UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF ELECTRONICS, TELECOMMUNICATIONS AND INFORMATION TECHNOLOGY

AUTOMATIC SCANNING AND PROCESSING OF BALLOT PAPERS

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the Degree of Engineer in the domain Technology and Telecommunication Systems
Study program: Telecommunications and Information Technologies

Thesis advisor(s):
Conf. Dr. Ing. Horia CUCU

Student:
Vlad UBLEA

Bucharest
2020

Appendix 1

University “Politehnica” of Bucharest **Anexa 1**
Faculty of Electronics, Telecommunications and Information Technology
Department **EAII**

DIPLOMA THESIS
of student **UBLEA V.C. Vlad , 441F-ELA.**

1. Thesis title: Automatic scanning and processing of ballot papers

2. The student’s original contribution will consist of (not including the documentation part) and design specifications:
During a voting process (i.e. voting the members for the Department Council in ETTI), the commission needs to count the votes visually, before sending the results to the central department. If there are many ballots, this process is very time consuming.
In this context, the aim of the project is to make this process more efficient by using scanning devices, computer vision algorithms for vote counting and a central server for storing the results.
The main program on the scanning device is responsible for fetching pictures from a webcam and processing them, by (i) detecting positioning markers on the paper, (ii) finding all shaded boxes and (iii) doing the correspondence between the votes and the candidate names. There can be many such devices, for example a Raspberry PI with a webcam, in many different voting locations. These will be used to scan the ballots and send the results to a central server, which aggregates them. The central server will generate a final report.
The ballot will be created by the central commission out of an Excel template containing the positioning markers, title headers and empty boxes with candidate names next to them. The commission will fill in the headers and candidates in the Excel ballot and finally print a number of ballots. The voters will express their vote by shading the empty box next to their desired candidate(s).

3. Pre-existent materials and resources used for the project’s development:
Python programming language, Raspberry PI, Webcam

4. The project is based on knowledge mainly from the following 3-4 courses:
PC, MC, DEPI

5. The Intellectual Property upon the project belongs to: U.P.B.

6. Thesis registration date: 2019-11-21 11:49:35

Student,

Thesis advisor(s),
Conf. dr. ing. Horia CUCU

Departament director,
Prof. dr. ing Sever PAȘCA

Dean,
Prof. dr. ing. Mihnea UDREA

Validation code: **135b4463a3**

STATEMENT OF ACADEMIC HONESTY

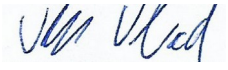
I hereby declare that the thesis "Automatic scanning and processing of ballot papers", submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of Engineer of Science in the domain Technology and Telecommunication Systems, study program Telecommunications and Information Technologies, is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited "as is" or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offense punishable by law.

I declare that all the results I present as coming from simulations and measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations and measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, *2020*

Student Name



(Student's signature)

TABLE OF CONTENTS

Table of Contents....6
List of Acronyms....6
Chapter 1: Introduction....9
1.1 Thesis Motivation....9
1.2 Proposed Objectives....9
1.3 The structure of the project....9
1.4 The current state of voting using computers....10
Chapter 2: hardware elements used to design the system....13
2.1 The Raspberry Pi Single Board Computer....13
2.1.1 The Cortex A7 Processor – ARMv7 Architecture....16
2.2 Peripherals....17
2.3 Camera....22
2.4 Mounting mechanism....25
Chapter 3: Software Technologies used....28
3.1 Programming Languages....29
3.2 The Python Programming Language....30
3.3 Linux....33
3.4 Excel....34
3.5 Networking....34
3.5.1 The File Transfer Protocol (FTP)....36
3.6 Digital image representation....38
3.6.1 Image storage....40
Chapter 4: Practical realization of the work....41
4.1 Creating the Excel template....41
4.2. Camera capture....44
4.3. Program workflow....45
Chapter 5: Conclusions....54
5.1 General Conclusions....54
5.2 Personal Contributions....54
5.3 Future Work....54
References....56
Annex 1....57

LIST OF FIGURES

Figure 2.1 – Raspberry Pi 2.....13
Figure 2.2 – 16 pin LCD to I^2C.....18
Figure 2.3 – I^2C data communication.....18
Figure 2.4 – Button bouncing.....19
Figure 2.5 – GPIO connector.....20
Figure 2.6 – LED configuration.....21
Figure 2.7 – Button configuration.....21
Figure 2.8 – LCD configuration.....21
Figure 2.9 – Buzzer configuration.....21
Figure 2.10 – Bayer color filter.....22
Figure 2.11 – Dynamic range of a camera.....24
Figure 2.12 – Visible light wavelengths.....24
Figure 2.13 – Photodiode sensitivity in relation to wavelength.....24
Figure 2.14 – Camera mounting assembly.....26
Figure 3.1 – Excel data example.....34
Figure 3.2 – TCP/IP model.....35
Figure 3.3 – Client-Server model.....37
Figure 3.4 – Digital image representation.....38
Figure 3.5 – Image sampling and quantization.....40
Figure 4.1 – Voting ballot in Excel form.....42
Figure 4.2 – Computer generated square.....43
Figure 4.3 – Camera captured square.....43
Figure 4.4 – Matrix representation of computer generated square.....43
Figure 4.5 – Matrix representation of camera captured square.....43
Figure 4.6 – Errors due to image rotation.....44
Figure 4.7 – Correct identification of squares.....44
Figure 4.8 – Errors due to lighting conditions.....45
Figure 4.9 – Main program flowchart.....46
Figure 4.10 – Detection of markers with tolerance regarding x-axis positioning.....49
Figure 4.11 – Calculation of center.....50
Figure 4.12 – Applying offsets to squares.....51
Figure 4.13 – Algorithm for processing entire paper.....52

List of Acronyms

ARM: Advanced RISC Machines
CCD: Charge Coupled Device
CMOS: Complementary Metal Oxide Semiconductor
CPU: Central Processing Unit
FTP: File Transfer Protocol
GPIO: General Purpose Input Output
GUI: Graphical User Interface
Hz: Hertz
IP: Internet Protocol
ISP: Internet Service Provider
IT: Information Technology
JPEG: Joint Picture Expert Group
LCD: Liquid Crystal Display
LED: Light Emitting Diode
MP: Mega Pixel
PWM: Pulse Width Modulation
RAM: Random Access Memory
RISC: Reduced Instruction Set Computer
SBC: Single Board Computer
SOC: System On a Chip
TCP: Transmission Control Protocol
UDP: User Datagram Protocol
USB: Universal Serial Bus

CHAPTER 1: INTRODUCTION

1.1 THESIS MOTIVATION

Most elections, whether the ones inside educational institutions (for example in faculties, voting for students' councils or professors' departments leaders) or countries affairs (more specifically, in politics, for choosing candidates in the parliament or mayors of towns), are paper based and can take a long time to scan using people, and there is potential for errors when introducing the results on a computer using a keyboard (accidental keypresses). Generally the elections are counted at night when the voting ends (from 9PM to 3AM for example). This project aims to reduce the time and errors of such verification. Paper voting is still popular compared to electronic voting since it is more secure, since it offers physical proof of the vote and it can be checked afterwards. Also, there are no hacking problems to consider compared to purely electronic voting.

1.2 PROPOSED OBJECTIVES

For this project the following objectives have been proposed:

1. Create a template for the ballot paper in Excel, a graphical common spreadsheet editing program used and known by many users for its simplicity, that is easy to use by people and easy to process by computer.
2. Create software for processing the data in the Excel file and the data coming from a PNG file exported from the Excel file, representing the scanned paper.
3. Create a mounting device that holds the papers and webcam fixed.
4. Modify the software to work for real world images.
5. Adapt for use with the Raspberry Pi single board computer, using the GPIO connectors for peripherals (LCD, buttons) to allow the working of the program without a monitor, mouse and keyboard to save costs.

1.3 THE STRUCTURE OF THE PROJECT

This project aims to develop a system for automatic processing of voting ballots using computer vision.

The second chapter will illustrate the hardware components of the project, mentioning the required components and mounting steps to create the physical assembly.

The third chapter will mention notions about the software technologies used, for example the programming languages available and the decision of which one to be used in the project, the Excel spreadsheet editor and the networking foundation.

The fourth chapter will outline the development process of the project, both in respect to software and hardware, and the problems encountered and the solutions found.

The fifth chapter will focus on the concluding remarks made after the work on the project was done.

1.4 THE CURRENT STATE OF VOTING USING COMPUTERS

Nowadays, it is increasingly common to see very advanced algorithms designed to recognize certain patterns. However, the downside to these algorithms is that they require increased computing power, which means more expensive hardware to support it. The present work aims to achieve a fast but still reliable way of registering votes, using a special voting ballot template.

The reason why a paper based system is still desirable, instead of a purely electronic approach, using for example a webpage, is the fact that paper votes are easier to verify in terms of authenticity.

First of all, using a paper voting system, the person must move to a voting center, and present his or her ID card. This is more secure than establishing a website, where anyone from anywhere in the world can access it, and could potentially bypass the login credential verification by using various attacks.

Secondly, setting up a website is time consuming, especially if designed for such an important application. A database must be made and integrated with the website containing all the information regarding the voters. The security system must be advanced enough to block potential attackers but easy to use at the same time. Additionally, some people may consider electronic voting to be more complicated, since they have to input credentials and may encounter technical problems, either with the website or their device’s internet connection.

Thirdly, paper votes can be used as a physical proof of an election’s results, which is harder to counterfeit or modify maliciously. If there is for example a power outage, or there are Internet problems, the votes can still be counted manually by people, proving their utility as a backup.

To conclude, a combination of old paper based voting with new computer recognition software is a good solution to include both the benefit of ease to use and security, and the benefit of faster result counting times.

The proposed solution for this project is designed in such a way that there are very few variables involved in the scanning and processing procedure. Therefore, instead of designing an algorithm that is very complex for various situations (for example: the pictures are taken from different angles and the angle must be calculated and compensated, or from different distances, which requires cropping), the project aims to have very little variations, so the program will not have to compute extra tasks, which saves time, especially if the program is deployed on small, portable systems, and also does not have as many points of failure. This is why a fixed camera mounting system was designed and why the template has positioning markers, to ensure that the program will have to do only some basic image processing tasks.

The work proposes to have achieve good performance with respect to both processing time and accuracy. The positioning markers are very easily detectable using very simple image processing algorithms with good accuracy. Thus, the program can be run on small power systems in order to save the price, which is sometimes the most important factor when deciding on what system to choose.

CHAPTER 2: HARDWARE ELEMENTS USED TO DESIGN THE SYSTEM

2.1 THE RASPBERRY PI SINGLE BOARD COMPUTER

The Raspberry Pi is a single board computer (SBC) (figure 2.1) with small dimensions, approximately the same as a credit card, at an accessible price for the general public. The Raspberry Pi was created by the Raspberry Pi Foundation, a foundation registered as an educational and charity one in Great Britain. The scope of this foundation is to advance the education of children and adults, especially in the IT domain, computer science and adjacent domains. [1]

Figure 2.1 – Raspberry Pi 2

Since it is a computer, it can be connected to a monitor or a television and can be controlled

using a keyboard and mouse. Unlike microcontrollers (like the Arduino for example), the Raspberry Pi has a dedicated Linux based desktop operating system called Raspbian, based on Debian, and doesn't require an external computer to program it (unlike the Arduino which requires programming on a separate PC using the Arduino IDE). This SBC allows people with any level of programming background to explore the world of programming and information processing using different programming languages like Scratch or Python. The Raspberry Pi has performance similar to an entry level desktop, meaning it can perform basic tasks like Internet browsing, playing video games, managing documents and spreadsheets.

Additionally, the Raspberry Pi has the ability to interact with the outer world; it was and continues to be used in a great number of digital projects, ranging from audio applications (home-made pianos with push buttons), weather stations (with different temperature and pressure sensors) or security surveillance (with infrared sensors). One of the primary reasons this system was invented was to allow children to discover the world of programming, to understand how computers work and to learn how to use them.

The most used connector for such home projects is the General Purpose Input Output header (GPIO), which offers the Raspberry Pi the ability to send and receive digital signals. It uses 3.3V input and output signals, but also has a separate 5V power pin for devices requesting more voltage, for example LCDs. The GPIO also supports protocols like I2C, SPI, UART. As with any microcontroller, the power that can be drawn or sinked into the pins is limited, usually a few milliamperes. Thus, more power demanding devices require a separate power supply.

Currently, there are 5 generations of Raspberry Pi on the market, in chronological order: 1,2, Zero, 3,4. Each generation has multiple models, as presented in table 1:

Family	Model RAM	Ethernet	Wireless	GPIO	Released	Discontinued	CPU
Raspberry Pi 1	B (512 MB)	Yes	No	26-pin	2012	Yes	1 Core, 700MHz
	A (256MB)	No			2013	Yes	1 Core, 700MHz
	B+(512 MB)	Yes		40-pin	2014		1 Core, 700MHz
	A+(512 MB)	No			2014		1 Core, 700MHz

Raspberry Pi 2	B 1 GiB	Yes	No		2015		4 Cores, 900 MHz
Raspberry Pi Zero	Zero 512 MiB	No	No		2015		1 Core, 1 GHz
	W/WH 512 MiB		Yes		2017		1 Core, 1 GHz
Raspberry Pi 3	B 1 GiB	Yes	Yes		2016		4 Cores, 1.2 GHz
	A+ 512	No			2018		4 Cores, 1.4 GHz
	B+ 1 GiB	Yes			2018		4 Cores, 1.4 GHz
Raspberry Pi 4	B (1 GiB)	Yes	Yes		2019		4 Cores, 1.5 GHz
	B (2 GiB)						
	B (4 GiB)						

Table 1

Although different from a processing power standpoint, all RPi models have some common software and hardware characteristics. Firstly, all the processing happens on a single chip, hence the name system on a chip (SOC), made by Broadcom. This chip contains a central processing unit (CPU) that is ARM compatible and a graphics processing unit, namely the Broadcom VideoCore IV. The speed and number of cores of the CPU varies according to the model, as the amount of RAM. Since the Raspberry Pi does not have any non volatile memory on board, a microSD (or SD card for older models) is needed to store an operating system and user data. All models include at least one USB port for peripherals like mice, keyboards, USB memory sticks, webcams and an HDMI video and audio output for digital monitors. There is also a 3.5mm

composite jack for analog video and audio. Most models have built in networking capabilities, either through an RJ45 Ethernet jack or using WiFi 802.11n.

2.1.1 The Cortex A7 Processor – ARMv7 Architecture

The ARM architecture forms the basis for any ARM processor. Over time, the ARM architecture evolved by including architecture attributes that satisfy the increasing demands regarding new functionality, better security and higher performance. The ARM architecture supports implementations in a large range of performance points, establishing itself at the peak of many markets. This architecture type is used in a large range of applications, from simple implementations to advanced implementations using cutting edge micro-architecture techniques. The key attributes of the ARM architecture are the implementation dimensions, the performance and the small power consumption.

