POLITECNICO DI TORINO

Master Degree in Biomedical Engineering

Master Thesis

A rapid detection method of Urinary Tract Infection

Use of an optofluidic sensor based on fluorescence and development of two communication systems with computer interface display

Thesis Advisor
prof. Alberto Vallan

Candidate
Luigi Ungaro

Academic Year 2018 – 2019

To my father, my
mother and my sister
for their continuous
support
and infinite love

Acknowledgments

First of all, I would like to thank professor Tao Dong for accepting me in his research group and giving me all the helps I needed to enjoy my stay in Norway and work to the best of my capabilities.

I would like to show my gratitude to Dr. Nuno Pires for pointing me the right direction to follow and his precious advices and support, and to Dr. Birgitte Hønsvall for her assistance during my laboratory tests.

I am also thankful to Francesco Dell'Anna for introducing me to the research group and helping me in the difficulties.

Finally, I would like to acknowledge my research colleagues Fredrik Grønvold, Helene Berntsen and Luis Silvia for make me feel comfortable since the first moments and their priceless assistance in the experiments. A special mention to João Simões for our daily exchange of views and his inestimable help.

Contents

List of Tables

List of Figures

Summary

Urinary tract infection is one of the most common infection in the world, affecting a lot of people every year. Children, adults and elderly may suffer from this disease, with the elderly being the part of population with the most incidence.

In this work it has been presented an overview about urinary tract infection and all the consequences linked to it. In particular one of the big concern is the lack of a rapid screening method that allow to eliminate as soon as possible all the negative results and avoid the overuse of antibiotics that results in antibiotic resistance patterns from different microorganisms. An optofluidic sensor based on the intrinsic fluorescence of tryptophan has been used as diagnostic tool. The sensor has been developed by other students of the University of South-Eastern Norway as part of the BiWAS project (Biological Water Alarm System), with the purpose to detect biological contaminants in the drinking water distribution network. The design, the electronic and the optical components of the sensor will not be objects of deepened considerations. This report focused on the use of this sensor in urine pathogen detection, especially Escherichia coli, comparing the spectrometry analyses of urine and drinking water and proposing possible adjustments to improve the performances.

Then, two different communication systems were realized to allow the access of the output sensor remotely. First, two radio frequency antennae were used for a short distance communication. For a remote communication instead, with the help of a Raspberry Pi, it was realized a client-server type connection to access to the sensor through Internet. Both the communication systems were linked with two MATLAB interfaces conveniently adjusted depending from the communication kind. The two interfaces consist of two different windows. The first one shows the signal in a real time graph and allows the user to trigger an alarm based on a double threshold method. The second one displays a specific time interval signal, chosen by the user. For the second interfaces a third window was realized to read remotely and simultaneously two sensors, indicating for each the geographical position. Tests with water and non-biological urine were performed.

Chapter 1

Urinary Tract Infection

In this chapter the causes, epidemiology, etiolozogy, disease evolution and incidence of Urinary tract infection have been analyzed. A review of the standard detection methods has been performed.

Urinary tract infection (UTI) is one of the most frequent infection, afflicting 150 million people every year worldwide [1]. UTI is caused by microorganisms penetrating the urethra, the lowest region of the urinary system, and infecting a segment of the urinary tract [2] (in physiological conditions the tract above the urethra is sterile).

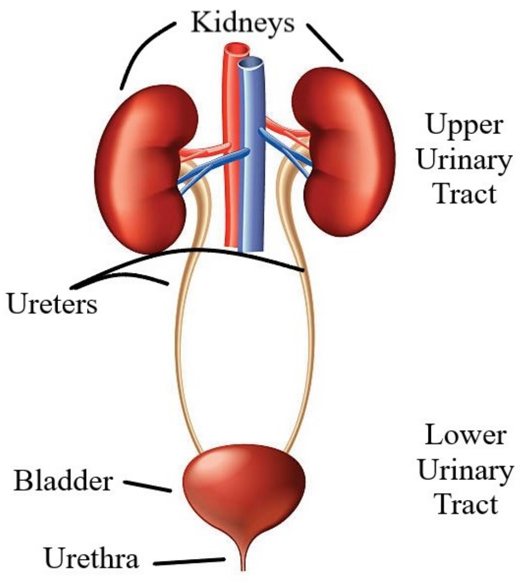


Figure 1.1: Urinary system representation. Adapted from [3]

The infection may be restricted to the lower tract or bladder, defined as cystitis, prostatitis or urethritis or may be related to the kidney as renal or upper tract infection, also defined as acute or chronic pyelonephritis, interstitial nephritis or renal abscesses [4].

Medically, UTIs are also categorized into uncomplicated and complicated. Uncomplicated UTIs usually take place in subjects without any anatomical, structural or neurological urinary tract anomalies [5], and they are the most frequent kind of infection [6]. Complicated UTIs are infections related to the presence of an irregular urinary tract, or conditions that damage the urinary tract, comprising urinary retention as neurological disease effect, renal failure, urinary obstruction, renal transplantation, pregnancy or they are associated with external bodies, like calculi, catheters or other drainage devices [7].

Table 1.1: List of frequent causes of complicated UTI. Credits to [6]

Bladder outflow obstruction
-prostatic disease
-urethral disease
-bladder neck obstruction
Neuropathic bladder
-multiple sclerosis
-diabetes mellitus
-spina bifida
-spinal cord trauma
Catheterization or stent
Renal tract stone disease
Instrumentation of the renal tract
Renal or urinary tract malignancy
Reflux
Duplex system
Ileal conduit
Pregnancy
Glycosuria
Kidney transplant
Immunosuppression

Additionally, UTIs are divided in health-care associated (nosocomial) infections and community acquired infections, according to patient typology.

UTIs are the second most frequent infections, behind only the upper respiratory infections [9], representing 25% of all the infections [10]. UTIs are the fourth most common kind of healthh-care associated infections (HAI) in the United States, about 13% of the entire HAIs recounted every year [11]. It has been recorded that more than 1 million cases in the health-care facilities are due to catheter associated UTIs (CAUTI). In the USA, the estimated societal costs of this disease, together with the health care costs and the time missed from work, are \$ 3.5 billion every

Table 1.2: UTIs classification. Credits to [8]

Lower UTIs
Cystitis, prostatitis,urethrithis
Upper UTIs
Acute pyelonephritis, chronic pyelonephrtis, renal abscess, interstitial nephritis, perirenal abscess
Uncomplicated Lower or Upper UTIs
Community acquired infection; frequent in women and not recurrent
Complicated Lower or Upper UTIs
Often nosocomial (acquired in hospital) and related to catheterization

year [12]. UTI is the third most frequent infection in India [13] and the most prevalent bacterial infection in newborns and children [14].

1.1 Epidemiology

Everyone is vulnerable to urinary tract infection, but the incidence of this disease changes with age, sex, a UTI history, diabetes, obesity, sexual habits, genetic susceptibility and other conditions [15]. Different studies, conducted on distinct age groups (preschool, school age, young and elderly) demonstrate that infections may be contracted both in the first days after the birth and during life.

During life, females have more infection possibility than males, except for infants and catheter-associated infection [9]: in fact bacteria more effortlessly may colonize the bladder of females because of the shorter urethra and the closeness of urethra to vaginal cavity and anal rectum [16]. According to some authors 50% of women will have a UTI episode at least once in their lifetime [17].

In the first months of life, infection is more frequent in males than in females, with a 4:1 proportion, instead, in the preschool children the ratio is inverted, passing to 15:1 and increasing to 30:1 in school age children. Almost 5% of all girls have a UTI episode before completing high school [18]. Kass et al. reported that 30% of interviewed women between 20 and 40 years have admitted that they have had UTIs, around 25% of these subjects heal spontaneously, but more than 50% have been reinfected within one year [19].

UTI in pregnancy may occur in up to 20% of cases and may be related to harmful consequences for both the mother and the fetus if left untreated: it may drive to acute and symptomatic pyelonephritis, which often happens before the third semester [20]. Pyelonephritis in pregnancy is the most common reason of hospitalized women during this period and the possibility of preeclampsia, premature birth, perinatal mortality and insufficient development of the child is very high [20, 21].

The prevalence of infections in elderly is very different than that in younger adults. In males is pretty infrequent before 50 years, but anatomical anomalies due to deteriorated bladder tone, prostatic obstruction or neurogenic bladder after a stroke, may block total emptying of the bladder and cause persistent and chronic UTI [8, 22]. Elderly women usually have partial evacuation of the bladder and urinary stasis and the postmenopausal estrogen lack increase the susceptibility to colonization and adherence of bacteria to the urinary tract [8]. The incidence of infection in the elderly goes from 5 to 20% in men and to 10 to 20% in women [23].

1.2 Etiology

Bacteriuria refers to the presence of bacteria in urine [24] and it may be symptomatic or asymptomatic. A significant bacteriuria is characterized by the presence of more than or equal to 10^5 colony forming units (CFU) of one identified organism per mL of urine (Kass criteria) in urine culture [25].

Asymptomatic bacteriuria (ASB) is a condition of significant bacteriuria without UTI symptoms and almost half of all subjects with bacteriuria does not exhibit any symptoms. A lot of indications and symptoms are correlated with UTI: frequency (urination more than every 3 hours), urgency, incontinence, dysuria (pain during urination) and suprapubic or pelvic pain are often associated to a lower urinary tract infection; flank pain, fever, flank tenderness, instead, may be signs of a pyelonephritis [8, 12]. Often, only microorganism's presence in urine sample does not have importance because not just infection, but also colonization and contamination may be causes of bacteriuria. Many times, bacteriuria occurs with pyuria (presence of white blood cells or pus in the urine [26]), clear sign of foreign microorganism invasion in the urinary tract [9].

Gram-negative and Gram-positive bacteria are the main cause of UTI, but virus, fungi and parasite may also begin an infection [27]. Most of the UTIs origin from gram negative bacteria, with uropathogenic *Escherichia coli* (UPEC) being the prevalent causative organism for both complicated and uncomplicated and both community acquired and hospital acquired infections [2, 4–10, 12–22, 28, 29]. In uncomplicated and complicated UTIs, *E. coli* is succeeded by *Klebsiella pneumoniae, Staphylococcus saprophyticus, Enterococcus faecalis, group B Streptococcus, Proteus mirabilis, Pseudomonas aeruginosa, Staphylococcus aureus and Candida*(fungal pathogen) [2, 30].

Viruses infections are caused by *BK Virus, adenovirus* and *cytomegalovirus* and they are responsible for haemorrhagic cystitis, principally in patients who had a kidney transplantation [27].

Uropathogens may be transferred through different ways like fecal or oral contact, contaminated water, contaminated food and the transmission occurs through direct contact [10].

As stated above UPEC is the most frequent pathogen causing an infection, hence in this work the attention is focused on it and in the nex chapters tests on *E. coli* detection have been performed.

UPEC, usually, lives in the lower intestine and it may colonize the urinary tract through ascending mechanism. Adherence is a fundamental event in UTI pathogenesis. A UTI begins with periurethral contamination, then the pathogen colonizes the urethra and migrates to the bladder, a phenomenon possible thanks to its mobility provided by appendices, like flagella and pili. These extremities allow the bond to bladder epithelium receptor cells (uroepithelium) and subsequent invasion of the bladder epithelium, producing toxins and proteases, essential for taking out nutrients from the host cells. The uropathogen may rise up to the kidneys using the same mechanism described before, colonizing the renal epithelium and causing tissue-damaging toxins [30]. UTI may lead to bacteraemia[28] (defined as "the presence of bacteria in the blood" [31]), bacteraemiato sepsis (defined as "the host response to bacteria" [31]) and septic shock, which has an high mortality rate [29]. Because of the possible deadly consequences of UTI, it is important to have a rapid diagnosis to quickly care of the infection.

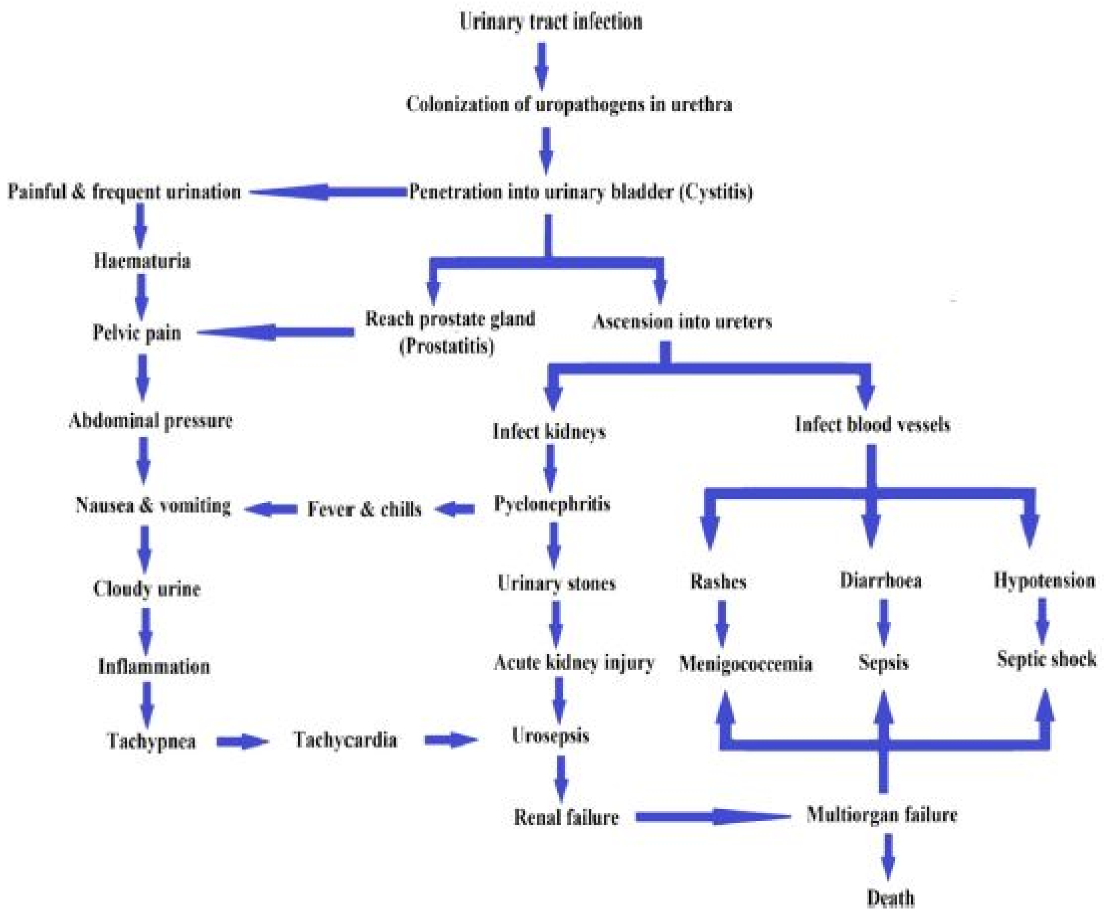


Figure 1.2: Flow diagram describing possible fatal effects of urinary tract infection. Credits to [12]

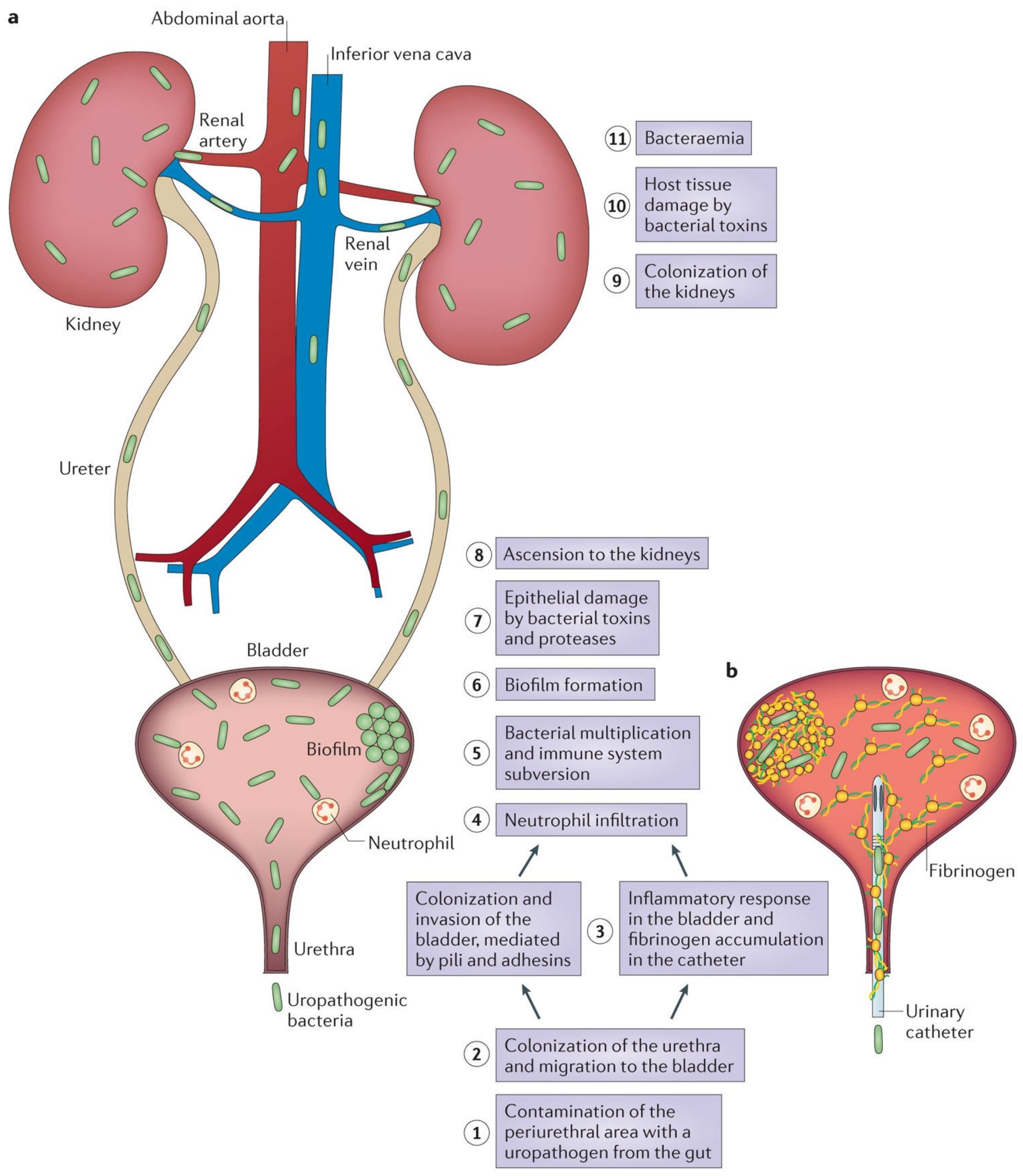

a
Abdominal aorta
Inferior vena cava
Renal artery
Renal vein
Kidney
Ureter
Bladder
Biofilm
Neutrophil
Urethra
Uropathogenic bacteria
11 Bacteraemia
10 Host tissue damage by bacterial toxins
9 Colonization of the kidneys
8 Ascension to the kidneys
7 Epithelial damage by bacterial toxins and proteases
6 Biofilm formation
5 Bacterial multiplication and immune system subversion
4 Neutrophil infiltration
Colonization and invasion of the bladder, mediated by pili and adhesins
3
Inflammatory response in the bladder and fibrinogen accumulation in the catheter
2 Colonization of the urethra and migration to the bladder
1 Contamination of the periurethral area with a uropathogen from the gut
b
Fibrinogen
Urinary catheter

Figure 1.3: Uncomplicated a) and complicated b) urinary tract infection pathogenesis. Credits to [30]

1.3 Susceptibility factors

There are a lot of factors that increase the possibility to contract an infection. They varies with age, gender, genetic susceptibility and a lot of risks and comorbid factors are correlated with UTI. Below some of them are described.

1.3.1 Urinary incontinence

Urinary incontinence (UI) refers to the uncontrolled release of urine [32], and it is very frequent in the population over 65 years and increases with age [33–35]. Women appear to suffer more of UI than men [33]. Often a diaper use is related to UI to reduce the side effects, but it has some advantages and disadvantages [36]. From service provider and nurse point of view, diapers are simple to manage and avoid too much clothes or sheets changes. From patient point of view, diapers allow more motility than catheter use, but it is not an appropriate solution and does not cure incontinence. Some studies reveal that a long-term diaper use irritates the skin and increase the possibility in UTI pathogenesis, compared to subjects that do not wear it [37].

1.3.2 Catheterization

Urinary catheters are the reason of the most part health care-associated UTIs [7]. Catheter insertion may occur during a surgery, urine output measurement, to drain urine in case of urine retention or in subject with UI [38], and in this moment that bacteria penetration may happen, through catheter lumen or at the interface between urethra and catheter. Then bacteria, attached to the surface, start to reproduce forming microcolonies and maturing in biofilms [7, 39]. To reduce infection risk it is possible to use catheters with antimicrobials or microagents that decrease the bacterial absorption to the surface. The catheterization duration is the most relevant factors of infection risk. Inadequate catheterization period is caused by "forgotten" catheters by care provider [39].

1.3.3 Postvoid Residual Volume

Postvoid Residual Volume (PVR) is the remaining urine volume in the bladder after voluntary urination [40]; PVR estimation may help to recognize patients that need more accurate evaluation. Considerable PVR is related with UTI, in particular in children or subjects with spinal cord injury or diabetes. Very high PVR (>300 ml) may increase the possibility of upper UTI and renal insufficiency [40]. Jung et al. concluded that subjects with more than 65cc of PVR have elevated chances to contract UTI even if no symptoms are detected [41].

1.3.4 Pregnancy

Possibility of UTI in pregnant women is much higher than in non-pregnant ones and the consequences for the mother and the child may be very severe [21]. The increased incidence is associated with anatomical and functional urinary tract changes, especially the expanded uterus contracts the bladder, augmenting the pressure, which may cause urethro-vescical reflux and urine stasis in the bladder after the urination [42].

1.3.5 Diabetes

The incidence of bacteriuria is three times bigger in diabetic women in comparison with women without diabetes and more than half of diabetic subjects with ASB have upper UTI [43]. Furthermore, in diabetics, UTI is harder to care for and there is an increased possibility of chronic UTI [44]. Geerling et al. revealed that in subjects with diabetes clinical complications associated with UTI are more frequent, with higher risk of pyelonephritis and consequent renal impairment [45].

1.4 Standard diagnostic procedures

Urine is an important body fluid, especially for UTI's, kidney diseases or diabetes diagnostic procedures [46] and it has the advantage of inexpensive and non-invasive sampling compared to blood. As stated before, UTI may have potential dangerous consequences for the subjects who have been infected, hence it is very important to have a rapid diagnostic tool to analyze urine samples almost in real time and to apply therapy before the infection leads to bacteraemiaand urosepsis event [47].

The gold standard for UTI diagnosis includes pathogens investigation in urine samples in presence of clinical symptoms [48]. The detection and identification occurs by urine culture (using midstream urine), often in agar medium, but different bacteria need a specific culture medium, particular additions for their best growth and the incubation time is not the same for every pathogens [12, 49]: the time to see visible colonies varies from 24 to 48 h [49]. So, bacterial culture is a time consuming and costly method [50] and a cost-effective and reliable screening test would have a great influence on laboratory time and economy [51].

Many articles in literature establish a 10^5 CFU/ml as threshold value to identify a case of bacteriuria, even if in this way it may miss some relevant infections [52]. Others recommend 10^3 CFU/ml, depending on the gender and the bacteria [9, 48].

Standard diagnosis is usually established on a first clinical part with estimation and documentation of signs and symptoms, followed by laboratory test, like urine culture, and sometimes pre-screening test is performed [2]. A pre-screening test has the purpose to assist the handling for groups at high infection risk, such as residents of long term care facilities (LTCF), to adopt more loosen infection or

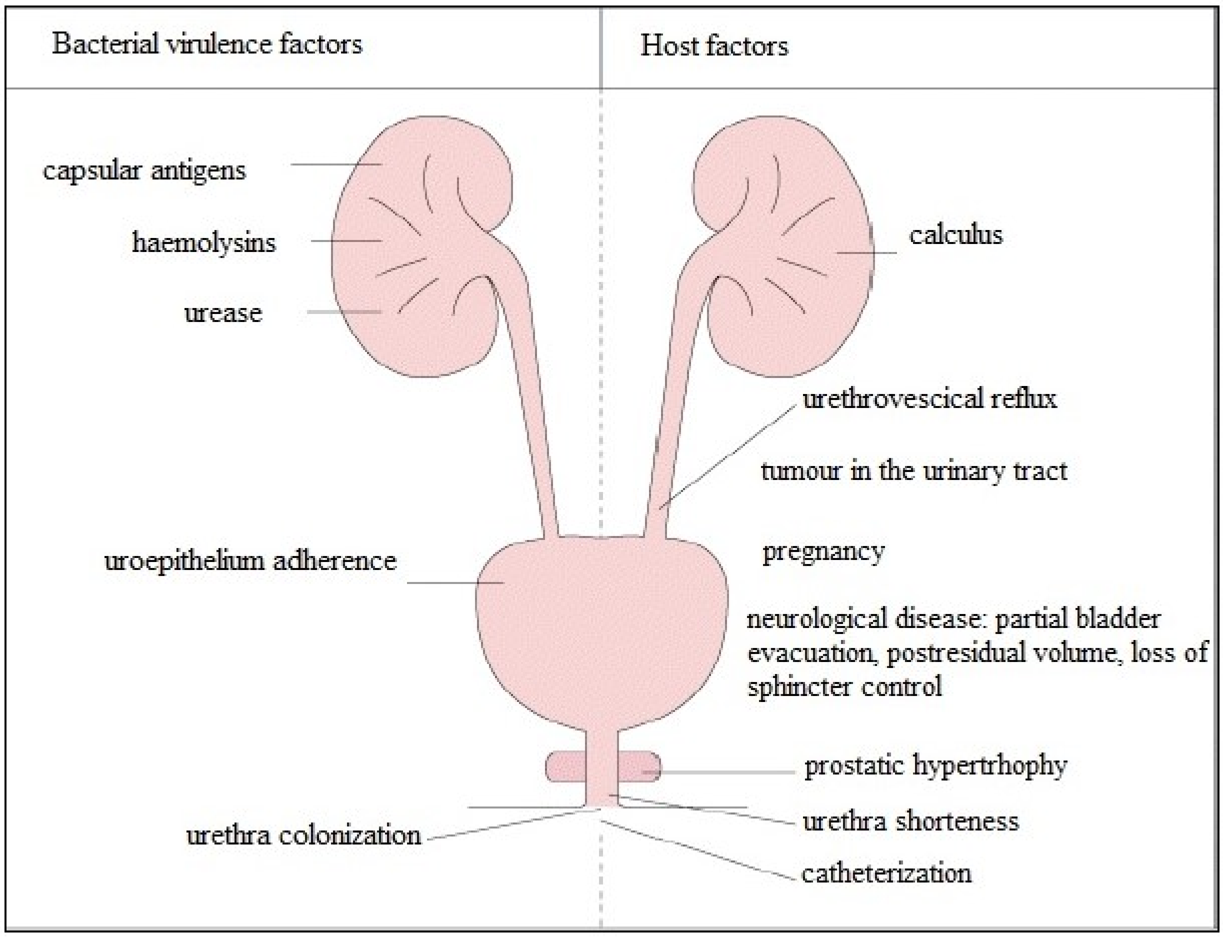


Figure 1.4: UTI predisposing factors

antibiotic prescription criteria [53, 54] and to make more rapid the analysis [55]. This may point to antibiotics abuse and subsequent bacterial resistance [56]. An accurate and quick analysis may give the possibility to eliminate a lot of negative samples and avoid worthless antibiotic prescription [55]. Nevertheless, there is a deficit of accessible screening tools successfully employed by the clinicians.

1.4.1 Urine dipstick

Urine dipstick is the main rapid instrument and commonly used screening tool to diagnose UTI if there are clear evidences of infection [48]; it can detect vary biomarkers, such as leukocyte esterase, protein and blood (markers of infection) [57] or nitrite, metabolic product of standard infecting microorganisms of urinary tract [58]. Nitrite detection rises the infection probability. Grude et al. [59] reached a sensitivity of 84% and specificity of 45% for leukocytes and McIsaac et al. [60] achieved a sensitivity of 36% and specificity of 89% for nitrite.

