UNIVERSIDAD POLITÉCNICA DE MADRID

ETSI y Sistemas de Telecomunicación y

ETSI y Sistemas Informáticos

Máster Universitario en Internet of Things

MASTER'S THESIS

"BIOMETRIC WORK ATTENDANCE MANAGEMENT AND LOGGING WITH A BLOCKCHAIN SYSTEM"

JEIDI PADRON NÚÑEZ

JULY 2021

Máster Universitario en Internet of Things
Master in Internet of Things

Master's Thesis		
Title	Biometric Work Attendance Management and Logging With A Blockchain System	
Author	Jeidi Padron Nuñez	Sign 
Tutor	Sergio Arevalo Viñuales	Signature
Co-Tutor	Maria Isabel Muñoz Fernández	Signature
Director		Signature
Board of examiners		
Presidente/ President	José Fernan Martínez	
Secretario/ Secretary	Rubén de Diego Martínez	
Vocal	Ivan Pau de la Cruz	
Date		
Grade		

Secretary Rubén de Diego Martínez

Acknowledgments

This project would not have been possible without the support of many people. Many thanks to my daughter Sabrina and my husband Pedro for encouraging me to follow my dreams. With their support and love, together we share this journey.

A special thanks to Sergio Arevalo Viñuales and Maria Isabel Muñoz Fernandez, my tutoring professors. Their lessons at Distributed System influenced me to continue investigating the uses of distributed technologies in the IoT world. They offered support, guidance, and dedication during the project development.

To all professors and students at UPM MIoT class, who have been my support during all lessons, projects, exercises, and presentations, I will always appreciate all we have done.

Measure what can be measured and make measurable what cannot be measured.

Galileo Galilei

Table of content

List of figures

List of tables

Summary

The maximum legal working hours in Spain are generally 9 hours per day (maximum 40 hours per week), with a maximum of 80 overtime hours per year. Since May of 2019, the Law 08/2019 requires companies to keep a daily record of the working hours, including the exact hour of entry and exist of each worker, regardless of whether overtime was performed or not. This register should be available and accessible for consultation to avoid possible manipulations.

The project objective is to develop an Internet of Things end-to-end prototype for generating, collecting, storing, and getting reports about employee attendance time. This biometric attendance system uses NodeMCU Arduino board with fingerprint sensor, for generate data and send instant collected information to an Edge node using a Wi-Fi wireless communication. At the edge node, the data is enrichment with extra information and can be filtered and stored inside traditional SQL database. The daily report is generated in Excel format, using a traditional web page.

Finally, a decentralized application was developed to store the daily report inside peer-to-peer InterPlanetary File System (IPFS) and use smart contracts from Ethereum blockchain to keep the daily report immutable and auditable.

The solution has been tested and confirms that it is not possible to modify the daily record thanks to the blockchain properties.

This project has a wide application in school, college, business organization, offices, etc., where marking of attendance is required accurately with time. Thus, by using the fingerprint sensor, the system will become more secure for users.

Keywords: IoT, Arduino, NodeMCU, Blockchain, Ethereum, Decentralized Applications, DApp, web3, Smart Contract, IPFS, Infura, Remix.

1 Introduction

The regulation in Spain Royal Decree Law 8/2019 [1] specified it is mandatory for all companies to keep a "daily record contains the specific start and end time of work for every employee, during four (4) years and accessible for the employees, legal representative and work inspector". The aim of this regulation is to end with a lot of extra work time, unjustified employee absence, and a fraudulent labor contract [2].

General objective:

- In this thesis project, the innovation consists of develop an IoT smart tool to generate a daily attendance report, avoid data modification, keeping this data history immutable, auditable, and available for specific people.

Specific objectives:

- Study of biometric access technology.
- Study of blockchain and peer-to-peer technologies.
- Design and build a prototype biometric attendance system
- Design and build an Edge Node, between the device and cloud computing.
- Design and implementation of Smart Contract in Ethereum
- Design a web service for managing blockchain wallet and managing the access to peer-to-peer and blockchain systems.

This project used quantitative experimental methods to collect and analyze the data stored. Each time any employee enters or exits to office or factory, they must "sign the assistance", that is, the system validates the data using ID biometric authentication. [3]. If the data is valid, then it will be stored their clock-in and clock-out time. At the end of every day, a single report can be generated that contains all the employees' working hours. With the use of a Decentralized Applications (DApps) the daily report can be stored inside distributed nodes Inter Planetary File System (IPFS) [4]. The DApps applications interacts directly with blockchain [5] with full replication on every peer in an "*untrusted peer-to-peer network*". Blockchain transactions are sent to and processed by “random” nodes via Proof of Work and obtained a "*hash*" with a unique value.

Finally, is obtained a fully trustworthy system that we can trust will not be manipulated. There is no more 'middleman' or centralized authority that holds and controls the information. Every node on the network holds a copy of the ledger, which allows for confirmations, for validity, and for a truly trustless system.

2 State-of-the-art

2.1. Legal considerations

The main objective of this project is to keep track of employees' working hours by daily recording their clock-in and clock-out time. These records are traditionally used by companies to know how long has been working an employee at the company.

The attendance system will benefit both the worker and the company, because it offers a legal report to have justification for any legal issue regarding absenteeism and overtime. The Spanish Royal Decree-Law 8/2019, of 8 March specified it is mandatory to have an attendance record with the following legal considerations [1]:

- The company must guarantee "the daily record of working time", including of the specific start and end times of each employee.
- Such record will be organized and documented through collective consultation or company agreement.
- The records must be kept by the company during four (4) years.
- The records must be available to workers, the statutory body of worker representatives, and the Labor and Social Security Inspectorate.
- Noncompliance with the obligation to record working time is classified as a serious infringement, with penalties ranging from 626 euros to 6,250 euros.
- It is mandatory since May 12, 2019.

Therefore, we can extract the following insights:

- The registry must be daily. It is not valid a weekly or monthly report
- The registry must include the start time and end time and it is not necessary to record work breaks. For example, if the employee has time schedule from 9:00 to 14:00 and from 16:00 to 19:00 it must record the entrance at 09:00 and exit at 19:00
- If a company does not register the breaks, it will assume that the record is only for effective work.
- The law does not specify how should be done the registry (paper, Excel data), so in principle, it will be valid any system that was reliable, objective and that cannot be modified or altered by the company

2.2. IoT Platform

2.2.1. Biometric Sensor

Attendance systems are used to track when employees start and stop working. It enables an employer to monitor their employees' working hours, late arrivals, early departures, time taken on breaks, overtime, and absenteeism.

Figure 1. Attendance system using biometric sensor

There are five main different technologies used to keep track of attendance:

- Manual system: This system is still used in many companies, record the entrance and exit of employee in a sheet of paper, and later use an Excel sheet to manually count the working time. Disadvantage: data can be modified, damage or lost.
- Mechanical clock: the first employee time clock was invented in 1888. Its purpose was to record the time an employee entered and left the factory. The employee put a paper card and it were stamp the time. This clock machines were used for about 100 years. Disadvantage: employees can register on behalf another employee and company need to count manually the working hours.
- Automated systems using RFID card: In the mid 1990's the first RFDI reader was invented. The employee receives a plastic card, that contains contains their information like department, section, name and ID number. The employee just put or swipe the card close to reader, the system automatically registers the attendance and saves the data in the system. Disadvantage: employees can register on behalf another employee using another card.

- Automated systems using biometric characteristics: The system uses biometric data in the form of the iris, voice, fingerprint, or face recognition. Disadvantage: high cost and false positive.
- Mobile phone application systems: the latest technology use a mobile APP, based on Android or iOs operating system allowing to register the time for employee working at home or any remote location. In 2020, about 77% of smart phones include biometric capacitive fingerprint reader [7]. The majority of facial or voice recognition processes performed on smartphones are software-based rather than hardware-based. Disadvantage: if someone take the mobile phone, it can delete the biometric identification and replace for new data.

In this project we use the biometric attendance system, using a very cheap sensor that can be connected to any IoT device, compatible with Adafruit Arduino library [8]. It was selected the JM-101 optical fingerprint module [9].

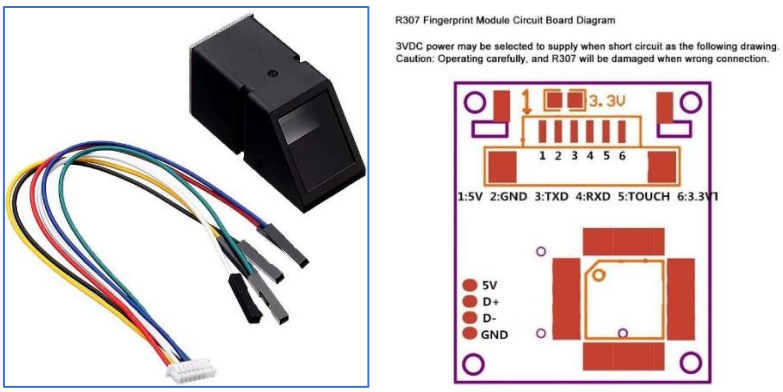


Figure 2. Fingerprint sensor model JM-101

The fingerprint module can be directly interfaced with any microcontroller as well as Arduino Board. This optical biometric fingerprint reader can be embedded into a variety of end products like access control system, attendance system, safety deposit box, car door locking system.

Features:

1. Integrated image collecting and algorithm chip together, with functions such as fingerprint input, image processing, feature extraction, template generation, template storage, fingerprint comparison (1:1) and fingerprint search (1:N)
2. Low power consumption, low cost, small size, Excellent performance
3. Professional optical technology, precise module manufacturing techniques

4. Good image processing capabilities can successfully capture image up to resolution 500 dpi
5. Provide TTL UART interface for direct connections to microcontroller UART or to PC through MAX232 / USB-Serial adapter

For detailed information, please read JM-101B User Manual [10]

2.2.2. OLED Display module

The project includes also an Organic Light Emitting Diode (OLED) display module, used to visualize the system status, like if Wi-Fi signal is available, or the name of employee that enters or exits the working place. OLED is a self light-emitting technology composed of a thin, multi-layered organic film placed between an anode and cathode. In contrast to LCD technology, OLED does not require a backlight.

For this project, it is used 0.96-inch blue OLED display module from AZ Delivery. The package includes display board, display, and 4 pin male header pre-soldered to board.

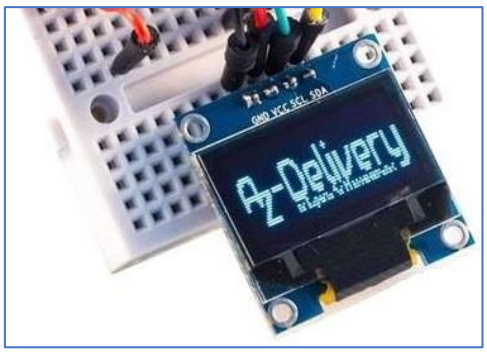


Figure 3. AZ Delivery 0.96-inch I2C OLED Display

Features:

1. Chip set: SSD1306, Arduino compatible
2. Resolution: 128 x 64 pixels
3. Operating voltage: 3.3V to 5V, Low power consumption: 0,04W
4. Pine coating: VCC, GND, SCL, SDA. Control: I2C / TWI

For detailed information please read Quick Start Guide OLED Display [11].

2.2.3. NodeMCU Development kit

Development kit is an open-source electronics platform based on easy-to-use hardware and software that helps to prototype IoT projects. There are many developments kit available in the market. The following table show a comparation for the most common development kit:

Type	B-L072Z-LRWAN1	Rasperry Pi Pico	Arduino nano 33 IoT	Arduino nano	NodeMCU V3
CPU	ARM Cortex-M0+	Dual-core RP2040	ARM Cortex-M0+ SAMD21	Single core ATmega 328	ESP8266
Clock	@ 32 MHz	@ 133 MHz	@ 48MHz	@ 16MHz	@ 80MHz
Flash memory	192 Kbytes	2 MB	32 Kb	1 K	4 MB (32 MBit)
Data memory	20 KBytes	264 KB	256Kb	32 Kb	64 KB
number of pins	52 pins Lora 32 GPIO	26 GPIO	22 GPIO	22 GPIO	16 GPIO
USB port	yes	yes	yes	yes	yes
wireless communication	LoRa WAN	Bluethooth	Bluethooth, Wi-Fi	none	Wi-Fi
Ultra Low power	yes	yes	yes	yes	yes
cryptography	no	no	ATECC608A (TLS)	no	SHA256; AES 128 and 256
Power output	3,3V and 5V	3,3, v	3,3 V, 5V if solder	3,3V and 5V	3,3 V
Price	45 €	7 €	20 €	20 €	15 €
Embedded sensors	no	no	Accelerometer and gyroscope	no	no
Program IDLE	Keil (C++)	micro Python	Arduino	Arduino	Arduino / LUA
Reset button	yes	external	yes	yes	yes

Table. I. Development kit comparation.

To use LoRa WAN it is difficult because there is not any access point with this technology in the near area. Then, it was decided to use a device with Wi-Fi as wireless communication: Arduino nano 33 or NodeMCU.

Arduino nano 33 IoT is a new 2020 product, with few references of uses. Then, it was selected NodeMCU development kit [12], which is stable product, low price, low power consumption, many libraries available (Arduino), USB port available and the use of Wi-Fi as communication interface.

2.3. IoT Edge computing

IoT devices typically are very cheap devices with limited data processing and few storage capabilities. For this reason, is needed a node that handle data processing outside the device. Edge computing allows IoT deployments to perform data processing closer to the end device. This results in lower latency and improved efficiencies in data transport. The location of the edge has various possibilities and will differ according to the use case.

As this project is just a Proof-of-concept, the Edge computing is done inside a laptop with the function of data aggregation, employee validation and generate one daily report. Next figure shows a schema of IoT Edge node, as an Apache web server configured in the laptop.

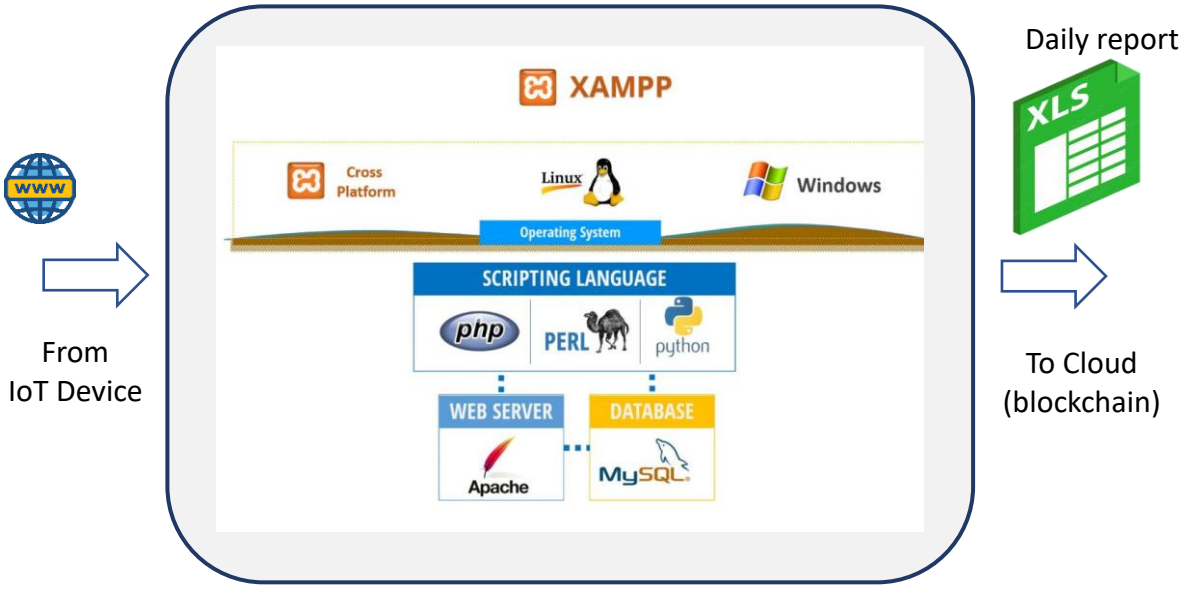


Figure 4. IoT Edge - server development

Note: icons available at Flaticon web page [13]

IoT Edge include real time data processing, data visualization, data enrichment, data filtering and basic analytic. The main objective is to provide daily report with time attendance for all employees. The server-side of the website is known as backend.

- Cross-platform, Apache, MySQL, PHP and Perl, (XAMPP) is an open-source package Apache distribution used for to build a local web server.
- Hypertext Preprocessor (PHP) is a scripting language especially suited to web development. It is considered a backend scripting language.

- Node.js is an open-source and cross-platform runtime environment for executing JavaScript code outside a browser. NodeJS is not a framework, and it's not a programming language. We often use Node.js for building back-end services like Web App
- Database is a collection of data, in our project, the information about employee's time logs. Using Structured Query Language (SQL) let the user access the databases and generate a report in Excel format (table with rows and columns).

2.4. Decentralized Application

Decentralized application (short for DApp) *"is an application built on a decentralized network that combines a smart contract and a frontend user interface*." [14]. DApp are also known as serverless web3. The main characteristics of DApp are:

- **Decentralized** means they are independent, and no one can control them as a group.
- **Deterministic** means they perform the same function irrespective of the environment they are executed.
- **Turing complete**, which means given the required resources, the DApp can perform any action.
- **Isolated**, which means they executed in a virtual environment known as Ethereum Virtual Machine so that if the smart contract happens to have a bug, it won't hamper the normal functioning of the blockchain network.

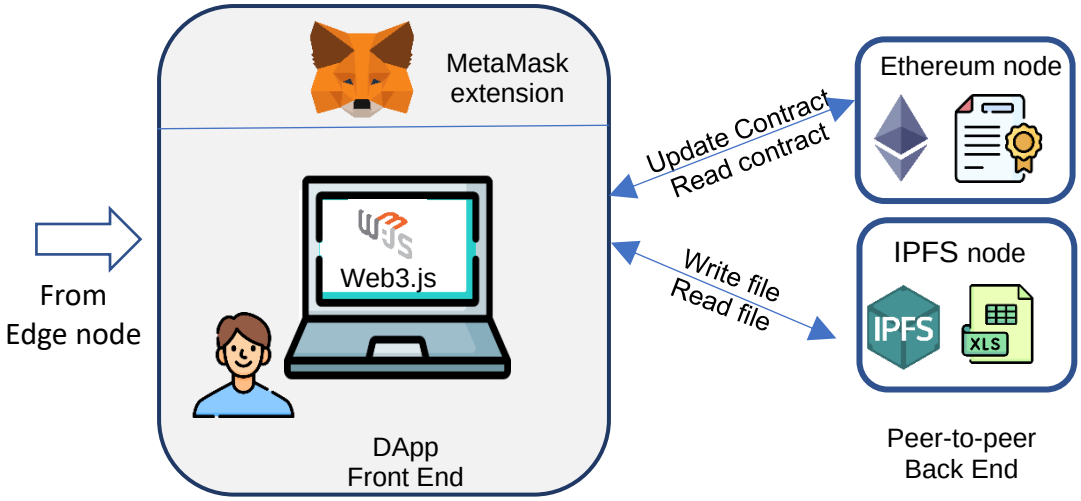


Figure 5. Decentralized application interfaces

MetaMask is a web extension, which allows to manage Ethereum private keys via web browser. It serves as a wallet for Ether and ERC20 (Ethereum Requests for Comments) tokens and allows to execute Ethereum Decentralized Apps (DApps) right in your browser without running a full Ethereum node [20]. That means the execution of smart contracts are deployed in the active account of the wallet in use.

Once DApps are deployed on the Ethereum network nobody can't change them. DApps can be decentralized because they are controlled by the logic written into the contract, not an individual or a company. DApp code can’t be taken down.

2.4.1. Ethereum and blockchain

A blockchain is a chain of blocks that contains information. Each block contains some data, the hash of the block itself, and the hash of the previous block.

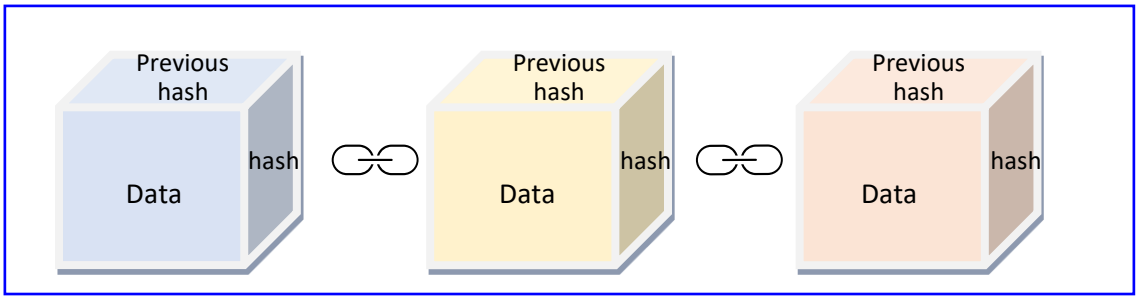


Figure 6. Blockchain of three blocks

The *hash* is a mathematical algorithm used to map data (of arbitrary size) to data of fixed size. Ethereum uses the Keccak-256 cryptographic hash function, where always is returned a value of 256 bits. It is a unidirectional function, since given a data is possible to calculate the hash but given a hash it is impossible to get the data value.

Ethereum is an open-source decentralized infrastructure that executes smart contracts. It uses a blockchain to synchronize and store the system’s state changes.

Ropsten is a test blockchain where Ether doesn’t have any real monetary value. It’s a free blockchain used for development. It is needed it because developers don’t want to spend real money when testing and designing any DApp. [15]

"Miners" are network nodes that solve a complex math algorithm to add a new block, finally obtaining the *hash* creating a "proof of work". This task takes time, consume computer resources and energy. Since each Ethereum transaction requires

computational resources to execute, each transaction requires a fee. The miner who performs the calculation and solve the puzzle first obtain economical reward.

