Design Details of Software System

The Software system will be implemented in two stages. The first stage involves completion of the minimal requirement for the system. This stage involves fully implementing and fine tuning a plugin for Image J which is an open source image processing software. The plugin computes the goniophotometry from two images taken by the camera. The current plugin is incomplete and still needs some added features, the current additions being worked on include:

- Region Selection for selective computation and aggregation
- Baseline calculation
- Adding a Clear feature

However, more features will be added to this plugin. The camera's program remains untouched and is used to observe the image, adjust exposure level and capture both the bright and dark images (the images differ with the angle of the polarized light).

This minimal solution is an improvement over the previous existing solution, which uses three programs for computing the goniophotometry of a given sample: The camera software, MathCad and Image J. This was cumbersome, and MathCad had a huge delay when computing the desired values for the goniophotometry. The java plugin aggregates the functionality of both image processing and math computations. It can also be used as a stand alone java application. It prompts the user to enter the dark and bright images. The user then inputs a couple of parameters, some of which correspond to the camera settings. It then does the goniophotometry calculations and displays a table with the results as well as a plot. Figure 1, shows a snap shot of the plugin and figure 2 shows the flowchart of the program.

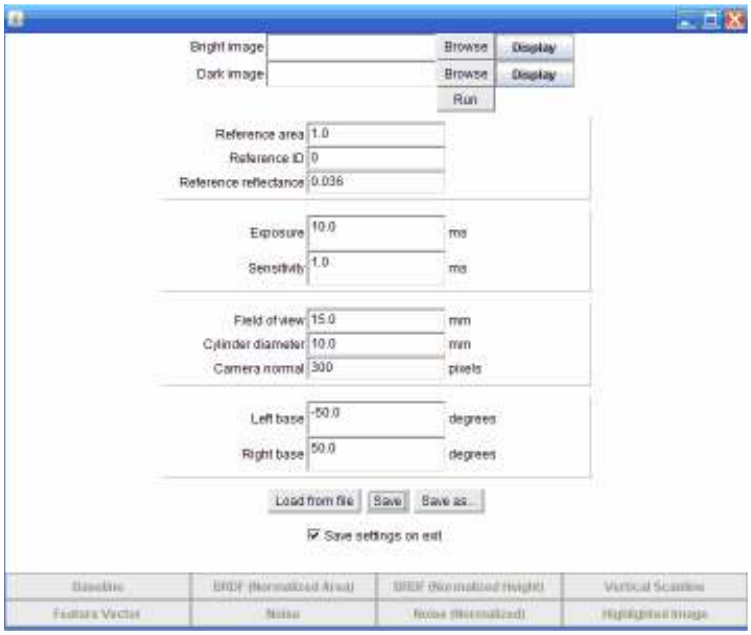


Figure 1. Image J plugin snapshot

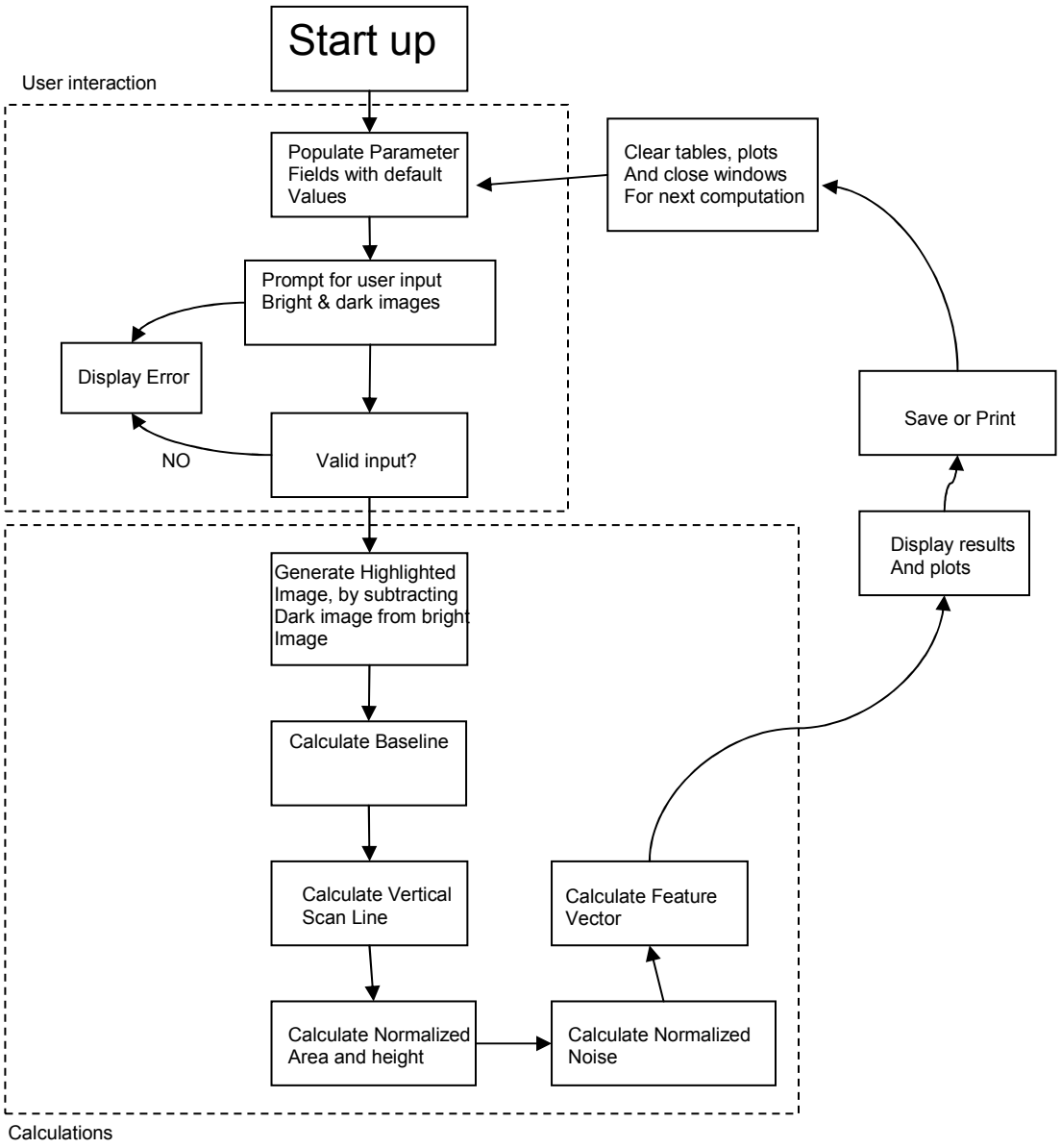


Figure 2. Flowchart for Image J plugin

Risk Assessment of minimum Solution:

The current solution involves the improvement of the existing Java plugin. This is the minimum software requirement for the project and has little risk associated with it.

However since the plugin has been separated from the camera software, there may be a filetype compatibility issues with Image J and camera software (since this can change from user to user). This is a low risk and a file converter integrated into the system would be a mitigation plan.

High Risk Second Stage

Depending on the time of completion of minimum solution, a high risk solution will be started. An ultimate customer need for this product is for it to be easily used and to automate the procedure as much as possible. This second step is a huge iterative step towards an automated simple to use system. This involves software and hardware aggregation to automate the whole system. This would involve adding an extra USB interface to monitor the mechanical parts of the system. C# or Java will be used as the programming language as well as BASIC to control the servo motors and monitor the positioning of parts. Right now, a basic stamp will be used to control the servo motor and a USB interface board will be used to communicate with the stamp and the PC. The USB interface board ports will be connected to stamp ports and will be used to trigger events and check the status of the motors. Therefore the final program will be able to control the servo motors remotely.

