**General software report**

The following document aims to show the software required for this project to compete with the minimum requirements of the client, the diagram flowchart shows the development that the organization took to execute the process by using an ESP32 microcontroller.

Diagrama

Descripción generada automáticamente

*Image 4.PLC to ESP32 communication flow diagram*

The graph show the basic structure, the module will remain in deep sleep mode until it receives a trigger from the PLC, if it does, the data processing determines the entire process to gather all the data that the PLC send through the RS232 module and the signal processing to obtain the power consumption, the module will get connected to the network and will publish the data into the MQTT topics to falling asleep at the end.

**Deep sleep mode**

The following section will be devoted to providing an in-depth understanding of the sleep mode, its advantages, the functions necessary for its optimal operation, among other relevant information.

ESP32 is capable of light sleep and deep sleep power saving modes.

In light sleep mode, digital peripherals, most of the RAM, and CPUs are clock-gated, and supply voltage is reduced. Upon exit from light sleep, peripherals and CPUs resume operation, their internal state is preserved.

In ***deep sleep mode***, CPUs, most of the RAM, and all the digital peripherals which are clocked from APB\_CLK are powered off. The only parts of the chip which can still be powered on are:

* RTC controller
* RTC peripherals
* ULP coprocessor
* RTC fast memory
* RTC slow memory

**WiFi/BT and sleep modes**

In deep sleep and light sleep modes, wireless peripherals are powered down. Before entering deep sleep or light sleep modes, applications must disable WiFi and BT using appropriate calls

([esp\_bluedroid\_disable()](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/bluetooth/esp_bt_main.html" \l "_CPPv421esp_bluedroid_disablev), [esp\_bt\_controller\_disable()](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/bluetooth/controller_vhci.html#_CPPv425esp_bt_controller_disablev), [esp\_wifi\_stop()](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html#_CPPv413esp_wifi_stopv))

WiFi and BT connections will not be maintained in deep sleep or light sleep, even if these functions are not called.

**Entering deep sleep**

[esp\_deep\_sleep\_start()](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html#_CPPv420esp_deep_sleep_startv) function can be used to enter deep sleep once wakeup sources are configured. It is also possible to go into deep sleep with no wakeup sources configured, in this case the chip will be in deep sleep mode indefinitely, until external reset is applied.

**External wakeup (ext0)**

RTC IO module contains logic to trigger wakeup when one of RTC GPIOs is set to a predefined logic level. RTC IO is part of RTC peripherals power domain, so RTC peripherals will be kept powered on during deep sleep if this wakeup source is requested.

Because RTC IO module is enabled in this mode, internal pullup or pulldown resistors can also be used. They need to be configured by the application using [rtc\_gpio\_pullup\_en()](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html#_CPPv418rtc_gpio_pullup_en10gpio_num_t) and [rtc\_gpio\_pulldown\_en()](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/gpio.html#_CPPv420rtc_gpio_pulldown_en10gpio_num_t) functions, before calling esp\_sleep\_start().

In revisions 0 and 1 of the ESP32, this wakeup source is incompatible with ULP and touch wakeup sources.

[esp\_sleep\_enable\_ext0\_wakeup()](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html#_CPPv428esp_sleep_enable_ext0_wakeup10gpio_num_ti) function can be used to enable this wakeup source.

**Network communication**

IoT platforms provide multiple benefits to companies in the industrial environment: from boosting automation and productivity to generating new management models. However, without network protocols, these systems could not be implemented.

In other words, IoT protocols are to machine-to-machine communication what languages, gestures, or body language are to human-to-human communication. So, just as two humans need to speak the same language in order to understand each other, devices need to use the same IoT protocols to exchange information.

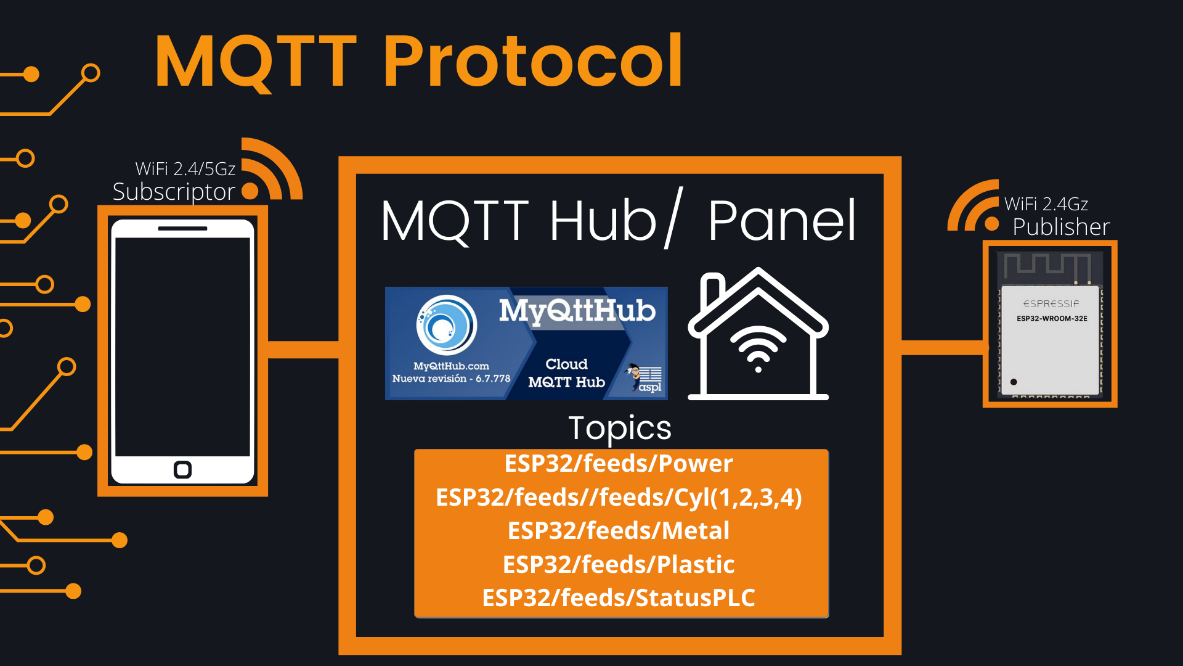
For their operation, the emerging data protocols used in IoT networks have several levels:

* **Application:** the interface between user and device.
* **Network:** it enhances the communication between the router and each of the devices connected to the network.
* **Transport:** facilitates data communication between the different levels and ensures its security.
* **Physical:** the physical communication network between devices.
* **Data link:** responsible for transporting data in the system and detecting and correcting problems.

In IoT device communication with the Internet, the most common protocols are MQTT, CoAP and HTTP. They are highly flexible as they are designed to transmit any type of information.

In this case, the MQTT (MQ Telemetry Transport) protocol will be used. It follows a publish-subscribe model, allowing communication between many devices. For its operation, a central server called broker oversees receiving the messages from the transmitting devices and distributing them to the receivers. The messages are also organized hierarchically by tags.

The central server called broker is linked to the MyQTTHub platform, which is responsible for receiving messages from the sending devices and distribute them to the receivers, in this case would be to receive the variables of the actuators that make up the PLC, the amount of metal parts, plastic and current consumed by the machinery in real time.



*Image 1. Description of communication protocol*

**SOFTWARE MANUAL RS232 to UART**

The RS-232 protocol is an asynchronous serial communication protocol, it does not have an order in which data is sent between devices. that is, it does not have a data sending order between devices, so it is necessary to take care of the synchronization of the therefore it is necessary to take care of the synchronization of the sending to avoid loss of information or communication failures. information or failures in the communication.

Another of its main characteristics is that it is a point-to-point protocol, that is, it only allows communication from one device to another using a communication terminal. to another using a specific communication terminal. It does not allow the creation of networks.

It is a protocol developed for serial communication of single devices, widely used due to the ease of communication and cost advantages of serial communication. It makes use of DB-25 type connectors, however, it is common to observe devices with DB-9, 9-pin connectors due to their lower cost.

The electrical interface uses an asymmetrical electrical connection with unbalanced circuits, all referenced to ground. The logic states are defined by the following voltage levels:

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

*Image 2. RS-232 circuit voltage specifications.*

*As can be seen, the standard considers a logic one to any voltage value between -5 and -12 volts, and a logic zero to those values between 5 and 12 volts positive, with a transition zone (forbidden operation zone) from 5 to -5 volts.*

*The interface is used at a rate of less than 20Kbps for a distance of less than 15m. In practice, these limits can be exceeded by using low capacity cables in electrically low-noise environments.*

Diagrama, Esquemático

Descripción generada automáticamente

*Image 3. RS-232 circuit connector with indication of each of its meanings.*

The data sent over RS232 lines is simply positive (+) and negative (-) voltage pulses relative to a ground reference. A group of +/- pulses sent by a device are carefully timed by the receiving device and decoded into what the hardware settings consider data bit packets. In other words, the RS-232 standard only defines a relatively flexible general electrical framework for transmitting and receiving electrical pulses. What one does with all these pulses ultimately depends on the connected hardware. Things like character encoding, spacing, start bits, stop bits, bit order, error detection, bit rate, etc. are not the responsibility of the RS-232 scope and are set by the user's connected circuitry, usually in the form of a serial communication port and its associated chips and transistors.

The job of the COM port is to make sense of the pulses on behalf of the connected computer or peripheral. For reference, an RS-232 system must transmit from one device (sending on its Tx pin), to a receiving device (receiving on its Rx pin) and vice versa. Do not try to connect Tx to Tx or Rx to Rx in a three-wire RS-232 system! The only pin that is directly connected is the ground connector, which gives both ends a common reference point for measuring pulses. Each RS-232 controller uses inversion logic and employs a single terminal, bipolar output voltage to feed a UART (Universal Asynchronous Receiver/Transmitter). Because the system has three wires and two separate communication channels, it is considered a "Full Duplex" system. Data can be transmitted at the same time as it is receive.

