Google Glass Project

Fabio Busignani

March 8, 2015

The *organs-on-a chip* platforms seek to recapitulate human organ function at micro-scale by integrating microfluidic networks with three-dimensional organ models, which are expected to provide robust and accurate predictions of drug/toxin effects in human bodies. In fulfilling this aim, a set of physical/chemical parameters need to be monitored and stored in order to capture such effects of drug/toxin administered into the system.
Such kind of monitoring and storing process has been designed using the Google Glass platform as illustrated in (Fig.1).

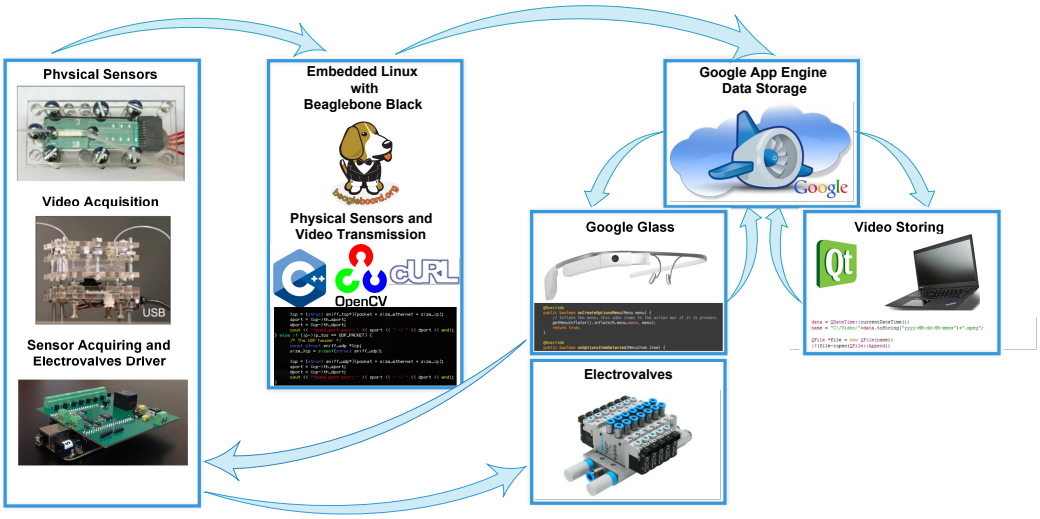


Figure 1: Block diagram of the system

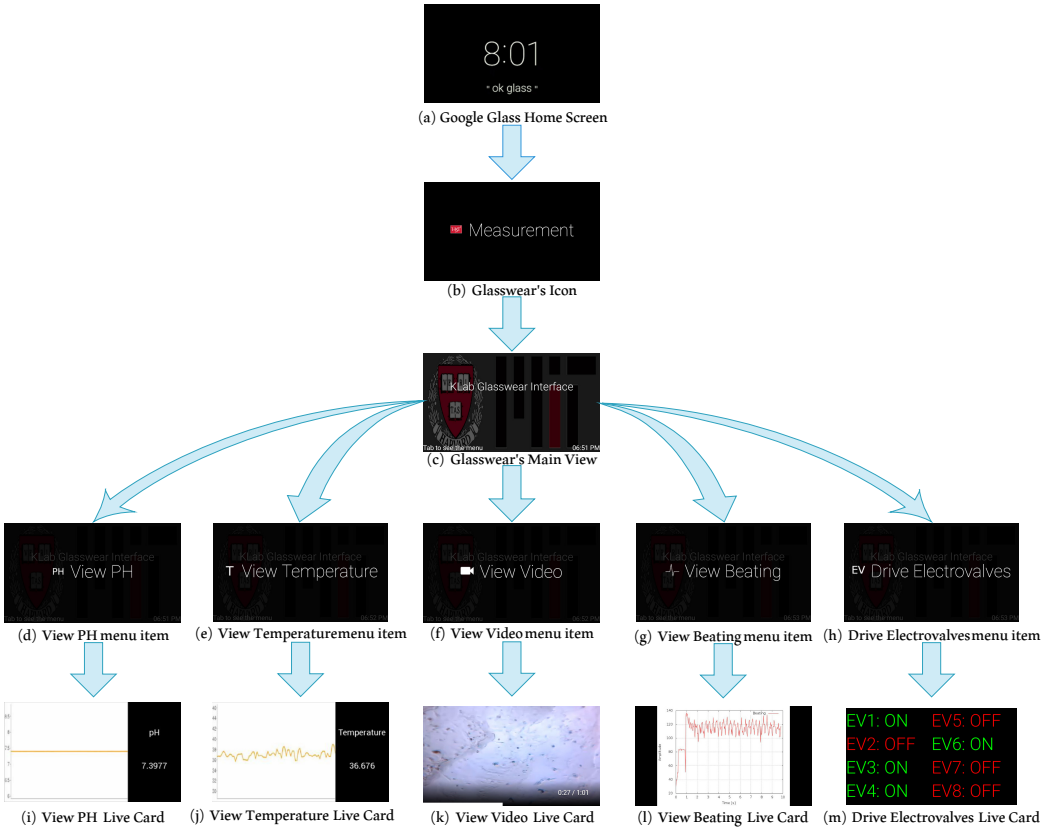
The (Fig.1) shows the principal steps of data transmission from physical and video sensors to the Google Glass via an *Embedded Linux System* performed using the **Beaglebone Black**.
The Beaglebone Black runs processes that are in charged to:

- acquire the sensors value and to store them onto *Google App Engine Data Storage*;
- acquire the video, perform the beating plot, and to store them onto *Google App Engine Data Storage*;
- get from the *Google App Engine Data Storage* the electrovalves status set from the user through the Google Glass and to drive the electrovalves.

The whole designed environment includes a program, written using the framework *Qt*, for storing the recorded video from microscope.

Preview of the Glasswear

The (Fig.2) shows the structure of the Glasswear. From the Home Screen (Fig.2a), using the voice trigger "*Show Measurement*" or tapping on the "*Measurement*" card (Fig.2b) user is allowed to enter in the application (Fig.2c). From this point, tapping and swiping, it's possible to navigate into the glasswear's menu (Fig.2d-h) and choose which card has to be shown. *View PH* (Fig.2i) and *View Temperature* (Fig.2j) cards plot on the card's left side the value of pH and temperature, respectively. While on the right side they show the average value. The microscope's video is shown by tapping on *View Video* (Fig.2k). The *View Beating* card (Fig.2l) shows the graph of the beating associated to the video.



(a) Google Glass Home Screen (b) Glasswear's Icon (c) Glasswear's Main View (d) View PH menu item (e) View Temperaturemenu item (f) View Video menu item (g) View Beating menu item (h) Drive Electrovalvesmenu item (i) View PH Live Card (j) View Temperature Live Card (k) View Video Live Card (l) View Beating Live Card (m) Drive Electrovalves Live Card

Figure 2: Glasswear's Block Diagram

From the *Drive Electrovalves* card (Fig.2m), the user can set the value of each electrovalve. The main view of this card shows the status of each electrovalve (written in green if it is on and in red if it is off).

(a) Elecrovalves Status View (b) Toggle EV1 menu item (c) Toggled EV1

Figure 3: Drive Electrovalves Steps

The (Fig.3) shows the steps to toggle the status of the first electrovalve:

1. (Fig.3a) shows the initial status of the whole electrovalves (all off);
2. tapping on the card and swiping the user is allowed to change the status of each electrovalve from the menu, as shown in (Fig.3b);
3. after that the electrovalve has been chosen, a toast message pops up (Fig.3c), and the new values of the electrovalves are shown.

To return on the main card of the glassware, the user has to tab on *Back* item (Fig.4) from every menu.

Figure 4: *Back* menu item

To terminate the glassware, from every menu, the user has to swipe up to the final item and tab on *Exit* item (Fig.5).

