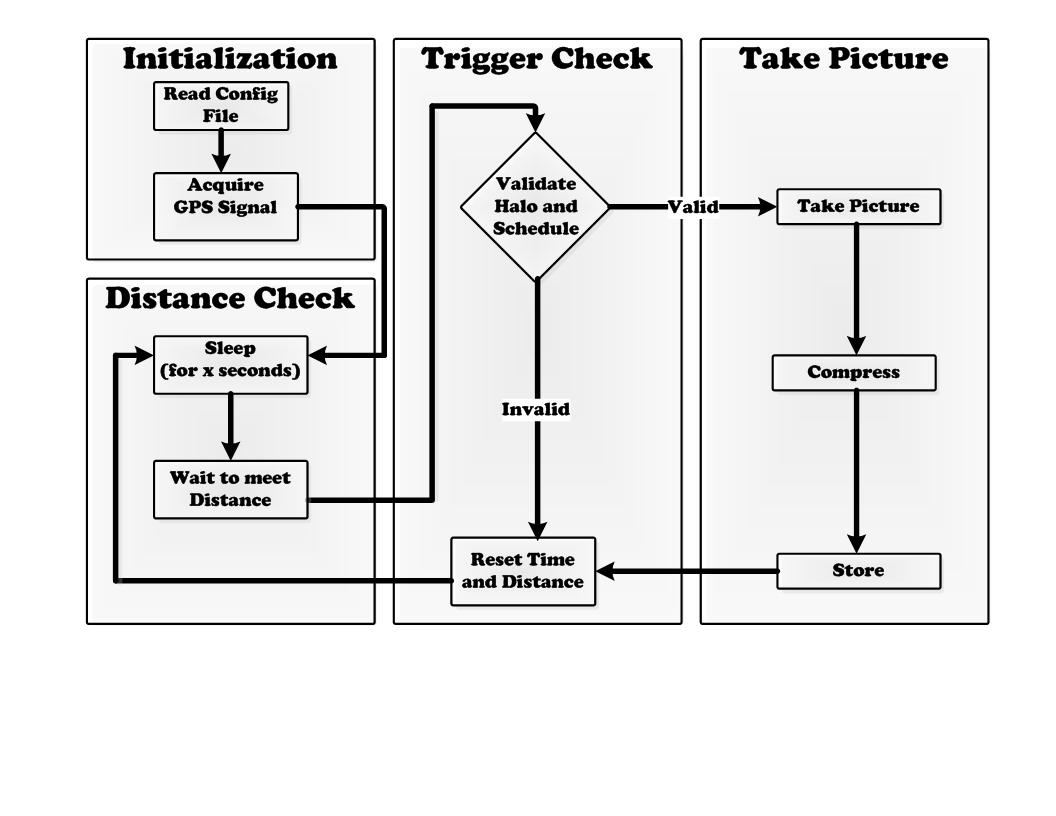


Figure : CMUCam3 program

Upon entering the main loop it will first put the CMUCam3 to sleep for the delay, it will wake up 8 seconds before to give camera time to focus. Putting the camera module to sleep will save much power usage since it is the one that consumes the most. After waking up it will loop without taking pictures until the user has covered the minimum distance specified. We will call this the distance check stage.

After it has waited the minimum delay and the user has covered the minimum distance it will enter the trigger check stage where it will first validate the halo and schedule trigger. It will be valid if they are both disabled. If either one is enabled the requirements must be met in order for it to be valid. If it comes out invalid it will just reset the change in time and distance. The system must act like it took a picture in order for it to go back to sleep. For example if we had the halo enabled then it would stay awake waiting for the user to enter the halo zone before it can take a picture and go back to sleep.

If the triggers are valid it will enter the take picture stage. There it will take a picture compress it with the JPEG library, and store it and its relevant data in the SD memory card. After that it will reset the change in time and distance. That is the end of the current iteration it will loop by going back to sleep.

If at any point it can’t open or close the files from the SD card, in example it the SD card has been removed, it will enter an infinite loop. User must restart the CMUCam3 to restart the program. The CMUCam3 will keep a file called picNum.txt so that when it is turned off and turned on without reformatting the SD card it can go ahead and start the picture numbering where it left off. Otherwise the CMUCam3 would overwrite pictures and the metadata file would not be consistent with the images.