Gas (Ethereum) refers to the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network. There are two parts that determine the cost of your transaction: gas limit and gas price.

- **Gas limit** is the maximum amount of gas the originator is willing to buy for this transaction. The Gas limit cannot be zero as every transaction need a gas. If gas is insufficient, the transaction will not finish, and there is not refund
- **Gas price** is the amount (in wei) that the originator is willing to pay for each unit of gas. The minimum value that gasPrice can be zero, which means a fee-free transaction.

To calculate how much will cost a transaction like create a new block, or execute a contract, just multiply the amount of gas limit by the gas price:

total cost = gas limit x gas price

Example = 21.000 gas x 140 GWei/u

= $2{,}1 \times 10^4$ gas x $1{,}4 \times 10^{11}$ wei/u = $2{,}94 \times 10^{15}$ wei

= $0{,}00294 \times 10^{18}$ wei = 0,00294 ETH

The price for 1 Ether is 1.593,41 euros [16] (March 2021). This transaction has a cost 4,68 €. Read transactions do not modify the state, that means, it does not cost any gas.

Security: Blockchain is intrinsically a very secure technology: if data changed, it is also needed to modify all blocks. To modify one block, take times (as July 2021, between 10 and 20 seconds) [17]. The hacker needs to pay to miners. The blockchain is replicated on several nodes and is impossible to know the data from hash using reverse engineering process. If one block is changed by someone, then the blockchain will alert the complete network to the fact that one block is fake.

Ethereum accounts: To access from real world to Ethereum network is needed to use an account. The access to Ethereum use address and private key for authentication These values are not stored in any database. Only the account owner knows it.

Ethereum wallets are software applications to interact with Ethereum account. The wallet lets users to read balance, send transactions, and connect to any decentralized applications. Most of the wallets also allows to generate an Ethereum account.

Ethereum has two (2) types of accounts or addresses which are:

- Accounts that hold Ethers (ETH): This account is called “Externally Owned Accounts (EOAs)”. You can make an ETH transfer or payment by signing transactions. This account is controlled by anyone with the private keys.
- Accounts that hold ETH and smart contracts: This account is for smart contracts deployed to the network This account is controlled by code.

2.4.2. Smart contract

Smart contract are self-executing pieces of software code that run on a blockchain and contains an agreement between two entities. Smart contract automatically "run" or execute when a relevant set of terms are met. As such, these "contracts" can automatically verify and perform a transaction between different parties.

Smart contracts are a type of Ethereum account. This means they have a balance, and they can send transactions over the network. However, they're not controlled by a user, instead they are deployed to the network and run as programmed. After a smart contract has been coded and uploaded, it will sit in this account and wait to be activated. User accounts can then interact with a smart contract by submitting transactions that execute a function defined on the smart contract. Smart contracts can define rules, like a regular contract, and automatically enforce them via the code.

The best metaphor for a smart contract is a vending machine, as described by cryptographer Nick Szabo in his blog page [18]: vending machine can logic programmed to get a snack, check inputs, and produce output, into be as following:

```
1     money + snack selection = snack dispensed
```

Smart contract characteristics are described in *devteam* blog [19]:

- It must have an open-source code.
- The code is stored in a blockchain.
- The code can’t be altered after it’s deployed.

- The execution is automatic.
- The code and execution results are visible in a decentralized blockchain.
- The execution results are irreversible.

Web3 is the interface where the smart contract interacts with the user. it will use the same structure that traditional web pages:

- Cascading Style Sheets (CSS) allows to apply styles to web pages
- Hypertext Markup Language (HTML) give the structure of the website. It can be used to integrate formatting attributes and layout specifications
- JavaScript is a runtime language for web browsers. The JavaScript will load and can perform actions and make decisions.

2.4.3. InterPlanetary File System - IPFS

In the 2014, Juan Benet wrote a paper explaining InterPlanetary File System (IPFS) as "peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files" [4]. In other words, IPFS is a distributed system for storing and accessing files. These files can be website, application, and data.

HTTP	IPFS
It uses a centralized client server approach	It uses a decentralized peer to peer approach
Data is requested using the address on which data is hosted	Data is requested using the cryptographic hash of that data.
Data cannot be accessed if the server is down or fails or any link gets broken.	Data is copied to multiple nodes, hence it can be accessed whenever needed.
The bandwidth provided is low, as multiple clients request from a single server at the same time.	Bandwidth is high, as data is requested from the closest peer who has the copy of that data.
One has to set up a hosting server or pay for one, in order to make content public available.	Uploading content on the IPFS network does not require a host server, every node hosts the data on the network.
HTTP is well established as an industry standard.	IPFS is relatively newer and is not yet as popular as HTTP.

HTTP	IPFS
HTTP support is inbuilt on almost all machines.	To run IPFS you need to access it using the HTTP to IPFS portal or manually setup up an IPFS node on your machine.
HTTP is used by almost everyone to access the web.	There is a shortage of IPFS nodes due to its low popularity.

Table. II. HTTP and IPFS comparation

IPFS is like HiperText Transfer Protocol (HTTP), used in world wide web to store and access files. The main difference is, in HTTP, when you request a resource, it is needed to know the *domain name* or *IP address* of the server that is hosting the file, while IPFS need to know the hash associated to that content [21]. The following table shows the differences between these protocols. Every piece of content that uses the IPFS protocol has a Content IDentifier (CID), that include its hash.

A distributed hash table (DHT) is a distributed system designed to map values to keys. It is used to support routing and discover content and peers on the network

JS IPFS is the implementation of IPFS protocol in JavaScript. It can run on any browser, inside a service or web worker, browser extensions, Electron and in Node.js.

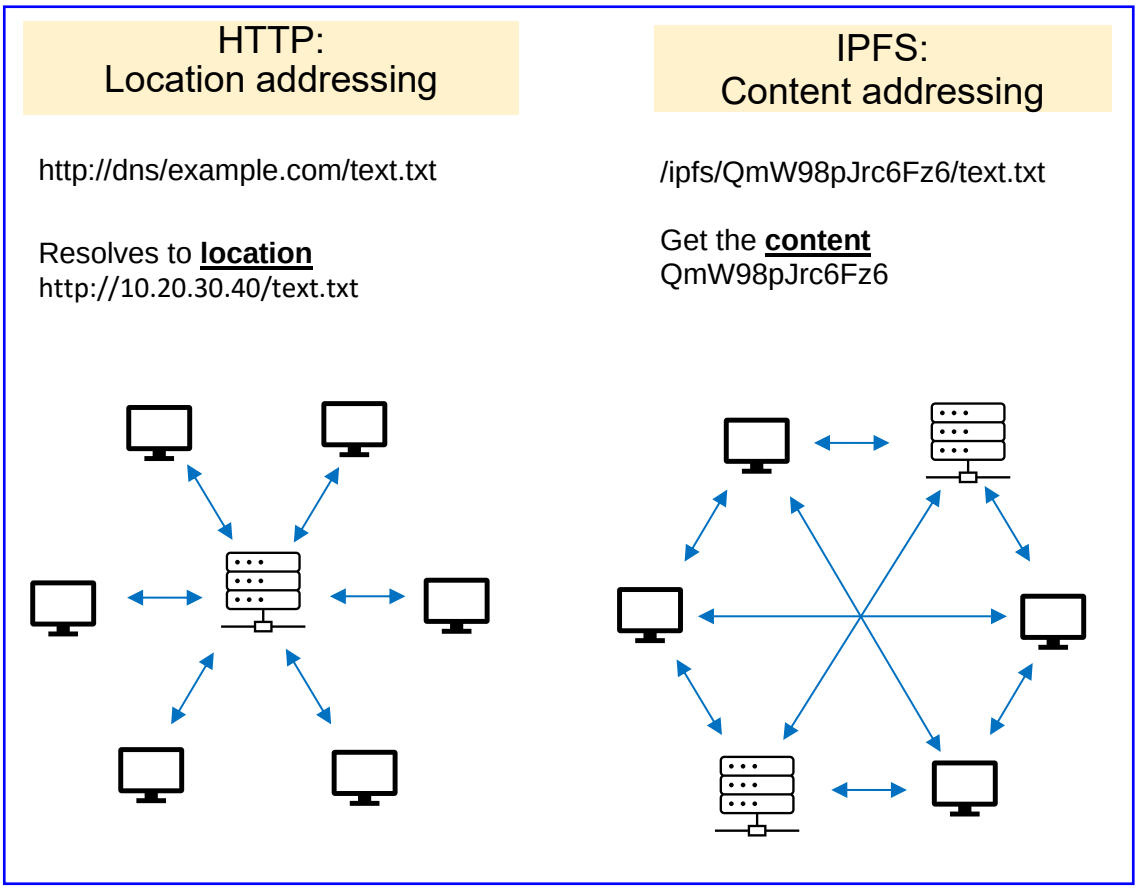


Figure 7. HTTP and IPFS approach

Infura is a collection of full nodes on the IPFS network that enable developers to connect to these nodes through its interface. As such, a significant portion of DApp traffic runs through Infura due to its ease of use, no requirement for developers to run a full node locally, and continual maintenance.

3 Design and Configuration

3.1. Architecture design

This project implemented a complete end to end Proof of Concept. The final goal is to obtain the daily report for employee' time attendance from the blockchain network. The data is generated in IoT device with fingerprint sensor. Then, IoT Edge node process all data into a single Excel file. Finally, connect to distributed Cloud to store the logs in a blockchain. Thanks to blockchain characteristics, this logs, cannot be modified, and people that read the file can trust in it.

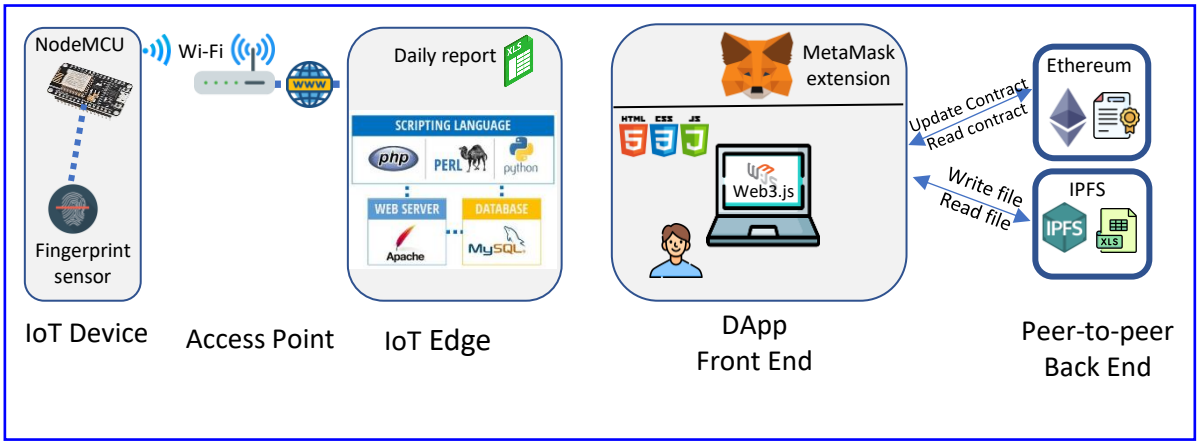


Figure 8. System architecture

3.2. Use case

The system will be used as attendance system, with three main actors:

- Employees: are responsible to use IoT device every time they start and finish to work, using fingerprint sensor. Also, any employee can access the daily report, to verify the attendance time, late arrival, early exit, or overtime period.
- Labor inspector: can get daily report from blockchain.
- Company responsible: it the person responsible for obtaining every day the daily log stored in IoT Edge inside traditional SQL database. As soon as the daily log is extracted, the company is responsible to upload the report in the blockchain network.

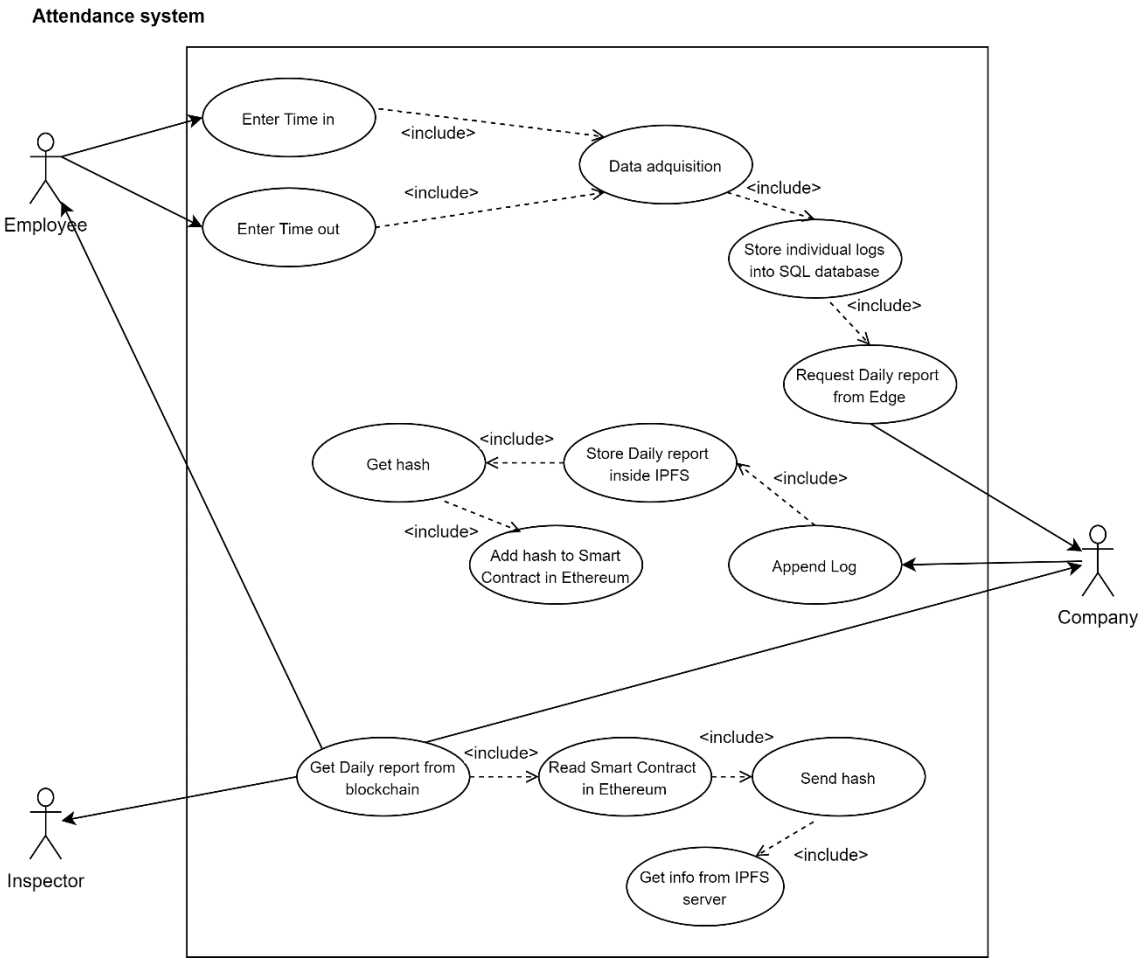


Figure 9. Use Case diagram

3.3. IoT device configuration

3.3.1. Hardware connection

In the following figure is showed the IoT device NodeMCU diagram connection to fingerprint sensor and OLED Display (actuator). Fingerprint sensor use to UART pins and OLED Display use I2C pins, SDA & SCL, as showed in the NodeMCU pinout connection table.

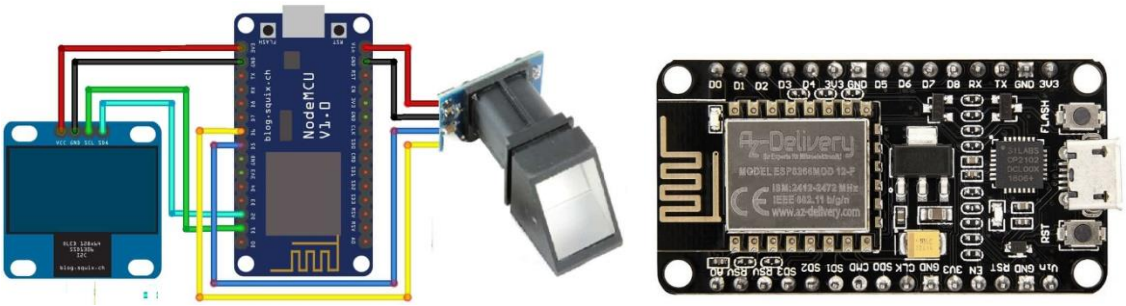


Figure 10. Circuit Diagram for IoT Fingerprint Attendance System

Device	Description	NodeMCU pin number	Color cable
Fingerprint sensor	Vin 3-5.5V	3V3	Red
	GND	GND	Black
	Serial TX	D7	Yellow
	Serial RX	D8	Dark blue
OLED screen	Vin 3-5 V	3V3	Red
	GND	GND	Black
	SCL i2c	D3	Green
	SDA i2c	D2	Light blue

Table. III. NodeMCU pinout connection.

3.3.2. Configure NodeMCU Lolin V3

The configuration was done using NodeMCU user Manual [12]:

1) Download the Arduino IDE from web [23]. In this project was used version 1.8.13
2) Download CH3408 drive, to recognize USB port [24] and connect NodeMCU
3) Set up your Arduino IDE as: Go to File > Preferences. Copy the URL below: http://arduino.esp8266.com/stable/package_esp8266com_index.json
4) Go to Tools > Board > Board Manager> Type "esp8266" and install.
5) Set up your chip as:
   - Tools -> Board -> NodeMCU 1.0 (ESP-12E Module)
   - Tools -> Flash Size -> 4M (3M SPIFFS)
   - Tools -> CPU Frequency -> 80 Mhz
   - Tools -> Upload Speed -> 921600
   - Tools-->Port--> (whatever it is, in our case was COM5)

6) Download and run the 32 bit / 64 bit flasher exe (ESP8266Flasher.exe) at GitHub [25] (Search for NodeMCU flasher at GitHub)
7) In Arduino IDE, look for the Blink program. Load, compile, and upload.
8) Go to FILE> EXAMPLES> ESP8266> BLINK, it will start blinking.

If LEDs start to blink, the setup is finished

3.3.3. Configure OLED display.

The configuration was done using OLED display User Manual [26]:

1) Download external library Adafruit_SSD1306 and Adafruit_GFX
2) Sketch > Include library > Manage libraries
3) After installing the library, restart the Arduino IDE.
4) access the sample code by navigating through menus in this order:
   File > example > Adafruit_SSD1306 > ssd1306_128x64_i2c
5) Define variables to be compatible with NodeMCU and 128x64 resolution

```
#define SCREEN_WIDTH 128          // OLED display width, in pixels
#define SCREEN_HEIGHT 64          // OLED display height, in pixels
#define OLED_RESET     0          // Reset pin
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
#define SCREEN_ADDRESS 0x3D       // 0x3D for 128x64 and 0x3C for 128x32
```

6) compile and upload.

If worked, you will see a moving line at the screen. The setup is finished

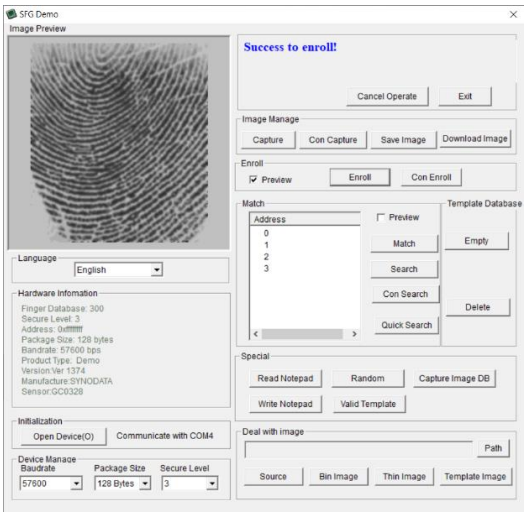
3.3.4. JM-101 Fingerprint Scanner Sensor

The sensor is relatively easy to use. Although this is not the exact same sensor, it is compatible with the Adafruit library [8]. The first step is to test the fingerprint stand alone, using the board as a "bridge" to USB port and configure from Windows PC using free Windows based software. In this way, NodeMCU do not interact with the sensor.

