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 that will contain 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.

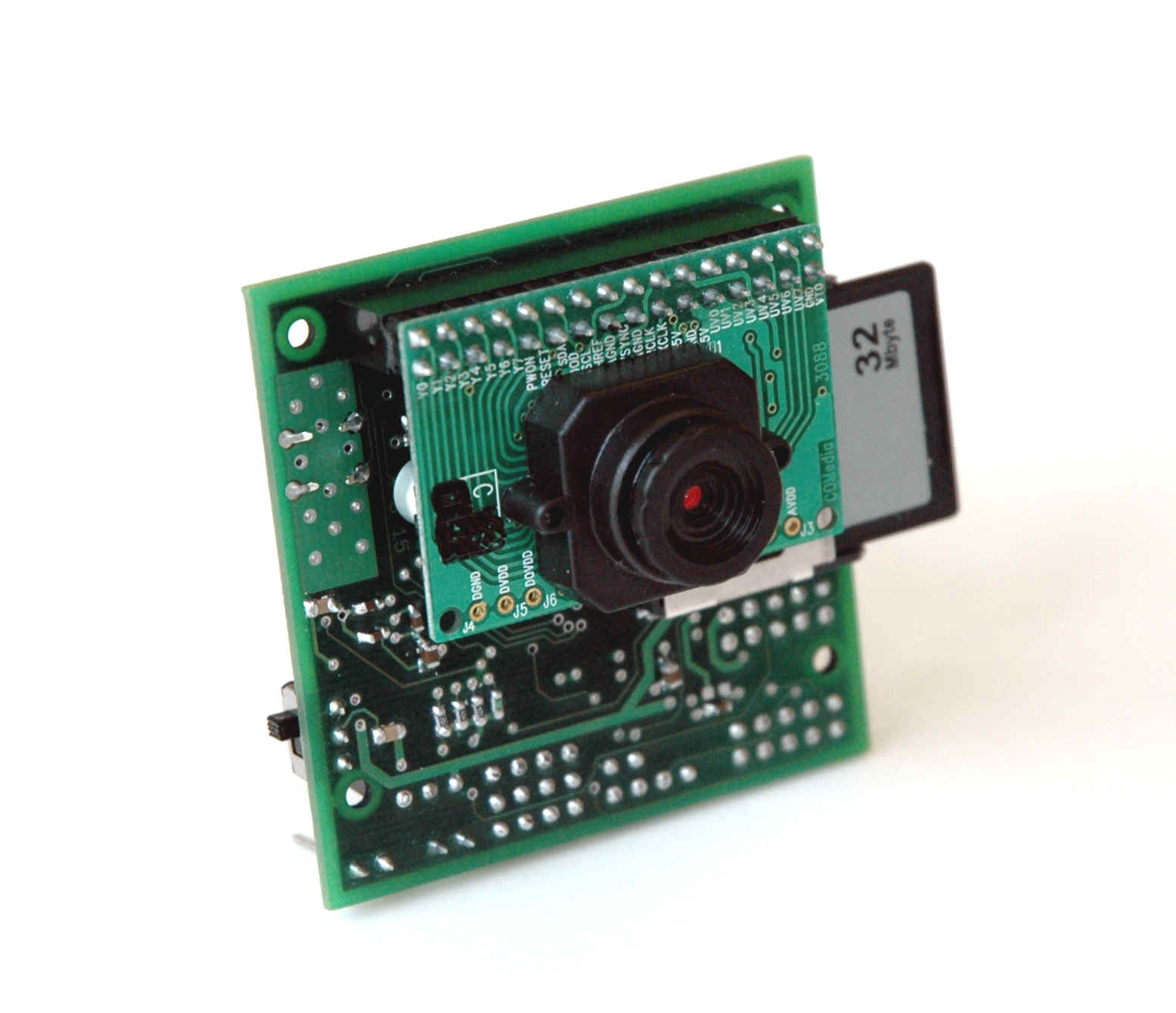
Next, user will transfer the media card from the PC to the CMUcam and power on the system. The ReLive system will then proceed with the commands given to it by the user during the initial step. This step will perform all of the image processing functions. The user will not need to have any interaction 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.

The ReLive system will be run from a bundle of hardware called a CMUcam. The CMUcam 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 CMUcam 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 CMUcam will connect to a Copernicus GPS module that will allow the CMUcam 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.

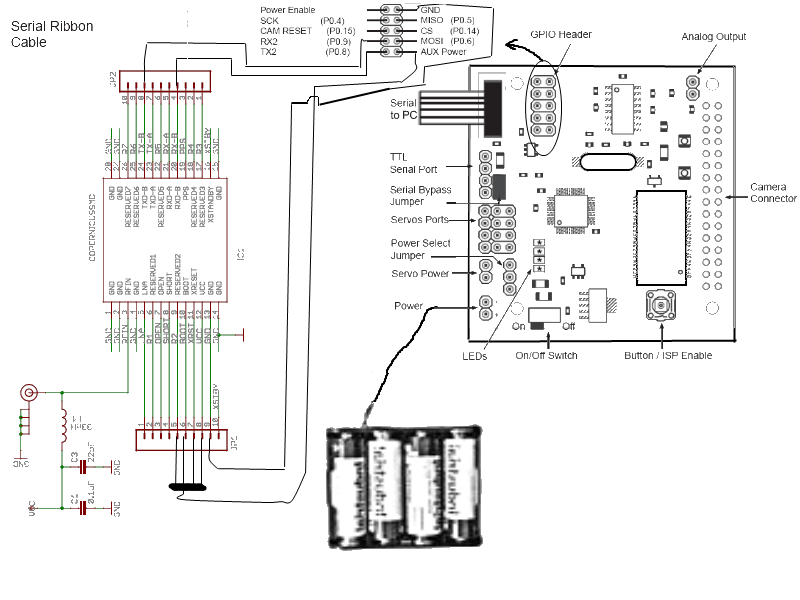


The software solution that will be running on the CMUcam will rely on a configuration file that will exist on the media card. For the software to begin execution, the media card must be detected. The CMUcam API allows for the detection of a compatible media card if it is formatted to FAT16. Once the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

This configuration file will contain information needed to inform the CMUcam 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 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 CMUcam 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 CMUcam 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 CMUcam. 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 CMUcam 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.