Portable Iso-Dielectrophoresis System

by

Lisa Liu

B. S. Electrical Engineering, Massachusetts Institute of Technology, 2014

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTERS OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2016

Signature of Author____

Lisa Liu
Department of Electrical Engineering and Computer Science
May 19, 2016

Certified by____

Joel Voldman
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by____

Albert R. Meyer
Professor of Electrical Engineering and Computer Science
Chairman, Masters of Engineering Thesis Committee

Portable Iso-Dielectrophoresis System

By
Lisa Liu

Submitted to the Department of Electrical Engineering and Computer Science
May 19th, 2016
in partial fulfillment of the requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Iso-dielectrophoresis (IDS) can be used to characterize and separate cells by their electrical characteristics. An IDS system usually consist of a RF frequency voltage source up to 24 V, syringe pumps, microscope, and computer. Much of this equipment is difficult to transport, making off-site sample collection of time-sensitive samples, such as blood, difficult. The miniaturization of the IDS system would allow measurements to take place at the time of sample collection, improving the confidence in the measurement. This thesis details an attempt to miniaturize each system. The Raspberry Pi microcomputer was used to control and process signals from the hardware. A portable, affordable system for the electrical and optical equipment systems are described. A custom PCB was designed to provide voltage signals up to 3 MHz and 12 Vpp. A microscope using a Raspberry Pi camera and reversed webcam lens achieved 240x magnification with 13 µm resolution. Alternatives to the syringe pump, including air pressure regulated pumps and peristaltic pumps, were considered for the fluidic component.

Thesis Supervisor: Joel Voldman
Title: Professor of Electrical Engineering and Computer Science

Acknowledgements

A lot of work and time was spent going into this thesis project, and I couldn’t have done it without the help and support of many people.

Firstly, I’d like to acknowledge my family for their support, which has allowed me the luxury of pursuing my interests in school and in life in general.

Thanks to my advisor, Joel Voldman, without whose advice and support this project would not be possible. I’ve learned a lot technically and professionally. I know I’m not the only one who sees your passion for teaching.

Thanks also to Haowei Su for teaching me the ropes and helping me with experiments. Your enthusiasm for technology and science is infectious.

One last and tiny thanks to Alisha Schor and Michelle Rybak. You guys have been role models throughout my student life, and I may have skimmed your Masters’ theses as inspiration while writing my own.

Table of Contents

List of Figures

List of Tables

1 Introduction and objective

Cell isolation and characterization has shown promise to advance research in a variety academic and clinical settings. The isolation of a heterogenous sample separates it into its subpopulations grouped by similar characteristics. Further characterization of these subpopulations allow for focused study of each constituent population. Cells can be isolated based on a variety properties including size, deformability, electrical polarizability, surface markers, etc. [1]. A few examples demonstrating the potential of cell isolation and characterization include CTC counting for cancer diagnostics and monitoring [2], studying malaria-infected red blood cells [3], etc.

There has also been a trend towards having the isolation and characterization taking place on microfluidic devices. Microfluidic devices offer several advantages including small sample volumes, faster sample processing, low device cost, and increased portability [4]. These aspects allow the devices to have potential to be widely used in point-of-care (POC) settings, where well-stocked labs or skilled technicians are not available.

Many fluid-based microfluidic devices require extra machinery in order to be operated. In particular, typical microfluidic devices in our lab might require the use of a computer, microscope, syringe pump, function generator, and RF amplifier. All of these are larger than the device, and most are heavier. This reduces the portability of the system, and therefore the potential impact of the device. For time-sensitive samples, like blood, it could be desirable to transport the measuring equipment to the site of sample measurement, rather than transport the sample to the site of measurement, in order to reduce any loss of sample integrity.

This thesis describes the miniaturization of the support equipment for the IDS microfluidic device, with a focus on increased portability and ease of use. Chapter 1 provides an introduction to the theoretical basis of the device and the context for the system's current setup. Chapter 2 gives a system level overview of the proposed miniaturization of the system. Chapter 3 describes the electrical portion of the project, where the most contributions were made. Chapter 4 describes marginal contributions in the optical and fluidic system. Lastly, Chapter 5 concludes this thesis, discussing the contributions made as well as potential future work. Additional appendices show code written for the GUIs and for data analysis.

1.1 Theoretical basis of di-electrophoresis

The IDS system relies on di-electrophoresis (DEP) to separate particles. Cells experience a DEP force based on their electrical characteristics relative to the medium they're suspended in. Therefore, cells with different electrical properties can be pushed away from each other or can be pushed with different magnitudes, allowing them to become physically separated. The equation governing the force that a particle under DEP experiences is given by

$$F_{DEP} = 2\pi r^3 \epsilon_m Re\{f_{cm}\} \nabla \boldsymbol{E} \cdot \boldsymbol{E} \quad \text{(Equation 1)}$$

where r is the particle's radius, ϵ_m represents the permittivity of the media surrounding the particle, $\boldsymbol{E}$ is the electrical field that the particle experiences. f_{cm}, the Clasius-Mossitti factor, represents the difference in electrical properties between the particle and the media, and is given by

$$f_{cm} = \frac{\epsilon_p^* - \epsilon_m^*}{\epsilon_p^* + 2\epsilon_m^*} \quad \text{(Equation 2)}$$

Lastly, ϵ^* is a value of complex permittivity, which is given by

$$\epsilon^* = \epsilon + \frac{i\sigma}{\omega} \text{(Equation 3)}$$

where ϵ is the real permittivity, σ is the conductance, and ω is the frequency of the field applied.

These equations predict that cells will experience larger forces, and therefore larger movement, when they are larger, experience high gradients of electric field, or are very electrically different from the media. Furthermore, if any of those parameters are 0, no movement would be expected. $\epsilon^* = \epsilon + \frac{i\sigma}{\omega}$ (Equation 3 also implies a frequency dependent behavior.

IDS relies on a location where $Re\{f_{cm}\}$ becomes 0. In general with IDS, the particles are exposed to media of various known conductivities. When the particles experience no DEP force, inferences about the particles can be made based off the surrounding media's conductivity [5] [6].

For the particular IDS system used in this thesis, the media is introduced as a spatial gradient. Figure 1(a) shows the active sorting region on an IDS device. The cells suspended in high-conductivity buffer are introduced from the top-left and flow towards the bottom of the device. Low conductivity buffer is introduced from the top-right side of the channel, establishing a gradient from left to right. The electrodes are at a slant, pushing the cells gently from left to right when the proper electric field is applied. Under ideal operating conditions, the particles are pushed to the right, and the real part of the Clasius-Mossitti factor gets closer and closer to 0 until they are no longer pushed. This point is called the iso-dielectric point (IDP). From this point on, the cells only flow downward. The position of the cell is measured at the bottom of the device to get the IDP. In reality, the IDP is not exactly the point where the DEP force is 0, but rather the point where DEP force balances with drag force.

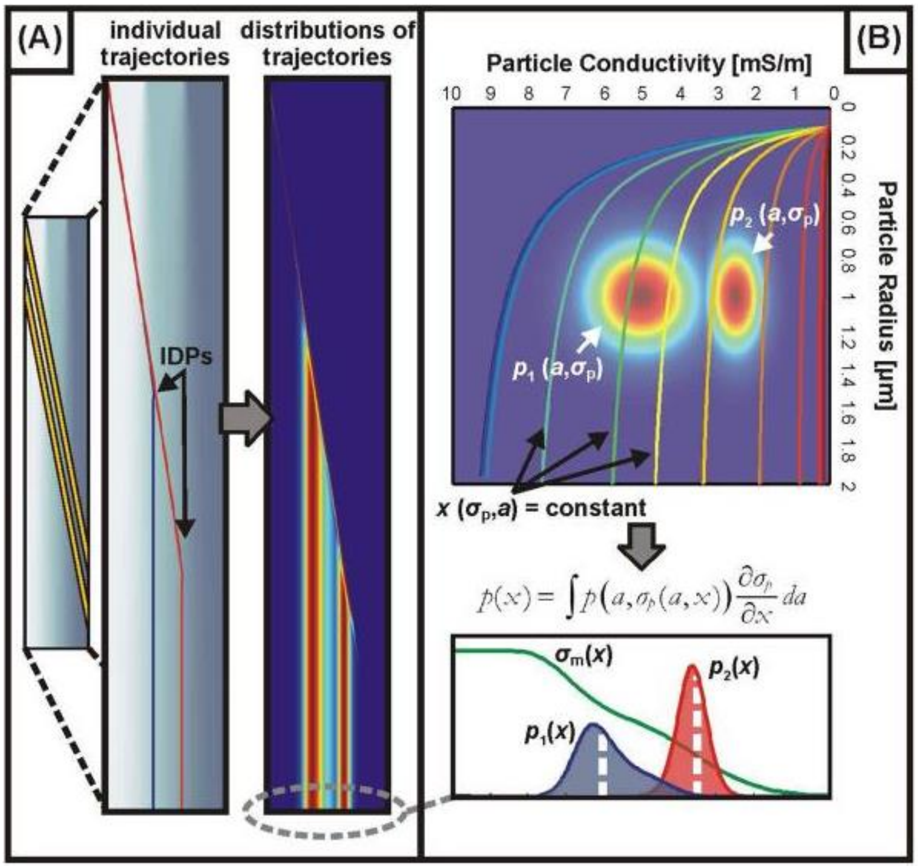


Figure 1 Modeling results of the IDS system. (a) Shows the media gradient across a channel. Individual trajectory show how two cells of the different electrical properties. At right, a heat map of the distribution over a population of cells is shown. (b) The relation between physical properties and electrical properties and their predicted trajectory is mapped. Image from [5].

At the bottom of Figure 1(b), the graph shows a histogram of the cells' IDP as the media's conductivity decreases. Inevitably, there is some overlap in IDPs between the two populations. The success of the separation can depend on all the parameters in Equations 1-3 as well as the fluid drag. Therefore good control of these factors are desired – particularly flow rate, magnitude and frequency of the electric field. Optical resolution to determine the particle's IDP is also desired.

1.2 Microfluidic device miniaturization

There are two main reasons why miniaturization of microfluidic devices is of use. In order to truly be a point-of-care device, all the equipment needed for measurement must be readily portable. As experiments move toward human trials or towards time-sensitive samples, this point-of-care functionality can impact the reliability of the results. The other reason is often miniaturized systems can be made cheaply, or can be shared more widely. This is useful in learning environments, to provide scientific tools to a wider audience.

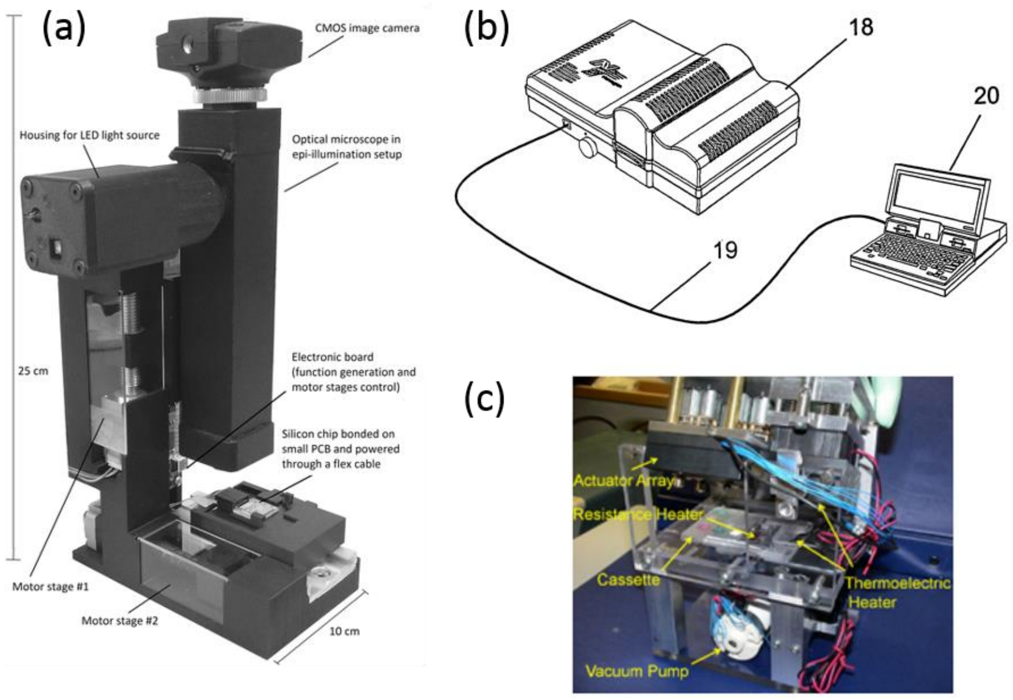

(a)
CMOS image camera
Housing for LED light source
Optical microscope in epi-illumination setup
25 cm
Electronic board (function generation and motor stages control)
Silicon chip bonded on small PCB and powered through a flex cable
Motor stage #1
Motor stage #2
10 cm
(b)
18
20
19
(c)
Actuator Array
Resistance Heater
Cassette
Thermoelectric Heater
Vacuum Pump

Figure 2 Examples of integrated portable systems. (a) This portable DEP system combines the microscope and function generator into one system 25 cm x 10 cm x 10 cm. Image taken from [7] (b) A patented integrated system to separate bacterial cells from cancer cells. Image taken from [8]. (c) A portable system to process saliva with a disposable cassette. Image taken from [9].

There have been some efforts to create integrated, portable systems, ready for point-of-care use. Some of them approach the problem by miniaturizing different components of the system, and housing them in a tighter space. Figure 2 shows three such examples. Since all the elements are already integrated, the setup time is minimal for the systems.

Some other approaches to miniaturization have focused on miniaturizing only one part of the system, either because it is easier to focus on one element, or so that it may be applied to a wide variety of microfluidic devices. One example is a miniaturized valve-less peristaltic pump, designed at the Biodynamic Optical Imaging Center at Peking University [10]. It used a PDMS channel with a stepper motor in order to accomplish flow rates on the order of 8 nL/min. In this example, although the size of the pump is reduced, they still use a function generator to drive the stepper motor. However, the stepper-motor driving equipment is also ready portable. Another example is the development of a portable microscope by Andrew Miller [11], which replicated all the elements of a compound microscope, and added battery power to make it more portable.

Some approaches attempt to create a novel architecture to solve the same problem. For example, Foldscope used origami inspired design in order to achieve cheap, mass-produce-able, and small microscopes [12]. Other approaches are inspired from old architectures, but simply try to recreate using readily accessible or off-the-shelf components, like webcams or smart phones. For example, there are a variety of smart-phone based systems that are being used to monitor cardiovascular diseases [13]. Some require the use of external hardware, but some use just the hardware integrated onto the smartphone.

Taking inspiration from these studies, we decided that miniaturizing the system would could be useful for future studies done by our lab, where some samples may be time-sensitive and need to be measured on-site.

1.3 Current system setup

The system can be divided into 4 subsystems: electrical, fluidic, optical, controls. The electrical system generates the electric field gradient used to drive the cells. The fluidic subsystem introduces the cells into the system and allows to flow towards the device egress. The optical system is stationed above the fluidic device to capture images for later analysis. Figure 3 shows how these systems are interconnected.