1.4.2 Flow cytometry

Urine flow cytometry use scatter and fluorescence properties to recognize pathogens [61]. Many studies revealed that it may be possible to achieve high sensitivity level in contaminants determination with flow cytometry, reducing the samples handled with urine culture [62, 63]. For example, the Sysmex UF-1000i Urine flow cytometer may reach a sensitivity/specificity of 100%/62% at a cut-off of 10^5 CFU/ml and sensitivity/specificity of 95%/43% at a cut-off 10^3 CFU/ml [61]. The flow cytometry tends to have low specificity at the lower cut-off levels.

1.4.3 CultureStat Rapid UTI Detection System (CSRUDS)

The CSRUDS estimates bacterial growth during log phase of growth by observing collection of bacterial mass and dehydrogenase [47]. The CSRUDS may analyze samples in 30-90 minutes and it was demonstrated that the device has high negative prediction value with a cut-off of 10^5 CFU/ml, but it need 90 minutes to eliminate a lot of false positive. Despite 90 minutes is rapid compared to urine culture, it is not as rapid as acceptable from a rapid screening method.

In this project it has been proposed an attempt to overcome the limitations from the current diagnostic tools, especially it aims to reduce the results time, producing a quick response, within few seconds, about presence/absence of contaminants in urine and decreasing the cost. This it has been achieved using an optofluidic sensor based on the intrinsic fluorescence light coming from the constituent molecules of the main pathogen *E. coli.* In addition, it may be extended to other biomedical fields, analyzing different body fluid like blood, serum or saliva and diagnose multiple diseases, or in drinking water distribution network.

Chapter 2

Fluorescence

In this chapter it has been described the fluorescence phenomenon and its possible applications in biomolecules detection. Particular attention has been given to the intrinsic fluorescence.

Fluorescence is a luminescence phenomenon that occurs when certain molecules, called fluorophores, adsorb high energy or low wavelength radiation and emits radiation at lower energy or longer wavelength [64]. It comprises three stages: fluorophore excitation (femtoseconds scale), vibrational relaxation to a lower energy state (picoseconds scale) and light emission (nanoseconds scale) [65].

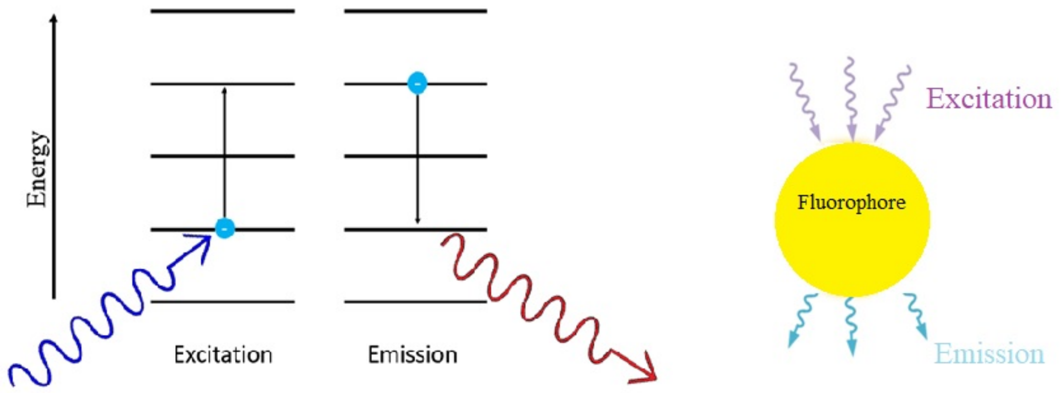


Figure 2.1: On the left: Jablonski diagram. On the right: representative illustration of energy conversion process in fluorescence

The excitation energy is adsorbed by the molecule and excites an electron to a higher energy state. The fluorescence light emission occurs when the electron falls back to the pre-excitation energy level, emitting a photon of higher wavelength than the exciting photon [64]. The fluorophores will emit light only if a specific wavelength radiation excites electrons and specific wavelength will be reemitted [66]. The difference between the excitation wavelength (E_x) and the emission wavelength (E_m) is called *Stoke's shift* [67] and allow to distinguish the two signals.

$$Stoke's\ shift = E_m - E_x \tag{2.1}$$

The magnitude of the reemitted light is directly proportional to the concentration of a fluorophore in a solution [66], and it is also proportional to the intensity of the excitation light [68]

2.1 Intrinsic Fluorescence

The sensor, which will be described in the next chapters, uses a natural fluorescence technique called intrinsic fluorescence. This method is based on the light emitted by the microorganisms when exposed to deep targeted UV light. It is label free, so it is cheap because there is no need for extra reagents [69]. Intrinsic fluorescence has been applied in protein analysis and other molecular studies. Living bacteria contain biological molecules that exhibit fluorescent characteristics. These molecules include tryptophan, tyrosine, phenylalanine, nucleic acids, flavins, cytochromes [70].

Tryptophan is the molecule that most contribute to protein fluorescence [70]: it is found richly in transmembrane proteins playing an essential role in anchoring and stabilising proteins [71]. Supposing the amino acids, and especially tryptophan, are present in normal percentages in a cell membrane, they may give a size of the totality of cells in a solution [72]. Furthermore, tryptophan has an higher quantum yield (defined as "the efficiency of the fluorescence process" [73]) and stoke's shift (70 nm) compared to other molecules, like tyrosine and phenylalanine, which make tryptophan a good selection for the sensor [74, 75]. It has an excitation wavelength of 280 nm and an emission wavelength of 350 nm in its natural state, but when present in more complex samples or even in cells, it may change its characteristics due to the fact tryptophan is solvent dependant [76].

Intrinsic fluorescence shows great potential for detection of bacteria. Kilungo et al. [77] proved a detection limit of 50 bacteria/L for living *Bacillus thuringiensis* in water by measuring intrinsic fluorescence at E_x 365 nm/ E_m 440 nm. Giana et al. [78] classified *Escherichia Coli, Enterococcus feacalis* and *Staphylococcus aureus* based on their intrinsic fluorescence. But both of them operated under unrealistic conditions for real life applications: Kilungo et al. with one bacteria in deionized water, and Giana et al. with three bacteria in culture media. The urine may interfere the signal from the intrinsic fluorescence of the bacteria. Perinchery et al. explored the possibility to use autofluorescence of human urine to discrimate and diagnose UTI [79].

Chapter 3

Spectrometry analysis

In this chapter it has been described the materials and methods used in the spectrometry analysis conducted on the urine sample, to find the interferences in the tryptophan wavelength range starting from the theoretical values, since the amino acid is integrated in protein and in the pathogen *E. coli.* A comparison between the urine sample and water sample has been conducted.

3.1 Rayleigh and Raman scatterings

Rayleigh and Raman scatterings are the elastic and inelastic scatterings, or diffusions, of the light occurring when the light travels through a turbid medium, such as gas or fluids. Since the Rayleigh diffusion is elastic, the emitted radiation has the same frequency or wavelength of the incident light [80], whereas the Raman effect is responsible of the diffusion of the light at more or less energy (Stoke or Anti-Stoke) [81]. In this case it has been analyzed only the higher wavelength radiation, since the fluorescence light has always higher wavelength emission.

3.2 Analysis

For the spectrometry analysis was used a *FS5 Spectrofluoremeter* (Edinburgh Instruments, UK) (figure 3.1). As shown in the figure 3.2, the source and the detector form a 90º angle to avoid that the light from the source reach the detector, so most the radiation will come from the sample fluorescence.

The samples were placed inside a Suprasil quartz cuvette with a 10x10 mm pathlength, which should annul, or at least decrease any refractive, reflective or fluorescence light coming from the material which the cuvette is made on.

The excitation source ranged from 260 nm to 300 nm, while for the emission, wavelengths from 260 nm to 400 nm were analyzed. The spectroscopy technique used was Excitation Emission Matrix (EEM), which allows to summarize multiple

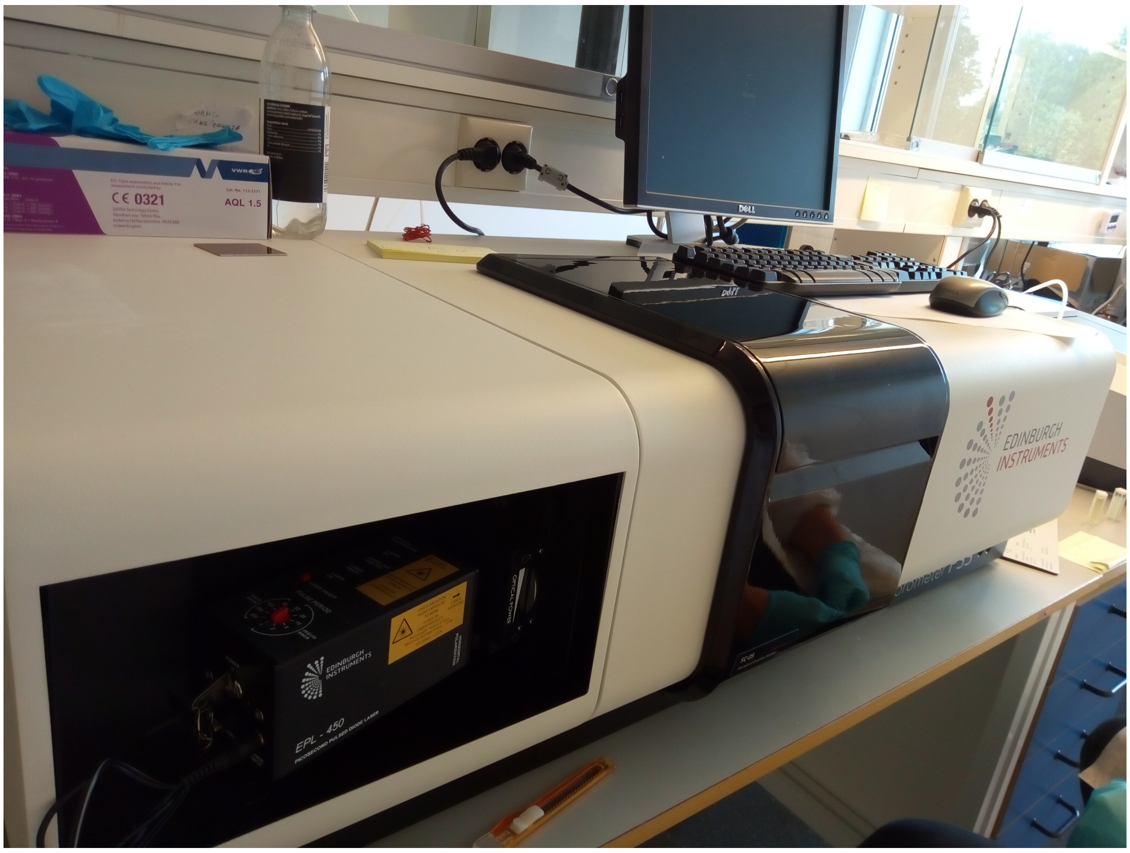


Figure 3.1: FS5 Spectrofluorometer

analysis in one graph to display all the spectrum features. The signal processing was done with the spectrofluorometer signal processing software.

The urine and water samples were at 25°C; for analyze signal from tryptophan, the pathogen was added up like it has been described in the chapter 5.

3.3 Results

3.3.1 Rayleigh and Raman scatterings

To visualize the effect of Rayleigh scattering two EEM scan were executed, one without the sample in the chamber and one with the sample, one time urine sample and one time water sample. The first scan was subtracted to the second and the Rayleigh scattering was visible. As shown in the figure 3.3 the Rayleigh scattering produce a red line on the EEM scan, which make evident that this effect happens at the same excitation emission wavelength.

The second blu/yellow parallel line represents the Raman effect, which shows

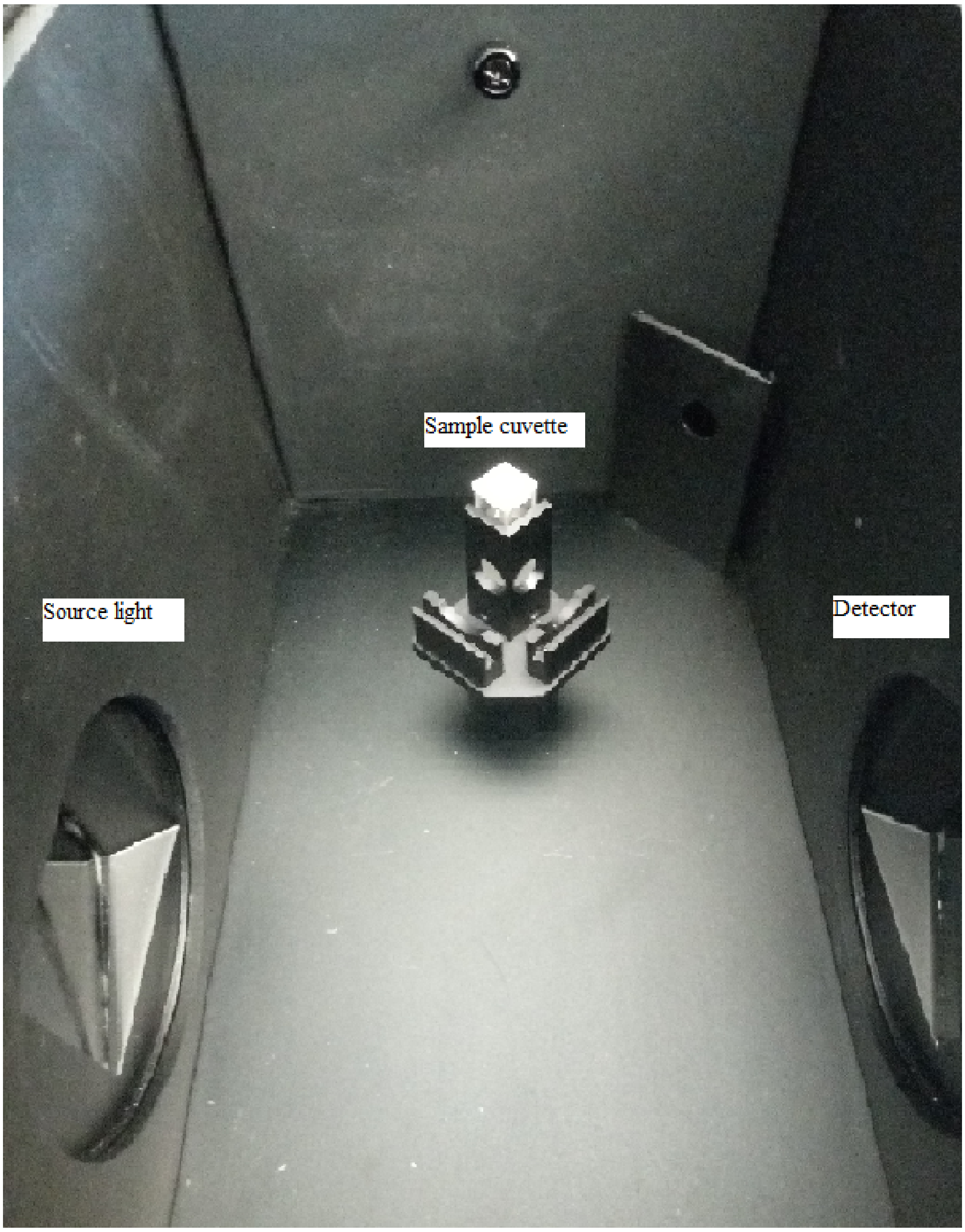


Figure 3.2: Interior chamber of the spectrofluorometer

that when radiation pass through the sample a fraction of the energy is lost and reemitted at higher wavelength.

For urine and water sample similar results were reached.

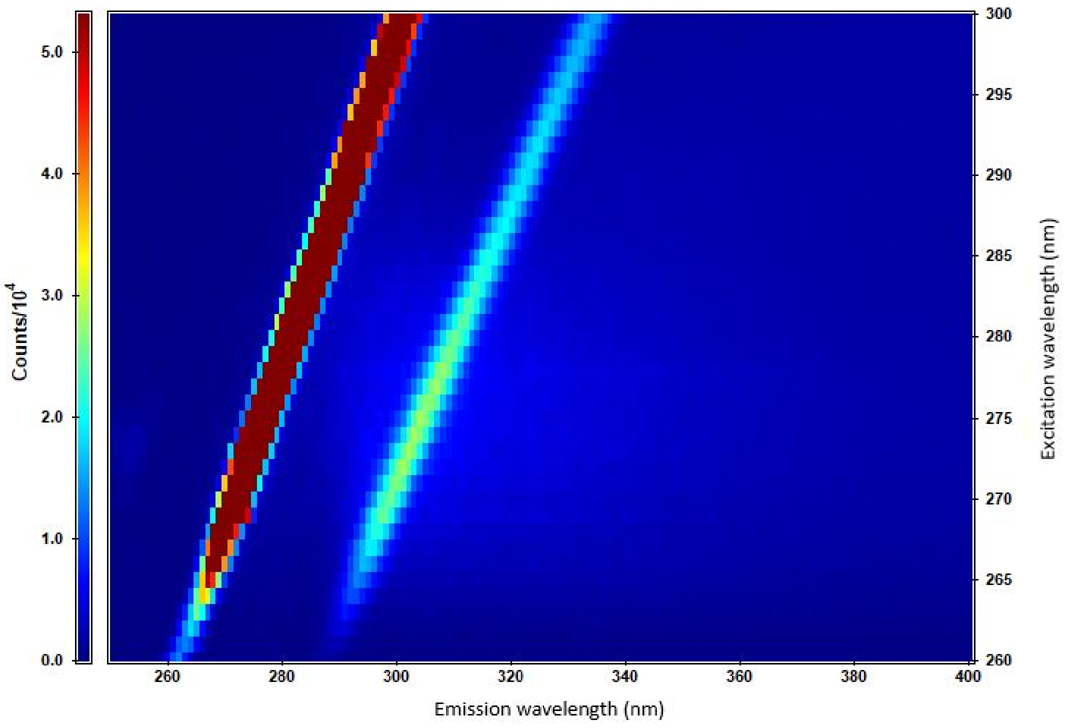


Figure 3.3: EEM scan. The red line represents the Rayleigh effect, while the blue/yellow line the Raman effect

3.3.2 Tryptophan

To obtain the tryptophan spectrum high concentration of *E. coli* was placed in urine and water samples. As we can see from the figure 3.4 the tryptophan fluorescence has high signal at 290-295 nm excitation wavelength and then higher emission at 320-340 nm. In the figure 3.5 it is showed the tryptophan spectrum but in water sample. The signal is a bit shifted from the previous one, presenting high signal at excitation 270-280 nm and emission at 320-340. These results are different from tryptophan theoretical emission peak. This is probably due to the way the tryptophan is folded within the pathogen and the two different fluids with specific polarity and optical properties may influence the spectrum. Further considerations have not been done due to the lack of time and because the focus was the sensor characterization for water monitoring.

However, the analysis proved that it is possible to measure the tryptophan signal inside the *E. coli* dissolved in urine and water, since it is possible to discriminate the signal from the interferences. To decrease the Rayleigh and Raman effect, which creates more interference because it may overlap the signal from the fluorescence, it should be used an optic filter, as it has been done. More considerations will be clarified in the chapter 4.

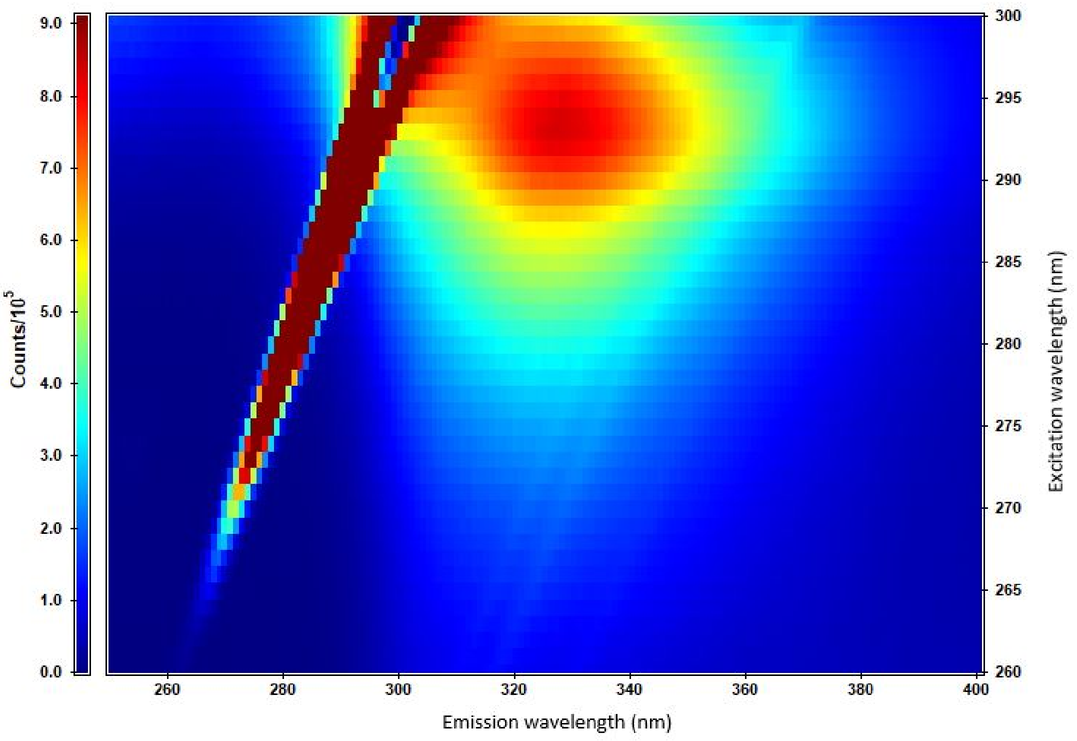


Figure 3.4: EEM scan with E. coli in urine sample. The red line is the signal from the Rayleigh effect, while the other contributions is the signal from the tryptophan

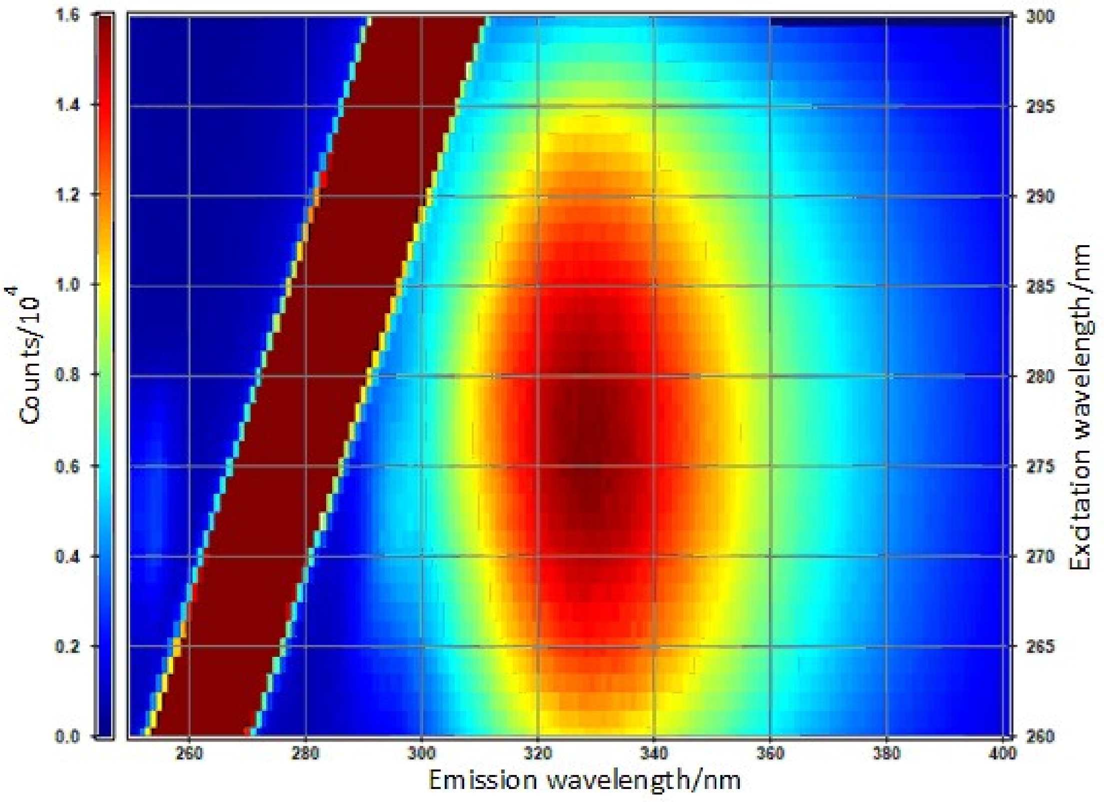


Figure 3.5: EEM scan with E. coli in water sample. The red line is the signal from the Rayleigh effect, while the other contributions is the signal from the tryptophan. In this case the highest signal is at 270-280 nm emission wavelength. Adapted from [82]

Chapter 4

Optofluidic sensor

In this chapter the main components of the optofluidic sensor have been presented. The electronic and optical parts, such as the design have been developed by other students. Possible modifications to the system will be suggested and discussed in the next chapters.

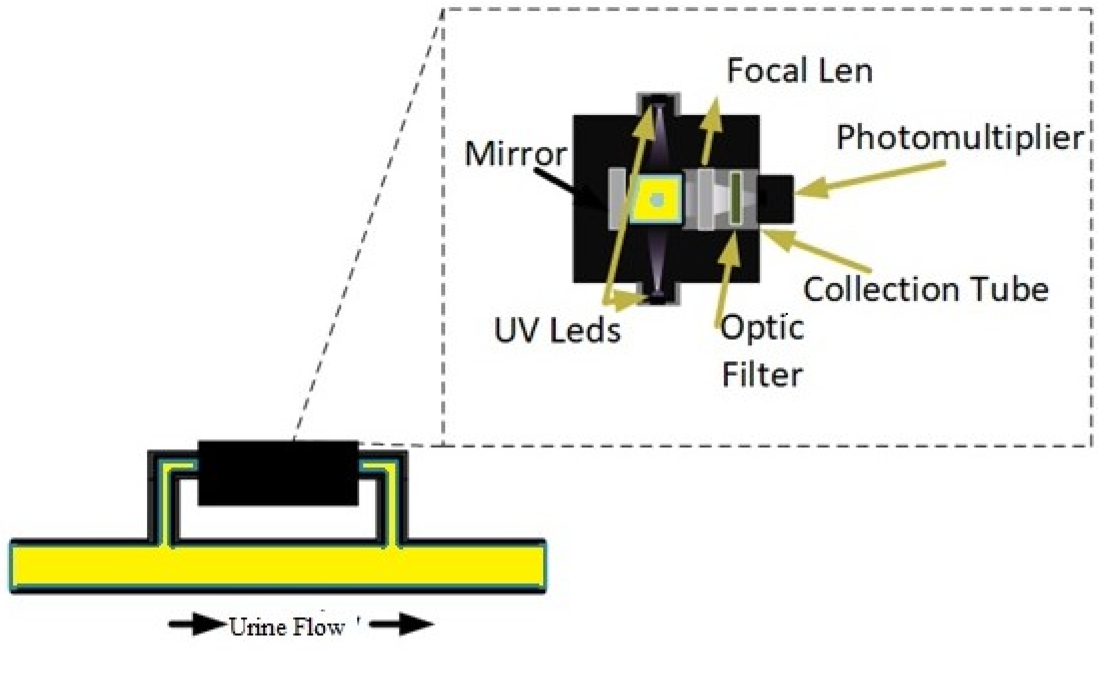


Figure 4.1: Schematic illustration of the optofluidic sensor

The figure 4.1 shows all the components of the optofluidic sensor proposed as possible diagnostic tool for detecting urinary tract infection. Following the indications given by the spectrometry analysis the master student João Carlos Gomes Simões proposed the use of the optical parts needed to obtain the signal from the tryptophan fluorescence, and then, with the help of the bachelor students Fredrik Sommerfelt Grønvold and Helene Sundt Berntsen the optimal design to reach the lowest detection limit, and the necessary power supply were developed.

4.1 Deep UV LED and optic filter

To trigger the fluorescence from tryptophan a centred light source is required. The sensor makes use of two deep UV Light Emitting Diodes (LEDs) with an emission peak wavelength at 280 nm (emission band: 275-285 nm). The LEDs were disposed at 90°C from the detector to increase the signal to noise ratio [77], with a small viewing angle (6°C) to reduce the dispersion of the light before the radiation gets to the sample and in the same plane of the detector and the sample to optimize the light going from the source to the sample and from the sample to the detector. The LEDs require 20 mA as forward current and 7 V as voltage.

