agri4.0

المدرسة العليا للتكنولوجيا -أكادير
ⵜⵜⵉⵛⵀ ⵜⴰⵙⴷⴰⵡⵜ ⵏ ⵜⵜⴽⵏⵓⵍⵓⵊⵢⴰⵜ - ⴰⵅⴰⴷⵉⵔ
ÉCOLE SUPÉRIEURE DE TECHNOLOGIE - AGADIR

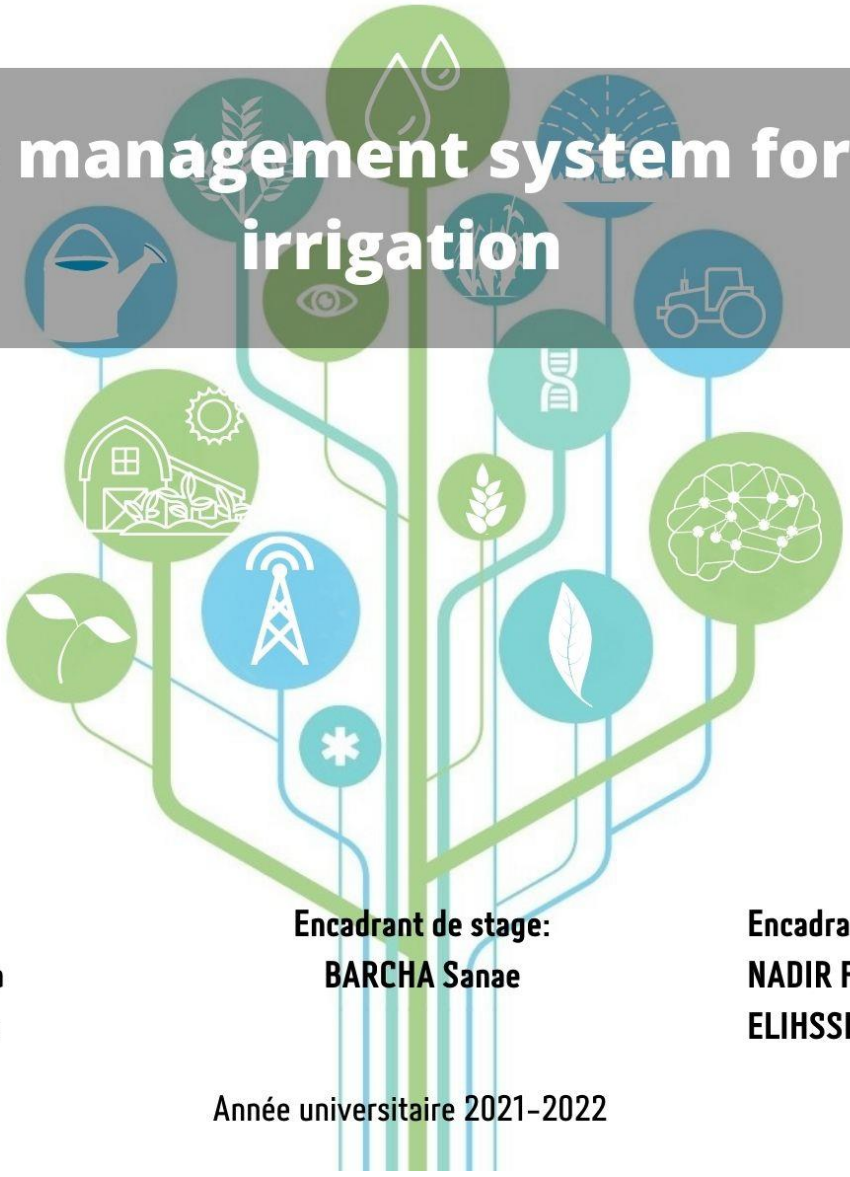
DEPARTEMENT : GENIE ELECTRIQUE

Rapport de stage de Fin d'Études

au sein de l'entreprise Agri4.0

Présenté en vue de l'obtention du Diplome Universitaire de Technologie En Génie Electrique

remote management system for smart irrigation



Realisé par
KADDOURI Nassira
AK-HAIL Oussama

Encadrant de stage:
BARCHA Sanae

Encadrant pédagogique:
NADIR Fatima-ezzahra
ELIHSSINI Hossine

Année universitaire 2021-2022

Appreciation:

First we want to thank everyone who helped elaborating this project, we want to say thank you to anyone in EST Agadir in general, and in electrical engineering department in particular who helped giving us a high quality two years of studies.

By this chance we want to express our deep gratitude to Ma'am Fatime Ezzahra Nadir and Mister Hossine Ellihssini for being supportive, helpful and extremely dedicated teachers, no words can describe how thankful we feel towards the two of them, for helping as and providing us with the necessary tool to elaborate the project all the last two months long.

We want to thank also Agri4.0's startup, Mr.TABIT Fouad, Mrs.BARCHA Sanae , for giving us the chance to pass this internship, and providing us with the necessary tools.

Table of *Contents*

Table of figures

general introduction:

Today, the agricultural sector is among the most important pillars of the Moroccan economy, and its contribution to the gross domestic product is considered significant. Since the growth rate in Morocco is strongly wired to the rate of agricultural production, agriculture is constantly evolving, as the farmer is always looking for a middle ground between productivity, profit and quality.

Reaching out to efficiency in agriculture requires several accurate studies about consumption of water, raw materials and energy, identifying the tasks that costs more and needs optimization, then exploiting the right techniques to reduce consumption without losing the production rate.

As a result of the notable decrease in precipitations (rain, snow, sleet, hail, etc.) and the desertification phenomena caused by climate changes, combined with the expansion in demand for water, Morocco is facing major water scarcity. The sector of agriculture is the most affected one by the drought, yet the biggest contributor in the water shortfall in Morocco, because the hugely deployed traditional ways of irrigation induce major water loss. Besides having an extremely terrible impact on nature, traditional irrigation is considered as one of the most costly tasks for farmers since it is highly demanding in terms of energy, this is due to the use of electrical pumps to provide large amounts of irrigation water that consume a lot of energy. That is why letting the plant itself control the irrigation process by demanding water if needed, then telling if it is not thirsty anymore in order to turn the irrigation pump off, the best choice to optimize this task. To make it possible, smart sensors must be put around the plant, to take the necessary measures such as humidity and temperature; then use them to command irrigation. To avoid wired communications between the plant and the irrigation pumps which requires physical supports that can be expensive in case of long distance data transaction, it was better to use GSM or GPRS protocols.

As being aware of the previously mentioned issue, caring about our planet, and knowing well the unpleasing impact of scarcity and changes in crop yields on Moroccan economy, since it could induce rising prices of grains and other agricultural products as well as The industrial products demanding agricultural products as row materials for their production, or reduce the GDP of Morocco at constant prices, and most importantly eliminate many job opportunities, in particular in the rural areas of this country. We decided for our end-of-studies internship project to touch the issue by creating an ecological irrigation system based on data logging the status of the plants using a data logger to activate or deactivate an actuator (irrigation pump) dependably with this status; this system must basically assure the communication through GSM between the data logger and the actuator.

This this report consists of three principal chapters:

- The first represents the company where the internship took place in, and its sector of activity.

- The second deals with the struggles that agriculture faces in Morocco, and then suggests technological solutions for the issues.
- While the third and last is a detailed representation of the project concerning the exploitation of remote management technology in creating an automated irrigation system via GSM.

Chap1.GENERAL PRESENTATION:

1.1 Field of activity (AGRITECH)

Agritech refers to the exploitation of technological innovation to improve the efficiency of agricultural processes, the main purpose of all agritech technologies is to grow more outputs (food and other agricultural products) in less space and/or with fewer inputs (land, water, seeds…). Agritech also saves farmers time and money, by automating tasks and replacing much of the labor needed on a farming operation - which usually constitutes the highest cost input to a farming system.

1.1.1 Agri 4.0

AGRI 4.0 Company is active in the design and production of electronic sensors and the development of web and mobile applications dedicated to agriculture which allow the intelligent management of farms through tools for tracking, monitoring and real-time analysis facilitating the process of decision-making and can lead to predictive management processes.
The major goal of Agri 4.0 is making agriculture profitable in Morocco and Africa in general to reduce the gap between our agriculture and that of world leaders in this field, considering this as our last chance to catch up and take advantage of new technologies to better exploit the wealth we have in Morocco and Africa while preserving for future generations, In addition to reducing the hurtful behavior of agricultural systems towards our planet, and make sure to generate a strong environmental impact.

1.1.2 Company data sheet

Address	RUE NISRINE CITE RYAD SALAM, N°8 , Agadir, SOUSS MASSA
Tax identity :	Tax identity :
Phone number :	Phone number :
ICE :	002056833000046
RC:	38275
Sector of activity :	Agriculture de précision, big data, intelligence artificielle, Monitoring, and surveillance à distance
Creation date :	2018
CEO :	Fouad Tabit
capital:	200 000 DH

Figure 1:Agri4.0 data sheet

1.1.3 The actual project:

Using remote management technologies in a communicating irrigation system: assuring the ability of the plants to get water when needed and stop the irrigation process at the right time, by measuring the humidity using a humidity sensor, collecting the measured values using the data logger, and then sending a SMS that holds the proper command to the electronic card that receives the SMS through a GSM module, and commands the pump according to it.

Chap2.AGRICULTURE BETWEEN STRUGLS AND TECHNOLOGICAL DEVELOPMENT

Agricultural as the field that faces new constraints and struggles repeatedly has made researchers all around the world go ahead and find brand new efficient technics to help improving the profitability of this sector, for its high influence on economic growth (in average, agriculture contributes 15% of total GDP).

2.1. Agriculture in Morocco and Africa:

2.1.1. Overview:

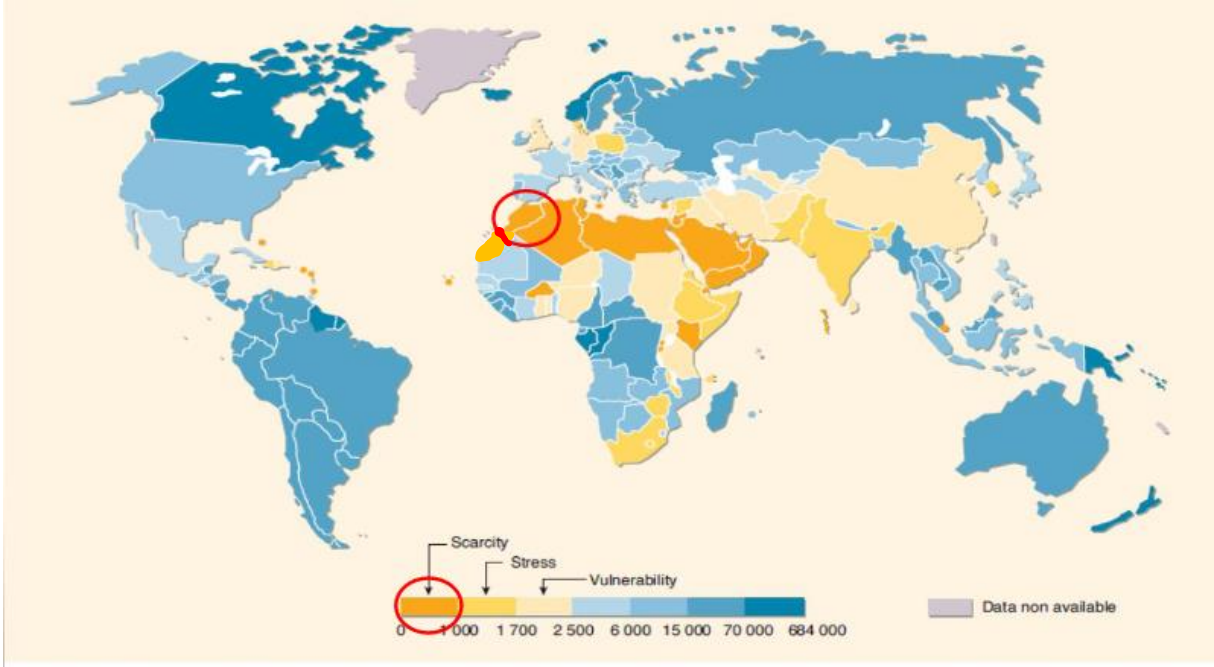
The majority of African countries were under the colonial powers for several years during the last century, after gaining independence and escaping from the over control of the colonies, many African governments were occupied with providing food security, employment chances and economic growth as well as fighting against poverty. Thus, they paid a lot of attention to the agricultural sector and gave it a lot of importance, considering it the biggest chance to overcome the retard, the slow growth rate and the low human development index, this couldn't became possible without exploiting the domestic wealth of the continent properly, and progressively developing this field and make it more efficient and profitable by using the new trends of agritech technologies.
Morocco was among the most aiming countries for getting advances in agriculture, which leaded this sector to end up being the largest employer in the country (40 percent of all employment), yet one of the most costing ones due to the climatic and geographic changes it faces.