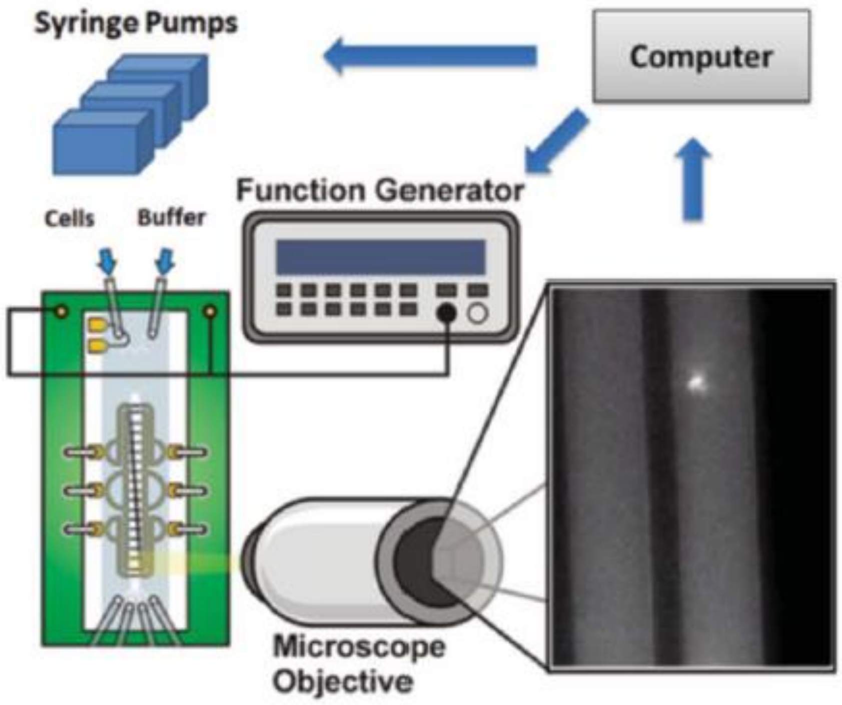


Figure 3 The current IDS system. A computer is used to control the function generator and syringe pumps. A microscope looks at the device outlet and the images are sent to the computer for processing. Image taken from [6]

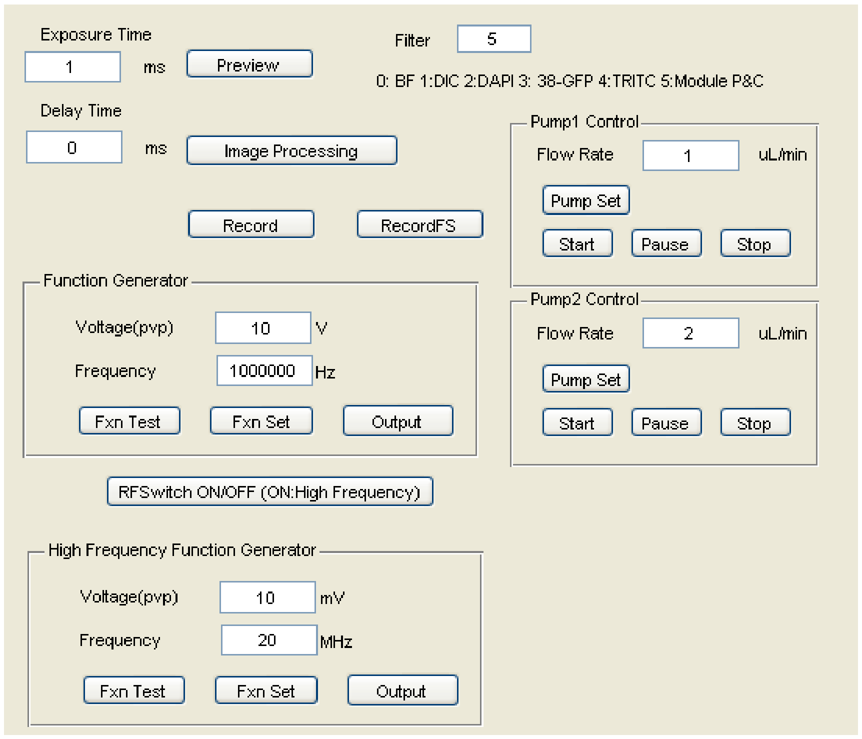
The current system uses a computer-controlled function generator (Agilent 33 250A) to generate the oscillatory electric field. The frequency response of the electrodes was characterized by an impedance analyzer (Agilent 4294A). A set of syringe pumps (Chemyx Fusion 200) and syringes (Hamilton 1 mL Gastight Syringes) deliver media and samples to the device via a 0.004 I.D PEEK *TM* tubing (IDEX Health and Science). Imaging is performed on a fully automated upright microscope (Zeiss Axio Imager) with stage control (MAC 5000). Images are acquired by a PCO Sensicam QE camera. The camera and microscope are controlled via MATLAB.

Most recently, the IDS system has been used in mouse and human studies for neutrophil activation as a potential prognostic or diagnostic tool for monitoring sepsis [7].

Each of these IDS’s subsystems has the potential to be miniaturized for enhanced portability and ease-of-use. The current system is a benchtop setup, with most of the equipment fairly bulky and difficult to transport. A system that can be packed up in a suitcase and requires minimal setup is desirable. In Chapter 2 through Chapter 4, a target specification in terms of function, size, and cost is laid out. The design process and results are described.

2 Controls

The electronics, optics, and fluidics are controlled from a desktop computer in the original system. A MATLAB GUI, shown in Figure 4, was developed to allow the user to set parameters for each system and to receive images.


Exposure Time 1 ms

Delay Time 0 ms

Filter 5

0: BF 1:DIC 2:DAPI 3: 38-GFP 4:TRITC 5:Module P&C

Function Generator

Voltage(pvp) 10 V

Frequency 1000000 Hz

High Frequency Function Generator

Voltage(pvp) 10 mV

Frequency 20 MHz

Pump1 Control

Flow Rate 1 uL/min

Pump2 Control

Flow Rate 2 uL/min

Figure 4 Control panel of MATLAB GUI

However the size and weight of a desktop computer makes it difficult to transport the system. The desired factors of a replacement system include:

- Portability – easy to carry
- Ability to drive external hardware
- Easy interaction and setup
- Image processing – potentially for on-board image analysis

A few alternatives to replacing the desktop were considered:

- Installing the GUI and software at the other locations
- Laptop
- Microcontroller kit (Arduino)
- Microcomputer kit (Raspberry Pi 2 vs Beaglebone Black Rev C)

The first alternative, installing the software on computers at other locations, was discarded for two reasons. Firstly, every new location would need a new installation, which adds to the setup time. It is easier the user only has to setup once and can bring it everywhere. Also, since this portability project will use new hardware, the software would be rewritten anyway.

The microcontroller and microcomputer were preferred over laptops because of their smaller footprint and the direct availability of GPIO controls. Between these two options, microcomputers were favored for their higher processing power and computational flexibility. In order to read data from the Arduino, it has to be connected to a computer. However, data can be read directly from the microcomputer. Or it can be stored on a flash drive or on the on-board SD card before being transferred to a computer.

A specification comparison between Raspberry Pi B+ and the Beaglebone Black Rev C – two popular microcomputers at the beginning of this project – is given in Table 1. While both tools were sufficient in regards to memory and GPIOs, ultimately, the Raspberry Pi was chosen because of its camera supported faster framerates and its overall lower cost. Also, The Raspberry Pi has a larger community support. Although it's difficult to quantify size of community, more Raspberry Pi's have been sold than Beaglebones [8], and, as of May 2016, the Adafruit forum has 108 pages related to Raspberry Pi, as opposed to 16 pages for Beaglebones'.

	Raspberry Pi	**Beaglebone**
GPIO	40	92
RAM	1 Gb	512 MB DDR3
Ports	4 USB, 10/100 Ethernet	1 USB, 10/100 Ethernet
Camera	CSI connected camera board ($20)	USB connected camera board () or Camera Cape ($50)
Cost	$40	$55

Table 1 Comparison of Raspberry Pi 2 B+ and BeagleBone Black Rev C features.

2.1 Setup and external hardware

While the Raspberry Pi can analyze data and control hardware, the user still needs to interact with the Pi. In normal operation, the Raspberry Pi is connected to a monitor with an HDMI connection, a keyboard, and mouse. The SD card is loaded with an operating system, and the system can be run from command line or from a GUI, much like a desktop computer. The addition of a touchscreen (Adafruit PiTFT 3.5”) allowed the Raspberry Pi to operate independently from other hardware, making it more portable and easier to set up. This touchscreen would replace the monitor and mouse functions.

Since the touchscreen covered up many of the GPIO pins, making it difficult to access them, a ribbon cable was purchased for easier access to the GPIO pins. The last hardware addition was the Raspberry Pi Camera Board (v1) which allows pictures and videos to be captured by the Raspberry Pi.

The Raspberry Pi was set up with the Jessie version of the Raspbian operating system. In order to set up the Raspberry Pi for the touchscreen, a customized kernel from Adafruit was downloaded and installed [8].

By default, the Raspberry Pi boots to command line. In order to open the GUI, the command “startx” must be entered. A new file was added and the configuration settings were changed according to Adafruit’s recommendations [9]. Lastly, another tip was followed for easier gesture control of the touchscreen, which opens files with a single click instead of a double-click [10].

Once the Raspberry Pi was set up, code was developed to take user input from a GUI on the touchscreen. The input was used to drive the hardware and the subsystems of the portable IDS system.

3 Electrical System – Sine wave generation

A sinusoidal voltage is a key input to di-electrophoretic experiments. Cells exhibit frequency dependent behavior, and the ability to apply a range of frequencies allows the selection of an ideal frequency for a particular experiment. It is also important that the signal can provide enough power. The ideal voltage would be 24 Vpp across a 50 ohm load in a range of frequencies from 0.1 to 10 MHz.

3.1 Current system and specifications relevant for the IDS

The system uses a function generator (Agilent 33220A) to generate the signal and an RF Power Amplifier (Mini-Circuits TVA-R5-13) to provide the power. Each of these are about the size of a large shoebox, weigh about 8 pounds, and are rather expensive pieces of equipment – a new Agilent 33220A costs $2400 and the RF amplifier costs about $1500. All of these factors make a portable system difficult.

One reason why they may be this expensive is that they can perform scientifically accurate experiments across a range of functions, frequencies, and power requirements. An IDS experiment only needs a fraction of those capabilities. For example, the Agilent 33220A can output not only sine waves, but also square and triangle waves, all with DC offsets as well. The TVA-R5-13 can output 3.2 W in a range of 0.5 to 1000 MHz frequencies. It is worth thinking that a system can be developed specifically for IDS requirements at a lower price.

3.2 Previous attempts needed specifications

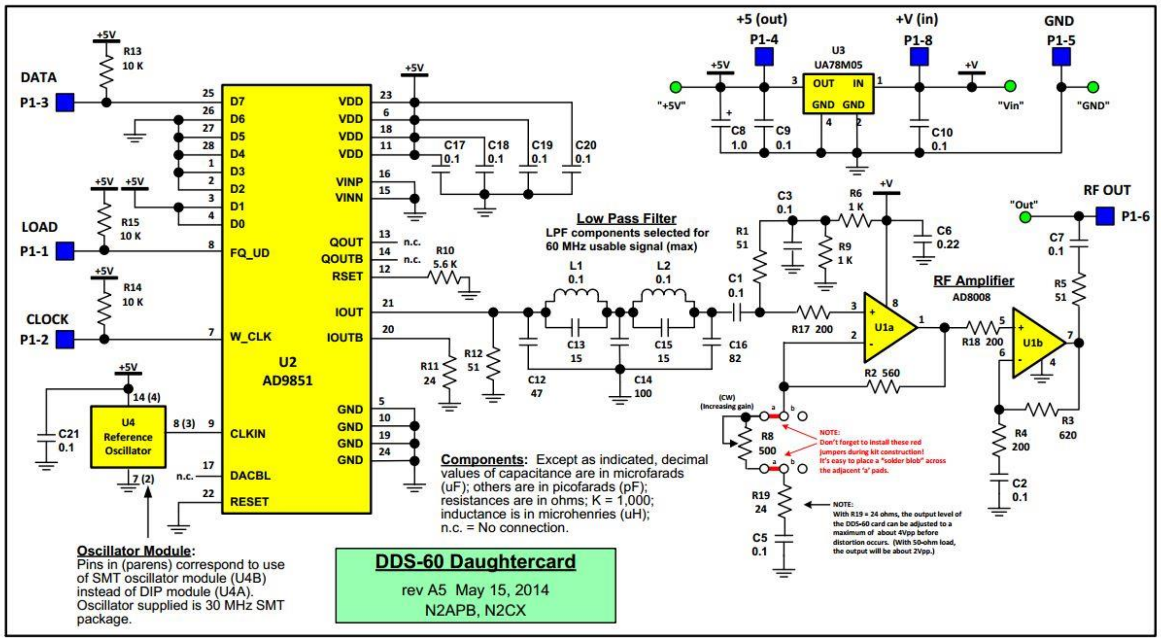
There have been several attempts to build a dedicated electrical board for IDS experiments which replaces the signal generation and amplification. Some of these design decisions in these earlier designs informed the eventually developed system.

3.2.1 DDS-60 Daughterboard

Earlier development focused on the integration of the DDS-60 daughterboard from Midnight Design Solutions. The DDS-60 provides a low-cost and modular way of adding a 1-60 MHz variable frequency oscillator (VFO) into a project. It provides up to 4 Vpp to a 50 ohm load.

The DDS-60 daughtercard consists of the following components:

- direct digital synthesizer (DDS), AD9851 – an IC that converts a digital signal into a sine wave. The waveform can be internally converted to a square wave.
- a 5V regulator, UA78M05 – to supply constant voltage to the ICs on board
- a reference oscillator, SG-615PTJC - the DDS uses this as a reference to calculate the frequency of the output signal
- a band pass filter – to reduce any DC offsets and noise. Although the schematic in Figure 5 lists it as a low, pass filter, a simulation shows a cut-off frequency at approximately 10 kHz. The transfer function of this filter is shown in Figure 6, and is roughly unity for the frequencies of interest.
- 2 RF operational amplifiers, AD8008 - the DDS60 itself only goes up to 1 Vpp, so the RF amplifiers are used to boost the signal up to 4 Vpp.



NOTE:
Don't forget to install these red jumpers during kit construction!
It's easy to place a "solder blob" across the adjacent 'a' pads.

NOTE:
With R19 = 24 ohms, the output level of the DDS-60 card can be adjusted to a maximum of about 4Vpp before distortion occurs. (With 50-ohm load, the output will be about 2Vpp.)

Components: Except as indicated, decimal values of capacitance are in microfarads (uF); others are in picofarads (pF); resistances are in ohms; K = 1,000; inductance is in microhenries (uH); n.c. = No connection.

Oscillator Module:
Pins in (parens) correspond to use of SMT oscillator module (U4B) instead of DIP module (U4A). Oscillator supplied is 30 MHz SMT package.

Figure 5 Schematic of DDS60 daughtercard.

The daughtercard has 3 digital input pins – data, load, and clock. Clock is used to time when to load a new bit of data. Data sets the digital signal. And load indicates when a command is ready to be issued to the on-board DDS. There are two output pins – a five volt output from the voltage regulator and the sinusoid output. Lastly, there are ground and power connection pins and an unused pin.

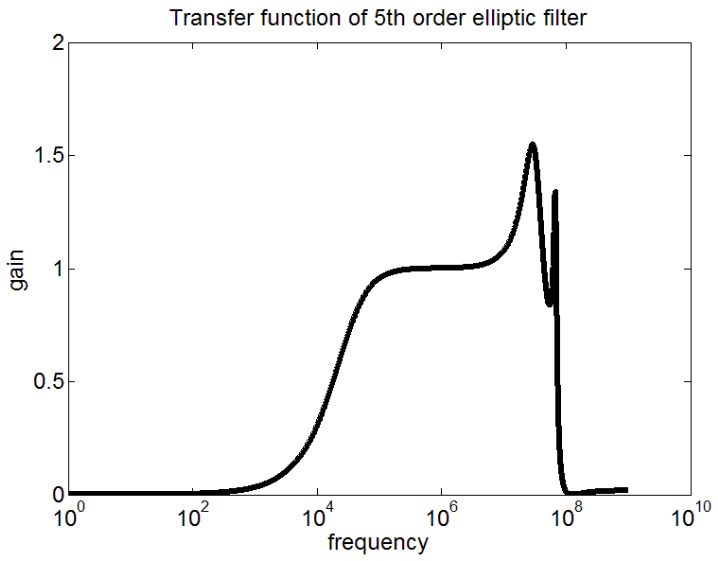


Figure 6. MATLAB simulation of DDS-60's on board filter's transfer function.

This board successfully generated sinusoids in the desired frequency range, and so the design of the final electrical system used a similar architecture. However, the daughtercard on its own could neither provide enough power nor enough voltage. External circuitry was required, which is described below.

3.2.2 Controlled amplification (and digital resistor)

The DDS-60 provided controlled frequency, but did not meet the power requirements. Therefore, previous implementations have used an external PCB to amplify the signal.