The ARM architecture is similar to a RISC architecture, incorporating the following typical attributes:
-Uniform register storage architecture, meaning the data processing takes place only in the content of the registers, not directly in the memory
-Simple addressing modes, with storage addresses determined only by the content of the registers and the instruction field.

Additionally, the ARM architecture bring the following improvements:
-Ability to control the ALU and the shifter in the majority of data processing instructions to maximize their usage
-Addressing modalities with auto incrementation and auto decrementation to optimize program loops
-Loading and storing multiple instructions to maximize the data quantity
-Conditional execution of the majority of the instructions to maximize the execution rate.

These improvements made to a base RISC architecture allows ARM processors to obtain a good balance between high performance, low sizes of written code, low power consumption and low silicon areas.

The processor inside the Raspberry Pi is a multimedia Broadcom BCM2836 system on a chip (SOC). Most of the system components, including the CPU and GPU, together with the audio unit and the hardware communication part are housed together on a single chip. The random access memory (RAM) is located on a separate chip, as with the networking controller. This SoC construction, along with the ARM instruction set architecture (ISA) differentiates it from ordinary laptops or desktops.

The ARM architecture, developed by Acom Computer around the end of the 1980’s, is not so common on desktops. Instead, it excels in the area of mobile equipment, as an example, most modern Android and iOS based smartphones have a processing core based on the ARM architecture. Having a reduced instruction set and a low power consumption, an ARM processor

is the ideal choice for portable devices, instead of using common desktop x86 based processors.

Since the BCM2836 is based on the ARM architecture, this enables its small size: low power consumption means it can be powered from a phone charger or portable battery pack using an USB connector at 5V with a maximum current of 3A for the most powerful models (15W under full load). Another advantage is that no active cooling is required, keeping the cost, noise and size at low levels.

2.2 PERIPHERALS

LCD

An LCD is an electronic display module which uses liquid crystals and a backlight to produce a visible image. The 16×2 LCD display used in this project is a basic module commonly used for outputting information from microcontroller or single board computer projects. The 16×2 refers to a display with 16 characters per line in 2 such lines. In this LCD each character is displayed in a 5×8 pixel matrix. [7]

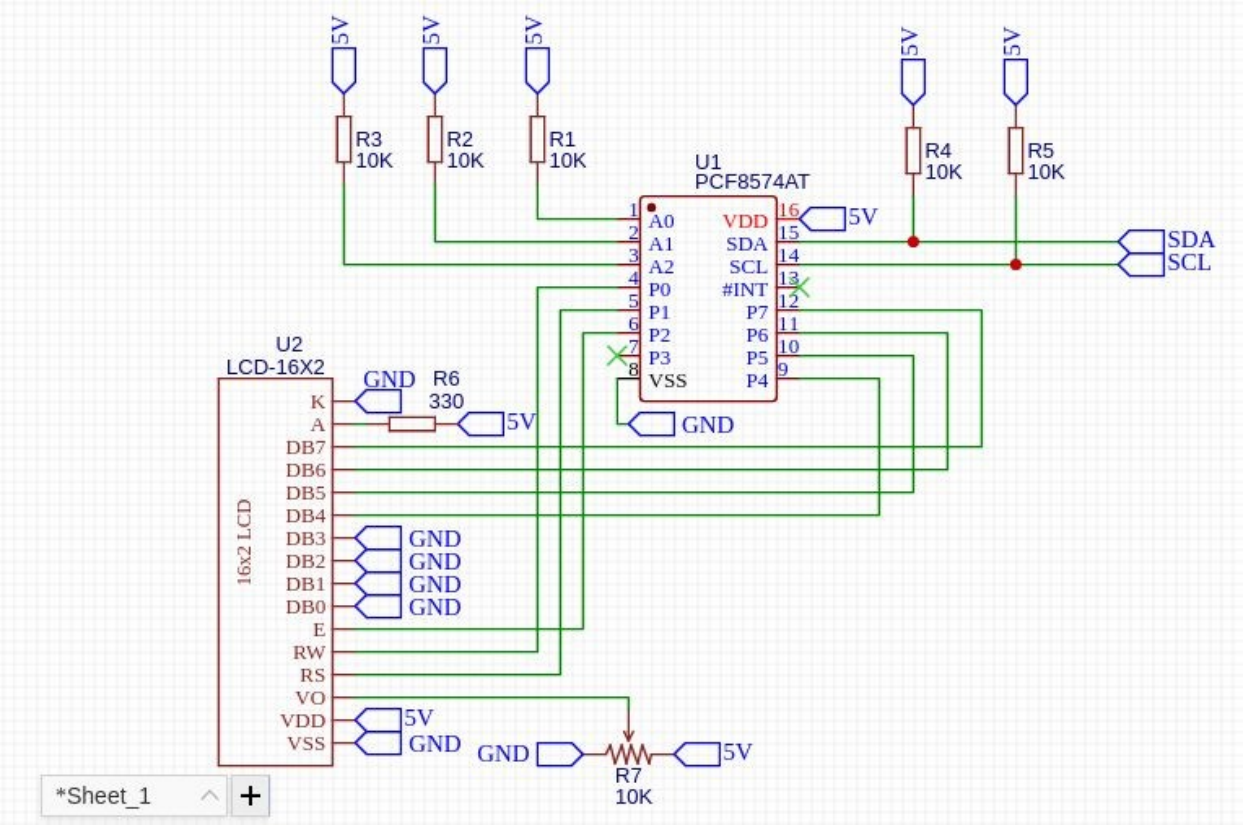


Figure 2.2 – 16 pin LCD to I²C

The LCD has 16 pins, but we use an I2C interface in order to have fewer connections to the Raspberry Pi in order to simplify manufacturing and to allow more free pins for other devices. An example schematic is presented in figure 2.2, where a special integrated circuit named PCF7564AT is used to convert the I²C signals into digital signals that are recognizable by the LCD. The connection is done using only 4 data pins, DB0 through DB3 for even lower complexity. The 10KOhm potentiometer is used to adjust the contrast.

There are only 4 pins for the I2C interface, 2 for power (+5V and Ground) and 2 for the I2C communication, SCL (the clock signal) and SDA (the data signal). The I2C protocol is synchronous and serial, meaning that the clock dictates the flow of data, and only one bit is sent at a time. [6][15]

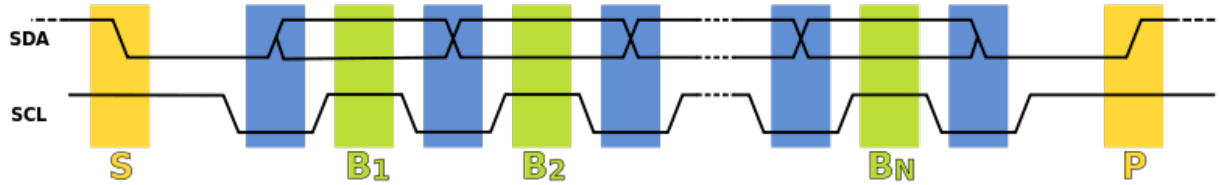


Figure 2.3 – I²C data communication

The LCD is used to display information about the capturing process (if the scan is successful or if there are errors encountered).

Button

The push buttons are used to control the flow of the main program.
A button simply alters the state of the circuit between an open or short circuit. A resistor is used in series to limit the current, because a simple short circuit would cause a high current to be drawn from the Raspberry Pi which may cause problems. In software, the rising edge is detected when pressing the button. A bounce timer is added because a button press is not perfect.

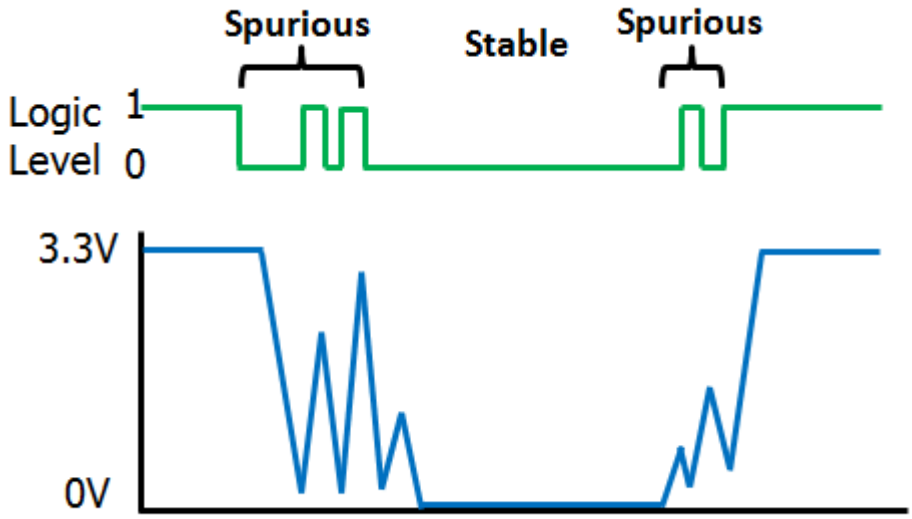


Figure 2.4 – Button bouncing

In figure 2.4 such behavior can be seen, where a button press can be interpreted as multiple ones, due to the fact that when pressing the button, the switch inside may be temporarily in a position that is very close to the on and off position simultaneously. This poses a problem, since if the user presses the button from an angle, the program will take multiple pictures. Thus, adding a delay will solve the problem, and allow button presses only after one picture is taken and processed. [8]

LED

The Light Emitting Diodes are used for easy visual signalling of the program status, namely the red color will indicate an error in identifying the paper sheet (for example, the markers are not detected properly, due to an improper positioning), the yellow colour will announce that the sheet paper has been successfully scanned, but there are invalid votes as defined by the limit of votes per election in the Excel file. Lastly, the green colour will signal the total success of the scan; all the votes are valid. The LEDs are connected to the Raspberry Pi GPIO directly, since

they are low power ones. 2kOhm resistors are added in series to limit the current to around 7mA, which is safe and ensures a prolonged lifespan.

Buzzer

The speaker is used to acoustically warn the user of an error, or if he/she may proceed. The selected buzzer is an active one, meaning we only need to apply a DC voltage for it to work, we don't have to worry about generating a PWM signal to apply to the input. The PNP transistor makes the buzzer to beep when the input is LOW. This means that a pin that is by default high needs to be connected, otherwise, the buzzer will start beeping when the Raspberry Pi is turned on and stop only when the program is loaded. To combat this problem, looking at the documentation to see which pins are by default high and using them is the solution. I ended up using pin number 26, which can be used for SPI or GPIO functionality.

General Purpose Input Output interface

The GPIO uses 3.3V digital outputs. 5V power pins are also available for powering the LCD, but they cannot be controlled in software. It also supports I^2C protocol interfacing, among other functionalities.

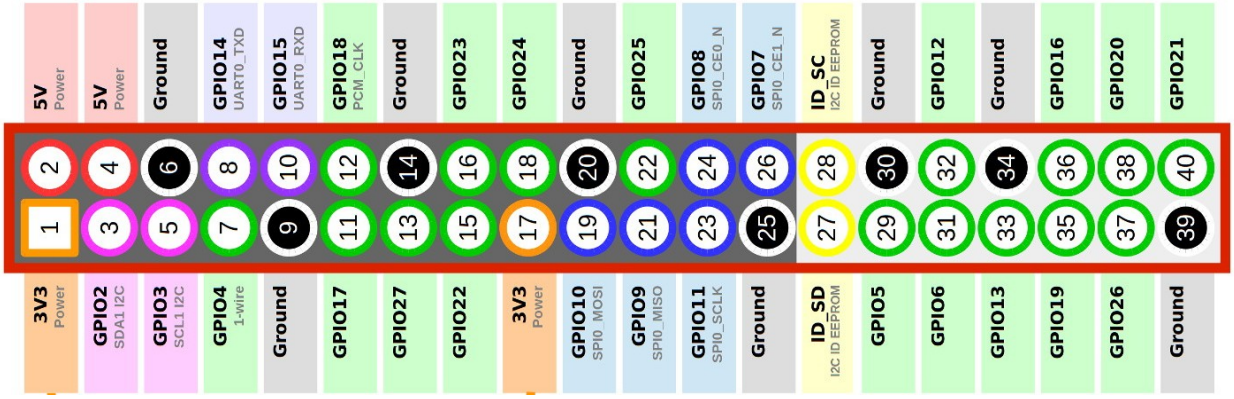


Figure 2.5 – GPIO connector

As mentioned, the LEDs are connected to resistors in series to limit the current.

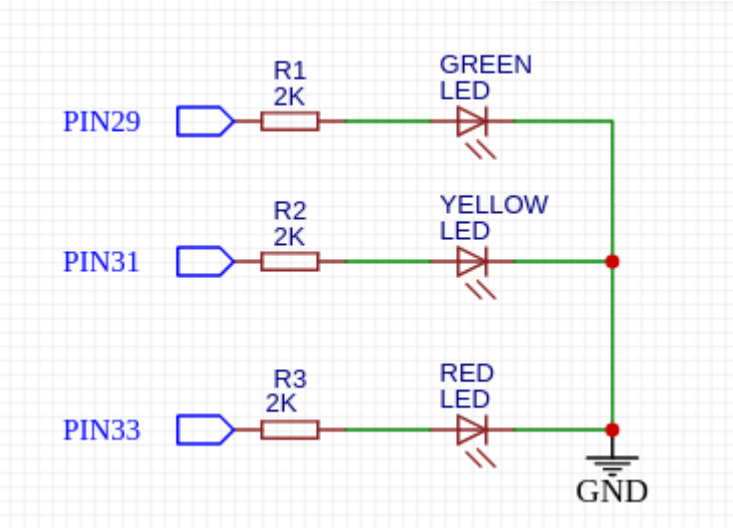


Figure 2.6 – LED configuration

The buttons have resistors in series to limit the current, and it is supplied by the on-board 3.3V header.

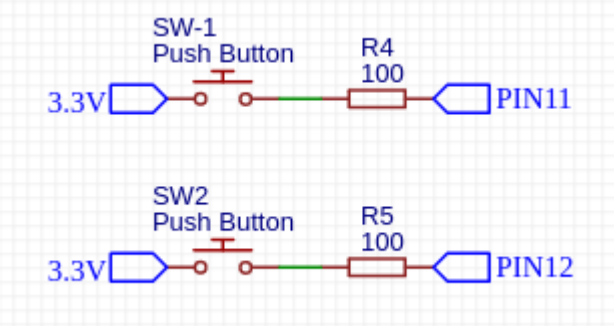


Figure 2.7 – Button configuration

The pins 3 and 5 are specifically designed for I^2C communication. The LCD needs 5V to work properly.