2.1.2. Challenges

The sector of agriculture is the most affected one by the changing international demands, in addition to the climate change, especially in Morocco and other African countries where land is becoming no longer suitable for cultivation, and WS&D (Water Scarcity and Drought) is starting to represent an alerting struggle against the growth and success of agricultural production, hence, Morocco is grappling with this problem, and several experts are exploring new ways of developing agriculture in search of a solution to the drawbacks, by reducing the ratio between the inputs and outputs.

2.1.3. Water scarcity and drought in Morocco

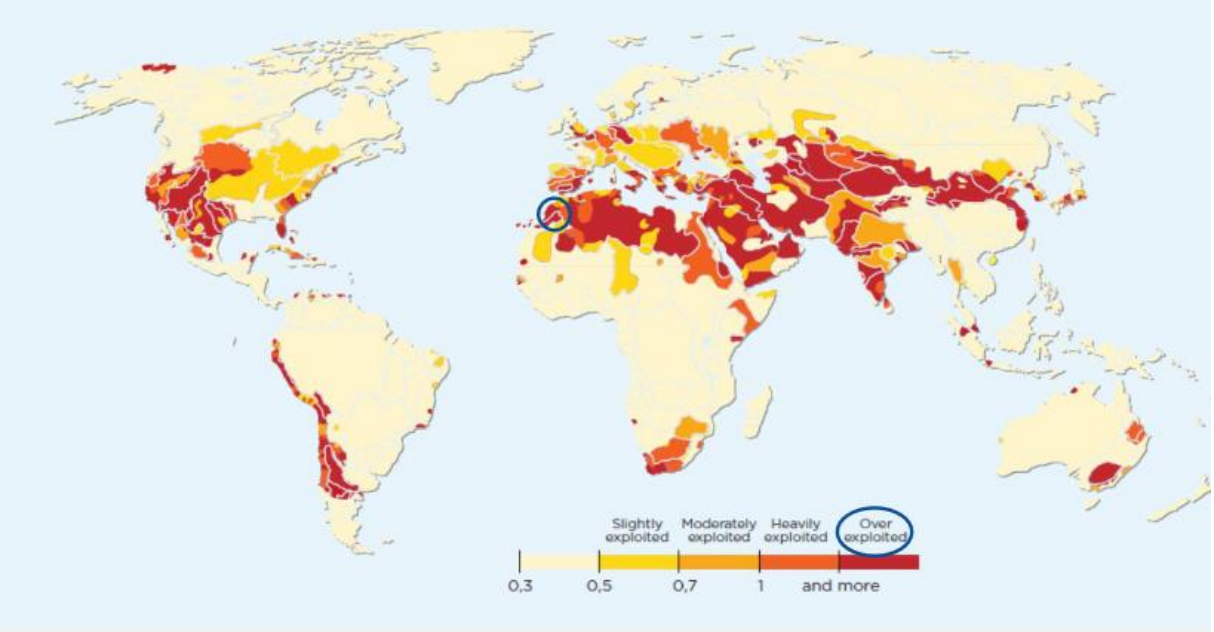
Morocco is considered as one of freshwater scarcity countries, and is expected to face severe drought periods.



Source: UNEP/GRID-Arendal (2008) (http://maps.grida.no/go/graphic/global-waterstress-and-scarcity, P. Rekacewicz [cartographer] (Le Monde diplomatique), with sources FAO and WRI).

Figure 2:Fresh water availability (m^3 per person per year, 2007)

Morocco's 1.4 million hectares of irrigated crops, which represents 15 percent of the country's irrigated lands, consume, an average of 85% of available water resources (60 to 70% in a dry year), while public water supply do not pass 12%, and industry uses 3%. This can lead to a conclusion that agricultural sector is the most over exploiting one of Moroccan water resources, inducing huge water stress in the country's hydrological basin.



Source: UNEP/GRID-Arendal (2008) (http://maps.grida.no/go/graphic/water-scarcity-index, P. Rekacewicz [cartographer], with sources Smakhtin, Revenga and Döll [2004]).

Figure 3:Water stress Indicator in (WSI) in major basins

This leaded the country to take advanced ways in terms of WS&D management.

2.2. The impacting technological developments on agriculture

2.2.1. Agritech and the modernization of agriculture

Over time, agricultural engineers have developed many methods in order to improve production in ways that do not harm the consumer nor the producer. Perhaps one of the most important of these methods is cultivation inside greenhouses where the sun's rays enter, carrying its heat into the glass greenhouses, and then the heat does not leak out at the same rate, which leads to a high temperature inside the greenhouse in a way that does not hurt the plant. The sizes of greenhouses range from small rooms that do not need special heating to accommodate a few plants to spacious tents that may include a few acres of artificially heated, where flowers or fruit trees are raised to bloom and fruit in the off-season; It is now widely used for growing plants and growing fruits and vegetables. This type of farming is called protected farming, but greenhouses alone are not always enough to get an ideal product. It is also very expensive, especially for small farmers; here is where the importance of introducing technology to the agricultural field appears. Now the farmer can know what the plant needs (water, heat, mineral salts...) through sensors developed to suit agricultural standards and give accurate reports. Agritech researchers and scientists have discovered many advanced ways to make agriculture more efficient, among the most interesting and important technologies we find:

- **Remote management:** Remote management systems mainly allow farmers to remotely control many actions in their farms, like controlling the irrigation flow, which helps to avoid wasting water. The remote management is a big advantage for agricultural professionals because any system with this technology is simply controlled from the smartphone or any other connected device.
- **Smart irrigation:** this technology uses soil moisture data and weather data determine the irrigation need of the landscape, which maximizes irrigation efficiency by reducing water waste, while maintaining plant health and quality.
- **Fertilization: S**ometimes the farmer resorts to cross-breeding between two different plants in order to collect the desired traits in each of them, for example they take pollen grains from one plant and spreads them on another plant, when the seeds are ripe, they sow them, on new plants that combine desirable traits, for example, to obtain fruits that keep their appearance ripe for a longer period without losing their taste
- **Big data in agriculture:** It refers to the process of collecting huge databases throughout time then analyzing them to give farmers detailed insights into areas such as how different areas of their land behave through the seasons, the granular efficiency of their specific farming practices, and where environmental impact can be reduced.
- **Communication protocols and smart sensor / actuators:** allowing the communication between the different equipment, as well as the real time monitoring and control over the divers parts of the farm.
- **Farming by machines**: after agritech robots started being used in farms, a golden era was begun for farmers being able to use 90% less of inputs, by replacing the majority of human labor with robots whom responsibilities could include herbicide spraying, shepherding, seeding, harvesting, and more. Usually, their operation is much more precise and therefore cost efficient for farmers. As well as using agritech drones for irrigation, seeding and weeding or analyzing farm activity, and

also collecting the data that will be used to improve the product, all these allowed farmers to reduce the time they spend on manual labor and repetitive tasks.

- **Satellite farming**: (also known as GPS agriculture) that is being used to minimize agriculture inputs while getting the yields boosted, which enables the real-time mapping of the farmland, and giving a chance to exploit self-driving tractors.

2.2.2. Remote management

2.2.2.1. Remote management meaning

Remote management (or télégestion in the fresnsh dictionary) is the implementation of computers and other electronic devices, to remote control an isolated or remote positioned device/technical installation, in such a way that the data transaction is based on telecommunication technics.

2.2.2.2. Remote management tools

To elaborate a remote management system and unsure the functioning of its desired process, the following three elements are needed:

- A remote terminal unit (RTU) that represents the responsible equipment for ensuring the acquisition of information and passing on the command and control.
- A wired or wireless modem that ensures a network communication: switched telephone network, GSM, GPRS, radio, Wi-Fi, etc.
- A central data collection and management system.

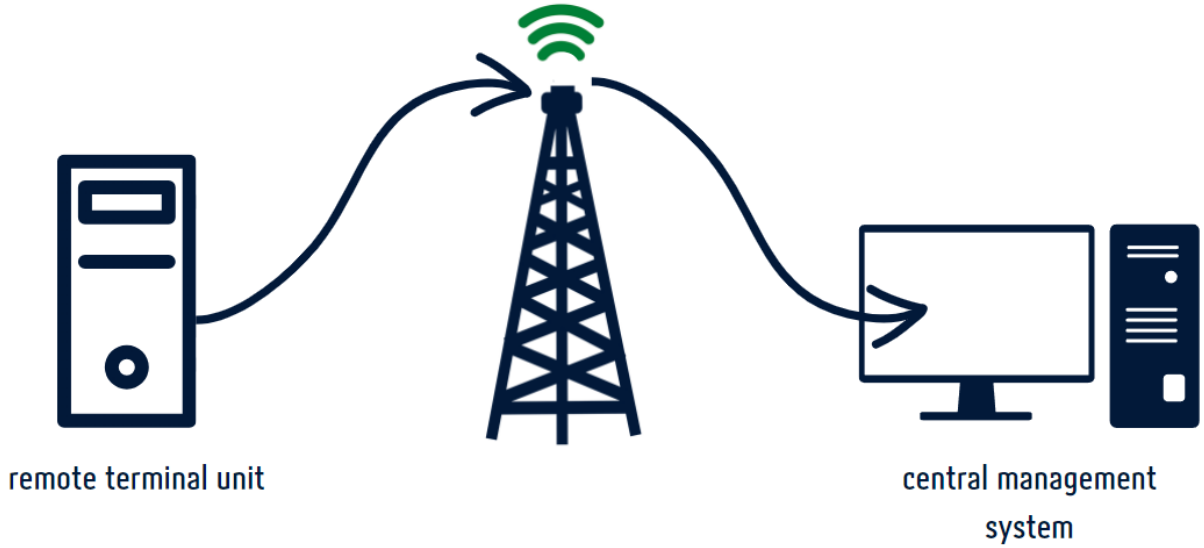


Figure 4: remote management tools

2.2.2.3. The uses of remote management

Since the remote management process provides numerous tools, such as **remote alarms** (automatically alerted in the case of a malfunction of an installation), **Telemetry**, which is a technology that allows remote measurement and logging of information of interest to the system designer or an operator (permanently and remotely check the functioning of an installation), and **Remote control** (commanding the functioning of geographically distributed device or installation), in addition to **Remote maintenance** (To ensure remote maintenance of certain installations), It meets the needs of a wide range of applications like:

- Water network management
- Public light
- Climat engineering
- Smart irrigation
- Domotique and house Security

2.2.2.4. The uses of remote management in agriculture

The remote management in the agricultural dictionary is always wired to remote sensing technology which is mainly used for agricultural mapping, forecasting, and crop tracking that can help increase yields. Not only a better management decision taking is enabled with the collected data, but also the automation of activities such as irrigation. Not to mention how the automation and the well-done management of irrigation will be the most positively impacting action in the current situation of water stress.

2.2.3. Technology of supervision

2.2.3.1. Supervision and the interest behind it

The term supervision in industrial sectors refers to the act of continuously viewing/checking on equipment's and the different working forces to verify if they are working as planned, as well as controlling them.
The main reason of doing supervision is to ensure the continuity and safety of industrial installations. Now after the huge advancement that the field of artificial intelligence has known, the used data in supervision can be collected to be exploited to predict future events or constraints in the future to avoid them or perfectly manage them. For example: predicting the right time to do the maintenance and machinery previewing…

2.2.3.2. Supervision's evolution through time

We may think that supervision has started under the forma we are seeing it nowadays, being done through multiple screens in specified rooms for that specific task; far from the worksite, but actually humans were always supervising their production, even before IHMs (man-machine interfaces) were invented, this task used to be done directly, humans has to visualize the production evolution, and check if the working forces and machines are respecting the time schedules by themselves. After the industrial revolution, the production kept getting bigger and bigger. Hence the supervision task started to be more complicated and challenging, and needed to be done in more effective ways, so after the third industrial revolution (also known as industry 3.0) in the late 1900s, which was characterized by the spread of automation and digitalization by using computers…, the supervision as well as other process has got automated. The manner to supervise machines has become representing actuators by LEDs (lighting up means that the actuator is ON and vice versa) and controlling them with actual physical buttons. Then it went to visualizing the working forces including machines through screens and controlling them from distant places. This turning point was due to evolution of protocols like Modbus (1979) which allow the read write of bit/ bits as well as word/words. With the evolution of automated systems, switching from hard-wired logic to programmed logic with microprocessors and programmable logic controllers, HMIs have got also developed to facilitate supervision tasks. They have gone from the control desk and screen printing (with buttons and indicator lights, etc.) to supervision screens and PCs, which is making the option range of human-machine interfaces wider, since graphic user interfaces are getting easy to develop, which is giving them a huge diversity.