1) Connect fingerprint cables to UART NodeMCU (TX and RX)
2) Load empty sketch to bypass NodeMCU
3) Download DOS program SFGDemoV2.0 from Adafruit web page [27]

4) Start up the SFGDemo software and click Open Device from the bottom left corner. Select the COM port used by the Arduino. If you get an error when you Open Device, check your wiring, try swapping the RX and TX wires on the sensor. Also, close Arduino program as the communication is only bypass

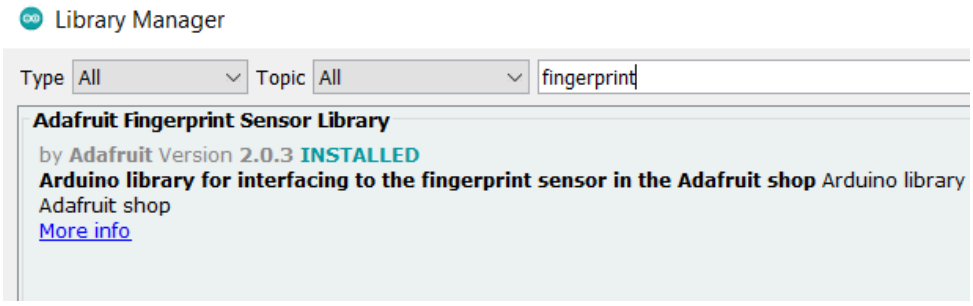
5) Enroll your first fingerprint



6) Now, you are sure the sensor is working, it is time to connect to NodeMCU. Reconnect cables to D7 and D8 pins

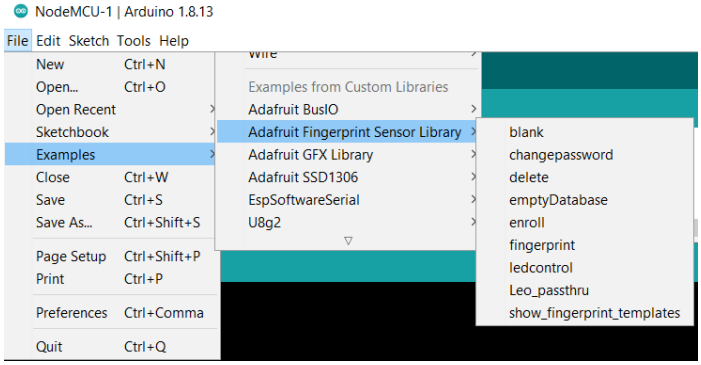
7) Download external Adafruit Fingerprint Sensor Library

Sketch > Include library > Manage libraries



8) After installing the Adafruit Fingerprint Sensor Library, restart the Arduino

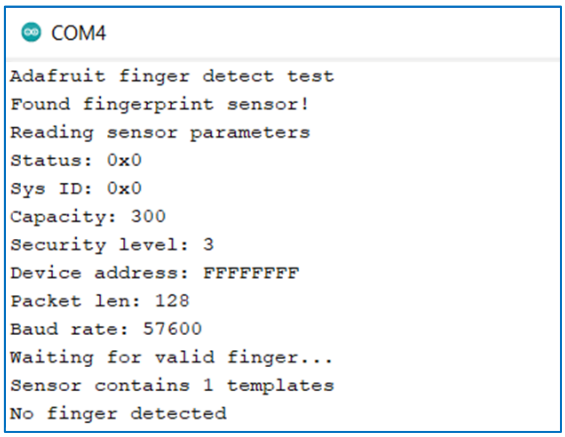
9) You should now be able to access the sample code by navigating through menus in this order: File > example > Adafruit Fingerprint Sensor Library > enroll



10) Declare variables for NodeMCU

```
// * file: enroll.ino - Fingerprint connection settings */
#include <Adafruit_Fingerprint.h>
#define Finger_Rx 14                              //D5
#define Finger_Tx 12                              //D6
SimpleTimer timer;
SoftwareSerial mySerial(Finger_Rx, Finger_Tx);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
```

11) Compile and upload. If worked, you will see on the console the message about fingerprint detection. The setup is finished



3.3.5. Configure IoT Device

On previous steps, we have tested all components stand alone to guarantee proper cabling connections and performance of the device. The NodeMCU use only a single *.ino code for all components: NodeMCU, fingerprint sensor, messages displayed on OLED screen and Wi-FI connection to IoT Edge Node.

For this project, the following code was used as a library and modified according to our project: https://github.com/InfinityWorldHI/Biometric-attendance-system-V1.0

1) Configure Wi-Fi credentials

```
const char *ssid = "Wi-Fi_name";                  //Enter your WiFi values
const char *password = "12345678";                //Enter your WiFi pasword
```

2) Initialize devices and services. Loop to check if any fingerprint is available

```
void loop() {
  CheckFingerprint();     }         //Check the sensor if there is a fingerprint.

void CheckFingerprint(){           // Get the Fingerprint ID from the Scanner
```

```
  FingerID = getFingerprintID();
}

int getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:             //Serial.println("Image taken");
    case FINGERPRINT_NOFINGER:       //Serial.println("No finger detected");
    case FINGERPRINT_IMAGEFAIL:      //Serial.println("Imaging error");
  }
```

3) Add the communication with IoT Edge, that is, send and receive data

```
const char* device_token  = "6b07332bc7f1c0b9";           // IoT device ID
String getData, Link;

void SendFingerprintID(int finger){                  //Sending Fingerprint ID

    if (payload.substring(0, 5) == "login") {
      String user_name = payload.substring(5);
      display.print(F("Welcome"));
      display.print(user_name);                     // Employee enter to work
    }
    else if (payload.substring(0, 6) == "logout") {
      String user_name = payload.substring(6);
      display.print(F("Good Bye"));
      display.print(user_name);                     // Employee ends to work
    }
```

To see complete configuration file, see Annex B: IoT Device Arduino Code and Annex C: change Bitmap to array.

3.4. Edge node configuration (centralized)

The main function of the edge node is to aggregate input values over each time interval and calculate a single output value. The input is obtained from IoT device when the employee registers the attendance, and the output is a daily Excel report.

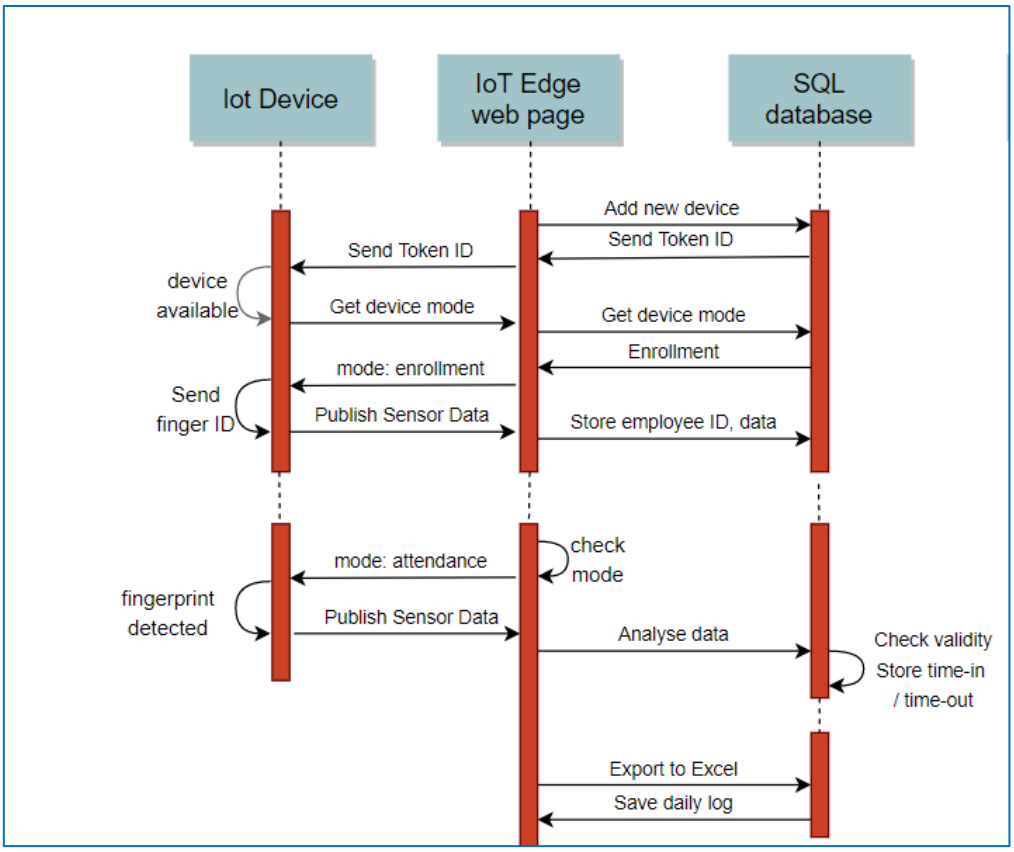
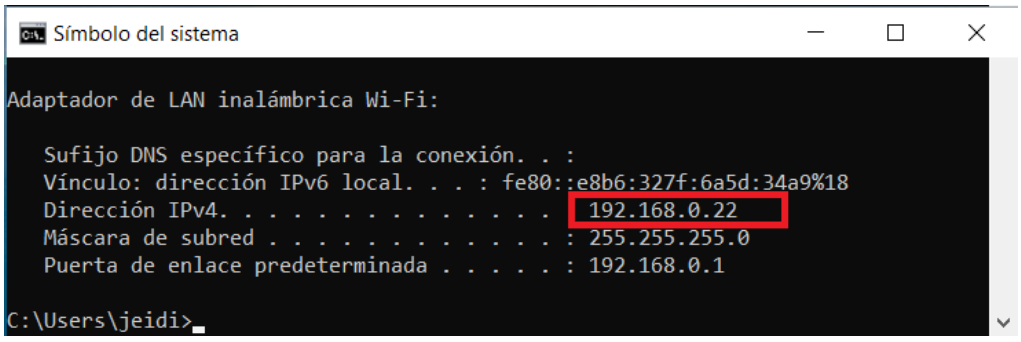


Figure 11. IoT Edge Sequence Diagram

Everyday can be generated a single Daily report containing all time attendance for all employees. If assume a small company with 25 employees, every year the system handles 26.000 logs a year, and can be reduce by 52 registry a year.

To create an IoT Edge server, the following code was used as a template and modified according to our project: https://github.com/InfinityWorldHI/Biometric-attendance-system-V1.0. Steps to configure:

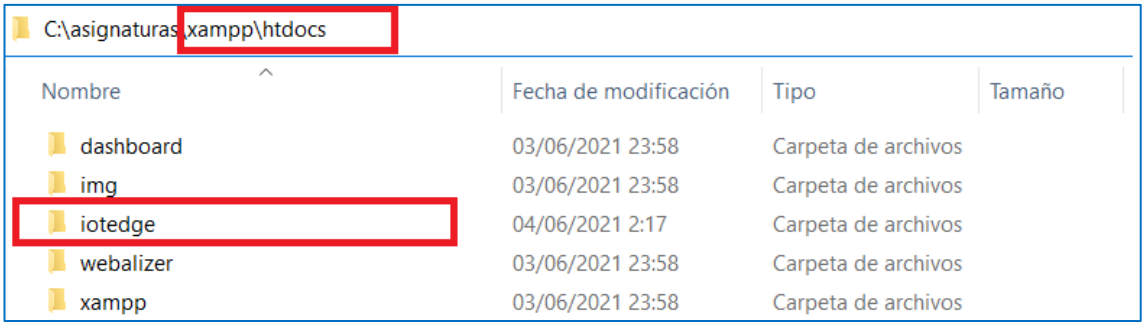
1) Open CMD window and check the server IP (IP config command)




```
String URL = "http://192.168.0.22/iotedge/getdata.php"
```

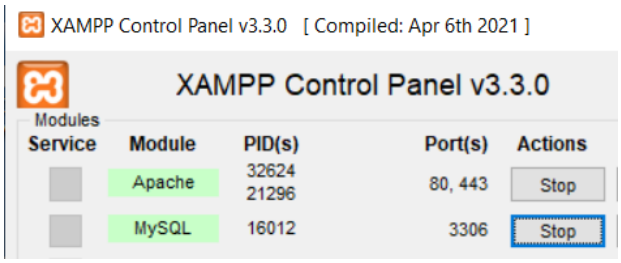
2) Install a local server xampp server [28]

3) Go to GitHub [40]. Copy folder "*biometricattendance*" into C:\...\xampp\htdocs folder and rename it to " *iotedge* "

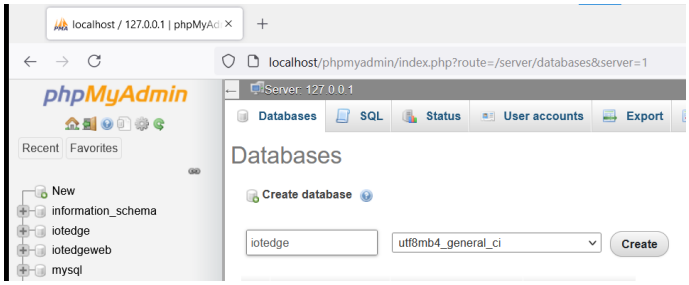


4) Add new database to server.

- run xampp -> start apache -> start MySQL services (click start button)

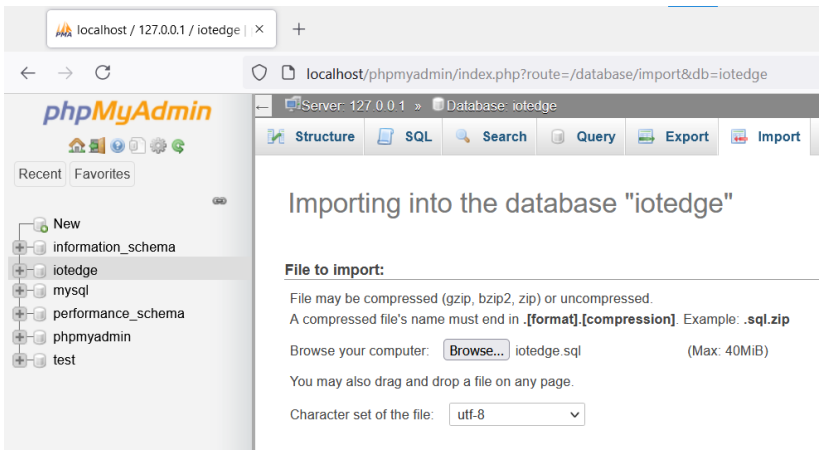


- open Firefox browser -> open http://localhost/phpmyadmin/
- click on new -> database -> create



5) Import database to server, click on import, browse

- select file " iotedgedb.sql", click on go button



6) check the database is complete.

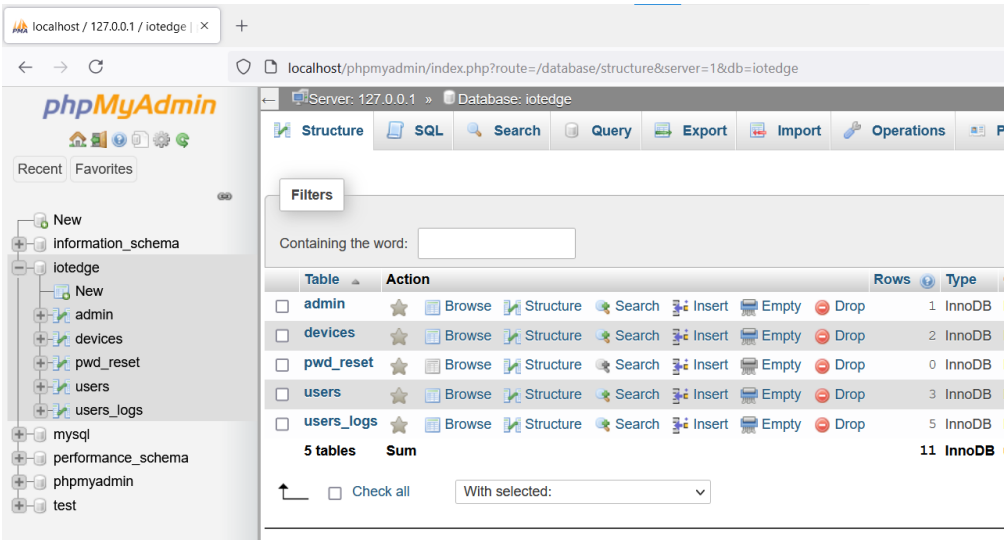


Figure 12. Import database to server

7) Open a browser and login to http://localhost/iotedge/index.php to start.

E-mail: admin@gmail.com password: 123

8) Add device: devices > New Device -> Create

Device Name: NodeMCU1 Device Location: Madrid_building1 Dev.UID: ea16b938f7e73a9d

9) Update NodeMCU code with Dev UID vale and load program

3.5. Decentralized application configuration

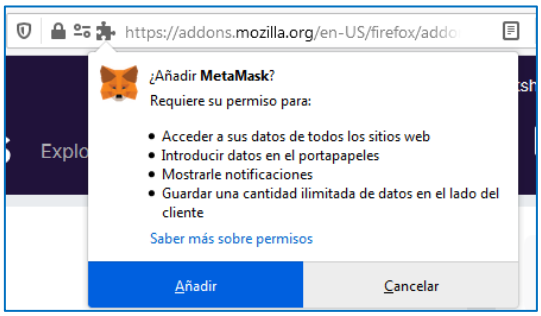
3.5.1. Smart contract

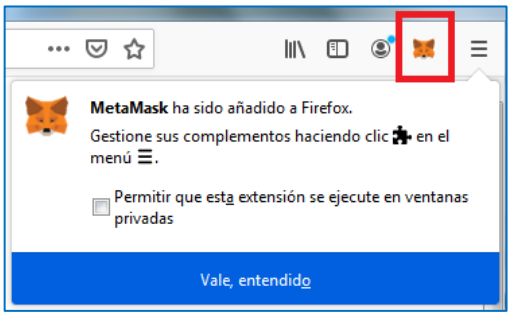
Install MetaMask.

MetaMask is a web extension that runs in many browsers like Chrome, Firefox, Opera or Brave. With MetaMask, private keys are only stored on your device. With this identity, you can perform different actions using the website as an access point and/or as a graphical user interface.

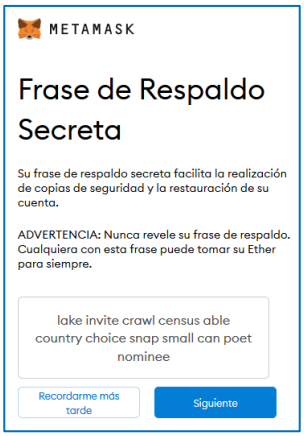
In this project uses the tutorial about create a simple DApp from MetaMask webpage [20]. Installation of the MetaMask Wallet:

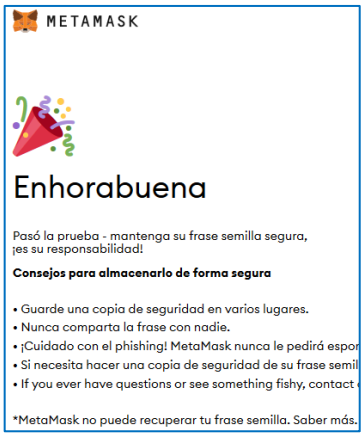
1) Open official MetaMask [29] page in Firefox browser and Click on "Install"
2) The installation will open Mozilla Firefox Addons extension. Click on Add to Firefox. Approve all permissions. If extension was added, it is possible to see a confirmation message, and see the MetaMask logo in the tools bar.





3) The system will ask to use existing account or create a new one. Click on Create Wallet. The system will show a seed phrase, with 12 words, to be used in case forgot the password. It is important to store the seed phrase in a safe place because it is the only way to recover the account.





4) See the account number: click on three dots to display account details

Get founds to deploy Smart Contract

Deploying a smart contract is technically a transaction, so there is needed to pay for Gas in the same way that it is needed to pay gas for a simple ETH transfer. A standard ETH transfer requires a gas limit of 21.000 units of gas, while a contract deployment need about 100.000 units of gas, depend on contract complexity.

For deploy smart contract on the main network real ether costs money and handling it requires a bit more experience. For this reason, it will be used the Ropsten Ethereum, a testing network. Faucet is a pool of Ropsten Ethereum from where you require rETHs

Request ETH from faucet at no cost:

1) Open faucet page [30] using the same web browser with MetaMask, and click on MetaMask extension
2) Select Ropsten network. Click on "request 1 ether from faucet" button and wait some seconds (the transactions need time to be executed)
3) Verified the account contains 1 ETH and show the transaction ID.

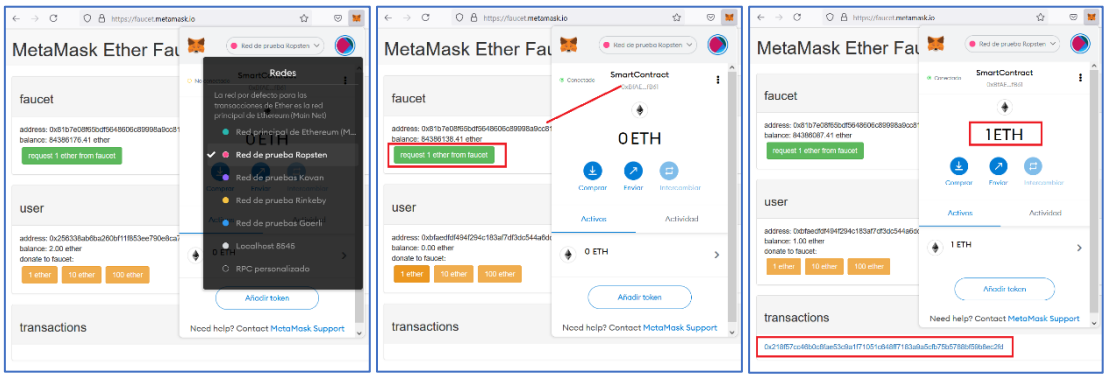


Figure 13. Request ETH from faucet

Create and deploy Smart contract

Ethereum has developer-friendly languages for writing smart contracts like Solidity [31] and Vyper [32].In this project the smart contract was writing as a text file, named "*timinglog.sol*" using Visual Code and compiled in Remix IDE (Integrated Development Environment) [33] using Solidity language. This web application can run on Windows or Unix environment, and it is used to write, debug, and deploy Ethereum Smart Contracts.