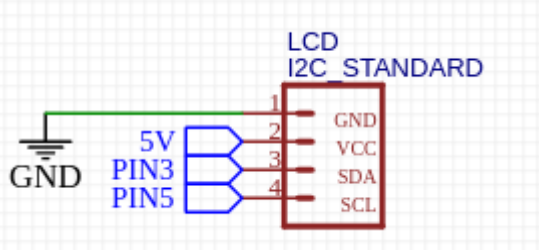


Figure 2.8 – LCD configuration

The buzzer requires only a 3.3V supply and a high or low signal to deactivate or activate it.

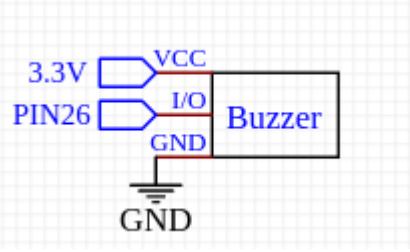


Figure 2.9 – Buzzer configuration

2.3 CAMERA

The camera used for this diploma project is the Logitech C270 HD webcam. It uses a CMOS sensor for capturing images. It has a resolution of 640×480 pixels, using a 4:3 aspect ratio. The reason a normal PC webcam with an USB connection was chosen is due to the fact that it is compatible with either the Raspberry Pi or a normal desktop or laptop. Thus, if in the voting section there is already a PC present, the webcam can be connected and the program can be run on it, without the need of a Raspberry Pi, thus saving costs. If there is no computer, the Raspberry Pi is a low cost alternative to a PC. [11]

CMOS image sensors are less expensive to produce than their counterpart, the CCD image sensors, since they use small digital circuits using less power, instead of high power analog circuits. The main components inside a CMOS image sensor are a photodiode and one or more transistors. The photodiode accumulates charge carriers when it is exposed to light. The transistors are used to select individual pixels via the photodiodes and amplify the signal. [5]

A problem with photodiodes is that they are unable to detect photon colors. With silicon photodiodes, any photon that is in the visible light spectrum has enough energy to activate the photodiode by creating an electron-hole pair in the intrinsic layer. Thus, only a monochrome image can be obtained.

A solution to this problem is to use color filters before the diode. Such filters can allow only a certain wavelength to pass, while blocking (reflecting) the others. These filters are usually composed of the base colors: red, green and blue, because all other colors can be created by additive color mixing. With regard to colors that are between the primary colors, for example purple, which is a combination of red and blue, both the red and blue wavelengths will pass through both filters.

The most used color filter used in digital cameras is the Bayer color filter:

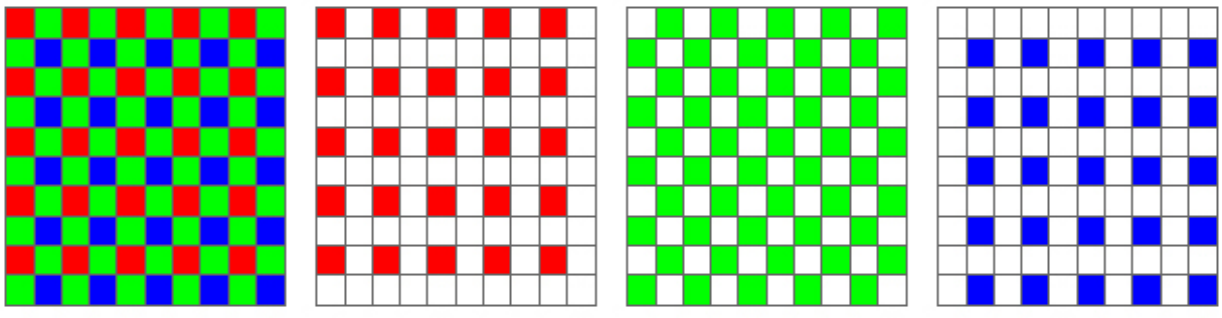


Figure 2.10 – Bayer color filter

As seen above, 50% of the spectrum is allocated to green colors, while the rest are divided equally in two halves; 25% for red and 25% for blue. The reason this layout was used is because of the nature of the human eye, which is more sensitive to green wavelengths, which means that green will contribute more to the perception of brightness and contrast. A drawback of this

approach, however, is that each square, or pixel, can only detect one color, thus interpolation is used by the camera processing unit to calculate the missing color information based on the neighboring pixels.[12]

One problem encountered is the noise that can be present in an image. This noise affects the quality of the useful signal and gives artifacts to captured images. A few reasons why the noise occurs:

-Thermal radiation: a photodiode can also be excited by heat. The solution is to ensure that the camera is not placed under direct sunlight or next to heat producing devices.

-Fixed pattern noise: caused by slight production variations which in turn cause differences in the responsiveness of photodiodes. This causes a constant noise pattern in the picture. The solution is incorporated into some digital cameras, that use a technique known as “dark frame subtraction”. The way it works is by taking two pictures, one with the shutter open and one with the shutter closed, and subtracting the two images to obtain an image without any fixed pattern noise.

-Poisson noise: a light source does not emit photons uniformly, instead it is emitting according to a certain probability that can be calculated using the Poisson distribution. The solution is to capture more signal, meaning longer exposure times, but due to the fact that the effect is negligible, and that longer exposure times increase the likelihood of more problems appearing (for example sensor saturation), it can be ignored.

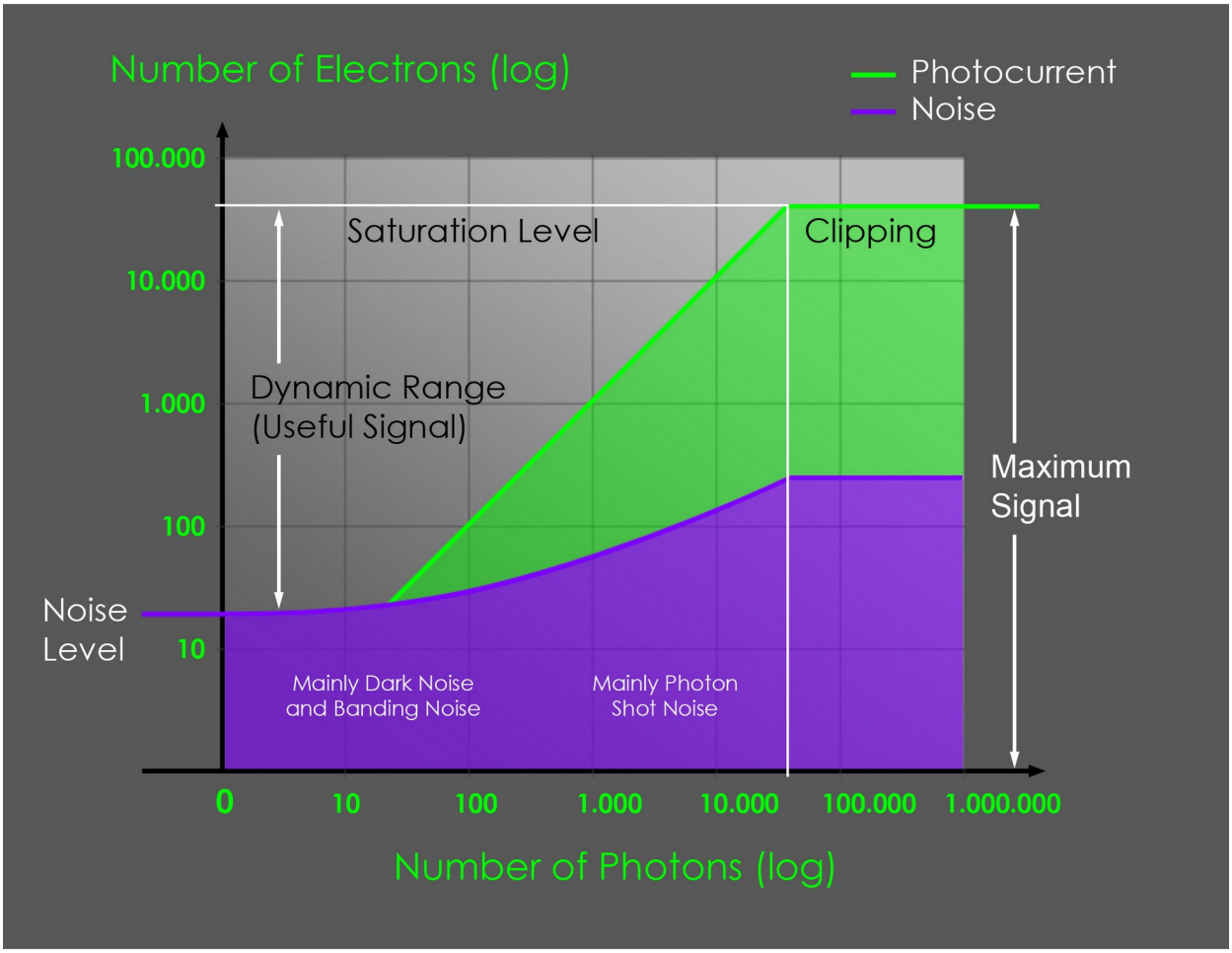


Figure 2.11 – Dynamic range of a camera

As it can be noticed in the above figure, a balance must be achieved regarding the background lighting. In a dimly lit room, the picture will contain a lot of noise, but in a situation where the light source is very bright and next to the sheet of paper, the sensor will start to enter the clipping region.

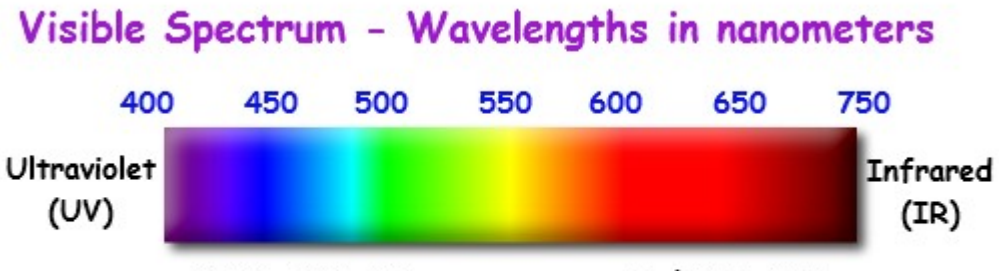


Figure 2.12 – Visible light wavelengths

Another aspect to consider is that the visible light required to be captured by the camera resides in the 400-750nm range, by observing figure 3.5.

By quickly looking at a datasheet of a usual silicon photodiode we can notice its characteristics in figure 3.6:

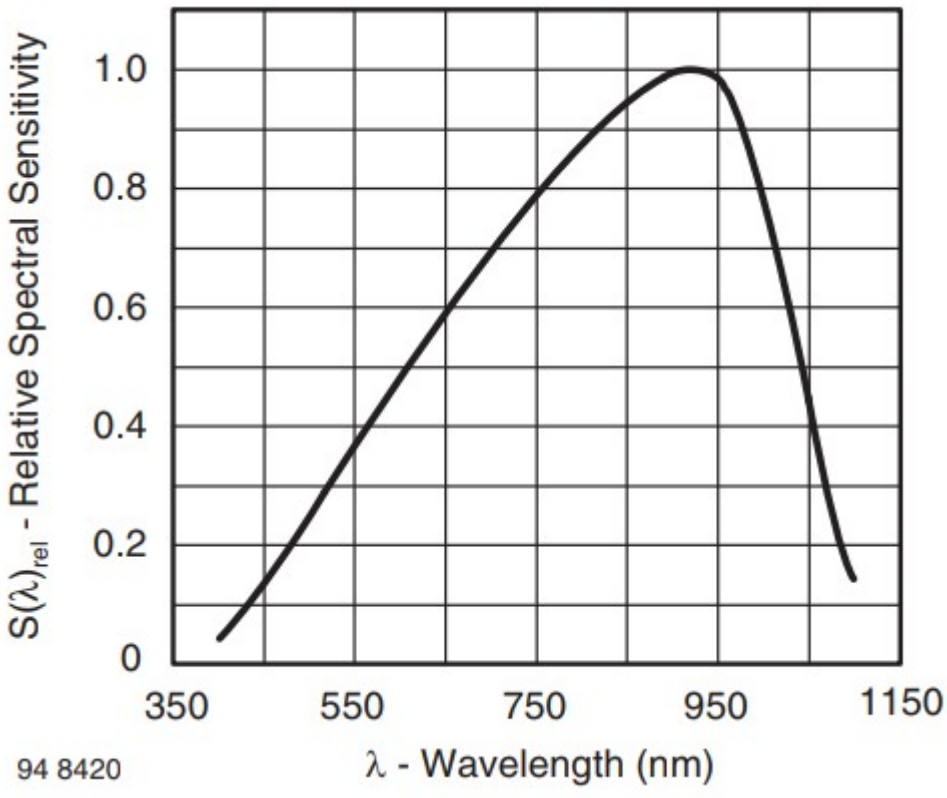


Figure 2.13 – Photodiode sensitivity in relation to wavelength

To prevent unwanted activation of the photodiode, special filters that block light outside the visible light spectrum are added, namely infrared and ultraviolet. Otherwise, we would get inaccurate reading of the pixels.

2.4 MOUNTING MECHANISM

For the non-electrical hardware, we use the following objects:

1. A paper tray. The paper tray must be flat, in order to have equal distances across the image when it is scanned. Additionally some walls are required for easy positioning of the paper in the corner of the tray.
2. Wood beams, 100×15×15 mm in size.
3. Glue for fixing the camera to the wood beam.

Figure 2.14 – Camera mounting assembly

The camera needs to be positioned 20cm away from the start of the paper sheet in order for the sheet to be centered. Also, the height from the center of the paper sheet should be 52cm to achieve a good focusing.

The reason why this mechanism was chosen is to allow the user plenty of space for movement under the camera where the sheets would be positioned. Additionally, such a mechanism can be produced cheaply using regular hardware store supplies, like for example wood or metal beams, thus excluding the cost implicated with custom 3D printing, which requires expensive hardware.

CHAPTER 3: SOFTWARE TECHNOLOGIES USED

3.1 PROGRAMMING LANGUAGES

A programming language is a collection of words and rules for a computer in order to perform certain tasks. The role of it is to allow a programmer to transmit to a computer, via the use of programs, the actions it must perform in an exact manner specifying also the order of operations and the data to use.

We can classify the programming languages into two large categories:
1.Low level languages, dependent on the hardware being used

-The machine language, which is regarded as a primitive, lowest level, hardware dependent programming language. It is rarely being used since it is error prone and tedious, the programmer having to manually manage individual addresses allocations, for example. The instructions, represented as a series of binary 1s and 0s, differ from one computer to another.

-The assembly language, based on a piece of software called assembler, converts a program written into a special syntax, usually unique to each microprocessor, into machine code. This language allows the programmer great control over everything that happens inside the computer, with the help of easily memorisable syntax words (for example ‘mov’ is used for moving data, ‘jmp’ is used to jump to a specific label)

2. High level languages, independent on the hardware being used
Initially called autocodes, they rely on a compiler to convert a source text program code to an executable program in lower level code that can be understood by the computer. The first ones were for example, COBOL, for economic use, introduced in 1959, or FORTRAN, for mathematics and science use, introduced in 1957.