This advancement can't be anything other than profitable for diverse sectors, because the easier the supervision task gets, the less costing it becomes.

2.2.3.3. Supervision in agriculture

Farmers used to supervise their farms since ever. Before the industrial revolution, when agriculture was at its most traditional phases, agricultural land owners or the actual farmers used to do their regular checks in the sake for ensuring the keeping everything under control by themselves, or assigning some workers to do so in case of huge fields. Human beings were partially good supervisors, but rising the efficiency and accuracy of this process, required the involvement of advanced technics. Therefore, the supervision thing in agriculture kept evolving forwards alongside with the total technological development and industrial modernization. Now the supervision is being done in a way that allows visualizing the entirety of the farm (the environment/weather, the crops and all of the machines) simultaneously and remotely, through screens. Not only the monitoring but the control has become easier and more efficient too. For example: monitoring the water use alongside with the envirment conditions to well exploit it without wasting a single drop.
Graphical based applications can be easy to use for everyone no matter what are their background and knowledge level in agriculture so they can do the control and monitoring of some agricultural installations in the easiest ways.

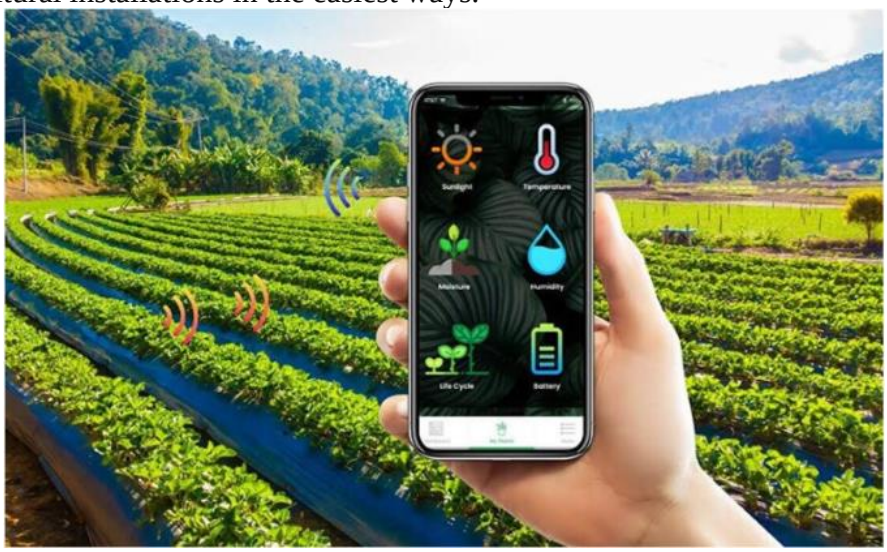


Figure 5: type of supervision in agriculture using phone app

2.2.4. Graphic user interfaces

2.2.4.1. Definition

The GUI (graphic user interface) is the kind of interface through which the user interacts with the electronic equipment using graphic elements and virtual objects. The development of this type of interfaces has helped replacing the challenging textual interfaces of older computing with simpler and easier to use ones that can let the largest category of people interact with electronic devices with fewer struggles, and made the user more comfortable by making the interface more natural and aesthetically pleasing. Hence the GUIs have become the world's standards computing interfaces.

2.2.4.2. History of the GUI

Figure 6: Xerox Star workstation

The birth of the GUI dates back to 1979, when the Xerox Palo Alto Research Center developed the first prototype for a GUI, which used a bunch symbols and visual elements, such as windows, icons and menus.
The highly advanced GUIs that we all are using nowadays are the result of a continuous and cumulative work, thus, while Quantel developers working on their interface Paintbox that was released in 1981 as the first as a color graphical workstation with supporting of mouse input, lips machines was being developed at MIT to be later commercialized by many manufacturers such as Symbolics, they were early high-end single user computer workstations with advanced graphical user interfaces, windowing, and mouse as an input device. Then the simplified mackintosh that was designed to be relatively low costing was released in 1984 to be the first commercially successful product to use a multi-panel window interface, as a result of years of ideas development from the part of the Apple Lisa and Macintosh teams at Apple Computer (which included former members of the Xerox PARC group).

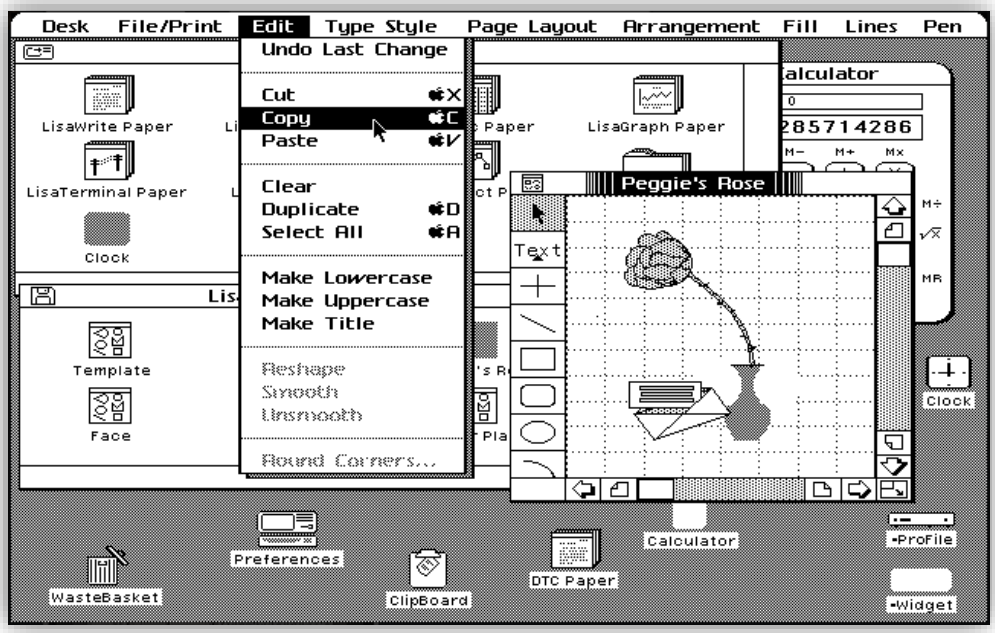


Figure 7: Apple Lisa the first GUI personal computer

2.2.4.3. designing/developing GUIs

A large range of platforms, coding languages and libraries can be used to make GUIs, and it is up to the developer to choose the ones that provides the needed widgets that suits their GUI. What really matters about elaborating a GUI is implementing the right widgets to make the user interact smoothly with the device, letting them enter their commands through text boxes, option menus and buttons, and allowing them to get information's from the electronic equipment as well using the proper widgets. The most used programing languages in this type

of development are Java and C#, python can be used too since it provides some good frame works for GUI like pyQT, pygui, tkinter, etc.

2.2.4.4. The use of GUIs in agriculture

Agriculture, like all other sectors, requires a high level of accuracy and rationality in its management, as it requires continuous tracking of the inputs, the crops (outputs), as well as accurate control of electrical installations, regularly previewing them to predict their maintenance dates, which raises the needs of flexible mediators between the farmers and their machines/ other parts of the farms, this leaded to a large use of GUIs in farms in the recent few decades, this increasing demand for this type of interfaces in farms is due to its simplicity, as it makes it an easy task for the user to configure, visualize and command their systems in the smoothest ways.

Conclusion:

Despite all the challenges that agricultural sector faces, farmers can still improve their production, with the use of electronic and digital systems involving several sensors specially designed for agriculture, so that the farmer can follow the evolution of the plants, and monitor the production as well as reducing the input/output ratio. Water crises can also be a surpassed problem with the supervision of irrigation combined with the advanced management of this task

Chap3. PROJECT DEVELOPMENT

In the previous chapter, we have come to a conclusion that the ultimate way to overcome the struggles that agriculture faces, is the implementation of advanced technologies to manage the production, make it less costing and more profitable at the same time, as well as having intermediate tools to do such an accurate and continuous supervision, so in this chapter we are going to represent a project that aims to reduce cost of irrigation and it's horrifying impact on environment as well, the chapter is going to show the specifications, the used tools and the sequence of total phases to elaborate the project.

3.1. Exploitation of agro technologies to make a remote management irrigation system

3.1.1. Specifications

To overcome the water scarcity obstacle, alongside with the production of genetically modified products that are able to withstand dry weather and grow naturally under semi-harsh living conditions, several other procedures should have been followed to reduce the water consumption, and therefore reducing the intensity of agriculture's bad imprint on nature. The most important one among those procedures is water rationing, which urges to manage the use of water perfectly, therefore this project aims to manage irrigation remotely, using remote management technology. In order to build an irrigation system that is capable of controlling the responsible pump of the irrigation task according to the plants' needs, it was obligatory to assure the following points:

- Commanding the ON/OFF states of a distant actuator (a pump in this case) through a wireless data transaction protocol (such as GSM) depending on a collected data using sensors, with no intervention of the human being.
- The system must be capable of detecting any sort of malfunctions and threats, then alerting them (continuously verify the GSM Network connection and the pump's response comparing to the command sent to it).

The following block diagram explains briefly how the system works:

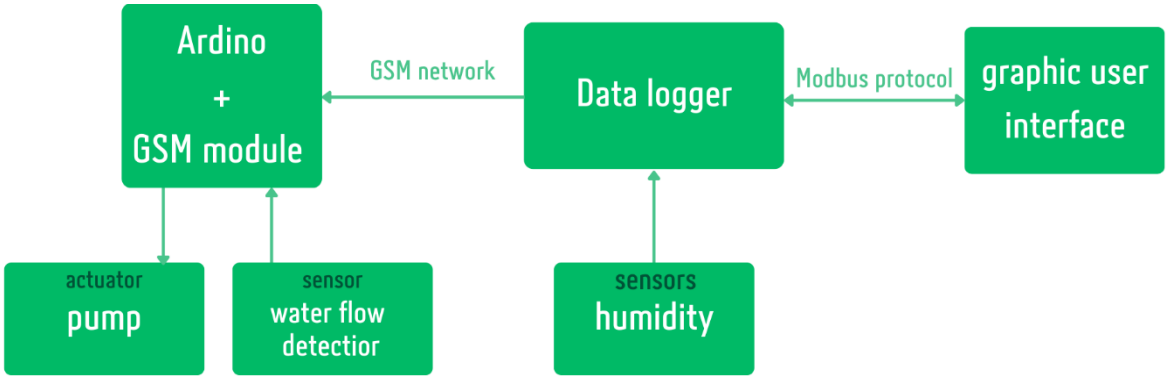


Figure 8: block diagram of the system

3.1.2. Tools:

3.1.2.1. Hardware:

The RTU:

Every remote management system requires an RTU that is responsible of the data acquisition and passing on the command and control. The RTU in this case is **King Pigeon's data logger S272**, which is a GSM GPRS 4G cellular M2M IoT RTU, we have chosen this data logger from king pigeon because it has 6 analog inputs and 8 digital one, plus a preconditioned tow for humidity/temperature measurement, hence it has a high ability of collecting data from different sensors simultaneously, plus its ability to connect to GSM and GPRS networks, send and receive messages and calls, which make it a good option to get the soils moisture, and consider the reached values as the triggering events for the commands to be sent as SMS via GSM, furthermore this data logger is specially designed to work in harsh industries which make it a suitable option in agriculture, moreover, its 4 digital outputs can be commanded by intern or extern events (programming as actions token according to some specific event, controlling their state through SMSs or through Modbus, if configuring the data logger as Modbus server) in a desirable timing thanks to its 42-bit high performance microcontroller unit. Not to mention its affordable price considering the modules it provides and the number of inputs it has.

Figure 9:king pigeon S272 data logger

Sensors

In our prototype we used a **soil humidity sensor**, since humidity is one of the most necessary parameters for irrigation. A soil humidity sensor indirectly measures volumetric water content using another soil property, such as electrical resistance, dielectric constant or interaction with neutrons, as an indicator of humidity content, which can give an idea about the condition of a plant or even an entire field.

Figure 10: soil humidity sensor

To verify if every given command to the actuator gets accomplished successfully without any issues, a **water flow detector** was used to return the state of the meant to verify actuator (pump).

Figure 11: water flow detector

The central data collection and management system

This part is a necessary one, in every remote management system, because it is the one getting the collected data from the data logger, and managing it. For this task, we choose an **Arduino Uno**, since is very suitable for testing our prototype, what makes it a satisfying choice is the amount of pins (input outputs) it provides, its ability to make a virtual serial canal for data transaction which will be very useful when integrating an GSM module to get the access to GSM network for data acquisition, moreover the price of it is very suitable comparing to other electronic cards that can handle the same applications, furthermore it is the easiest to use one for testing this kind of prototypes.