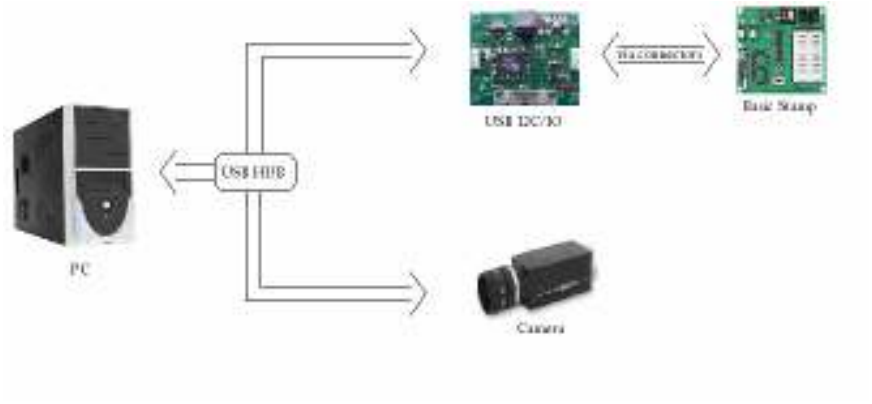


Figure 3. USB network

The resulting program should look similar to the interface of the camera software, and will allow the user to adjust exposure levels and well as focal length of the lens. It would have a manual step through interface to walk through the system step by step and an "Automate all" button. The program will include the library for the camera software and all the classes for the Image J plugin as well as a dll file to allow for USB interfacing. The program first of all will reset the position of the polarizer and return the exposure and focal length values to default. The user will then modify the camera settings