Procedural programming (i.e. Pascal, C), introduced in the 1970’s, appeared with the purpose to create programs capable to be safe in functioning for a long time. It is a programming method

that decomposes complex problems into simpler subproblems called modules. This approach has the name “top-down”. Any algorithm can be composed from only three control structures, according to the Bohm and Jacopini theorem:
-The sequential structure – the sequence
-The alternative structure – the decision
-The repetitive structure – the loop

Object oriented programming (OOP) (i.e. C++, Java, Python), introduced in the 1980’s, proposes the grouping of data and codes from one structure into individual code blocks that interact with each other. Objects in OOP are representations of real life entities most of the time, and that led to a better understanding of the programs and how to debug them.

Another important characteristic that is used to classify programming languages is by the modality of translation:

-Compiled languages: for example C, C++, Java, Pascal. Using a compiler, the source text code program is translated into machine language and produces a separate file which can then be executed. The advantage is that the programs run faster, because it is only compiled once, and then an optimized machine code is produced, which can be run multiple times. The compiled file can then be used on other machines, without the presence of a source file, which is advantageous for intellectual property protection, although there are decompilers that can create a source code from the compiled file. Another advantage is that during the compilation process, the compiler will check the entire code and can warn users about errors, so the program can be executed only if it is correct from a syntax perspective at least, it does not prevent runtime errors, like division by 0, stack overflows, etc..

-Interpreted languages: For example PHP, JavaScript, Python, Matlab. These languages rely on an interpreter, which converts the source code to machine code line by line. They are slower than compiled languages, because each time the program is executed, the code must be translated. Another disadvantage is that the code is verified for errors as it is being executed, which may lead to time losses, because the code has to be run in order to be verified. Nevertheless, modern computers are able to perform the translation quickly, using various techniques like prefetching and caching. Additionally, if the program is modified frequently, the difference between compiling it after a change or interpreting it after a modification becomes less apparent.

Therefore, as a conclusion, the ideal programming language for many use cases, except in time critical applications where the code must run at a very specific timing, is a high level one, since the time spent developing the application is lower due to more friendly syntax and less constraints to be considered, for example managing the memory addresses and registers. Object oriented programming languages are easier to use since they are easier to understand and easier to modify, using the philosophy of polymorphism for example, an object can be added or changed requiring major changes to the original program. Since the computers today are very powerful compared to when programming emerged, the time difference between compiled and interpreted languages can be considered negligible.

3.2 THE PYTHON PROGRAMMING LANGUAGE

Python is a very popular programming language in the IT domain, according to the latest statistics on Github. This programming language appeared in the year 1991 and was created by Guido van Rossum. Currently, Python is used by companies like Google and Yahoo for web applications but also for embedded projects. YouTube and Amazon are websites that were developed using Python. A big advantage that led to the high popularity of the Python programming language is its implementation using CPython. CPython can be defined as both an interpreter and a compiler as it compiles Python code into bytecode before interpreting it. CPython is included in operating systems like Linux and Mac OS X. Due to the fact that it is interpreted and the CPython utility can be found on many devices, it is possible to run a Python program on any operating system without any additional adjustments.

A fundamental concept that is at the basis of Python programming language is indentation. The indentation is the placement of the code on lines under a set of rules which helps the IDE interpret the code written by the programmer. This modality of writing code has helped reduce the number of lines a programmer has to write to obtain the same algorithm (for example, omission of curly braces "{,}" in loops like in the C programming language), but at the same time it forces the maintaining of a clean and easy to read code. A well written and easy to read code is a serious argument for a company to adopt and code in that programming language since it makes team work more efficient.

From a programming type perspective, Python can be used both in functional object oriented programming and in procedural programming, especially used for embedded systems. Similarly to Java, the memory administration is managed through a service called garbage collector, which automatically deletes from memory variables that are not being used anymore, unlike C, which requires manual management of memory data.

The concept "Batteries included" suggests that any programming language, no matter the mode it was thought and projected, requires a set of libraries to have a practical utility. The popularity of Python resulted in an important advantage of this open-source programming language since it led to a very large quantity of libraries that can be accessed by any programmer. The Python Application Programming Interfaces (API) offer a wide range of functionalities from basic ones (modifying a string) to work with processes and threads. An example can be implementing a Python code with the scope of executing commands that are usually done with Matlab (which is a non-free, proprietary software), like for example working with matrices and plotting graphs. This is possible with the help of "matplotlib". Another example is the wxPython package which offers methods and data structures specific to the creation of a graphical user interface.

Another characteristic that makes the code written in Python to be more compact is its permissive nature of the language to allow not specifying the type of a variable when declaring it. With the help of the interpreter, the type of the variable is given by the content the programmer gives it. For example:

```
variable_a=10
#the interpreter will consider this variable of type integer
variable_b=”this is a string”
```

#the interpreter will consider this variable of type string

Python does not allow operation with different types of objects and has the concept of mutable and immutable variables. An immutable variable is a variable whose contents cannot be modified after being initialized. For example:
variable_c=”HOME”
variable_c[2]=”L”
#will result in an error: TypeError: 'str' object does not support item assignment

The term “sugar coding” appeared recently in programming ,replacing the functions from C like “strcmp” that had the function of checking if two strings are equal, with an expression used in Python:
if(str1 == str2):

instead of

if(strcmp(str1,str2)==0)

Functions in Python are different from classical ones found in C and Java. In C, the function can either be void, if it returns nothing, or of a specific type in case the function has to return that type in the main program or in another function. In Python, the functions are declared with “def” followed by the name of the function and the parameters it has to receive inside brackets, like other programming languages. The difference is that the programmer does not need to think ahead if his function needs to return something, it can actually return multiple types if there is an if-else statement, including void.

Python 2 vs Python 3

The third version of the Python programming language was launched in 2008, 8 years after the second version. The principal scope of the new version was to eliminate the redundancy that appeared in the previous version. This decision meant that duplicate structures were eliminated.

Although this change was meant to bring improvements, Python 3 created problems and was slowly adopted, since most projects written in Python 2 were incompatible with the newer version, and needed to be adapted in order to function. A part of the developer community was reluctant to migrate their software to the new version and it took lots of time and effort before large applications were ported to the new version. A patch called 2to3 was made in order to “translate” older programs written in Python 2 to Python 3.

A few of the differences between the two versions will be presented below.

A change regarding data structures was the elimination of the “long” data type, encountered also in C. In both C and Python 2, long was used to store an integer number between [-2000000;2000000], while int could store an integer number between [-32000;32000]. With the introduction of Python 3, int expands its domain and renders the use of long obsolete.

Another difference between the two versions consists in the way they can display text in the console. In Python 2, print was not considered a function, instead it was considered a command. Henceforth, to print a string the user would type:
print “Text”
In the new version, the only way to print text into a console is by using print as a function, and the text to be displayed is received as an argument. Therefore, to print a string the user would type:
print(“Text”)

Yet another difference that caused problems porting the programs from version 2 to version 3 of the Python programming languages was comparison. In the second version, the comparison was made without taking into account if the object belonged to the same class or not. For example:
0<”p“ #would return a true boolean value, comparing 0 as an int with “p” as a string in Python 2
0<”p” #would return an error: TypeError: '<' not supported between instances of 'int' and 'str'

Due to its ease of use, vast libraries and multi-platform compatibility (Windows, Mac OS, Linux 32bit (x86), 64bit (x86_64) or ARM), Python was the programming language of choice for this project.

The IDEs used to develop this project were PyCharm Community Edition, on my laptop, and Geany on the Raspberry Pi, for its reduced memory footprint. The Python version used was obviously Python 3, since Python 2 has been declared End Of Life, not receiving any more bug fixes and improvements.

Most modern Linux distributions come with Python version 3 preinstalled. All the libraries required for the project were installed using the “apt install” command using the terminal. For example, to install the “openpyxl” library, we would type: “sudo apt install python-openpyxl”. The “sudo” command is required because administrator or super user permissions are required to install packages.

3.3 LINUX

Linux is an operating system, meaning a piece of software that manages all the physical hardware present on the machine it is installed on. The reason Linux has become increasingly popular among developers is the fact that it is free and open-source. Compared to its main rival, Microsoft Windows, it has fewer security risks since most applications are installed using a package manager that contains only trusted packages. Due to its open-source nature, Linux is being continuously improved by various organizations and volunteers and therefore it reached a

great diversity in terms of hardware support, running from small single board computers like the Raspberry Pi all the way to supercomputer servers.

Linux also has the advantaged of higher security than its main competitor, Microsoft Windows, and does not require an antivirus, which further improves its performance. All the programs that are available for Linux are centralized into repositories, which contained trusted open-source programs that are peer-reviewed by members of the Linux programming community.

The operating system chosen to run on the Raspberry Pi is Raspbian, due to the fact that it was developed by the creators of the Raspberry Pi itself, it is heavily optimized for this specific platform. Such optimizations include a lightweight graphical user interface, meant to reduce the RAM usage, easy configuration of the GPIO pins settings or easy to install packages compiled specifically for ARM architecture. With its over 35000 available packages being able to be installed from the internet using a simple “apt-get install program-name”, Raspbian has applications meant to tailor a large number of domains.

Raspbian is meant to be run off of a SD card, meaning it is small in size and can run on as little as 2GB of storage space. It also includes a few optimizations to minimize the write operations on the main system disk, since SD cards have lower endurance than desktop class solid state drives.

In Linux, to gain escalated privileges the word “sudo” must be used before the command, which stands for Super User Do.

3.4 EXCEL

Microsoft Excel is a spreadsheet editor software for Windows, Mac OS, Android and iOS. A spreadsheet is a grid of cells arranged in numbered rows and letter-named columns to organize data. There are free alternative programs that are also supported on Linux, for example LibreOffice.

The reason the Excel spreadsheet was chosen editor is because it is a widespread program and thus it is well known by many people. It provides an easy and attractive GUI to modify the voting ballot. It is also able to effortlessly preview a document before printing in order to see the layout on a physical sheet of paper. Furthermore, using the power of Python, the file and its contents can be read, thus avoiding the need for a separate console input when running the scanning program.

The simplest way to see an Excel file in programming terms is as a matrix.

	A	B
1	100	200
2	150	400

Figure 3.1 – Excel data example

This is interpreted by Python as a matrix: [[100,200],[150,400]] using numpy.

3.5 NETWORKING

The communication between the Raspberry Pi and the central server is achieved using the TCP/IP protocol. This protocol model for computer networking was created in the early 1970s due to the necessity of defining a suite of protocols to ensure end-to-end delivery of information between two computers on the Internet. This model defines the four functions that must be accomplished such that a communication transaction can be successfully completed.

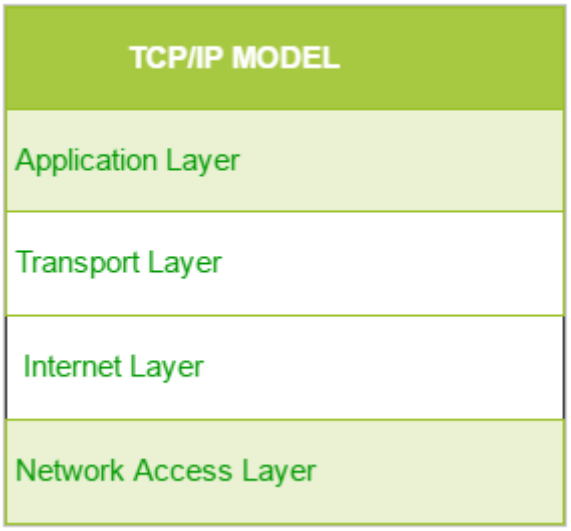


Figure 3.2 – TCP/IP model

The TCP/IP suite is a standard available freely to anyone, meaning that any person or vendor can implement these protocols in the hardware or software they produce.[4][13]

- The application layer is the top layer of the TCP/IP model and is the one the user interacts more closely. This level includes a set of protocols that accomplish specific functionalities for a variety of applications. Such protocols are:
  - DNS – converts URLs into IP addresses
  - SSH – allows for a secure remote connection to a server or networking

equipment, like routers and switches
    - DHCP – used for automatic assigning of IP, network mask, default gateway and DNS server information to an equipment connecting to a new network.
    - HTTP – Used for transferring web pages
    - FTP – Used for transferring files

- The transport layer is responsible for establishing a communication session between two applications and sending data between them. The application that sends data does not account for the type of the receiver host, the medium through which the data will travel or the path of the data in the network. The layer primarily:
    - Segments the data from the application layer and reassembles it at the destination. There are two important protocols
        - TCP, which has checksum verification for data integrity, and is important for essential data transfers, like documents.
        - UDP, which has no verification so it's faster since no computing is required for calculating checksums. It is primarily used for video streaming for example, where if a frame is not transmitted properly it is only a minor inconvenience, but speed and latency is more important.
    - Monitors the connection between the server and client.
    - Identifies the application that is meant to receive each segment
- The Internet layer is used for multiple tasks:
    - Assign IP addresses to devices in order for them to be identified on the network.
    - Routing packets through a network using devices called routers. A packet can be redirected through many routers before it reaches its destination.
    - Encapsulating data by adding to the segment from the transport layer information about the receiver and sender IP.
    - Decapsulating the data by first checking if the IP contained in the packet matches the IP of the device that received it.
- Network access layer is used for sending data through a physical medium (either by copper wire, optical fiber or radio waves). This is achieved using encoding of data using different schemes like for example Manchester or NRZ. It can send data synchronously, without a clock signal associated, meaning there is the need for start and stop indicators, or synchronously with a clock signal.