Figure 5: *Exit* menu item

Experiments With Driver for Electrovalves

In order to try the reverse control from the Google Glass to the electrovalves we made different kind of experiments.

LED Experiments

First of all we tried the circuit on a breadboard using LEDs instead of electrovalves. The aim of this step is to demonstrate that the firmware running on the Beaglebone Black, the Java code running on the Google Glass and the Python code running on the Google App Engine (used to store the information about the electrovalves status) work well.
Moreover the LED and the electrovalve have basically the same behavior so, if everything works well with the LEDs, there are all the reasons to believe that everything is going to work well with the electrovalves, too.

The circuit that actually drives the LEDs is very simple, and it is based on a MOS transistor (*BS170*) used as a switch voltage-controlled, as shown in (Fig.6).

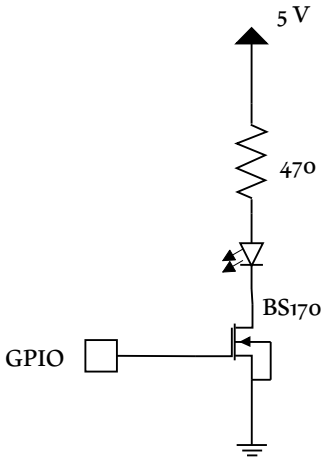


Figure 6: Driver for LED

The (Fig.7) shows the circuit used for this step of testing. As can be seen the number of LEDs used is eight, the same number of electrovalves that can be driven from this system.
In order to test all of them we made 2 different kind of trials:

1. *In order turning on&off*, first all the LEDs are turned on starting from the first one (on the top right corner) to the last one (on the bottom

left corner). Then the LEDs are turned off following the same order. The result of this can bee watched in **this** video.

2. *Out of order turning on&off*, in this trial, like before, all the LEDs start from a condition where all of them are off and then we turned on and off all the LEDs, but in this case following a random order.
The result of this can bee watched in **this** video.

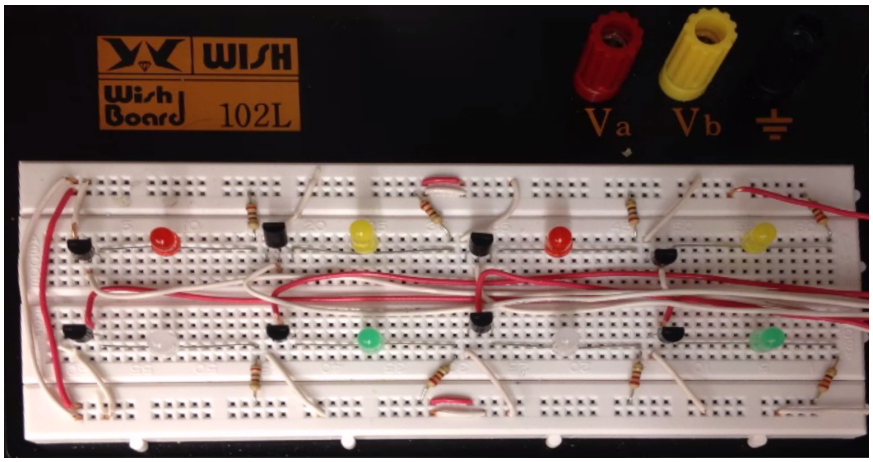


Figure 7: LEDs experiments board

Electrovalves experiments

Breadboard Phase

The (Fig.9) shows from the top view the circuit used during the second phase of experiment, the one where we started using electrovalves in a real microfluidic application.
On the left side of the figure we can see the conditioning circuits for the temperature sensor (on the top) and pH sensor (on the bottom). While, on the other side, we can see the part of circuit in charge to drive the electrovalves. In this last one we are going to focus for now. Each electrovalve is driven by the circuit shown in (Fig.8).