and parameter fields or leave as is. Once the "automate all" button is pushed, the program captures the first image. It would then send a signal to the USB interface board which will then trigger and event in the stamp to rotate the polarizer to the next position. Once the stamp is done, it will send a signal back to the USB interface which will then make the program to send a signal to the camera to capture the next image. After this the goniophotometery calculations would be done automatically and the results will be displayed. Figure 4 shows the flow chart for the final system.

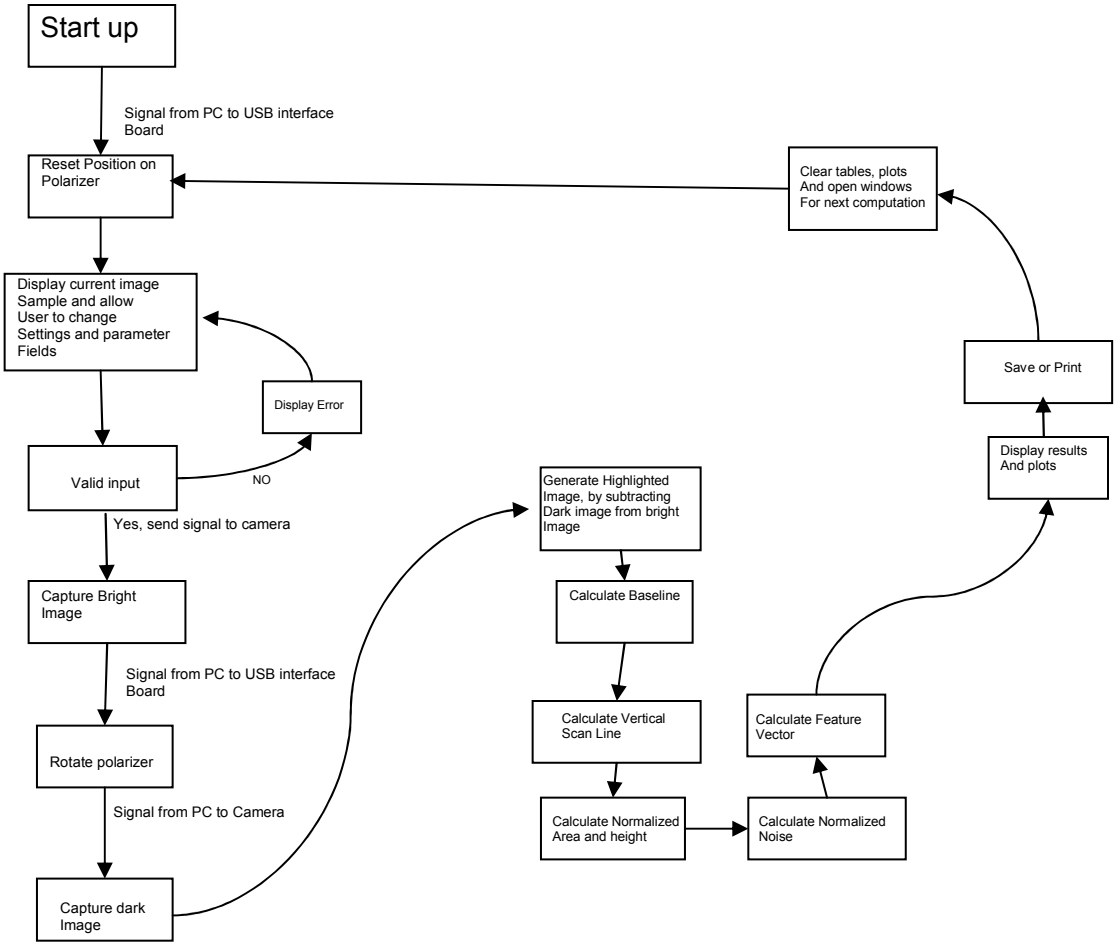
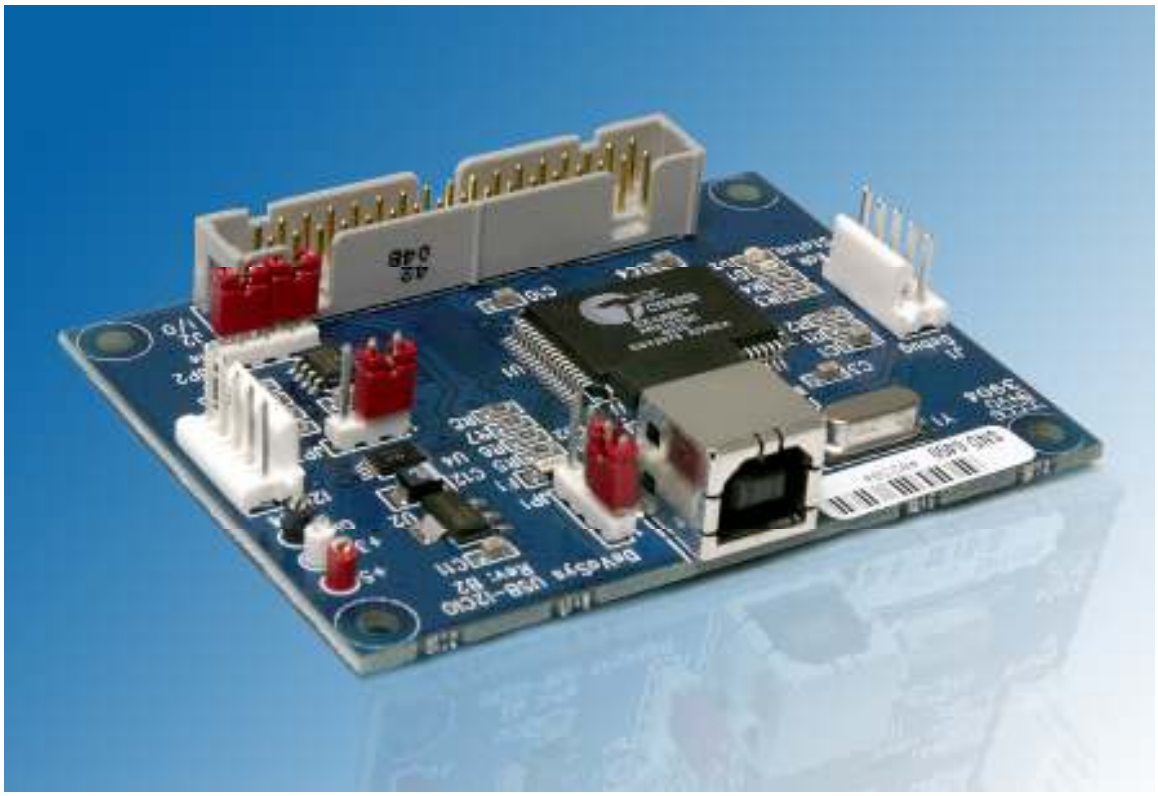