3.5.1 The File Transfer Protocol (FTP)

FTP is a communication protocol mainly used to transfer files between computers in a network. This protocol has a set of standards and rules for sending files. It works using a client-server system. First we will describe the Client-Server model. It was developed around 1960-1970s, by

Xerox. It is an architecture that shares resources between service providers, known as servers, and service requesters, known as clients. The clients and servers communicate using a network, either local or more frequently external, like the Internet.
The client-server model is a layered structure, having three base components:
-The client
-The network infrastructure
-The server
It is worth noting that almost the entirety of the Internet is based on this model. Millions of servers are connected to a global network, running 24/7.
There is also the possibility for clients to connect directly to each other (for example BitTorrent applications) or for servers to connect directly to each other (usually for exchanging data for backups).[14]

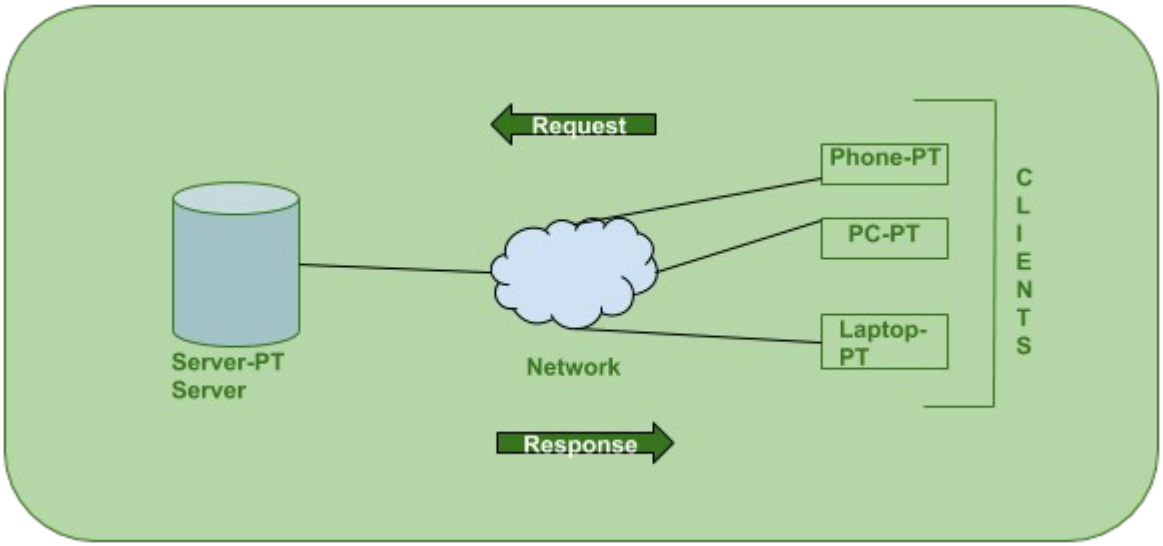


Figure 3.3 – Client-Server model

Such behavior can be noted when a computer connects to a FTP server:

Server: 220 Welcome to my FTP server
Client: USER username
Server: 331 “Password required for username”
Client: PASS ****
Server: 230 Logged on
Client: PWD
Server: 257 “E:\ is current working directory”
Client: RETR file.txt
Server: 150 Opening data channel for file download from server of file.txt
Server: 226 Successfully transferred file.txt
Client: STOR Buletin (2).xlsx
Server: 150 Opening data channel for file upload to server of "/Buletin (2).xlsx"

Server: 226 Successfully transferred "/Buletin (2).xlsx"
Client: QUIT
Server: 221 Goodbye

The standard port for FTP is 21, but it can be changed if for example, the ISP does not allow it to be used.
An IP address is an identifier for a computer in a network. In most networks, the IPv4 standard is used, which consists of a 32 bit number in the following format: A.B.C.D, with each letter having a possible value between 0-255 (8 bits). But due to the limited number of available IP addresses, and the rapid explosion of devices connected to the Internet, especially Internet of Things devices, the IPv6 standard was introduced.

FTP allows for multiple users to have different access privileges, using passwords to prevent unauthorized entry. Additional security settings can be added to prevent brute force attacks.

The server in my case is a laptop running Microsoft Windows 10, with Filezilla FTP Server installed. After allowing inbound and outbound connections through the firewall, and creating an user for the Raspberry Pi, it was ready to receive the voting results.

On the client side, we need to know the IP address, the port, the user and the password.
The IP address can be found from the server by issuing the “ipconfig” command in the command prompt of the operating system. For most home networks, it usually starts with “192.168”. In case the server was outside the home network, we would have to enable port forwarding in the router and we will have a different IP address.

The Raspberry Pi 2 used in this project does not have built in Wi-Fi, so it can connect to a nearby router only using an Ethernet cable. Another solution is to use a cell phone with a SIM card with a mobile internet plan, with tethering functionality, either by Wi-Fi, Bluetooth or USB, or using an USB modem directly.

3.6 DIGITAL IMAGE REPRESENTATION

An image is a two dimensional continuous signal that is defined as a function of two continuous variables: f(x,y). A digital image represents a real image that has a set of finite numerical values, encoded according to a system.[9]

The image has a representation in the discrete plane as a matrix with a limited number of lines and columns. This mode of representation divides the image into very small units called pixels and their position can be calculated using two planar coordinates.

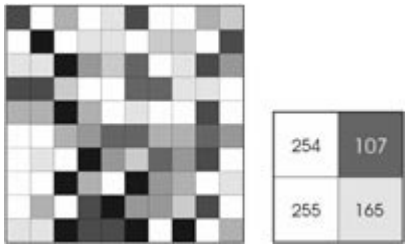


Figure 3.4 – Digital image representation

Each pixel of an image is associated with its relative position and with its characteristic value of the emitted light signal. The digital signals may be classified as:
-Black/White or binary, with one bit per pixel
-Grayscale, with 8 or more bits per pixel
-Color, with 8 or more bits for each primary color

There are multiple ways to represent color:
-RGB (Red, Green, Blue) by using primary colors
-HSV (Hue, Saturation, Value)
-HSL (Hue, Saturation, Lightness)

To calculate the intensity of a colored pixel and to convert it to grayscale, the following relation is used:
I=0.299xR+0.587xG+0.114xB

According to the type of data structures, the images can be classified as following:

-Scalar images, where each component is a scalar (monochrome images, e.g. black and white or grayscale).

For example, if we want to get the value of one pixel, we will get a number between 0-255.

-Vectorial images, where each component is a vector (in color images, the vector has three components, corresponding to one of the three primary colors).

For example, if we want to get the value of one pixel, we will get a matrix [x, y, z], where x, y and z can take values between 0-255 according to the level of each primary color. This makes their manipulation in software more complex, since they are viewed as a 3D matrix, instead of a 2D matrix as with scalar images.

To obtain a digital image, we must convert a continuous signal to a discrete signal, thus two types of operations are needed: sampling and quantization.

Sampling determines the spatial resolution of the digital image. This is because an image, like any analog signal, has an infinite number of points, and it is impossible to store an infinite number of data on a digital system. Therefore, only some points of an image are captured. Quantization determines the number of gray levels in the digital image. This is mainly because for each pixel there are only 8 bits or 512 bits for grayscale or color images, respectively, so

some rounding errors may occur, though most are not noticeable by the human eye.[2]

A more familiar example of sampling and quantization is in audio recordings. Sampling refers to the time period between two measurements of the amplitude of the signal. Too little and the reproduced signal is inaccurate. Too much and we obtain data that is redundant. Quantization refers to the rounding of the amplitude of the signal. Again, a balance between accuracy and data size must be achieved.[10]

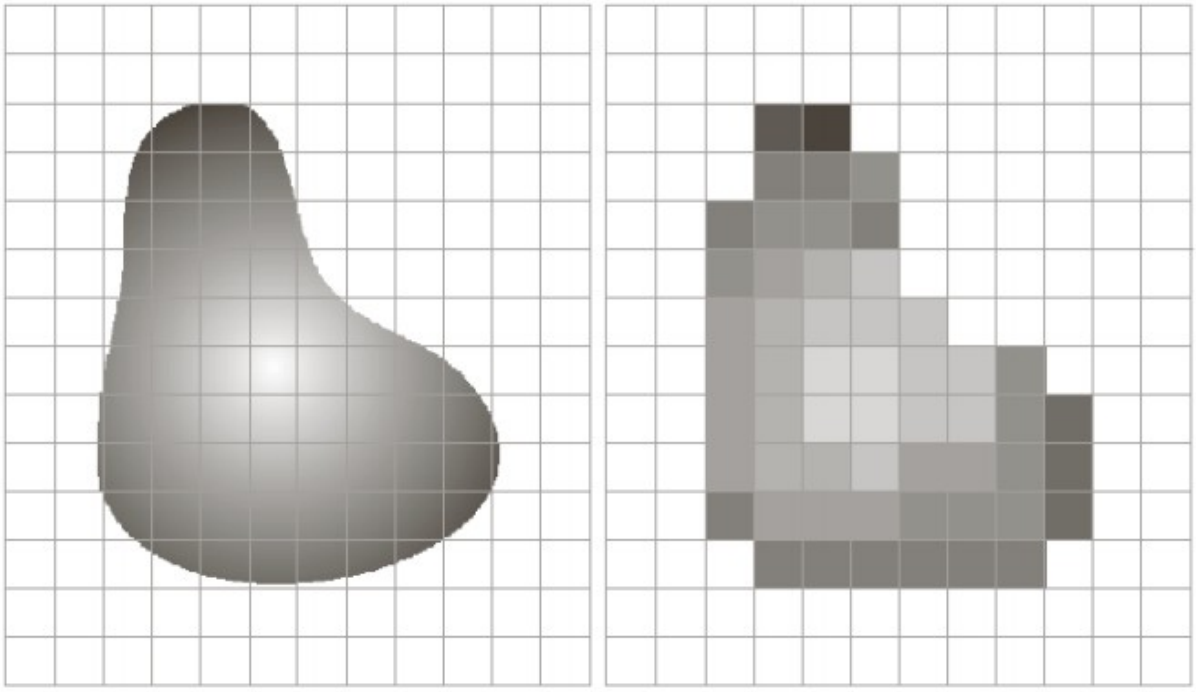


Figure 3.5 – Image sampling and quantization

In the figure above, it can be noted the sampling and quantization of data from a continuous analog image to a discrete digital image. The higher the number of pixels, the higher the fidelity of the reconstructed image. The contents of each square are averaged and transformed into a value represented in binary. The more bits allocated, the higher the color accuracy. [3]

3.6.1 Image storage

The image can be stored on a computer using non volatile memory, such as a hard disk, a solid state drive, etc. This is done by storing them as files which contain, alongside the values of the pixels themselves, the dimension of the image, aspect ratio, the color table, indexing method, bits per pixel.
Images can be stored in a wide variety of formats, like BMP, TIFF, GIF, JPG, PNG.

When accessed by a program, the image is loaded into RAM, which has a lower capacity but a much faster speed. The images are declared as a vector or matrix vector, and then the corresponding values of the pixels are decoded and loaded into memory, which allows for fast processing of the image. The images are declared dynamically in order to limit the memory usage.

CHAPTER 4: PRACTICAL REALIZATION OF THE WORK

4.1 CREATING THE EXCEL TEMPLATE

In this chapter the design of the Excel template will be explained.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1														cod tip alegeri	
2															
3		Titlu alegeri 1								A36				C-2	
4															
5			A11							A37					
6															
7			A12					Titlu alegeri 4							B
8															
9			A13							A41					
10															
11			A14							A42					
12															
13			A15							A43					
14															
15										A44					
16															
17		Titlu alegeri 2								A45				B	
18															
19			A21							A46					
20															
21			A22							A47					
22															
23			A23							A48					
24															
25			A24												

Figure 4.1 – Voting ballot in Excel form

As it can be noticed, the titles for the first half at column B and for the second half at column H. Despite the title cells being merged, the first column B with C, D, E and F, and the second

column H with I, J, K and L, the data read by Python is only at columns B and H.
The candidate names are placed at columns D for the first half and J for the second half. Likewise, despite being merged with E, F and K, L respectively, the data is only at D and J. The type of election is written on column N for the first half and O for the second half of the titles on the same row as the title.

The spreadsheet contains black square positioning markers next to every possible voting area (every odd row). The reason there are so many squares is to improve detection accuracy, especially if the image is slightly rotated.
Initially, only 4 squares were used, two at the top row and two at the bottom row and the positions of the voting areas were calculated by dividing the distance between the top and bottom row with the total number of rows. But due slight errors in detection of the squares due to the limited resolution of the camera and slight angle changes when the printed paper was placed under the webcam, this was the template used.

The right side contains the types of votes used for validation, and it is not printed. The letter B signifies an unlimited number of votes, so a person can vote for 1 person or 10 persons, it does not matter. The letter C followed by a number N signifies that a maximum of N votes must be marked, otherwise the votes for that particular election are considered null.

As a great man once said, "*where there are advantages, there are also disadvantages*". The most important advantage is increased accuracy of votes detection, whereas the disadvantages can be considered to be the increased ink usage during printing and the unaesthetic design, but it is a sacrifice worth making for fast and correct vote identification.

The user is can take this template and modify it to suit his or her needs, with the condition that the same dimensions must be kept for the cells.

Next, it will be presented the differences between theoretical image processing, using a PNG file created directly from a PDF file exported from the Excel file, and real world image processing using a JPG file obtained from a photo capture of the webcam.

Figure 4.2 – Computer generated square

As it can be noticed in figure 4.2, representing the first case, the picture contains only two colors, black and white, and they can be easily identified using an "if equal" comparison.

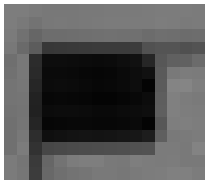


Figure 4.3 – Camera captured square

In the figure 4.3 however, it can be noticed that the picture contains multiple shades of gray, meaning that there is not a unique value for white or black, but rather intervals that require an "if greater or equal than" comparison.

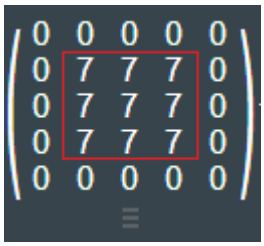


Figure 4.4 – Matrix representation of computer generated square

In figure 4.4 it is presented the matrix representation of the image in figure 5.2. As it can be observed, to detect the presence of a black pixel, we must have the value of the pixel equal to 7, otherwise, if it is 0, we have a white pixel.

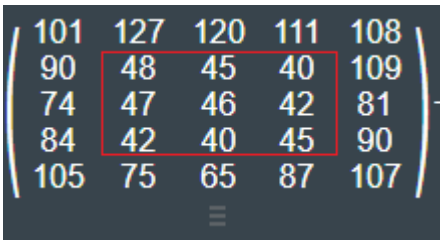


Figure 4.5 – Matrix representation of camera captured square

In figure 4.5, the situation for the case at figure 4.3 is shown. Due to the presence of noise and other disturbing factors, the values for a white or black pixel vary greatly. Therefore, it is required to use a threshold for determining if a pixel is white or black.

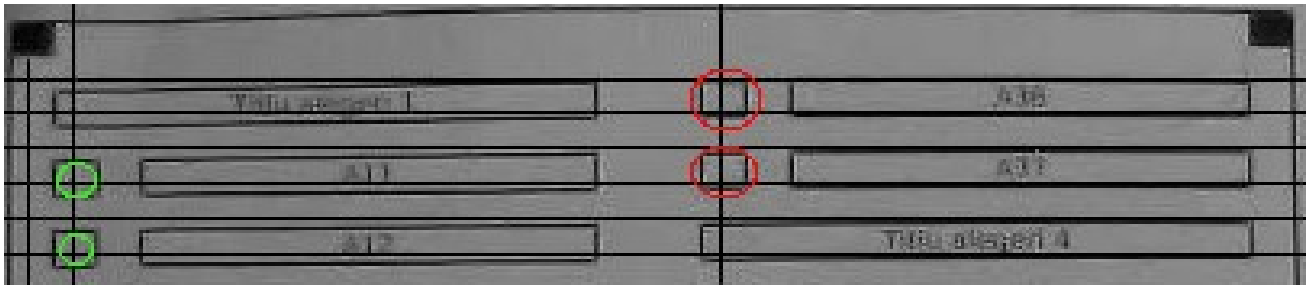


Figure 4.6 – Errors due to image rotation

As it can be noticed in figure 4.6, without having squares at each vote, the boxes on the left centers are correctly identified by the program (marked with green circles), where the horizontal and vertical lines meet, whereas the boxes on the right centers are not correct (marked with red circles). This is due to various artifacts encountered: image rotation, which may be due to imperfect design of the mounting system or slight movement of the camera if it is accidentally bumped, fisheye effect, which is caused when the edge of the paper is also at the edge of the image sensor, or imperfect paper which may have been folded and is not straight anymore.