Many of the implementations included a THS3091 operational amplifier, which was satisfactory for use in external amplification. It could source up to 350 mA and 26.4 V_{pp}, and could amplify signals up to 180 MHz (at a gain of 10). In addition to power amplification, it was realized that there was a need for controlled amplification, since the voltage amplitude was not constant across all frequencies. One method implemented was to use a digital potentiometer as part of an inverting operational amplifier configuration. Since the gain of the amplifier circuit was a ratio of resistors, one resistor was kept constant, while the digital potentiometer’s resistance was set by a digital signal, as needed.

3.3 New circuit layout

Previous solutions had some disadvantages. First was a voltage offset, which is undesired because any DC voltage can harm the electrodes and heat the buffer solution. This is bad for the experiment either because it can kill the cells or affect the properties of the cells. Also, the daughterboard increases the space requirement by requiring two circuit boards, both of which need power sources.

Furthermore, previous solutions assumed the use of an Arduino for control signals. The previous PCB took the form of an Arduino shield, making the connection of wires straightforward and user-friendly. With our decision to use the Raspberry Pi for potentially wider range of control and on-board analysis, a completely new layout was needed.

Therefore a design was proposed to create one PCB to control all aspects of function generation. Similar to previous designs, the PCB was separated into stages of (1) signal generation, (2) amplification control, (3) power amplification, and (4) filtering.

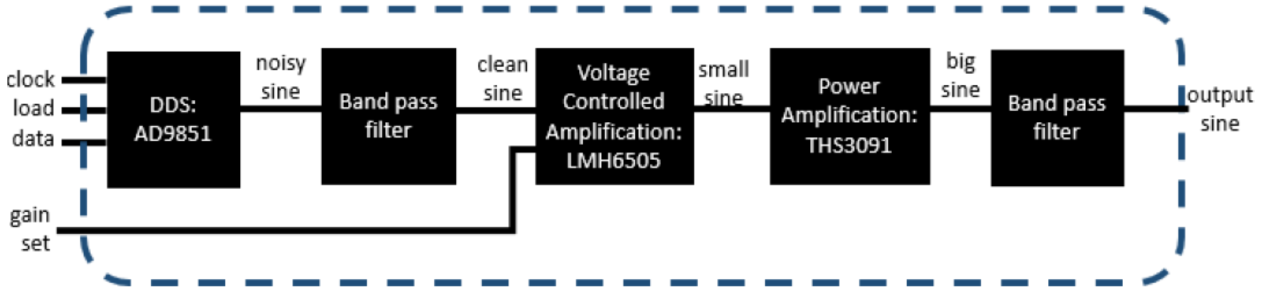


Figure 7 Architecture for function generation.

As in previous designs, we used the AD9851 since the control mechanism was straightforward and it gave clean results. The filter used on the output was the same as the one on the DDS-60 daughterboard.

The THS3091 was also kept for power amplification. It could support the power output across a wide range of frequencies. The THS3091 required +/- 12 V power supply. This was supplied by using a barrel jack to connect to an external 12 V supply. The -12 V supply was created using the

MAX765, a DC-DC inverter, capable of delivering 250 mA or 1.5 W. Its supply voltage ranged from 3-16 V, and so was connected to the Raspberry Pi 5V power lines.

An optional first-order RC bandpass filter, shown in Figure 8, was included at the end, in case the output signal from the amplification step was too noisy. It is optional in the sense that a wire can be connected directly to the output of the board to bypass this filter. Values for the resistors and capacitors were chosen to cutoff frequencies below and above 0.1 MHz and 10 MHz, respectively.

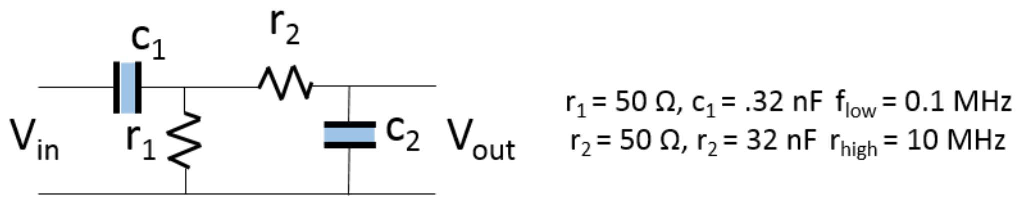

c_1
r_2
V_{in}
r_1
c_2
V_{out}
$r_1 = 50\ \Omega$, $c_1 = .32$ nF $f_{low} = 0.1$ MHz
$r_2 = 50\ \Omega$, $r_2 = 32$ nF $r_{high} = 10$ MHz

Figure 8 1st order RC bandpass filter.

One difference was the method used for amplification control. Instead of using a digital potentiometer, a voltage controlled amplifier (VCA) was used. A voltage controlled amplifier uses an analog input to determine the gain. The component LMH6505 was chosen, whose maximum gain is set by the ratio of two resistors.

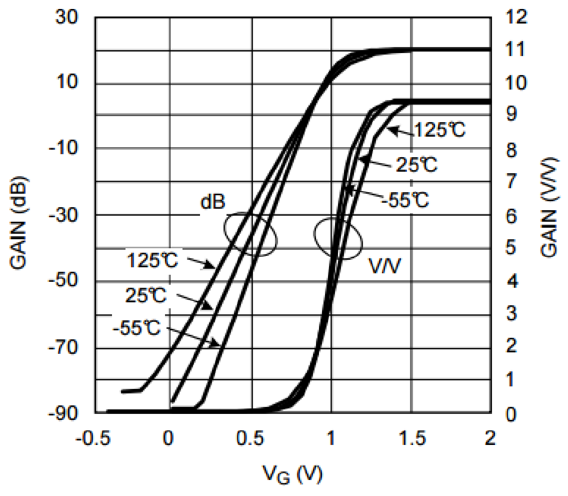


Figure 9 Voltage-Gain relation for the LMH6505. [11]

This required a new input to the board – the gain set input. The gain set signal provided a PWM signal, whose duty cycle was related to the desired input voltage. An on board first-order RC low pass filter then converted the PWM signal to a DC voltage, which would be used to determine the gain. The cutoff frequency of this RC filter was set to be 1.8 orders of magnitude smaller than the frequency of the PWM signal, which was 1 kHz. The resistor was R = 100 KΩ, and C = 0.1 uF. Through-hole components were used for this filter in case a different PWM frequency signal was later desired.

3.3.1 Board layout

3 attempts were made to design the schematic and build the board that represented the system architecture given by Figure 7. In order to achieve a compact size, a double-sided board was used. The bottom side was primarily a ground plane, but also included some capacitors, an inductor, and the reference oscillator. These items were placed on the back such that they followed recommended guidelines as given by the relevant IC datasheets.

The active components followed the structure of the DDS60 daughtercard. Details of these components are given in Table 2.

Part Number	Function	Package
AD9851	DDS – signal generation	Ultrasmall 28-pin SSOP
n/a	5th order elliptic band-pass filter (same as the one given in Figure 5)	0402 capacitors and inductors
LM6505	Voltage-controlled amplification	8-pin VSSOP
THS3091	Power amplification. Non-inverting amplifier with gain of 10 (R_f = 866 Ω, R_g = 95.3 Ω)	8-pin SOIC
n/a	1st order RC band pass filter	0603 capacitors and resistors
MAX765	-12 V inverting switching regulator	8-pin SOIC
LTC1983-5	-5 V inverting charge DC/DC converter	6-pin ThinSOT
SE2330CT- ND	30 MHz reference oscillator	SOJ-4

Table 2 Active part list for the board layout.

The schematic and layout of the board were designed in KiCad, an open source electric design automation tool. Some of the packages were not in the KiCad library, so their schematic had to be manually drawn in the software. This was relevant for the -12 V regulator, the oscillator, and the voltage controlled amplifier. Once the PCB was designed, the layout files were sent to Advanced Circuits for printing. And circuit components were ordered from Digikey. The boards were populated by hand. A bill of materials is provided in Appendix A, and the layout and schematic of the final version of the board is provided in Appendix B.

In version 1, a few components missed necessary connections to ground and a resistor was missing as well. Once soldered on externally, the board worked, but was very fragile. The connections between ground and resistors broke easily upon handling. In version 2 of the board, some bypass capacitors were corrected to the right signal, but the wrong pin, making the DDS unresponsive to input signals. This emphasizes the need for proper layout. In the third rendition of the board, all components worked, and it became a matter of tuning filters by proper selection of resistors and capacitors. All boards are approximately 3.5 cm x 6.4 cm, though the last one is larger than the first two, since it adds an additional optional filtering stage.

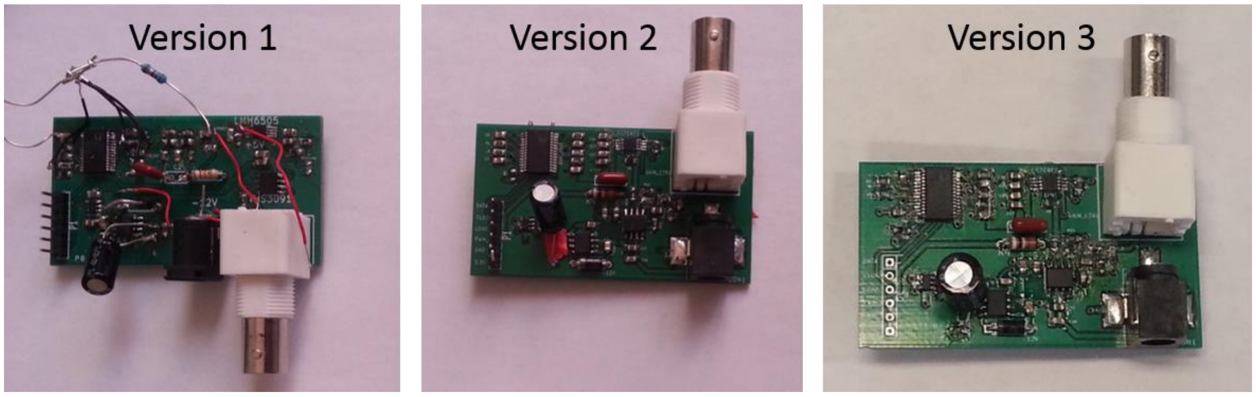


Figure 10 Three versions of the function generator board.

The board is powered from a combination of 5V pin from the Raspberry Pi power signals, and a 12 volt input at the barrel jack. The barrel jack can get its 12 volts from either a battery or wall-based power supply.

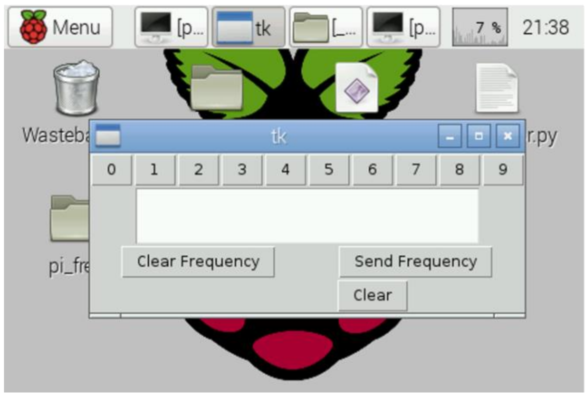
The output is taken at the BNC connector. There are 6 header pins for the input signals. The few connections make the board easy to use and hook up.

3.4 GUI and Portability

Another goal of the portable IDS was to make it easier to use. The control signals use a Raspberry Pi in order to control the signals. However, the Raspberry Pi has no display or non-GPIO inputs, so a system to connect to it must be developed. The options included (1) wifi module + personal computer (2) touchscreen display (3) GPIO input.

A wifi module relied on a working internet connection, which is not guaranteed or may be complicated to set up at other locations. It also requires a computer to control the Raspberry Pi from. The GPIO input would require external wiring, and would be hard to scale up for many features and to precisely select the amplitudes and frequencies desired. The touchscreen would be connected directly to Pi and could be flexible enough to control a variety of features.

The Adafruit PiTFT 3.5” (Adafruit 2441) was selected, and an external touchscreen pen was also purchased for reliable input, although a finger also works. Some modifications to the Raspberry Pi operating system was necessary for convenience.



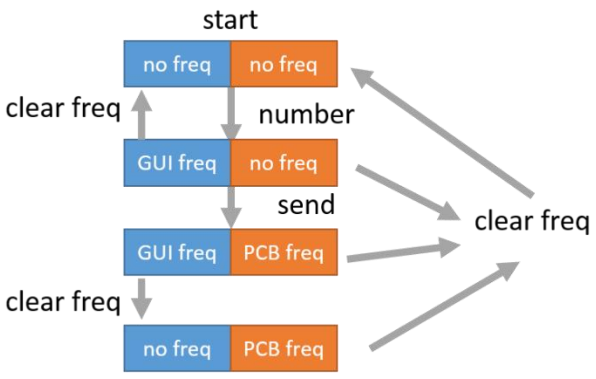


Figure 11 Python GUI used to input frequency and send command to PCB. The right shows a block diagram showing the effects of the buttons. The blue boxes show the state of the GUI, and the orange boxes show the state of the function generator PCB.

To avoid the requirement for an external keyboard, the GUI was designed to allow touchscreen input of numbers. It also included a clear button, clear frequency button, and a send button. The user only needs to send the frequency, and the python code that generates the control signals is run. The ideal user process would be to enter the program, with no frequency stored in the GUI or on the board. The user would press number buttons to achieve the frequency desired. Once the user presses “send”, then the board will generate the desired frequency. The “clear frequency” button allows the user to reset the GUI frequency, in case an error was made in typing, or a new frequency is desired. A “clear” button is also implemented, which stops the signal with one tap. As a result, the PCB board will output a 0 V signal.

3.5 Signal output

The final setup for the electrical system is shown below in Figure 13 and set up according to the following parameters:

- The 12 V and 5 V power supply were powered from a wall outlet.
- Output pins of PCB board were connected to an oscilloscope (Agilent InfiniiVision MSO-X 3054 A).
- A 6” GPIO ribbon cable was used to make the GPIO connections more accessible.
- Last stage of filtering was bypassed, to avoid signal attenuation.

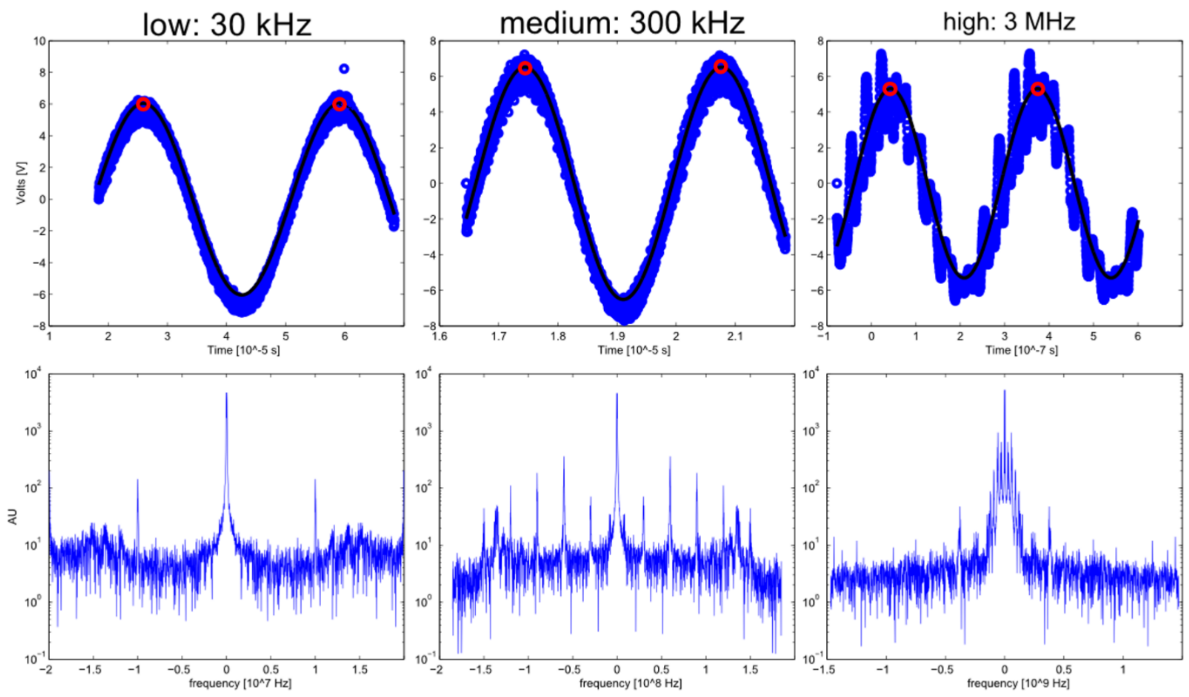


Figure 12. Comparison of three relatively low, medium, and high frequencies in the time and frequency domain. The red dots identify peaks of the signal in the time domain.