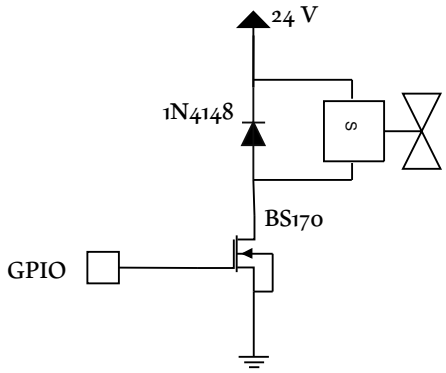


Figure 8: Driver for electrovalves

As you can see, this circuit is pretty close to the one of (Fig.6), indeed the only difference is given by freewheeling diode, mandatory because of inductive behavior of electrovalve's solenoid.

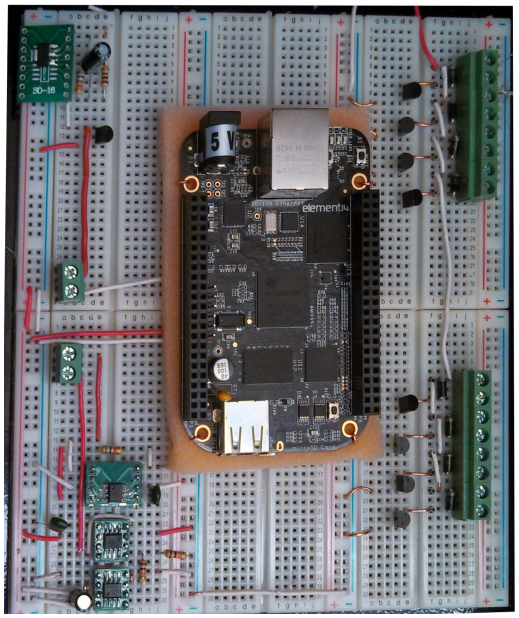


Figure 9: Circuit mounted on breadboard top view

The result of the experiments with electrovalves in a real microfluidic case can bee watched in **this** video.

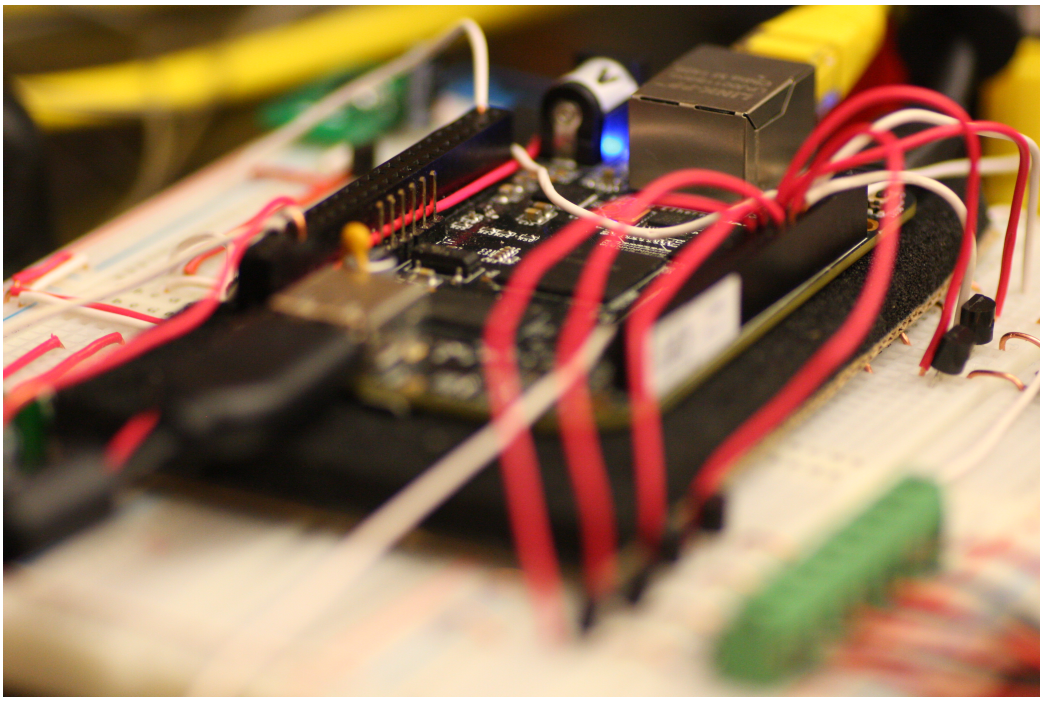
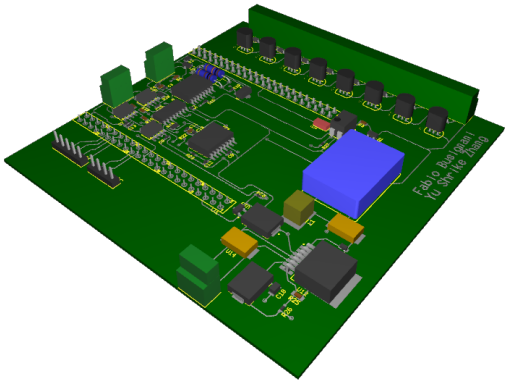