Steps to create a contract named *timinglog.sol*:

1) Install and open Visual Studio Code (VSC) [34] to create the file. Add state variables: solidity version and "contract" name, for example *timinglog*

```
pragma solidity ^0.6.1;
contract timinglog {
    uint constant DAY_IN_SECONDS = 86400;
    uint constant YEAR_IN_SECONDS = 31536000;
```

2) Define functions. In Ethereum is important to distinguish between a write function and read function, because write function will modified blockchain creating a hash and read will only search specific hash

```
function append_log(uint _company, uint year, uint month, uint day, string memory
_log) public returns (bool OK) {
  bytes memory tempEmptyStringTest = bytes(comp_date_hash_map[_company][local_date]);
```

```
  require(tempEmptyStringTest.length == 0); // must be empty the log at this day
  comp_date_hash_map[_company][local_date] = _log;
  return true;
  }
```

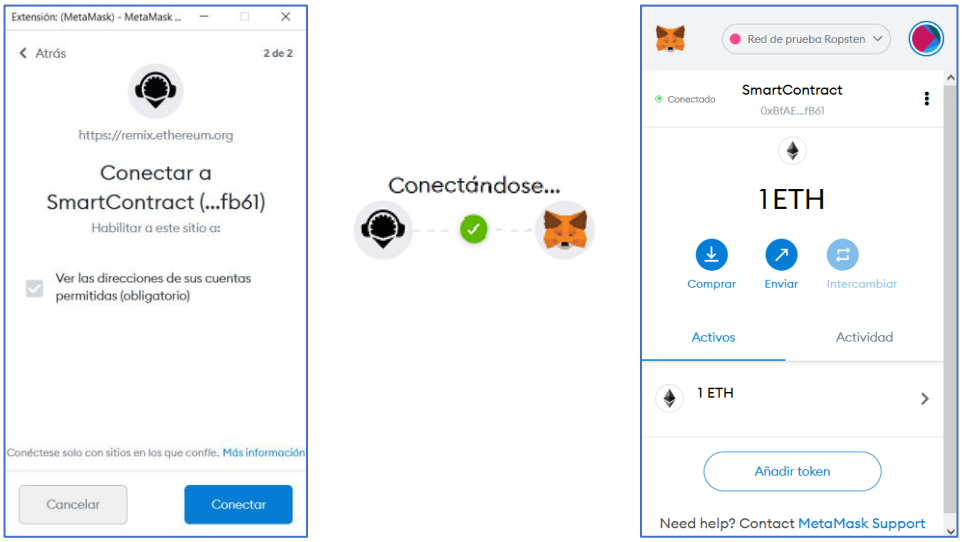
```
function get_log(uint _company, uint year, uint month, uint day) view public returns
(string memory _log) {
  bytes memory tempEmptyStringTest= bytes(comp_date_hash_map[_company][local_date]);
  require(tempEmptyStringTest.length != 0); // must have data the log at this day
  _log = comp_date_hash_map[_company][local_date];
  return _log;
  }
```

To see the complete file, please see the Annex D: Smart contract - timinglog.sol

3) Upload Solidity code. Visit Remix [33] and create a new file. Click in the option you will open the smart contract: New File, Publish to GitHub or Load Local File.

4) Click on " Solidity compiler" icon, click on "compile timinglog.sol" button. If compile is successful, it shows a button for "compilation details" and a green tick on solidity button. Store ABI Code, click on ABI and store in safe place

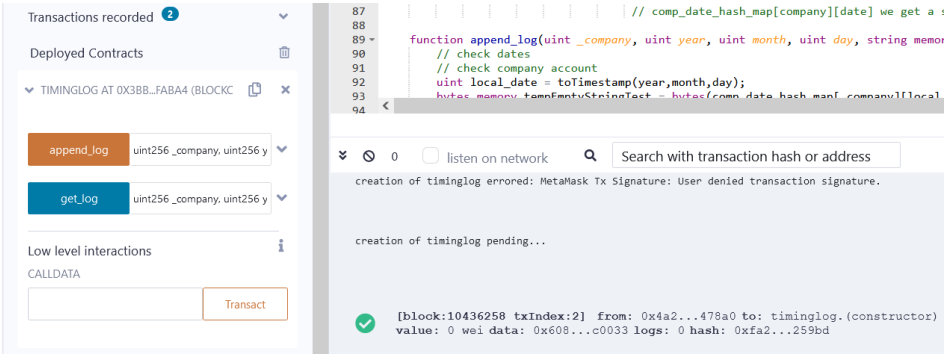
5) Deploy contract. Click on Ethereum logo, select Injected Web3 environment. MetaMask plugin will be ask for password. Select which account will be connected to Remix, and what network is used (Ropsten).



6) Click on "Deploy" button at Remix page. MetaMask will ask for Gas fee (1GWEI) and Gas Limit (613508), and show the price, in ETH. Click on "Confirm" button. If there is not enough gas, the contract will not be deployed, but the computation

cost will be charged to the account. Wait some seconds, as the process need time, until see the confirmation message

7) Once the blockchain is created, Remix will notify the transaction was done. To see details of transaction, see debug panel and click on hash. Also save contract address.



Contract: 0x3bb37250390a96f6dc99fedd6386359c03cfaba4

Transactions details available at:
https://ropsten.etherscan.io/tx/0xfa2da61a1918ecdfdc22e78411cd92aca00e44a75b1ef4519e802c7e4c2259bd

For a detailed explanation, see Annex E: How to deploy a Smart contract.

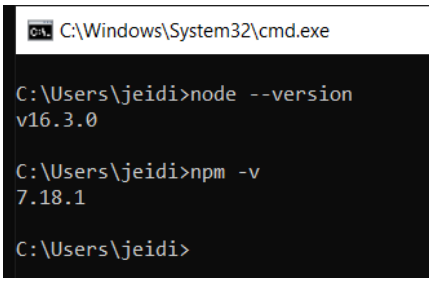
3.5.2. Decentralized application

Node.js lets developers to write command line tools and to produce dynamic web page content before the page is sent to the user's web browser. Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language. Node has its own pre-installed package management, known as Node Package Manager (npm).

To create DApp, it was followed the instructions on MetaMask tutorial stored in: https://github.com/BboyAkers/simple-dapp-tutorial that give a basic template for anyone that want to develop a decentralized application. Steps to configure:

1) Download Node.js program from web [35]. Click Run, click Next to follow the wizard, finally click on Finish. Verified installation. Open cmd command prompt and write the command:

```
> node --version
> npm -v
```



2) Clone GitHub repository [41]. For this project it is needed only the "start" folder".

3) Go to start folder. Rename folder to "DAppTimeLog". Open this folder and check the following files are available:

|-- index.html
|-- contract.js
|-- metamask.css
|-- package.json

The files index.html configure the web design. Contract.js define the actions. Metamask.ccs define style and packaje.json define dependencies.

HTML file structure

The *index.html* file is divided by sections. The first step is to design the layout to connect it to smart contract, DApp need three sections: Basic (Connect to Ethereum), store attendance report (upload daily logs, that means to add new block in blockchain) and read attendance report (read daily logs, that means get IPFS file)

The web3 page can interact with the user with actions button, text input and date selection as specified in the following table. It is important to maintain the same id name inside the *index.html* file and inside the *contract.js* file, otherwise, the program can stop.

	action	type	id
button	Connect to MetaMask	listener on click	id="connectButton"
output	show text	show text	id="accounts"
input	CIF company that uploads	text	id="company_cif_write

	action	type	id
input	Date of upload logs	date	id= "date_write"
input	Select file	file	id="fileToUpload"
button	Add daily log to blockchain	listener on click	id="append_log"
output	IPFS Url result	show text	id= "url_result"
input	CIF company required logs	text	id="company_cif_read
input	Date of required logs	date	id= "date_read"
button	Get daily log from blockchain	listener on click	id="get_log"
output	Link to Daily log file	listener on click	id= "contenedor"

Table. IV. DApp variables name and function

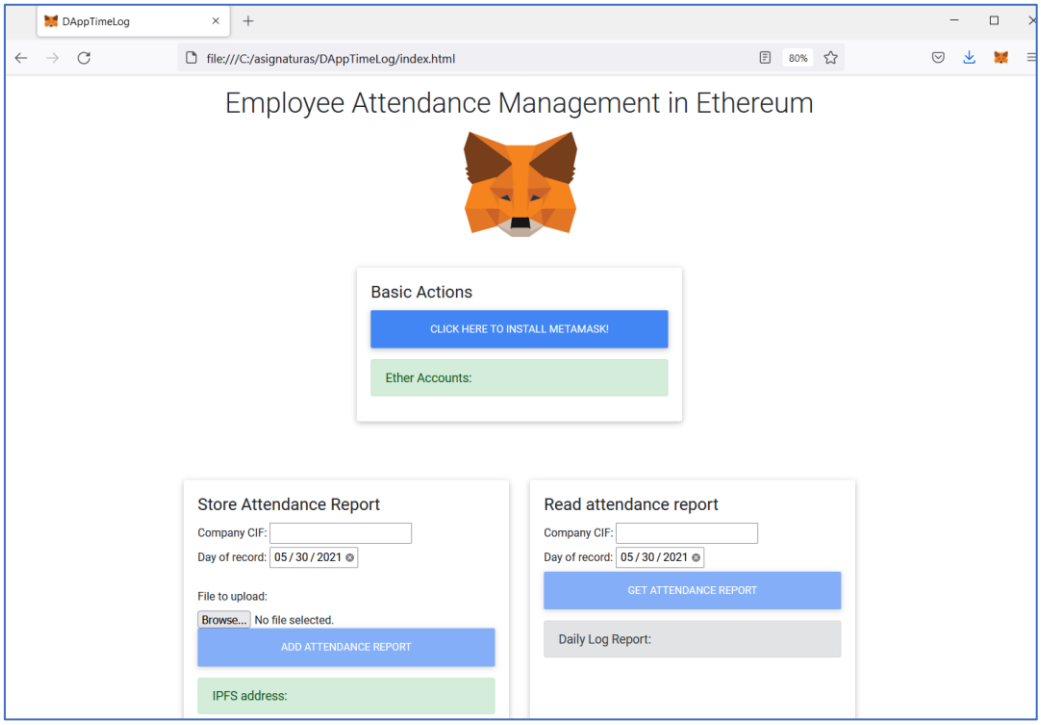


Figure 14. DApp "Employee Attendance Management" layout

1) Update header, update the name of the web page to " DAppTimeLog " and check there is a reference to use "metamask.css"

```
    <title>DAppTimeLog</title>
    <link href="metamask.css" rel="stylesheet">
```

2) Inside body, define first section as " Basic Actions": connect to MetaMask.

```
    <!-- Part 1 Setting up Basic Actions and Status-->
           <h4 class="card-title">  Basic Actions  </h4>
           <button class="btn btn "
             id="connectButton" >  Connect to MetaMask   </button>
```

3) Define second section as " Store Attendance Report"

```
    <!-- Append Log => write attendance registries-->
         <h4 class="card-title"> Store Attendance Report </h4>
          <label>Company CIF: </label>
          <input type="text" id="company_cif"> </input><br>
          <input type="date" id=" date_write" value="2021-05-30" </input><br>
          <label>  <br>File to upload: <br> </label><br>
          <input type="file" id="fileToUpload"></input><br>
          <button class="btn btn " id="addEvent">
             Add Attendance Report  </button>
```

4) Define third section as " Read Attendance Report"

```
  <!-- Get log => Reading attendance registries -->
      <h4 class="card-title"> Read Attendance Report </h4>
       <label>Company CIF: </label>
       <input type="text" id="company_cif">  </input><br>
       <input type="date" id="date_write" value="2021-05-30" </input><br>

       <button class="btn btn- " id="get_log">
          Get Attendance Report  </button>
```

5) Add external scripts to html file. A script is a small piece of program that can add interactivity to website. The DApp example contains two scripts: "MetaMask-onboarding.js" used to check if MetaMask is installed, and "contract.js" used to configure all functions declared in smart contract.

```
<script src=node_modules/@metamask/onboarding/dist/metamask-onboarding.
bundle.js  defer></script>
<script src="contract.js" defer></script>
```

6) Add IPFS script, as a technology to store data in decentralized servers. The JS-IPFS HTTP client [36] is a smaller library implements the IPFS Core API. To understand the IPFS implementation in JavaScript please see Alberto Lassa tutorial in YouTube [37]

```
 <script src="https://cdn.jsdelivr.net/npm/ipfs-http-client/dist/index.min.js">
</script>
```

7) Add Ethereum script implementation [38] that creates JavaScript objects

```
<script src="https://cdn.ethers.io/lib/ethers-5.0.umd.min.js"
type="text/JavaScript">   </script>
```

To see the complete config file please see Annex F: DApp - index.html

CSS file structure

The updated version on E2E Test DApp showed an image with a fox.

```
#mm-logo {
  width: 20%;
  justify-content: center;
}
```

To see the complete config file please see Annex G: DApp - metamask.css

JS file structure

DApp MetaMask tutorial give a very basic *contract.js* file, based mainly on a contract for digital coins. In "*contract.js*" file are defined the interaction with Smart Contract.

1) Define all variables, buttons and text defined on DApp web page layout.

```
const forwarderOrigin = 'http://localhost:9010';
const onboarding = new MetamaskOnboarding({ forwarderOrigin });
const date_write              = document.getElementById('date_write');
const addReportButton         = document.getElementById(append_log);
const date_read               = document.getElementById('date_read');
const getReportButton         = document.getElementById(get_log);
```

2) Save the Contract Account Address and ABI code, obtained during the Smart Contract deployment

```
const ContractAccountAddress = '0x3Bb37250390a96f6dc99fEDD6386359C03CFABA4';
const ABI = [ . . . ]
```

3) Since write on blockchain need to pay to miners for computation work, check if the browser has MetaMask extension installed. Connect MetaMask account. The function "*onClickConnect*" is asynchronous, and uses "*await*" instruction to indicate in cannot continue until the the user select Ethereum account in MetaMask.

```
const initialize = () => {
  //Check if the browser has the MetaMask extension installed
  const isMetaMaskInstalled = () => {
    const { ethereum } = window;
    return Boolean(ethereum && ethereum.isMetaMask);
  };
```

```
    //Show new MetaMask windows to select account to be connected
  const onClickConnect = async () => {
    try {
      await ethereum.request({ method: 'eth_requestAccounts' });
      onboardButton.disabled = true;
    }
  };
    //Check if MetaMask is installed
    const MetamaskClientCheck = async () => {
      if (!isMetaMaskInstalled()) {
        onboardButton.innerText = 'Click here to install MetaMask!';
        onboardButton.onclick = onClickInstall;
```

4) Convert the structure and content of the HTML document into an object model that can be used by programs, using Document Object Model (DOM). First instantiated Ethers provider and define signer as authorized to do transactions

```
window.addEventListener('DOMContentLoaded', initialize);

const provider = new ethers.providers.Web3Provider(window.ethereum);
const signer = provider.getSigner();
```

5) Interact with smart contract. The function "*myContract*" is defined a contract function of ABI and provider. The method "*connect" r*eturns a new instance of the Contract, but connected to providerOrSigner. By passing in a Provider, this will return a downgraded Contract which only has read-only access (i.e. constant calls). By passing in a Signer. this will return a Contract which will act on behalf of that signer. Now, the object is ready to deploy transactions.

```
const myContract = new ethers.Contract(ContractAccountAddress, ABI, provider);
const myContractWithSignature = myContract.connect(signer);
```

6) Once the user upload the file to web browser,for example, the daily Excel file, the program uses the element with id=*fileToUpload* to read metadata information and store it in a new variable named "*fileData*". fileData is an object with file information like filename, size, type of file, date of creation, etc.

```
var fileData = '';
const fileSelector = document.getElementById('fileToUpload');
fileSelector.addEventListener('change', (evento) => {
    fileData = evento.target.files[0];
});
```

7) Declare IPFS server as a IPFS server node located in INFURA [39],. Check that *ipfs-http-client* is also defined in the script of *index.html* file.

```
const ipfs = window.IpfsHttpClient.create(
  { host: 'ipfs.infura.io', port: '5001', protocol: 'https' });
```

8) Add the file to IPFS server (function: ipfs.add). It is created an asynchronous function followed by "*await*" instruction. The auxiliar variable will store the hash code used to store the file in the IPFS.

```
appendLogButton.onclick = async () => {
 try {
   let company = company_cif_write.value;
   let date_write = date_write.value;
   let result = await ipfs.add(fileData);
   let _url = "http://ipfs.io/ipfs/" + result.path;
```

9) Add a new block in blockchain according to smart contract. Use the variables declared: "*company_cif_write*", "*date_write*" and "*_url*".

```
// Upload the event to Ethereum
  console.log(await myContractWithSignature. append_log(cif, date_write, _url));
```

10) Get attendance report from IPFS (function: read blockchain). This function will be launched when the user clicks on "*Get Attendance Report*" button. It will appear an icon, and as the user click on it, it will be redirected to the corresponding IPFS stored file.

```
getLogButton.onclick = async () => {
 try {
   let company = company_cif_read.value;
   let date_read = date_read.value;
   let result = await myContract.get_log(company, date_read);
   //It is show a hyperlink to IPFS address
   document.getElementById("contenedor").innerHTML = '<table>'
       + '<a href="' + result + '">' + result + '</a>'
       + '</table>';
```

11) Execute and run all programs. Open cdm terminal. To install all dependencies needed for the project, run "npm install" command. Once the installation is finished, it will be created a new folder named "*node_modules*" with several files inside. Also install the package to interact with Smart contract and IPFS

```
> npm install
> npm install --save ethers
> npm install ipfs-core
```

To see complete code please see Annex H: DApp - contract.js . To check the procedure to fix errors, please see Annex I: DApp - debugging code in VSC.
https://github.com/JeidiPadron/TFM_Attendance_System_with_Blockchain

4 Deployment and Test Results

4.1. Connect IoT device to network

The first test is to step, it is to power the IoT device. For this project, it can be used USB cable and connected to laptop, AC power, or used a power bank. Once the system is powered, the OLED screen shows:

- Adafruit logo
- The device found a wireless network, Wi-Fi identification (Vodafone E3)
- connected successful to Wi-Fi

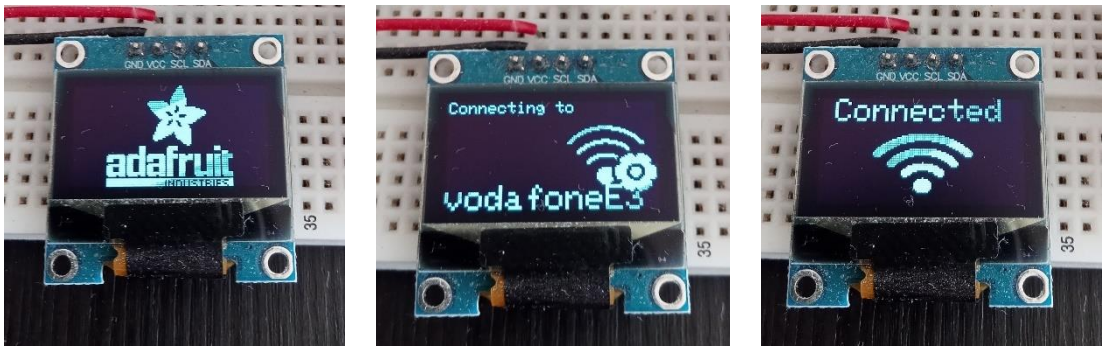


Figure 15. Turn on IoT Device and connect to network

The OLED will show a fingerprint image, indicating the system is working and waiting for a fingerprint. In case of error, the OLED will show a fingerprint with lock, indicating is not working. The main reason is because there was wrong pin connection.

Figure 16. System indication (OK, Not OK)

4.2. Register employee data

The IoT Edge is the main interface between the system administrator in the company and the IoT device in charge of capture the fingerprint of all employees. The fingerprint sensor is capable to store finger ID and fingerprint image. Thanks to IoT Edge, this information can be enriched with: Name, Social Security number, department, and date of enroll. To fill the employee data, it is needed to initialize the Edge service. It is very simple, just execute XAMPP program, start services Apache and MySQL. Then, open any browser, like Firefox, Chrome, or Explorer at http://127.0.0.1/iotedge/index.php. Login credentials are:

E-mail: admin@gmail.com
password: 123

To add new employee data, first check the working mode for IoT Device. The Attendance device can work in two modes:

- Enrollment mode: to capture employee fingerprint, and store in the system, only one time, and
- by default, Attendance mode: to capture employee fingerprint every time they enter or leave the company.

Click con "Devices" tab and select the device mode, click on Enrollment.

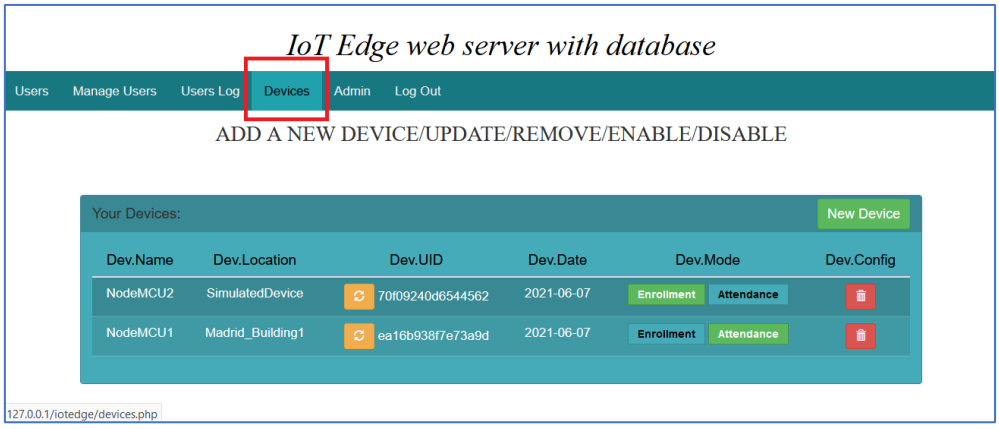
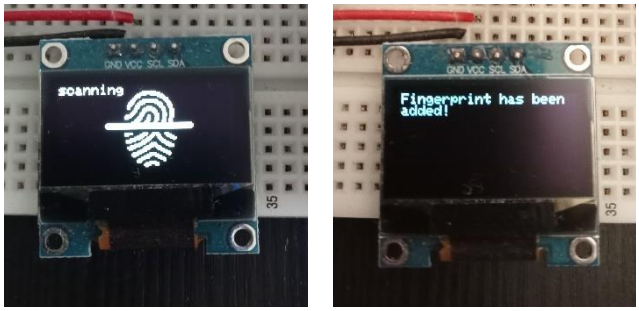


Figure 17. Device working mode

Click "*Manage Users*" tab. Add the location of fingerprint sensor (Building Name). Enter Fingerprint ID. The device will receive the Finger ID, the OLED will show a scanning image and the user can put the finger, two times. The scanner will compare the pictures and confirm image correctly stored. After storing the image, the device will send a

confirmation to Edge node and the status will change to "added" to indicate successful enrollment



Click on Finger ID and fill User Information: name, Social Security Number, and gender. Click "Add". The system will show the message: A new User has been added.