The digital oscilloscope's max sampling frequency is 4x10^9 samples/second. A time range was picked was such that 1.5 periods were visible. The signal was acquired in the normal acquisition mode. A MATLAB script was written to extract data quickly and without bias. The signals were analyzed for the following metrics, as summarized in Figure 15.

- Dominant frequency
- Mean voltage
- Signal to noise ratio (SNR)
- Peak-to-peak voltage (V_{pp})
- VCA is set to produce maximum gain (10x). The duty cycle is 100%.

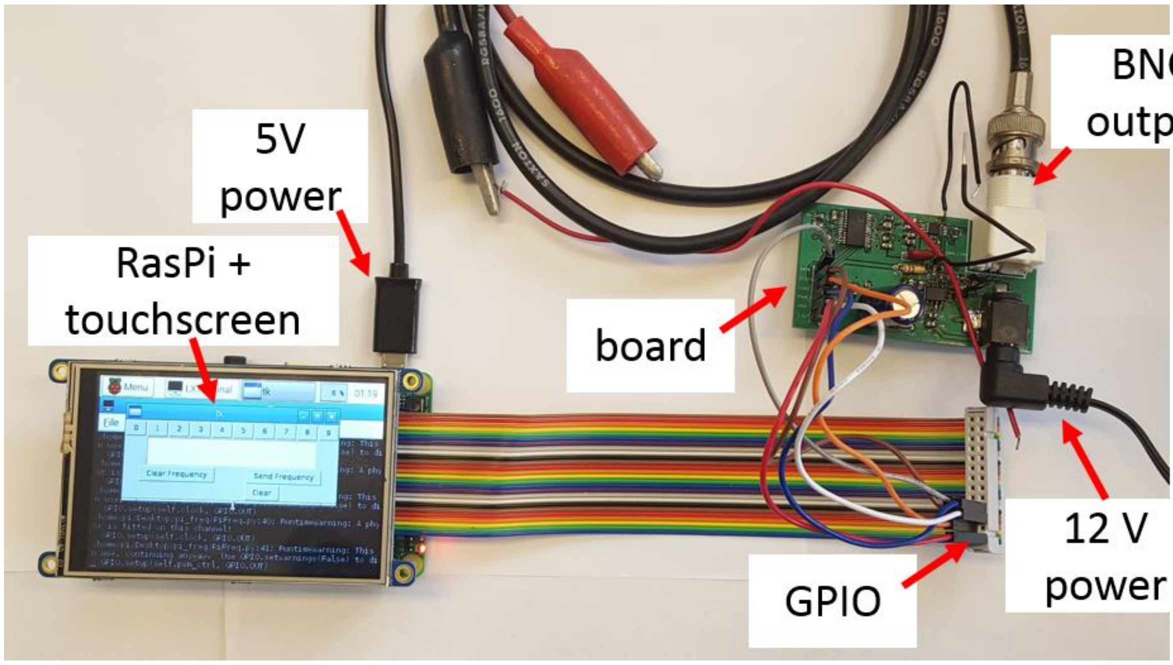


Figure 13. The complete setup for function generator. The Raspberry Pi can be run off a 5V microUSB power source, though the PCB needs to be connected to an external 12 V supply. BNC cables are used to read the output.

A frequency domain calculation of SNR and dominant frequency was attempted, but had low resolution due to the resolution of the oscilloscope. The length of the signals was between 1360-2000 samples, and they all were taken at 4 Gigasamples/second. Using the following relation between frequency and sampling rate, the frequency resolution can be calculated.

$$\Omega = 2\pi \frac{k}{N} = 2\pi \frac{f}{f_s} \text{ (Equation 4)}$$

This resulted in a resolution in the frequency domain of 2 MHz, which is larger than some of the frequencies of interest. Therefore, all values were calculated in the time domain.

The peak to peak voltage was found by filtering the signal, and subtracting the minimum voltage from the maximum voltage. A 6-point moving average filter was applied in order reduce the effect of any spurious samples. The filter had negligible impact on the signal. As seen in Figure 14, the transfer function attenuates to 1/e when k = 234, which corresponds to a frequency of 500 MHz. The dominant frequency was calculated by using the same filtered signal and finding the distance between peaks. The MATLAB Function findpeaks was used, with the parameter specifying the minimum distance between peaks to be 0.9 of the expected period. As seen by the red dots in Figure 12, this method was able to accurately find the peaks over a variety of frequencies.

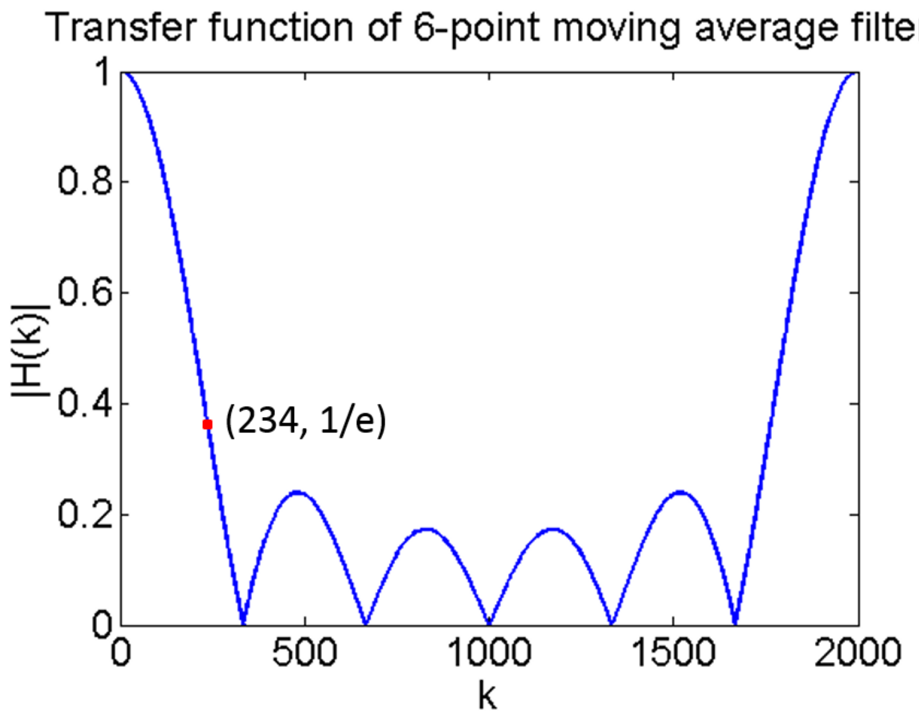


Figure 14 Transfer function for a 6-point moving average filter for a signal with 2000 samples and a sampling frequency of 4 Gigasamples/second. The filter decays to 1/e when k =234.

Likewise, the SNR was calculated in the time domain. First, a pure signal was found by fitting the data to a sinusoid of the form $A\sin(2\pi Bt + C)$. Values of A, B, and C were found by using MATLAB's lsqcurvefit function, with the following initial values:

- A = Vpp/2
- B = the calculated frequency
- π/3

The noise was calculated by subtracting this the pure signal from the data. The energy of the signal and the noise was given by $E = \sum_{i=0}^{N} x_i^2$ (Equation 5. The SNR is given in dB and calculated using $SNR = 10\log_{10}\left(\frac{E_{signal}}{E_{noise}}\right)$ (Equation 6.

$$E = \sum_{i=0}^{N} x_i^2 \text{ (Equation 5)}$$

$$SNR = 10\log_{10}\left(\frac{E_{signal}}{E_{noise}}\right) \text{ (Equation 6)}$$

The mean voltage was calculated by taking the mean across the signal. The mean displayed no correlation with frequency. The mean voltages had an average of -0.04 V with a standard deviation of 0.4 V.

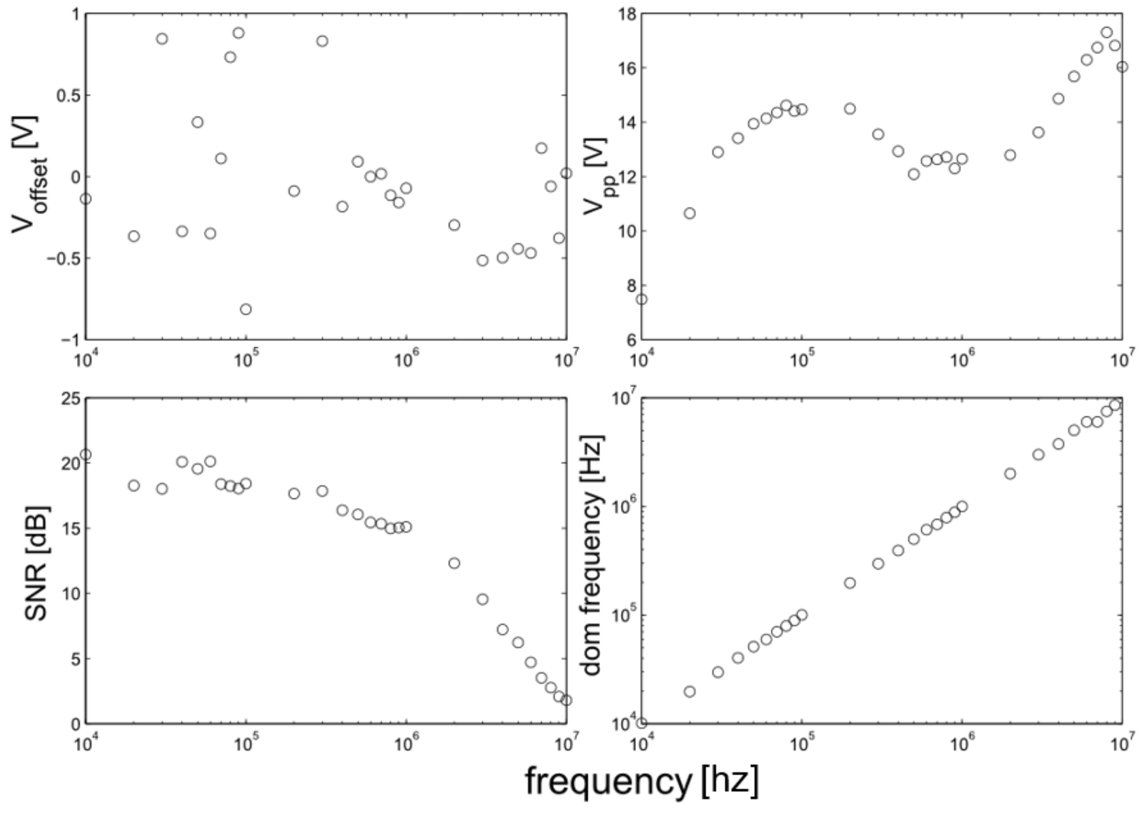


Figure 15 Summary of mean voltage, Vpp, SNR, and dominant frequency across a range of set frequencies.

Vpp is not entirely constant across the range of interest. It drops off quickly below 0.1 MHz, remains constant from 0.5 MHz to 1MHz, before increasing. Some of the increase can be attributed to the 5th order elliptical filter, whose gain increases as it approaches 10 MHz.

Environmental RF interference might cause noise on pre-amplified signal, which might also account for the change in amplitude. The Vpp can be controlled by the user by setting the VCA to apply a lower gain to the signal.

The SNR is best at lower frequencies – approximately 17 dB until it drastically falls at 1 MHz. At 10 MHz, the waveform displays visual signs of interference, as seen in Figure 12, and is unusable for IDS experiments.

An IDS experiment was conducted with beads and Ba/F3 cells. The IDS protocol was unchanged, except that the function generator PCB was swapped in for the laboratory function generator and RF amplifier. At 1 MHz, 15 V_{pp}, deflection of Ba/F3 cells was observed, as seen in Figure 16. The trajectories for these particles were identified by manually following the cell from frame-to-frame. The successful deflection of cells shows that the system can be used in IDS experiments.

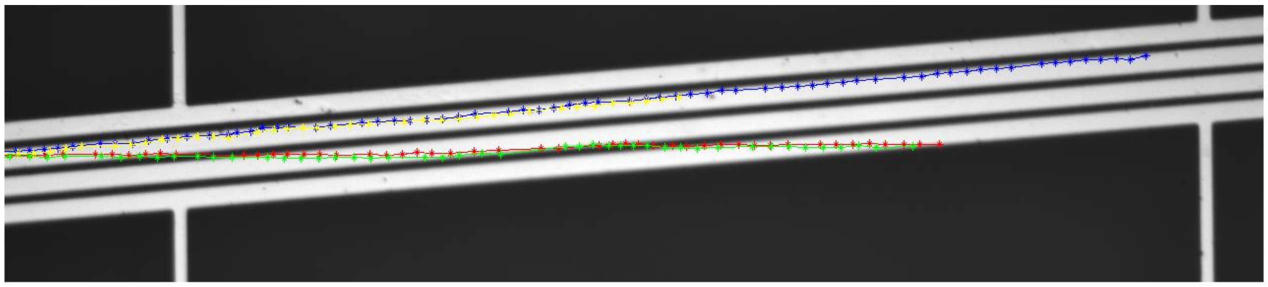


Figure 16 Trajectory of four different cells during 10 second video.

3.6 Conclusion

A partial set of the initial specifications were achieved. We did not get 24 Vpp, but we achieved a stable range of 12 Vpp. We had some attenuation at especially high frequencies, but were stable in the range of 60 kHz to 3 MHz. This was good enough to observe deflection of beads and Ba/F3 cells.

To ensure the precision of IDS experiments, one focus for future development would be to reduce attenuation at higher frequencies and have a more even behavior across frequencies. A related goal would be to improve SNR. Some ways to achieve this would be to build a faraday cage around the PCB in order to block possible RF interference.

There are also a few ways to improve the ease of use of the system. For example, like the Arduino-shield idea from earlier developments, a one-step connection between the Raspberry Pi and PCB would be much easier than connecting 6 individual wires. Another point worth addressing is the separation of power supply for the Raspberry Pi and the -12 voltage regulator. The -12 voltage regulator gets power from the same place as the Arduino. Occasionally, this regulator draws too much current, which causes the Raspberry Pi to reboot. This occasional and unexpected reboot may interfere with experiments, so it should be addressed in future layouts by making sure that any part supplying power to the power amplification part of the circuit, should draw power from a separate power supply. One last thing to improve usability of this system is to have just 1 power supply – the 12 volt one. A five volt regulator can be used to power the Raspberry Pi from this.

4 Fluidics and Optical System

Possible alternatives for the fluidics and optical systems were investigated. As with the previous section on the electrical system, portability, cost, and ease of use were considered throughout the design process. Furthermore, each system had specifications related to their function, as listed below.

4.1 Fluidics

In the original IDS system, the fluidics system consists of a syringe pump, a mechanical fluid injection valve, an inertial-based spiral sorter, and the IDS microfluidic device. The pump introduces the sample and buffer at a controlled rate. The spiral sorter sorts particles based on their size. This preliminary sorting stage is useful when performing experiments with blood, where the target of IDS is the neutrophils. The red blood cells, which are larger and undesired, are therefore sorted out. The switch is used to control which fluid enters the microfluidic device, since there is a variety of possible input fluids – for example, cleaning fluid, buffer, neutrophils, or raw sample. The microfluidic device is where the IDS occurs. The device requires a constant flow rate in order to work. In this section, an alternative to the syringe pump is investigated.

The proposed pump should have the following specifications:

- A range of flow: 0.1 μL/min to 10 μL/min
- Constant flow rate: +/- 0.1 μL/min
- Low dead volume: relative to the size of sample
- Programmable control
- Streamlined loading protocol (compared to syringe pump protocol)
- Under $100/pump cost

This list of specifications comes from a mixture of expectations from typical IDS experiments and hopeful margin of error. Typical IDS experiments with the original system used flow rates on the order of 0.5 – 2 μL/min. Therefore, the proposed system should comfortably fit anything on the order of magnitude in this range. The flow rate must also be fairly constant in order for the experiment to work. An initial goal was set for +/- 0.1 μL/min, so that it was within 10% of a typical 1 μL/min flow rate. This, however, is problematic at lower flow rates. Dead volume is the amount of sample not used in IDS. Most of the dead volume comes from the sample contained in the tube leading to the microfluidic device, and the unused sample in the spiral inertial sorter. Programmable control and streamlined loading protocol were desired for ease of use. With the syringe pump protocol, one challenge is ensuring a good connection between syringe and tubing so that no bubbles are introduced to the system. Lastly, a budget of $100/pump was targeted, where two pumps are needed for each system – one to send in sample and one to send in buffer.