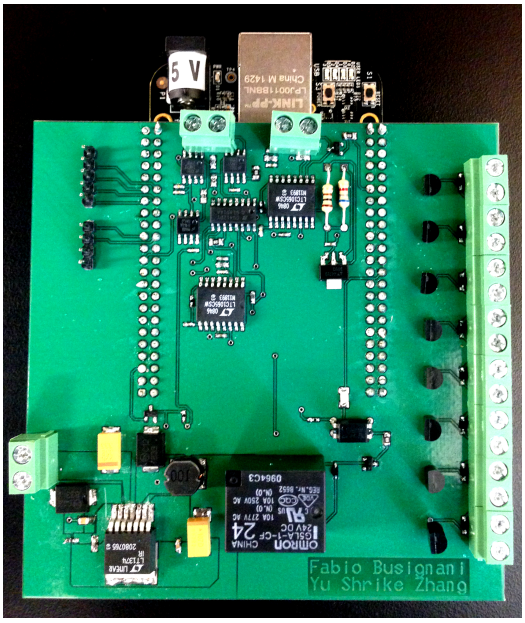
Figure 10: Circuit mounted on breadboard side view

PCB Phase

Finally we replied the last experiment using a PCB (Fig.11), designed for this system.
In (Fig.11a) the 3D model made during the project phase is shown, while (Fig.11b) shows its real realization.
As expected the result of this experiment is the same of the previus step.