Figure 12:arduino uno

Modem

As it is said previously, the data logger must send SMSs that hold the proper command and the Arduino receive them, to make it possible for Arduino to get SMSs and calls by accessing to GSM network; a GSM module has to be inserted to ensure the data flow from/to Arduino card smoothly. For this, we have used **SIM900A GSM module** that can be connected to any network by just inserting SIM card in it.

Figure 13: arduino GSM module

Adapters

The USB to TTL converter kind of chips is usually used to interface microcontrollers with computer through USB cable. The one we used specifically is **FTDI** chip which allow the conversion of a USB signal to an UART signal that can be understood by devices like Arduino. This component was actually used to communicate with the GSM module via a serial port terminal through USB cable.

Figure 14:USB to TTL converter

Other converters like USB to RS242 were also implemented to establish the Modbus communication.

3.1.2.2. Software

In order to elaborate the remote management irrigation system, the following programs were used:

King pigeon S272's configuration software

We used the configuration official software to access our data logger to test it, then make the proper configurations for this project. This could be done by interfacing the data logger with the computer trough USB.

Figure 15:S272 configuration software

Emulators/ Simulators

Hercules emulator helped to preview the GSM module, modify the necessary configurations for it and even flashing it before starting to use it. This serial port terminal could be also used to test the GSM module every now and then when facing any issues related to it, as well as hyper terminal.

In order to simulate a Modbus server, **Modbus poll** was exploited, which allowed us to test every Modbus RTU related application, considering the data logger as a server, in addition to **the Modbus simulator**, for testing the opposite case's applications.

Figure 17: Huclues emulator

Figure 16:Modbus poll

Integrated development environment

An integrated development environment (or an IDE) represents the

programing software application that provides comprehensive facilities and tools to programmers and developers. The one we used is Arduino IDE since it's the proper one to program the electronic card used for data collection and management.

Figure 18: Arduino IDE

3.1.3. Project elaboration phases

3.1.3.1. The material tools' check

Considering the rule of any realization project that consists of checking every equipment or component before using them to avoid getting in troubles while integrating them in the total project, the following functions of the data logger were tested:

- ✓ Analogic inputs: checked using a potentiometer that the six of them can give accurate varying values.
- ✓ Digital inputs: checked that all of them get detected as high when affecting ground to them, and as low when affecting a tension value between 0.5 volts and 5 volts.
- ✓ Digital outputs: checked that the relays close when setting the output to high and open when it is set to low.
- ✓ GSM module: checked that the data logger can send messages and make calls according to the preset conditions to the right ID, and that it can do actions (close/open digital output's relays) when receiving a SMS that Holds that command.
- ✓ Modbus configuration: this one was tested by configuring the data logger as a server, and then trying to demand the data from it (read coils with the function code 01, read input registers with function code 04 and read holding registers with the function code 04…) or sending it some commands (write multiple coils with the function code 15 or write multiple registers using the function code 16…) through Modbus RTU protocol using the Modbus client simulator. Furthermore, we checked the functioning of the data logger as a Modbus client with the server simulator.
- ✓ And the last check was the GSM module check, using The FTDI ship to interface it with the compute in order to test its response to AT commands. After installing the necessary firmware to flash the GSM module for the sake of making it work with the available SIM card, it was possible to check its ability to send and receive messages as well as making calls, for example: reading a register number *n* message stored in the r using AT+CMGR= *n*, switching the messages reading format from ASCII to text with the command AT+CMGF=1, and then deactivating notifications, so that we don't have to get rid of them too while receiving and treating the data from the part of Arduino…

3.1.3.2. The data logger's configuration

In order to adapt the data logger to the project's needs, it must have been interfaced to the computer containing the configuration software through an USB cable. Then lunching the configuration program (while the device is on configuration mode) and choosing the proper port (the one to whom the USB cable was wired) as shown in the following figure:

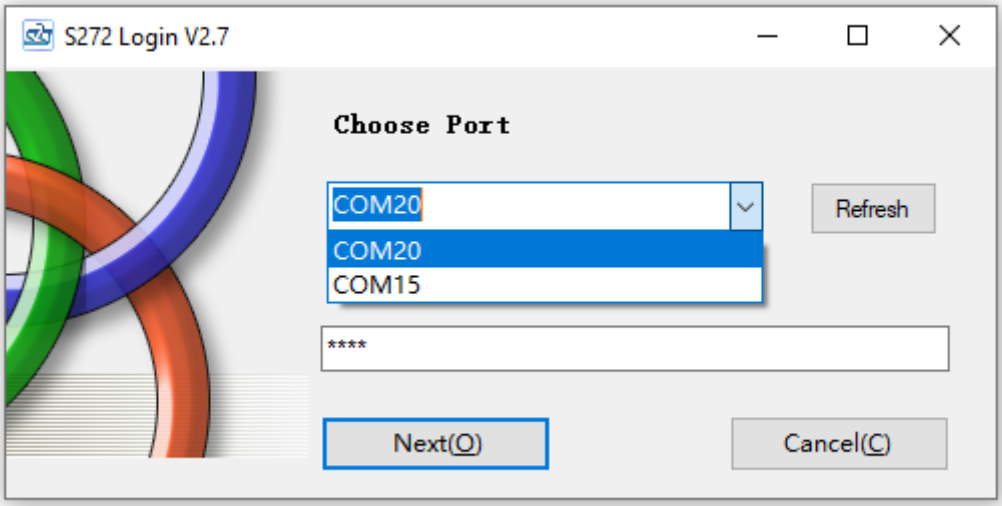


Figure 19: configuration of the port

The next step is choosing the type of the SIM card that is installed in the data logger:

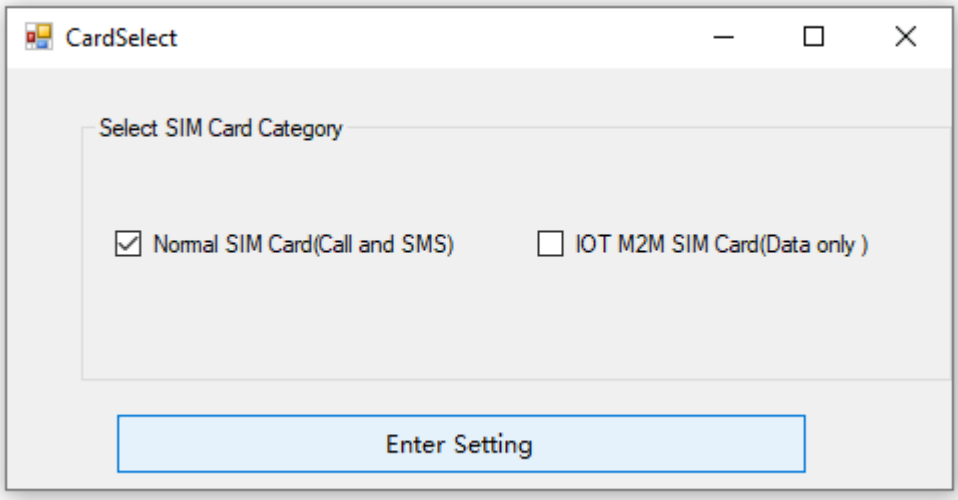


Figure 20: selection of SIM card type

By clicking Enter settings (after choosing the proper options for the port and SIM card category), the following page will appear:

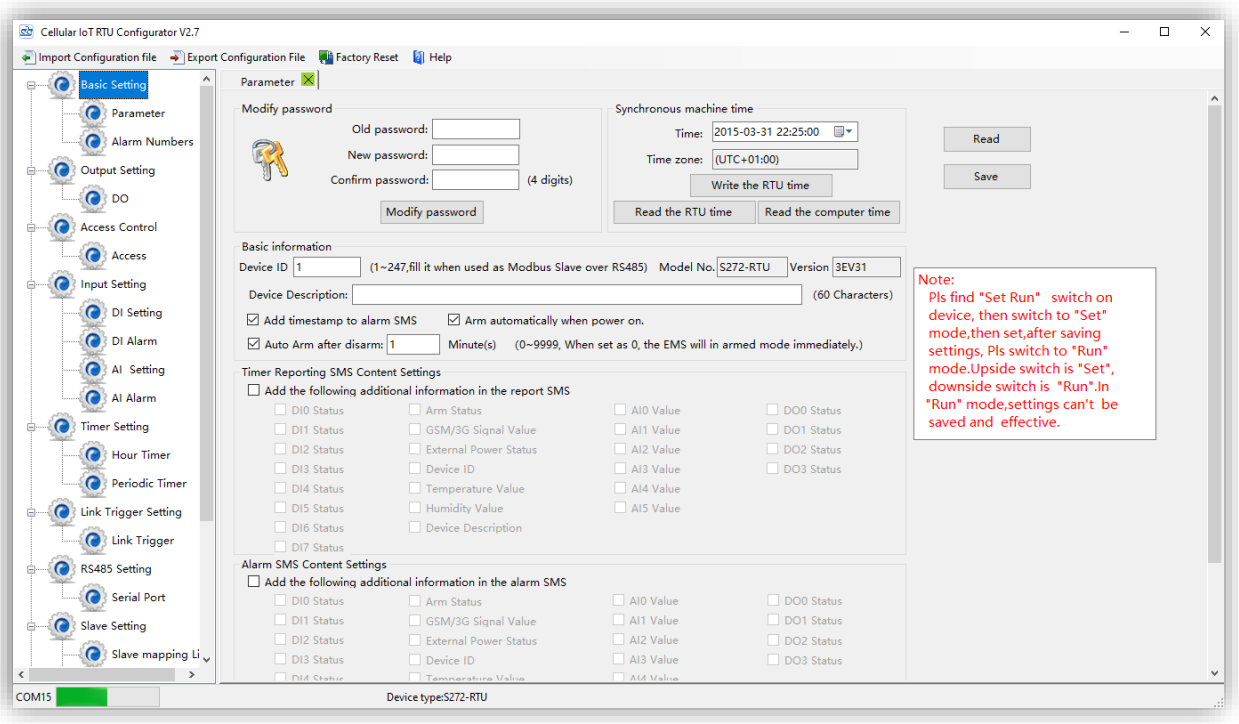


Figure 21: home page

The left side of the interface represents a list of all the categories of configurations the user can make. By clicking each element in the list, a page holding the possible configurations for that category appears, with 2 principal buttons that are the read and save buttons. The first is to show the current configuration of the data logger on the interface, while the second allows

Figure 22:save and read buttons

the user to apply every new modification in the configuration to the data logger.

To enter the settings, it is necessary to set the device to the configuration/setting mode, by placing the switch button (down on the left side) to the upper position. This step must always be done before clicking Read or Save buttons, then switching to the bottom to make the data logger work under the newly done configurations.

Figure 24: instructions for the use of switch buttons

Figure 23: data logger's switche buttons

The left button is responsible of switching between set and run mode, the other six buttons allows the adaptation of the analogic inputs to the used captors, by choosing the measurement type to be in volts or melliamperes.
After getting to know this basic principal, making in configuration would be a simpler task. In our case we have done the following settings to make the RTU's configuration meet the project's needs:
Setting the IDs (phone numbers) with which the data logger may communicate in the project.