In this section, a comparison of pumps is presented. Although peristaltic pumps were picked as a promising candidate for an easy to use pump, results showed it was unfeasible to achieve accuracy at low flow rates while using low-cost components. This section shows the design and experimentation with peristaltic pumps.

4.1.1 Pump Comparison

First, a variety of pumps were considered. The options considered were syringe pump, a peristaltic pump, a pressure driven pump, or a gravity driven pump. Strengths and weaknesses of each pump are shown in Table 3. The setup for these systems is shown in Figure 17.

	Strengths	**Weaknesses**
Syringe	Easy to control Precise control Already used	Cumbersome to load Costly Limited amount of fluid
Peristaltic	Easy to load Flow is proportional to voltage applied	Pulsatile flow Tube aging
Gravity	Low power consumption	Dependent on tubing used
Air Pressure	Constant flow rate. Fast response time.	Digital pressure controller is expensive. Flow varies with fluidic resistance.

Table 3 - Comparison of select pumping techniques. Table adjusted from [12].

Although the original IDS system uses a syringe pump, an alternative pump that was cheaper and easier to load was desired. A pressure driven system was avoided because an electronic valve control alone typically costs a few hundred dollars. The gravity driven system depends heavily on the tubing used and the resistance of the downstream system. While it is possible to tune it once, it makes it difficult to use if it needs to be returned for every length of tubing. This left the peristaltic pump as an option. The major weakness of the peristaltic pump – the pulsatile flow – could be overcome with pulse dampening. Although adding this additional stage would add additional dead volume, we believe we could achieve it without adding too much dead volume. Therefore, peristaltic pumps were investigated.

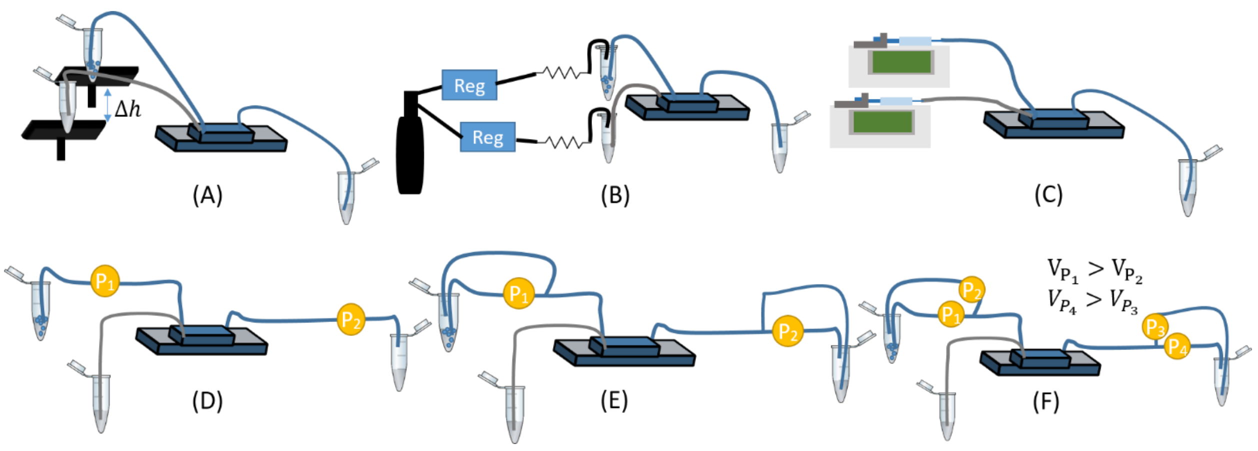


Figure 17 Pumps considered. (A) gravity-driven pump (B) air pressure regulated pump (C) syringe pump (D) peristaltic pump (E) flow-splitting peristaltic pump (F) controlled flow-splitting peristaltic pump

4.1.2 Peristaltic Pumps

The key mechanism of peristaltic pumps is a set of rollers which occlude a section of the tubing by pushing the tubing towards the inner wall of the pump. A motor is used to roll the rollers along the inside wall of the pump body. This rolling action pushes fluid from entrance to exit. Two tube collars are used to hold the tube in place, since the rollers pull on the tubing. Figure 18 shows the active parts of the pump. Peristaltic pumps are easy to load - unlike the syringe pump, the inlet tube can be simply placed in the sample reservoir. However, they have pulsatile flow, which is especially noticeable at lower flow rates. The pulsatile flow results from the fact that the roller breaks up the flow into separate sections. The flow rate of a peristaltic pump is given by $U = V * N * \omega$ (Equation 7 [13].

$$U = V * N * \omega \quad \text{(Equation 7)}$$

Where V is the volume of tubing between two adjacent occlusions, N is the number of rollers, and ω is the speed of the motor.

Lab grade peristaltic pumps, such as those from Dolomite or Takasago, cost in the range of hundreds of dollars per pump. In order to stay within the budget, a hobby peristaltic pump was chosen. A peristaltic pump from Adafruit was purchased (Adafruit 1150). The Adafruit 1150 uses a 12 V DC motor to move the rollers. The tubing provided had a 2 mm inner diameter and 4 mm outer diameter. At maximum voltage, the flow rate can reach 100 mL/min [14]. Although this flow rate is much higher than the target range, it can lowered by reducing individual terms in $U = V * N * \omega$ (Equation 7. For example, volume of occluded tubing can be decreased using tubing with a smaller radius. The speed of the motor can be lowered by operating it at a lower voltage. The Adafruit 1150's tubing had an inner diameter of 2 mm. Therefore, using a tubing of d [mm] inner diameter could theoretically reduce the flow rate by a factor of (2/d)^2.

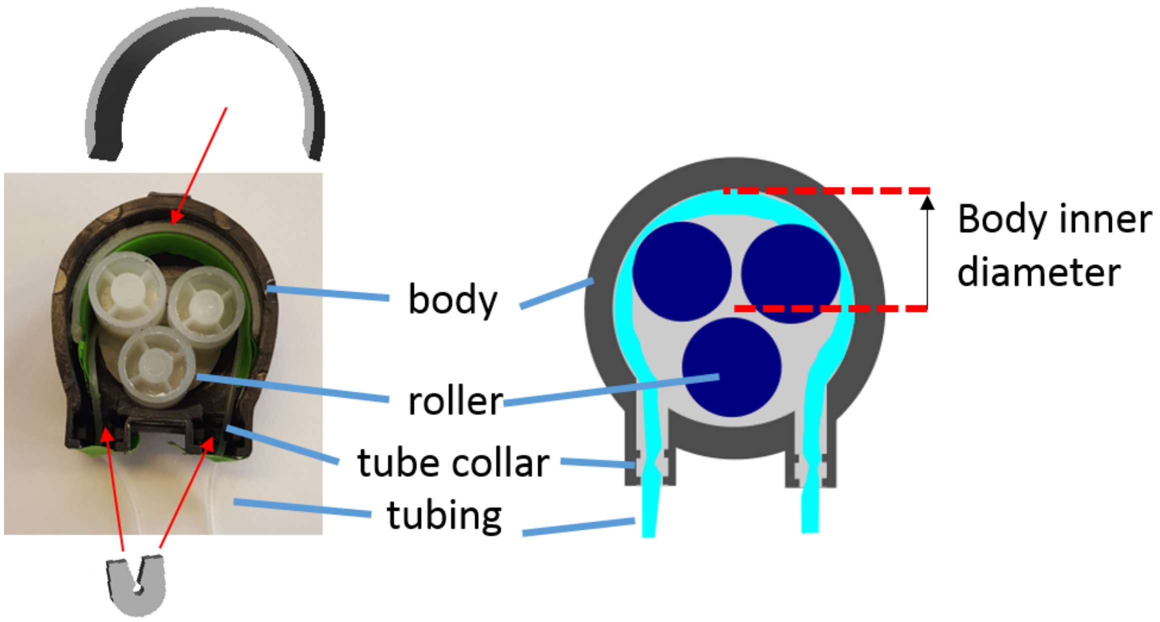


Figure 18 - Peristaltic pump. Left: a white insert lines the inside wall of the body, increasing the effective thickness of the wall. Right: a simplified diagram of the peristaltic pump. The gray figures above and below the peristaltic pump show 3D printed parts, and where they are placed in the pump. The 3D parts are designed so that the pump can function with a variety of tubing diameters.

The motor turn-on voltage acts as a lower bound for the voltage applied to the motor. Below the turn-on voltage, the motor will not turn, and therefore will not allow flow. Therefore, alternative setups using peristaltic pumps were also considered. Figure 17(E) shows a passive flow splitter. The flow from the pump is divided into two streams – one returns back to the sample, and the other is delivered to the microfluidic device. The relative values of these streams can be controlled by changing the ratio of resistances of the two tubes. Figure 17(F) shows an active flow splitter. It is similar to the passive flow splitter, except that the stream returning to the sample reservoir is controlled by another pump.

4.1.3 Peristaltic Pump Modifications

A lower flow rate can be achieved with a smaller tube diameter and a slower motor speed. There are two limitations on the smallest possible tube diameter. The first is that there must be space for the cells and beads to pass through. The cells and beads used are on the order of 10s of microns. Secondly, the material must be elastic enough so that the rollers can occlude a section of it. Therefore, tubing like PEEK tubing is not as suitable as silicone tubing. The smallest silicone tubing found had an inner diameter of 0.5 mm, which would theoretically lower the flow rate by a factor of 16. This brings the flow rate down to 6000 µL/min at 12 volts.

In order for the pump to function with smaller tube diameters, modifications were made to the peristaltic pump. Two problems arise from using a smaller tube diameter. Firstly, the rollers no longer occlude the tubing, since the edge of the tubing is now further away from the rollers. Secondly, the tube collars are too large for the tubing. When the tube collars are too large, the tubing is slurped into the peristaltic pump, removing the tubing from the sample reservoir. 3D parts, as seen in Figure 18, were printed to match the size of the smaller tubing to counteract both of these issues.

An insert was designed to counteract the first problem. It essentially extends the thickness of the peristaltic pump wall. Therefore, the outer radius of the insert matched the inner radius of the pump wall. For the Adafruit-1150 Pump, the center-to-inner-wall distance was 13.5 mm, for tubing that had an outer diameter of 4 mm. The thickness of the insert was designed to be the difference between the inner diameters of the tubes. Different tube diameters also required different tube collars. The tube collar was designed to have the same outside shape and same thickness, so that it fit into the slot in the pump body. However, the central radius was decreased according to the tube diameter. This radius was designed to be .9 of the tube's outer diameter. With these modifications, the pumps were tested to see what flow rates could be achieved.

4.1.4 Flow Rate Measurement

To measure the flow rate of the pumps, time was recorded as tap water was pumped from one tube to another. The empty tube was a either a 1 mL tube or a 10 mL falcon tube, depending on the flow rate. If the flow rate was < 10 mL/min, the smaller tube was used for finer granularity. To calculate the flow rate, the time required to get to 1 mL or 2 mL was recorded. The flow rate was therefore volume divided by time.

When running a single peristaltic pump, as seen in Figure 17(c), the voltage was supplied from a Harrison 6205B dual DC power supply. When run in the active flow splitter configuration, one pump was kept at a constant 5 V using a L7805 voltage regulator, powered from a 9V battery. The other pump was supplied from the power supply.

Tubes of different diameters the relationship between tube diameter and flow rate was verified. Using two different tube sizes, the flow rate was recorded at various voltages.

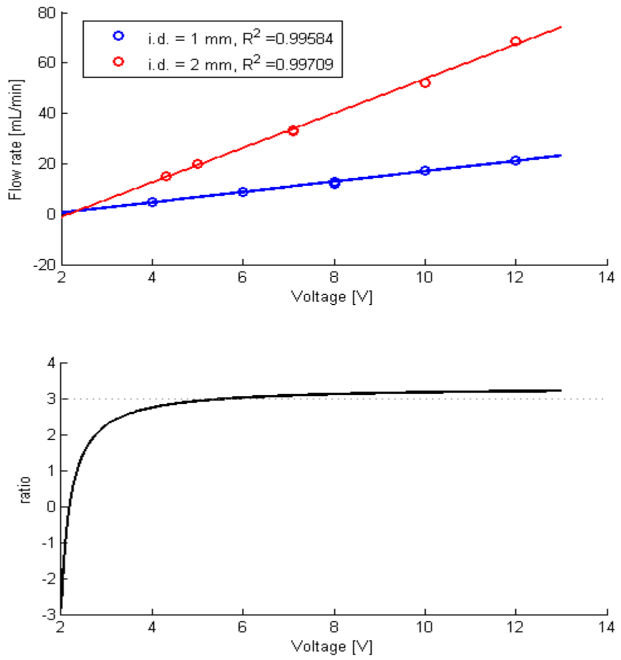


Figure 19 (a) Flow vs voltage for different tube thicknesses. (b) Ratio of flow at different voltages.

The flow-voltage for the 1 mm and 2 mm ID tubes can be modelled with the following equations, respectively:

$$flow_{1mm} = -3.70\ [mL/\min] + 2.06 \left[\frac{mL/min}{\mathrm{V}}\right] * V[V]$$

$$flow_{2mm} = -14.92\ [mL/\min] + 6.85 \left[\frac{mL/min}{\mathrm{V}}\right] * V[V]$$

4.1.5 Discussion

From the experiment, it was concluded that peristaltic pumps were an unfeasible way to achieve the specifications. Figure 19(b) shows that the flow rate does not lower as expected. From 2 mm to 1 mm, the flow rate is expected to lower by a factor of 4. However, the graph shows that factor is more around 3. This may be because the rollers occlude a larger length of the tubing, and therefore doesn't decrease it proportionally. Also, at lower voltages, the ratio is much lower than 3. This could be a result from inaccurate measurements at lower flow rates. It could also result from a weaker relationship between motor voltage and speed at lower voltages.

Another observation is that even with the 1 mm tubing, the flow rate/voltage is 3700 µL/min. This means that even if the flow rate was decreased as predicted, then flow rate/voltage needed would be 975 µL/min. Therefore, a voltage difference of 1 mV is needed in order to create differences of 1 µL/min. The voltage sources vary in the 0.01 V when driving the motor. This variation in supply voltage will make it impossible to drive the motor with enough precision.

4.1.6 Conclusion

Peristaltic pumps were explored as an easier-to-use and cheaper alternative to syringe pumps. However, it was unfeasible to achieve reliable and low flow rates using cheap peristaltic pumps, even with modifications. Although they did not work well in our desired range of 0.1 – 10 µL/min, they worked well in the range of .1 mL-10mL.

Although syringe pumps are expensive, a growing trend has been to make and share labware. One recent example is an open source 3D pump [15]. It uses a hobby stepper motor and 3D printed parts to drive a syringe. Each pump is reported to cost approximately $50 to make, fitting well within the budget. Using a syringe whose cross section is 4 cm^2, the pump can achieve 1 µL/min by taking 1 step every 6 seconds. This also runs the risk of pulsatility, but this can be reduced by using a syringe of smaller area. Potentially more costly ways to reduce the pulsatility would be to use a finer resolution stepper motor.

Whatever pump is finally used, for ease of use, it would be good to have it be integrated with the electronics and the GUI developed for the electronics.

4.2 Optics

The last subsystem of the IDS system is the optics. The original system used an Axio Imager M1.m microscope. It is an upright microscope capable of fluorescent microscopy. A GUI was developed to control the microscope settings and record functions. For use in IDS experiments, the portable optical system targeted the following specifications:

- 10x magnification
- 5 mm FOV
- Stage control in x, y, z directions
- If possible, fluorescence
- Quick setup and easy control
- Fit inside a shoebox

10x magnification was chosen since it was a commonly used when running IDS with the Axio microscope to image cells. 5 mm FOV was chosen to cover the width of the microfluidic channel, which is approximately 3 mm wide. Some experiments have involved fluorescent cells, therefore it was desired for the portable system as well. The other parameters were chosen for usability.