### Packaging

The packaging of the system took much creative work. For starters we will discuss the components used. At a local RadioShack we found a 5x2.5x2 Project Box and a nice enclosed battery pack for 4 AA batteries with an exterior switch.



Figure : Project Box



Figure : Battery Pack

Cutting the project box so that it matches the CMUCam3 length and width significantly decreased the volume of the enclosure. The battery pack was glued to the back of the smaller enclosure. To make it as neat as possible after cutting the hole for the lens, the SD memory card, and for wires to go through, epoxy with scraps of plastic was used as filler. It was then sand smooth. For a more aesthetically pleasing look it was painted with the best color, maroon.



Figure : Final Product

## 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

## Software Results

### Description of Tests Performed

The software was tested mainly in two different areas. First was testing for its usability. Second was testing for its speed. Numerical results came mainly from speed tests. This includes testing the speed of the SD card synchronization and speed of the upload. However, many of these numbers would most likely change depending on the connection and type of computer the user has. However, it is still important that the speed be satisfactory for what the user is typically used to. Usability tests were difficult to quantify but occurred through testing by the hardware team who were more unfamiliar with the software. These types of tests resulted in many error findings that went undetected when the software team tested the program.

### Numerical Results of Tests

|  |  |
| --- | --- |
|  | Time ( seconds ) |
| Login | 8.62 |
| Card Format | 8 |
| Photo Synchronization (For 40 pictures) | 33.62 (0.84 each) |
| Photo Upload (For 40 pictures) | 49.3 (1.23 each) |

### Analysis of Results

For the speed results, the login time was a bit concerning as it took a good bit of time. The format, photo upload, and sync speeds are actually ok since the user is still able to interact with the application while the processes occur. Pictures are synchronized and uploaded at about 1 second per picture. This doesn’t seem too slow and should be ok for the average user. Again, the results would vary with different connection speeds.

As for usability, the software has been thoroughly tested through usage with each hardware test run. A couple successful pictures are included with the final results of our album shown.

****

Figure : Halo Trigger Test run

****

Figure : Distance Trigger Test Run

## Hardware Results

### Description of Tests Performed

The hardware was first tested on a component by component basis. The CMUcam3 was tested by capturing an image, compressing it, and saving it to the SD memory card. Once the image was saved, it was analyzed for size and quality. To measure the quality, each test image was manually examined for major defects. Each test image’s size was also averaged to estimate the number of pictures that can be stored on the SD memory card.

To test the accuracy of the GPS, the device was set outside and a beacon was set at an arbitrary point. The distance calculated from the beacon was used to ensure the GPS was operating within the specifications described in the datasheet.

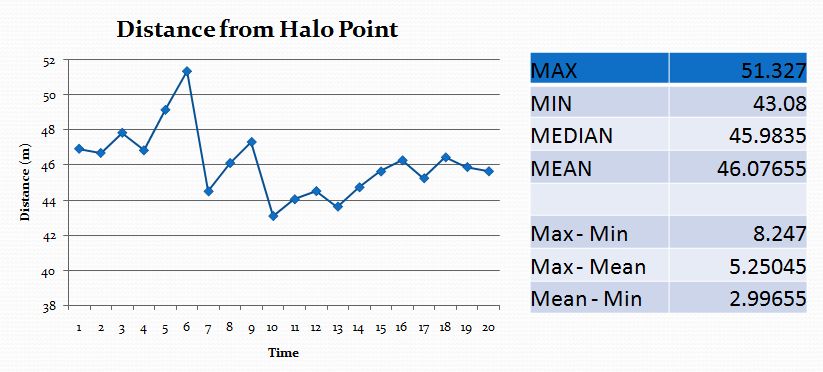
The battery life of the camera was also tested by setting the time trigger to the lowest setting and leaving the camera on for as long as possible. Once the batteries died, the log would be checked for when the camera stopped working.

### Numerical Results of Tests

Each image was captured at 352x287 pixels at 96 dpi and a bit depth of 24. This was verified on 48 test images taken outdoors. Each image was on average 20.4 kilobytes. When the images were manually inspected 3 out of the 48 (6.25%) images were found to have major defects, all of which were due to the camera facing the sun. The effects of motion blur were minimal, only 8 out of 48 (16.67%) had discernable blur.