The optic filter is a bandpass filter with a bandwidthincluded between 330 nm and 350 nm and 90% of peak transmission at 340 nm and an optical density of 6. The bandwidthwas chosen to minimize the Rayleigh and Raman effects, so the light reaching the detector is fluorescence light.

4.2 Photomultiplier tube

The detector is a photomultiplier tube (PMT). It has an analog voltage output range from 0 to 4.5 V, proportional to the number of photons that enter in the 10 mm diameter window. It needs an input voltage of ± 5 Volt and has an adjustable gain between 0.6 V and 1.1 V. The maximum sensitivity is around 400 nm, but it is possible to achieve good performances at 330-350 nm wavelength.

4.3 Focal lens

The focal lens represents one of the most important component of the entire system. Since the PMT has an effective area with a 10 mm diameter, the light need to be focus on this surface to collect the maximum number of photons from the sample. The lens is a Plano-Convex lens and for this reason it may be considered an infinite/finite conjugate system [83].

The equation that describes the system is the following:

$$f = \frac{1}{2 * NA} = \frac{F}{D} \tag{4.1}$$

where f is the lens ability to focus, NA in the lens numerical aperture, F is the focal point distance and D the lens diameter. Because the light has to be focus not in a single point but on the PMT area, the optimal distance was calculated following this equation:

$$\frac{F_e}{D_e} = \frac{F}{D} \tag{4.2}$$

resulting in a value of 15 mm. All the lens characteristics were taken from the manufacturer.

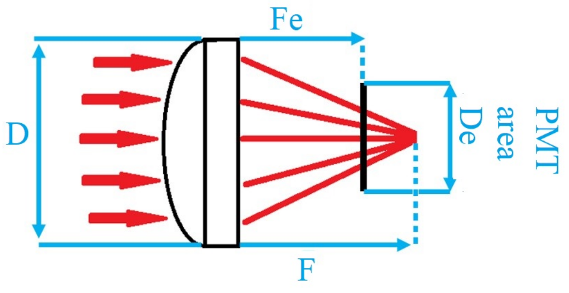


Figure 4.2: Focal lens illustration

4.4 Collection tube and Mirror

The fluorescence light does not have a favourite direction resulting in radiation dispersion. To increase the light reaching the PMT an aluminium mirror and tube are inserted in the sensor, preventing the absorption from the sensor material and incrementing the detector signal output.

4.5 Design

The design was drawn with the help of a CAD software and then printed using PLA (Polylactide Acid) heated at 200°C in a 3D printer. It consists in several parts to facilitate the assembly of the system in all its components.

4.6 Offline and online mode

The system can operate in two different ways. The first is the offline mode where samples with known contamination level are put in a quartz cuvette and then allocated in the sensor. The second is the online mode where the fluid is pumped in the sensor thanks a second quartz cuvette with an inlet and outlet (figure 4.4).

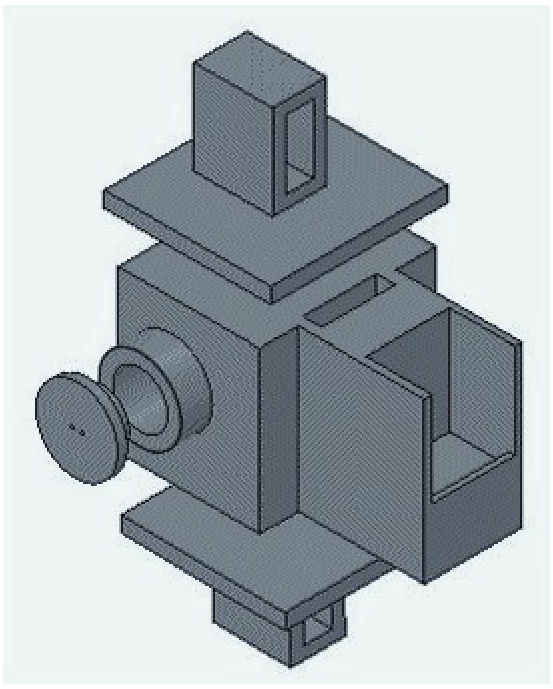


Figure 4.3: Sensor design

The online setup produces a signal ten times higher than the offline, probably due to the different materials and geometry of the two cuvettes.

For urine tests only offline tests were performed.

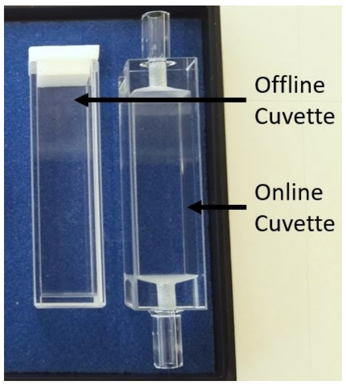


Figure 4.4: Online and Offline cuvette

4.7 Power source

A lithium battery was used to supply the necessary power to all the components of the sensor, included the communication system described in the chapter 6, and to make the system portable. Additionally, a printed circuit board (PCB) was developed to power up the PMT, the LEDs and the Arduino used in the communication system described in the chapter 6

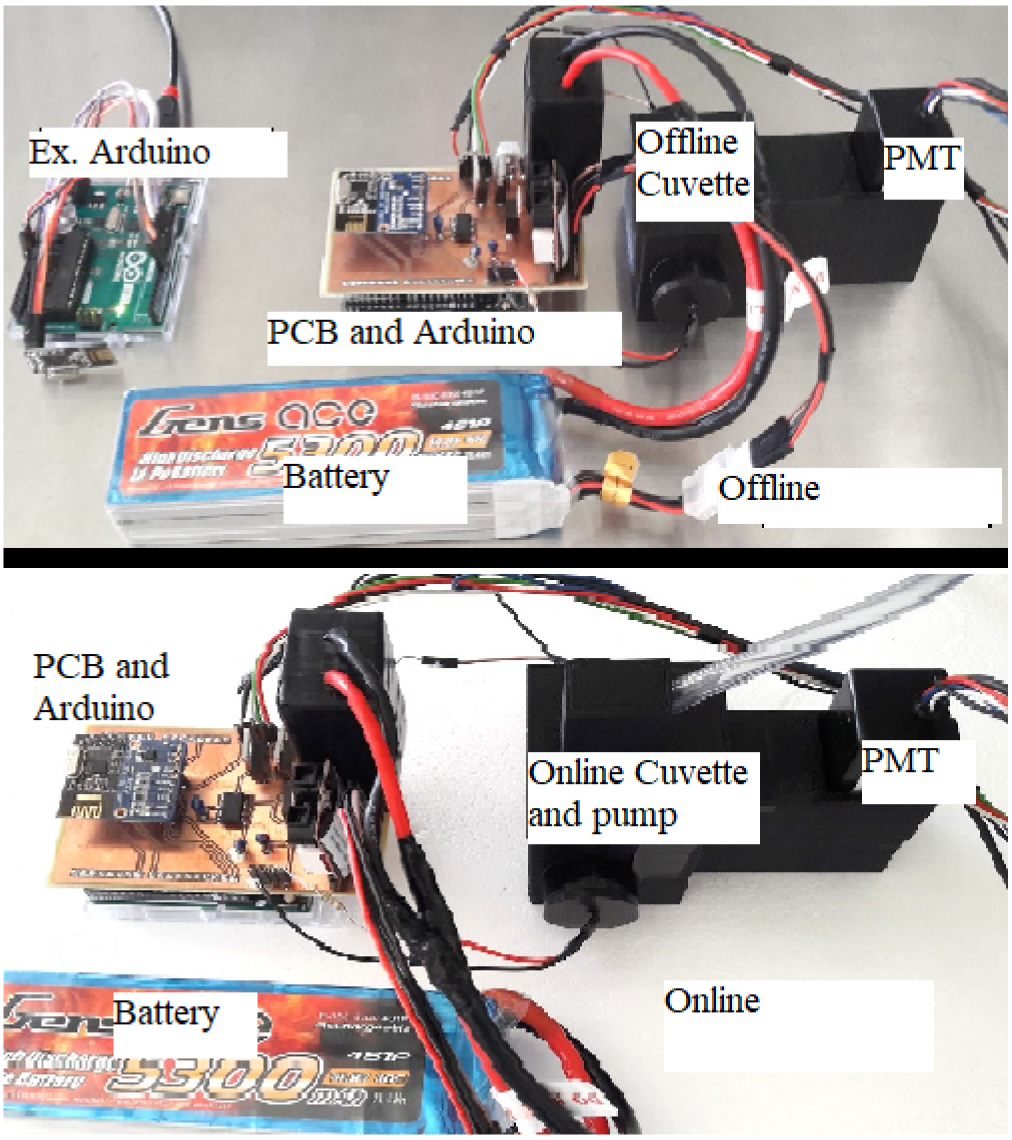


Figure 4.5: Offline and Online setup. Adapted from [82]

Chapter 5

Samples preparation procedure

This chapter describes the materials and the methods used in the samples preparation for the tests in water and urine. The *Escherichia coli* was grown and added in different concentrations to examine the sensor behaviour. Major focus will be given on the urine test. All the steps have been performed in the University of South-Eastern Norway BioLab.

5.1 Bacteria culture and counting

During the tests it was used *Escherichia coli* to reproduce the pathogens, since it is the most common contaminant agent in UTI. To ensure the maximum rate of live to dead *E. coli* cells, it was made a cell culture to the day before. The culture was done on Luria-Bertani broth (LB) solution at 120 rpm and 36ºC (figure 5.1) [84]. After 20h the cell culture was taken away from the incubation chamber. The cell counting was based on a Bürker counting chamber (figure 5.2) [85], and the counting was done with the microscope; to allow a better visualization of the cells it was used food dye.

For the cell count it was used the equation:

$$Particle\ per\ volume = \frac{Counted\ particles}{Counted\ surface * Chamber\ depth * Dilution} \tag{5.1}$$

In the figure 5.2 the squares have a 0.2 mm side and the depth under the glass was 0.1 mm, resulting in a volume of 0.004 $mm^3 = 0.004\ \mu L$. The bacteria were counted in at least 10 squares.

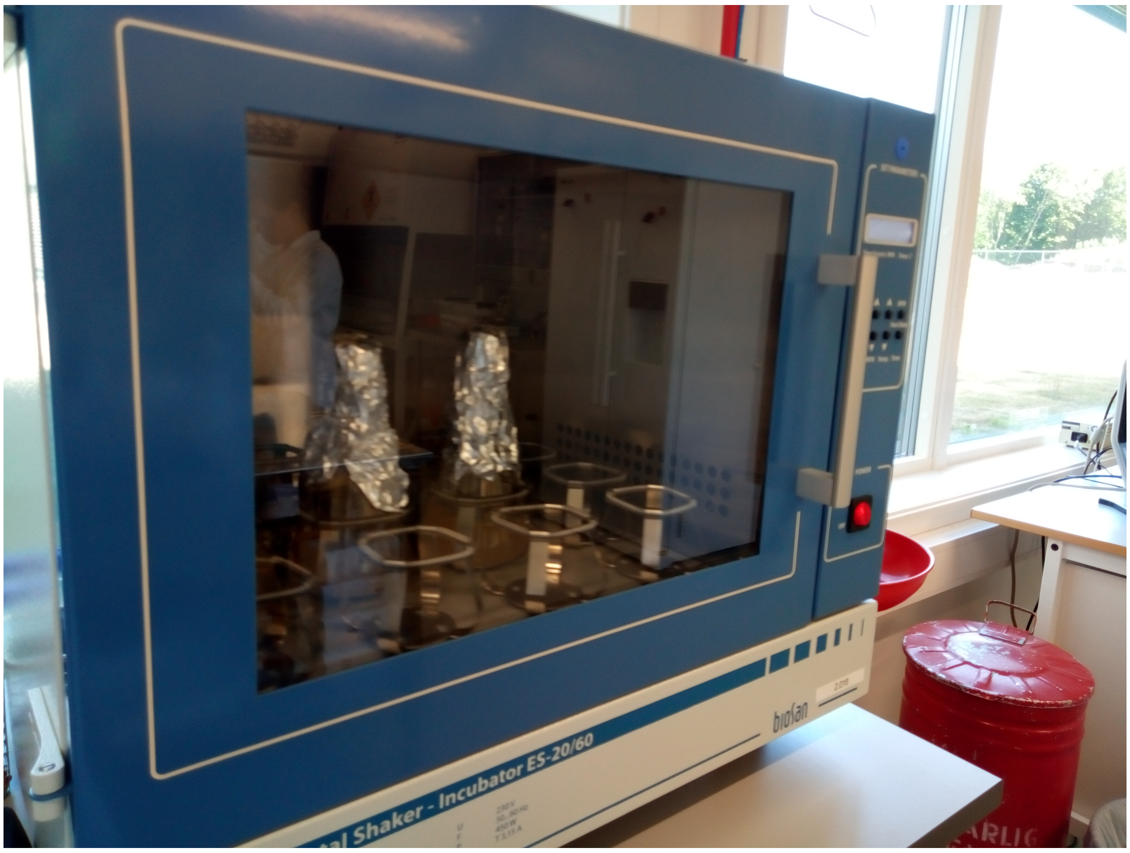


Figure 5.1: BioLab incubator with cell culture

5.2 Dilution series

First, the LB medium was removed through centrifugation at 3500 rpm for 3 minutes (figure 5.3) [86] from a 50 mL sample. The bacteria gathered at the bottom of the flask and medium flowed away. The samples were washed in phosphate buffered solution (PBS) and centrifugated again for 3 minutes; these steps were repeated 3 times to ensure all the medium was removed. The PBS was poured out too and the bacteria were added to 5 mL of urine. The solution was stirred with a BioLab shaker and starting from here the dilution series was made. $500\mu L$ of this first bacteria solution was taken with a pipette and added to a $4500\mu L$ of urine, diluting the solution 1:10. The container was stirred. Then, $500\mu L$ of 1:10 solution was extracted and added to $4500\mu L$ of urine in a new container creating a 1:100 solution. Similar steps are repeated to create 1:500, 1:1000, 1:5000 and 1:10000 solutions (figures 5.4 and 5.5) In the same way described above, water sample from 1:10 to 1:1000000 solution have been created.

The urine used for the tests was a non biological urine which replicates human urine features [87] (figure 5.6).

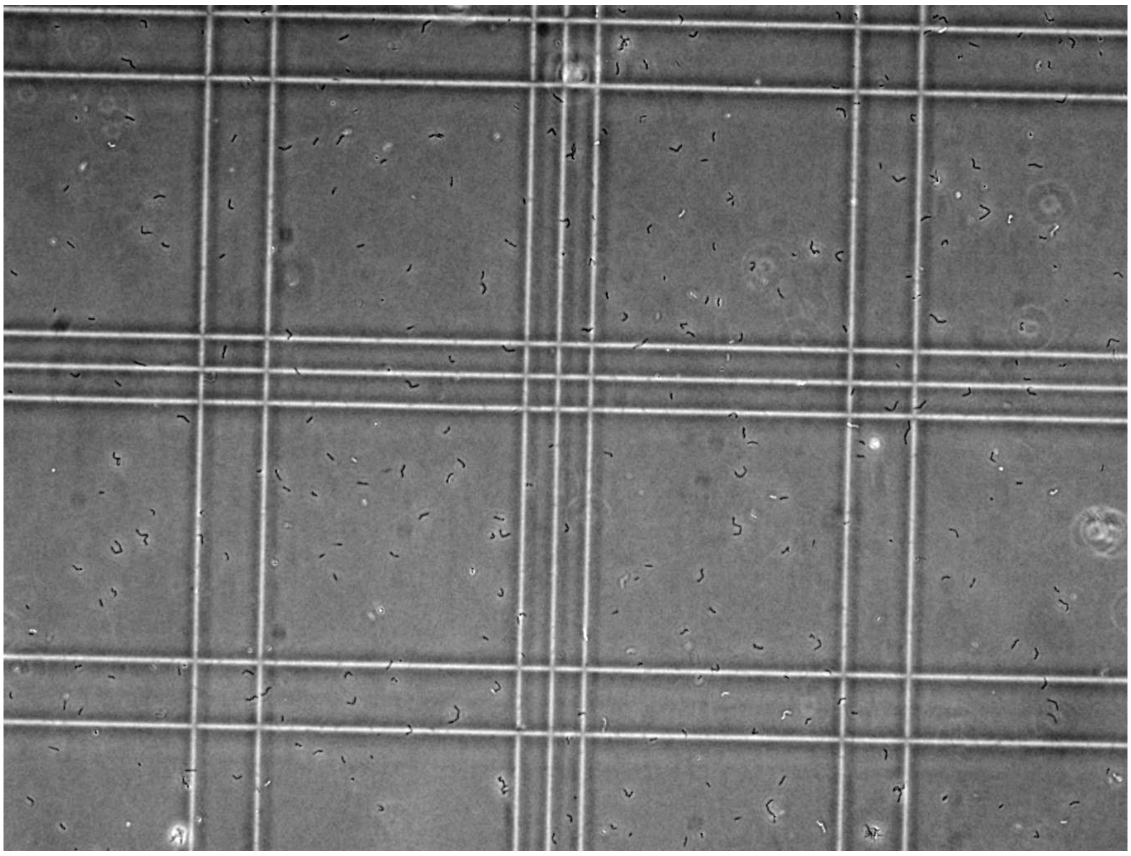


Figure 5.2: E. coli in a Bürker counting chamber

Figure 5.3: Bacteria with LB medium in the BioLab Centrifuge

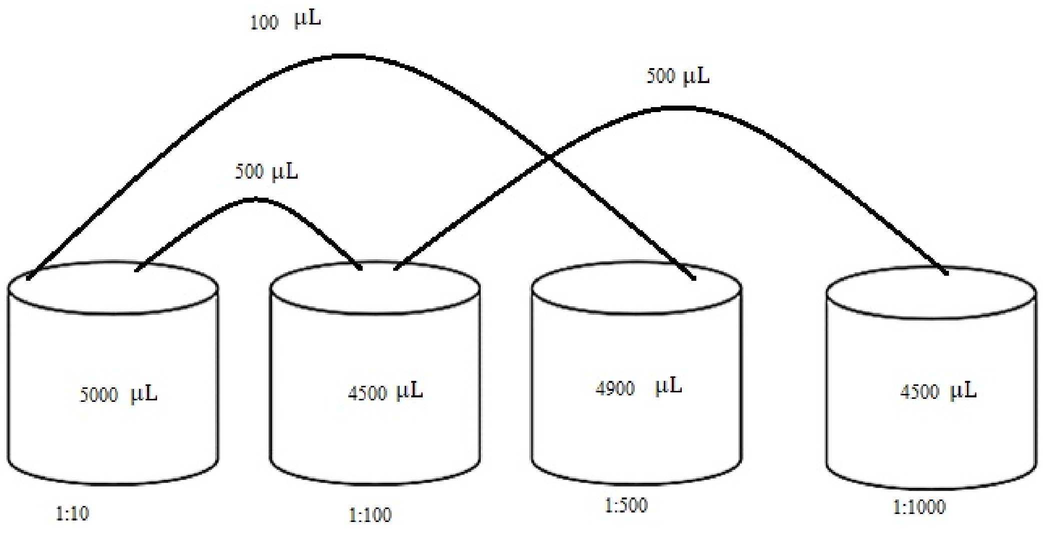


Figure 5.4: The dilution series was made moving liquids from one container to another

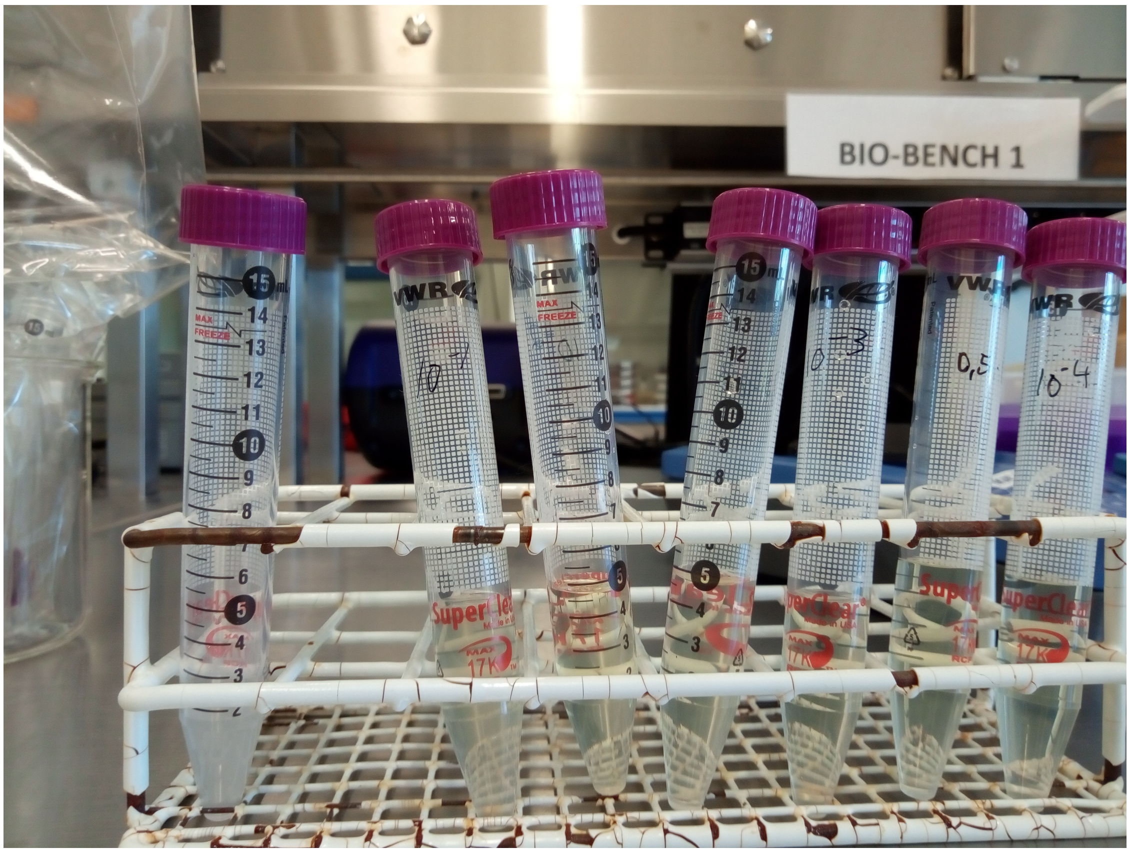


Figure 5.5: Containers with the dilution series used for the urine test

Cerilliant
S-020-50ML Lot: FN08311702
Surine™ Negative Urine Control
50 mL
Expires: See COA
Refrigerate (Do Not Freeze)
Not a dangerous substance or mixture according to the Globally Harmonised System (GHS).
Product of USA
For Analytical or R&D Use Only

Figure 5.6: Surine Negative Control Urine

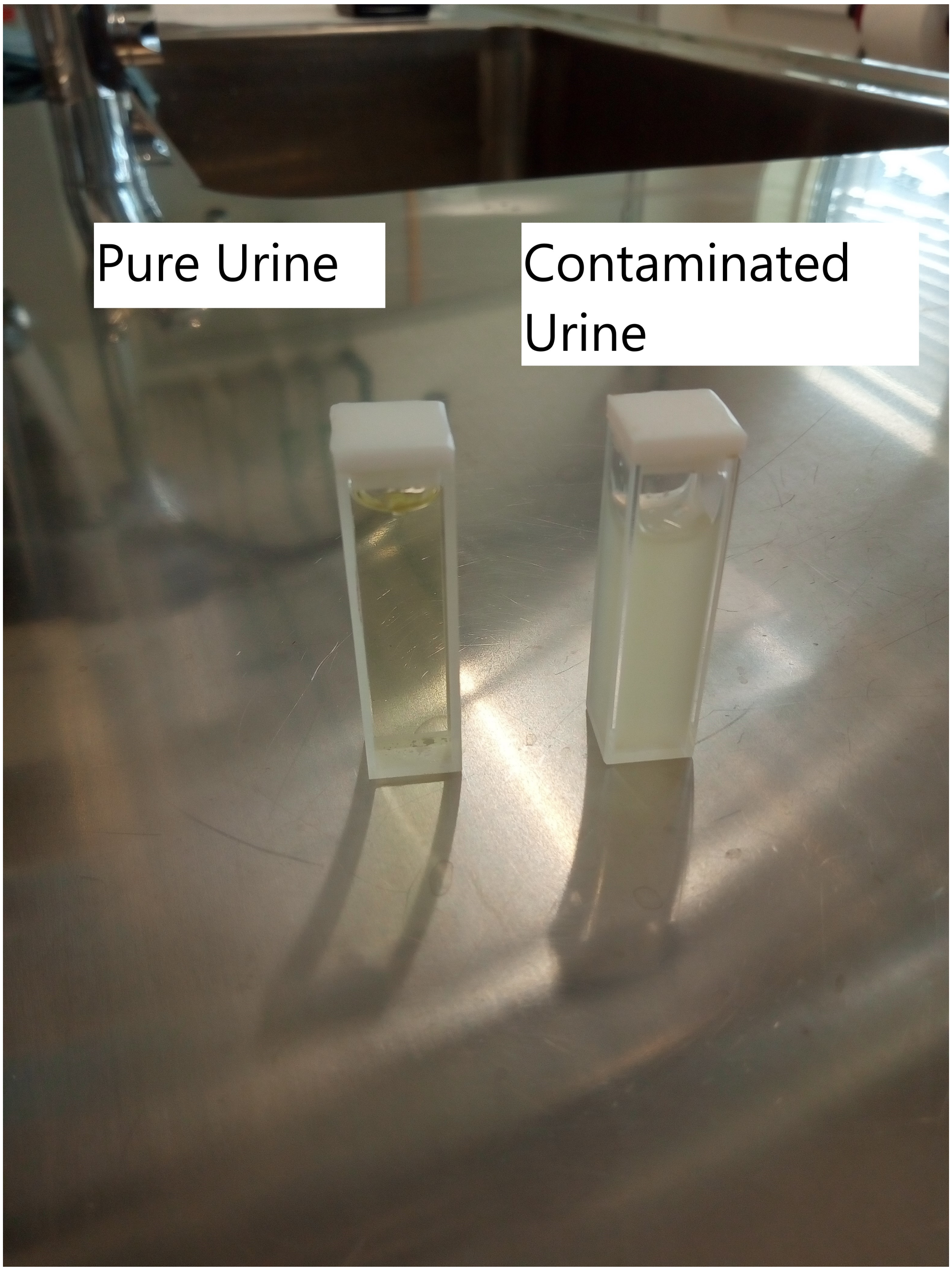


Figure 5.7: A sample of pure urine and a sample of contaminated urine

Chapter 6

Signal Readout: First version

In this chapter are described the materials and the methods used in the first communication system and the relative version of MATLAB interface is presented.

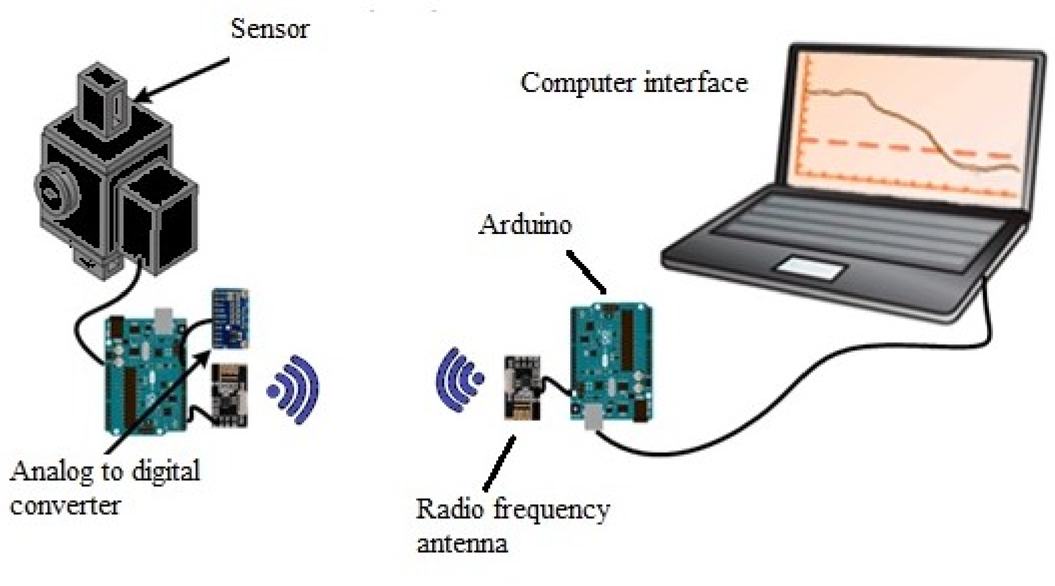


Figure 6.1: Schematic representation of the electronic components used for the radio frequency communication

An external 16-bits analog to digital converter (ADC), called ADS1115, was used to convert the signal from the PMT with high precision. Two radio frequency antenna (NRF24L01 Transceiver) communicate the digital value from the first Arduino Uno R3 to the second one at 1 Hz frequency (every second a new value is transmitted). A first version of a MATLAB Graphical User Interface receives, records and displays the signal in real time.