(a) PCB 3D Model



(b) PCB Real Result

Figure 11: System's PCB

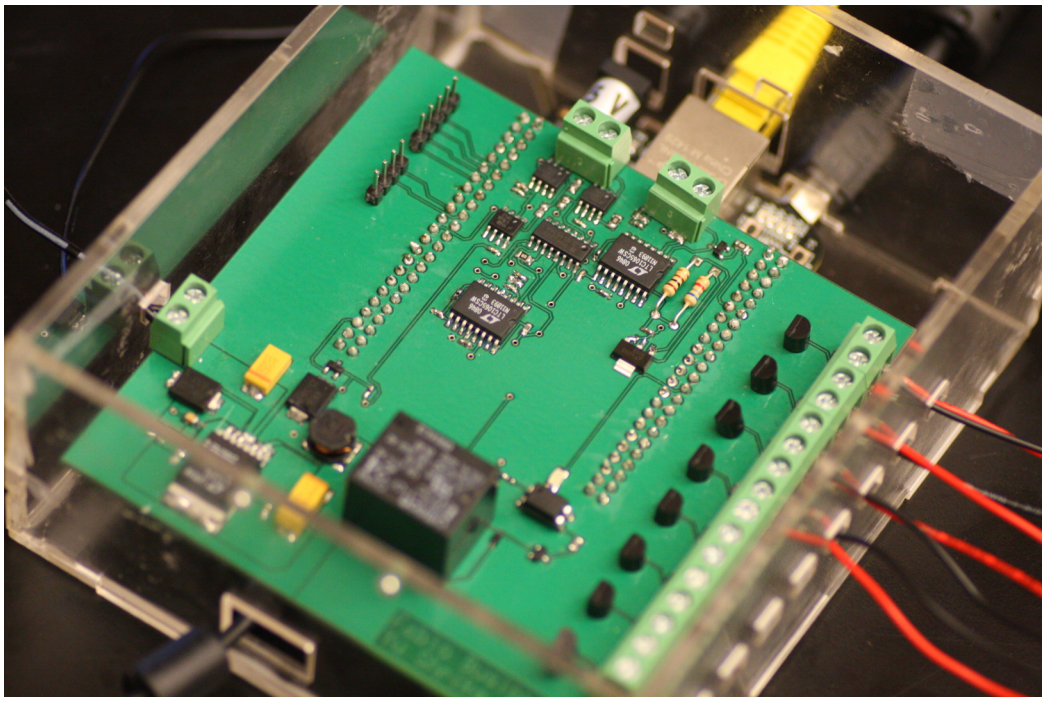


Figure 12: Final system mounted on a PCB

Store Microscope Video

The storing of the microscope video plays an important role of this system. It may be essential to review the recorded video during the experiment and in order to fulfill this aim a *Qt* program has been designed.
We chose *Qt* because in this way the program is available for different operating systems, *Linux*, *Windows*, and *MacOS*.

The program is very easy to use, the user just has to run the executable (Fig.13).

Figure 13: Storing Microscope Video Program's Icon

Once it has been launched, a console is opened (Fig.14). The program checks every 20 seconds if a new video has been uploaded on the server. If so, the new video will be stored inside the computer (directory `C:/Video`) with the current data and hour as name in the following form: $YYYY - MM - DD - HH - mmss$.

As shown in (Fig.14) on the console the user can read all the information about what the program is doing.

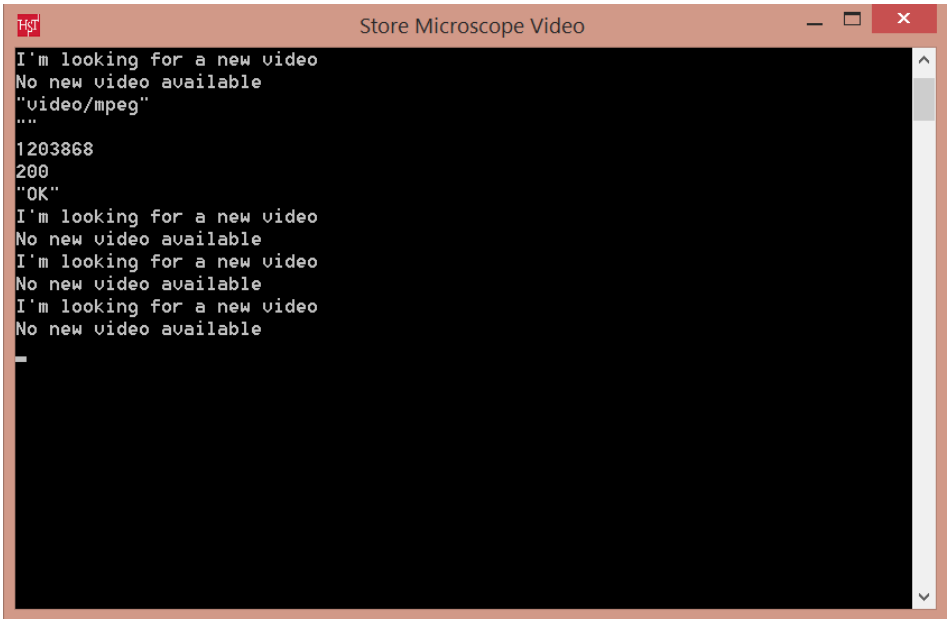


Figure 14: Storing Microscope Video Console