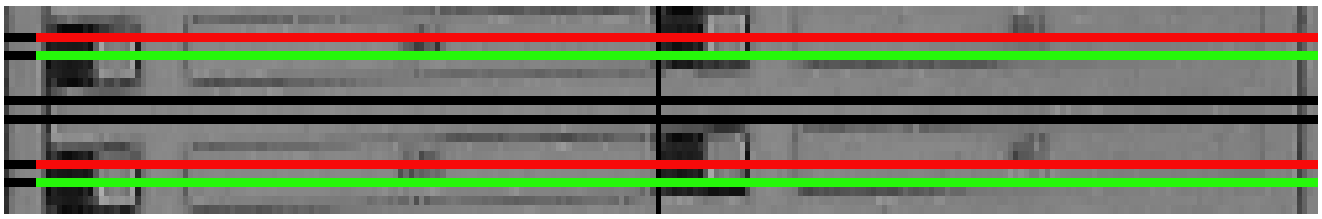


Figure 4.7 – Correct identification of squares

In figure 4.7, each square center position is independently calculated, and can be noticed how the green line is the center of the left square and the red line is the center of the right square.

Therefore, due to the fact that the paper may not be always placed at a straight angle (due to errors in the manufacturing of the webcam holding assembly or human error when placing papers in the tray), in order to have maximum accuracy I decided to use more squares.

The excel contains another sheet for storing the results, where next to the candidate names the total number of registered valid votes are written.

4.2. CAMERA CAPTURE

The initial camera used was a Microsoft LifeCam VX-2000, but due to the very poor quality and constant lighting problems, it was changed to a Logitech C270 HD camera.

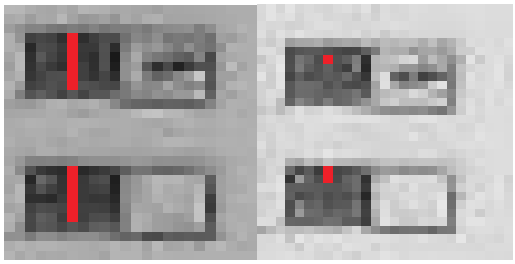


Figure 4.8 – Errors due to lighting conditions

In figure 4.8, there are presented two pictures taken seconds apart, under the same lighting conditions. Due to the camera's automatic changing of exposure, in the right picture the squares are not correctly identified, whereas in the left they are. This is seen by viewing the red line across them, drawn by the Python program by replacing each detected black pixel with a red one. The luminosity varied wildly between 117 and 218, and as such, the rate of detection varied as well.

One solution was to continuously take pictures from the camera until the luminosity was correct, which was done by measuring a white pixel on the paper and comparing it with a threshold. This was however discarded since sometimes the picture would be correct after 2 tries, whereas sometimes after 10 tries. The final solution was to replace the camera with a new one which has better behavior.

4.3. PROGRAM WORKFLOW

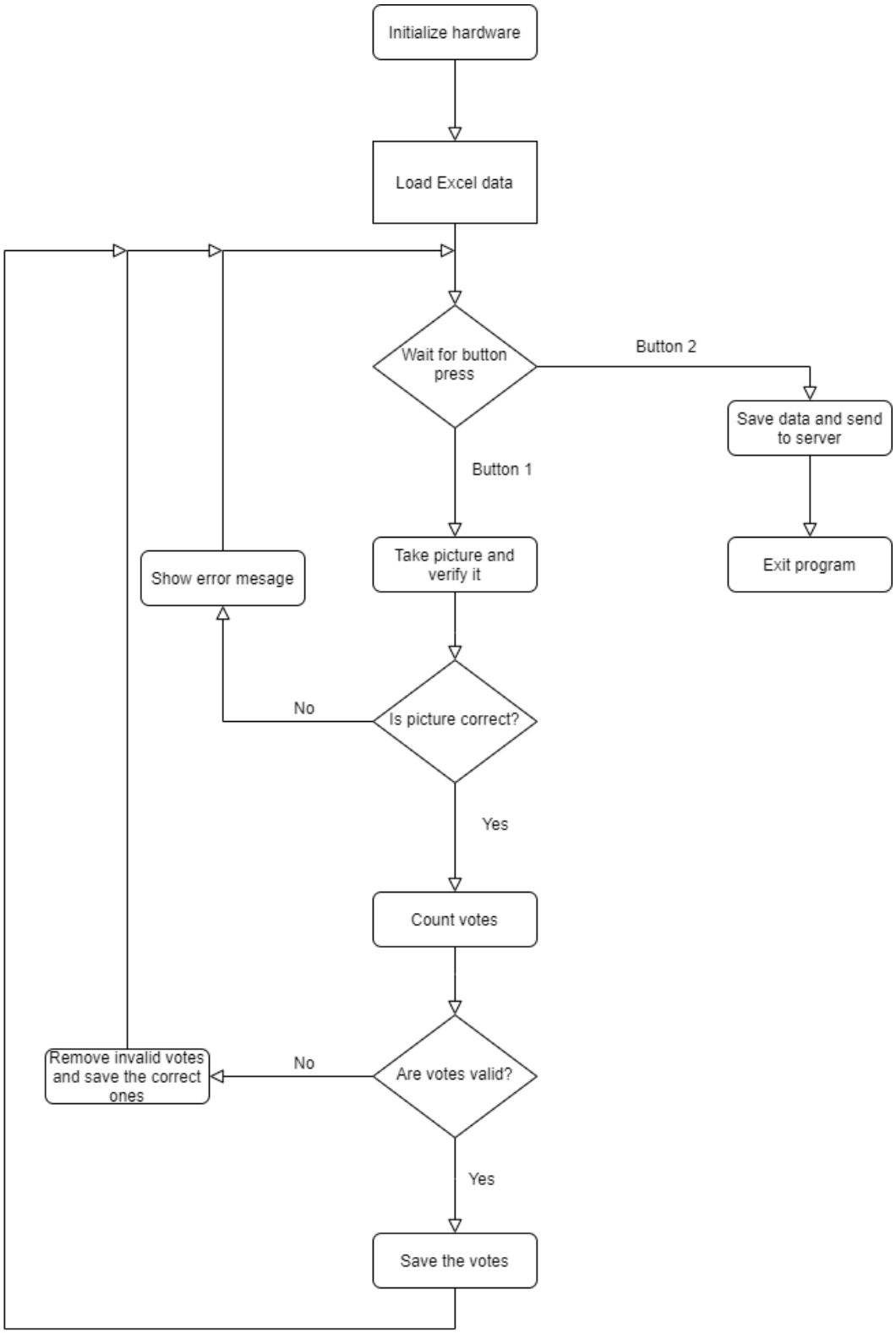


Figure 4.9 – Main program flowchart

In figure 4.9 the flowchart representing the basic functionality of the program is presented. The first step when starting the program is to initialize all the hardware, meaning the camera and the peripherals on the breadboard. After that, the Excel is loaded and all the variables required for processing are created. Afterwards, the program waits for the user to push one of two buttons.

If button number 1 is pressed, the scanning procedure starts. A picture is taken and then a verification is made, checking if the squares are detected as expected. If the paper is not in the proper position, the program will not find all the markers, and the user will be asked to take a new picture. If the markers are correctly identified, the paper is scanned by checking the areas meant to be marked by the voter. If there are any invalid titles, the program must subtract these votes from the temporary voting vector. The invalid votes position is known based on the position of the titles, which delimit an election. After the final votes vector is generated, the program will save these votes and proceed to wait again for an input from the user.

If button 2 is pressed, the program will save the final data to the Excel file, which is then sent to the server via FTP. Lastly, the program exits after freeing up the GPIO pins and disconnecting the hardware.

First of all, the required libraries for the functioning of the program are imported.

- ”os”:for working with files and folders recursively in order to process all images within a certain directory
- ”counter”:used to uniquely identify each election title and thus, determining if the number of candidates voted per election title is valid or not
- ”xlrd”:used to read the Excel file, in order to extract the important values: election titles, candidate names and number of maximum votes for each election title
- ”pil”:for working with images, allowing to import a wide variety of image formats and convert them into a matrix for convenient processing like rotation, converting to grayscale
- ”openpyxl”:used to write into the second sheet of the Excel file the results of the election
- ”cv2”: used for capturing images from the webcam
- ”smbus2”: for I^2C communication with the LCD
- ”RPi.GPIO”: for GPIO communication with the push buttons, LEDs and buzzer
- ”time”: for waiting for the user to press a button and conserving CPU cycles, thus saving power and heat by not overloading the processor with useless operations.
- ”ftplib”: for sending the results file to the central server via File Transfer Protocol

The first step is to prepare the Excel by clearing the report sheet so that previous results from other tests are deleted. This is done by writing an empty string into each cell that would contain vote results (columns B and H), and restoring the titles using the initial voting sheet.

The second step is to read the data from the Excel file, namely the titles of the elections, the candidate names and the types of votes. The resulting list contains exactly the content from the file, including blank cells, for easier association later on.
For example, for the Excel file shown in figure 5.1, the resulting vector for candidates would be: [‘’,’’,’’,’’,A11,’’,A12,’’,A13,’’,A14…].

The third step is initializing a list with the number of votes for each candidate, at first it will contain all elements equal to 0.

In the fourth step, a while loop is entered, which waits for the user to press a special button to start scanning a picture. If the button is pressed, the program will begin taking a picture from the webcam and process it. The while loop exits when the user presses a separate button meant for exiting the loop, signaling that all the papers to be scanned where processed and the program can exit.

The first function in the while loop is dedicated to only taking a picture from the webcam and preparing it for further processing. This is done by getting a frame from the webcam and writing it to a temporary file.

The file is then converted to grayscale, since we are only interested to see if the box is white (in this case there is no vote) or not (there has been a vote done by pen). This simplifies the processing, since with color images, three channels need to be processed, one for each primary color: red, green and blue. Also, it does not matter what color the pen used for the voting was. Therefore, it was decided for easier programming (not having to worry about 3 dimensional matrices) and faster processing to convert to grayscale.

Secondly, the file is rotated 270 degrees (or -90 degrees), since the camera is mounted in such a way that the image is rotated.

The second function in the while loop is dedicated to creating a list with the positions of the black squares. This is done in multiple steps. The program has a parameter an approximate location of the first black square in the upper left corner. This is done to save processing time since the program does not have to search for it across the whole paper, because the papers are in the same position, with very little relative movement, the black square is always in the approximate area. The program then calculates the actual center of the square, in order to get a real, exact position instead of an approximation. Afterwards, the rest of the squares are detected by moving down vertically on the image, and appending into a list when there is a black pixel preceded by a white one, representing the start of a box.

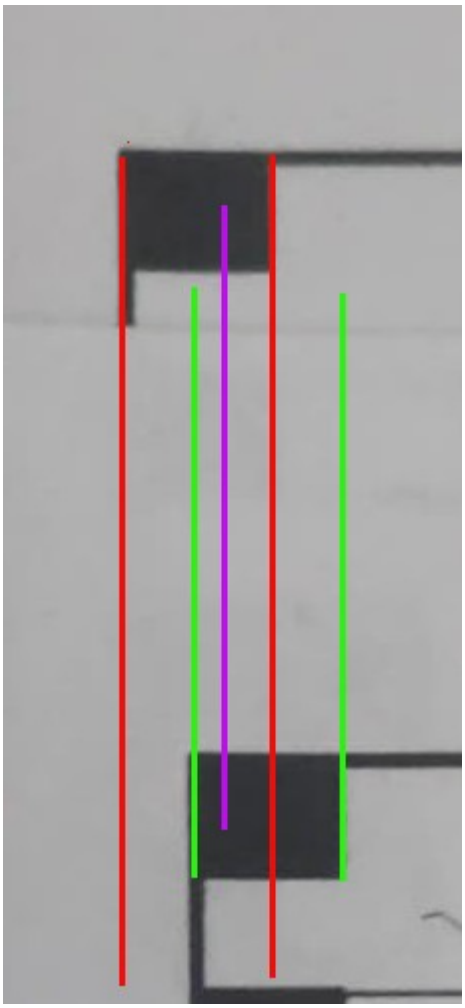


Figure 4.10 – Detection of markers with tolerance regarding x-axis positioning

To prove the function works, two papers were printed accidentally with different sizes by the printer. The upper square is delimited by red lines, whereas the lower one is delimited by green lines. The magenta line is the line used by the program for the approximate position. In both cases, the magenta line intersects the squares. After detecting the square, the real center is determined by moving left to right and from top to bottom on the square.