6.1 Signal conversion and wireless transmission

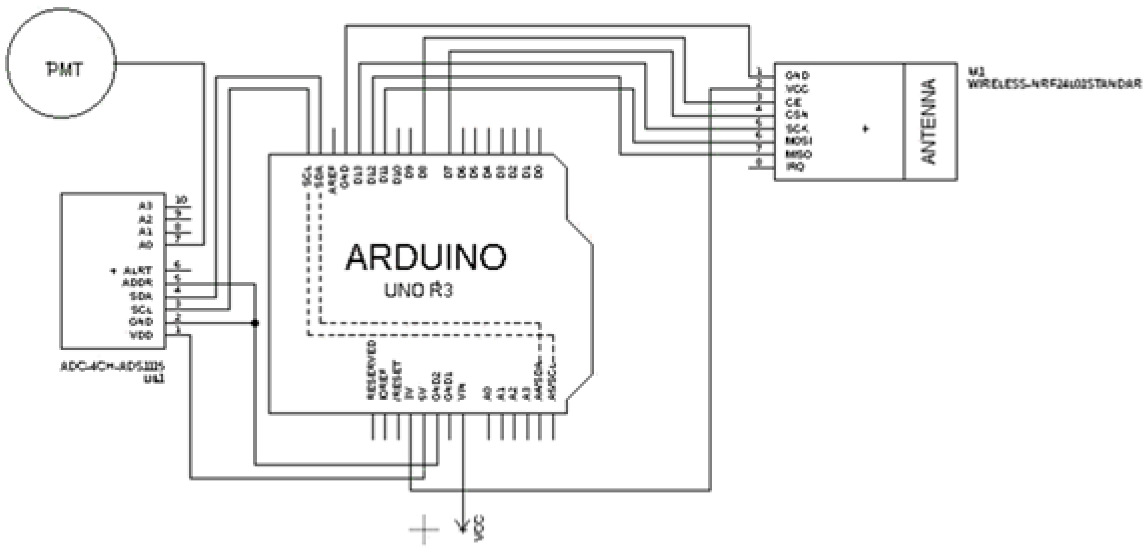


Figure 6.2: The picture shows the connections between the PMT output signal, Arduino Uno, ADS1115 and NRF24L01

6.1.1 Signal conversion

The pin A0 of the ADS1115 16-bit ADC gathers the signal output from the PMT. An ADC with 16-bit precision was used to not lose resolution. But the result of the conversion is a signed integer. This means that the first bit of is used to communicate the sign of the read value. Hence, the actual value is codified on 15 bits, the voltage resolution is, then, calculated from:

$$\frac{FSR}{2^{Nbit}} = \frac{6.144V}{2^{15}} = 187.5\mu V \tag{6.1}$$

The ADS1115 can complete conversions at rates up to 860 samples per second. An onboard programmable gain amplifier (PGA) is available on this ADC that presents input ranges from the supply to as low as ±256 mV. The maximum supply voltage is 5.5 V and the maximum supply current operating at 25°C is 200 µA. The ADC can operate in 2 different ways: single-shot mode and continuous mode. When the device is programmed in single-shot mode, it accomplishes a conversion of the input signal only when a master device connected to the ADC requires it and automatically powers down after a conversion reducing current consumption. On the other hand, in the continuous conversion mode a conversion of the input is performed as soon as the previous one is done with a rate equal to the programmed data rate.

The Arduino 5 V pin (max current supplied 500 mA) is used to power up the ADC and the gain of the amplifier has been programmed to 2/3 (Full-Scale Range=±6.144 V) allowing to cover the PMT output range from 0 to 4.5 V. The ADS1115 communicates with the master device through an I2C interface. The master device provides the clock signal on the SCL pin and the data are transferred via the SDA pin. The ADS1115 has an address pin, ADDR, that sets the I2C address and can be connected to 4 different pins allowing four addresses to be selected. The ADDR has been connected to the GND pin and the corresponding address is 0x48.

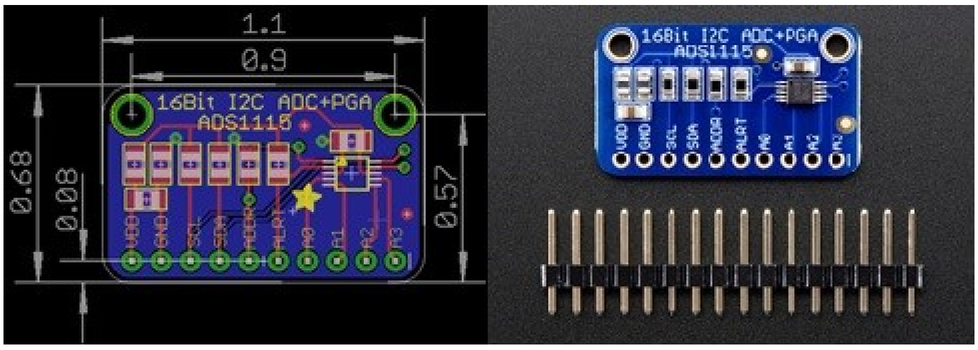


Figure 6.3: ADS1115 and its dimensions (cm). Adapted from `https://learn.adafruit.com/adafruit-4-channel-adc-breakouts/downloads`

6.1.2 Wireless transmission

The converted value is transmitted and then received by two radio frequency antennae called NRF24L01, a bi-directional radio transceiver. The maximum supply voltage allowed is 3,6 V. They drain very low current: 11,3 mA in TX (transmission mode) and 12,3 mA in RX (receiver mode). The operating frequency is 2,4 GHz and the transmission data rate may achieve 2 Mbits for a second. Both the antennae realize a SPI communication system with Arduino, denoted as the master device. The first has been instructed to convert and transfer the sensor output wirelessly to the second one, programmed to collect the data and to pass them serially to the USB port of the PC. The frequency of transmission coincides with the sampling frequency of the signal. The connection has a 100 m range.

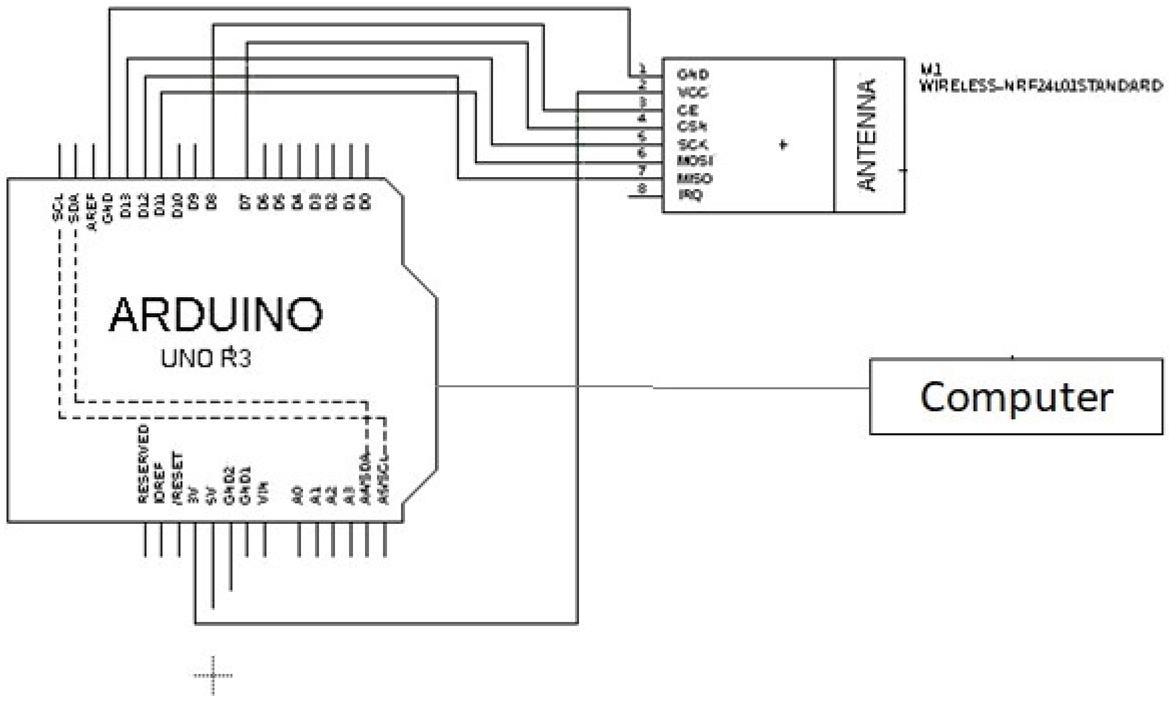


Figure 6.4: Connections between the second Arduino Uno and NFR24L01, and computer

6.2 MATLAB interface: first version

The MATLAB interfaces were optimized for the BiWAS project to offer a continuous and real time monitoring of the signal from the water distribution network. Some characteristics, as for example the need to have a time threshold to trigger the alarm is to avoid an alarm due to a transient signal that it is not a contamination event.

6.2.1 Real time signal reader

The figure 6.5 shows the first window of the first MATLAB interface version. It gives the possibility to impose a concentration threshold value (CFU/ml) and a time threshold (s) which on the alarm system is based. Once imposed these values the "Read signal" button is enabled and the signal from the serial port is imported and visualized in a 30 seconds wide graph, updated with a frequency of 1 Hz. The bacteria concentration, attached with the date (second resolution), the two thresholds and a coding value (0 if the signal shows a concentration under the threshold imposed, 1 otherwise) is saved in .txt file.

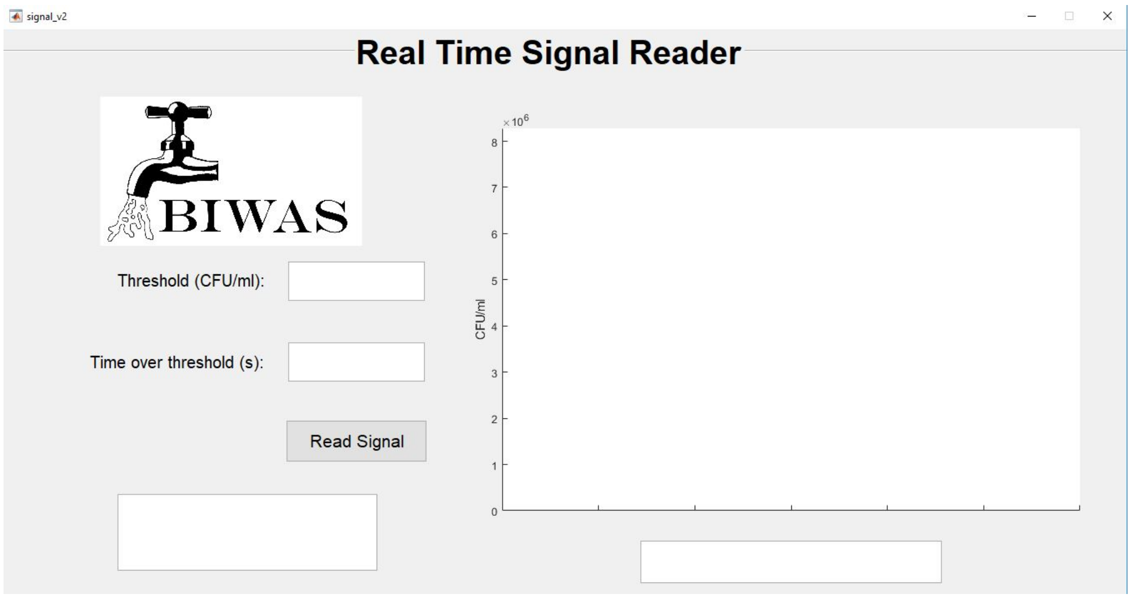


Figure 6.5: Real time signal reader window in the first interface version

If the signal overcome the threshold value for the time imposed two different kind of alarm are triggered. A local one that consists in a “DANGER” message visualized in a text box on the interface combined with a sound, and a push notification on a device. To achieve the second alarm system, it was used an app called Pushbullet, that allows to receive push notifications on the device on which it is installed.

Inside the code: Real time signal reader first version

At first the characteristics of the graph are declared, like x and y axis limits, x and y labels, window width; it has been created a serial object and then opened to access to the signal with a Baud Rate of 115200 (maximum for a USB 3.0 port); the mobile notification is set linking it to the Pushbullet account. For the alarm system it is necessary an initialization. First, an app called PushBullet (available in iOS and Android device) has been downloaded and installed on a mobile phone. An account has been signed and linked to the app. A MATLAB function available on Github [88] has been used to send the push notification on the mobile device. Once the account has been setup the access token has been specified in the notifier object initialization.

The user inputs are checked to avoid that wrong values are typed and compromise the system alarm. The concentration threshold value should be a number from 0 to $8 * 10^6$ CFU/ml (maximum bacteria concentration detectable corresponding to a voltage output of 4.5 V) and the time should be a positive number. If these conditions are not respected the reading is disabled. Otherwise, pressing the ‘Read signal’ button an infinite loop is performed: the concentration is read, compared with the threshold value imposed, saved in the .txt file with the current date and the proper code value attached (the date is given by the machine), and then plotted in the graph; a counter is incremented every time that the signal is over threshold; when this value is bigger than the product between the time imposed and the frequency sampling the two alarm system described above are triggered.

$$N = fs * time \tag{6.2}$$

Where fs is the frequency sampling and N is the number of samples over threshold.

6.2.2 Past data visualizer

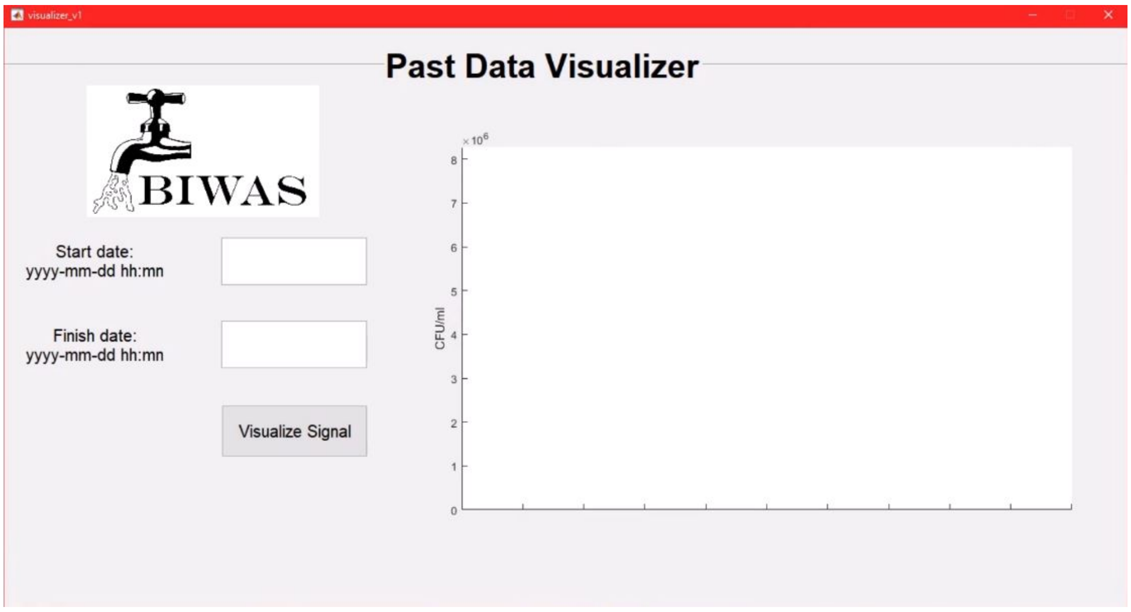


Figure 6.6: Past data visualizer window in the first interface version

The Past data visualizer window allows the user to visualize and control the data collected in a precise time interval to check if contaminations are occurred.

Inside the code: Past data visualizer first version

The user has to type the date in the specified format, as shown in the figure 6.6, or the window is disabled, and the visualization is not performed. The .txt file is open and read. Every line containing the data and the relative threshold collected between the start and the finish are taken and plotted in the graph. The x axis will show how many seconds of signal are displayed in order to check when the contamination occurred and how long it lasted.

Chapter 7

Signal Readout: Second version

In this chapter are described the materials and the methods used in the second communication system and the relative version of MATLAB interface is presented.

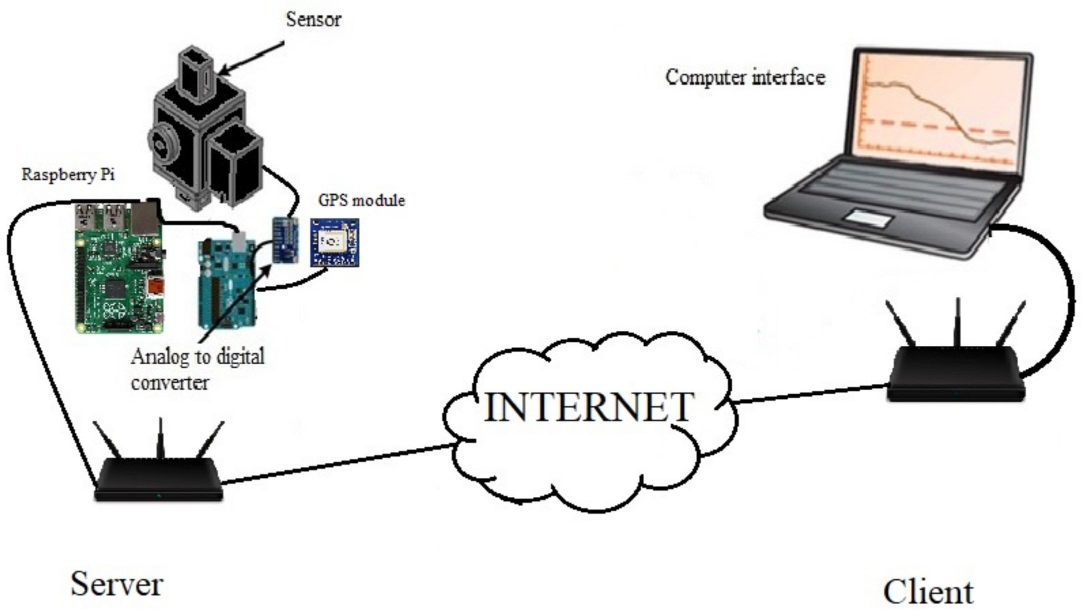


Figure 7.1: Schematic representation of the electronic components used for the server-client communication

The main purpose of this second communication system was to allow the access to sensor output from any location far from it and a first attempt to build a network, able to read from different sensors spread in a certain area, was achieved. In addition to the two windows presented in the previous chapter, the second version of MATLAB interface has a third window which guarantee a multiple reading and a GPS module has been integrated to each sensor to communicate the geographical position.

The remote communication uses a Raspberry Pi as server which collects, stores the data from the sensor and makes them available on the web.

7.1 PMT output and GPS signal acquisition

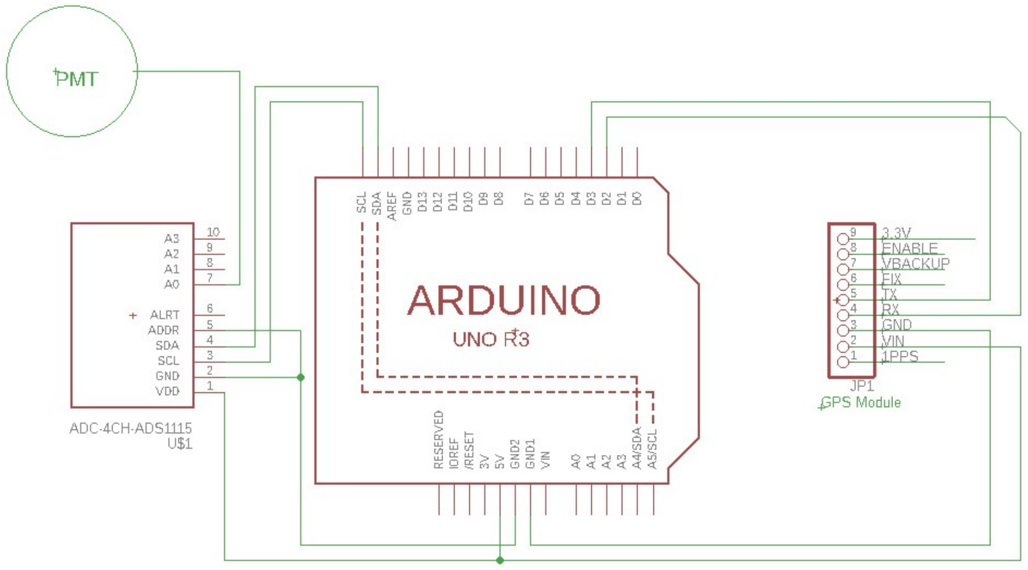


Figure 7.2: Connections between the PMT, Arduino Uno, ADS1115 and the GPS module

7.1.1 Signal conversion

The signal conversion is achieved with the same features described in the chapter 6.1.1.

7.1.2 GPS signal acquisition

The GPS signal is provided by ADAFRUIT ULTIMATE GPS. This board is a high-quality GPS module, that can track up to 22 satellites on 66 channels, with a high sensitive receiver (-165 dB tracking) and a built-in antenna. The power consumption is very low: it requires 20 mA during navigation ad 25 mA during tracking. Theoretically it has a position accuracy of 1.8 meters, and an update rate from 1 to 10 Hz. As shown in the figure 7.2 the module is supplied by the Arduino 5 V pin, the GND pin is connected to the Arduino ground pin, the RX a TX pins respectively to the digital pins 2 and 3. The RX pin receives the information from Arduino, whereas the TX sends the data collected by the GPS module.

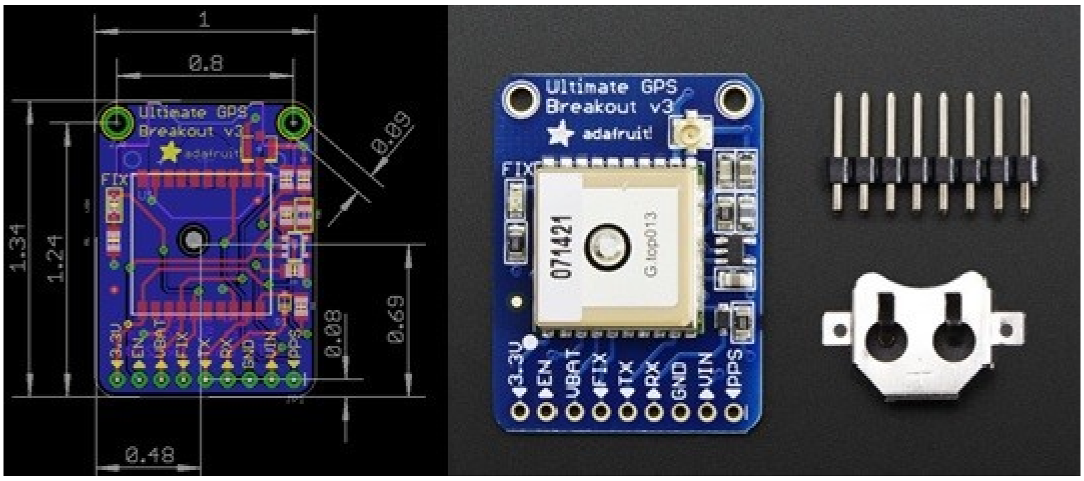


Figure 7.3: Adafruit Ultimate GPS breakout v3 and its dimensions (cm). Adapted from `https://www.adafruit.com/product/746`

7.2 Arduino programming

Arduino has been programmed using ARDUINO IDE. The code is in attachment. At first some libraries have been included to allow to use specific functions suited for the two components connected to Arduino. The baud rate was imposed to 115200 to be able to read the amount of data collected without any delay or losing. The gain of the amplifier has been set to 2/3. The GPS update rate to 1 Hz and only RMC and GGA sentences have been requested to GPS module. RMC and GGA are part of NMEA format: “a combined electrical and data specification for communication between marine electronics such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments” [89]. An example of a GGA sentence is shown below, which provide the current Fix data.

"$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M„*47
Where:
GGA Global Positioning System Fix Data
123519 Fix taken at 12:35:19 UTC
4807.038,N Latitude 48 deg 07.038’ N
01131.000,E Longitude 11 deg 31.000’ E
1 Fix quality: 0 = invalid

1 = GPS fix (SPS)
2 = DGPS fix
3 = PPS fix
4 = Real Time Kinematic
5 = Float RTK
6 = estimated (dead reckoning) (2.3 feature)
7 = Manual input mode
8 = Simulation mode
08 Number of satellites being tracked
0.9 Horizontal dilution of position
545.4,M Altitude, Meters, above mean sea level
46.9,M Height of geoid (mean sea level) above WGS84 ellipsoid
(empty field) time in seconds since last DGPS update
(empty field) DGPS station ID number
47 the checksum data, always begins with *"

RMC sentence, The Recommended Minimum, looks like:
"$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A
Where:
RMC Recommended Minimum sentence C
123519 Fix taken at 12:35:19 UTC
A Status A=active or V=Void.
4807.038,N Latitude 48 deg 07.038' N
01131.000,E Longitude 11 deg 31.000' E
022.4 Speed over the ground in knots
084.4 Track angle in degrees True
230394 Date - 23rd of March 1994
003.1,W Magnetic Variation
6A The checksum data, always begins with *" [90]

The core of the program consists in an infinite loop repeated every second in which a single conversion for the ADC A0 pin is requested, a reconversion of the read value is operated considering the resolution of the AD converter, the GPS signal is stored and at the end the voltage value, the latitude and longitude (in degrees) are printed on the serial port of Arduino. The latitude and longitude in degrees allow to localize the geographichal position typing them in Google maps. Arduino is powered up through the USB cable connected to a serial port of Raspberry Pi, ensuring the data communication between the two devices too.

7.3 Web server

As declared above the purpose of this second communication system was to make the output sensor accessible from the web. This goal was achieved using a Raspberry Pi (RPI) acting as a physical server, collecting and storing the data coming from Arduino.

Raspberry Pi is a credit-card sized computer that costs only 39$, making the all system sustainable for reproducibility keeping the price and the dimension low [91]. Like any other computers, the Raspberry Pi also uses an operating system (OS), in this case a flavor of Linux, called Rasbian has been used. There are many different flavors, or distributions, of Linux that is possible to install onto the Raspberry Pi. There are even a few non-Linux OS options available out there. For this project Raspbian has been chosen since Linux is a great match for Raspberry Pi, keeping the price low and a lot of documentation has been found.

The processor is a 32 bit, 700 MHz System on a Chip, which is built on the ARM11 architecture. The model B, used in this case, has a 512 MB of RAM. No hard drive is present of the Raspberry Pi, everything is stored on a SD card, including the OS. For this project a 16 GB memory SD card at least is required. There are 4 USB port that can draw up 500 mA (maximum current required from Arduino is 200 mA). An Ethernet port is available and an integrated Wi-fi module allow the access to internet. The HDMI provides digital video and audio output. The power supply is provided by a microUSB connector, a cheap and easy to find solution and it should supply 5 ± 0.25 V. The Raspberry Pi will only draw as much as current it need, but no more than 2.5 A for the considered model B [91]

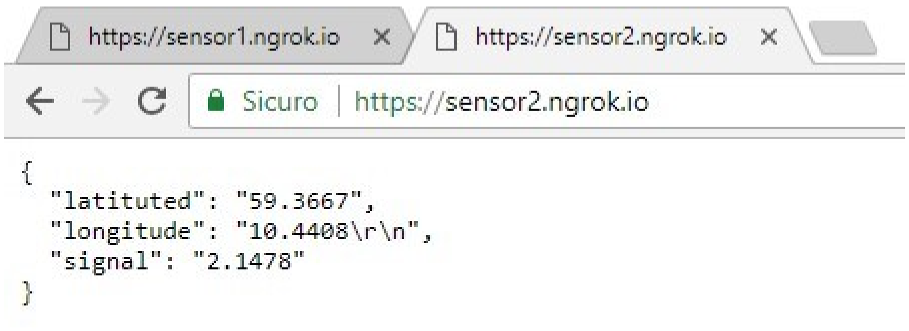
Another important reason why Raspberry Pi has been chosen for the declared purpose is that it comes preloaded with interpreters and compilers for many different programming languages. A Python script (in attachment) provides the web server building that dynamically generate the HTML when it receives the HTTP request from a web browser. A Python web framework called Flask [92] has been used to turn Raspberry Pi into a dynamic web server. To use Flask and the other libraries necessary for the right working a series of installation are needed using the terminal.