4.2.1 Camera and Lens System

Several other low-cost microscopy techniques have been investigated. Often, a cheap, easily found lens is used, like those in mobile phones [16] or webcams [17]. While previous systems using low-cost cameras involved conventional microscope objectives or ball lenses, these systems either suffered from poor field of view or optical aberrations as a result of the lenses [18] [19]. One way to enable the capture of high quality, wide field-of-view images is to reverse the lens of the imaging system [16]. The schematic for a reversed lens system is shown in Figure 20.

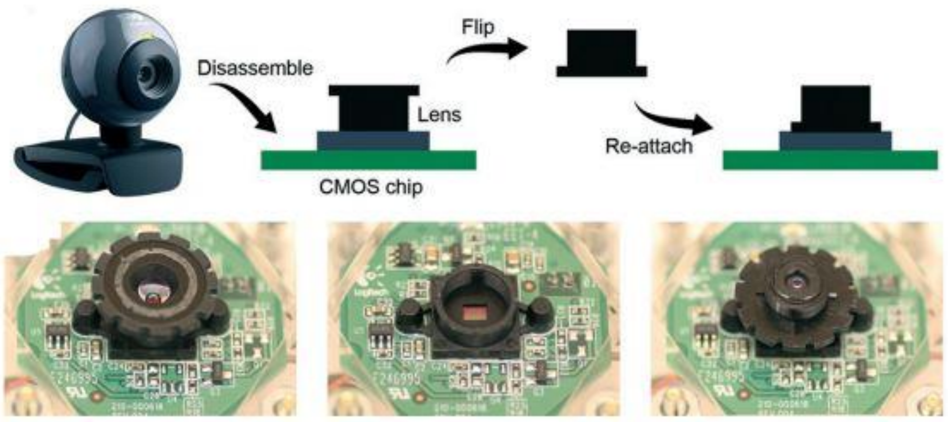


Figure 20 Lens flipping example. Adapted from Zhang YS, et al.

The microscope developed for the portable system using this technique. The lens was taken from the Logitech QuickCam webcam. The setup for the system is shown in Figure 21. An USB LED light (Daffodil ULT05) was used as a light source. It is partially taped over with opaque electrical tape in order to provide a point source of light. A mechanical stage (Newport MT Series) is used to control the x, y, and z movements of the sample. The sample is clasped down above the lens for imaging.

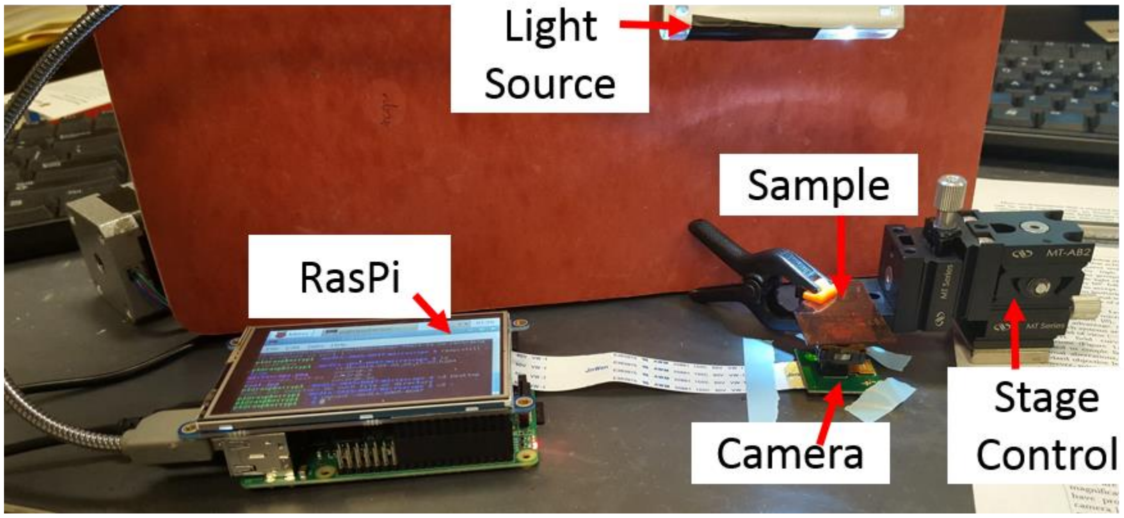


Figure 21 Microscope setup with Raspberry Pi, light source, camera, and sample.

4.2.2 GUI Background

A GUI was needed to control the microscope from the touchscreen. This GUI was inspired, though simplified, from the GUI of the original IDS system. The system must be simpler because of two limitations that result from the touchscreen. First, the small size of touchscreen limits how many buttons can comfortably manipulated on a single screen. Secondly, the lack of keyboard makes it difficult to name files. However, the availability of a preview screen, a start button, and stop button remained.

Another problem encountered was that the default Raspberry Pi camera libraries do not allow image previews, making it difficult to line up the sample with the camera, or to bring the image into focus. This is because the default libraries expect HDMI output. However, with the touchscreen, the Raspberry Pi uses a kernel that disables HDMI output, and enables GPIO output for display [20]. Therefore, code was modified from an open source camera project that used an earlier version of the PiTFT touchscreen [21]. This project relied on rapidly taking unencoded images, and loading the images to the touchscreen. Modifications to the code were required due to different versions of software and hardware. These modifications are described in the next section, GUI Integration.

The camera project used the pygame library to create an imaging GUI. Several simple helper classes were made in order to form basic elements of the GUI. An Icon class is used to organize information about various pictures associated with different elements in the GUI. A Button class is used to associate a button with a particular area pixels, an action, and an icon. In the setup, an image from the camera is loaded to the screen and the state of the GUI is initialized. The state represents different modes, such as “settings selection”, “preview”, among others. Then, an infinite while loop is used for interactive actions of the GUI. On each loop, three actions occur.

Firstly, the GUI checks if a button has been pressed. The button is associated with a pixel boundary, given by the tuple (x_min, x_max, y_min, y_max). When the user taps the touchscreen, it creates a MOUSEBUTTONDOWN event. This event is associated with an (x, y) coordinate. If this location is within the bounds of a button, the function associated with the button is executed. If there are overlapping buttons, it executes the function of the top button. Therefore, for unambiguous button execution, no buttons should overlap.

Secondly, it takes an unencoded YUV image with the camera, converts it to an RGB image, and displays it. This is the key part in making the preview functionality work with the touchscreen.

One important piece of helper code was the YUV2RGB shared object module. The Raspberry Pi camera only supported fast image capture in the YUV format, but the touchscreen display uses RGB format. Therefore, the developers of the camera project wrote C code to efficiently convert from YUV to RGB. A MakeFile was written so that the C code could be compiled into a shared library, which could be used in the python script.

Thirdly, state specific actions occur. For example, displaying the buttons for settings menus.

4.2.3 GUI Integration

The microscope GUI was adapted and integrated with the electronics GUI. As shown in Figure 22, there are buttons in both views that allow switching from one view to the other. Although the electronics GUI was written in TKinter, the optics GUI was written using the pygame library, matching the code from the open source camera project.

In the microscope GUI, there are three states: Preview Mode, Recording Mode, and Return Mode. Similar to the camera project, these states affect which state specific actions are executed in each iteration of the while loop. Also similar to the camera project, a preview is shown at all times.

There are 3 buttons on top of the preview which are always available. The red start button will enter the Recording Mode and start recording. The green done button will return to Preview Mode if a recording has been already started. When a recording is ended, the video will be saved in the .h264 format, and the file will be given a unique file name. The name of the video always begins with the string “video”, and is followed by a string representing the time in the format “YYYY-mm-dd_HH:MM:SS.” The blue return button will exit the microscope GUI and return to the electronics GUI.

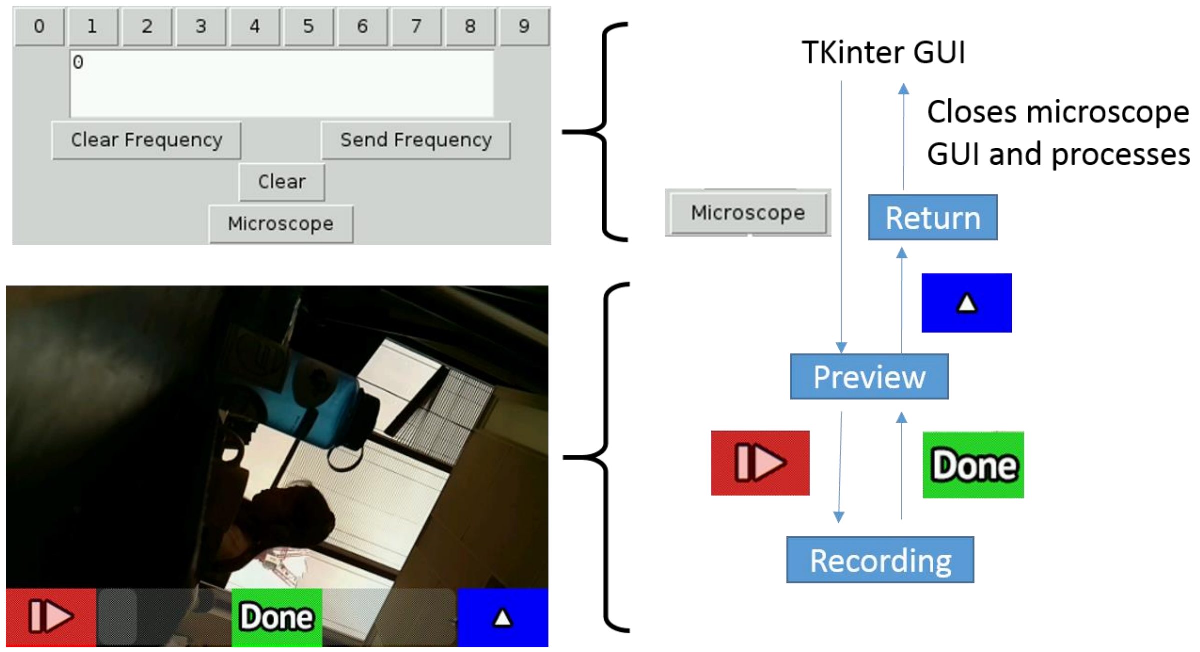


Figure 22 GUI state diagram and integration with the electronics GUI.

Each time, the loop takes executes state-specific actions. In Preview Mode, no extra actions occur. In Record Mode, the camera object would record for a set amount of time. If this time delay is shorter than the time it takes to execute the while loop, there is a chance that the camera would not record continuously. If too long, there is a chance that the camera would record a little extra. The later of these two scenarios was deemed less consequential, therefore a delay of 1 second was chosen. If the GUI is in Return Mode, the code would break out of the while loop and return to the electronics GUI.

Care was taken to handle extraneous cases of button presses. For example, if the Start button were pressed when the GUI was already in record mode, nothing would happen. Likewise if Done were pressed, when the button was in preview mode, nothing would happen. If return was pressed when the button was in record mode, it would act the same way as if Done had been pressed if the GUI was in the Record Mode. That is the recording would end and be saved to a file with a filename based off the time stamp.

Lastly, three other challenges came up when writing the GUI. These were primarily based in different in versions of software or hardware used between the open source camera project and this portable IDS microscope.

Firstly, the typical method for enabling the camera was not working on the raspberry pi. The typical method involves opening the raspi-config GUI, and using the keyboard to navigate to the "enable camera" option, and selecting it. However, while it was possible to navigate to the "enable camera" option, rebooting did not have any permanent effects. Therefore, a solution was found by editing the configuration file manually [22]. The key change was setting the variable start_x to 1.

Next was a modification to the YUV2RGB code. Modifications were needed in order for python3 compatibility. The original code was written for Python 2.7. But in order for Raspberry Pi GPIO compatibility, Python 3 was needed. Firstly, the include statement was changed in order to use the correct version of Python. Secondly, the syntax for shared objects changed between the two python versions, so the syntax in the yuv2rgb.c code also needed to change.

Lastly, there were issues with the touchscreen accurately getting the location of the mouse. This resulted in false positives and false negatives for button presses. Although the source of error is not entirely certain, changing the mouse.set_visible() from False to True allowed the true determination of cursor position.

4.2.4 Resolution Test

An Air Force target (MIL-STD-150A) was used to test resolution, magnification, and field of view. To test resolution, the intensity profile along successively smaller line pairs were examined until it was impossible to distinguish between lines. To test magnification, the image size was compared the pixel size. To calculate field of view, the pixel to distance ratio was calculated, which allows conversion from pixels to mm.

The USAF Target is manufactured to have consistent distances between line groups. Using these distances as knowns, the conversion from pixel to distance ratio can be calculated. The resolution between line pairs is given by the following equation:

$$r = 2^{g+\frac{e-1}{6}} \quad \text{Equation 8}$$

Where r is the resolution in line pairs per mm; g is the group number, given by the large number to the side of a group of lines; and e is the element number, given by the smaller number directly next to the lines.

As shown in Figure 23(a), six elements from group 4 were chosen for calibration. First, the square at the top was used to determine the tilt in the image, called θ_{tilt}. The red line is parallel to the edge of the square, while the green line is parallel to the side of the image. The angle between the two is calculated. Secondly, for every element, a point above the line pairs was picked. A script extended a line downwards at the angle θ_{tilt}. This ensured that the line was perpendicular to the line pairs, and would correspond to the resolution given by $r = 2^{g+\frac{e-1}{6}}$ Equation 8. For the clearest signal possible, the line was selected such that it would avoid crossing through scratches or discolorations on the target. Next, the image was converted from RGB to grayscale, so that an intensity profile could be drawn along the line.

The pixel difference between 2 line pairs was determined by the gap between two rising edges in the intensity profile. However, the rising edge is not sharp. There are many pixels where the intensity is still rising. Therefore, a method was devised to unambiguously pick the end points. The derivative was taken and the end points were determined to be the local maximum of the derivative during the intensity profile's rising edge. The end to end difference in pixels was matched to the theoretical distance calculated value from $r = 2^{g+\frac{e-1}{6}}$ Equation 8 for all six elements. The average of these values was used as the mapping for the image.

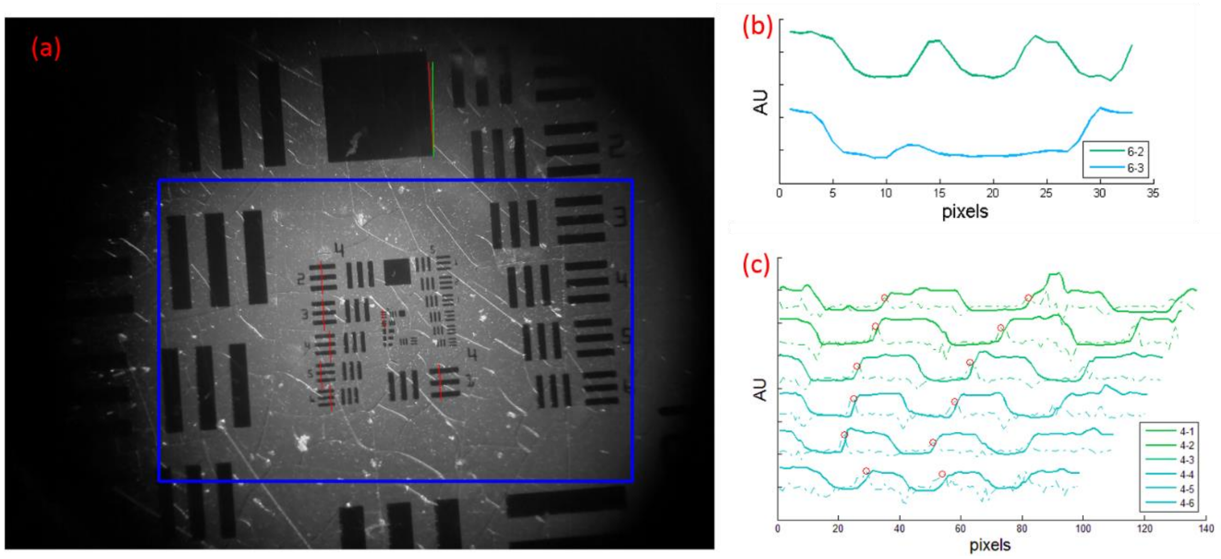


Figure 23 (a) An image from the setup. The square used for angle calibration is on top. The red line is parallel to the square and the green line is parallel to the image. The blue box represents a visible box for viewing samples. In the center, the red lines show where the intensity profiles were taken for each element. (b) The intensity profile at the lower edge of resolution. (c) The intensity profile for all elements in group four.