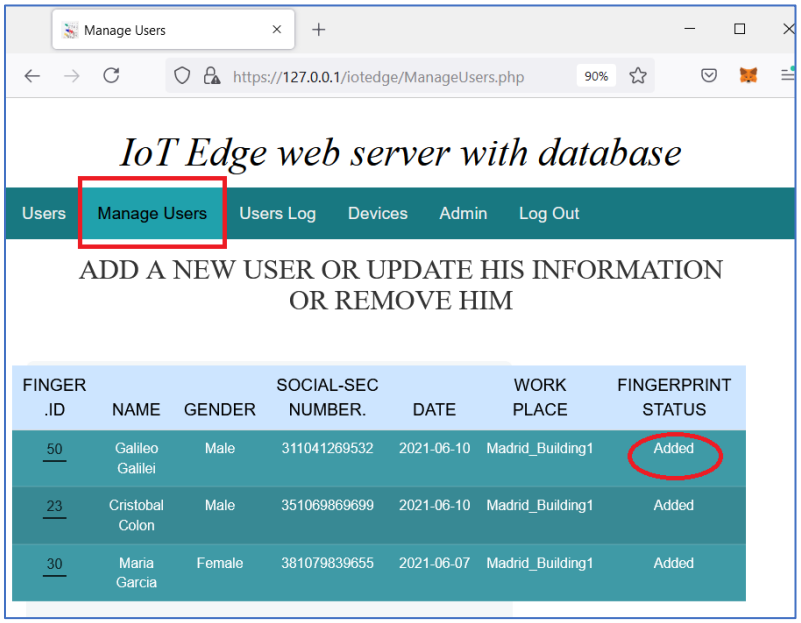


Figure 18. New employee fingerprint added

4.3. Register employee time

Every time an employee starts to work, he must use the fingerprint attendance system to register his entry to the office. The screen will display a message, confirming that it has correctly received the information, through a welcome message: Welcome Bob, for example.

Later, the employee should use the system to indicate that his working day has ended. The worker will receive the confirmation through a farewell message: Goodbye Bob

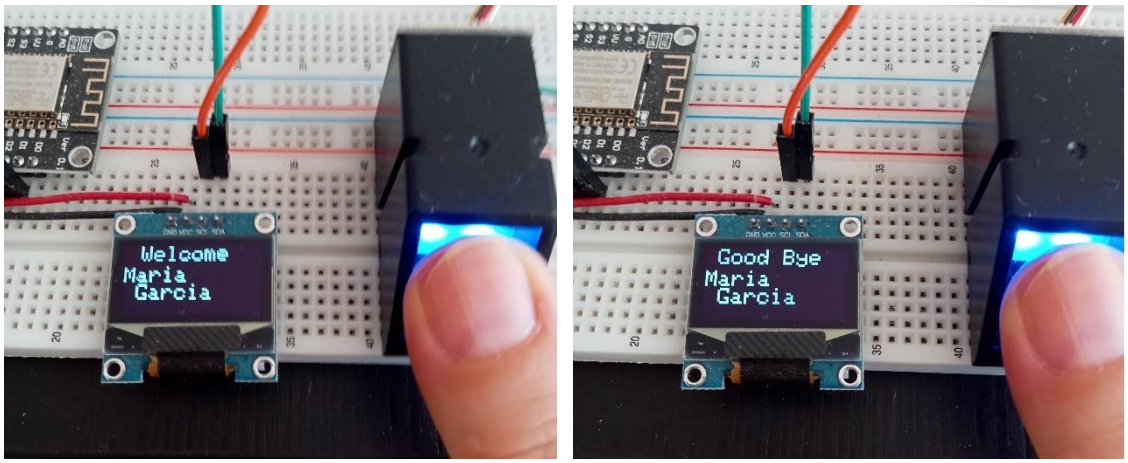
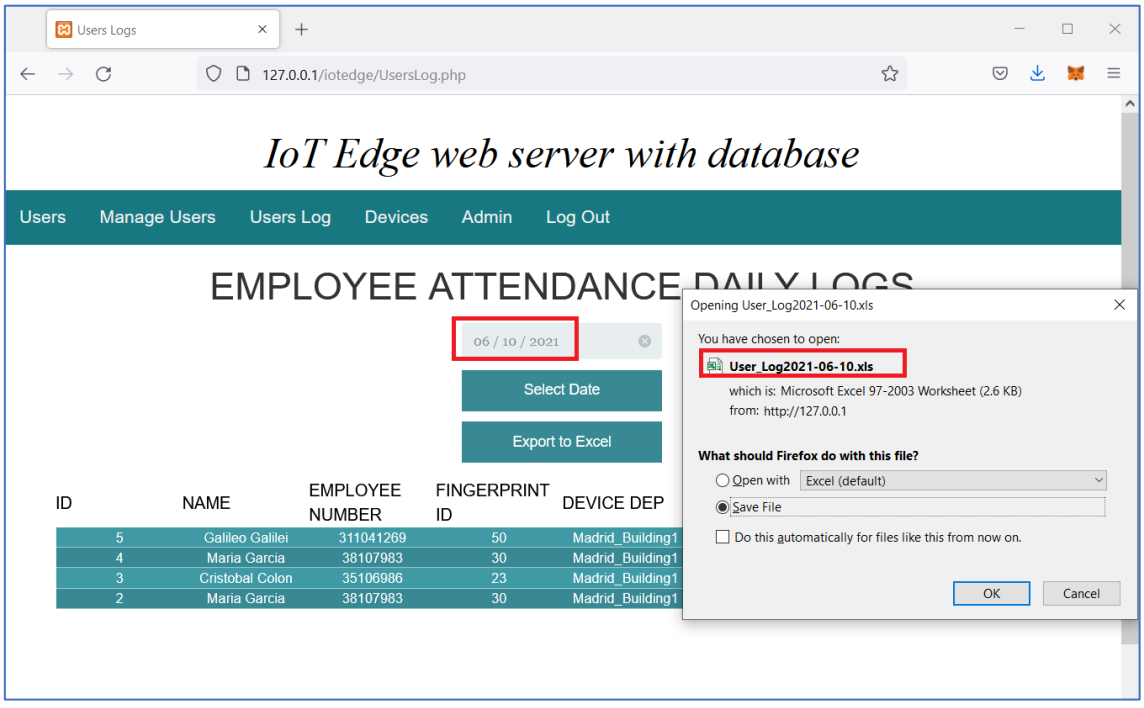


Figure 19. Confirmation message with the name of the employee.

4.4. Obtain Daily Register in Excel

The administrator of the company is the person responsible to download the daily attendance report. "*User Log*" tab shows the attendance report. It can be filtered by employee, by day or by location. Once the system administrator selects a filter, the result can be exported, in Excel format. To obtain the daily report, click "*Users Log*" tab, select the day, click on "*Export to Excel*" button and click on Save the file option.



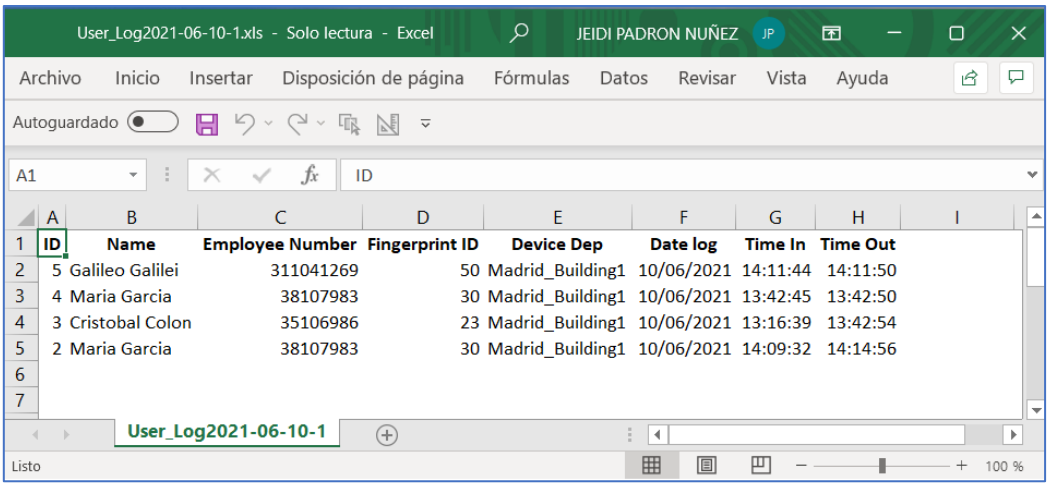
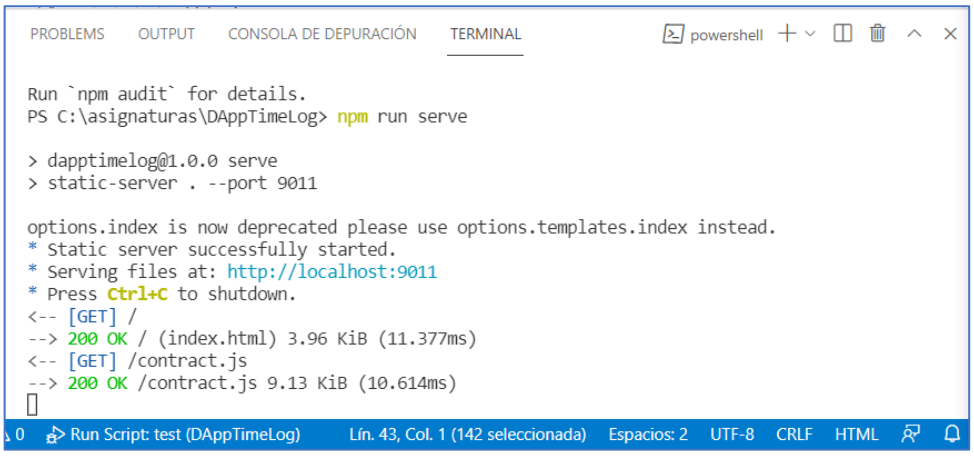


Figure 20. Export daily attendance report to Excel

4.5. Upload Daily log to Blockchain

The Distribute Application DApp allows the company system administrator to store the daily attendance report in a blockchain, according to the instructions given by the Smart contract. To initialize DApp service, open a command console in the same directory where is the code and write the command:

```
C:\asignaturas\DAppTimeLog> npm run serve
```



Open any browser, like Firefox, Chrome or Explorer and write the DApp location at http://localhost:9011. In case the browser does not has the MetaMask extension, DApp will show a message: "Click here to install MetaMask". A new window will be open, to add MetaMask extension in the browser. Click on "Add to Chrome". The DApp will show the message "Onboarding in progress" until the extension is installed. Follow the instructions indicated in the chapter 3 of this document, about MetaMask installation.

Once browser has MetaMask extension installed, the DApp will show a message: "Connect". Login to MetaMask, connect to account. As a result, the DApp will show which account is connected. DApp will use this account to paid miners for add a new block.

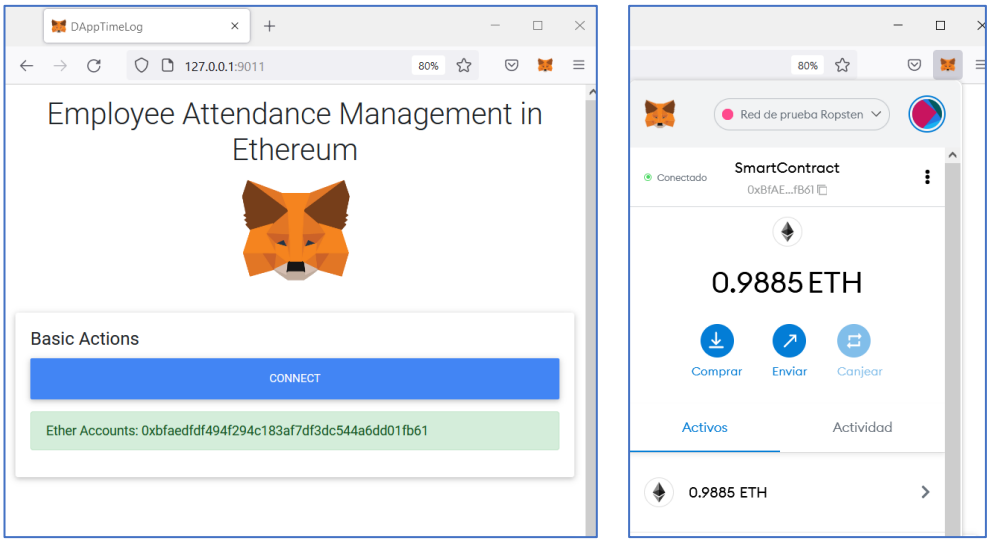


Figure 21. DApp successfully connected to wallet

To send the report, fill de company CIF (only numbers), the date of report, click on "Browse..." to select the file and click on "Add Attendance Report". The application will suggest the cost of transaction (gas limit and gas price that company is willing to pay. Click on "*Confirm*" button, and wait some time, because add a new block takes time. Finally, a message confirm the transaction.

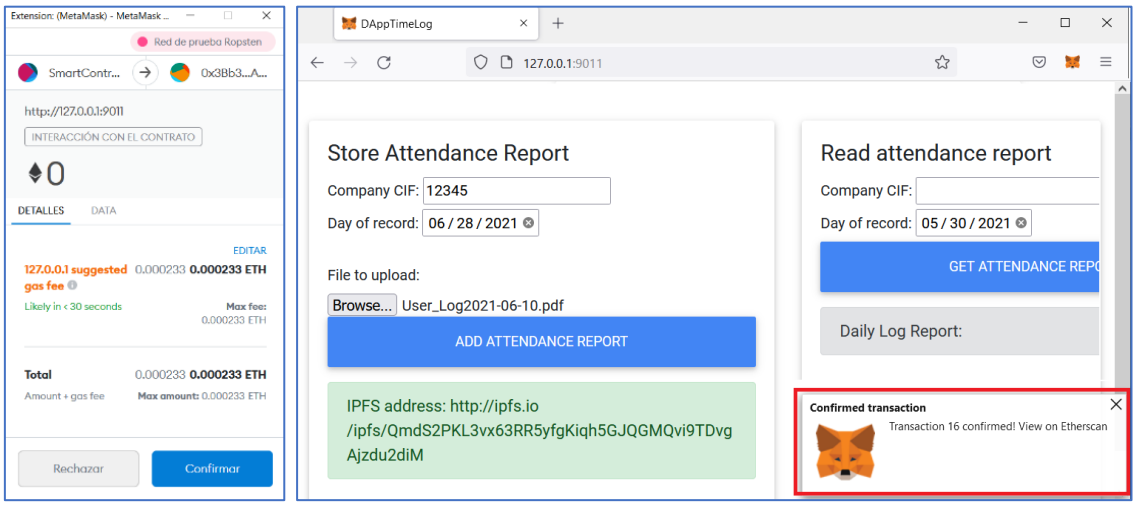
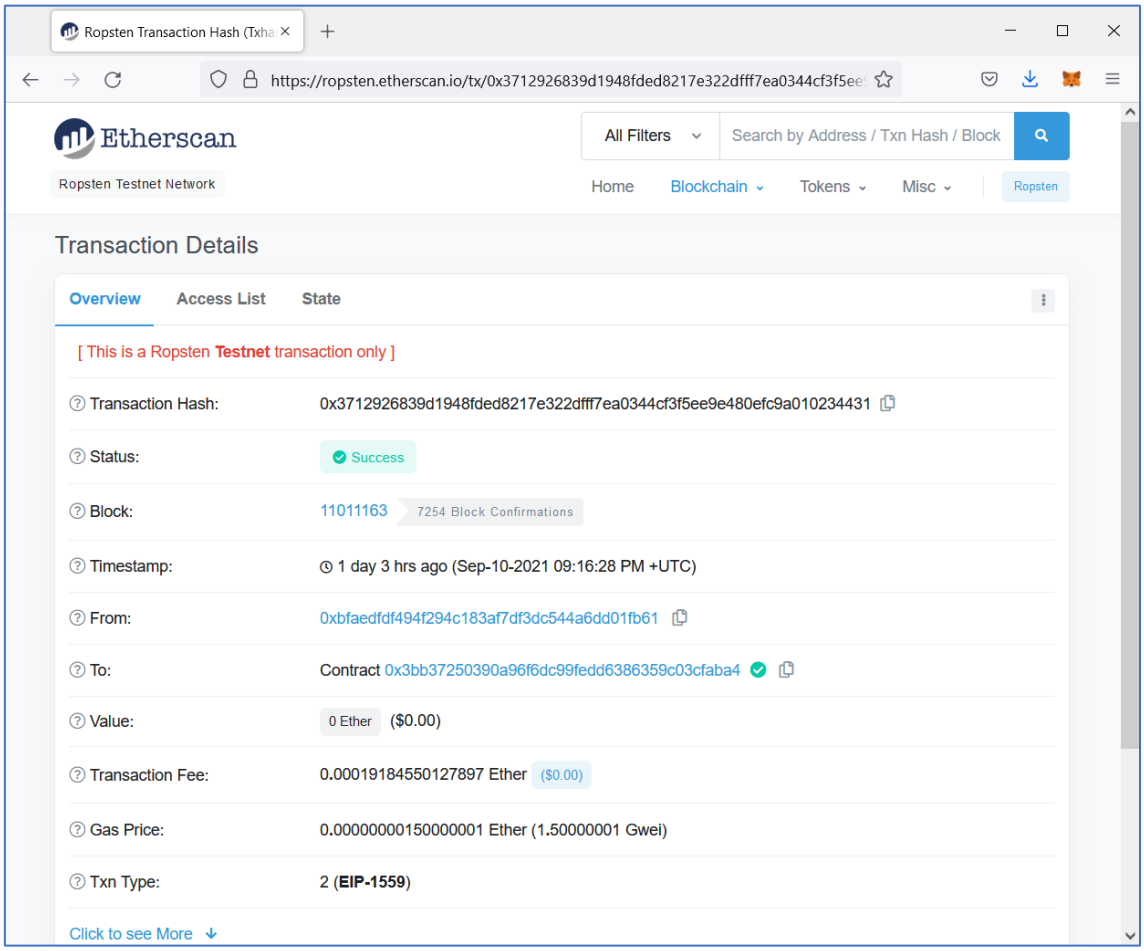


Figure 22. DApp successfully added a new block

To see transaction details, open MetaMask and see transactions on Etherscan page

https://ropsten.etherscan.io/address/0x3bb37250390a96f6dc99fedd6386359c03cfaba4



4.6. Get daily log from blockchain

The process of reading a block inside a blockchain is simpler, faster and it is not necessary to make any payment for the service. To search for the attendance record, the interested people must enter the DApp and fill in the data required in Smart Contract, which are: CIF of the company and date of the report. With this information, the smart contract returns a text string that contains the file location, which is in another decentralized network, such as IPFS. When the user clicks on the link, the file is searched in many peer-to-peer nodes. Finally, appear a window indicating if you want to view the report, or if you want to store it in a particular place.

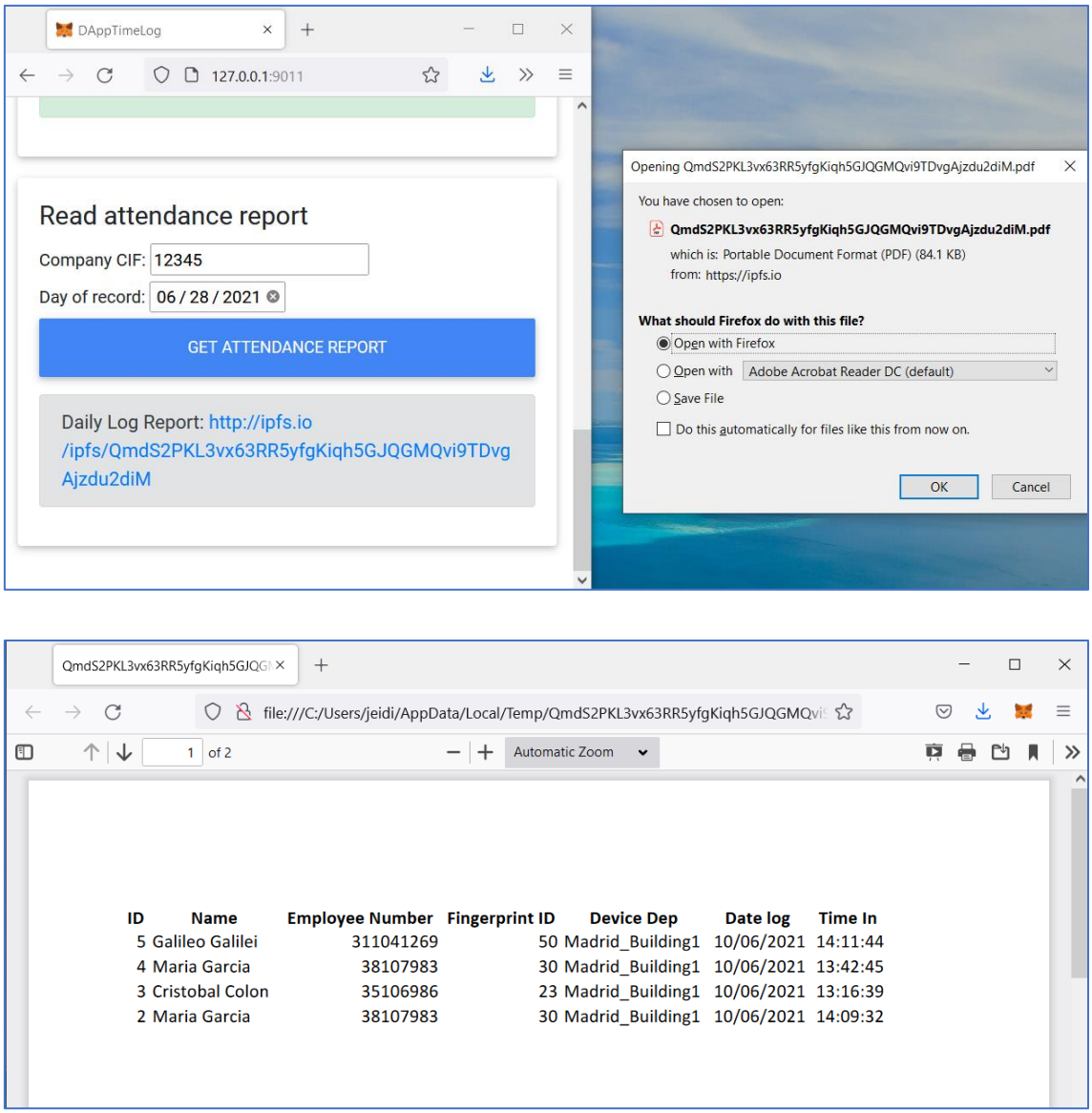


Figure 23. Daily log recovered from IPFS blockchain

5 Budget

5.1. Project Budget

The following are the components, material and engineering development cost required to make IoT Based Biometric Fingerprint Attendance System. All the components were be purchased from Amazon Spain. The purchase links are given below.