Battery life was tested to be 15 hours with pictures being taken every 1 minute and 30 seconds.

The GPS test, showed an error of less than 10m. The average distance from the halo was 46.08 meters while the maximum was 51.33 meters and the minimum was 43.08 meters.

Figure : Stationary GPS Test

### Analysis of Results

The image size was the maximum resolution available from the CMUcam3. While a higher resolution image would be beneficial, the contents of the image could be viewed and distinguishing features on faces were easily identifiable. The autofocus tool on the CMUcam3 allows the camera to change environments and adapt. The autofocus has issues when it is in very bright environment or when facing a bright light source. The images do not come out in a dark environment due to the lack of a flash.

The because of limitations of the CMUcam3’s file system, the maximum number of images per directory was limited to 128. This will allow for a maximum of 3072 images on the SD media card. At 3072 images, assuming the average size found during the test, the SD media card would be required to have 62668.8 kilobytes or about 100 megabytes of free space. The system will ship with a 1 gigabyte SD media card, so space will not be a factor for images at this resolution and compression. To fill a 1 gigabyte media card, the user would need to store 49020 pictures (averaging 20.4 kilobytes per image).

The GPS offered a one sigma error of 10m. This error was verified with the test run with the difference of the maximum difference from the halo and the minimum distance from the halo was only 8.247 meters. This error is sufficient for the view from Google maps since a few meters is hardly distinguishable.

### Hardware Pictures



Figure : Hardware Packaging

# 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 memory 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 Installation

To install the ReLive camera software, first click on the reLive Installer Windows Installer Package and follow these instructions:



Figure : Installation - FIRST STEP

Click Next



Figure : Installation - SECOND STEP

Browse to the correct installation directory, select whether the installation will be made public, and press next.



Figure : Installation - THIRDSTEP

Click next



Figure : Installation - FOURTH STEP

Wait until the installation completes.



Figure : Installation - FIFTH STEP

Close the installation window. The reLive software has been successfully installed onto the computer. Look for an icon or the desktop or navigate to it through the start menu in order to run the program.

## Complete operation Instructions

For operation instructions, please refer to Figure 3 for a full view of the software application.

### Requirements for SD Card Actions

*-In order to perform the following actions, an SD card must be selected.*

*-Any SD card with a name of “Relive” will be automatically selected and used when the SD card manipulation actions are performed.*

*-If no SD card with the name “Relive” is detected, you will be prompted with a list of all detected removable drives that may be used by the application.*

### SD Card Formatting

In order to format an SD card, press the *Format SD Card* button in the lower right hand corner.

Formatting the SD card will delete all previous content on the SD card and create a new fat16 file system that is usable for the Relive camera. Twenty-Four folders will be created for every hour of a day.

### Syncing Pictures From SD Card

***-Must have metadata.txt file on SD card (should be created by the CMUcam during usage)***

To sync with the SD card, press the *Sync Pictures From SD Card* button in the lower right hand corner.

The SD card synchronization process creates a folder in your Relive default directory (~\My Documents\My Pictures\reLive) according to the date. All pictures within the SD card will be copied into this folder along with the metadata.txt file.

### Creating the Configuration File for the SD Card

In order to create the configuration file for the camera system, select the *Camera Settings* tab. Once there, check the enable checkbox next to each trigger desired. Fill in the information required for every enabled trigger. Once finished, press the *Save Config* button. Below is a list of the possible triggers and their functions.

-*Time Delay (Minutes)*: This option corresponds to the time trigger describing the amount of time the camera must wait between each picture taken. A list of recommended delays is presented but it is also possible for the user to directly enter a delay.

-*Minimum Distance*: This setting corresponds to the distance trigger describing the distance a user must travel before another picture is taken. The distance should be entered in meters.

-*Scheduler*: This setting specifies during what times the picture should be taken. A start and end time must be specified and set for the configuration file.

-*Location Halo*: This setting specifies the center and radius of a halo where the picture must be taken. The user can either directly enter the latitude and longitude or enter the address of the halo’s center and press the *Get GPS* button. The *Range* field should be entered in meters. The halo’s center can also be viewed when the *view* button is selected.