Figure 23(c) shows the intensity profiles along the six elements in group 4. For each element, the ratio of pixels to theoretical distance is calculated. The mean value is 738.36 pix/mm, with a standard deviation of 13.98 pix/mm.

When testing resolution, the ability to distinguish line pairs was lost between element 2 and 3 from group 6. Figure 23(b) shows three rising edges, corresponding to the 3 line pairs, for element 2. The distance between rising edges is 10 pixels, which corresponds to 0.0135 mm. This matches well with the theoretical resolution of 0.0139 mm, as calculated from $r = 2^{g+\frac{e-1}{6}}$ Equation 8. However, for element 3, there is only 1 distinguishable rising edge.

The magnification was determined to be 0.96x. The image distance per pixel is the inverse of the scale, 1.35 um. The size of a pixel is given as 1.4 um in the hardware documentation. This deviation from the expected 1x magnification as the samples axial position and thus autofocus vary. However, the effective magnification can be considered as the ratio of the display pixel pitch of the image compared to the size of the pixel. The image has 72 dots per inch (dpi) resulting in an effective magnification of 240x.

The field of view can be calculated by taking the usable resolution of the image and converting it to distance. The blue box in Figure 23(a) shows a usable area of 2.3mm x 1.5 mm.

4.2.5 Conclusions

A working prototype was built that achieved 240x magnification and had a resolution limit of 13.5 um. The field of view was 2.3 mm x 1.5 mm. Although stage control and lighting control was not automated, it could be manually tuned. The setup does not require any assembly, but requires alignment of the sample. Once the camera is plugged in, it can easily be controlled from the GUI.

However, it still needs further tests before it can be certain that it works with IDS experiments. Further additions could be made to the hardware to improve redundancy. For example, a way to keep the light source at a constant distance from the camera, or ensure that it lights the target evenly. Also, further additions could be made to the code to adjust camera settings. Currently, the camera settings are statically set in the code, and the user has no method of changing them from the GUI. If a different exposure length or ISO was desired, the code would have to be adjusted, and the GUI would have to be restarted from the adjusted code. It could make experimentation easier and smoother if it could be set and adjusted from the GUI. Additionally, it would be nice to have pop-up messages, such as “cannot take new video, already recording.” Or “file saved as NAME.”

5 Contributions and Future Directions

The overarching goal of this thesis was to create a portable version of the IDS system. Such a system was desired primarily to reduce the delay between sample collection and sample measurement. A reduced delay would reduce decay of time-sensitive samples, and thereby increase our confidence in the results. Normally, samples were taken at a nearby hospital, but measured using equipment in our lab, introducing at least an hour of delay. The portable version would allow us to measure immediately after sampling.

5.1 Summary

This goal was approached with divide and conquer outlook. Smaller systems of the IDS system were identified, and key functional specifications were targeted. The systems were divided into controls, electrical, fluidics, and optical. While each system had function-specific requirements, they were all guided by the goals of being small, cheap, and user-friendly. The electrical system and optical system were accomplished to a degree of success, while the fluidic system needs further investigation.

For the electrical system, a customized function generator PCB was built. It was tested with the other IDS equipment to ensure that it could function successfully in IDS experiments. Deflection of BAF/3 cells was observed with a flow rate of 0.5 µL/min. The PCB generated clean signals between 0.1 MHz to approximately 3 MHz.

For the fluidic system, a variety of pumping mechanisms were explored. However none of the attempts at building a pump achieved a low enough flow rates. Most of the work consisted of modifications to a peristaltic pump. The lowest flow rate achieved was ~1 mL/min. Other options consisted of recirculation setups or pressure driven pumps. Ultimately, the prospect that seemed most promising was an open source 3D printed syringe pump.

The optical system used a webcam lens and a Raspberry Pi camera to image cells at approximately 240x. The field view covers 2.7 cm of the 3 cm channel.

Lastly, the Raspberry Pi and touchscreen replaced the computer in controlling all the hardware. A GUI was written for user interaction with the function generator and microscope. The control for the fluidics was not yet written, since it was not determined which hardware would work best.

5.2 Accomplishments

During the design process, a system description was created that allowed for modularity and flexibility in the design process. Integrating the optical system into the work done on the electronics system was relatively seamless. Future directions of the project should be able to integrate themselves easily. The choice of Raspberry Pi has many GPIO pins, allowing for many external hardware extensions. Furthermore, the Raspberry Pi has good back-compatibility. Recently a new version of the Raspberry Pi was released, with a higher CPU and on-board wireless and Bluetooth features, and it is still compatible with the same touchscreen and camera.

Analog/digital mixed signal board design is often plagued with noise. Good design is especially important when the analog signals include high frequency components. Although it did not cover the entire range of frequencies desired, the PCB designed for this thesis project managed gave the user a decent amount of control over frequency and amplitude of the voltage.

The focus on ease-of-use was apparent with the end product. Although the fluidics feature was not included, the user could control a voltage source and optical source at the touch of a few buttons.

5.3 Future Directions

Perfect is the enemy of done. While all parts of the project could use more improvement and characterization, some optimizations are more impactful than others.

The largest improvement would be to find a working fluidic system. Further work should investigate whether this system works for IDS. It should be able to achieve consistent flow rates on the order of 1 µL/min. In any case, if desired, the syringe pumps from the original IDS system were the most portable part of the setup, and could also easily fit in a suitcase.

Further improvements focus on (1) expanding the reliable frequency range up to 10 MHz and (2) reworking the PCB so that the power-hungry components draw their power from a different source than the control signals power source.

Reduced magnification would be beneficial in order to image the entire width of the channel. This could be achieved by the use of additional lenses. However, some consideration should be taken to the tradeoff between additional lenses and increased cost or reduced image quality. Furthermore, it is possible to observe fluorescence with the setup. Fluorescence would be beneficial by expanding the types of studies that this system could perform. However, since the setup does not use any filters, more investigation is needed to see if the results are reliable.

Although it was not the main purpose or motivation of this project, many of the goals of this project are aligned with free open-source hardware (FOSH). With recent advances making 3D printing and microcontroller/microcomputer board more accessible, the Maker community has turned its eye towards lab equipment [23]. This open source labware has decreased the costs of a many lab equipment, while allowing users to customize and improve the equipment for their own uses. Increased work in creating robust, cheap, and functional pieces of equipment can bring benefit to both scientific and maker communities.

6 Appendix

A: Bill of Materials

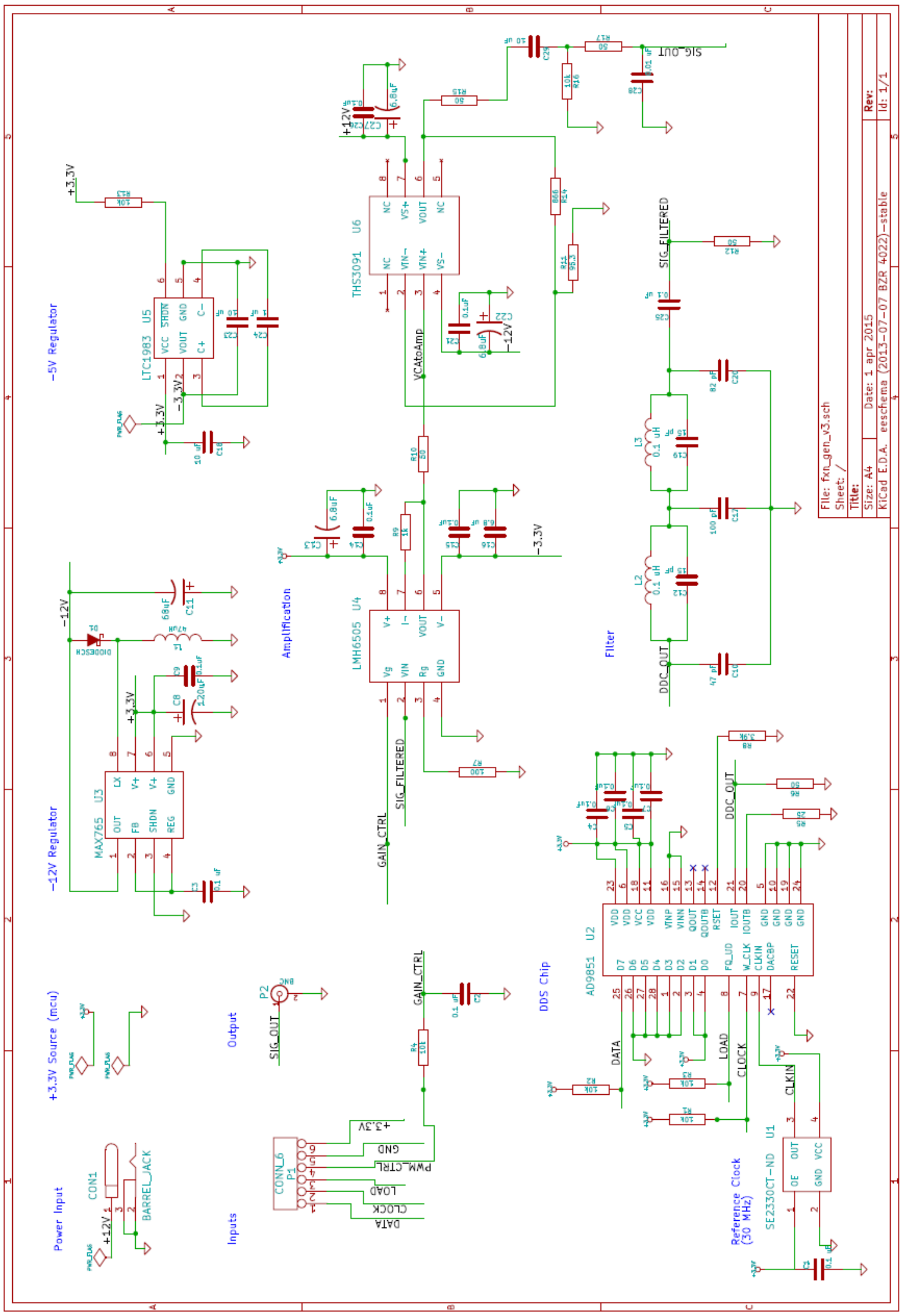
Electrical Parts

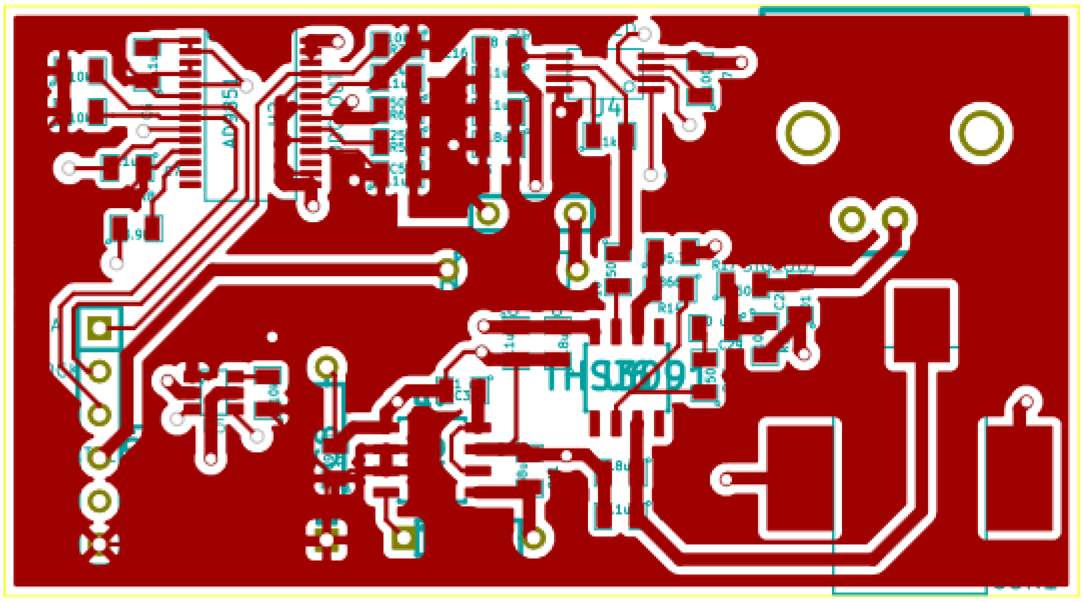
ITEM	DIGIKEY PART NUMBER	COST/ITEM	NUMBER	TOTAL
PCB	n/a	$33.00	1	$33.00
SMD CAP	1276-2193-1-ND	$0.02	2	$0.04
	1276-1037-1-ND	$0.03	1	$0.03
	1276-1242-1-ND	$0.59	1	$0.59
	1276-2451-1-ND	$0.02	9	$0.18
	587-1291-1-ND	$0.33	1	$0.33
	445-7642-1-ND	$0.03	4	$0.12
	587-1312-1-ND	$0.03	2	$0.06
	P16218CT-ND	$0.10	1	$0.10
	587-3152-1-ND	$0.12	1	$0.12
SMD RESISTOR	311-10KARCT-ND	$0.02	5	$0.10
	311-49.9CRCT-ND	$0.02	6	$0.12
	311-24.9BCT-ND	$0.12	1	$0.12
	311-100CRCT-ND	$0.02	2	$0.04
	311-3.9KARCT-ND	$0.02	1	$0.02
	311-1.0KARCT-ND	$0.02	1	$0.02
	311-866CRCT-ND	$0.02	1	$0.02
SMD INDUCTOR	587-1685-1-ND	$0.08	2	$0.16
	587-2104-1-ND	$0.25	1	$0.25
SCHOTTKY DIODE	MUR105GOS-ND	$0.30	1	$0.30
OSCILLATOR	SE2330CT-ND	$2.26	1	$2.26
VCA	LMH6505MM/NOPBCT-ND	$5.27	1	$5.27
DDS	AD9851BRSZ-ND	$22.68	1	$22.68
-12V REG	MAX765CSA+-ND	$5.87	1	$5.87
-5V REG	LTC1983ES6-5	$4.28	1	$4.28
POWER OP-AMP	296-16671-5-ND	$6.60	1	$6.60
BNC CONNECTOR	A97553-ND	$1.67	1	$1.67
BARREL JACK	CP-202A-ND	$0.93	1	$0.93
6 PIN HEADER	0022303063-ND	$0.18	1	$0.18
			TOTAL	$52.46

Other Parts

ITEM	PART NUMBER	COST/ITEM	NUMBER	TOTAL
RASPBERRY PI B+	Adafruit 1914	$29.95	1	$29.95
RPI TOUCHSCREEN	Adafruit 2097	$44.95	1	$44.95
RPI CAMERA	Adafruit 1367	$19.95	1	$19.95
			TOTAL	$94.85