7.3.1 Inside the code: Web server building

After the libraries inclusion a Flask object called "app" and a dictionary object called "rect" were created. The USB port name and the BaudRate for the serial communication with Arduino were initialized.

“@app.route(’/’, methods=[’GET’])”: this line runs the code below every time someone accesses the root URL of the server, that consists in the serial input reading, splitting of the read line in 3 parts and including in the specific field (voltage value, latitude, longitude). The data is returned using jsonify to convert them to json format. The communication provided by Flask makes use of HTTP (Hypertext Transfer Protocol), an application protocol for distributed, collaborative, and hypermedia information systems [93]. The GET method has been used to send data in unencrypted format to the server.

Operating in this way the web server is running only locally on the port 5000 of the Raspberry Pi. A service called “ngrok” was used to host the web server on the public Internet. A business account has been created to take advantages of all the necessary characteristics of this service [94]. The figure 7.4 shows how the data are returned and posted on the web.



```
{
  "latituted": "59.3667",
  "longitude": "10.4408\r\n",
  "signal": "2.1478"
}
```

Figure 7.4: Web page collecting data from sensor 2

7.4 MATLAB interface: second version

7.4.1 Multiple reader

This window has been designed to read simultaneously two sensors. For each sensor is possible to set the threshold in CFU/ml and the time in s over threshold needed to trigger the alarm. In the last text box, the bacteria concentration is printed as soon as the data are acquired.

Inside the code: Multiple reader

Like in the first version of the MATLAB interface, specifically in the Real time visualizer, a check on the user inputs is performed for the same reasons described in the chapter 6.2.1.

Pressing the ‘START TO READ’ button an infinite loop is performed. The user inputs are constantly taken, to update potential changes in the threshold and time values and stored in two different .txt files. Using the ‘webread’ MATLAB function the data returned by the server in json format are stored in two different variables. This function automatically converts the json objects to structures that are convenient for analysis in MATLAB. The concentrations are compared to the

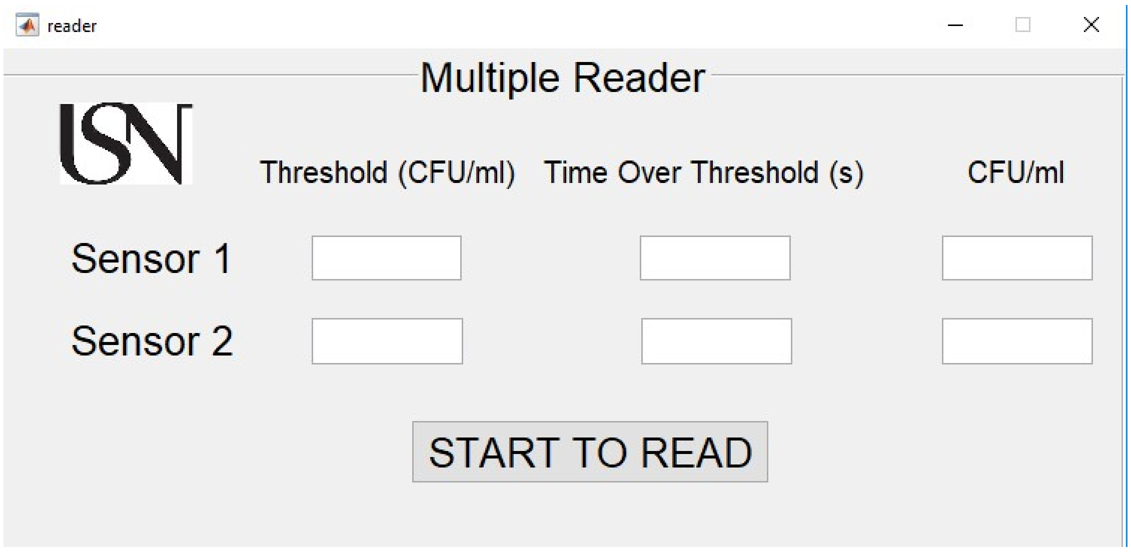


Figure 7.5: Multiple reader window in the second MATLAB interface version

imposed thresholds. Attached with the date, the threshold and the time imposed, and an appropriate code (1 if they are over threshold, 0 otherwise) the data are saved in two different .txt files to allow the access by the past data visualizer. If the bacteria concentrations are over the imposed thresholds two different counters (each for sensor) are incremented. When one of the counter is greater than the product between the time and the frequency sampling the system alarm is triggered. The alarm for this window consists in the push notification only. When the alarm is triggered a personalized message indicating the number of the sensor in danger and its relative GPS position collected from the GPS module will be received on the mobile phone.

A note about the frequency sampling. The loop is repeated every second to synchronize the voltage conversion from Arduino to the data collection from the interface. Actually, this interface reads and saves values with a non-constant frequency (1 sample every 3 or 4 seconds) due to the delay from the internet connection. Hence, a good internet connection is required if you don't want to lose many data. For this reason, the alarm and the data coming from the sensors are not perfectly in real time but with a delay of few seconds.

7.4.2 Real time visualizer

Like in the first version of the interface this window displays on a graph the signal evolution during the time. Once the user has choosen the sensor from a pop-up menu and press the 'READ SIGNAL' button the graph is updated with the incoming data.

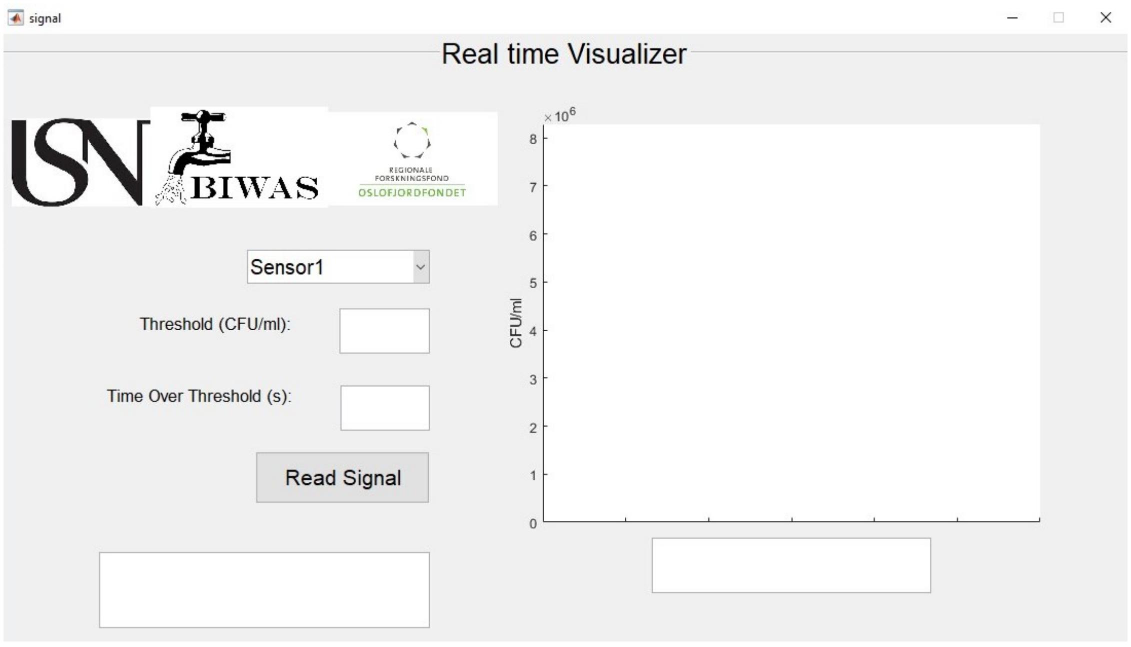


Figure 7.6: Real time signal visualizer window in the second version of MATLAB interface

Inside the code: Real time signal reader first version

Operating the sensor choice, the appropriate file name containing the parameters imposed by the user in the first window and the specific url name are selected. The parameters are taken from a .txt file to allow the two different MATLAB windows to work separately and simultaneously without interference and because MATLAB cannot run two infinite process at the same time. If the user modifies the parameters in the first window the real-time visualizer will show the new imposed values. When the button is pressed, at first, the parameters are printed in the appropriate text boxes and then the data are collected in the same way of the first window and as soon as they are available are plotted in a 30 s window wide graph, in which the y axis shows the bacteria concentration and the x axis shows the date and the time going on.

7.4.3 Past data visualizer

This window allows the user to visualize and control the data collected in a precise time interval to check if contaminations are occurred.

The user has to type the date in the specified format, or the window is disabled and the visualization is not performed. In this case too, a sensor has to be selected from a pop-up menu. Doing in this way the specific .txt file is open and read. Every line containing the data and the relative threshold collected between the

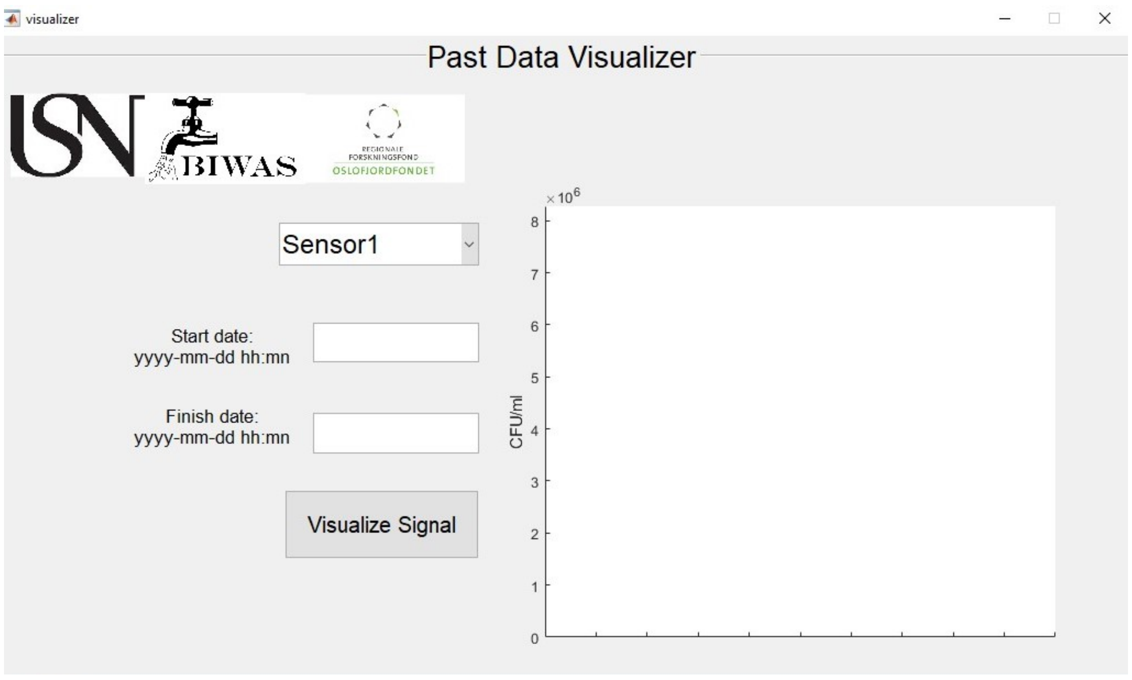


Figure 7.7: Past Data visualizer window in the second version of MATLAB interface

start and the finish date are taken and plotted in the graph. Because in this case the frequency sampling is not constant the x axis will not show exactly how many seconds of signal are displayed. It was supposed that the frequency was one sample every 3 seconds.

7.4.4 Desktop Applications

Using MATLAB Compliler three different standalone applications has been created. All applications created with MATLAB Compiler use the MATLAB Runtime, which enables royalty-free deployment to users who do not need MATLAB. You can package the runtime with the application, or have your users download it during installation. In this way it is possible to release them to the user, after the push notification is launched [95]. Moreover, the three windows run separately without interfering each other.

Figure 7.8: Desktop application icons

Chapter 8

Results

8.1 Calibration curve: water

Offline tests were performed to establish a relationship between the tryptophan intrinsic fluorescence and the bacteria concentration in water and in urine. The curve was measured with a start concentration of 8,3*10^6 CFU/ml, and the dilution described in the chapter 5was done. The presented results are an average value of the 3 measures performed by each dilution.

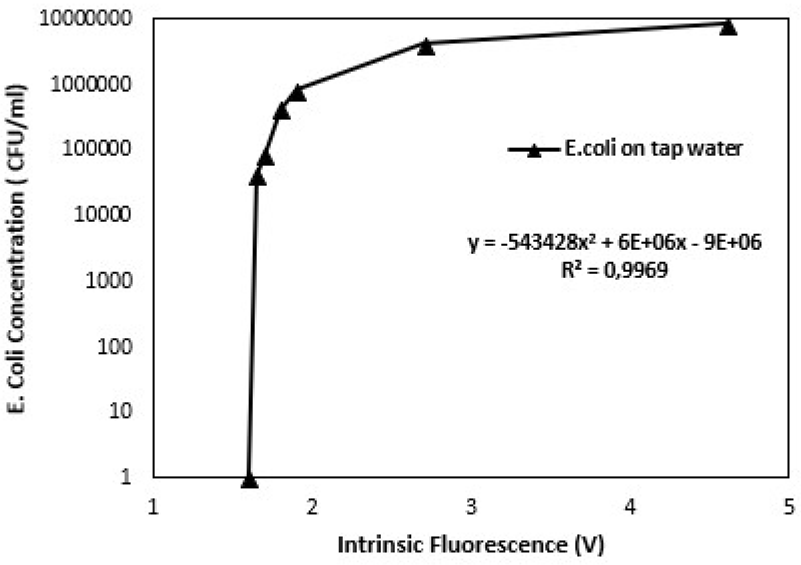


Figure 8.1: Calibration curve derived from tests in water

The result from the measurement with *E. Coli* is shown in figure 8.1. The curve fits quadratic model, y = -543428x^2 + 6*10^6x – 9*10^6, with a R^2 = 0,9969, which shows the model is highly correlated with the cell concentration. The equation presented on the graphic was implemented in the MATLAB code through the following way:

```
v(k)=str2double(data.signal); % voltage value
y(k) = (-543428*v(k)^2)+(6*10^6*v(k))-(9*10^6); % y is the concentration
```

More accurate tests were conducted by the other students, finding a top detection limit at $7{,}5 * 10^5$ CFU/ml and the lowest detectable bacteria concentration was $1{,}4 * 10^3$ CFU/ml.

8.2 Calibration curve: urine

In this case the start concentration was 8,9*10^6 CFU/ml, and the dilution series described in the chapter 5 was done. The results are an average value of 3 measures performed by each dilution. The result from the measurement with *E. Coli* is

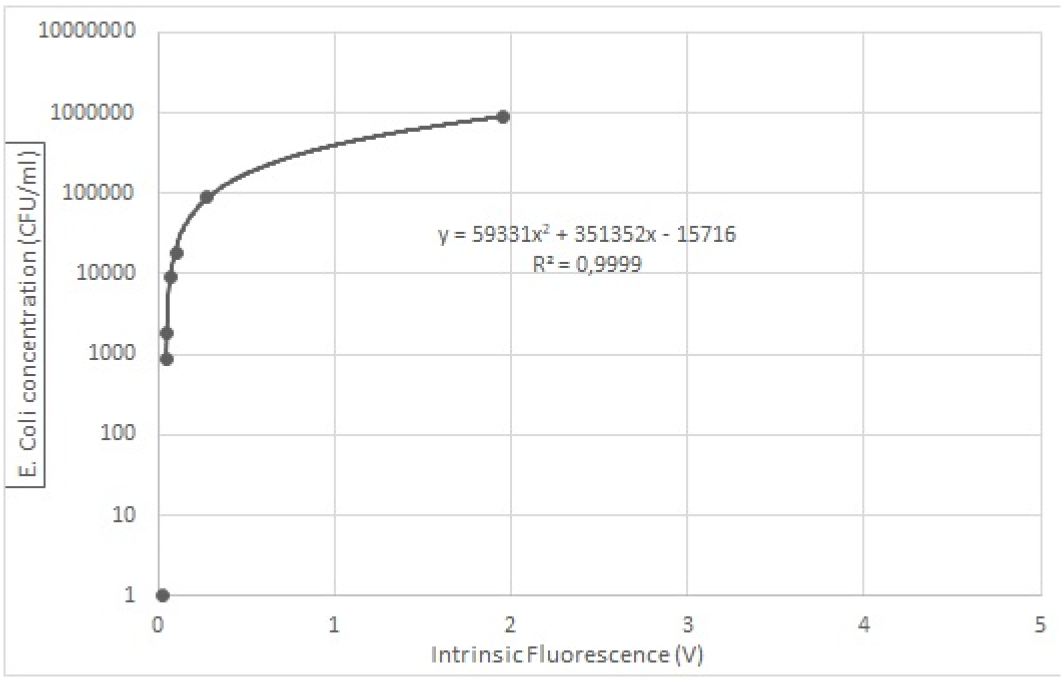


Figure 8.2: Calibration curve derived from tests in urine

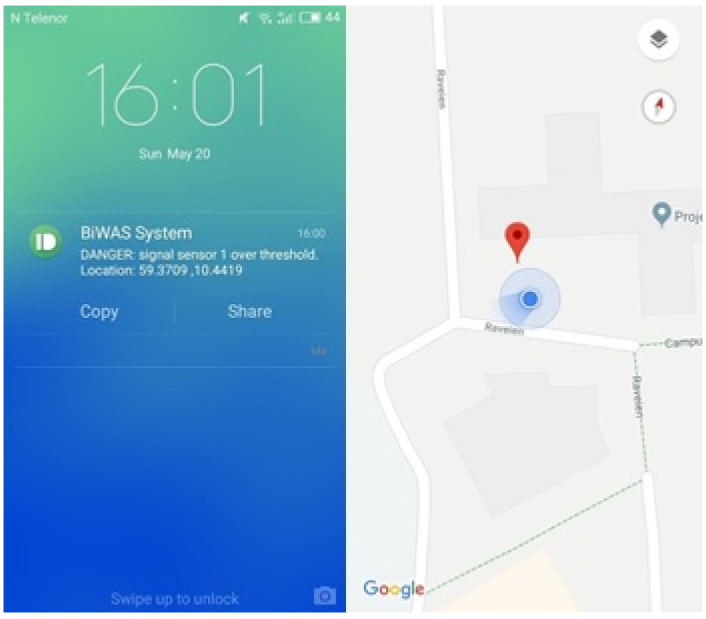
shown in figure 8.2. In this case too, the curve fits quadratic model, y = 59331x^2 + 351352x − 15716, with a R^2 = 0,9999, which shows high correlation with the bacteria concentration in urine.

Table 8.1: Results from urine test

Dilution	Concentration (CFU/ml)	Voltage (V)
$10^{-}1$	$895 * 10^3$	1.95
$10^{-}2$	$89.5 * 10^3$	0.28
$0.5 * 10^{-}2$	$17.9 * 10^3$	0.11
$10^{-}3$	$8.95 * 10^3$	0.07
$0.5 * 10^{-}3$	$1.79 * 10^3$	0.05
$10^{-}4$	895	0.05
pure urine	0	0.03

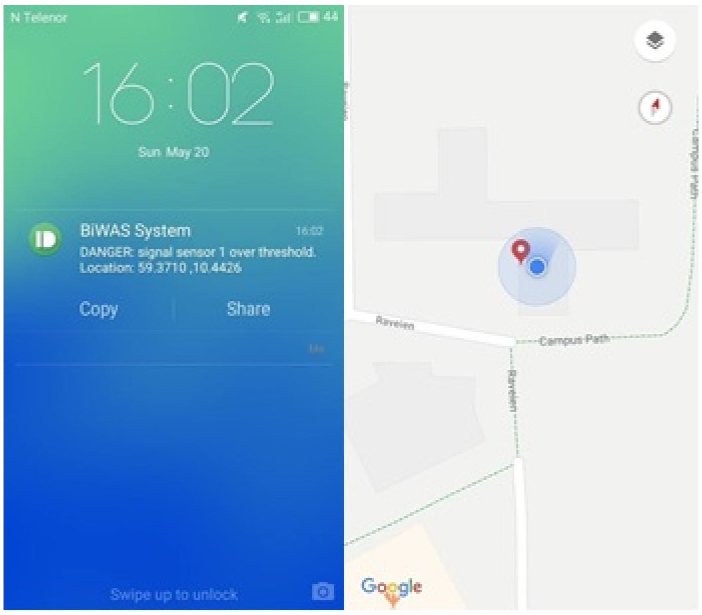
8.3 GPS accuracy

GPS accuracy has been tested. Two different position using the GPS module have been recorded and compared with the position given by Google Maps on Android Device. A push notification with the GPS coordinates given by the system has been triggered and from the same place the coordinates have been typed on Google Maps. The red spot indicates the GPS module, whereas the blue spot is the android device. Both the positions were taken in the research park building of the University of South-Eastern Norway.



BiWAS System 16:00
DANGER: signal sensor 1 over threshold.
Location: 59.3709 ,10.4419

Figure 8.3: On the left: push notification with first position coordinates. On the right: difference between the coordinates send by the GPS module and the Google maps position



BiWAS System 16:02
DANGER: signal sensor 1 over threshold.
Location: 59.3710 ,10.4426

Figure 8.4: On the left: push notification with second position coordinates. On the right: difference between the coordinates send by the GPS module and the Google maps position

As shown in the figures 8.3 and 8.4 the position given by the GPS module is comparable to the Android device's one. To achieve best results the GPS module should be placed outside and faced out.

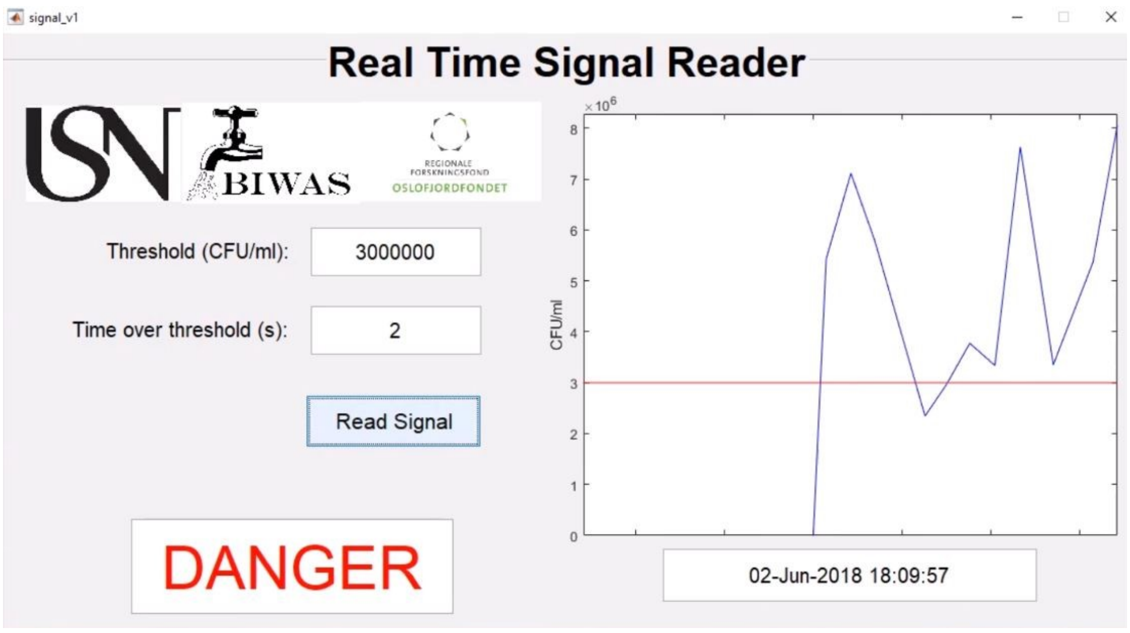
8.4 MATLAB interface: first version

Both the interfaces were tested with water samples because the urine was not available until the last days of work.

8.4.1 Real time signal reader

The first system was tested in online mode simulating a possible contamination event. First, the system was linked to a container with "pure" tape water, then to a container with a known bacteria concentration. As shown in the figure 8.5 the local alarm has been activated and the message has been displayed in the box. The push notification will be discussed in the section 8.5.2 since it is the same in the second version of the interface.

As shown in the figure 8.6 the data are correctly saved in the .txt file maintaining a constant frequency sampling of the incoming data. When the concentration is below the threshold 0 is placed at the end of the line, otherwise 1 is printed.



Real Time Signal Reader

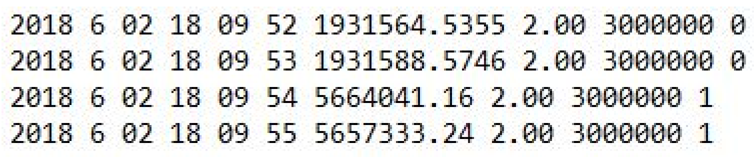
Threshold (CFU/ml): 3000000

Time over threshold (s): 2

DANGER

02-Jun-2018 18:09:57

Figure 8.5: Contamination event detection with the first communication system



```
2018 6 02 18 09 52 1931564.5355 2.00 3000000 0
2018 6 02 18 09 53 1931588.5746 2.00 3000000 0
2018 6 02 18 09 54 5664041.16 2.00 3000000 1
2018 6 02 18 09 55 5657333.24 2.00 3000000 1
```

Figure 8.6: Lines extracted from the .txt file where the data are saved. In order: year, month, day, hour, minute, second, concentration, time threshold, concentration threshold, coding value

8.4.2 Past data visualizer

The simulated contamination event described above was, then, displayed in the "Past data visualizer" window. The x axis indicates the duration expressed in seconds.

8.5 MATLAB interface: second version

8.5.1 Multiple reader

To test this MATLAB interface an experiment was conducted. One of the two system described in the chapter 7 was connected directly to the PMT and so to the sensor, as shown in the figure 8.8, whereas at the input on the second one a signal was simulated using the signal generator shown in the figure 8.9.

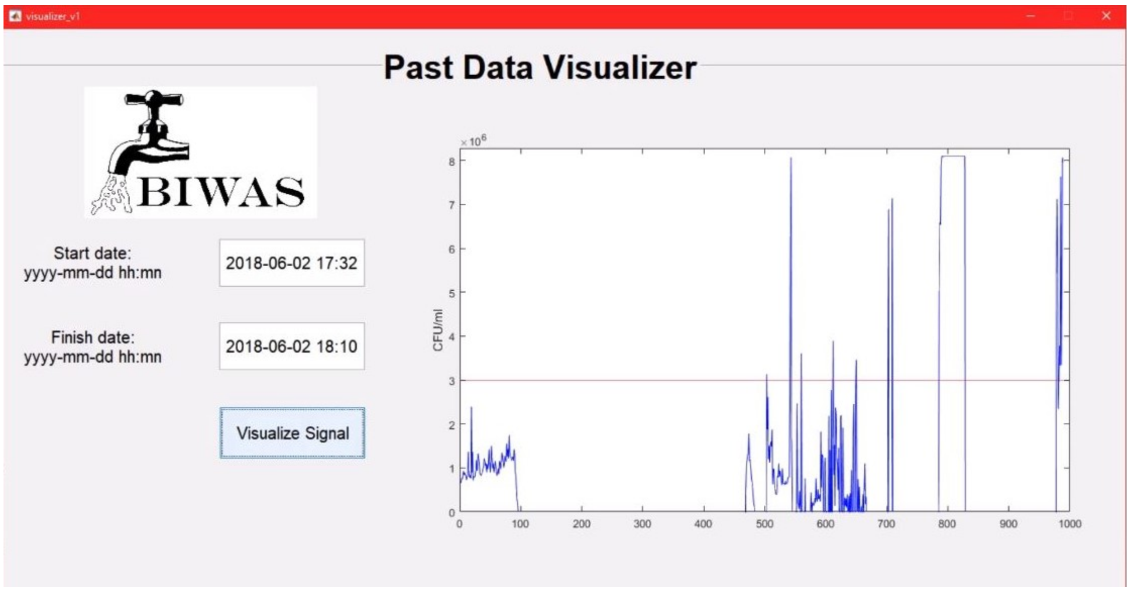


Figure 8.7: Past data visualizer window in the first version showing a contamination simulation

The values are properly collected by the multiple reader window showing the calibrated concentration values of bacteria for each sensor. The results are visible in the figure 8.10.