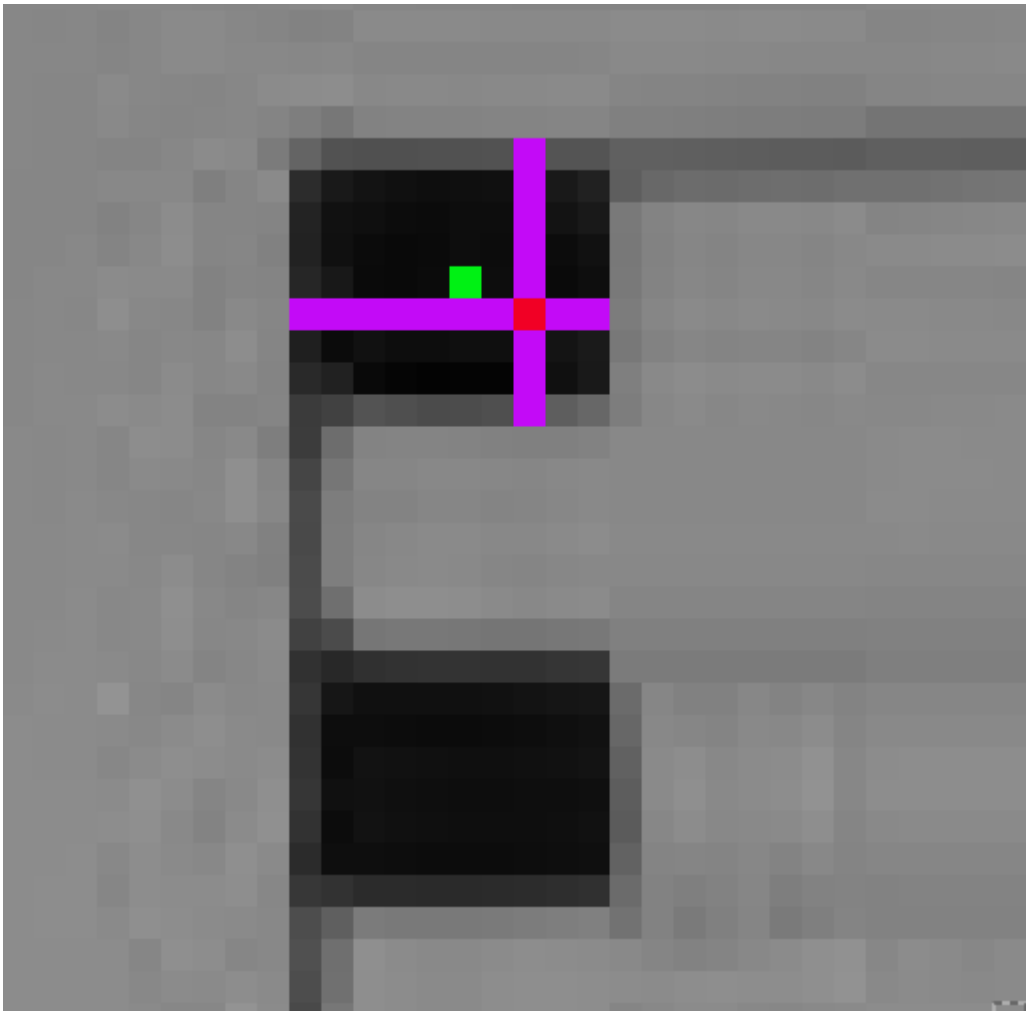


Figure 4.11 – Calculation of center

An example where the approximate initial center is determined by the red pixel. The program scans in all directions (signified by the magenta pixels) the pixels which are below a certain brightness threshold. Once the limits are found, the center is calculated and is represented by the green pixel

Initial red square coordinates=(117,76)
X coordinate of the real center=(119+110)/2=114.5*
Y coordinate of the real center=(79+71)/2=75

*Since images are matrices, and as such non-integer index values are not permitted, the number 114.5 is rounded to 115

If there is an error detecting the squares, the program will signal it with a loud 1 second beep, a red LED and a message on the LCD. It will ask the user to take a new picture. If the first square is successfully detected, the program moves down on the paper to detect the rest of the squares and marks in a vector the start and end of each box, and thus calculating by using a geometric formula the center of the rest of the boxes. The formula is:

Center=(Start+End)/2

The third function is used to detect the markings on the paper based on the list of positions from the previous function. It iterates over that list and whenever it detects a pixel lower than a

threshold, the vote is registered. The threshold was experimentally determined to be 80 under lighting from the LED lightbulb in my room.

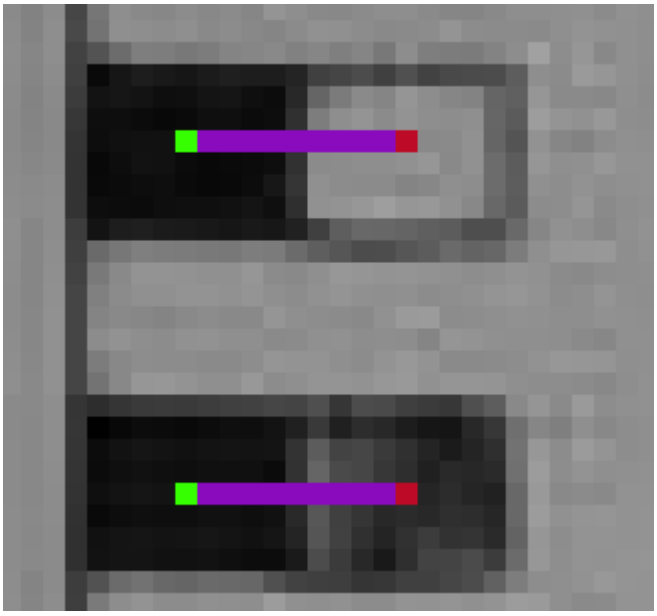


Figure 4.12 – Applying offsets to squares

As it can be noticed in the above figure, 5.12, the program relies on a fixed offset that is applied to the centers of the detection squares, that is shifted to the right, to obtain the centers of the voting squares. The offset has a numerical value of 10 pixels. Since the resolution of the picture is constant, as is the dimension of the A4 paper ballot sheet, as defined by the standard to be 210mmx297mm. Therefore, this technique is very easy and fast to compute by a microprocessor. If the pixel where the red dot is in the figure is less then the defined threshold, the box is considered empty. Otherwise, it is considered marked.

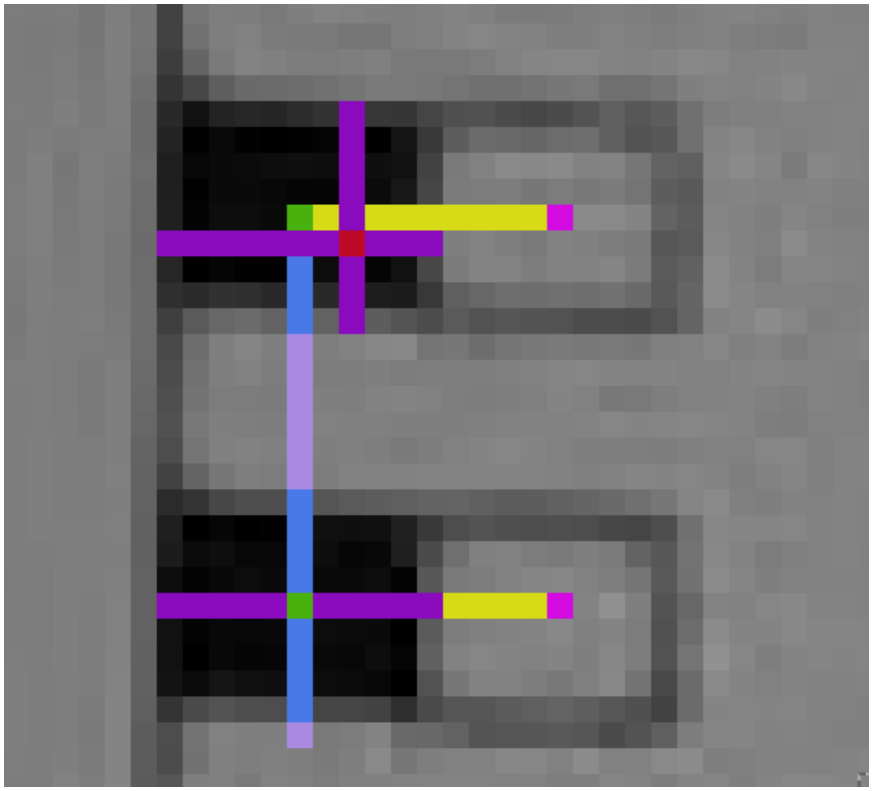


Figure 4.13 – Algorithm for processing entire paper

To sum up the functioning in one picture, figure 4.13 is presented: the first red dot is where the program is told to start. Afterwards, it searches vertically and horizontally, visualized by the magenta lines, the edges of the square. The calculated center is represented by a green dot. With the yellow line, the program applies the offset, and the pink dot is the pixel which will be compared.

After the first square is done, the program goes down, indicated by the blue and light pink lines. The blue represents that a pixel underneath it is dark, and the light pink represents a light pixel. When the program detects a square, by having found a succession of white, black and white. A new vertical center is calculated, shown by the green dot in the second square. The magenta shows the searching for the horizontal center, which happens to be exactly where the green dot was already positioned because the paper is not tilted. Then, the program applies the offset like in the first square.

The fourth function is needed for invalidation purposes. This means that if a certain election title has more than the imposed limit of votes, that title's results are eliminated. This is done by using the data from the Excel and the data from the sheet of paper and comparing them.

For example, if a paper has two election titles, one with a maximum of three and one with a

maximum of four, the check vector will be:
[3,4]
If the user votes for two candidates on the first votes and five candidates for the second, the voting validation vector will be:
[2,5]
By comparing each element on the same position, we can notice that the first is correct (2<=3) but the second is incorrect (5>4) so the results are nullified for it (equivalent to the situation that the voter did not vote for any candidate for that title).

The final function is meant to write the report into the Excel file. This will write to the second sheet in the file named “Report”, and in each box next to the votes the number of total valid votes will be written.

The program can be operated fully by the GPIO interface using buttons for input and an LCD, LEDs and buzzer for output. The LCD is used to display detailed messages about the state of the program. The LED is useful for checking at a glance the status. The buzzer is useful for having also an audio signaling. However, the program can be easily adapted to work using a normal desktop or laptop by simply replacing the condition of GPIO button presses with the condition of a keyboard press, using the “x=input()” function of Python. The input will be a single character, for example 1 or 2, and depending on the value of x the program will continue or exit.

If the user decides to exit by pressing a second special button, the program takes the following steps:
-Displays a goodbye message.
-Cleans up the GPIO interface connections by resetting them to the default values.
-Sends the file to the central collection server.

In the program, the variables for the IP of the server, the username and password are written inside the function. In my particular case, the IP of the server is “192.168.1.9”, the user is “raspberry1”, and the password is “licenta”.

CHAPTER 5: CONCLUSIONS

5.1 GENERAL CONCLUSIONS

This thesis presents the successive steps which were employed by the author in order to create a system for capturing and processing information from pictures of vote ballots. After describing the state-of-the-art in Chapter 1 and the theoretical aspects regarding hardware and software in Chapter 2 and 3, Chapter 4 presents the practical implementation.

5.2 PERSONAL CONTRIBUTIONS

The personal contributions of the author of this thesis are highlighted in chapter 5, where the program for scanning pictures was defined and explained thoroughly. Also, creating a template that is robust for scanning was mentioned. The entire hardware assembly was also made through lots of experimentation to determine the optimal focusing distance. The program suffered many revisions through which many improvements and bug fixes were made.

5.3 FUTURE WORK

There are many improvements that can be made for the current design, including:

- Add a high power LED for uniform, constant illumination. The system currently implemented depends on the light inside the room, provided either by the sun or the lightbulbs in the room. In order to have more predictable output, we should add a LED next to the webcam for better lighting. Since the Raspberry Pi has a limited current output on the GPIO pins or the USB interface. A separate power source may be needed, along with a MOSFET acting as a switch or even a relay.
- Create an automatic paper feeder, using normal or stepper motors, although this is more of a mechanical than an electronic problem; the system must be able to grab one and only one paper at a time.

- Use automatic detection of papers, so that the user does not have to push a button at every new paper, instead the program detects if a new paper is inserted below the camera.

REFERENCES

1. https://ro.rsdelivers.com/product/raspberry-pi/raspberry-pi-2-model-b/raspberry-pi-raspberry-pi-2-model-b-sbc-computer/8326274
2. http://www.sci.utah.edu/~arpaiva/classes/UT_ece6962/image_representation_and_discretization.pdf
3. https://www.slideserve.com/jaquelyn-roth/image-processing-chapter-2-digital-image-fundamentals
4. https://www.geeksforgeeks.org/tcp-ip-model/
5. https://www.tel.com/museum/exhibition/principle/cmos.html
6. https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/
7. https://electronicshobbyists.com/raspberry-pi-lcd-display-interfacing/
8. https://www.raspberrypi.org/forums/viewtopic.php?t=134394
9. https://spacetelescope.org/projects/fits_liberator/improc/
10. https://slideplayer.com/slide/15463344/
11. https://www.manualdeinstructiuni.ro/microsoft/lifecam-vx-2000/manual?p=10
12. http://www.exclusivearchitecture.com/?page_id=749
13. https://techdifferences.com/difference-between-tcp-ip-and-osi-model.html
14. https://www.geeksforgeeks.org/client-server-model/
15. https://en.wikipedia.org/wiki/File:I2C_data_transfer.svg