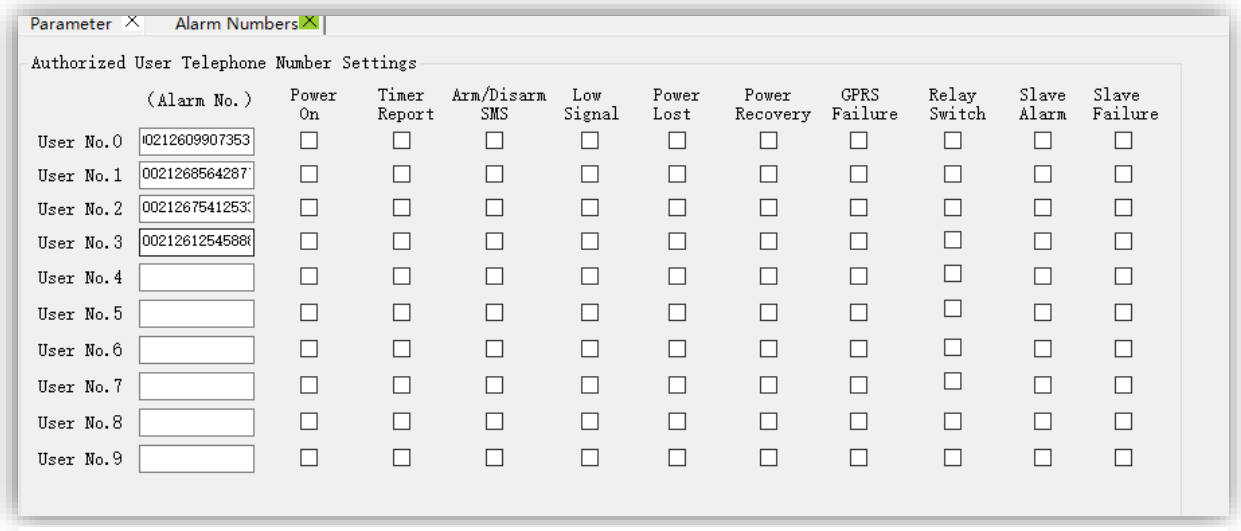


Figure 25: alarm numbers configuration page

The image above is the page where the user can enter IDs and choose what events can trigger the alarms (call+SMS) by checking by checking marking the proper case.
Before wiring the required sensors to the inputs, a configuration to the inputs should be done. For example, we made the following settings to the Analogic input 0:
(When clicking the AI settings category, this page appears)

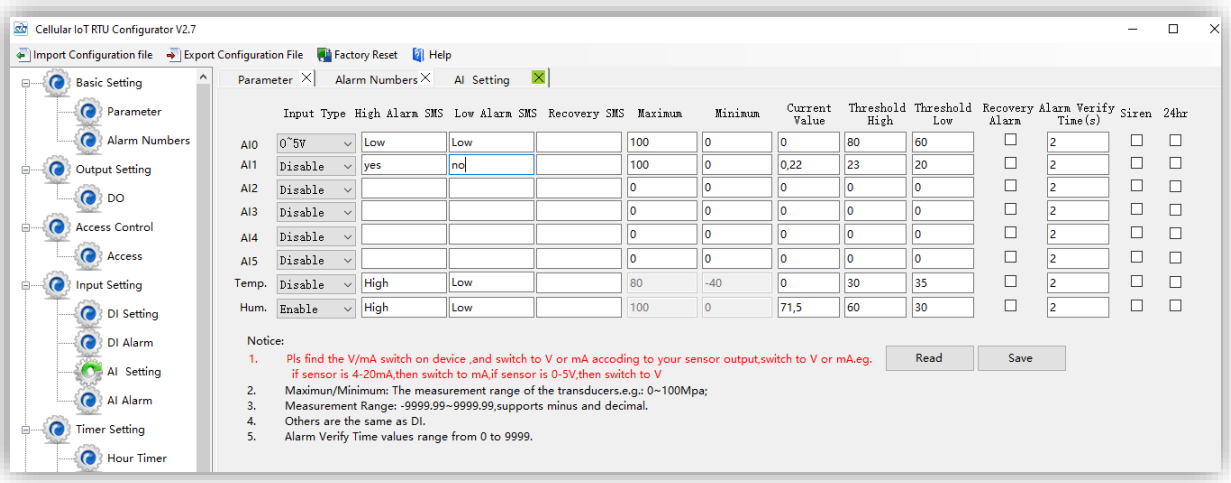


Figure 26: AI settings

After choosing 0-5V as the input type for AI0, we scaled the measurement to 0-100 (100 corresponds to 5 volts in the input, and 0 to 0 volts). We set the high threshold to 80 as it's the humidity value percent of wet soil, and it means that no irrigation is needed in fine soil, while the low threshold was set to 60 percent which refers to the humidity value of dry soil, every humidity bellow this threshold for fine soil means that it needs irrigation. Then we entered the texts to send to specific IDs when surpassing a threshold (sending "Low" when reaching a humidity value beyond the high threshold to turn the pump down, and "High" when the measured humidity value is under the low threshold to turn the pump off). While the alarm recovery time was set to two seconds, since the humidity is not an instantly changeable parameter, so every measured value that doesn't last longer than two seconds won't trigger any alarm.

The next configuration to do was choosing the users to which the data logger will send the alarms when AI0 measurement reaches the thresholds or surpass them by entering the AI alarm category of settings as shown in the image right below.

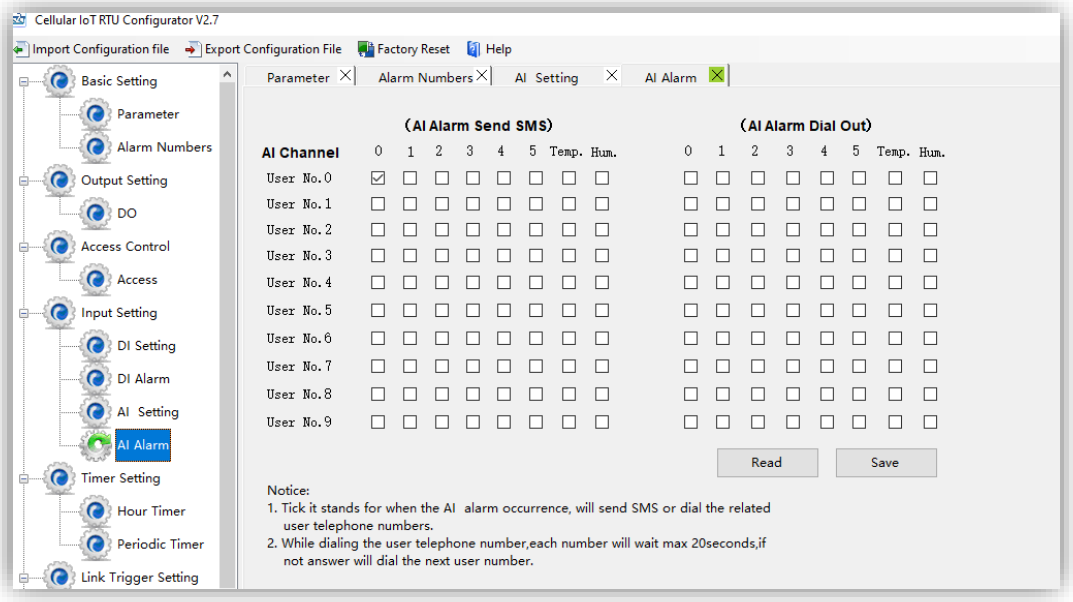


Figure 27: AI alarm configuration

3.1.3.3. Arduino adaptation:

AT commands:

To well exploit the GSM, and make a script that can handle the GSM communication on its own, many AT commands must have been used properly. The **AT** in AT commands stands for attention, these commands allow the control of MODEMS. The following AT commands are the ones we used in our script that will allow Arduino card to receive the data logger's messages continuously:

AT: this command was used to check if the GSM module is connected to the network by responding with "OK".

AT+CMGR = X: this command is for reading the existing SMS in the index X.

AT+CMGD = X: this one, for deleting the SMS founded in the index X, we used it to delete the SMSs after reading them.

AT+CMGF=X: switch the reading and writing mode to text form when X=1, or into PDU form when X=0.

AT+CMGS = "X": sending SMS to the number X, after entering this command, enter the SMS and press "Ctrl+Z" to send it.

Arduino script:

To make understanding clear we split the script into steps:

- Step 1:

Once the Arduino is plugged with power, it starts sending the command "AT" to the GSM module to check if it was connected to the network properly, the next script shows how we do send commands to the module using SoftwareSerial library:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial (4,2);
void setup () {
mySerial.begin(19200);
}
void loop () {
mySerial.print("AT");
mySerial.print("\r");
}
```

if the module couldn't connect, a buzzer will start to ring, if not the Arduino moves to the next step.

- Step 2:

The Arduino sent the command "AT+CMGR= 1" repeatably with the same way shown previously, if the response was "OK", the Arduino send the same command because it didn't find any SMS in the index 1, if the response was something other than "OK" the Arduino moves to the step number 3 because it found an SMS.

```
AT+CMGR=1
OK
+CMGR:RECREAD" ,"+85291234567",,"07/04/20,10:08:02+32",145,4,0,0,"+85290000000",145,49
hello
```

Figure 28: example os an SMS received

The following script shows how to the data acquisition could be done:

```
P=0;
while(mySerial.available()){
data[p]=mySerial.read();
p+=1;
}
```

While this one shows how it was compared with the expected response:

```
o=strcmp(data,ok);
if(o==0){// next step}
```

- Step3:

Taking the sender number and compare it with the data logger number we've already declared, if these numbers are not the same, we send the AT+CMGD command to delete the SMS and go back to the first step for security purpose (avoiding to get the wrong commands from other numbers other than the data logger's), if not we move to the step4.

- Step4:

Testing the content of the SMS, if it contains “High” the Arduino will turn the pump ON, if the content was “Low” it will turn it OFF, after that it will delete that SMS.

- Step 5:

Checking if the Arduino indeed turned the pump OFF or ON by using a water flow detector, if the pump is not working at the right time (for example it should be ON but it is OFF), the Arduino will send an alert SMS to the supervisor that we declared his number already, at the same time the buzzer will start ringing continuously until the supervisor hit the button that breaks the alarm, So the script can retake the first step and so on.

This organizational chart explains the whole process:

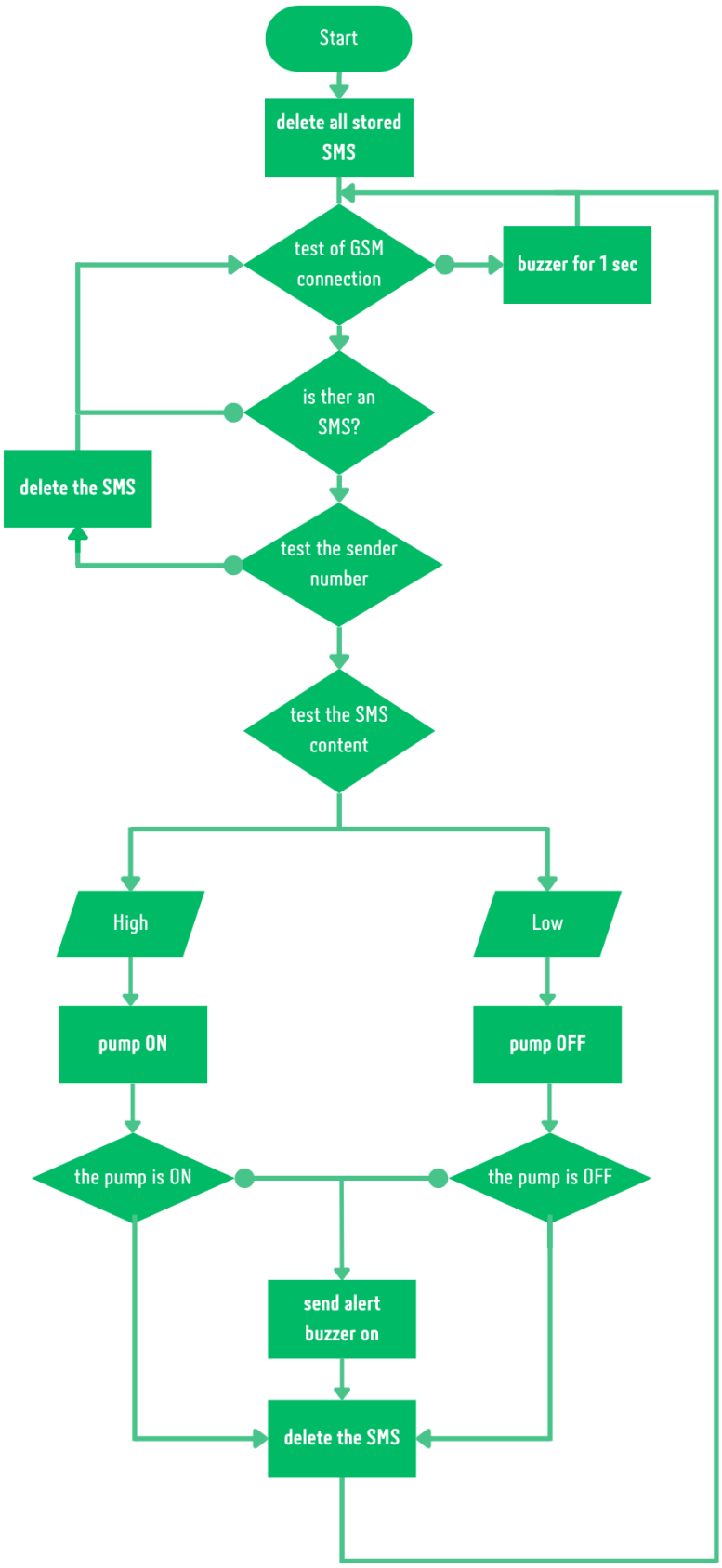


Figure 29: the irrigation system organizational chart

3.2. Creating the Graphical user interface

3.2.1. Specifications

During the configuration of the data logger, we noticed that the official configuration software is truly hard to use especially for a farmer, which mean that the configuration need someone who had a humble experience in the automation, that's why we thought of designing a new graphic user interface (GUI) easy to use to someone who had no idea about this domain.

Our objective now is designing and programming a GUI as the follows:

- ✓ The GUI has to be simple and clear.
- ✓ Gives the farmer the ability of configuring and supervising the data logger
- ✓ It does all the complicated configuration on itself once it gets launched and let the rest simple configurations to the farmer needs.
- ✓ The communication between the data logger and the GUI has to be under the Modbus protocol and through the data logger's RS485 serial port.

3.2.2. Tools

3.2.2.1. Software :

Tkinter

Tkinter is a free graphical library for the Python language, allowing the creation of graphical interfaces, came with all the version of python which mean it doesn't need to be installed, it had all the needed components for a GUI like Labels, dropmenue, buttons …, and we have chosen it for being simple and easy to learn.

Figure 30: Tkinter icon

Pyserial

Pyserial library covers the access for the serial port, it is important to ensure the transaction of the configuration from the GUI to the data logger.

Figure 31: Pyserial icon

Modbus poll:

(has been explained), it will help us to find and count all the needed registers in our GUI.

The data logger official configuration software

we talked about it previously, it will be used this time to accelerate the process of finding register by configurating the data logger in a way that make registers clear to find.

3.2.2.2. Hardware

The physical connection between the GUI and the data logger it has to be a cable capable of converting from USB protocol (GUI side) to RS485 (data logger side), in our case we don't

have this cable so we made one by using two adapters that we have, one from USB to RS232, and the other one from RS232 to RS485

USB/RS232 Adapter

This serial Adapter Cable lets you connect DB9 RS232 serial devices to your PC laptop or desktop computer through a USB port, as though the computer offered an onboard DB9M connector.

Figure 32: USB/RS232 Adapter

RS232/RS485 converter

As it is named, this converter is a transformer from the RS232 protocol to the RS48

Figure 33: RS232 / RS485 converter

Modbus RTU

Modbus RTU is one of the Modbus's communications protocols made by **Schneider Electric** that provides master/slave serial communication via RS-232 or RS-485, to be able to use this Modbus protocol, the serial communication parameters must first be known or set. These include baud rate, parity, and stop bits. In addition, the slave addresses to be addressed by the master are added. The cable length of RS-232 is limited to 15 m and that of RS-485 to 1200m.

Figure 34: Modbus RTU icon

Modbus protocols are communication protocols having in general:

- A single CLIENT
- One or more SERVER (up to 247 slaves from address 1 to address 247).

The client is the only one who can start the transaction by polling a uniquely numbered server on the network and then this server can answer by supplying the requested data to the master in case of monodiffusion, or sending commands to the all of the existing servers in the network using the address 0 in case of general diffusion.

As the other communication protocol Modbus RTU has an unique form of requests and responses, this form represents the way the data was coded, this figure shows the codding of data in Modbus RTU:

Figure 35: Modbus RTU frame

- Slave address: This field is for the address of the slave we want to send the request to
- Function code: the field of the function we want to applicate on that slave
- CRC: (cyclic redundancy check) is an errors detecting code used to detect the accidental errors happened while the transaction.

The functions:

There are 20 different Modbus functions, the table below represent the most used ones,

Function name	Function code
Read Discrete Inputs	2
Read Coils	1
Write Single Coil	5
Write Multiple Coils	15
Read Input Registers	4
Read Multiple Holding Registers	3
Write Single Holding Register	6
Write Multiple Holding Registers	16

Figure 36: the most used Modbus functions

According to our needs we will only use two functions, one for reading and one other for writing (supervising & configuring):

Read multiple holding registers: the function code is 03 when we send a request that has this function the slave will respond by sending the registers we want to read (the maximum of words we can read in one response is 125), this function is the essence of the supervision part in our project.

■ Demande.

N° esclave	3 ou 4	Adresse du 1er mot à lire : PF * | pf *	Nombre de mots à lire n ⩽ 125 : PF * | pf *	CRC 16 pf * | PF *
1 octet	1 octet	2 octets	2 octets	2 octets

Figure 37: request of reading multiple registers

■ Réponse.

N° esclave	3 ou 4	Nombre d'octets lus	Valeur 1er mot PF * | pf *	Valeur du dernier mot PF * | pf *	CRC 16 pf * | PF *
1 octet	1 octet	1 octet	2 octets	2 octets	2 octets

Figure 38: response of readding multiple registers

Write multiple holding registers: the function code is 16, and as you can see by the name this function enables us to write values we want in any register (the maximum number of words we can write in this case is 123 and not 125!).

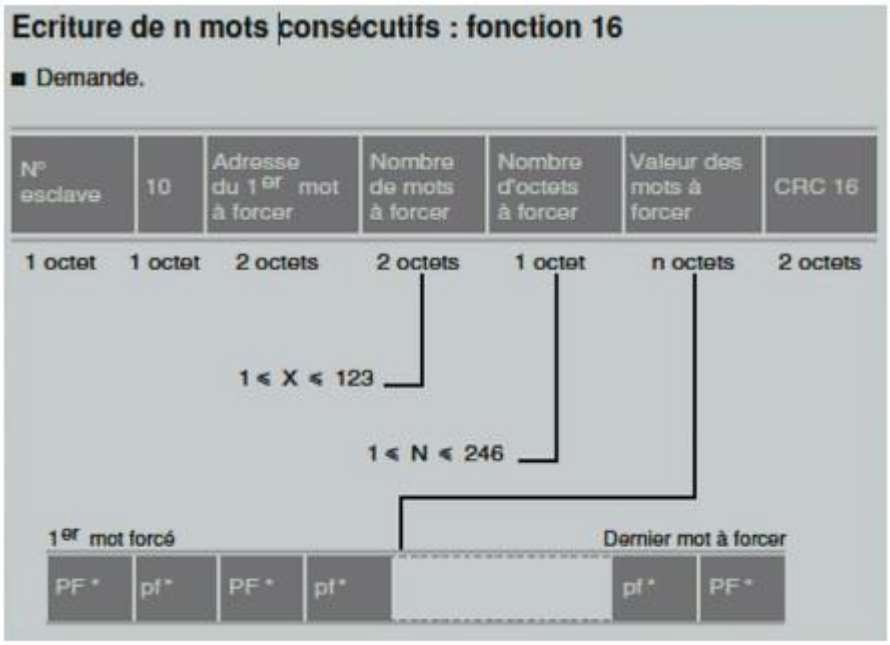


Figure 39: request of writing multiple registers

Figure 40: response of writing multiple registers

CRC calculation:

After constructing the frame (slave address + function code + data), the sending device calculates a CRC code and appends it to the end of the message.

The slave calculates this CRC code from the message it has received. If this CRC code is different from the transmitted code, there has been a communication error.

The following algorithme shows how CRC is calculated:

```
DEBUT
    CRC = FFFFh
    OCTET SUIVANT = premier octet de la trame
    REPETER
        CRC = CRC ⊕ OCTET SUIVANT décalé de 8 bits à gauche
        POUR CPT VARIANT DE 1 A 8
                FAIRE
                CRC = CRC décalé d'un bit à droite
                SI BIT DECALE := 1 ALORS
                        FAIRE
                        CRC = CRC ⊕ A001h
                        FIN SI
                FIN POUR
        OCTET SUIVANT = octet suivant dans la trame
    TANT QU'IL RESTE DES OCTETS DANS LA TRAME
    FIN TANTQUE
FIN
```

Figure 41: the algorithme of CRC

3.2.3. Finding the registers:

finding registers is one of the most important steps of programming a GUI, because simply without knowing the right registers to show or modify their content our GUI is useless, however if we have registers, we can splay any information from the data logger or send any configuration to it without any problems.

To achieve this goal, we start first by configuring the data logger as Modbus slave, connecting its RS485 port with the serial port of the PC and using the software Modbus poll to send a request to read registers.

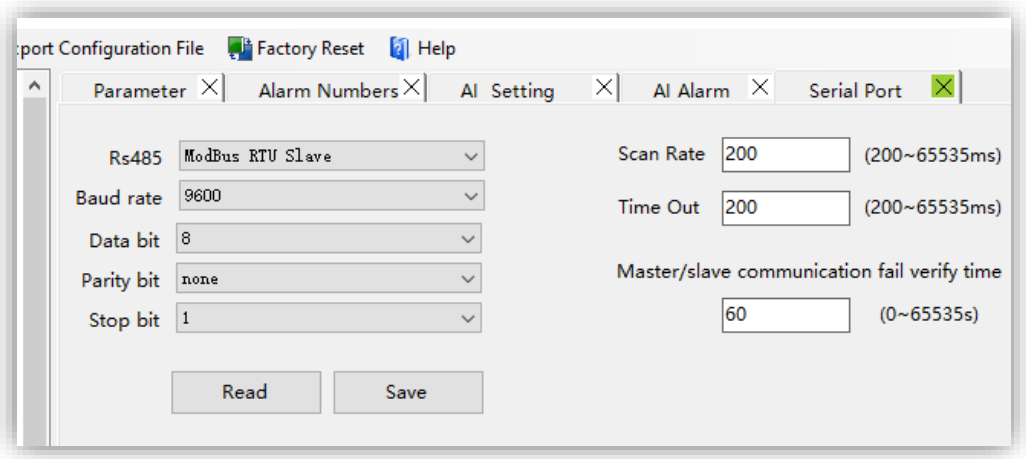


Figure 42: serial port configuration page

Modbus poll guide:

After installing Modbus poll from www.modbustools.com website and open it, you will see the home page of it .

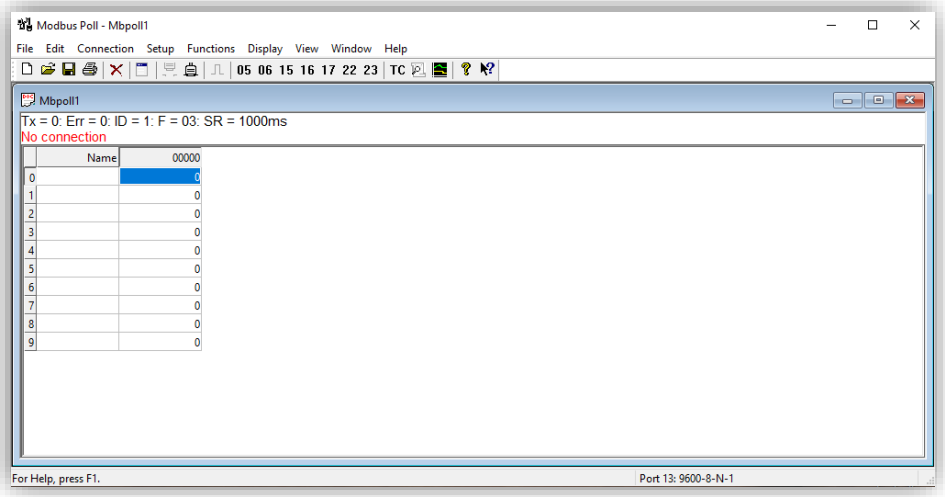


Figure 43: home page of Modbus poll

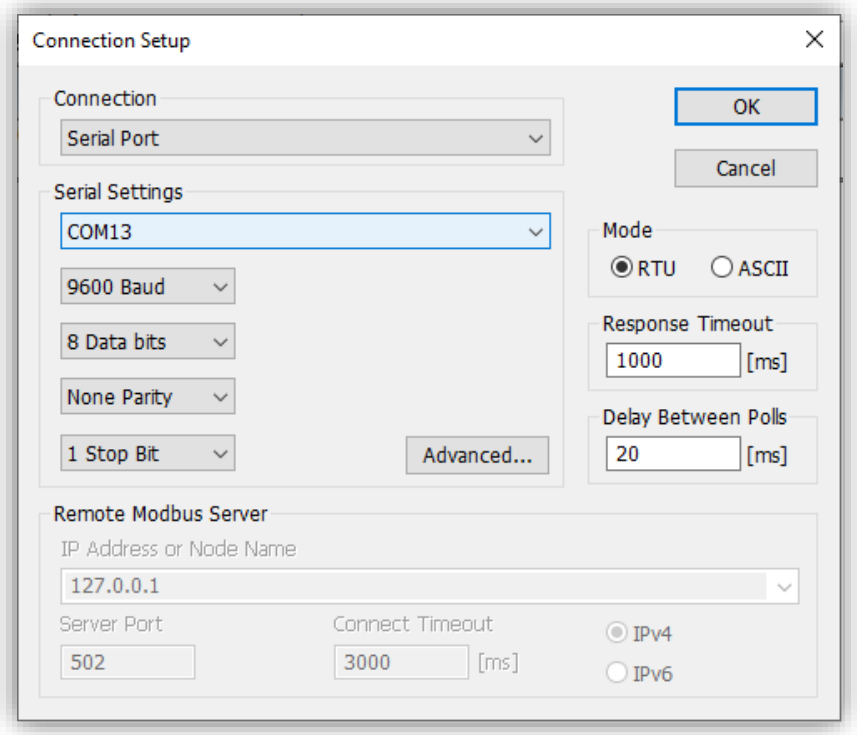


Figure 44: Modbus connection setup page

I.N: make sure that the RTU mode is selected!

The first thing you have to do before start sending frames is to configure the serial communication parameters (the port used, baud rate, Data bits, parity and stop bit), to do that go to connection>connect>ok>ok>ok, then you should see a window like this.

Now you are ready to assembly your frame, go to setup > read/write definition, a window like this will appear

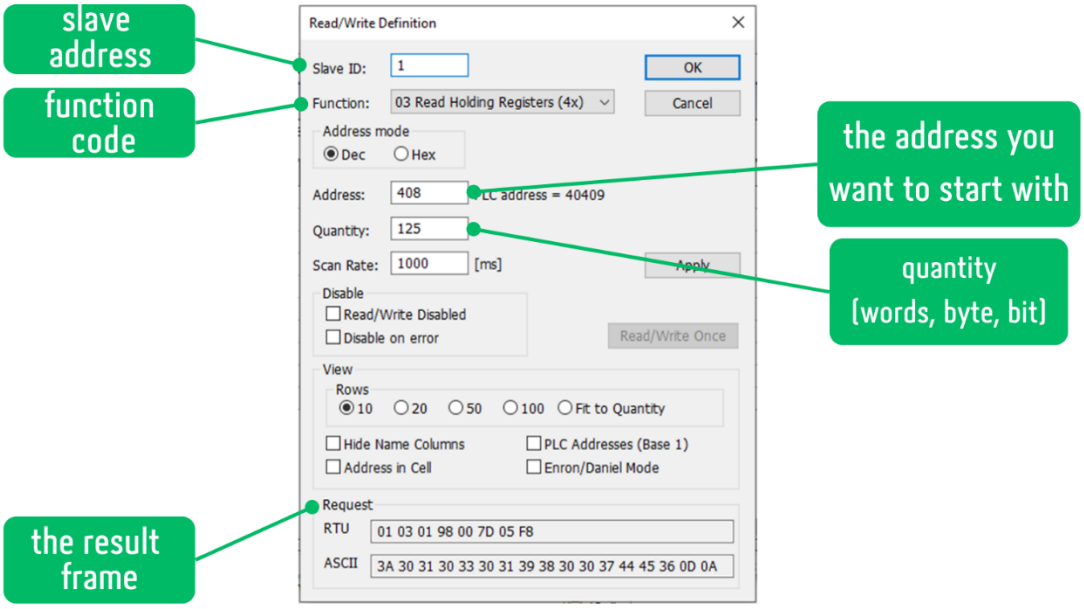


Figure 45: frame assembly in Modbus poll

you can use any function you want, in our case use the data logger address which is 1, and the function 3 to read 125 registers (so we can see the biggest number of registers at once), from the address 408 for example, the frame we get is this: **01 03 0198 007D 05F8**, (the last two bytes are the CRC code), now click on apply and ok to send the request.

Now on the home window the values of each register will be written on a box that has the same address.

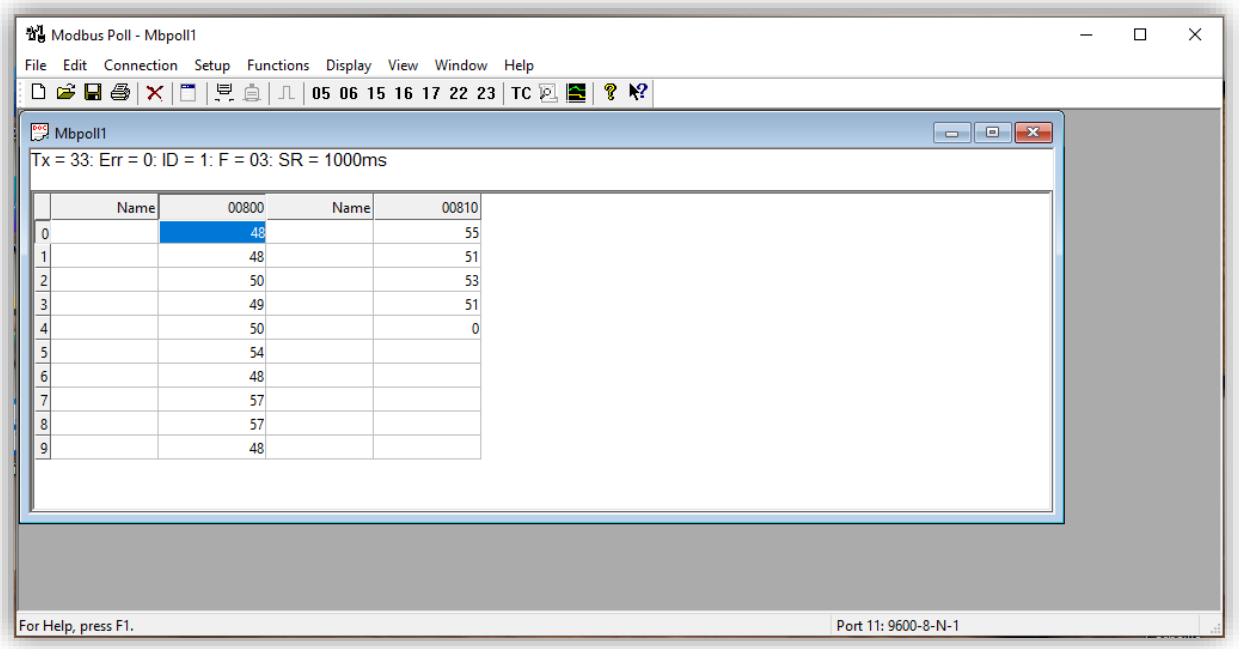


Figure 46: home page shown the registers of phone number in the data logger

But how will we know if these are really the registers we need, here comes the role of the data logger's official configuration software, what we are going to do now is to configure the parameters that will be in our GUI with values easy to recognize, for example we have used number phone of one of us and the high value to be 35 … and for the input registers we've connected a potentiometer to one of the analog inputs and start variating it while searching the registers.

After a while we end up with this result:

The type of analog inputs is a number from 0 to 3 coding on one register with addresses from 280 to 285 for each input. The maximum, minimum, high and low values are codded in four registers (two words) for each value and all of them in sequential addresses starting from 403, the phone number was codded in ASCII code in 14 registers start from the address 800, all of that using the function 03. The current values of each input are codded in the first 6 registers when using the function code 04 (read input registers).

	TYPE OF INPUT	MAX VAL	MIN VAL	HIGH VAL	LOW VAL	current value
A0	280	304-305	306-307	308-309	310-311	1
A1	281	312-313	314-315	316-317	318-319	2
A2	282	320-321	322-323	324-325	326-327	3
A3	283	328-329	330-331	332-333	334-335	4
A4	284	336-337	338-339	340-341	342-343	5
A5	285	344-345	346-347	348-349	350-351	6

phone num	SMS activation	activation of the GSM
800,...,813	370*4,....,406*4	14*2

Figure 47: list of all the needed registers

3.2.4. Designing and programming the GUI:

Now after finding all needed registers, we can actually move to the GUI, we start first by doing some sketch and chose the best and smooth design, the sketch below is the best one we got.

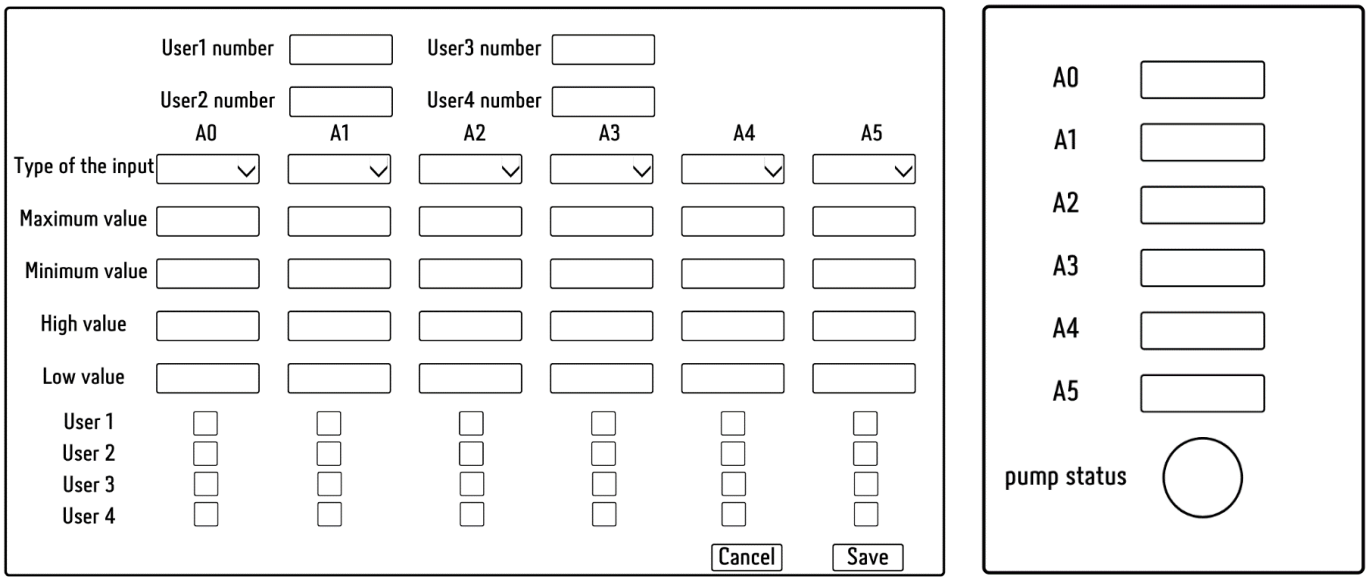


Figure 48: sketch of the configuration and supervision windows

Apparently, these GUI's configuration window gives the user the ability of configuring the data logger phone number that the SMSs will be sent to, the maximum and minimum value of measuring, the high and low threshold values that to trigger the SMSs sending, the type of the input and the activation of the GSM network for each analog input.

Regarding the supervision window, it has 6 text boxes to show the current value of each input and a circle represents a LED that lights up green when the pump is ON and red when it's OFF

To make this design we will use the following function provided by Tkinter library:

- **Entry():** The Entry widget is used to accept single-line text strings from a user in other word it is an input field, we will be using it to get the number phone and all the varying values from the user, the declaration of this function is done as follows

```
Entry1= Entry(root)
```

Entry1: the name of the component, to make difference between the other input fields
Root: This represents the parent window.

To get the text entered by the user, we will use the function **get()** and put the output of it in a variable to use it easily whenever we need it.

```
Number phone = Entry1.get()
```

- **Label():** This widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want.

```
lable =  Label(root, text= ”hello”)
```

text: To display one or more lines of text in the label, using ("\n") will force a line break.
There are two ways to update the text written, either edit it manually or use a variable declared as string or int and use it in the option text.

```
Variable= StringVAr()
variable = x
 Label (root, text= x)
```

Figure 49: input fieled

- **Buttons:** by the name widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button

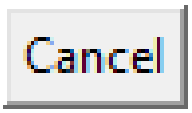


Figure 50: button

```
Button1 = Button(root, text= ”Cancel”,command= function)
```

Command: Function or method to be called once the button is clicked.

- **Option Menu:** is an important part of any GUI. It creates a popup menu, and a button to display it. It is similar to the combobox widgets commonly used on Windows.

```
clicked = StrinVar()
Options = [
    "disable",
    "0V~5V",
    "0mA~20mA",
    "4mA~20mA",]
```

```
drop =  OptionMenu(root , clicked , *options )
```

clicked: the type of menu text
Options: the dropdown menu options that will appear when the user click on the button

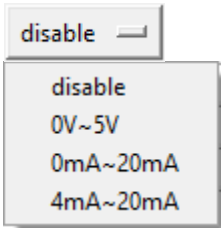


Figure 51: option menu

After employing these functions to reach the sketches we proposed in the first, the result was these GUIs

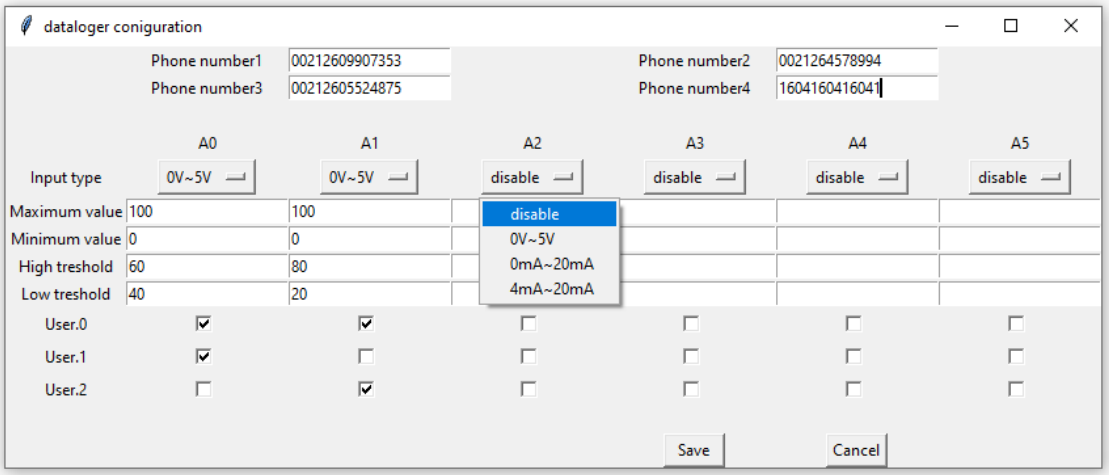


Figure 52:the final GUI

These GUIs alone are useless, because they cannot configure the data logger or even supervise it, what will solve this problem is to give them the ability of communicate with the data logger under Modbus RTU protocol, by sending it frames contain our orders and configuration, all this will be done with Pyserial library, let's start first by giving an overview about its functions.