### Uploading the Pictures onto Picasa Web Albums

*-Pictures should first be synchronized from the SD Card first along with the correct metadata.txt file*

Upload simply by browsing to the desired dated directory containing the pictures and metadata.txt file (actually go inside the directory) and pressing the *Upload* button just left of the calendar. Pictures will be uploaded to a new album using the date in the directory’s name. Only one album can be created for each day so albums should not be uploaded until the user is sure he won’t be taking anymore pictures that day.

One important note to make is that there is a limit of 250 albums if the Picasa account used by the user is a free account. Each album also has a maximum limit of 500 photos. This means that the user should not take more than 500 pictures in a day and will not be able to store more than 250 days worth of pictures.

### Viewing the Pictures

-Selecting the *view web albums* link just right of the *login* link in the top left corner will pop up a window showing all uploaded web albums.

-Uploaded web albums can also be viewed through the calendar in the top right corner of the application. Dates with albums are bolded. Once a bolded date is selected, the album’s preview picture will be shown below the calendar.

-Each album itself can be viewed by selecting the *View Album* link under the album preview picture (after the bolded date has been selected). Pictures will be shown ordered by time.

-Albums can also be viewed by selecting the *View Map* link. Picture thumbnails will then be shown on a map in the location they were taken. When selected, each thumbnail will enlarged.

## 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.

****

Figure : CMUcam3 Enclosure

### Hardware Operation

The operation of the reLive system should require no tweaking from the user after the user programs the SD memory card with the software solution. All programming will be done using the SD memory card. The user will be required to have the SD memory card in the reLive system during operation. The software solution will program the SD memory 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 SD memory 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
* 5V TTL output
* Outputs NMEA “GPGGA” string
* No other pins required by GPS for normal operation
* 4800 Baud, 8 data bits, no stop bit, and no parity bit

A new GPS must be connected to the SRX and STX pins on the TTL connection with the serial bypass jumper removed and powered by the 5V pin and ground from servo port 0

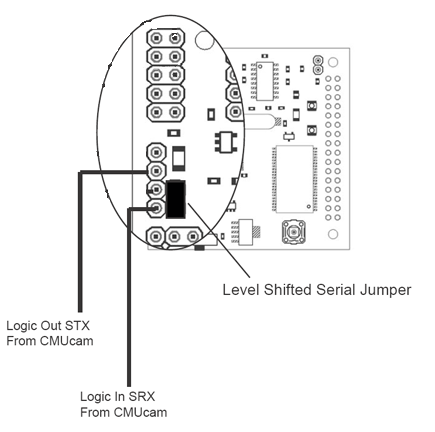


Figure : 5V TTL SERIAL Port Connections

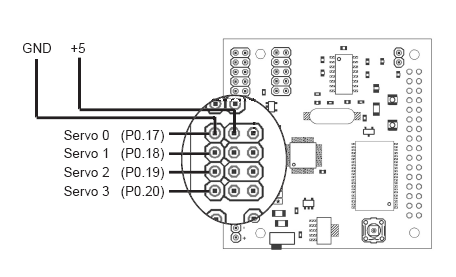


Figure : Servo Port Connections

# Course Debriefing

If you were to do the project again, what would you do the same, what would you do differently?

One of the main challenges towards the end was making the product as small as we could. A big contributor to the size then was the Copernicus GPS, which we later replaced. After working with the SiRFstar III module we realized that we could have avoided some of the earlier complications if we were to start with that GPS module.

Are there any particular *safety* and/or *ethical* concerns with your product(s)? What steps did your group take to ensure these concerns were addressed? Are there any additional steps you would have taken if you were to do the project again?

One of the main safety concerns was any possibility of it exploding and harming the user because of the batteries. We decided to use AA batteries because they are subject to extensive test by the manufacturer to prevent much risk to the consumer. Since the battery life came out to be better than expected some more research on a smaller equally as safe voltage supply would have benefited us in keeping the size small.

Did you test your product(s)? Do they work as proposed? Can you think of any relevant situations in which you haven’t tested your product(s)? If you were to do this project again, what additional *verification* and *testing* procedures might you add?

We completed the entire set basic test to ensure that the product would function at least at a basic level. We were very pleased to have most of the test work out, and the ones that did not we fixed right away. We did not have time to test the battery consumption in more varied situations. If we were to do it again we would think of many more situations in which the unit would consume more power.

# 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