Figure 4. Condensed flowchart of whole software system

Tools and Parts:

1. Eclipse: development environment
   - Java has be chosen as the development language and eclipse works very well with the language

2. USB I2C/IO
   - This board was chosen because it comes in a bundle with driver installs, sample code, and the ports available with the board can easily connected to the basic microcontroller to be used.



Key Features

- 12Mbps USB interface to host P.C.
- Cypress AN2131QC micro-controller.
- 20 bits of user configurable, digital I/O, via a commonly available, 34 pin connector.
- 90Kbps I^2C interface, onboard 16KB I^2C eeprom, 5 pin connector for attaching external I^2C hardware.
- USB Status LED, lights on enumeration, blinks to indicate USB traffic, off when suspended.
- Break LED, useful when developing new firmware, also will be controllable via our API software
- Small form factor (3.0" X 2.25") with 0.125" mounting holes in corners.
- Downloadable board firmware, simplifies updates and allows for code customization.
- Included API (applications programming interface) software gets your application up and running fast!

Applications

- USB to I2C bridge for interfacing to a wide variety of I2C components.
- USB to digital I/O bridge for interfacing to switches, LEDs, and other hardware.
- USB to FIFO bridge for interfacing to FPGA's and other hardware.
- Low cost USB development platform for AN2131QC (note: no address bus availability).
- Test fixture interface.
- Rapid prototyping interface.

- Data Acquisition.