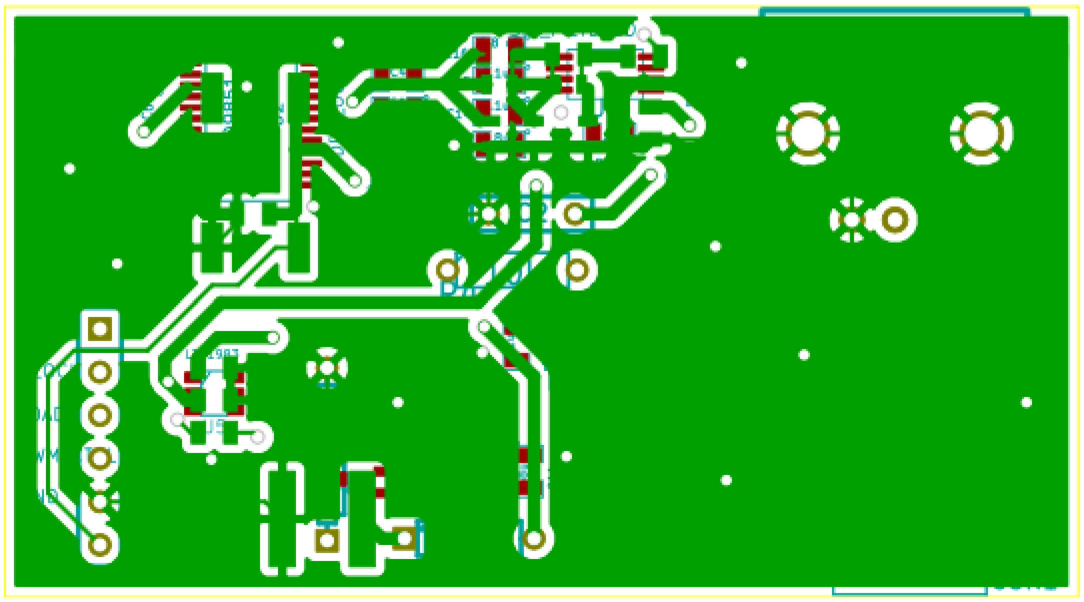
B: Circuit board schematic and layout





Above: Front of board

Below: Back of board



C: Code

GUI Code:

```
#!/usr/bin/python3

from tkinter import *
from PiFreq import PiFreq
import time
import datetime
import picamera
import yuv2rgb
import io
import pygame
from pygame.locals import *
import atexit
import os
import fnmatch

# UI classes -----------------------

class Icon:
        def __init__(self, name):
                self.name = name
                try:
                        self.bitmap = pygame.image.load(iconPath + '/' + name +
'.png')
                except:
                        pass

class PyButton:
        def __init__(self, rect, **kwargs):
                self.rect = rect        # bounds
                self.color = None
                self.iconBg = None
                self.iconFg = None
                self.bg = None
                self.fb = None
                self.callback = None
                self.value = None
                for (key, value) in kwargs.items():
                        if key == 'color': self.color = value
                        elif key == 'bg' : self.bg = value
                        elif key == 'fg' : self.fg = value
                        elif key == 'cb' : self.callback = value
                        elif key == 'value' : self.value = value

        def selected(self, pos):
                x1 = self.rect[0]
                y1 = self.rect[1]
                x2 = x1 + self.rect[2] - 1
                y2 = y1 + self.rect[3] - 1
                if ((pos[0] >= x1) and (pos[0] <= x2) and
                    (pos[1] >= y1) and (pos[1] <= y2)):
                        if self.callback:
                                if self.value is None: self.callback()
                                else: self.callback(self.value)
                        return True
                return False
        def draw(self, screen):
                if self.color:
                        screen.fill(self.color, self.rect)
                if self.iconBg:
                        screen.blit(self.iconBg.bitmap,
                                    (self.rect[0] + (self.rect[2] -
self.iconBg.bitmap.get_width())/2,
                                     self.rect[1] + (self.rect[3] -
self.iconBg.bitmap.get_height())/2))
                if self.iconFg:
                        screen.blit(self.iconFg.bitmap,
                                    (self.rect[0] + (self.rect[2] -
self.iconFg.bitmap.get_width())/2,
                                     self.rect[1] + (self.rect[3] -
self.iconFg.bitmap.get_height())/2))

        def setBg(self, name):
                if name is None:
                        self.iconBg = None
                else:
                        for i in icons:
                                if name == i.name:
                                        self.iconBg = i
                                        break
#############################
## main gui
#############################

class IDS_GUI:
        def __init__(self, master):

        ## create PiFreq class to prepare sending signals
                #self.pi_freq = PiFreq(10000)

        ## creates GUI

                self.buttons = [None]*10
                self.frequency = ""
                self.freq_text = Text(master, width=40, height = 3)

        # initialize the number buttons
                for x in range(0, 10):
                        self.buttons[x] = Button(master, text = str(x),
                                                 command = self.number_press2(x)
                                                 )
                        self.buttons[x].grid(row = 0, column = x)


        # reset to empty
                self.frequency = "0"

        # display
                self.freq_text.delete('1.0', END)
                self.freq_text.insert('1.0', self.frequency)
                self.freq_text.grid(row = 1, columnspan = 10, rowspan = 3)

        # clear frequency button

                clear_freq_button = Button(master, text = "Clear Frequency",
                                           command = self.clear_freq
                                           )
                clear_freq_button.grid(row = 4, columnspan = 5)
```

```
# send frequency button
        send_freq_button = Button(master, text = "Send Frequency",
                                  command = self.send_freq
                                  )
        send_freq_button.grid(row = 4, column = 5, columnspan = 5)

# clear all button
        clear_button = Button(master, text = "Clear",
                                  command = self.clear_voltage)
        clear_button.grid(row = 5, columnspan = 10)

# microscope button
        clear_freq_button = Button(master, text = "Microscope",
                                   command = self.open_microscope_view
                                   )
        clear_freq_button.grid(row = 6, columnspan = 10)


def number_press(self, x):
        self.frequency += str(x)
        self.freq_text.insert('1.0', self.frequency)
        self.freq_text.grid(row = 1, columnspan = 10, rowspan = 3)

def number_press2(self, x):
        def wrapper():
                self.frequency += str(x)
                self.freq_text.delete('1.0', END)
                self.freq_text.insert('1.0', self.frequency)
        return wrapper

def clear_freq(self):
        self.frequency = "0"
        self.freq_text.delete('1.0', END)
        self.freq_text.insert('1.0', self.frequency)

def send_freq(self):
        print ("frequency sent! " + self.frequency)
        self.pi_freq.send_freq(int(self.frequency))

def clear_voltage(self):
        self.frequency = "0"
        self.send_freq()
        self.clear_freq()

def open_microscope_view(self):

# initialize camera things
#############
# global variables ----------
###############

# -- global variables --------
        iconPath = './icon'
        PREVIEW = 0
        RECORDING = 1
        STOP = 2
        RETURN_HOME = 3
        state = PREVIEW

        # init pygame and screen for display
        pygame.init()
        pygame.mouse.set_visible(True)
        screen = pygame.display.set_mode((480,320), pygame.FULLSCREEN)
```

```
                # init framebuffer/touchscreen environment variables
                # this tells the RPi to use a touchscreen device
                os.putenv('SDL_VIDEODRIVER', 'fbcon')
                os.putenv('SDL_FBDEV', '/dev/fb1')
                os.putenv('SDL_MOUSEDRV', 'TSLIB')
                os.putenv('SDL_MOUSEDEV', '/dev/input/touchscreen')

                sizeData = [# Camera parameters for different size settings
                # Full res      Viewfinder      Crop Window
                [(2592, 1944), (480, 320), (0.0, 0.0, 1.0, 1.0)],
                [(1920, 1080), (320, 180), (0.1296, 0.2222, 0.7408, 0.5556)],
                [(1440, 1080), (320, 240), (0.2222, 0.2222, 0.5556, 0.5556)]]
                sizeMode = 0


                # int for image capture
                camera = picamera.PiCamera()
                atexit.register(camera.close)
                camera.resolution = sizeData[sizeMode][1]
                camera.crop = (0.0, 0.0, 1.0, 1.0)
                rgb = bytearray(480 * 320 * 3)
                yuv = bytearray((480 * 320 * 3)//2)

                # buttons
                video_name = ''

                def start_record():
                        print('start pressed')
                        global camera
                        global state
                        state = RECORDING
                        global video_name
                        video_name = ('video' +
datetime.datetime.fromtimestamp(time.time()).strftime('%Y-%m-%d_%H-%M-%S') +
                                '.h264')
                        camera.start_recording(video_name)      # write to mjpeg
format

                def stop_record():
                        print('stop pressed')
                        global state
                        state = STOP
                        global camera
                        camera.stop_recording()
                def return_main():
                        #print('return pressed')
                        global state
                        state = RETURN_HOME


                buttons = [PyButton((0, 268, 80, 52), bg='start', cb=start_record,
color=(255, 0, 0)),
                           PyButton((200, 268, 80, 52), bg='stop', cb=stop_record,
color=(0, 255, 0)),
                           PyButton((400, 268, 80, 52), bg='return', cb=return_main,
color=(0, 0, 255))]

                icons = []
                # find icons for each button
                for file in os.listdir(iconPath):
                        if fnmatch.fnmatch(file, '*.png'):
                                icons.append(Icon(file.split('.')[0]))
                # associate buttons with icons
                for b in buttons:
                        for i in icons:
                                if b.bg == i.name:
                                        b.iconBg = i
                                        b.bg = None

                t0 = time.time()
                while(True): # refresh display continually
                        for event in pygame.event.get():
                                if(event.type is MOUSEBUTTONDOWN):
                                        pos = pygame.mouse.get_pos()
                                        #print(pos)
                                        for b in buttons:
                                                pos_selected = b.selected(pos)
                                                if pos_selected : break

                        stream = io.BytesIO()
                        camera.capture(stream, use_video_port=True, format = 'raw')
                        stream.seek(0)
                        stream.readinto(yuv)
                        stream.close()
                        yuv2rgb.convert(yuv, rgb, sizeData[sizeMode][1][0],
                                sizeData[sizeMode][1][1])
                        img = pygame.image.frombuffer(rgb[0:
                                (sizeData[sizeMode][1][0] *
                                sizeData[sizeMode][1][1]*3)],
                                sizeData[sizeMode][1], 'RGB')

                        screen.blit(img,
                                ((480 - img.get_width())/2,
                                (320 - img.get_height())/2))

                        if (state == RECORDING):
                                camera.wait_recording(1)
                        for (i, b) in enumerate(buttons):
                                b.draw(screen)
                                #print((i, b))

                        pygame.display.update()

                        if state == RETURN_HOME or time.time() - t0 > 1:
#                        if time.time() - t0 > 5:
                                break
                pygame.display.quit()
                pygame.quit()
                camera.close()



root = Tk()

app = IDS_GUI(root)

root.mainloop()
root.destroy()
```

7 Bibliography

[1] D. W. W. D. C. D. e. a. Gossett, "Label-free cell separation and sorting in microfluidic systems," *Analytical and Bioanalytical Chemistry,* vol. 397, no. 8, pp. 3249-3267, 2010.

[2] E. J. C. M. H. e. a. Park, "Continuous Flow Deformability-Based Separation of Circulating Tumor Cells Using Microfluidic Ratchets," *Small,* vol. 12, no. 14, pp. 1909-1919, 2016.

[3] P. M. C. R. M. S. J. W. P. B. F. Gascoyne, "Microsample preparation by dielectrophoresis: isolation of malaria," *Lab on a Chip,* vol. 2, pp. 70-75, 2002.

[4] A. B. H. H. H. T. S. H. J. L. C. Bhagat, "Microfluidics for cell separation," *Med Biol Eng Comput,* vol. 48, pp. 999-1014, 2010.

[5] M. V. J. Vahey, "HIGH-THROUGHPUT CELL AND PARTICLE CHARACTERIZATION USING ISO-DIELECTRIC SEPARATION," *Anal. Chem.,* vol. 81, no. 7, pp. 2446-2455, 2009.

[6] H. P. J. V. J. Su, "Rapid dielectrophoretic characterization of single cells using the dielectrophoretic spring," *Lab on a Chip,* vol. 13, pp. 4109-4117, 2013.

[7] S. Burgarella and M. Di Bari, "A portable and integrated instrument for cell manipulation by dielectrophoresis," *Electrophoresis,* vol. 36, no. 13, p. June, 2015.

[8] J. Cheng, L. Wu, M. J. Heller, E. Sheldon and e. al., "Integrated portable biological detection system". USA Patent 7857957 B2, 28 Dec 2010.

[9] X. Qiu, D. Chen, C. Liu, M. G. Mauk, T. Kientz and H. H. Bau, "A portable, integrated analyzer for microfluidic – based," *Biomedical Devices,* no. 13, pp. 809-817, 2011.

[10] X. Zhang, Z. Chen and Y. Huang, "A valve-less microfluidic peristaltic pumping method," *Biomicrofluidics,* no. 9, pp. 1-8, 2015.

[11] A. Miller, G. Davis, R. Richards-Kortum and e. al, "Portable, Battery-Operated, Low-Cost, Bright Field and," *PLoS One,* vol. 5, no. 8, pp. 1-3, 2010.

[12] J. S. Cybulski, J. Clements and M. Prakash, "Foldscope: Origami-Based Paper Microscope," *PLoS ONe,* vol. 9, no. 6, pp. 1-11, 2014.

[13] J. Hu, X. Cui, X. Xu, B. Gao, T. Wen, T. Jian Lu and F. Xu, "Portable microfluidic and smartphone-based devices for monitoring of cardiovascular diseases at the point of care," *Biotechnology Advances,* vol. 34, no. 3, pp. 305-320, 2016.

[14] H. P. J. V. J. H. J. Su, "Monitoring sepsis using electrical cell profiling in a mouse model," RSC, Cambridge, 2014.

[15] M. Leonard, "Raspberry Pi or Beaglebone Black," Michael Lenoard Blog, 18 July 2013. [Online]. Available: http://michaelhleonard.com/raspberry-pi-or-beaglebone-black/. [Accessed 5 17 2016].

[16] L. Fried, "Adafruit PiTFT 3.5" Touch Screen for Raspberry Pi," Adafruit, 2 April 2016. [Online]. Available: https://learn.adafruit.com/adafruit-pitft-3-dot-5-touch-screen-for-raspberry-pi/easy-install. [Accessed November 2015].

[17] L. Fried, "Extras!," Adafruit, 28 July 2015. [Online]. Available: https://learn.adafruit.com/adafruit-pitft-28-inch-resistive-touchscreen-display-raspberry-pi/extras#boot-to-x-windows-on-pitft. [Accessed March 2014].

[18] L. Fried, "More Tips," Adafruit, 28 July 2015. [Online]. Available: https://learn.adafruit.com/adafruit-pitft-3-dot-5-touch-screen-for-raspberry-pi/more-tips. [Accessed March 2014].

[19] Texas Instruments Incorporated, "LMH6505 Wideband, Low Power, Linear-in-dB, Variable Gain Amplifier," Texas Instruments, 2013.

[20] G. Casuillas, "How to choose the right microfluidic flow control system?," Elveflow, [Online]. Available: http://www.elveflow.com/microfluidic-tutorials/microfluidic-reviews-and-tutorials/how-to-choose-right-microfluidic-flow-control-system/. [Accessed 14 April 2016].

[21] Saint-Gobain, "Peristaltic Pump Tubing Formulations, Flow and Pressure Rates," 14 April 2016. [Online]. Available: http://www.processsystems.saint-gobain.com/product_detail.aspx?id=258992#Theoretical.

[22] Adafruit, "Peristaltic Liquid Pump with Silicone Tubing," [Online]. Available: https://www.adafruit.com/products/1150. [Accessed 14 April 2016].

[23] B. Wijnen, E. J. Hunt, G. C. Anzalone and J. M. Pearce, "Open-Source Syringe Pump Library," *PLoS ONE,* vol. 9, no. 9, 2014.

[24] N. A. Switz, M. V. D"Ambrosio and D. A. Fletcher, "Low-Cost Mobile Phone Microscopy with a Reversed Mobile Phone Camera Lens," *PLOS One,* vol. 9, no. 5, pp. 1-7, 2014.

[25] Y. Zhang, J. Ribas and K. A., "A cost-effective fluorescence mini-microscope for biomedical applications.," *Lab on a Chip,* vol. 15, no. 18, pp. 3661-9, 2015.

[26] B. DN, M. RN, S. NA, L. WA and F. DA, "Mobile Phone Based Clinical Microscopy for Global Health Applications," *PLoS ONE,* vol. 4, no. 7, p. 6320, 2009.

[27] S. Z, C. K, E. A, R. M, G. A and e. al, "Cell-Phone-Based Platform for Biomedical Device Development and Education Applications," *PLoS ONE,* vol. 6, no. 3, p. 17150, 2011.

[28] D. Jones, "FAQ," Picamera, 2014. [Online]. Available: http://picamera.readthedocs.io/en/release-1.10/faq.html. [Accessed 24 April 2016].

[29] P. Burgess, "DIY WiFi Raspberry Pi Touchscreen Camera," Adafruit, 16 July 2015. [Online]. Available: https://learn.adafruit.com/diy-wifi-raspberry-pi-touch-cam/overview. [Accessed 23 April 2016].

[30] Octopus, "How can I enable the camera without using raspi-config?," Stack Exchange, 25 April 2015. [Online]. Available: http://raspberrypi.stackexchange.com/questions/14229/how-can-i-enable-the-camera-without-using-raspi-config. [Accessed 25 April 2016].

[31] T. C. C. e. a. Baden, "Open Labware: 3-D Printing Your Own Lab Equipment," *PLoS Biology,* vol. 13, no. 5, pp. 1-12, 2015.