The figure 8.11 shows some lines from one of the .txt files used to save the data collected. The template is the same as the one showed in the figure 8.6. Even if it has been imposed that the reading loop should be performed every second, from one sample to another there is a distance of 3 or 4 seconds, as a consequence af the delay derived from the internet connection. Therefore, the multiple reader window will display the concentration with a variable delay, which will influence the time of the push notification coming too.

8.5.2 Real time visualizer

In the Real time visualizer window is shown the simulated signal in order to control in a better way a contamination event and to make a comparison between the time on the interface window and the time of the push notification. In the figures 8.12 and 8.13 have been highlighted the time on the interface window and the time of the notification event on the mobile phone. Despite the time imposed was two seconds, the push notification is not immediately sent to the mobile phone, but after a delay of few seconds and in the same minute of the contamination, preserving a early alarm system. For the first communication system developed the notification was more rapid and compatible with the time imposed by the user.

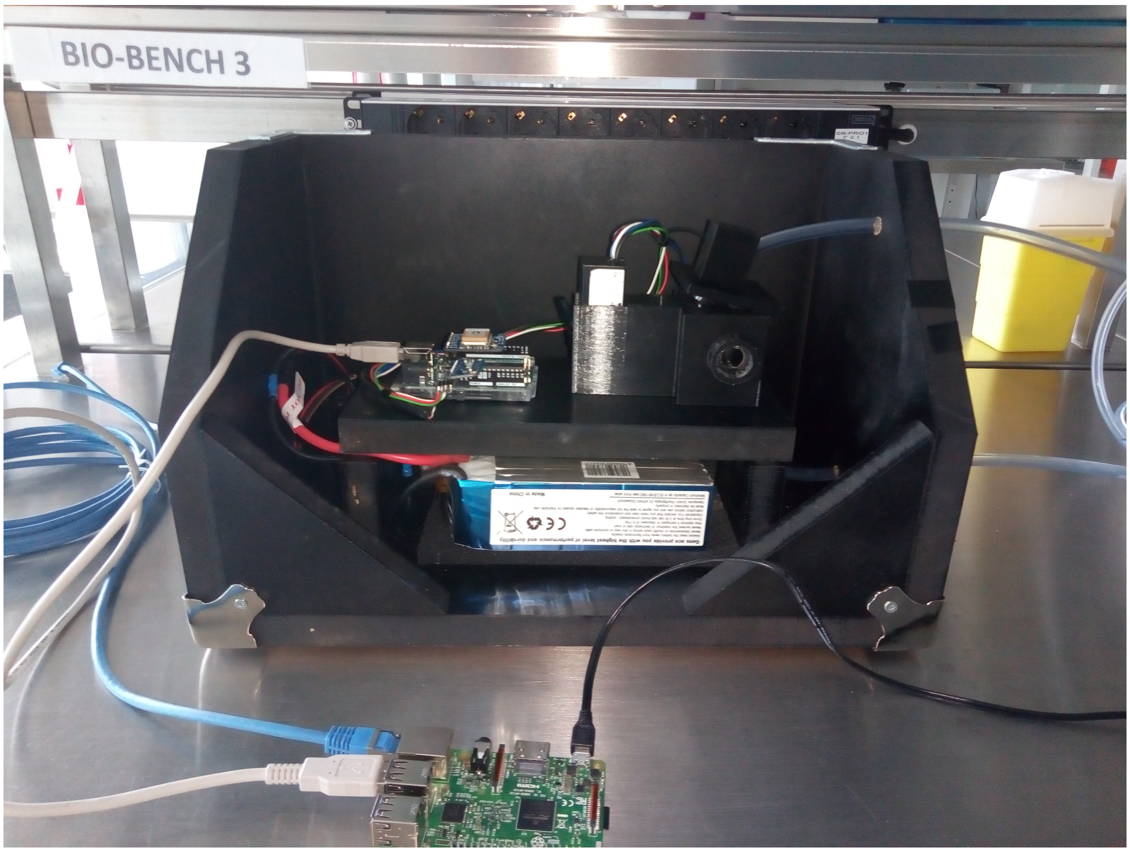


Figure 8.8: The picture shows the connections between the sensor, Arduino and Raspberry Pi. The Raspberry Pi is connected to Internet through ethernet cable

8.5.3 Past data visualizer

The time interval selected is the time in which the simulation has been performed. In the graph is possible to display the simulated contamination peaks.

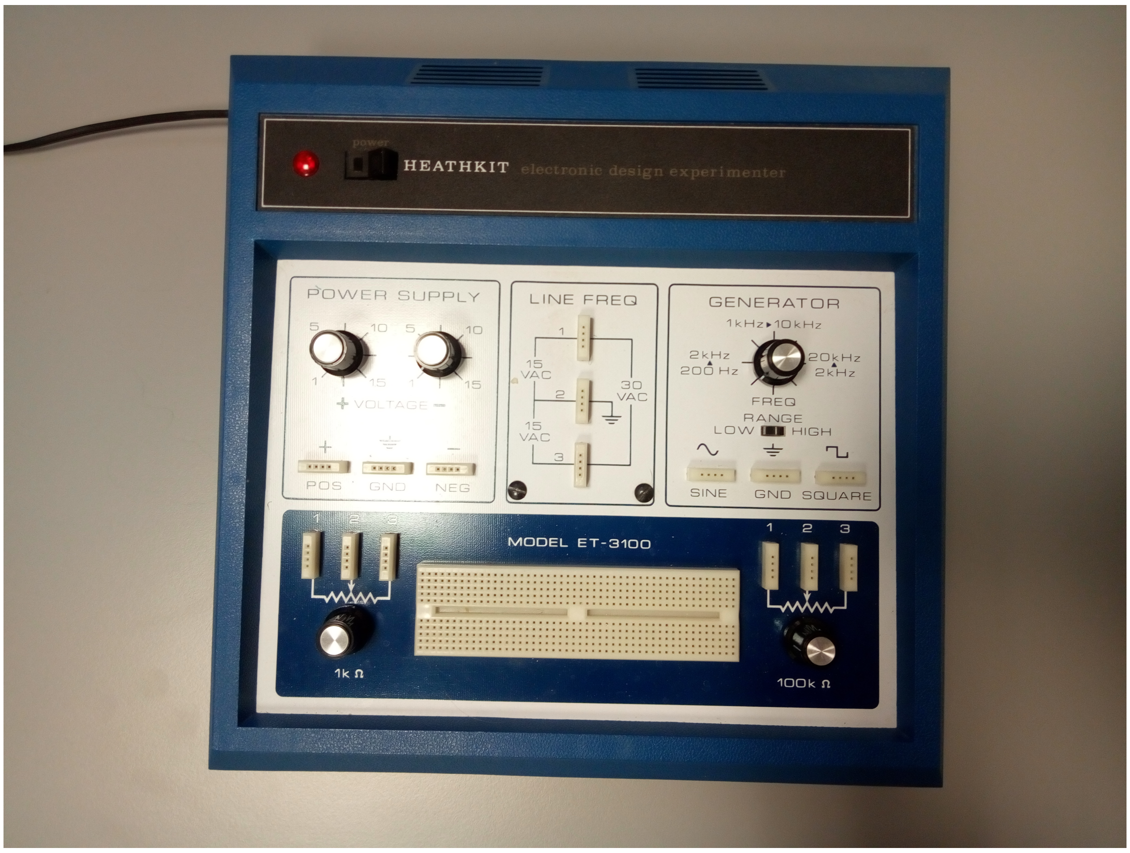
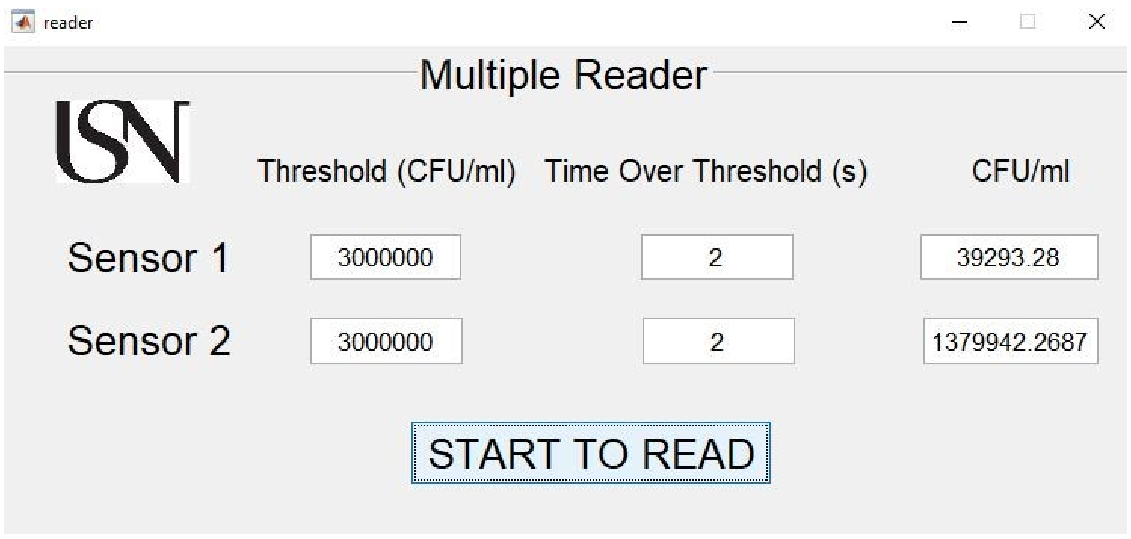


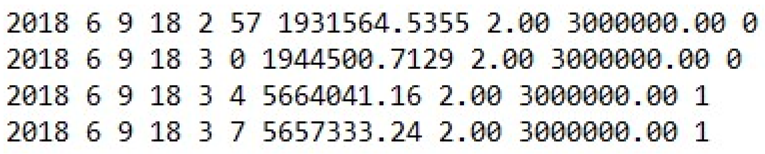
Figure 8.9: Signal generator



Multiple Reader

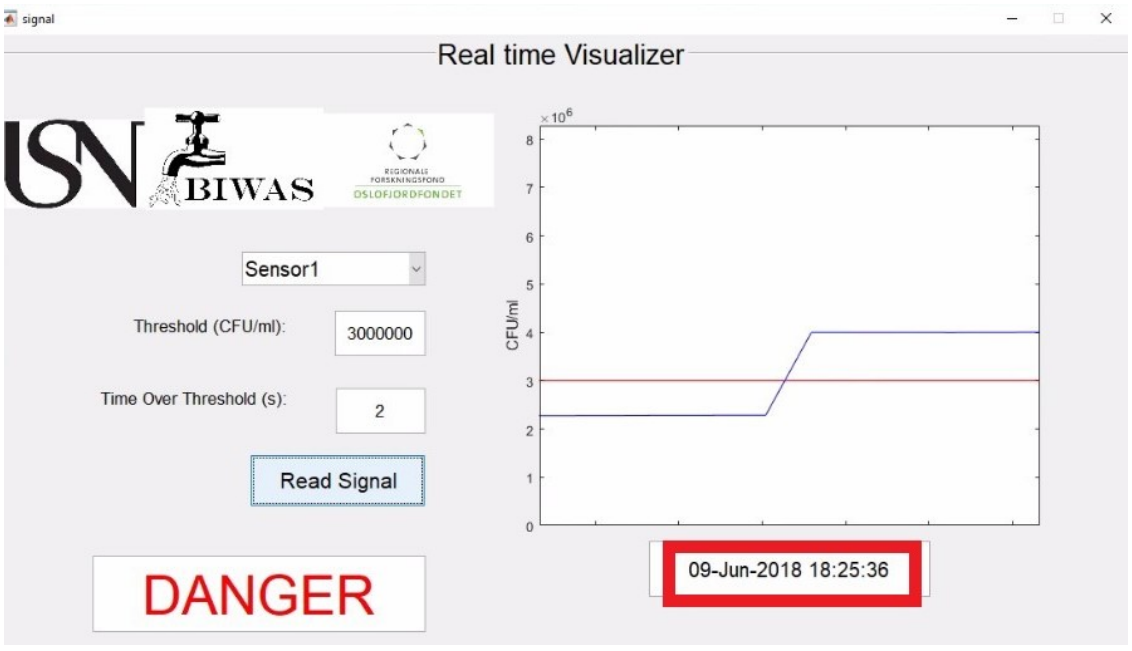
	Threshold (CFU/ml)	Time Over Threshold (s)	CFU/ml
Sensor 1	3000000	2	39293.28
Sensor 2	3000000	2	1379942.2687

Figure 8.10: Multiple reader window collecting and showing the bacteria concentration



```
2018 6 9 18 2 57 1931564.5355 2.00 3000000.00 0
2018 6 9 18 3 0 1944500.7129 2.00 3000000.00 0
2018 6 9 18 3 4 5664041.16 2.00 3000000.00 1
2018 6 9 18 3 7 5657333.24 2.00 3000000.00 1
```

Figure 8.11: Lines from one of the .txt file



signal

Real time Visualizer

Sensor1

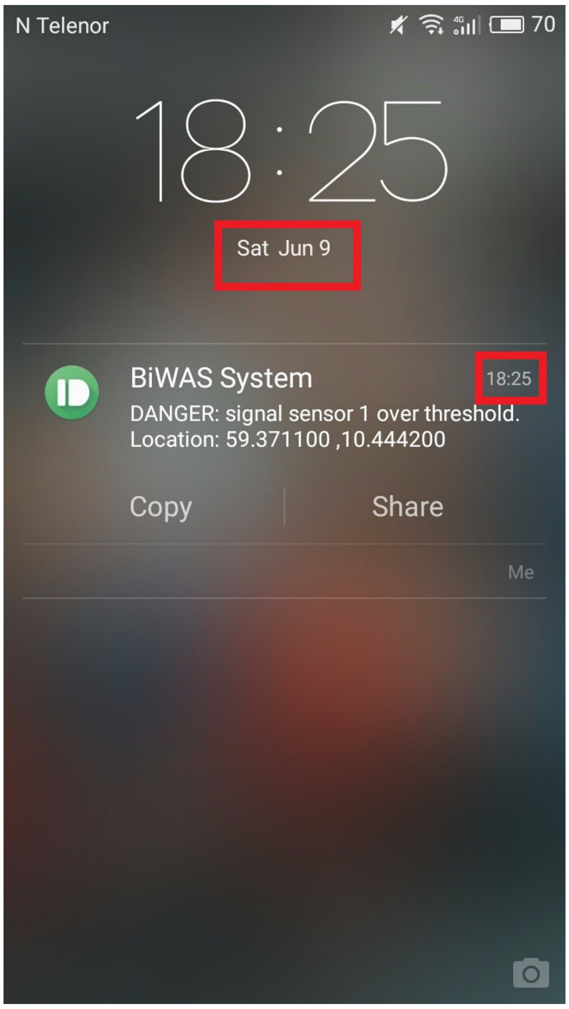
Threshold (CFU/ml): 3000000

Time Over Threshold (s): 2

DANGER

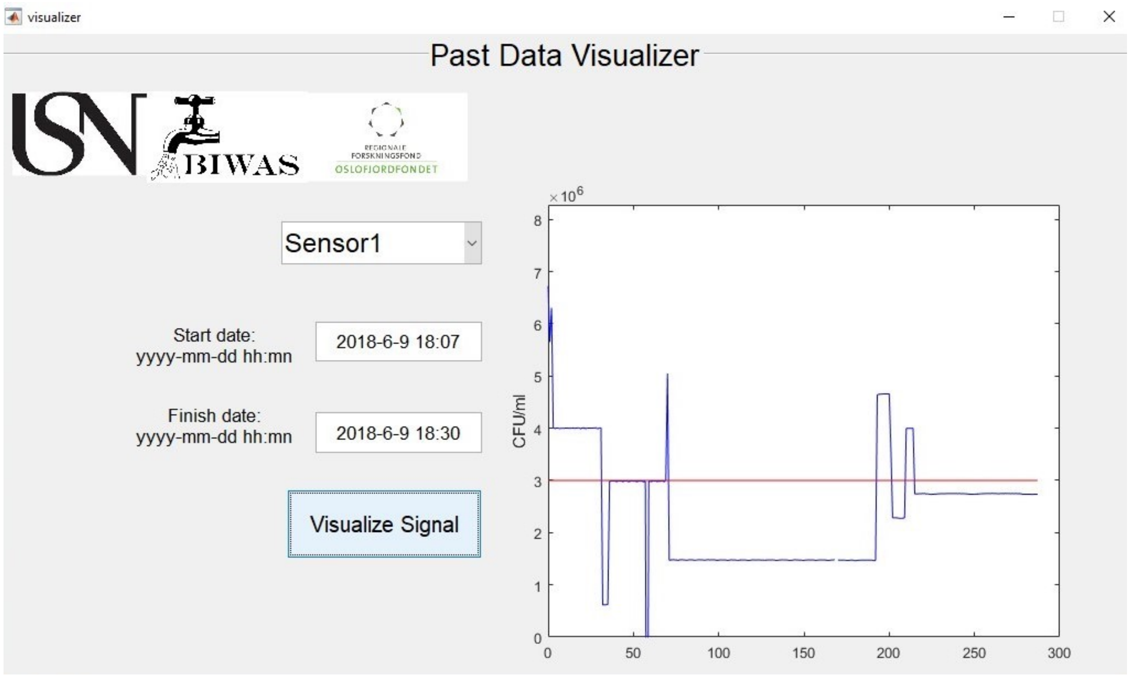
09-Jun-2018 18:25:36

Figure 8.12: Real time visualizer window in the second version on MATLAB interface showing a simulated contamination event



BiWAS System 18:25
DANGER: signal sensor 1 over threshold.
Location: 59.371100 ,10.444200

Figure 8.13: Push notification



Sensor1

Start date: yyyy-mm-dd hh:mn 2018-6-9 18:07

Finish date: yyyy-mm-dd hh:mn 2018-6-9 18:30

Figure 8.14: Past data visualizer showing the simulation

Chapter 9

Discussions

9.1 Spectrometry analysis

The spectrometry analysis shows a difference between the tryptophan spectrum in water and in urine, especially in the excitation wavelength: tryptophan signal in water has a great intensity at excitation wavelengths comprised between 275 and 280 nm, while in urine the major signal magnitude is at a 290 nm. The emission band of the two signal is similar, even if there is a shift from the expected theoretical value 350 nm to 330 nm. The peak of emission intensity is principally caused by the protein conformation which the tryptophan is folded in, dependent on polarity and properties of the solvent [96].

This suggests that to have better results in urine tests a new light source centred at about 290 nm should be used.

The interferences caused by the Rayleigh and Raman effects have the same weight in urine and water and the optic filter used in the sensor prototype, which allows only wavelengths higher than 320 nm, can be kept.

However, the analysis demonstrates that is possible to isolate the signal from tryptophan fluorescence and to detect this molecule inside the *E. Coli* giving a quantification of the pathogen concentration.

9.2 Water and urine tests

The tests in water and in urine have showed that using low cost and low power instrumentation was achievable to detect different concentrations of *Escherichia coli* in the two fluids in a time interval of few milliseconds, since the fluorescence response is very rapid compared to other detection techniques. According to Simões the lowest detection limit in water was $1{,}4 * 10^3$ CFU/ml, even if the results showed here are a bit different. In particular the two calibration curves have different equations, despite both of them show high correlation between the signal output of the sensor and the *E. coli* concentration. In particular the tests performed in urine have revealed a lower signal, generating only 1.95 V at the highest concentration. First, it was thought that the cause was the excitation wavelength, in agreement to what the spectrometry analysis showed, but at the end of the experiment it was noticed that one of the LEDs was broken, so the intensity of the excitation was lower and consequently the intensity of the emission light. A proper comparison between the tests was not possible.

Nevertheless, given the test results, the signal of pure urine was 0.03 V, instead the urine with a concentration of 894 CFU/ml produce a signal of 0.05 V establishing this value as the lowest detectable bacteria concentration. This is a good result, since some authors agree to the fact that a concentration of 10^3 CFU/ml is a sign of UTI, even if the most of the guidelines determine a threshold value of 10^5 CFU/ml as a symptom of bacteriuria [9, 19, 25, 48, 52]. A concentration of 10^5 was not available, but following the water results and the calibration equation, such concentration will generate an evident signal for sure.

9.3 Communication systems

Both the communication systems can be used in online and offline mode providing the connection between the sensor and the computer interfaces in real time. The first communication system, which uses radio frequency antenna, is more rapid than the second one but it has a limited range. The second system, of course, has an infinite range, since it uses the web to transfer the signal to a computer, but some delay may appear if the internet connection velocity is low.

A PCB was printed for the first one to have a more compact design and avoid the use of wires which can results in disconnecctions between the components. A second PCB should be done for the second system too, including the Arduino and the RPI. The use of RPI and Arduino, such as the radio frequency antenna, allow to keep the cost and the dimensions of the two systems low.

The code used for building the web server in the second system should be optimized, performing more checks on the incoming values, possibly eliminating the serial connection between Arduino and RPI, which can be source of errors. The server should be built as soon as the Raspberry Pi starts up to avoid the use of a monitor, keyboard and mouse in the installation phase. In addition, the system developed can read only two sensors because the 'ngrok' service used to expose the data to the public internet allows 120 connections/minute. Asking every second the samples from the sensors, like the interface does, it is clear that only two sensors can be implemented. After contacting the ngrok client service it has been discovered that a new plan could be signed allowing the exposure of many sensors as needed.

Since the interfaces have been thought to be released to the public, new interfaces in a different code should be implemented, especially the second version, due to the

MATLAB limitations: no multithreading, difficulties in creating catchy app and in integrating all the windows in a unique application, easier to use and sell.

Both systems are suitable in providing a continuous and real time monitoring in water and urine.

Chapter 10

Conclusions and future works

The sensor and the fluorescence technique show a great potential in contaminants detection in urine: it has been found that the system can correlate the level of the intrinsic fluorescence to presence/absence of UTI and, even better, it gives a quantification of *Escherichia coli* concentrations in the samples. The most important thing is that the system gives a response in few milliseconds, contrary to the standard diagnostic procedures presented in this report. The sensor is not able to provide the contaminants species or antimicrobial susceptibility informations, convenient in the antibiotics prescription, but it may avoid worthless prescriptions in negative patients, reducing the development of resistance patterns to antibiotics.

Some authors analysed intrinsic fluorescence excited at 290 nm in urine and they were able to distinguish between two groups of samples: UTI and control [79, 97], but the within-group variation between spectra was high. Davenport *et al.* reports that most UTI sensors performed well under laboratory conditions, but the integration in a real system and in the clinic conditions remains one of the most significant challenge [55]. Norazmi & al in a paper published in 2017 found a correlation between absorbance at 294 nm and uric acid concentration, proving that UV intrinsic fluorescence is not independent from other urine solutes.

Therefore, the next step is to perform tests with urine samples collected from control subjects and patient subjects with diagnosed urinary tract infection and evaluate the performances of the sensors, comparing it to the gold standard.

Adjustments and improvements are needed before proceeding in that direction.

Rapid and continuously operating rapid screening of UTI can be useful in different applications, but especially among patients with indwelling catheters. Through the feature of continuous flow, the sensor system may facilitate continuous monitoring of urine from catheters, which allows the establishment of an individual baseline level and monitoring how intrinsic fluorescence develops over time. This can allow better management of indwelling catheter use, since it can notify when there are significant elevations occurring in the urine, indicating a risk for bacterial colonization in the bladder, asymptomatic bacteriuria or symptomatic UTI infection (figure 10.1).

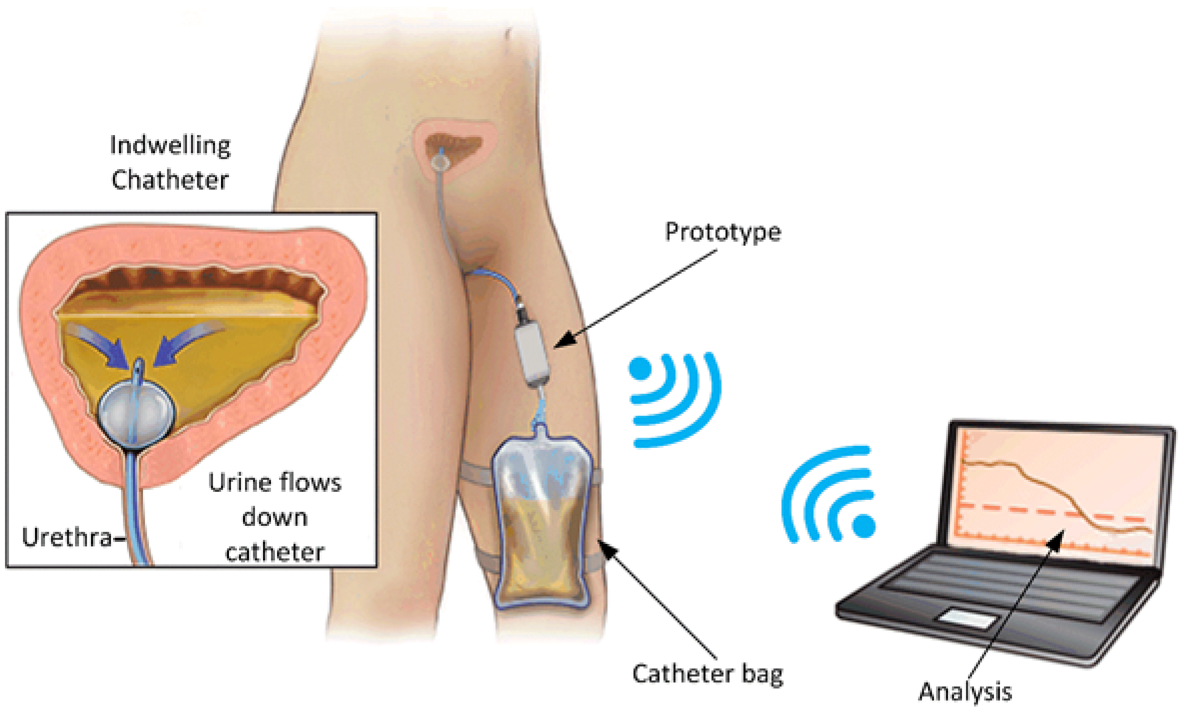


Figure 10.1: Sensor connected to an indwelling catheter performing real time analysis of urine passing through the device. Adapted from `https://myhealth.alberta.ca/Health/pages/conditions.aspx?hwid=zm6285`

Appendix A

Appendix

A.1 First communication system: Arduino codes for the wireless transmission

A.1.1 Transmitter

```
#include <SPI.h> // to handle the communication interface
   with the modem
#include <nRF24L01.h> //to handle this particular modem
   driver
#include <printf.h>
#include <RF24.h> //library which helps us to control the
   radio modem
#include <RF24_config.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>   //Initializing external ADC

Adafruit_ADS1115 ads1115;

RF24 radio(7, 8);  //create an object called "radio", 7 8
   are digital pin connected to CE and CSN (modem pins)

const byte rxAddr[6] = "00001"; //In this array we wrote
   the address of the modem, that will receive data from ADC


void setup(void)
{
  Serial.begin(115200); //to modify?
  ads1115.begin();
    // put your setup code here, to run once:
  ads1115.setGain(GAIN_TWOTHIRDS);

  radio.begin(); //It activates the modem
  radio.setRetries(15, 15);  //shows how many times the
     modem will retry to the send data in case of not
     receiving by another modem. The first argument sets how
      often modem will retry. It is a multiple of 250
     microseconds. 15 * 250 = 3750. Second argument is the
     number of attempts
  radio.openWritingPipe(rxAddr); //sets the address of the
     receiver
  radio.stopListening(); //switch the modem to data
     transmission mode.
}

void loop(void)
{
  int16_t adc0;

   adc0 = ads1115.readADC_SingleEnded(0); //adc0 stores the
       value of analog input A0
   radio.write(&adc0, sizeof(adc0)); // we send adco value
      through the radio to the modem. First argument is the
      variable that stores the data. The second argument is
      the number of bytes

   delay(1000); // the program will read, convert and send
      the signal every second


}
```

A.1.2 Receiver

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8);  //we creates a "radio" object with
   selected control pins
```