```
1| para = serial.Serial(port = 'COM13', baudrate = 9600, timeout=0.001)

2| frame= "hello worled !"

3| para.write(frame)

4| para.read()
```

Line1: the declaration of the parameters of communication (port, baudrate and timeout)

Line 2: the data we are about to send

Line3: the action of sending data

Line4: the action of reading received data

Because the parameters of Serial communication are variating from device to another, we add a window to the GUI that appears once the GUI get launched, this window is for the configuration of these parameters

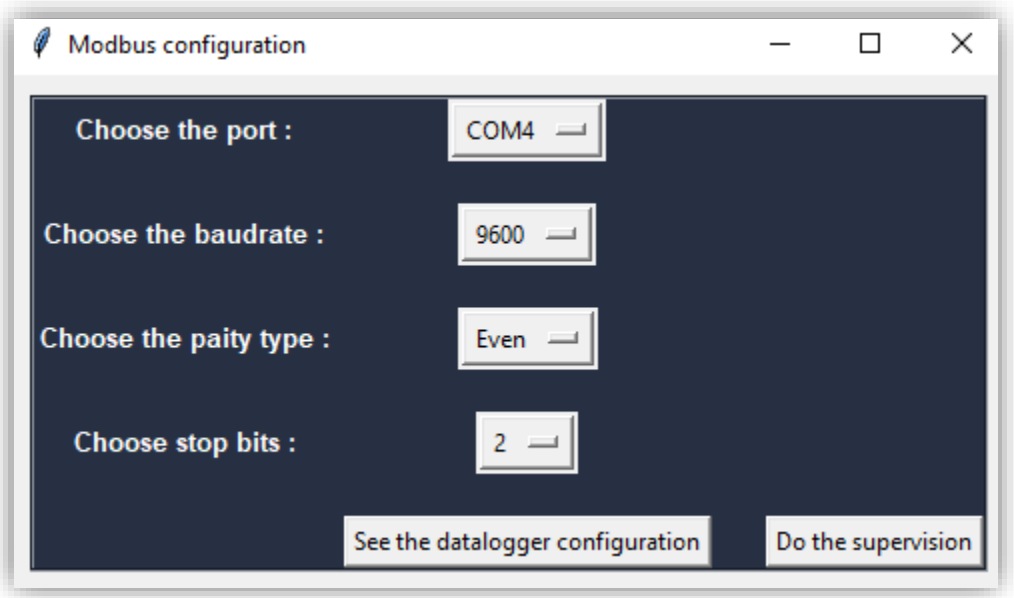


Figure 53:the configuration of serial communication page

3.2.5. frame assembly

In this PHASE we will build our RTU frame and send it with the same method shown previously but with some modifications.

The type of input configuring:

the input type as we have said is numbers coded in one register, if that number equals 0 the input is disabled, if it equals 1 the input is on 0~5v mode, for 2 it is on 0~20mA and for 3 on 4~20mA; the register of all the inputs are sequential from the address number 280 to 285, so we can send them all in one frame, all we need to do is to append the number fits to the user's choice in a frame that has already the static parameters of this case of configuration.

The result frame is this:

`01 16 0118 0006 0C 00IN0 00IN1 00IN2 00IN3 00IN4 00IN5`

: static parameters.

: parameters that depend the user choice.

The configuration of values:

Registers that contain the max/min and high/low values are sequential, and each value is coded on two registers, which mean we can send it in one big frame with the same method, but while searching the registers we figured out that the all values are multiplied by 100, so we should multiply them by 100 and slice them into two bytes (msb and lsb).

This is the first part of the frame:

`01 16 0130 0018 30 0000` **msb(max)lsb(max)** `0000` **msb(min)lsb(min)** `0000` **msb(high)lsb(high)** `0000` **msb(low)lsb(low)** … … …

: static parameters.

: parameters that depend the user choice.

The user number phone:

the configuration of the number is abbreviated in the following organizational chart:

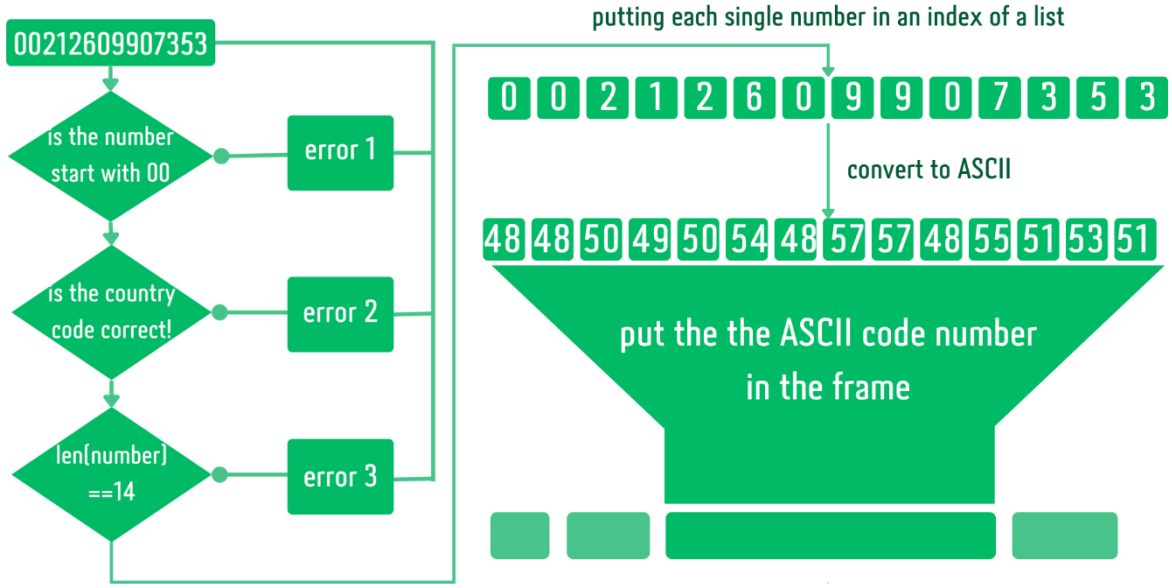


Figure 54: organizational chart of assembling the frame of number

as you can see the msg errors appears when something is wrong is depending the error itself, for example when the user forgot to add 00 At the beginning of the number an error box will appear and showing the next msg.

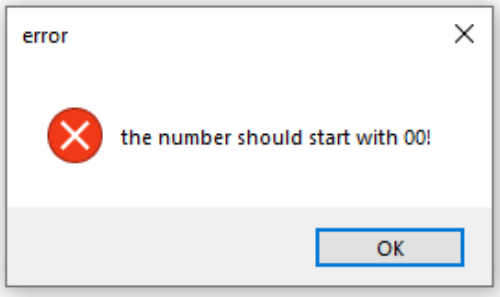


Figure 55: Example of an error

The frame we will get if we used the number in the organizational chart is this:

```
01 10 0320 001C 38 0048 0048 0050 0049 0050 0054 0048 0057 0057 0048 0055 0051
0053 0051
```

the same thing goes with the activation of sending SMSs to each user.

3.2.6. CRC calculation:

As known the CRC is one of the important part of any frame in all Modbus protocols, there is no doubt that if we did not calculate it and appended it to our frame many problems will happen. (see the code that represents the function used to calculate and append CRC to the frame in annexes).

3.2.7. The save and cancel buttons:

the saving button works as follows, once it gets clicked our code starts collecting each frame separately, appending to it the dynamic parts (selected by the user), calculate the CRC and append it then it sends the frame to the data logger through serial port.

Regarding the cancel button, it just cancelling everything and close the configuration window.

3.2.8. Reading the last configuration:

when you focus a little, you will discover that the parameters we didn't configure the code will put zeros in their registers, that means the loss of any last configuration, to solve this problem we make our code reads all registers from the data logger once the user selects the port in the beginning of launching the GUI, in this solution we relied on read function, by reading each register and work on it according to how much registers the values are codded in and if these values need conversion to a readable form for the human users or not, then display each value on the right place in the GUI, so now the user has just to edit the values he want and press save without fear of losing the last configuration.

The previous method of reading is the same used method to show the current value in the supervision window, the only new thing we have added is the function of reading one single coil to read the digital output D0 which configured to close once the SMS of pump activation is sent, if the read bit equals 0 the LED lights up green and for the opposite case red.

3.2.9. Challenges:

Until the day this file was printed, there is one problem we face, which is if the power was lost the data logger lost the configuration, and return the last configuration that was done with the official configuration software instead, the only way to solve this problem is to access to

the data logger's memory, this problem most likely can't happen, because the data logger has a battery, but that doesn't mean it is impossible.

1.1.4 Conclusion:

As a future vision, adding a solar panel is one of the best things that could be done, since the farms don't contain power supplies in every part of it especially the big ones, considering other parameters than humidity in the decision taking of irrigation, such as temperature, calcium, and ph level, and making the user interface wireless (using Modbus TCP IP for example).

General conclusion:

Over the years, agriculture kept facing diverse challenges mostly induced by the climate change and global warming, which hindered its growth and made it harder to reach for agricultural activities' efficiency. Morocco, like all of the world's countries, was affected by these disasters, even was one of the most hit countries by WS&D, because of its geographic location. Instead of giving up on this sector, despite the hardships and difficulties it faces, and the fact that it is highly vulnerable to many natural conditions, the Moroccan government has chosen to stick with the plan of developing agriculture, and leaning on it to refresh the country's economy in order to reduce the gap with the other advanced countries. So it managed to adopt strong strategies to cope effectively with the obstacle of climate change, such as building 170 dams by 2030 and achieving a storage capacity of 30 billion m3 of rainwater, as well as developing the use of non-conventional water resources, by the reuse of treated wastewater and desalination of seawater. This leaded Morocco to become the most advanced country in its region, In terms of WS&D management.

As Moroccan citizens in particular and earth inhabitants in general, who are affected by the previously discussed issue, and who should take environmentally responsible actions, we tried to create an automated irrigation system as a coping mechanism with the actual disaster. This project aims to ensure the communication between the field and the irrigation equipment's, so it won't get irrigated until it is dry and needs to, and the irrigation stops when the meant to be irrigated areas are wet enough. This was possible by creating a network that contains the humidity captors connected to a data logger and a pump controlled by an Arduino Uno with a GSM module connected to it. The data logger send messages holding the proper commands (turn on or turn of the pump) according to measured humidity values in the field, Arduino card with its integrated GSM model receives the message, then implements the received command and make alerts in case the connection with GSM is lost, or the pump is not responding properly to the command. The irrigation model was provided with a graphical user interface that allows its supervision.

To make this project ready to work effectively in the farms, it would be better to add other captors concerning temperature, ph, and other chemical measurements, with an accurate study to decide right moment to command the pumps to tune on or off, making it a smart irrigation model.

"Droughts are hard to avert, but their effects can be mitigated. [...] The price of preparedness is minimal compared to the cost of disaster relief. Let us therefore shift from managing crises to preparing for droughts and building resilience."

UN Secretary-General Ban Ki-moon's Message for
2013 World Day to Combat Desertification
17 June 2013

```
def crc(frame):


      temp=0xFFFF

      for x in range(0, len(frame)):

        temp=temp ^ buf[x]

        for x in range(1,9):

          flag =temp & 0x0001
          temp=temp >> 1
          if (flag) : temp=temp ^ 0xA001

       temp2=temp >> 8
       temp=(temp << 8) | temp2
       temp &= 0xFFFF
       return(temp)

  PFCRC15 = crc(tram15)//256
  PfCRC15= crc(tram15)-256*PFCRC15
  frame.append(PFCRC15)

  frame.append(PfCRC15)
```

CRC calculation code