ANNEX 1

```
from PIL import Image
import cv2
import os
import xlrd
from collections import Counter
from openpyxl import *
import smbus2 as smbus
import RPi.GPIO as GPIO
import time
import ftplib

GPIO.setmode(GPIO.BOARD)
GPIO.setup(26, GPIO.OUT)
GPIO.setup(11, GPIO.IN,
pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(12, GPIO.IN,
pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(29,GPIO.OUT)
GPIO.setup(31,GPIO.OUT)
GPIO.setup(33,GPIO.OUT)

GPIO.output(26,GPIO.HIGH)

####
#!/usr/bin/python3



# Define some device parameters
I2C_ADDR = 0x3f     # I2C device
address, if any error, change this
address to 0x3f
LCD_WIDTH = 16      # Maximum
characters per line

# Define some device constants
LCD_CHR = 1     # Mode - Sending
data
LCD_CMD = 0     # Mode - Sending
command

LCD_LINE_1 = 0x80   # LCD RAM
address for the 1st line
LCD_LINE_2 = 0xC0   # LCD RAM
address for the 2nd line
LCD_LINE_3 = 0x94   # LCD RAM
address for the 3rd line
LCD_LINE_4 = 0xD4   # LCD RAM
address for the 4th line

LCD_BACKLIGHT = 0x00  # On

ENABLE = 0b00000100     # Enable
bit

# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005

# Open I2C interface
#bus = smbus.SMBus(0)  # Rev 1 Pi
uses 0
bus = smbus.SMBus(1)    # Rev 2 Pi
uses 1


def lcd_init():
    # Initialise display
    lcd_byte(0x33, LCD_CMD)     #
110011 Initialise
    lcd_byte(0x32, LCD_CMD)     #
110010 Initialise
    lcd_byte(0x06, LCD_CMD)     #
000110 Cursor move direction
    lcd_byte(0x0C, LCD_CMD)     #
001100 Display On,Cursor Off, Blink
Off
    lcd_byte(0x28, LCD_CMD)     #
101000 Data length, number of
lines, font size
    lcd_byte(0x01, LCD_CMD)     #
000001 Clear display
```

```
    time.sleep(E_DELAY)


def lcd_byte(bits, mode):
    # Send byte to data pins
    # bits = the data
    # mode = 1 for data
    #        0 for command

    bits_high = mode | (bits &
0xF0) | LCD_BACKLIGHT
    bits_low = mode | ((bits << 4)
& 0xF0) | LCD_BACKLIGHT

    # High bits
    bus.write_byte(I2C_ADDR,
bits_high)
    lcd_toggle_enable(bits_high)

    # Low bits
    bus.write_byte(I2C_ADDR,
bits_low)
    lcd_toggle_enable(bits_low)


def lcd_toggle_enable(bits):
    # Toggle enable
    time.sleep(E_DELAY)
    bus.write_byte(I2C_ADDR, (bits
| ENABLE))
    time.sleep(E_PULSE)
    bus.write_byte(I2C_ADDR, (bits
& ~ENABLE))
    time.sleep(E_DELAY)


def lcd_string(message, line):
    # Send string to display
    message =
message.ljust(LCD_WIDTH, " ")

    lcd_byte(line, LCD_CMD)

    for i in range(LCD_WIDTH):
        lcd_byte(ord(message[i]),
LCD_CHR)

###

def takepicture():
    print("reached camera")
    path = "img"
    camera = cv2.VideoCapture(0)
    return_value, image =
camera.read()
    cv2.imwrite(os.path.join(path,
'waka.jpg'), image)
    del (camera)
    for file in os.listdir(path):
        current_file =
os.path.join(path, file)
    img =
Image.open(current_file).convert('L
A')
    img = img.rotate(270,
expand=True)

    img.save("V4.png")
    return (img)


def checkpicture(img):
    ysquare = 70
    xsquare = 108
    xsquare2 = 237
    ysquare2=76
    squaredetect=[]
    pix = img.load()
    doonce1 = 1
    for i in range(97,113):
        temp=pix[i,ysquare2]
        temp2=temp[0]

        if(temp2<50 and
doonce1==1):
            squaredetect.append(i)
            doonce1=0
        print(squaredetect)#103,108
        #pix[i,ysquare2]=(255,255)

    img.save("temp123.png")
    if not squaredetect:
        print("ERROR")
        errorvar=1
        finalpositions=0

finalpositions2=finalpositions
        xsquare=finalpositions
        xsquare2=finalpositions
        return(finalpositions,
finalpositions2,xsquare,xsquare2,er
rorvar)
        exit(0)
    xsquare=squaredetect[0]+4
    xsquare2=xsquare+129


    width, height = img.size
    #print(width, height)
    # black=4,
white=127=>whitetrigger=50?
    whitetrigger = 50
    positions = []
    positions.append(70)
    finalpositions = []
    bw = 1
    for j in range(0, height -
ysquare):
        temp = pix[xsquare, ysquare
+ j]
        temp2 = temp[0]
        if (temp2 < whitetrigger):

positions.append(ysquare + j)
            pix[xsquare, ysquare +
j] = (255, 255)
```

```
            if (bw == 1):

finalpositions.append(ysquare + j)
                bw = 0
        else:
            bw = 1
    #print(finalpositions)

    positions2 = []
    positions2.append(70)
    finalpositions2 = []
    bw = 1
    for j in range(0, height -
ysquare):
        temp = pix[xsquare2,
ysquare + j]
        temp2 = temp[0]
        if (temp2 < whitetrigger):

positions2.append(ysquare + j)
            pix[xsquare2, ysquare +
j] = (255, 255)
            if (bw == 1):

finalpositions2.append(ysquare + j)
                bw = 0
        else:
            bw = 1
    #print(finalpositions2)
    #img.show()
    errorvar=0
    return finalpositions,
finalpositions2,xsquare,xsquare2,er
rorvar


def countvotes(img, finalpositions,
finalpositions2, candidati,
titlualegeri, tipalegeri,
titluriunice,

titluriunicenumar,xsquare,xsquare2)
:
    # add title detection, go
backwards!
    for j in
range(len(titluriunicenumar)):
        titluriunicenumar[j]=0
    pix = img.load()
    whitetrigger = 80
    xsquare = xsquare+11
    xsquare2 = xsquare2+11
    offset = 3

    vect1 = []
    vect2 = []
    listacandidativotati1 = []
    listacandidativotati2 = []
    listatitluri1 = []
    listatitluri2 = []
    for i in
range(len(finalpositions)):
        temp = pix[xsquare,
finalpositions[i] + offset]
        temp2 = temp[0]
        pix[xsquare,
finalpositions[i] +
offset]=(255,255)
        if (temp2 < whitetrigger):
            vect1.append(i)

listacandidativotati1.append(candid
ati[(i * 2)])
            listacandidatiexcel[i *
2] = listacandidatiexcel[i * 2] + 1
            x = 0
            j = i * 2
            while x == 0:
                if (titlualegeri[j]
!= ''):

listatitluri1.append(titlualegeri[j
])
                    x = 1
                else:
                    j = j - 1
        pix[xsquare,
finalpositions[i] + offset] = (255,
255)
    for i in
range(len(finalpositions2)):
        temp = pix[xsquare2,
finalpositions2[i] + offset]
        temp2 = temp[0]
        pix[xsquare,
finalpositions[i] + offset] = (255,
255)
        if (temp2 < whitetrigger):
            vect2.append(i)

listacandidativotati2.append(candid
ati[(i * 2) + 51])
            listacandidatiexcel[i *
2 + 51] = listacandidatiexcel[i * 2
+ 51] + 1
            x = 0
            j = i * 2 + 51
            while x == 0:
                if (titlualegeri[j]
!= ''):

listatitluri2.append(titlualegeri[j
])
                    x = 1
                else:
                    j = j - 1
        pix[xsquare2,
finalpositions2[i] + offset] =
(255, 255)
    img.save("votecheck.png")
    print(vect1)
    print(vect2)
    #print(listacandidativotati1)
    #print(listacandidativotati2)
    listtest1 = [2, 4, 9, 11, 12,
24, 25]
```

```
    listtest2 = [1, 4, 5, 6, 16,
18, 24]
    listtest3 = [3,5,9,11,21,22,23]
    listtest4 = [2,4,7,8,16,20,24]
    listtest5=[4, 6, 9, 10, 12, 21,
24]
    listtest6=[2, 9, 11, 14, 25]

    if (vect1 == listtest1):
        print("1-1ok")
    if (vect2 == listtest2):
        print("1-2ok")
    if(vect1==listtest3):
        print("2-1ok")
    if(vect2==listtest4):
        print("2-2ok")
    if(vect1==listtest5):
        print("3-1ok")
    if(vect2==listtest6):
        print("3-2ok")
    listafinalatitluri =
listatitluri1 + listatitluri2
    #print(listafinalatitluri)
    for i in
range(len(listafinalatitluri)):
        for j in
range(len(titluriunice)):
            if (titluriunice[j] ==
listafinalatitluri[i]):

#print(titluriunice[j])

titluriunicenumar[j] =
titluriunicenumar[j] + 1
    #print(titluriunicenumar)
    #print(tipalegeriprocesat)
    #print(listacandidatiexcel)
    #img.show()
    return
(listacandidatiexcel,titluriunicenu
mar)


def readexcel():
    location = "Buletin (2).xlsx"
    excel =
xlrd.open_workbook(location)
    sheet = excel.sheet_by_index(0)
    row = sheet.nrows
    col = sheet.ncols
    candidati = []
    titlualegeri = []
    tipalegeri = []
    for i in range(row):

titlualegeri.append(sheet.cell_valu
e(i, 1))

candidati.append(sheet.cell_value(i
, 3))

tipalegeri.append(sheet.cell_value(
i, 13))

    for i in range(row):

titlualegeri.append(sheet.cell_valu
e(i, 7))

candidati.append(sheet.cell_value(i
, 9))

tipalegeri.append(sheet.cell_value(
i, 14))
    #print(titlualegeri)
    #print(candidati)
    #print(tipalegeri)
    counterstuff =
Counter(titlualegeri).keys()
    counterstuff =
list(counterstuff)
    titluriunice = []
    titluriunicenumar = []
    for i in
range(len(counterstuff)):

titluriunice.append(counterstuff[i]
)
        titluriunicenumar.append(0)
    titluriunice.pop(0)
    titluriunicenumar.pop(0)
    #print(titluriunice)
    #print(titluriunicenumar)

    return (titlualegeri,
candidati, tipalegeri,
titluriunice, titluriunicenumar)


def proctipalegeri(tipalegeri):
    proc = []
    for i in
range(len(tipalegeri)):
        if (tipalegeri[i] != ''):

proc.append(tipalegeri[i])
    proc.pop(0)
    #print(proc)
    for i in range(len(proc)):
        if (str(proc[i]) == "B"):
            proc[i] = 100
        elif (str(proc[i]) != "B"):
            proc[i] = proc[i][2]
            proc[i] = int(proc[i])
    #print(proc)
    return proc


def clearexcel():
    location = "Buletin (2).xlsx"
    wb = load_workbook("Buletin
(2).xlsx")

    sheets = wb.sheetnames
    Sheet1 = wb[sheets[1]]
```

```
    excel =
xlrd.open_workbook(location)
    sheet = excel.sheet_by_index(0)
    row = sheet.nrows
    col = sheet.ncols
    j = 0
    for i in range(51):
        mystr = " "
        Sheet1.cell(row=i + 1,
column=2).value = mystr
        j = j + 1

    j = 0
    for i in range(51):
        mystr = " "
        Sheet1.cell(row=i + 1,
column=8).value = mystr
        j = j + 1

    wb.save("Buletin (2).xlsx")
    wb.close()


def writeexcel(listacandidatiexcel,
titlualegeri,excelstuff,titluriunic
e,fiseprocesate):
    location = "Buletin (2).xlsx"
    wb = load_workbook("Buletin
(2).xlsx")

    sheets = wb.sheetnames
    sheet1 = wb[sheets[1]]

    excel =
xlrd.open_workbook(location)
    sheet = excel.sheet_by_index(0)
    row = sheet.nrows
    col = sheet.ncols
    j = 0
    for i in range(51):
        mystr =
listacandidatiexcel[i]
        if (mystr != 0):
            sheet1.cell(row=i + 1,
column=2).value = mystr
        j = j + 1

    j = 0
    for i in range(51):
        mystr =
listacandidatiexcel[i + 51]
        if (mystr != 0):
            sheet1.cell(row=i + 1,
column=8).value = mystr
        j = j + 1
    # scriere titluri
    j = 0
    for i in range(51):
        mystr = titlualegeri[i]
        if (mystr != ''):
            sheet1.cell(row=i + 1,
column=2).value = mystr
        j = j + 1
    j = 0
    for i in range(51):
        mystr = titlualegeri[i +
51]
        if (mystr != ''):
            sheet1.cell(row=i + 1,
column=8).value = mystr
        j = j + 1

    for i in
range(len(excelstuff)):
        mystr=str(titluriunice[i])+
" are " + str(excelstuff[i]) + "
vot(uri) invalide din " +
str(fiseprocesate) + " vot(uri)"

sheet1.cell(row=i+1,column=14).valu
e=mystr

    wb.save("Buletin (2).xlsx")
    wb.close()

def
invalidation(tipalegeriprocesat,
titluriunicenumar,listacandidatiexc
el,titluriunice,titlualegeri,excels
tuff):
    #print(tipalegeriprocesat)
    #print(titluriunicenumar)
    #print(titluriunice)
    ok=0
    okvector=[]
    for i in
range(len(tipalegeriprocesat)):

if(titluriunicenumar[i]<=tipalegeri
procesat[i]):
            ok=ok+1
            okvector.append(1)
        else:
            okvector.append(0)
    #print(okvector)
    gotcha=""
    validvariable=0
    for i in range(len(okvector)):
        if(okvector[i]==0):
            #print(titluriunice[i])
            gotcha=titluriunice[i]

excelstuff[i]=excelstuff[i]+1
    if (gotcha!=""):
        for i in
range(len(titlualegeri)):

if(titlualegeri[i]==gotcha):
                #print(gotcha,i)
                start=i

#print("!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!",start)
        for i in
range(start+1,1000):
            if(titlualegeri[i]!
```

```
='):

#print(i,titlualegeri[i])
                finish=i
                break
        for i in
range(start,finish):
            if
listacandidatiexcel[i]!=0:

listacandidatiexcel[i]=listacandida
tiexcel[i]-1
        validvariable=1


return(validvariable,excelstuff)
    #print(listacandidatiexcel)



lcd_init()
clearexcel()
titlualegeri, candidati,
tipalegeri, titluriunice,
titluriunicenumar = readexcel()
listacandidatiexcel = []
for i in range(len(candidati)):
    listacandidatiexcel.append(0)
tipalegeriprocesat =
proctipalegeri(tipalegeri)
x='a'
GPIO.output(29,GPIO.LOW)
GPIO.output(31,GPIO.LOW)
GPIO.output(33,GPIO.LOW)
GPIO.output(26,GPIO.HIGH)
print("GOGOGOGO")

fiseprocesate=0


excelstuff=[]
for i in range(len(titluriunice)):
    excelstuff.append(0)
print(excelstuff)
lcd_string("Welcome!",LCD_LINE_1)
while x=='a':



    time.sleep(0.5)
    if GPIO.input(11) == GPIO.HIGH:
        GPIO.output(29,GPIO.LOW)
        GPIO.output(31,GPIO.LOW)
        GPIO.output(33,GPIO.LOW)
        img = takepicture()
        finalpositions,
finalpositions2,xsquare,xsquare2,er
rorvar = checkpicture(img)
        if(errorvar==1):
            GPIO.output(26,False)
            time.sleep(1)
            GPIO.output(26,True)
            GPIO.output(33,GPIO.HIGH)
                lcd_string("Wrong
marker!",LCD_LINE_1)

lcd_string("Next!",LCD_LINE_2)
        else:

fiseprocesate=fiseprocesate+1

listacandidatiexcel,titluriunicenum
ar = countvotes(img,
finalpositions, finalpositions2,
candidati, titlualegeri,
tipalegeri,

titluriunice,
titluriunicenumar,xsquare,xsquare2)

validvariable,excelstuff=invalidati
on(tipalegeriprocesat,titluriunicen
umar,listacandidatiexcel,titluriuni
ce,titlualegeri,excelstuff)

writeexcel(listacandidatiexcel,
titlualegeri,excelstuff,titluriunic
e, fiseprocesate)
            if(validvariable==1):

GPIO.output(31,GPIO.HIGH)

GPIO.output(26,False)
                time.sleep(0.15)

GPIO.output(26,True)
                time.sleep(0.15)

GPIO.output(26,False)
                time.sleep(0.15)

GPIO.output(26,True)
                print("GOGOGOGO")
                lcd_string("Bad
votes!",LCD_LINE_1)

lcd_string("Next!",LCD_LINE_2)
            else:

GPIO.output(29,GPIO.HIGH)

GPIO.output(26,False)
                time.sleep(0.15)

GPIO.output(26,True)
                time.sleep(0.15)
                lcd_string("Good
votes!",LCD_LINE_1)

lcd_string("Next!",LCD_LINE_2)
                print("GOGOGOGO")


    if GPIO.input(12) == GPIO.HIGH:
        print("Goodbye!")
```

```
        print(excelstuff)
        #GPIO.output(26,False)
        #time.sleep(0.5)
        GPIO.output(26,True)

lcd_string("Goodbye!",LCD_LINE_1)
        GPIO.cleanup()

        #session =
ftplib.FTP('192.168.1.9','raspberry
1','licenta')
        #file = open('Buletin
(2).xlsx','rb')                  #
file to send
        #session.storbinary('STOR
Buletin (2).xlsx', file)     # send
the file
        #file.close()
# close file and FTP
        #session.quit()


        exit(0)
```