N	Component	Description	Qty	Price	Amazon site
1	NodeMCU	ESP8266-12E Board	1	7,79€	https://amzn.to/2Q7b5TG
2	Fingerprint Sensor	Dollatek Module	1	16.89 €	https://amzn.to/3lfhiJc
3	OLED Display	0.96" SSD1306 I2C OLED Display	1	6,79€	https://amzn.to/3d19h7C
4	Connecting Wires	Jumper Wires	10	4,99€	https://amzn.to/3nqvEt2
5	Breadboard	Proto Board	2	7,00 €	https://amzn.to/3aoWZnX
6	Arduino	Donation	1	3€	
7	Web server	Laptop Lenovo	1	600 €	
8	Developer	Engineer	1	4.000 €	Payment for 9 week = 4K
		Total		4.646 €	

Table. V. Project budget.

5.2. Time schedule.

Tasks The prototype will be carried out in different phases:

- Select and buy IoT platform, IDE development: Arduino IoT board, fingerprint sensor and LED screen for real data generation. Arduino already has a library for fingerprint sensors.
- Configure and test the Arduino board, check, and test Wi-Fi connection, configure LED screen
- Configure and test the fingerprint sensor and connect it to Arduino

- Develop a traditional web page (client-server topology) to keeps a data history of enter / exit time of employees in a company inside a traditional database. Export daily report
- Develop IPFS server for daily data storing, write a smart contact using solidity language. The upload of daily record is done manually by authorized people
- Configure a Distributed app (DApp) in JavaScript [6] to take the daily record and stored it using blockchain network
- Generate Smart contract with the daily information, using DApp, Ethereum blockchain and IPFS peer-to-peer technologies.
- Test end to end system and make adjust (if any)
- Write final report and conclusion
- Prepare presentation

Time-schedule: the work uses 40 hours a week, total eight weeks (320 hours)

	W1	W2	W3	W4	W5	W6	W7	W8	W9
	May 17-21	May 24-28	Jun 1-4	Jun 7-11	Jun 14-18	Jun 21-25	Jun 28-31	Jul 5-9	Jul 12-16
Task 1: buy material, configure Arduino, Wi-Fi									
Task 2: configure Led screen and Wi-Fi									
Task 3: configure fingerprint sensor									
Task 4: develop web page (client server)									
Task 5: develop IPFS server, write smart contract									
Task 6: develop DApps with JavaScript									
Task 7: generate smart contract, Ethereum									
Task 8: testing and correct (if any)									
Task 9: write final report									
Task 10: prepare presentation									

6 Conclusions

6.1. General conclusions.

Biometric work attendance management and logging with a blockchain system is used to know the exact number of days for which the employee has worked for. As the main objective of this thesis was to develop a system capable to generate a daily attendance report, and store it in a secure place, keeping the data immutable, I can say with great satisfaction that the proposed objectives have been fulfilled.

- Data store secure and immutable: once the daily report is stored in a blockchain, nobody can modify at any way, not even the head of the company can manipulate the data. If one block is changed by someone, then the blockchain will alerting the network to the fact that one block is fake.
- Reduce company cost: the attendance system helps to control labor costs by reducing over-payments, and eliminates transcription error, interpretation error and intentional error.
- User Friendly: the user interface is friendly and intuitive. The company responsible can add the employee information, such as name, working location, employee number and fingerprint easily.
- Trust Daily Reports: the daily report is generated, according to the selected day, in excel format, according to the employee data. As a fingerprint cannot be cloned, this daily report gives the possibility to know employee assistance.
- IoT technology adoption: The cost for the device itself is affordable, with low power consumption and compatible with Arduino.

Curiosity, the great advantage of the system is a great disadvantage: the use of blockchain. It is a complex technology, difficult to understand and work different from the traditional ways, which makes people have a fear of adopting decentralized networks. Most people preferred a centralized cloud computing from a trusted vendor, like Amazon or Microsoft, instead of giving the data to untrusted peer-to-peer node. The key concept is to understand the *consensus algorithm* that allows nodes to trust each other, a given piece of data is valid and that it has been synchronized with all other nodes.

6.2. Future works

This project can be enhanced in many ways. I take notes during the project development, thinking in all aspect learned in the Master studies.

- Power Supply: add small battery instead use USB port.
- In case the Wi-Fi communication is broken, the device can use MQTT protocol, connected to MQTT server, with high Quality of Service, to store the data and send later, when communication service is restored.
- Develop a simulated device, using node-red or another technology. This simulation can help a lot with testing procedure and make new scenarios.
- Enhance IoT device security. The NodeMCU documentation show there is implemented security access using hash. I think is a good opportunity to test it SSL support, TLS, SHA-1, and ciphers in IoT device.
- During the testing phase, I write a wrong date, and I did not receive any alert. If the block is not found in the blockchain, the system must give a message, like: blockchain not created, information not available, check the company's CIF and the date of the report
- Another issue is for duplicated report. In case the company upload two documents the same day, the system can only handle the last hash generated. Maybe an alarm: "there is a file for the date you want to upload"
- For remote working employee, it can be used any biometric sensor, like fingerprint of face recognition at mobile phones.
- Create a new project, for control access, adding an actuator that simulate turnstile using a servomotor.

7 References

7.1. References

[1] Royal Decret- Law 8/2019 on urgent measures of social protection and the fight against job insecurity in the workday (online) https://www.boe.es/boe/dias/2019/03/12/pdfs/BOE-A-2019-3481.pdf

[2] Ministerio del Trabajo y Economia Social (mites). Frequented asked questions about working day registry. (online) https://www.mites.gob.es/ficheros/ministerio/GuiaRegistroJornada.pdf

[3] Biometric attendance system - International Research Journal of Engineering and Technology (IRJET) (online) https://www.irjet.net/archives/V6/i4/IRJET-V6I4866.pdf

[4] Benet, Juan: "IPFS - Content Addressed, Versioned, P2P File System" (arxiv.org) https://arxiv.org/pdf/1407.3561.pdf

[5] Andreas M. Antonopoulos, Gavin Wood (2019) Ethereumbook: Mastering Ethereum, (online) https://github.com/ethereumbook/ethereumbook/blob/develop/book.asciidoc

[6] Ethereum.org - Anatomy of smart contracts | (online) https://ethereum.org/en/developers/docs/smart-contracts/anatomy/

[7] PaymentsSource - Smartphone design is the next challenge for biometrics https://www.paymentssource.com/payments/opinion/smartphone-design-is-the-next-challenge-for-biometrics

[8] Adafruit Optical Fingerprint Sensor (online) https://learn.adafruit.com/adafruit-optical-fingerprint-sensor

[9] Adafruit Learning System - optical fingerprint sensor (online) https://cdn-learn.adafruit.com/downloads/pdf/adafruit-optical-fingerprint-sensor.pdf

[10] JM-101 Optical Fingerprint Module User manual (online) https://3v3.com.ua/data/files/JM-101-OPTICAL-FINGERPRINT-MODULE-USER-MANUAL-V1.8A.PDF

[11] AZ Delivery - OLED display 1.3 inch User Manual (online) https://www.az-delivery.de/a/downloads/-/e68ccc911afe63a6/403e1e3183023714

[12] AZ Delivery - NodeMCU Lua Lolin V3 Module ESP8266 ESP-12F WIFI Development Board, User Manual (online) https://www.az-delivery.de/a/downloads/-/52395f57d55a704f/3abbabde78085841

[13] Flaticon.com - Free Vector Icons and Stickers - PNG, SVG, EPS, PSD and CSS (online) https://www.flaticon.es/autores/prosymbols

[14] ethereum.org - Introduction to decentralized application (online) https://ethereum.org/en/developers/docs/dapps/

[15] Merunas Grincalaitis "Ultimate Guide to Convert a Web App To a Decentralized App Dapp". Ethereum Developers in Medium blog page. March 2018 (online) https://medium.com/ethereum-developers/ultimate-guide-to-convert-a-web-app-to-a-decentralized-app-dapp-f6112a079509

[16] CoinMarketCap - Ethereum price today, ETH live marketcap, chart, and info (online) https://coinmarketcap.com/currencies/ethereum/

[17] Quora - What is the blocktime of Ethereum? (online) https://www.quora.com/What-is-the-blocktime-of-Ethereum

[18] Nick Szabo blog page (online) http://unenumerated.blogspot.com/

[19] DevTeam.Space - How to Deploy Contract Ethereum? (online) https://www.devteam.space/blog/how-to-deploy-smart-contract-on-ethereum/

[20] MetaMask Create A Simple Dapp (online) https://docs.metamask.io/guide/create-dapp.html

[21] GeeksforGeeks: Difference between HTTP and IPFS (2019) (online) https://www.geeksforgeeks.org/difference-between-http-and-ipfs/

[22] ichi.pro - Cree su primer Dapp con Web3.js (online) https://ichi.pro/es/cree-su-primer-dapp-con-web3-js-169818826304097

[23] Arduino - Software Download (online) https://www.arduino.cc/en/software

[24] Nanjing Qinheng Microelectronics - Windows USB controller CH341SER (online) http://www.wch.cn/download/CH341SER_ZIP.html

[25] GitHub - nodemcu flasher Win64 Release at master (online) https://github.com/nodemcu/nodemcu-flasher/tree/master/Win64/Release

[26] Adafruit Learning System: Monochrome OLED Breakouts (online) https://learn.adafruit.com/monochrome-oled-breakouts/arduino-library-and-examples

[27] Adafruit Learning System: Optical Fingerprint Sensor - Downloads https://learn.adafruit.com/adafruit-optical-fingerprint-sensor/downloads

[28] Apachefriends.org - XAMPP Installers and Downloads (online) https://www.apachefriends.org/index.html

[29] Install MetaMask for your browser. Download for browser (Chrome, Firefox, Brave, Edge), iOS and Android. (online). https://metamask.io/download.html

[30] Test Ether Faucet (metamask.io) (online). https://faucet.metamask.io/

[31] soliditylang.org - Contracts , Solidity 0.8.8 documentation (online) https://docs.soliditylang.org/en/latest/contracts.html

[32] Vyper — Vyper documentation https://vyper.readthedocs.io/en/latest/

[33] Remix - Ethereum IDE (online) https://remix.ethereum.org

[34] Visual Studio Code - Code Editing. Redefined. Download for Windows, macOS, Linux (online) https://code.visualstudio.com/

[35] Node.js JavaScript runtime. Download for Windows (x64) (online) https://nodejs.org/en/

[36] IPFS Documentation - JS-IPFS (online) https://docs.ipfs.io/reference/js/api/

[37] Alberto Lasa - Youtube channel - IPFS - Subir y Descargar Archivos, Regular File API (online) https://www.youtube.com/watch?v=J8YGaKD0wd8

[38] The Ethers Project (npmjs.com) ethers - npm (online) https://www.npmjs.com/package/ethers

[39] Infura IPFS API - Scalable and distributed IPFS storage infrastructure for your application (online) https://infura.io/product/ipfs

7.2. Repositories

[40] GitHub InfinityWorldHI - Biometric-attendance-system-V1.0 (online) https://github.com/InfinityWorldHI/Biometric-attendance-system-V1.0

[41] GitHub - BboyAkers/simple-dapp-tutorial: A tutorial based on the MetaMask E2E Test Dapp(decentralized app) (online) https://github.com/BboyAkers/simple-dapp-tutorial

[42] GitHub JeidiPadron - TFM Attendance System with Blockchain (online) https://github.com/JeidiPadron/TFM_Attendance_System_with_Blockchain

8 ANNEX

Annex A - Glossary

Term	Definition
Blockchain	A decentralized public ledger that records all transactions within a given network.
CIF	Código de Identificación Fiscal (Spain)
Consensus algorithm	A consensus algorithm allows nodes on the network to trust that a given piece of data is valid and that it has been synchronized with all other nodes.
CSS	Cascading Style Sheets
DAGs	Directed Acyclic Graphs.
DApp	Decentralized Application
DLT	Distributed Ledger Technology
DHT	Distributed Hash Table
DOM	Document Object Model
EOAs	Externally Owned Accounts
ERC	Ethereum Requests for Comments
ETH	Ether (virtual coin, cryptocurrency)
EVM	Ethereum Virtual Machine
GPIO	general-purpose input/output
HTML	HyperText Markup Language
HTTP	HiperText Transfer Protocol ()
Infura	It is a set of tools to create an application that connects to the blockchain. It interacts with the Ethereum blockchain
IDE	Integrated Development Environment
IP	Internet Protocol
IPNS	InterPlanetary Naming System
IPFS	InterPlanetary File System
MCU	Micro Controller Unit
npm	Node Package Manager
OLED	Organic Light Emitting Diode
P2P	Peer-to-peer
PHP	Hypertext Preprocessor
PoC	Proof of Concept
rETH	Ropsten ETH (test ETH, no value)

Term	Definition
RFID	Radio Frequency IDentification
SoC	System on Chip
SQL	Structured Query Language
UMD	Universal Module Definition
VSC	Visual Studio Code
wei	lowest denomination for ETH. (1 ETH = $1x10^{18}$ wei).)
XAMPP	Cross-platform, Apache, MySQL, PHP and Perl