**Data readout**

**-** The first step of the data reading process starts with a check to evaluate if the ESP is connected and ready to receive information, if the buffer remains empty, the ESP will go to deep sleep state to avoid significant power consumption. Otherwise, if the buffer is correctly fulfilled by the proper header command “$”, the ESP will continue the process.

**-** The second step is fundamental, is where the translation of the data obtained is done, for example, each PLC actuator used will be considered as a variable that will be sent in a string of information which will be translated and presented on the Web server to facilitate the visualization of the data.

Each actuator, sensor and component that transmits an important data is represented by a variable, in this case it is represented by variables that are represented from 1-7 respectively, this string of data will be sent by the PLC to the RS232, which will translate the information and with the help of the ESP can interface to the web server, obtaining the data in real time.

* **Actuator 1** :Start-up
* **Actuator 2**: Drill
* **Actuator 3**: Secondary displacer
* **Actuator 4**: thirdth displacer
* **Metal:** Metal products
* **Plastic:** Plastic products
* **PLC Status:** Current state of PLC

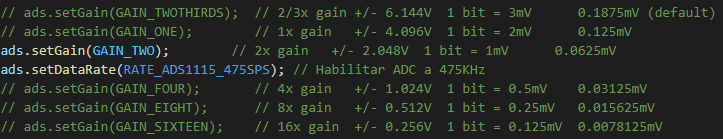
The data is transferred in registers to represent big numbers with two bytes, so it was implemented a set of bitwise operators to obtain the data from the message obtained.

Texto

Descripción generada automáticamente

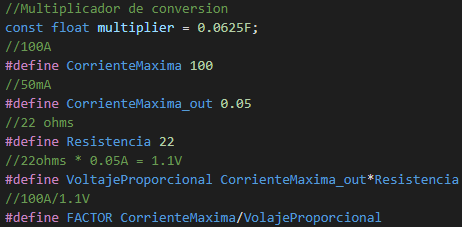
**Power consumption**

The power consumption is carried out by the noninvasive current sensor, which is attached to the ADS1115, it is better known as a twelve-bit Analog to Digital Converter, it is integrated with a Programable Gain Amplifier which is helpful to acquire very low signals. It is normally set to a gain of four that may ensure the proper measurement of the low voltage generated by the current sensor.

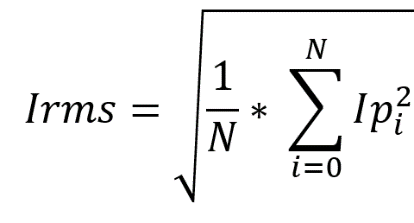


Also, the usage of a sample rate of 475 samples per second provide us a very precise deterministic measurement for a 60 Hz signal according to the Nyquist theorem.

The current sensor is the SCT-013 noninvasive current sensor that it has a fixed factor of 100A over 50mA. The fact that our current sensor is a coil that produces a proportional current brings the possibility to introduce a very precise resistor to obtain a voltage output from the sensor. The usage of a 22 ohms resistor ensures a 1.1 Vp as a maximum peak voltage from the 50mA maximum output current from the sinusoidal wave of 60Hz acquired by the sensor. The calculations may vary from the accuracy of the resistor used but the sensor might remain as 100A over 1.1V.

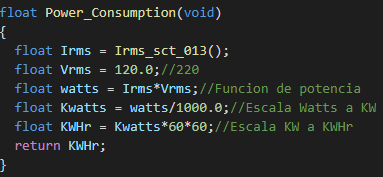


The equation to measure the current is the famous root mean squared that provides the Irms function in a deterministic way, the raw value will be measured by the differential of both terminals which is attached the sensor and will be multiplied by the constant multiplier of the PGA factor, then the millivolts are converted into voltage to being finally converted into current, which will be powered by two and being added into the summatory that represents the function, every cycle will be repeated in one second, the enough time to realize 475 samples according to the sample rate of the ADC.

Texto

Descripción generada automáticamente

The final power consumption is measured by analyzing the total power according to the power equation Irms by Vrms is equal to the power in Watts, which can be represented as Joules per second.



The power is converted from watts to kilowatts, to finally represent the total consumption in an hour, which is finally represented as Kilowatts per hour.

**Server Connection**

**-**The next step is to verify the Internet connection to be able to have connection with the Web server, it is necessary to guarantee the network quality and verify that the server does not present any technical problem, that’s why MyQTTHub was one of the best options, due the flexibility and it is o pen source.

Texto

Descripción generada automáticamente

- Once connected to the server it is time to transfer the data to the server, once the PLC collects the necessary data it sends the information through the RS232 protocol, so that later it can be translated the information collected through variables as mentioned above which will be presented in the UART as $$UPY,XX,XX,XX,XX,XX,XX,X,29$$& and then that information will be sent to the Adafruit server through the MQTT protocol where the decoded information will be observed through graphs and counters in real time according to the different topics in the MQTT broker.

Texto

Descripción generada automáticamente

**Real time data update**

**-**Finally, the information and updates will be presented in real time, so that users have the information about their production process whenever they require it, in this way they can make sales projections, as well as predictions to make changes in machinery or raw material for their optimal production process, and to make changes in the production process.

Texto

Descripción generada automáticamente