```
const byte rxAddr[6] ="00001";  //address of the receiver
   âĂŞ the same as in the transmitter
float voltage;
void setup()
{
  while (!Serial);
  Serial.begin(115200); //to modify

  radio.begin(); //set the nRF24L01 modem
  radio.openReadingPipe(0, rxAddr); //determines the
     address of our modem which receives data. The first
     argument is the number of the stream. The second
     argument is the address

  radio.startListening(); //enable receiving data via modem
}

void loop()
{
  if (radio.available()) //First checks whether any data
     have arrived at the address of the modem using the
     method"radio.available ();âĂİ. This method returns a"
     true" value if we received some data, or"false" if no
     data.
  {
    int16_t adc0 = {0}; //creates a 16-element"int" type
       array called"adc0" and filled with zeros
    radio.read(&adc0, sizeof(adc0)); //read the data
    voltage = (adc0 * 0.1875) / 1000;
    Serial.println(voltage, 4);
  }
delay(1000);
}
```

A.2 MATLAB interface: first version

A.2.1 Signal reader

```
function varargout = signal_v2(varargin)
% SIGNAL_V2 MATLAB code for signal_v2.fig
%       SIGNAL_V2, by itself, creates a new SIGNAL_V2 or
   raises the existing
```

```
%       singleton*.
%
%       H = SIGNAL_V2 returns the handle to a new SIGNAL_V2
    or the handle to
%       the existing singleton*.
%
%       SIGNAL_V2('CALLBACK',hObject,eventData,handles,...)
    calls the local
%       function named CALLBACK in SIGNAL_V2.M with the
    given input arguments.
%
%       SIGNAL_V2('Property','Value',...) creates a new
    SIGNAL_V2 or raises the
%       existing singleton*.  Starting from the left,
    property value pairs are
%       applied to the GUI before signal_v2_OpeningFcn gets
    called.  An
%       unrecognized property name or invalid value makes
    property application
%       stop.  All inputs are passed to signal_v2_OpeningFcn
     via varargin.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI
     allows only one
%       instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
    signal_v2

% Last Modified by GUIDE v2.5 05-Jun-2018 08:55:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
     'gui_Singleton',  gui_Singleton, ...
     'gui_OpeningFcn', @signal_v2_OpeningFcn, ...
     'gui_OutputFcn',  @signal_v2_OutputFcn, ...
     'gui_LayoutFcn',  [] , ...
     'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
       varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before signal_v2 is made visible.
function signal_v2_OpeningFcn(hObject, eventdata, handles,
   varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
% varargin   command line arguments to signal_v2 (see
   VARARGIN)

% Choose default command line output for signal_v2
handles.output = hObject;

axes(handles.axes6);
imshow('biwas_logo.png');
axes(handles.axes1);
ylabel('CFU/ml');
axis([0 30 0 8280000]);
set(gca,'xticklabel',[]);
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes signal_v2 wait for user response (see
   UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the
   command line.
function varargout = signal_v2_OutputFcn(hObject, eventdata
    , handles)
% varargout  cell array for returning output args (see
   VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
    of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function threshold_Callback(hObject, eventdata, handles)
% hObject    handle to threshold (see GCBO)
% eventdata  reserved - to be defined in a future version
    of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of
    threshold as text
%        str2double(get(hObject,'String')) returns contents
     of threshold as a double
threshold_im=str2double(get(hObject,'String'));
time_im= str2double(get(handles.time,'String'));
if isnan(threshold_im) || threshold_im<0 || threshold_im
   >8280000 || isnan(time_im) || ~isreal(time_im) || time_im
   <0
     msgbox('Choose a Threshold value between 0 and
         8,28*10^6 CFU/ml and a correct time value');
     set(handles.read_signal,'Enable','off'); beep
    %      uicontrol(hObject);
else
     set(handles.read_signal,'Enable','on');
     guidata(hObject,handles)
end
```

```
% --- Executes during object creation, after setting all
   properties.
function threshold_CreateFcn(hObject, eventdata, handles)
% hObject    handle to threshold (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function time_Callback(hObject, eventdata, handles)
% hObject    handle to time (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of time as
   text
%        str2double(get(hObject,'String')) returns contents
    of time as a double
time_im = str2double(get(hObject,'String'));
threshold_im=str2double(get(handles.threshold,'String'));
if isnan(time_im) || ~isreal(time_im) || time_im<0 || isnan
   (threshold_im) || threshold_im<0 || threshold_im>8280000
    msgbox('Choose a Threshold value between 0 and
       8,28*10^6 CFU/ml and a correct time value');
    set(handles.read_signal,'Enable','off'); beep
else
    set(handles.read_signal,'Enable','on');
    guidata(hObject,handles)
end
```

```
% --- Executes during object creation, after setting all
    properties.
function time_CreateFcn(hObject, eventdata, handles)
% hObject    handle to time (see GCBO)
% eventdata  reserved - to be defined in a future version
    of MATLAB
% handles    empty - handles not created until after all
    CreateFcns called

% Hint: edit controls usually have a white background on
    Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in read_signal.
function read_signal_Callback(hObject, eventdata, handles)
% hObject    handle to read_signal (see GCBO)
% eventdata  reserved - to be defined in a future version
    of MATLAB
% handles    structure with handles and user data (see
    GUIDATA)

fs=1;
pbO = pbNotify('o.4qbGe8fa6UdNmBX9deXxN8DyfuIF3Y5q'); %
    initialize push notifications
pbO.computer='BiWAS System';
% ROUTINE FOR REAL TIME READING
% create arduino object
delete(instrfind)
a = serial('COM3');                               %define
    serial port
a.BaudRate=115200;                                  %define
    baud rate

%open serial port
fopen(a);
```

```
k = 0;  %index
v = 0;  %voltage
t = 0;  %time

volte=0;
axes(handles.axes1)

scrollWidth=30; %window width
delay=1/fs;  %distance between two samples

tic % Start timer
while(1)
    threshold_im=str2double(get(handles.threshold,'String')
       );
    time_im= str2double(get(handles.time,'String'));

    k=k+1;
    v_read=fscanf(a,'%s'); %read arduino

    t(k) = toc;     %Extract Elapsed Time
    v(k)=str2double(v_read);
    th_y(k)=threshold_im;
    y(k) = (-543428*v1^2)+(6*10^6*v1)-(9*10^6); % y is the
       concentration of bacteria in CFU/mL
    date=clock;
    set(handles.clock,'String',string(datetime('now')));

    if y(k)>threshold_im
        A=[date y(k) time_im threshold_im 1];%A is vector
           with date (minute resolution), voltage amplitude
           (V), time over threshold imposed and threshold
        volte=volte+1;
        fileID=fopen('pastsignal.txt','a');
        fprintf(fileID,'\n%.0f %.0f %.0f %.0f %.0f %.0f %.2
           f %.2f %.2f %.0f\r\n', A(1),A(2),A(3),A(4),A(5),A
           (6),A(7),A(8),A(9),A(10));
        fclose(fileID);

        if(scrollWidth > 0)
            plot(t(t > t(k)-scrollWidth),y(t > t(k)-
               scrollWidth),'b'); hold on
            plot(t(t > t(k)-scrollWidth),th_y(t > t(k)-
                scrollWidth),'r')
            axis([t(k)-scrollWidth t(k) 0 8280000]); set(
                gca,'xticklabel',[]);axes(handles.axes1);
            ylabel('CFU/ml');
        else
            plot(t,y,'b'); hold on
            plot(t,th_y,'r')
            axis([0 t(k) 0 8280000]); set(gca,'xticklabel
                ',[]);axes(handles.axes1);
            ylabel('CFU/ml');
        end
        pause(delay);

        if volte>time_im*fs
            volte=0;
            beep; %msgbox('System in DANGER');
            set(handles.edit4,'String','DANGER');
            pbO.notify('Danger: signal over threshold');
        end

    else
        set(handles.edit4,'String',' ');
        A=[date y(k) time_im threshold_im 0];%A is vector
            with date (minute resolution), voltage amplitude
            (V), time over threshold imposed and threshold

        fileID=fopen('pastsignal.txt','a');
        fprintf(fileID,'\n%.0f %.0f %.0f %.0f %.0f %.0f %.2
            f %.2f %.2f %.0f\r\n', A(1),A(2),A(3),A(4),A(5),A
            (6),A(7),A(8),A(9),A(10));
        fclose(fileID);

        if(scrollWidth > 0)
            plot(t(t > t(k)-scrollWidth),y(t > t(k)-
                scrollWidth),'b'); hold on
            plot(t(t > t(k)-scrollWidth),th_y(t > t(k)-
                scrollWidth),'r')
            axis([t(k)-scrollWidth t(k) 0 8280000]); set(
                gca,'xticklabel',[]);axes(handles.axes1);
            ylabel('CFU/ml');
        else
```

```
            plot(t,y,'b'); hold on
            plot(t,th_y,'r')
            axis([0 t(k) 0 8280000]); set(gca,'xticklabel
               ',[]);axes(handles.axes1);
            ylabel('CFU/ml');
        end
        %Allow MATLAB to Update Plot
        pause(delay);



    end
end
% close the serial port!
fclose(a);



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as
    text
%        str2double(get(hObject,'String')) returns contents
    of edit4 as a double


% --- Executes during object creation, after setting all
   properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
```

```
%          See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function clock_Callback(hObject, eventdata, handles)
% hObject    handle to clock (see GCBO)
% eventdata  reserved - to be defined in a future version
    of MATLAB
% handles    structure with handles and user data (see
    GUIDATA)

% Hints: get(hObject,'String') returns contents of clock as
     text
%        str2double(get(hObject,'String')) returns contents
     of clock as a double


% --- Executes during object creation, after setting all
    properties.
function clock_CreateFcn(hObject, eventdata, handles)
% hObject    handle to clock (see GCBO)
% eventdata  reserved - to be defined in a future version
    of MATLAB
% handles    empty - handles not created until after all
    CreateFcns called

% Hint: edit controls usually have a white background on
    Windows.
%          See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

A.2.2 Past data visualizer

```
function varargout = visualizer_v2(varargin)
% VISUALIZER_V2 MATLAB code for visualizer_v2.fig
%      VISUALIZER_V2, by itself, creates a new
   VISUALIZER_V2 or raises the existing
%      singleton*.
%
%      H = VISUALIZER_V2 returns the handle to a new
   VISUALIZER_V2 or the handle to
%      the existing singleton*.
%
%      VISUALIZER_V2('CALLBACK',hObject,eventData,handles
   ,...) calls the local
%      function named CALLBACK in VISUALIZER_V2.M with the
   given input arguments.
%
%      VISUALIZER_V2('Property','Value',...) creates a new
   VISUALIZER_V2 or raises the
%      existing singleton*.  Starting from the left,
   property value pairs are
%      applied to the GUI before visualizer_v2_OpeningFcn
   gets called.  An
%      unrecognized property name or invalid value makes
   property application
%      stop.  All inputs are passed to
   visualizer_v2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
    allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
   visualizer_v2

% Last Modified by GUIDE v2.5 05-Jun-2018 08:33:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
                      @visualizer_v2_OpeningFcn, ...
```

```
                    'gui_OutputFcn',
                       @visualizer_v2_OutputFcn, ...
                    'gui_LayoutFcn',  [] , ...
                    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
       varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before visualizer_v2 is made visible.
function visualizer_v2_OpeningFcn(hObject, eventdata,
   handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
% varargin   command line arguments to visualizer_v2 (see
   VARARGIN)

% Choose default command line output for visualizer_v2
handles.output = hObject;

axes(handles.axes5);
imshow('biwas_logo.png');

axes(handles.axes1);
ylabel('CFU/ml');
ylim([0 8280000]);
set(gca,'xticklabel',[]);
% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes visualizer_v2 wait for user response (see
   UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the
   command line.
function varargout = visualizer_v2_OutputFcn(hObject,
   eventdata, handles)
% varargout  cell array for returning output args (see
   VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function date1_Callback(hObject, eventdata, handles)
% hObject    handle to date1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of date1 as
    text
%        str2double(get(hObject,'String')) returns contents
    of date1 as a double
try
start=datetime(get(hObject,'String'),'Format','yyyy-M-d h:m
   ');
set(handles.visualize,'Enable','on');
catch
    msgbox('The format is incorrect'); beep
    set(handles.visualize,'Enable','off');
end
```

```
% --- Executes during object creation, after setting all
   properties.
function date1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to date1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function date2_Callback(hObject, eventdata, handles)
% hObject    handle to date2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of date2 as
    text
%        str2double(get(hObject,'String')) returns contents
    of date2 as a double
try
finish=datetime(get(hObject,'String'),'Format','yyyy-M-d h:
   m');
set(handles.visualize,'Enable','on');
catch
    msgbox('The format is incorrect');
    set(handles.visualize,'Enable','off');
end
```

```
% --- Executes during object creation, after setting all
   properties.
function date2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to date2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in visualize.
function visualize_Callback(hObject, eventdata, handles)
% hObject    handle to visualize (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
start=datetime(get(handles.date1,'String'),'Format','yyyy-M
   -d H:m');
finish=datetime(get(handles.date2,'String'),'Format','yyyy-
   M-d H:m');
% start=str2num(get(handles.date1,'String'));
% finish=str2num(get(handles.date2,'String'));
cla reset
fileID=fopen('pastsignal.txt','r');
B=fscanf(fileID,'%f',[10 inf]);
fclose(fileID);
B=B';
fs=1;
k=0;
axes(handles.axes1);
for i=2:size(B,1)
    current_date=datetime(B(i,1:6));
    if current_date>=start && current_date<=finish
        k=k+1;
        d=k/fs;
        t=k-1:1/fs:d;
        plot(t,B(i-1:i,7),'Color','b'), hold on, ylim([0
           8280000]),ylabel('CFU/ml');

        plot(t,B(i-1:i,9),'Color','r'),hold on
    end
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as
    text
%        str2double(get(hObject,'String')) returns contents
    of edit3 as a double
```

A.3 Second communication system

A.3.1 Arduino code for the acquisition of the sensor and GPS signal and serial communication

```
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>   //Initializing external ADC
#include <Adafruit_GPS.h>       //Initializing GPS module

SoftwareSerial mySerial(3, 2);
Adafruit_GPS GPS(&mySerial);
Adafruit_ADS1115 ads1115;
float voltage;
#define GPSECHO  true
void setup(void)
{
  while (!Serial);
  Serial.begin(115200);
  ads1115.begin();
     // put your setup code here, to run once:
  ads1115.setGain(GAIN_TWOTHIRDS);
  GPS.begin(9600);
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA); //Sets
     output to only RMC and GGA sentences
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ); //Sets the
     output to 1/second. If you want you can go higher/lower
  GPS.sendCommand(PGCMD_ANTENNA); //Can report if antenna
     is connected or not
  delay(1000);
}

void loop(void)
{
  int16_t adc0;
  adc0 = ads1115.readADC_SingleEnded(0);
  voltage = (adc0 * 0.1875) / 1000;
  char c = GPS.read();
  GPS.parse(GPS.lastNMEA()); //This is going to parse the
     last NMEA sentence the Arduino has received, breaking
     it down into its constituent parts.
  Serial.print(voltage, 4);
 //Serial.print("Fix: "); Serial.print((int)GPS.fix);
  //Serial.print(" quality: "); Serial.println((int)GPS.
     fixquality);
  Serial.print(",");
  //Serial.print("Location : ");
  Serial.print(GPS.latitudeDegrees, 4);
  Serial.print(",");
  Serial.println(GPS.longitudeDegrees, 4);

  delay(1000); // the program will read, convert and send
     the signal every sec (f= 1 Hz)
}
```

A.3.2 Python script for the local web server

```
import serial
from flask import Flask, jsonify
import time
import pprint
```

```
# Rest API
app = Flask(__name__)
ret=dict()
port="/dev/ttyACM0"
#in the command line type: ls /dev/tty* before connecting
   arduino
#then plug in arduino and retype, take note of new port
   name appeared
#and type it in the line above



serialFromArduino = serial.Serial(port,115200)  #open the
   serial port connected to arduino
serialFromArduino.flushInput()                  #clear out the
    input buffer

@app.route('/', methods=['GET'])
def get_data():
   input = serialFromArduino.readline()
   pprint.pprint(input)
   serialFromArduino.flushInput
   a,b,c=input.decode().split(",")
   ret['signal']=a
   ret['latituted']=b
   ret['longitude']=c
   #pprint.pprint(ret)
   return jsonify(ret)
if __name__ == "__main__":
   app.run()
```

A.4 MATLAB interface: second version

A.4.1 Multiple reader

```
function varargout = reader(varargin)
% READER MATLAB code for reader.fig
%      READER, by itself, creates a new READER or raises
   the existing
%      singleton*.
%
```

```
%      H = READER returns the handle to a new READER or the
     handle to
%      the existing singleton*.
%
%      READER('CALLBACK',hObject,eventData,handles,...)
    calls the local
%      function named CALLBACK in READER.M with the given
    input arguments.
%
%      READER('Property','Value',...) creates a new READER
    or raises the
%      existing singleton*.  Starting from the left,
    property value pairs are
%      applied to the GUI before reader_OpeningFcn gets
    called.  An
%      unrecognized property name or invalid value makes
    property application
%      stop.  All inputs are passed to reader_OpeningFcn
    via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
     allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help reader

% Last Modified by GUIDE v2.5 05-May-2018 17:56:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
     'gui_Singleton',  gui_Singleton, ...
     'gui_OpeningFcn', @reader_OpeningFcn, ...
     'gui_OutputFcn',  @reader_OutputFcn, ...
     'gui_LayoutFcn',  [] , ...
     'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
       varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before reader is made visible.
function reader_OpeningFcn(hObject, eventdata, handles,
    varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
    of MATLAB
% handles    structure with handles and user data (see
    GUIDATA)
% varargin   command line arguments to reader (see VARARGIN
    )

% Choose default command line output for reader
handles.output = hObject;
axes(handles.axes1);
imshow('usn_logo.png');
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes reader wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the
    command line.
function varargout = reader_OutputFcn(hObject, eventdata,
    handles)
% varargout  cell array for returning output args (see
    VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
    of MATLAB
```

```
% handles    structure with handles and user data (see
   GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function threshold1_Callback(hObject, eventdata, handles)
% hObject    handle to threshold1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of
   threshold1 as text
%        str2double(get(hObject,'String')) returns contents
    of threshold1 as a double
threshold_1=str2double(get(hObject,'String'));
time_1= str2double(get(handles.time1,'String'));
if isnan(threshold_1) || threshold_1<0 || threshold_1
   >8280000 || isnan(time_1) || ~isreal(time_1) || time_1<0
    msgbox('Choose a Threshold value between 0 and
       8,28*10^6 CFU/ml and a correct time value');
    set(handles.reader,'Enable','off'); beep
    %     uicontrol(hObject);
else
    set(handles.reader,'Enable','on');
    guidata(hObject,handles)
end

% --- Executes during object creation, after setting all
   properties.
function threshold1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to threshold1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called
```

```
% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end



function time1_Callback(hObject, eventdata, handles)
% hObject    handle to time1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of time1 as
     text
%        str2double(get(hObject,'String')) returns contents
     of time1 as a double
time_1 = str2double(get(hObject,'String'));
threshold_1=str2double(get(handles.threshold1,'String'));
if isnan(threshold_1) || threshold_1<0 || threshold_1
   >8280000 || isnan(time_1) || ~isreal(time_1) || time_1<0
     msgbox('Choose a Threshold value between 0 and
        8,28*10^6 CFU/ml and a correct time value');
     set(handles.reader,'Enable','off'); beep

else
     set(handles.reader,'Enable','on');
     guidata(hObject,handles)
end

% --- Executes during object creation, after setting all
   properties.
function time1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to time1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called
```

```
% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function threshold2_Callback(hObject, eventdata, handles)
% hObject    handle to threshold2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of
   threshold2 as text
%        str2double(get(hObject,'String')) returns contents
    of threshold2 as a double
threshold_2=str2double(get(hObject,'String'));
time_2= str2double(get(handles.time2,'String'));
if isnan(threshold_2) || threshold_2<0 || threshold_2
   >8280000 || isnan(time_2) || ~isreal(time_2) || time_2<0
    msgbox('Choose a Threshold value between 0 and
       8,28*10^6 CFU/ml and a correct time value');
    set(handles.reader,'Enable','off'); beep

   %       uicontrol(hObject);
else
    set(handles.reader,'Enable','on');
    guidata(hObject,handles)
end



% --- Executes during object creation, after setting all
   properties.
function threshold2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to threshold2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end



function time2_Callback(hObject, eventdata, handles)
% hObject    handle to time2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of time2 as
     text
%        str2double(get(hObject,'String')) returns contents
     of time2 as a double
time_2 = str2double(get(hObject,'String'));
threshold_2=str2double(get(handles.threshold2,'String'));
if isnan(threshold_2) || threshold_2<0 || threshold_2
   >8280000 || isnan(time_2) || ~isreal(time_2) || time_2<0
     msgbox('Choose a Threshold value between 0 and
        8,28*10^6 CFU/ml and a correct time value');
     set(handles.reader,'Enable','off'); beep
else
     set(handles.reader,'Enable','on');
     guidata(hObject,handles)
end


% --- Executes during object creation, after setting all
   properties.
function time2_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to time2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in reader.
function reader_Callback(hObject, eventdata, handles)
% hObject    handle to reader (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
options = weboptions('Timeout',inf);
fs =0.5;
delay=1/fs;
volte_1=0;
volte_2=0;
pbO = pbNotify('o.4qbGe8fa6UdNmBX9deXxN8DyfuIF3Y5q'); %
   initialize push notifications
pbO.computer='BiWAS System';
while(1)
    date=clock;
    threshold_1=str2double(get(handles.threshold1,'String')
       );
    time_1= str2double(get(handles.time1,'String'));

    threshold_2=str2double(get(handles.threshold2,'String')
       );
    time_2= str2double(get(handles.time2,'String'));

    file1=fopen('parameters_1.txt','w');
    fprintf(file1,'%.2f %.2f', threshold_1, time_1);
```

```
fclose(file1);
file2=fopen('parameters_2.txt','w');
fprintf(file2,'%.2f %.2f', threshold_2, time_2);
fclose(file2);

data1=webread('https://sensor1.ngrok.io/',options);
data2=webread('https://sensor2.ngrok.io/',options);

v1=str2double(data1.signal);
lat1=str2double(data1.latituted);
long1=str2double(data1.longitude);
y1 = (-543428*v1^2)+(6*10^6*v1)-(9*10^6); % y is the
   concentration of bacteria in CFU/mL

v2=str2double(data2.signal);
lat2=str2double(data2.latituted);
long2=str2double(data2.longitude);
y2 = (-543428*v2^2)+(6*10^6*v2)-(9*10^6); % y is the
   concentration of bacteria in CFU/mL

s1=num2str(y1);
s2=num2str(y2);
set(handles.output1,'String',strtrim(s1));
set(handles.output2,'String',strtrim(s2));

message1=sprintf('DANGER: signal sensor 1 over
   threshold.                      Location: %f ,%f',lat1,
   long1);
message2=sprintf('DANGER: signal sensor 2 over
   threshold.                      Location: %f ,%f',lat2,
   long2);

if y1>threshold_1
     set(handles.output1,'String',strtrim(data1.signal))
        ;
     set(handles.output2,'String',strtrim(data2.signal))
        ;
     A1=[date y1 time_1 threshold_1 1];%A is vector with
         date (minute resolution), voltage amplitude (V),
         time over threshold imposed and threshold

     volte_1=volte_1+1;
```

```
        fileID=fopen('signal_1.txt','a');
        fprintf(fileID,'\n%.0f %.0f %.0f %.0f %.0f %.0f %.2
           f %.2f %.2f %.0f\r\n', A1(1),A1(2),A1(3),A1(4),A1
           (5),A1(6),A1(7),A1(8),A1(9),A1(10));
        fclose(fileID);


        if volte_1>time_1*fs
            volte_1=0;
            pbO.notify(message1);
        end
    else

        A1=[date y1 time_1 threshold_1 0];%A is vector with
            date (minute resolution), voltage amplitude (V),
            time over threshold imposed and threshold
        fileID=fopen('signal_1.txt','a');
        fprintf(fileID,'\n%.0f %.0f %.0f %.0f %.0f %.0f %.4
           f %.2f %.2f %.0f\r\n', A1(1),A1(2),A1(3),A1(4),A1
           (5),A1(6),A1(7),A1(8),A1(9),A1(10));
        fclose(fileID);

    end

    if y2>threshold_2
        A2=[date y2 time_2 threshold_2 1];%A is vector with
            date (minute resolution), voltage amplitude (V),
            time over threshold imposed and threshold
        volte_2=volte_2+1;
        fileID=fopen('signal_2.txt','a');
        fprintf(fileID,'\n%.0f %.0f %.0f %.0f %.0f %.0f %.2
           f %.2f %.2f %.0f\r\n', A2(1),A2(2),A2(3),A2(4),A2
           (5),A2(6),A2(7),A2(8),A2(9),A2(10));
        fclose(fileID);

        if volte_2>time_2*fs
            volte_2=0;
            pbO.notify(message2);
        end
    else
```

```
        A2=[date y2 time_2 threshold_2 0];%A is vector with
            date (minute resolution), voltage amplitude (V),
            time over threshold imposed and threshold

        fileID=fopen('signal_2.txt','a');
        fprintf(fileID,'\n%.0f %.0f %.0f %.0f %.0f %.0f %.4
           f %.2f %.2f %.0f\r\n', A2(1),A2(2),A2(3),A2(4),A2
           (5),A2(6),A2(7),A2(8),A2(9),A2(10));
        fclose(fileID);
    end
    pause (delay)
end



function output1_Callback(hObject, eventdata, handles)
% hObject    handle to output1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of output1
   as text
%        str2double(get(hObject,'String')) returns contents
    of output1 as a double


% --- Executes during object creation, after setting all
   properties.
function output1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to output1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function output2_Callback(hObject, eventdata, handles)
% hObject    handle to output2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of output2
   as text
%        str2double(get(hObject,'String')) returns contents
    of output2 as a double


% --- Executes during object creation, after setting all
   properties.
function output2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to output2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

A.4.2 Real time signal reader

```
function varargout = signal(varargin)
% SIGNAL MATLAB code for signal.fig
%      SIGNAL, by itself, creates a new SIGNAL or raises
   the existing
%      singleton*.
%
```

```
%       H = SIGNAL returns the handle to a new SIGNAL or the
     handle to
%       the existing singleton*.
%
%       SIGNAL('CALLBACK',hObject,eventData,handles,...)
    calls the local
%       function named CALLBACK in SIGNAL.M with the given
    input arguments.
%
%       SIGNAL('Property','Value',...) creates a new SIGNAL
    or raises the
%       existing singleton*.  Starting from the left,
    property value pairs are
%       applied to the GUI before signal_OpeningFcn gets
    called.  An
%       unrecognized property name or invalid value makes
    property application
%       stop.  All inputs are passed to signal_OpeningFcn
    via varargin.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI
     allows only one
%       instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help signal

% Last Modified by GUIDE v2.5 21-May-2018 11:01:05

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
     'gui_Singleton',  gui_Singleton, ...
     'gui_OpeningFcn', @signal_OpeningFcn, ...
     'gui_OutputFcn',  @signal_OutputFcn, ...
     'gui_LayoutFcn',  [] , ...
     'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
        varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before signal is made visible.
function signal_OpeningFcn(hObject, eventdata, handles,
   varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
% varargin   command line arguments to signal (see VARARGIN
   )

% Choose default command line output for signal
handles.output = hObject;
axes(handles.axes3);
imshow('usn_logo.png');
axes(handles.axes4);
imshow('biwas_logo.png');
axes(handles.axes5);
imshow('grant_logo.jpg');
axes(handles.axes1);
ylabel('CFU/ml');
axis([0 30 0 8280000]);
set(gca,'xticklabel',[]);
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes signal wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the
   command line.
```

```
function varargout = signal_OutputFcn(hObject, eventdata,
   handles)
% varargout  cell array for returning output args (see
  VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
  GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function threshold_Callback(hObject, eventdata, handles)
% hObject    handle to threshold (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
  GUIDATA)

% Hints: get(hObject,'String') returns contents of
   threshold as text
%        str2double(get(hObject,'String')) returns contents
    of threshold as a double