Annex B - IoT Device Arduino Code

```
  MSc in Internet of Things
  Master Thesis Project
  Biometric work attendance management and logging with a blockchain system
  Author: Jeidi Padron Nuñez
  Date: July 2021
  version 1.0

  ----------------------------------------------------------------------------
  This code was based on code created by Electronics Tech youtube channel for
  the Biometric attendance project with NodeMCU, available at
  https://github.com/InfinityWorldHI/Biometric-attendance-system-V1.0

  Test code provided by vendor
  Adafruit    - Fingerprint model JM-101B
  AZ delivery - OLED I2C 128 x 64 pixel display for Arduino and Raspberry Pi
  AZ delivery - NodeMCU V3 Lolin
  https://www.az-delivery.de/en

  ----------------------------------------------------------------------------*/


//////////////////////////////////////////////////
//     LIBRARIES
//////////////////////////////////////////////////

#include <SPI.h>
#include <Wire.h>
#include <WiFiClient.h>
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>
#include <Adafruit_GFX.h>                    //https://github.com/adafruit/Adafruit-
GFX-Library
#include <Adafruit_SSD1306.h>
//https://github.com/adafruit/Adafruit_SSD1306
#include <Adafruit_Fingerprint.h>            //https://github.com/adafruit/Adafruit-
Fingerprint-Sensor-Library
#include <SimpleTimer.h>
//https://github.com/jfturcot/SimpleTimer


//////////////////////////////////////////////////
//     Fingerprint scanner Pins
//////////////////////////////////////////////////
#define Finger_Rx 14                         //D5
```

```
#define Finger_Tx 12                          //D6
SimpleTimer timer;
SoftwareSerial mySerial(Finger_Rx, Finger_Tx);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);


////////////////////////////////////////////////
//     SSD1306 display -  I2C Pins (22 SCL, 21 SDA)
////////////////////////////////////////////////
#define SCREEN_WIDTH 128                      // OLED display width, in pixels
#define SCREEN_HEIGHT 64                      // OLED display height, in pixels
#define OLED_RESET     0                      // Reset pin # (or -1 if sharing
Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);


////////////////////////////////////////////////
//     Declaration of Wi_Fi credentials.
////////////////////////////////////////////////
const char *ssid = "Jeidi_AP";                   //Enter your WiFi values
const char *password = "12345678";


////////////////////////////////////////////////
//     Declaration of IoT Device ID
////////////////////////////////////////////////
const char* device_token  = "6b07332bc7f1c0b9";


////////////////////////////////////////////////
//     Declaration of IoT Edge Web Server and Database.
////////////////////////////////////////////////
String getData, Link;
String URL = "http://192.168.0.22/iotedge/getdata.php"; //computer IP or the server
domain and folder
int FingerID = 0, t1, t2;                          // The Fingerprint ID from the
scanner
bool device_Mode = false;                          // Default Mode Enrollment
bool firstConnect = false;
uint8_t id;
unsigned long previousMillis = 0;

////////////////////////////////////////////////
//     Biometric Icons
////////////////////////////////////////////////
#define Wifi_start_width 54
#define Wifi_start_height 49
const uint8_t PROGMEM Wifi_start_bits[] = {
  0x00,0x00,0x00,0x00,0x00,0x00,0x00
```

```
,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x1f,0xf0,0x00,0x00,0x00
,0x00,0x03,0xff,0xff,0x80,0x00,0x00
,0x00,0x1f,0xf0,0x1f,0xf0,0x00,0x00
,0x00,0x7e,0x00,0x00,0xfc,0x00,0x00
,0x01,0xf0,0x00,0x00,0x1f,0x00,0x00
,0x03,0xc0,0x00,0x00,0x07,0xc0,0x00
,0x0f,0x00,0x00,0x00,0x01,0xe0,0x00
,0x1c,0x00,0x00,0x00,0x00,0x70,0x00
,0x38,0x00,0x07,0xc0,0x00,0x38,0x00
,0x70,0x00,0xff,0xfe,0x00,0x1e,0x00
,0xe0,0x03,0xfc,0x7f,0xc0,0x0e,0x00
,0x00,0x1f,0x80,0x03,0xf0,0x00,0x00
,0x00,0x3c,0x00,0x00,0x78,0x00,0x00
,0x00,0xf0,0x00,0x00,0x1c,0x00,0x00
,0x01,0xe0,0x00,0x00,0x0c,0x00,0x00
,0x03,0x80,0x00,0x00,0x00,0x00,0x00
,0x03,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x3f,0xf8,0x07,0x1e,0x00
,0x00,0x00,0xff,0xfe,0x1f,0xbf,0x80
,0x00,0x03,0xe0,0x04,0x7f,0xff,0xc0
,0x00,0x07,0x80,0x00,0xff,0xff,0xe0
,0x00,0x0e,0x00,0x00,0xff,0xff,0xe0
,0x00,0x0c,0x00,0x00,0x7f,0xff,0xc0
,0x00,0x00,0x00,0x00,0xfe,0x07,0xe0
,0x00,0x00,0x00,0x03,0xf8,0x03,0xf8
,0x00,0x00,0x07,0xe7,0xf9,0xf1,0xfc
,0x00,0x00,0x1f,0xe7,0xf1,0xf9,0xfc
,0x00,0x00,0x1f,0xe7,0xf3,0xf9,0xfc
,0x00,0x00,0x3f,0xe7,0xf3,0xf9,0xfc
,0x00,0x00,0x3f,0xe7,0xf1,0xf1,0xfc
,0x00,0x00,0x3f,0xe3,0xf8,0xe3,0xfc
,0x00,0x00,0x3f,0xf3,0xfc,0x07,0xf8
,0x00,0x00,0x1f,0xf0,0x7f,0x0f,0xc0
,0x00,0x00,0x0f,0xe0,0x7f,0xff,0xe0
,0x00,0x00,0x07,0xc0,0xff,0xff,0xe0
,0x00,0x00,0x00,0x00,0x7f,0xff,0xe0
,0x00,0x00,0x00,0x00,0x3f,0xff,0x80
,0x00,0x00,0x00,0x00,0x1f,0xbf,0x00
,0x00,0x00,0x00,0x00,0x03,0x18,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};
```

```
#define Wifi_connected_width 63
#define Wifi_connected_height 49
const uint8_t PROGMEM Wifi_connected_bits[] = {
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x03,0xff,0xff,0x80,0x00,0x00
,0x00,0x00,0x3f,0xff,0xff,0xf8,0x00,0x00
,0x00,0x01,0xff,0xff,0xff,0xff,0x00,0x00
,0x00,0x0f,0xff,0xff,0xff,0xff,0xe0,0x00
,0x00,0x3f,0xff,0xc0,0x07,0xff,0xf8,0x00
,0x00,0xff,0xf8,0x00,0x00,0x3f,0xfe,0x00
,0x03,0xff,0x80,0x00,0x00,0x03,0xff,0x80
,0x07,0xfe,0x00,0x00,0x00,0x00,0xff,0xc0
,0x1f,0xf8,0x00,0x00,0x00,0x00,0x3f,0xf0
,0x3f,0xe0,0x01,0xff,0xff,0x00,0x0f,0xf8
,0x7f,0x80,0x0f,0xff,0xff,0xe0,0x03,0xfc
,0xff,0x00,0x7f,0xff,0xff,0xfc,0x01,0xfe
,0xfc,0x01,0xff,0xff,0xff,0xff,0x00,0x7e
,0x78,0x07,0xff,0xc0,0x07,0xff,0xc0,0x3c
,0x00,0x0f,0xfc,0x00,0x00,0x7f,0xe0,0x00
,0x00,0x1f,0xf0,0x00,0x00,0x1f,0xf0,0x00
,0x00,0x3f,0xc0,0x00,0x00,0x07,0xf8,0x00
,0x00,0x7f,0x00,0x01,0x00,0x01,0xfc,0x00
,0x00,0x7e,0x00,0x7f,0xfc,0x00,0xfc,0x00
,0x00,0x3c,0x03,0xff,0xff,0x80,0x78,0x00
,0x00,0x00,0x07,0xff,0xff,0xc0,0x00,0x00
,0x00,0x00,0x1f,0xff,0xff,0xf0,0x00,0x00
,0x00,0x00,0x3f,0xf0,0x1f,0xf8,0x00,0x00
,0x00,0x00,0x3f,0x80,0x03,0xf8,0x00,0x00
,0x00,0x00,0x3f,0x00,0x01,0xf8,0x00,0x00
,0x00,0x00,0x1c,0x00,0x00,0x70,0x00,0x00
,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x0f,0xe0,0x00,0x00,0x00
,0x00,0x00,0x00,0x1f,0xf0,0x00,0x00,0x00
,0x00,0x00,0x00,0x3f,0xf8,0x00,0x00,0x00
,0x00,0x00,0x00,0x3f,0xf8,0x00,0x00,0x00
,0x00,0x00,0x00,0x3f,0xf8,0x00,0x00,0x00
,0x00,0x00,0x00,0x3f,0xf8,0x00,0x00,0x00
,0x00,0x00,0x00,0x1f,0xf0,0x00,0x00,0x00
,0x00,0x00,0x00,0x0f,0xe0,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
```

```
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};
#define FinPr_start_width 64
#define FinPr_start_height 64
const uint8_t PROGMEM FinPr_start_bits[] = {
  0x00,0x00,0x00,0x1f,0xe0,0x00,0x00,0x00
,0x00,0x00,0x01,0xff,0xfe,0x00,0x00,0x00
,0x00,0x00,0x03,0xff,0xff,0x80,0x00,0x00
,0x00,0x00,0x0f,0xc0,0x0f,0xe0,0x00,0x00
,0x00,0x00,0x1f,0x00,0x01,0xf8,0x00,0x00
,0x00,0x00,0x3c,0x00,0x00,0x7c,0x00,0x00
,0x00,0x00,0x78,0x00,0x00,0x3e,0x00,0x00
,0x00,0x00,0xf0,0x3f,0xf8,0x0f,0x00,0x00
,0x00,0x01,0xe0,0xff,0xfe,0x07,0x80,0x00
,0x00,0x03,0xc3,0xff,0xff,0x03,0x80,0x00
,0x00,0x03,0x87,0xc0,0x07,0xc3,0xc0,0x00
,0x00,0x07,0x0f,0x00,0x03,0xe1,0xc0,0x00
,0x00,0x0f,0x0e,0x00,0x00,0xe0,0xe0,0x00
,0x00,0x0e,0x1c,0x00,0x00,0xf0,0xe0,0x00
,0x00,0x0c,0x3c,0x1f,0xe0,0x70,0xe0,0x00
,0x00,0x00,0x38,0x3f,0xf0,0x38,0x70,0x00
,0x00,0x00,0x78,0x78,0xf8,0x38,0x70,0x00
,0x00,0x00,0x70,0x70,0x3c,0x18,0x70,0x00
,0x00,0x00,0xe0,0xe0,0x1e,0x1c,0x70,0x00
,0x00,0x03,0xe1,0xe0,0x0e,0x1c,0x70,0x00
,0x00,0x0f,0xc1,0xc3,0x0e,0x1c,0x70,0x00
,0x00,0x3f,0x03,0xc3,0x8e,0x1c,0x70,0x00
,0x00,0x3e,0x03,0x87,0x0e,0x1c,0x70,0x00
,0x00,0x30,0x07,0x07,0x0e,0x18,0xe0,0x00
,0x00,0x00,0x0e,0x0e,0x0e,0x38,0xe0,0x00
,0x00,0x00,0x3e,0x1e,0x1e,0x38,0xe0,0x00
,0x00,0x00,0xf8,0x1c,0x1c,0x38,0xe0,0x00
,0x00,0x03,0xf0,0x38,0x3c,0x38,0xe0,0x00
,0x00,0x3f,0xc0,0xf8,0x78,0x38,0xe0,0x00
,0x00,0x7f,0x01,0xf0,0x70,0x38,0xf0,0x00
,0x00,0x78,0x03,0xe0,0xe0,0x38,0x70,0x00
,0x00,0x00,0x0f,0x81,0xe0,0x38,0x7c,0x00
,0x00,0x00,0x3f,0x03,0xc0,0x38,0x3e,0x00
,0x00,0x00,0xfc,0x0f,0x80,0x38,0x1e,0x00
,0x00,0x07,0xf0,0x1f,0x1c,0x1c,0x04,0x00
,0x00,0x3f,0xc0,0x3e,0x3f,0x1e,0x00,0x00
,0x00,0x7f,0x00,0xf8,0x7f,0x0f,0x00,0x00
,0x00,0x38,0x01,0xf0,0xf7,0x07,0xc0,0x00
,0x00,0x00,0x07,0xe1,0xe3,0x83,0xf8,0x00
,0x00,0x00,0x3f,0x87,0xc3,0xc0,0xfc,0x00
,0x00,0x01,0xfe,0x0f,0x81,0xe0,0x3c,0x00
,0x00,0x0f,0xf8,0x1f,0x00,0xf0,0x00,0x00
```

```
,0x00,0x1f,0xc0,0x7c,0x00,0x7c,0x00,0x00
,0x00,0x1e,0x01,0xf8,0x00,0x3f,0x00,0x00
,0x00,0x00,0x07,0xe0,0x78,0x0f,0xc0,0x00
,0x00,0x00,0x3f,0x81,0xfe,0x07,0xf0,0x00
,0x00,0x01,0xfe,0x07,0xff,0x01,0xf0,0x00
,0x00,0x07,0xf8,0x0f,0x87,0x80,0x30,0x00
,0x00,0x07,0xc0,0x3f,0x03,0xe0,0x00,0x00
,0x00,0x06,0x00,0xfc,0x01,0xf8,0x00,0x00
,0x00,0x00,0x03,0xf0,0x00,0x7e,0x00,0x00
,0x00,0x00,0x0f,0xc0,0x00,0x3f,0x80,0x00
,0x00,0x00,0x7f,0x00,0xf8,0x0f,0x80,0x00
,0x00,0x00,0xfc,0x03,0xfe,0x01,0x80,0x00
,0x00,0x00,0xf0,0x1f,0xff,0x80,0x00,0x00
,0x00,0x00,0x00,0x7f,0x07,0xe0,0x00,0x00
,0x00,0x00,0x00,0xfc,0x03,0xf8,0x00,0x00
,0x00,0x00,0x03,0xf0,0x00,0x78,0x00,0x00
,0x00,0x00,0x0f,0xc0,0x00,0x18,0x00,0x00
,0x00,0x00,0x0f,0x01,0xf8,0x00,0x00,0x00
,0x00,0x00,0x00,0x07,0xfe,0x00,0x00,0x00
,0x00,0x00,0x00,0x1f,0xfe,0x00,0x00,0x00
,0x00,0x00,0x00,0x1e,0x0e,0x00,0x00,0x00
,0x00,0x00,0x00,0x18,0x00,0x00,0x00,0x00
};
//------------------------------------------------------------------
#define FinPr_valid_width 64
#define FinPr_valid_height 64
const uint8_t PROGMEM FinPr_valid_bits[] = {
  0x00,0x00,0x03,0xfe,0x00,0x00,0x00,0x00
,0x00,0x00,0x1f,0xff,0xe0,0x00,0x00,0x00
,0x00,0x00,0x7f,0xff,0xf8,0x00,0x00,0x00
,0x00,0x00,0xfc,0x00,0xfe,0x00,0x00,0x00
,0x00,0x03,0xe0,0x00,0x1f,0x00,0x00,0x00
,0x00,0x07,0xc0,0x00,0x07,0x80,0x00,0x00
,0x00,0x0f,0x80,0x00,0x03,0xe0,0x00,0x00
,0x00,0x0e,0x03,0xff,0x01,0xe0,0x00,0x00
,0x00,0x1c,0x1f,0xff,0xe0,0xf0,0x00,0x00
,0x00,0x3c,0x3f,0xff,0xf0,0x78,0x00,0x00
,0x00,0x78,0x7c,0x00,0xf8,0x3c,0x00,0x00
,0x00,0x70,0xf0,0x00,0x3c,0x1c,0x00,0x00
,0x00,0xe1,0xe0,0x00,0x1e,0x1c,0x00,0x00
,0x00,0xe1,0xc0,0x00,0x0f,0x0e,0x00,0x00
,0x00,0xc3,0x81,0xfc,0x07,0x0e,0x00,0x00
,0x00,0x03,0x83,0xff,0x07,0x8e,0x00,0x00
,0x00,0x07,0x07,0x8f,0x83,0x87,0x00,0x00
,0x00,0x0f,0x0f,0x03,0xc3,0x87,0x00,0x00
,0x00,0x1e,0x0e,0x01,0xc3,0x87,0x00,0x00
,0x00,0x3c,0x1c,0x00,0xe1,0x87,0x00,0x00
,0x00,0xf8,0x1c,0x30,0xe1,0x87,0x00,0x00
,0x07,0xf0,0x38,0x70,0xe1,0x86,0x00,0x00
```

```
,0x07,0xc0,0x78,0x70,0xe3,0x8e,0x00,0x00
,0x02,0x00,0xf0,0xf0,0xe3,0x8e,0x00,0x00
,0x00,0x01,0xe0,0xe0,0xe3,0x8e,0x00,0x00
,0x00,0x03,0xc1,0xe1,0xc3,0x8e,0x00,0x00
,0x00,0x0f,0x83,0xc3,0xc3,0x8e,0x00,0x00
,0x00,0x7f,0x07,0x83,0x83,0x0e,0x00,0x00
,0x07,0xfc,0x0f,0x07,0x83,0x0e,0x00,0x00
,0x07,0xf0,0x1e,0x0f,0x03,0x0e,0x00,0x00
,0x07,0x80,0x7c,0x1e,0x03,0x07,0x00,0x00
,0x00,0x00,0xf8,0x3c,0x03,0x87,0x80,0x00
,0x00,0x03,0xf0,0x78,0x03,0x83,0xc0,0x00
,0x00,0x1f,0xc0,0xf0,0x02,0x00,0x00,0x00
,0x00,0xff,0x01,0xe1,0xc0,0x0c,0x00,0x00
,0x07,0xfc,0x03,0xc3,0xe1,0xff,0xc0,0x00
,0x07,0xe0,0x0f,0x87,0xc7,0xff,0xf0,0x00
,0x07,0x00,0x3f,0x0f,0x0f,0xff,0xfc,0x00
,0x00,0x00,0x7c,0x3e,0x3f,0xff,0xfe,0x00
,0x00,0x03,0xf8,0x7c,0x3f,0xff,0xff,0x00
,0x00,0x1f,0xe0,0xf0,0x7f,0xff,0xff,0x80
,0x00,0xff,0x83,0xe0,0xff,0xff,0xff,0x80
,0x01,0xfc,0x07,0xc1,0xff,0xff,0xe3,0xc0
,0x01,0xe0,0x1f,0x01,0xff,0xff,0xc3,0xc0
,0x00,0x00,0xfe,0x01,0xff,0xff,0x87,0xe0
,0x00,0x03,0xf8,0x13,0xff,0xff,0x0f,0xe0
,0x00,0x1f,0xe0,0x73,0xff,0xfe,0x1f,0xe0
,0x00,0x7f,0x81,0xf3,0xff,0xfc,0x1f,0xe0
,0x00,0xfc,0x03,0xe3,0xef,0xf8,0x3f,0xe0
,0x00,0x60,0x0f,0xc3,0xc7,0xf0,0x7f,0xe0
,0x00,0x00,0x3f,0x03,0xc3,0xe0,0xff,0xe0
,0x00,0x00,0xfc,0x03,0xc1,0xc1,0xff,0xe0
,0x00,0x07,0xf0,0x13,0xe0,0x83,0xff,0xe0
,0x00,0x0f,0xc0,0x7b,0xf8,0x07,0xff,0xe0
,0x00,0x0f,0x01,0xf9,0xfc,0x0f,0xff,0xc0
,0x00,0x00,0x07,0xf1,0xfe,0x1f,0xff,0xc0
,0x00,0x00,0x1f,0xc0,0xff,0x3f,0xff,0x80
,0x00,0x00,0x7e,0x00,0xff,0xff,0xff,0x80
,0x00,0x00,0xfc,0x00,0x7f,0xff,0xff,0x00
,0x00,0x00,0xf0,0x1f,0x3f,0xff,0xfe,0x00
,0x00,0x00,0x00,0x7f,0x1f,0xff,0xfc,0x00
,0x00,0x00,0x01,0xff,0x8f,0xff,0xf8,0x00
,0x00,0x00,0x03,0xe0,0xe3,0xff,0xe0,0x00
,0x00,0x00,0x01,0x80,0x00,0x7f,0x00,0x00
};
//------------------------------------------------------------------
#define FinPr_invalid_width 64
#define FinPr_invalid_height 64
const uint8_t PROGMEM FinPr_invalid_bits[] = {
  0x00,0x00,0x03,0xfe,0x00,0x00,0x00,0x00
,0x00,0x00,0x1f,0xff,0xe0,0x00,0x00,0x00
```

```
,0x00,0x00,0x7f,0xff,0xf8,0x00,0x00,0x00
,0x00,0x00,0xfc,0x00,0xfe,0x00,0x00,0x00
,0x00,0x03,0xe0,0x00,0x1f,0x00,0x00,0x00
,0x00,0x07,0xc0,0x00,0x07,0x80,0x00,0x00
,0x00,0x0f,0x80,0x00,0x03,0xe0,0x00,0x00
,0x00,0x0e,0x03,0xff,0x01,0xe0,0x00,0x00
,0x00,0x1c,0x1f,0xff,0xe0,0xf0,0x00,0x00
,0x00,0x3c,0x3f,0xff,0xf0,0x78,0x00,0x00
,0x00,0x78,0x7c,0x00,0xf8,0x3c,0x00,0x00
,0x00,0x70,0xf0,0x00,0x3c,0x1c,0x00,0x00
,0x00,0xe1,0xe0,0x00,0x1e,0x1c,0x00,0x00
,0x00,0xe1,0xc0,0x00,0x0f,0x0e,0x00,0x00
,0x00,0xc3,0x81,0xfc,0x07,0x0e,0x00,0x00
,0x00,0x03,0x83,0xff,0x07,0x8e,0x00,0x00
,0x00,0x07,0x07,0x8f,0x83,0x87,0x00,0x00
,0x00,0x0f,0x0f,0x03,0xc3,0x87,0x00,0x00
,0x00,0x1e,0x0e,0x01,0xc3,0x87,0x00,0x00
,0x00,0x3c,0x1c,0x00,0xe1,0x87,0x00,0x00
,0x00,0xf8,0x1c,0x30,0xe1,0x87,0x00,0x00
,0x07,0xf0,0x38,0x70,0xe1,0x86,0x00,0x00
,0x07,0xc0,0x78,0x70,0xe3,0x8e,0x00,0x00
,0x02,0x00,0xf0,0xf0,0xe3,0x8e,0x00,0x00
,0x00,0x01,0xe0,0xe0,0xe3,0x8e,0x00,0x00
,0x00,0x03,0xc1,0xe1,0xc3,0x8e,0x00,0x00
,0x00,0x0f,0x83,0xc3,0xc3,0x8e,0x00,0x00
,0x00,0x7f,0x07,0x83,0x83,0x0e,0x00,0x00
,0x07,0xfc,0x0f,0x07,0x83,0x0e,0x00,0x00
,0x07,0xf0,0x1e,0x0f,0x03,0x0e,0x00,0x00
,0x07,0x80,0x7c,0x1e,0x03,0x07,0x00,0x00
,0x00,0x00,0xf8,0x3c,0x03,0x87,0x80,0x00
,0x00,0x03,0xf0,0x78,0x03,0x83,0xc0,0x00
,0x00,0x1f,0xc0,0xf0,0x02,0x00,0x00,0x00
,0x00,0xff,0x01,0xe1,0xc0,0x00,0x00,0x00
,0x07,0xfc,0x03,0xc3,0xe1,0xff,0xc0,0x00
,0x07,0xe0,0x0f,0x87,0xc7,0xff,0xf0,0x00
,0x07,0x00,0x3f,0x0f,0x0f,0xff,0xf8,0x00
,0x00,0x00,0x7c,0x3e,0x1f,0xff,0xfe,0x00
,0x00,0x03,0xf8,0x7c,0x3f,0xff,0xff,0x00
,0x00,0x1f,0xe0,0xf0,0x7f,0xff,0xff,0x00
,0x00,0xff,0x83,0xe0,0xfe,0xff,0xbf,0x80
,0x01,0xfc,0x07,0xc0,0xfc,0x7f,0x1f,0xc0
,0x01,0xe0,0x1f,0x01,0xf8,0x3e,0x0f,0xc0
,0x00,0x00,0xfe,0x01,0xf8,0x1c,0x07,0xe0
,0x00,0x03,0xf8,0x13,0xf8,0x00,0x0f,0xe0
,0x00,0x1f,0xe0,0x73,0xfc,0x00,0x1f,0xe0
,0x00,0x7f,0x81,0xf3,0xfe,0x00,0x3f,0xe0
,0x00,0xfc,0x03,0xe3,0xff,0x00,0x7f,0xe0
,0x00,0x60,0x0f,0xc3,0xff,0x80,0xff,0xe0
,0x00,0x00,0x3f,0x03,0xff,0x00,0x7f,0xe0
```

```
,0x00,0x00,0xfc,0x03,0xfe,0x00,0x3f,0xe0
,0x00,0x07,0xf0,0x13,0xfc,0x00,0x1f,0xe0
,0x00,0x0f,0xc0,0x79,0xf8,0x08,0x0f,0xe0
,0x00,0x0f,0x01,0xf9,0xf8,0x1c,0x0f,0xc0
,0x00,0x00,0x07,0xf1,0xfc,0x3e,0x1f,0xc0
,0x00,0x00,0x1f,0xc0,0xfe,0x7f,0x3f,0x80
,0x00,0x00,0x7e,0x00,0xff,0xff,0xff,0x80
,0x00,0x00,0xfc,0x00,0x7f,0xff,0xff,0x00
,0x00,0x00,0xf0,0x1f,0x3f,0xff,0xfe,0x00
,0x00,0x00,0x00,0x7f,0x1f,0xff,0xfc,0x00
,0x00,0x00,0x01,0xff,0x8f,0xff,0xf8,0x00
,0x00,0x00,0x03,0xe0,0xe3,0xff,0xe0,0x00
,0x00,0x00,0x01,0x80,0x00,0x7f,0x00,0x00
};
//--------------------------------------------------------------
#define FinPr_failed_width 64
#define FinPr_failed_height 64
const uint8_t PROGMEM FinPr_failed_bits[] = {
0x00,0x00,0x3f,0xe0,0x00,0x00,0x00,0x00
,0x00,0x01,0xff,0xfe,0x00,0x00,0x00,0x00
,0x00,0x0f,0xc0,0x1f,0x80,0x00,0x00,0x00
,0x00,0x1e,0x00,0x03,0xc0,0x00,0x00,0x00
,0x00,0x78,0x00,0x00,0xf0,0x00,0x00,0x00
,0x00,0xe0,0x00,0x00,0x38,0x00,0x00,0x00
,0x01,0xc0,0x00,0x00,0x1c,0x00,0x00,0x00
,0x03,0x80,0x00,0x00,0x0e,0x00,0x00,0x00
,0x07,0x00,0x7f,0xe0,0x07,0x00,0x00,0x00
,0x06,0x01,0xff,0xf8,0x03,0x00,0x00,0x00
,0x0c,0x03,0xc0,0x3c,0x03,0x80,0x00,0x00
,0x1c,0x0f,0x00,0x0e,0x01,0x80,0x00,0x00
,0x18,0x0c,0x00,0x03,0x00,0xc0,0x00,0x00
,0x18,0x18,0x00,0x01,0x80,0xc0,0x00,0x00
,0x30,0x38,0x00,0x01,0xc0,0xe0,0x00,0x00
,0x30,0x30,0x0f,0x00,0xc0,0x60,0x00,0x00
,0x30,0x30,0x3f,0xc0,0xe0,0x60,0x00,0x00
,0x70,0x60,0x78,0xe0,0x60,0x60,0x00,0x00
,0x60,0x60,0x60,0x60,0x60,0x70,0x00,0x00
,0x60,0x60,0x60,0x60,0x60,0x30,0x00,0x00
,0x60,0x60,0x60,0x60,0x30,0x30,0x00,0x00
,0x60,0x60,0x60,0x30,0x30,0x20,0x00,0x00
,0x60,0x60,0x60,0x30,0x30,0x01,0xe0,0x00
,0x60,0x60,0x60,0x30,0x30,0x0f,0xfc,0x00
,0x60,0x60,0x60,0x30,0x30,0x3f,0xff,0x00
,0x60,0x60,0x60,0x30,0x18,0x78,0x03,0x80
,0x60,0x60,0x60,0x30,0x1c,0x60,0x01,0x80
,0x60,0x60,0x30,0x38,0x0c,0xc0,0x00,0xc0
,0x00,0x60,0x30,0x18,0x00,0xc0,0x00,0xc0
,0x00,0x60,0x30,0x18,0x00,0xc0,0x00,0xc0
,0x00,0xe0,0x30,0x0c,0x01,0xc0,0x00,0xe0
```

```
,0x00,0xc0,0x18,0x0e,0x01,0xc0,0x00,0xe0
,0x60,0xc0,0x18,0x07,0x01,0xc0,0x00,0xe0
,0x01,0xc0,0x1c,0x03,0x81,0xc0,0x00,0xe0
,0x01,0x80,0x0c,0x01,0xc1,0xc0,0x00,0xe0
,0x03,0x80,0x0e,0x00,0xf1,0xc0,0x00,0xe0
,0x0f,0x00,0x06,0x00,0x01,0xc0,0x00,0xe0
,0x3e,0x01,0x03,0x00,0x01,0xc0,0x00,0xe0
,0x30,0x03,0x83,0x80,0x1f,0xff,0xff,0xfe
,0x00,0x03,0x81,0xc0,0x3f,0xff,0xff,0xff
,0x00,0x07,0xc0,0xe0,0x30,0x00,0x00,0x03
,0x00,0x0e,0xc0,0x78,0x30,0x00,0x00,0x03
,0x00,0x3c,0x60,0x1e,0x30,0x00,0x00,0x03
,0x00,0x78,0x70,0x0f,0x30,0x00,0x00,0x03
,0x03,0xe0,0x38,0x03,0x30,0x00,0x00,0x03
,0x07,0x80,0x1c,0x00,0x30,0x00,0x00,0x03
,0xc0,0x00,0x0f,0x00,0x30,0x00,0x00,0x03
,0xc0,0x00,0x03,0x80,0x30,0x01,0xe0,0x03
,0x00,0x18,0x01,0xe0,0x30,0x03,0xf0,0x03
,0x00,0x18,0x00,0x7c,0x30,0x07,0x38,0x03
,0x00,0x0c,0x00,0x1f,0x30,0x06,0x18,0x03
,0x18,0x0e,0x00,0x07,0x30,0x06,0x18,0x03
,0x0c,0x07,0x80,0x00,0x30,0x07,0x38,0x03
,0x0e,0x03,0xc0,0x00,0x30,0x03,0x30,0x03
,0x07,0x00,0xf0,0x00,0x30,0x03,0x30,0x03
,0x03,0x00,0x7e,0x00,0x30,0x03,0x30,0x03
,0x01,0x80,0x1f,0xc0,0x30,0x03,0x30,0x03
,0x01,0xc0,0x03,0xe1,0x30,0x07,0xf8,0x03
,0x00,0xf0,0x00,0x01,0x30,0x03,0xf0,0x03
,0x00,0x38,0x00,0x00,0x30,0x00,0x00,0x03
,0x00,0x1e,0x00,0x00,0x30,0x00,0x00,0x03
,0x00,0x07,0xc0,0x00,0x30,0x00,0x00,0x03
,0x00,0x01,0xff,0x80,0x3f,0xff,0xff,0xff
,0x00,0x00,0x3f,0x80,0x1f,0xff,0xff,0xfe
};
//------------------------------------------------------------------
#define FinPr_scan_width 64
#define FinPr_scan_height 64
const uint8_t PROGMEM FinPr_scan_bits[] = {
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x1f,0xf8,0x00,0x00,0x00
,0x00,0x00,0x00,0x7f,0xff,0x00,0x00,0x00
,0x00,0x00,0x01,0xfc,0x7f,0xc0,0x00,0x00
,0x00,0x00,0x03,0xc0,0x03,0xe0,0x00,0x00
,0x00,0x00,0x07,0x80,0x00,0xf0,0x00,0x00
,0x00,0x00,0x0e,0x00,0x00,0x3c,0x00,0x00
```

```
,0x00,0x00,0x1c,0x1f,0xfc,0x1c,0x00,0x00
,0x00,0x00,0x38,0x7f,0xfe,0x0e,0x00,0x00
,0x00,0x00,0x78,0xf8,0x0f,0x87,0x00,0x00
,0x00,0x00,0x71,0xe0,0x03,0xc7,0x00,0x00
,0x00,0x00,0xe3,0x80,0x01,0xc3,0x80,0x00
,0x00,0x00,0xc3,0x83,0xc0,0xe3,0x80,0x00
,0x00,0x00,0xc7,0x0f,0xf0,0x71,0x80,0x00
,0x00,0x00,0x06,0x1f,0xf8,0x71,0xc0,0x00
,0x00,0x00,0x0e,0x1c,0x3c,0x31,0xc0,0x00
,0x00,0x00,0x1c,0x38,0x1c,0x31,0xc0,0x00
,0x00,0x00,0x38,0x70,0x0e,0x39,0xc0,0x00
,0x00,0x01,0xf0,0x71,0x8e,0x39,0xc0,0x00
,0x00,0x03,0xe0,0xe1,0x86,0x31,0xc0,0x00
,0x00,0x03,0x81,0xe3,0x8e,0x31,0x80,0x00
,0x00,0x00,0x03,0xc3,0x8e,0x33,0x80,0x00
,0x00,0x00,0x07,0x87,0x0c,0x73,0x80,0x00
,0x00,0x00,0x1f,0x0e,0x1c,0x73,0x80,0x00
,0x7f,0xff,0xff,0xff,0xff,0xff,0xff,0xfe
,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff
,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff
,0x7f,0xff,0xff,0xff,0xff,0xff,0xff,0xfe
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x03,0xf0,0x1e,0x3e,0x1c,0x00,0x00
,0x00,0x03,0x80,0x7c,0x77,0x0f,0x00,0x00
,0x00,0x00,0x01,0xf0,0xe3,0x07,0xc0,0x00
,0x00,0x00,0x07,0xe3,0xc3,0x81,0xf0,0x00
,0x00,0x00,0x3f,0x87,0x81,0xc0,0x60,0x00
,0x00,0x01,0xfc,0x1f,0x00,0xf0,0x00,0x00
,0x00,0x01,0xe0,0x3c,0x00,0x7c,0x00,0x00
,0x00,0x00,0x00,0xf8,0x78,0x1f,0x00,0x00
,0x00,0x00,0x07,0xe0,0xfc,0x0f,0xc0,0x00
,0x00,0x00,0x3f,0x83,0xef,0x03,0xc0,0x00
,0x00,0x00,0xfc,0x0f,0x87,0x80,0x00,0x00
,0x00,0x00,0x70,0x1f,0x03,0xe0,0x00,0x00
,0x00,0x00,0x00,0x7c,0x00,0xf8,0x00,0x00
,0x00,0x00,0x01,0xf0,0x00,0x3e,0x00,0x00
,0x00,0x00,0x0f,0xc0,0xf8,0x0f,0x00,0x00
,0x00,0x00,0x1f,0x03,0xfe,0x02,0x00,0x00
,0x00,0x00,0x0c,0x0f,0x8f,0x80,0x00,0x00
,0x00,0x00,0x00,0x3f,0x03,0xe0,0x00,0x00
,0x00,0x00,0x00,0xf8,0x00,0xf0,0x00,0x00
,0x00,0x00,0x01,0xe0,0x00,0x30,0x00,0x00
,0x00,0x00,0x01,0xc0,0xf8,0x00,0x00,0x00
,0x00,0x00,0x00,0x07,0xfe,0x00,0x00,0x00
,0x00,0x00,0x00,0x0f,0x8e,0x00,0x00,0x00
,0x00,0x00,0x00,0x06,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
```

```
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
};

////////////////////////////////////////////////
//     SETUP
////////////////////////////////////////////////
void setup() {
  Serial.begin(115200);
  delay(1000);

  //-----------initiate OLED display-------------
  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  // you can delet these three lines if you don't want to get the Adfruit logo appear
  display.display();
  delay(2000);                                  // Pause for 2 seconds
  display.clearDisplay();

  //-------------------------------------------
  connectToWiFi();
  //-------------------------------------------

  // set the data rate for the sensor serial port
  finger.begin(57600);
  Serial.println("\n\nAdafruit finger detect test");

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
    display.clearDisplay();
    display.drawBitmap( 34, 0, FinPr_valid_bits, FinPr_valid_width,
FinPr_valid_height, WHITE);
    display.display();
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    display.clearDisplay();
    display.drawBitmap( 32, 0, FinPr_failed_bits, FinPr_failed_width,
FinPr_failed_height, WHITE);
    display.display();
    while (1) { delay(1); }
```

```
  }
  //-----------------------------------------
  finger.getTemplateCount();
  Serial.print("Sensor contains "); Serial.print(finger.templateCount);
Serial.println(" templates");
  Serial.println("Waiting for valid finger...");

  //Timers----------------------------------------
  timer.setInterval(25000L, CheckMode);
  t1 = timer.setInterval(10000L, ChecktoAddID);      //Set an internal timer every
10sec to check if there a new fingerprint in the website to add it.
  t2 = timer.setInterval(15000L, ChecktoDeleteID);   //Set an internal timer every
15sec to check wheater there an ID to delete in the website.

  //-----------------------------------------
  CheckMode();

}

/////////////////////////////////////////////////
//     WHILE TRUE - LOOP
/////////////////////////////////////////////////

void loop() {
  timer.run();                       //Keep the timer in the loop function in order
to update the time as soon as possible
  //check if there's a connection to Wi-Fi or not
  if(!WiFi.isConnected()){
    if (millis() - previousMillis >= 10000) {
      previousMillis = millis();
      connectToWiFi();                  //Retry to connect to Wi-Fi
    }
  }
  CheckFingerprint();                     //Check the sensor if the there a finger.
  delay(10);
}

/////////////////////////////////////////////////
//     CHECK FINGERPRINT
/////////////////////////////////////////////////

void CheckFingerprint(){
  //  unsigned long previousMillisM = millis();
  //  Serial.println(previousMillisM);
  // If there no fingerprint has been scanned return -1 or -2 if there an error or 0
if there nothing, The ID start form 1 to 127
  // Get the Fingerprint ID from the Scanner
  FingerID = getFingerprintID();
  DisplayFingerprintID();
```

```
  //  Serial.println(millis() - previousMillisM);

}

////////////////////////////////////////////////
//     Display the fingerprint ID state on the OLED
////////////////////////////////////////////////
void DisplayFingerprintID(){
  //Fingerprint has been detected
  if (FingerID > 0){
    display.clearDisplay();
    display.drawBitmap( 34, 0, FinPr_valid_bits, FinPr_valid_width,
FinPr_valid_height, WHITE);
    display.display();

    SendFingerprintID( FingerID );            // Send the Fingerprint ID to the
website.
    delay(2000);
  }

//-------------------------------------------
  //No finger detected
  else if (FingerID == 0){
    display.clearDisplay();
    display.drawBitmap( 32, 0, FinPr_start_bits, FinPr_start_width,
FinPr_start_height, WHITE);
    display.display();
  }
  //-------------------------------------------
  //Didn't find a match
  else if (FingerID == -1){
    display.clearDisplay();
    display.drawBitmap( 34, 0, FinPr_invalid_bits, FinPr_invalid_width,
FinPr_invalid_height, WHITE);
    display.display();
  }
  //-------------------------------------------
  //Didn't find the scanner or there an error
  else if (FingerID == -2){
    display.clearDisplay();
    display.drawBitmap( 32, 0, FinPr_failed_bits, FinPr_failed_width,
FinPr_failed_height, WHITE);
    display.display();
  }
}

////////////////////////////////////////////////
//     send the fingerprint ID to IoT Edge website
////////////////////////////////////////////////
```

```
void SendFingerprintID( int finger ){
  Serial.println("Sending the Fingerprint ID");
  if(WiFi.isConnected()){
    HTTPClient http;    //Declare object of class HTTPClient
    //GET Data
    getData = "?FingerID=" + String(finger) + "&device_token=" + device_token; // Add
the Fingerprint ID to the Post array in order to send it
    //GET methode
    Link = URL + getData;
    http.begin(Link); //initiate HTTP request   //Specify content-type header

    int httpCode = http.GET();   //Send the request
    String payload = http.getString();    //Get the response payload

    Serial.println(httpCode);   //Print HTTP return code
    Serial.println(payload);    //Print request response payload
    Serial.println(finger);     //Print fingerprint ID

    if (payload.substring(0, 5) == "login") {
      String user_name = payload.substring(5);
  //  Serial.println(user_name);

      display.clearDisplay();
      display.setTextSize(2);             // Normal 2:2 pixel scale
      display.setTextColor(WHITE);        // Draw white text
      display.setCursor(15,0);             // Start at top-left corner
      display.print(F("Welcome"));
      display.setCursor(0,20);
      display.print(user_name);
      display.display();
    }
    else if (payload.substring(0, 6) == "logout") {
      String user_name = payload.substring(6);
  //  Serial.println(user_name);

      display.clearDisplay();
      display.setTextSize(2);             // Normal 2:2 pixel scale
      display.setTextColor(WHITE);        // Draw white text
      display.setCursor(10,0);             // Start at top-left corner
      display.print(F("Good Bye"));
      display.setCursor(0,20);
      display.print(user_name);
      display.display();
    }
    delay(10);
    http.end();  //Close connection
  }
}