% threshold_im=str2double(get(hObject,'String'));
% time_im= str2double(get(handles.time,'String'));
% if isnan(threshold_im) || threshold_im<0 || threshold_im
  >5 || isnan(time_im) || ~isreal(time_im) || time_im<0
%     msgbox('Choose a Threshold value between 0 and 5 V
  and a correct time value');
%     set(handles.read_signal,'Enable','off'); beep
%     %     uicontrol(hObject);
% else
%     set(handles.read_signal,'Enable','on');
%     guidata(hObject,handles)
% end
% ——— Executes during object creation, after setting all
   properties.
function threshold_CreateFcn(hObject, eventdata, handles)
% hObject    handle to threshold (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function time_Callback(hObject, eventdata, handles)
% hObject    handle to time (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of time as
   text
%        str2double(get(hObject,'String')) returns contents
    of time as a double
% time_im = str2double(get(hObject,'String'));
% threshold_im=str2double(get(handles.threshold,'String'));
% if isnan(time_im) || ~isreal(time_im) || time_im<0 ||
   isnan(threshold_im) || threshold_im<0 || threshold_im>5
%     msgbox('Choose a Threshold value between 0 and 5 V
   and a correct time value');
%     set(handles.read_signal,'Enable','off'); beep
% else
%     set(handles.read_signal,'Enable','on');
%     guidata(hObject,handles)
% end
% --- Executes during object creation, after setting all
    properties.
function time_CreateFcn(hObject, eventdata, handles)
% hObject    handle to time (see GCBO)
% eventdata  reserved - to be defined in a future version
    of MATLAB
% handles    empty - handles not created until after all
    CreateFcns called

% Hint: edit controls usually have a white background on
    Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in read_signal.
function read_signal_Callback(hObject, eventdata, handles)
% hObject    handle to read_signal (see GCBO)
% eventdata  reserved - to be defined in a future version
    of MATLAB
% handles    structure with handles and user data (see
    GUIDATA)

fs=0.5;

k = 0;  %index
v = 0;  %voltage
t = 0;  %time

volte=0;
axes(handles.axes1)
scrollWidth=30; %window width
delay=1/fs;  %distance between two samples

tic % Start timer
while(1)
     filename=getappdata(0,'filename');
     fileID=fopen(filename,'r');
    C=fscanf(fileID,'%f',[2 inf]);
```

```
fclose(fileID);
threshold_im=C(1);
time_im= C(2);
set(handles.threshold,'String',num2str(C(1)));
set(handles.time,'String',num2str(C(2)));

k=k+1;
url=getappdata(0,'url');
data=webread(url);
t(k) = toc;    %Extract Elapsed Time
th_y(k)=threshold_im;
v(k)=str2double(data.signal); % voltage value
y(k) = (-543428*v(k)^2)+(6*10^6*v(k))-(9*10^6); % y is
   the concentration of bacteria in CFU/mL
set(handles.clock,'String',string(datetime('now')));

if y(k)>threshold_im
    volte=volte+1;

    if(scrollWidth > 0)
        %               plotGraph=
        plot(t(t > t(k)-scrollWidth),y(t > t(k)-
           scrollWidth),'b'); hold on
        plot(t(t > t(k)-scrollWidth),th_y(t > t(k)-
           scrollWidth),'r')
        axis([t(k)-scrollWidth t(k) 0 8280000]); set(
           gca,'xticklabel',[]);axes(handles.axes1);
        ylabel('CFU/ml');

    else
        %               plotGraph=
        plot(t,y,'b'); hold on
        plot(t,th_y,'r')
        axis([0 t(k) 0 8280000]); set(gca,'xticklabel
           ',[]);axes(handles.axes1);
        ylabel('CFU/ml');

    end
    pause(delay);

    if volte>time_im*fs
        volte=0;
```

```
            beep; %msgbox('System in DANGER');
            set(handles.edit4,'String','DANGER');
        end

    else
        set(handles.edit4,'String',' ');
        if(scrollWidth > 0)
            %               plotGraph=
            plot(t(t > t(k)-scrollWidth),y(t > t(k)-
               scrollWidth),'b'); hold on
            plot(t(t > t(k)-scrollWidth),th_y(t > t(k)-
               scrollWidth),'r')
            axis([t(k)-scrollWidth t(k) 0 8280000]); set(
               gca,'xticklabel',[]);axes(handles.axes1);
            ylabel('CFU/ml');

        else
            %               plotGraph=
            plot(t,y,'b'); hold on
            plot(t,th_y,'r')
            axis([0 t(k) 0 8280000]); set(gca,'xticklabel
               ',[]);axes(handles.axes1);
            ylabel('CFU/ml');
        end
        %Allow MATLAB to Update Plot
        pause(delay);



    end
end




function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit4 as
    text
%        str2double(get(hObject,'String')) returns contents
    of edit4 as a double


% --- Executes during object creation, after setting all
   properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
   popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected
   item from popupmenu1
contents=get(hObject,'Value');
switch contents
    case 1
        filename='parameters_1.txt';
        setappdata(0,'filename',filename);
        set(handles.threshold,'String',' ');
        set(handles.time,'String',' ');
        url='https://sensor1.ngrok.io/';
        setappdata(0,'url',url);

    case 2
        filename='parameters_2.txt';
        setappdata(0,'filename',filename);
        set(handles.threshold,'String',' ');
        set(handles.time,'String',' ');
        url='https://sensor2.ngrok.io/';
        setappdata(0,'url',url);

    otherwise
        set(handles.read_signal,'Enable','off');
end
% --- Executes during object creation, after setting all
   properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: popupmenu controls usually have a white background
   on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function clock_Callback(hObject, eventdata, handles)
% hObject    handle to clock (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of clock as
    text
%        str2double(get(hObject,'String')) returns contents
    of clock as a double


% --- Executes during object creation, after setting all
   properties.
function clock_CreateFcn(hObject, eventdata, handles)
% hObject    handle to clock (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

A.4.3 Past data visualizer

```
function varargout = visualizer(varargin)
% VISUALIZER MATLAB code for visualizer.fig
%      VISUALIZER, by itself, creates a new VISUALIZER or
   raises the existing
%      singleton*.
%
%      H = VISUALIZER returns the handle to a new
   VISUALIZER or the handle to
%      the existing singleton*.
%
%      VISUALIZER('CALLBACK',hObject,eventData,handles,...)
    calls the local
%      function named CALLBACK in VISUALIZER.M with the
   given input arguments.
%
%      VISUALIZER('Property','Value',...) creates a new
   VISUALIZER or raises the
```

```
%       existing singleton*.  Starting from the left ,
   property value pairs are
%       applied to the GUI before visualizer_OpeningFcn gets
    called.  An
%       unrecognized property name or invalid value makes
   property application
%       stop.  All inputs are passed to
   visualizer_OpeningFcn via varargin.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI
    allows only one
%       instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
   visualizer

% Last Modified by GUIDE v2.5 06-May-2018 10:56:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @visualizer_OpeningFcn
                      , ...
                   'gui_OutputFcn',  @visualizer_OutputFcn,
                       ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
       varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before visualizer is made visible.
function visualizer_OpeningFcn(hObject, eventdata, handles,
    varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
% varargin   command line arguments to visualizer (see
   VARARGIN)

% Choose default command line output for visualizer
handles.output = hObject;
axes(handles.axes2);
imshow('usn_logo.png');
axes(handles.axes3);
imshow('biwas_logo.png');
axes(handles.axes5);
imshow('grant_logo.jpg');
axes(handles.axes1);
ylabel('CFU/ml');
ylim([0 8280000]);
set(gca,'xticklabel',[]);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes visualizer wait for user response (see
   UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the
   command line.
function varargout = visualizer_OutputFcn(hObject,
   eventdata, handles)
% varargout  cell array for returning output args (see
   VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function date1_Callback(hObject, eventdata, handles)
% hObject    handle to date1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of date1 as
    text
%        str2double(get(hObject,'String')) returns contents
    of date1 as a double
try
start=datetime(get(hObject,'String'),'Format','yyyy-M-d h:m
   ');
set(handles.visualize,'Enable','on');
catch
    msgbox('The format is incorrect'); beep
    set(handles.visualize,'Enable','off');
end



% --- Executes during object creation, after setting all
   properties.
function date1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to date1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called
```

```
% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function date2_Callback(hObject, eventdata, handles)
% hObject    handle to date2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of date2 as
    text
%        str2double(get(hObject,'String')) returns contents
    of date2 as a double
try
finish=datetime(get(hObject,'String'),'Format','yyyy-M-d h:
   m');
set(handles.visualize,'Enable','on');
catch
    msgbox('The format is incorrect');
    set(handles.visualize,'Enable','off');
end



% --- Executes during object creation, after setting all
   properties.
function date2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to date2 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called
```

```
% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in visualize.
function visualize_Callback(hObject, eventdata, handles)
% hObject    handle to visualize (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)
start=datetime(get(handles.date1,'String'),'Format','yyyy-M
   -d H:m');
finish=datetime(get(handles.date2,'String'),'Format','yyyy-
   M-d H:m');
% start=str2num(get(handles.date1,'String'));
% finish=str2num(get(handles.date2,'String'));
cla reset
filename=getappdata(0,'filename');
fileID=fopen(filename,'r');
B=fscanf(fileID,'%f',[10 inf]);
fclose(fileID);
B=B';
fs=0.5;
k=0;
axes(handles.axes1);
for i=2:size(B,1)
    current_date=datetime(B(i,1:6));
    if current_date>=start && current_date<=finish
        k=k+1;
        d=k/fs;
        t=k-1:1/fs:d;
        plot(t,B(i-1:i,7),'Color','b'), hold on,  ylim([0
           8280000]),ylabel('CFU/ml');
        plot(t,B(i-1:i,9),'Color','r'),hold on
    end
end
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as
    text
%        str2double(get(hObject,'String')) returns contents
    of edit3 as a double


% --- Executes during object creation, after setting all
   properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: edit controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    structure with handles and user data (see
   GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
   popupmenu1 contents as cell array
```

```
%         contents{get(hObject,'Value')} returns selected
   item from popupmenu1
contents=get(hObject,'Value');
switch contents
    case 1
        filename='signal_1.txt';
        setappdata(0,'filename',filename);
    case 2
        filename='signal_2.txt';
        setappdata(0,'filename',filename);
        otherwise
        set(handles.visualize,'Enable','off');
end


% --- Executes during object creation, after setting all
   properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version
   of MATLAB
% handles    empty - handles not created until after all
   CreateFcns called

% Hint: popupmenu controls usually have a white background
   on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Bibliography

1. Stamm, W. & Norrby, S. Urinary tract infections: disease panorama and challenges. *The Journal of infectious diseases* **183,** S1–S4 (2001).
2. Karlsen, H. *Smartphone-based urinary biomarker detection: an application-oriented device and algorithm* PhD thesis (University of Southeast Norway, 2018).
3. [Online; accessed june-2018]. `\url{https://www.dreamstime.com/stock-illustration-urinary-system-anatomy-isolated-white-vector-photo-realistic-illustration-image49749775}`.
4. Lindsay, E. N. Epidemiology of Urinary Tract Infections. *Clinical microbiology newsletter* **24** (2002).
5. Hooton, T. M. Uncomplicated Urinary tract infection. *The New England Journal of Medicine* **366,** 1028–1037 (2012).
6. Sheerin, N. S. Urinary tract infection. *Medicine* **43,** 435–439 (2015).
7. Lichtenberger, P. & Hooton, T. M. Complicated Urinary tract infections. *Current Infectious Disease Reports* **10,** 499–504 (2008).
8. Smeltzer, S. C. O., Bare, B. G., Hinkle, J. L. & Cheever, K. H. in (eds Williams, L. & Wilkins) 1359 (Philadelphia, 2010).
9. Pezzlo, M. Detection of Urinary tract infections by rapid methods. *CLINICAL MICROBIOLOGY REVIEWS* **1,** 268–280 (1988).
10. Foxman, B. Epidemiology of urinary tract infections: Incidence, morbidity, and economic costs. *Disease a Month* **49,** 53–70 (2003).
11. Magill, S. S. *et al.* Multistate Point-Prevalence Survey of Health Care–Associated Infections. *The New England Journal of Medicine* **370,** 1198–1208 (2014).
12. Kumar, M., Ghosh, S., Nayak, S & A.P.Das. Recent advances in biosensor based diagnosis of urinary tract infection. *Biosensors and Bioelectronics* **80,** 497–510 (2016).
13. Saha, S. *et al.* Understanding the patterns of antibiotic susceptibility of bacteria causing urinary tract infection in West Bengal, India. *Frontiers in Microbiology* **5,** 1–7 (2014).

14. Kaur, N., Sharma, S., Malhotra, S., Madan, P. & Hans, C. Urinary Tract Infection: Aetiology and Antimicrobial Resistance Pattern in Infants From A Tertiary Care Hospital in Northern India. *Journal of clinical and diagnostic research* **8,** 1–3 (2014).

15. Foxman, B. Urinary tract infection syndromes: occurrence, recurrence, bacteriology, risk factors, and disease burden. *Infectious Disease Clinics of North America* **28,** 1–13 (2014).

16. Foxman, B. The epidemiology of urinary tract infection. *Nature Reviews Urology* **7,** 653–660 (2010).

17. Nicolle, L. Urinary tract infection in geriatric and institutionalized patients. *Current opinion in urology* **12,** 51–55 (2002).

18. Kunin, C. . *Detection, prevention and management of urinary tract infections* (Lea and Febiger, 1987).

19. Kass, E. H., Savag, W. & Santamarina, B. A. G. in *Progress on pyelonephritis* (F. A. Davis Co.).

20. Kass, E. H. Bacteriuria and Pyelonephritis of Pregnancy. *AMA Arch Intern Med* **105,** 194–198 (1960).

21. Matuszkiewicz-Rowińska, J., Małyszko, J. & Wieliczko, M. Urinary tract infections in pregnancy: old and new unresolved diagnostic and therapeutic problems. *Archives of Medical Science* **11,** 67–77 (2015).

22. Lipsky, B. A., Plord, J. J., Tenover, F. C. & Brancato, F. P. Comparison of the automicrobic system, acridine orange-stained smears, and gram-stained smears in detecting bacteriuria. *Journal of Clinical Microbiology* **22,** 176–181 (1985).

23. Yoshikawa, T. Unique aspects of urinary tract infection in the geriatric population. *Gerontology* **30,** 339–344 (1984).

24. Wikipedia. *Bacteriuria —Wikipedia, The Free Encyclopedia* [Online; accessed 19-October-2018]. 2018. `\url{http://en.wikipedia.org/wiki/Bacteriuria}`.

25. Das, K. K. in (ed Ltd, J. M.) 1250 (The Health Sciences Publsher, 2017).

26. Wikipedia. *Pyuria —Wikipedia, The Free Encyclopedia* [Online; accessed 19-October-2018]. 2018. `\url{https://en.wikipedia.org/wiki/Pyuria}`.

27. Gorczynska, E. *et al.* Incidence, Clinical Outcome, and Management of Virus-Induced Hemorrhagic Cystitis in Children and Adolescents after Allogeneic Hematopoietic Cell Transplantation. *Biology of Blood and Marrow Transplantation* **11,** 797–804 (2005).

28. Barnett, B. & Stephens, D. Urinary tract infection: an overview. *The American Journal of the Medical Sciences* **314,** 245–249 (1997).

29. Girard, T. & Ely, E. Bacteremia and sepsis in older adults. *Clinics in Geriatric Medicine* **23,** 633–647 (2007).
30. Flores-Mireles, A. L., Walker, J. N., Caparon, M. & Hultgren, S. J. Urinary tract infections: epidemiology, mechanisms of infection and treatment options. *Nat Rev Microbiol.* **13,** 269–284 (2015).
31. Wikipedia. *Bacteremia —Wikipedia, The Free Encyclopedia* [Online; accessed 15-October-2018]. 2018. `\url{https://en.wikipedia.org/wiki/Bacteremia}`.
32. Abrams, P. *et al.* The standardisation of terminology of lower urinary tract function: Report from the standardisation sub-committee of the international continence society. *American Journal of Obstetrics and Gynecology* **187,** 116–126 (2002).
33. Irwin, D. E. *et al.* Population-Based Survey of Urinary Incontinence, Overactive Bladder, and Other Lower Urinary Tract Symptoms in Five Countries: Results of the EPIC Study. *European Urology* **50,** 1306–1315 (2006).
34. Malmsten, U. G., Molander, U., Peeker, R., Irwin, D. E. & Milsom, I. Urinary Incontinence, Overactive Bladder, and Other Lower Urinary Tract Symptoms: A Longitudinal Population-Based Survey in Men Aged 45–103 Years. *European Urology* **58,** 149–156 (2010).
35. Thomas, T. M., Plymat, K. R., Blannin, J & Meade, T. W. Prevalence of urinary incontinence. *British medical journal* **281** (1980).
36. Stare, P. & Libow, L. Obscuring urinary incontinence. Diapering of the elderly. *Journal of the American Geriatrics Society* **33,** 842–846 (1985).
37. Et al., R. O. Pad per day usage, urinary incontinence and urinary tract infections in nursing home residents. *Age and Ageing* **39,** 549–554 (2010).
38. Warren, J. W. Catheter-associated urinary tract infections. *International Journal of Antimicrobial Agents* **17,** 299–303 (2001).
39. Jacobsen, S., Stickler, D., Mobley, H. & Shirtliff, M. Complicated catheter-associated urinary tract infections due to Escherichia coli and Proteus mirabilis. *Clinical Microbiology Reviews* **21,** 26–+ (2008).
40. Kelly, C. E. Evaluation of voiding dysfunction and measurement of bladder volume. eng. *Reviews in urology* **6 Suppl 1** (2004).
41. Jung, T. *et al. UP-2.54: What volume of post-void residual urine causes urinary tract infection?* eng. 2010.
42. Wijma, J., Potters, A. E. W., Wolf, B. T.H. M., Tinga, D. J. & Aarnoudse, J. G. Anatomical and functional changes in the lower urinary tract during pregnancy. *An International Journal of Obstetrics and Gynaecology* **108,** 726–732 (2001).

43. Zhanel, G. G., Harding, G. K. M. & Nicolle, L. E. Asymptomatic Bacteriuria in Patients with Diabetes Mellitus. *Reviews of Infectious Diseases* **13,** 150–154 (1991).

44. Muller, L. M.A. J. *et al.* Increased Risk of Common Infections in Patients with Type 1 and Type 2 Diabetes Mellitus. *Clinical Infectious Diseases* **41,** 281–288 (2005).

45. Geerlings, S. E. *et al.* Risk factors for symptomatic urinary tract infection in women with diabetes. eng. *Diabetes care* **23** (2000).

46. Jung, O *et al.* Resistant hypertension? Assessment of adherence by toxicological urine analysis. *Journal of Hypertension* **31,** 766–774 (2013).

47. Olsson, C., Kapoor, D. & Howard, G. A Method for the Rapid Detection of Urinary Tract Infections. eng. *Urology* **79,** 761–765 (2012).

48. Schmiemann, G, Kniehl, E, Gebhardt, K, Matejczyk, M. & Hummers-Pradier, E. The Diagnosis of Urinary Tract Infection A Systematic Review. English. *Deutsches Arzteblatt International* **107,** 361–U9 (2010).

49. Kumar, M. & Das, A. Emerging nanotechnology based strategies for diagnosis and therapeutics of urinary tract infections: A review. *Advances in Colloid and Interface Science* **249,** 53–65 (2017).

50. Boonenand, K, Koldewijn, E, Arents, N, Raaymakersa, P & Scharnhorst, V. Urine flow cytometry as a primary screening method to exclude urinary tract infections. *World Journal of Urology* **31,** 547–551 (2013).

51. Broeren, M. A. C., Bahceci, S., Vader, H. L. & Arents, N. L. A. Screening for Urinary Tract Infection with the Sysmex UF-1000i Urine Flow Cytometer. *JOURNAL OF CLINICAL MICROBIOLOGY* **49,** 1025–1029 (2011).

52. Schmiemann, G, Gebhardt, K, Matejczyk, M & Hummers-Pradier, E. Brennen beim Wasserlassen. *Degam* **1,** 1–8 (2009).

53. Loeb, M & al. Development of minimum criteria for the initiation of antibiotics in residents of long-term-care facilities: results of a consensus conference. *Infect Control Hosp Epidemiol* **22,** 120–124 (2001).

54. McGeer, A & al. Definitions of infection for surveillance in long-term care facilities. *American journal of infection control* **19,** 1–7 (1991).

55. Davenport, M *et al.* New and developing diagnostic technologies for urinary tract infections. *Nature Reviews Urology* **14,** 296–310 (2017).

56. Ahmed, N. H., Biswal, I. & Hussain, T. Comparison of etiological agents and resistance patterns of the pathogens causing community acquired and hospital acquired urinary tract infections. *Journal of Global Infectious Diseases* **6,** 135–136 (2014).

57. Healthineers, S. *Health Care Siemens* [Online; accessed 27-October-2018]. 2018. `\url{https://www.healthcare.siemens.it/point-of-care/urinalysis/multistix-10-sg-reagent-strips}`.

58. POWELL, H. R., MCCREDIE, D. A. & RITCHIEl, M. A. Urinary nitrite in symptomatic and asymptomatic urinary infection. *Archives of Disease in Childhood* **62,** 138–140 (1987).

59. Grude, N., Tveten, Y., Jenkins, A. & Kristiansen, B.-E. Uncomplicated urinary tract infections Bacterial findings and efficacy of empirical antibacterial treatment. eng. *Scandinavian Journal of Primary Health Care* **23,** 115–119 (2005).

60. Mcisaac, W. J., Moineddin, R. & Ross, S. Validation of a decision aid to assist physicians in reducing unnecessary antibiotic drug use for acute cystitis. eng. *Archives of internal medicine* **167** (2007).

61. Boonen, K., Koldewijn, E., Arents, N., Raaymakers, P. & Scharnhorst, V. Urine flow cytometry as a primary screening method to exclude urinary tract infections. eng. *World Journal of Urology* **31,** 547–551 (2013).

62. Jolkkonen, S., Paattiniemi, E.-L., Kärpänoja, P. & Sarkkinen, H. Screening of urine samples by flow cytometry reduces the need for culture. eng. *Journal of clinical microbiology* **48** (2010).

63. Van der Zwet, W., Hessels, J, Canbolat, F & Deckers, M. Evaluation of the Sysmex UF-1000i® urine flow cytometer in the diagnostic work-up of suspected urinary tract infection in a Dutch general hospital. *Clin Chem Lab Med* **48,** 1765–1771 (2010).

64. JR., W. D. C. & Rethwisch, D. G. *Fundamentals of Materials Science and Engineering - An integrated approach* (John Wiley & Sons, 2016).

65. Olympus. *Fluorescence Microscopy - Basic Concepts in Fluorescence* [Online; accessed june-2018]. 2018. `\url{https://www.olympus-lifescience.com/en/microscope-resource/primer/techniques/fluorescence/fluorescenceintro/.}`.

66. Segel, I. H. *Biochemical calcualtions - How to Solve Mathematical Problems in General Biochemistry* (John Wiley & Sons, 1976).

67. Godbey, W. T. *An Introduction to Biotechnology: The Science, Technology and Medical Applications* (Elsevier, 2014).

68. Estes, C. *et al.* Reagentless detection of microorganisms by intrinsic fluorescence. *Biosens. Bioelectron* **18,** 511–519 (2003).

69. Zhu, H., Sikora, U. & Ozcan, A. Quantum dot enabled detection of Escherichia coli using a cell-phone. *Analyst* **137,** 2541—2544 (2012).

70. Ammor, M. S. Recent Advances in the Use of Intrinsic Fluorescence for Bacterial Identification and Characterization. *J. Fluoresc* **17,** 455–459 (2007).

71. Karp, G. *Cell Biology* (McGraw-Hill, 1979).

72. De Jesus, A. J. & Allen, T. W. The role of tryptophan side chains in membrane protein anchoring and hydrophobic mismatch. *Biochim. Biophys. Acta BBA - Biomembr.* **1828,** 864–876 (2013).

73. So, P. T. & Dong, C. Y. Fluorescence Spectrophotometry. *Encycl. Life Sci,* 1–4 (2001).

74. TEALE, F. W. & WEBER, G. Ultraviolet fluorescence of the aromatic amino acids. *Biochem. J.* **65,** 476–482 (1965).

75. Ghisaidoobe, A. B. T. & Chung, S. J. Intrinsic tryptophan fluorescence in the detection and analysis of proteins: a focus on Förster resonance energy transfer techniques. *Int. J. Mol. Sci.* **15,** 22518–22538 (2014).

76. *Solvent and Environmental Effects*

77. Kilungo, A. P., Carlton-Carew, N. & Powers, L. S. Continuous Real-time Detection of Microbial Contamination in Water using Intrinsic Fluorescence. *Journal of Biosensors & Bioelectronics* **12,** 1–5 (2013).

78. Giana, H. & al. Rapid Identification of Bacterial Species by Fluorescence Spectroscopy and Classification Through Principal Components Analysis. *Journal of Fluorescence* **13,** 489–493 (2003).

79. Perinchery, S. M., Kuzhiumparambil, U., Vemulpad, S. & Goldys, E. M. The potential of autofluorescence spectroscopy to detect human urinary tract infection. *Talanta* **82,** 912–917 (2010).

80. Wikipedia. *Rayleigh Scattering —Wikipedia, The Free Encyclopedia* [Online; accessed 19-October-2018]. 2018. `\url{https://en.wikipedia.org/wiki/Rayleigh_scattering}`.

81. Wikipedia. *Raman Scattering —Wikipedia, The Free Encyclopedia* [Online; accessed 19-October-2018]. 2018. `\url{https://en.wikipedia.org/wiki/Raman_scattering}`.

82. Simões, J. C. G. *Continuous and Real-Time detection of Drinking-water pathogens.* MA thesis (University of South- Eastern Norway, 2018).

83. optics, E. *Master Source book* [Online; accessed june-2018]. 2018. `\url{https://www.edmundoptics.com/digital-catalog/master-source-book/}`.

84. Sezonov, G., Joseleau-Petit, D. & D'Ari, R. Escherichia coli physiology in Luria-Bertani broth. *Journal of bacteriology* **189** (2007).

85. Arana, I., Orruño, M. & Barcina, I. *How to solve practical aspects of microbiology 2. basic methods for microbial enumeration* [Online; accessed june-2018]. 2018. \url{https://ocw.ehu.eus/file.php/253/Temas/2_BASIC_METHODS_FOR_THE_ENUMERATION_OF_MICROORGANISMS.pdf}.

86. Direct, S. *Phosphate Buffered Saline - an overview. ScienceDirect Topics.* [Online; accessed june-2018]. 2018. \url{https://www.sciencedirect.com/topics/chemistry/phosphate-buffered-saline}.

87. Cerilliant. *Surine negative Urine control* [Online; accessed june-2018]. 2018. \url{https://www.cerilliant.com/shopOnline/Item_Details.aspx?itemno=1cedba31-c7f4-4ab1-ba9f-a02f3328d204&item=S-020}.

88. Github. *Github* [Online; accessed june-2018]. 2018. \url{https://github.com/lordloh/pbNotifier}.

89. NMEA. *NMEA sentences* [Online; accessed june-2018]. 2018. \url{http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp.}.

90. information, G. *NMEA data* [Online; accessed june-2018]. 2018. \url{https://www.gpsinformation.org/dale/nmea.htm}.

91. Richardson, M. & Wallace, S. *Getting Started With Raspberry Pi* (O'Reilly, 2013).

92. Flask. *Flask-web development one drop at a time* [Online; accessed june-2018]. 2018. \url{http://flask.pocoo.org/}.

93. HTTP. *HTTP* [Online; accessed june-2018]. 2018. \url{https://tools.ietf.org/html/rfc2616}.

94. Ngrok. *Ngrok* [Online; accessed june-2018]. 2018. \url{https://ngrok.com/}.

95. Mathworks. *MATLAB compiler* [Online; accessed june-2018]. 2018. \url{https://se.mathworks.com/products/compiler.html..html}.

96. Akbar, S. M., Sreeramulu, K. & Sharma, H. C. Tryptophan fluorescence quenching as a binding assay to monitor protein conformation changes in the membrane of intact mitochondria. *J. Bioenerg. Biomembr* **48,** 241–247 (2016).

97. Anwer, A. & al. Distinctive autofluorescence of urine samples from individuals with bacteriuria compared with normals. *Clinica Chimica Acta* **401,** 73–75 (2009).

98. Norazmi, N. & al. Uric acid detection using uv-vis spectrometer. *Materials Science and Engineering* **257,** 12–31 (2017).