```

```
////////////////////////////////////////////////
//     Get the Fingerprint ID
////////////////////////////////////////////////

int getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      //Serial.println("No finger detected");
      return 0;
    case FINGERPRINT_PACKETRECIEVEERR:
      //Serial.println("Communication error");
      return -2;
    case FINGERPRINT_IMAGEFAIL:
      //Serial.println("Imaging error");
      return -2;
    default:
      //Serial.println("Unknown error");
      return -2;
  }
  // OK success!
  p = finger.image2Tz();
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      //Serial.println("Image too messy");
      return -1;
    case FINGERPRINT_PACKETRECIEVEERR:
      //Serial.println("Communication error");
      return -2;
    case FINGERPRINT_FEATUREFAIL:
      //Serial.println("Could not find fingerprint features");
      return -2;
    case FINGERPRINT_INVALIDIMAGE:
      //Serial.println("Could not find fingerprint features");
      return -2;
    default:
      //Serial.println("Unknown error");
      return -2;
  }
  // OK converted!
  p = finger.fingerFastSearch();
  if (p == FINGERPRINT_OK) {
    //Serial.println("Found a print match!");
```

```
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    //Serial.println("Communication error");
    return -2;
  } else if (p == FINGERPRINT_NOTFOUND) {
    //Serial.println("Did not find a match");
    return -1;
  } else {
    //Serial.println("Unknown error");
    return -2;
  }
  // found a match!
  //Serial.print("Found ID #"); Serial.print(finger.fingerID);
  //Serial.print(" with confidence of "); Serial.println(finger.confidence);

  return finger.fingerID;
}


////////////////////////////////////////////////
//     Check if there a Fingerprint ID to delete
void ChecktoDeleteID(){
  Serial.println("Check to Delete ID");
  if(WiFi.isConnected()){
    HTTPClient http;    //Declare object of class HTTPClient
    //GET Data
    getData = "?DeleteID=check&device_token=" + String(device_token); // Add the
Fingerprint ID to the Post array in order to send it
    //GET methode
    Link = URL + getData;
    http.begin(Link); //initiate HTTP request,
//    Serial.println(Link);
    int httpCode = http.GET();                //Send the request
    String payload = http.getString();        //Get the response payload

    if (payload.substring(0, 6) == "del-id") {
      String del_id = payload.substring(6);
      Serial.println(del_id);
      http.end();  //Close connection
      deleteFingerprint( del_id.toInt() );
      delay(1000);
    }
    http.end();  //Close connection
  }
}

////////////////////////////////////////////////
//      Delete Finpgerprint ID
////////////////////////////////////////////////
uint8_t deleteFingerprint( int id) {
```

```
  uint8_t p = -1;

  p = finger.deleteModel(id);

  if (p == FINGERPRINT_OK) {
    //Serial.println("Deleted!");
    display.clearDisplay();
    display.setTextSize(2);                   // Normal 2:2 pixel scale
    display.setTextColor(WHITE);              // Draw white text
    display.setCursor(0,0);                   // Start at top-left corner
    display.print(F("Deleted!\n"));
    display.display();
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    //Serial.println("Communication error");
    display.clearDisplay();
    display.setTextSize(1);                   // Normal 1:1 pixel scale
    display.setTextColor(WHITE);              // Draw white text
    display.setCursor(0,0);                   // Start at top-left corner
    display.print(F("Communication error!\n"));
    display.display();
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    //Serial.println("Could not delete in that location");
    display.clearDisplay();
    display.setTextSize(1);                   // Normal 1:1 pixel scale
    display.setTextColor(WHITE);              // Draw white text
    display.setCursor(0,0);                   // Start at top-left corner
    display.print(F("Could not delete in that location!\n"));
    display.display();
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    //Serial.println("Error writing to flash");
    display.clearDisplay();
    display.setTextSize(1);                   // Normal 1:1 pixel scale
    display.setTextColor(WHITE);              // Draw white text
    display.setCursor(0,0);                   // Start at top-left corner
    display.print(F("Error writing to flash!\n"));
    display.display();
    return p;
  } else {
    //Serial.print("Unknown error: 0x"); Serial.println(p, HEX);
    display.clearDisplay();
    display.setTextSize(2);                   // Normal 2:2 pixel scale
    display.setTextColor(WHITE);              // Draw white text
    display.setCursor(0,0);                   // Start at top-left corner
    display.print(F("Unknown error:\n"));
    display.display();
    return p;
  }
```

```
}


////////////////////////////////////////////////
//      Check if there a Fingerprint ID to add
////////////////////////////////////////////////
void ChecktoAddID(){
//  Serial.println("Check to Add ID");
  if(WiFi.isConnected()){
    HTTPClient http;                         //Declare object of class HTTPClient
    //GET Data
    getData = "?Get_Fingerid=get_id&device_token=" + String(device_token); // Add the
Fingerprint ID to the Post array in order to send it
    //GET methode
    Link = URL + getData;
    http.begin(Link);                        //initiate HTTP request,
//    Serial.println(Link);
    int httpCode = http.GET();               //Send the request
    String payload = http.getString();       //Get the response payload

    if (payload.substring(0, 6) == "add-id") {
      String add_id = payload.substring(6);
      Serial.println(add_id);
      id = add_id.toInt();
      http.end();                            //Close connection
      getFingerprintEnroll();
    }
    http.end();  //Close connection
  }
}

////////////////////////////////////////////////
//          Check the Mode
////////////////////////////////////////////////

void CheckMode(){
  Serial.println("Check Mode");
  if(WiFi.isConnected()){
    HTTPClient http;    //Declare object of class HTTPClient
    //GET Data
    getData = "?Check_mode=get_mode&device_token=" + String(device_token); // Add the
Fingerprint ID to the Post array in order to send it
    //GET methode
    Link = URL + getData;
    http.begin(Link); //initiate HTTP request,
//    Serial.println(Link);
    int httpCode = http.GET();   //Send the request
    String payload = http.getString();    //Get the response payload

```

```
    if (payload.substring(0, 4) == "mode") {
      String dev_mode = payload.substring(4);
      int devMode = dev_mode.toInt();
      if(!firstConnect){
        device_Mode = devMode;
        firstConnect = true;
      }
//      Serial.println(dev_mode);
      if(device_Mode && devMode){
        device_Mode = false;
        timer.disable(t1);
        timer.disable(t2);
        Serial.println("Deivce Mode: Attandance");
      }
      else if(!device_Mode && !devMode){
        device_Mode = true;
        timer.enable(t1);
        timer.enable(t2);
        Serial.println("Deivce Mode: Enrollment");
      }
      http.end();  //Close connection
    }
    http.end();  //Close connection
  }
//  Serial.print("Number of Timers: ");
//  Serial.println(timer.getNumTimers());
}

////////////////////////////////////////////////
//       Enroll a Finpgerprint ID
////////////////////////////////////////////////
uint8_t getFingerprintEnroll() {
  int p = -1;
  display.clearDisplay();
  display.drawBitmap( 34, 0, FinPr_scan_bits, FinPr_scan_width, FinPr_scan_height,
WHITE);
  display.display();
  while (p != FINGERPRINT_OK) {

    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image taken");
      display.clearDisplay();
      display.drawBitmap( 34, 0, FinPr_valid_bits, FinPr_valid_width,
FinPr_valid_height, WHITE);
      display.display();
      break;
    case FINGERPRINT_NOFINGER:
```

```
      //Serial.println(".");
      display.setTextSize(1);             // Normal 2:2 pixel scale
      display.setTextColor(WHITE);        // Draw white text
      display.setCursor(0,0);             // Start at top-left corner
      display.print(F("scanning"));
      display.display();
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      display.clearDisplay();
      display.drawBitmap( 34, 0, FinPr_invalid_bits, FinPr_invalid_width,
FinPr_invalid_height, WHITE);
      display.display();
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }

  // OK success!
  p = finger.image2Tz(1);
  switch (p) {
    case FINGERPRINT_OK:
      display.clearDisplay();
      display.drawBitmap( 34, 0, FinPr_valid_bits, FinPr_valid_width,
FinPr_valid_height, WHITE);
      display.display();
      break;
    case FINGERPRINT_IMAGEMESS:
      display.clearDisplay();
      display.drawBitmap( 34, 0, FinPr_invalid_bits, FinPr_invalid_width,
FinPr_invalid_height, WHITE);
      display.display();
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint features");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
```

```
  }
  display.clearDisplay();
  display.setTextSize(2);             // Normal 2:2 pixel scale
  display.setTextColor(WHITE);        // Draw white text
  display.setCursor(0,0);             // Start at top-left corner
  display.print(F("Remove"));
  display.setCursor(0,20);
  display.print(F("finger"));
  display.display();
  //Serial.println("Remove finger");
  delay(2000);
  p = 0;
  while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
  }
  Serial.print("ID "); Serial.println(id);
  p = -1;
  display.clearDisplay();
  display.drawBitmap( 34, 0, FinPr_scan_bits, FinPr_scan_width, FinPr_scan_height,
WHITE);
  display.display();
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image taken");
      display.clearDisplay();
      display.drawBitmap( 34, 0, FinPr_valid_bits, FinPr_valid_width,
FinPr_valid_height, WHITE);
      display.display();
      break;
    case FINGERPRINT_NOFINGER:
      //Serial.println(".");
      display.setTextSize(1);             // Normal 2:2 pixel scale
      display.setTextColor(WHITE);        // Draw white text
      display.setCursor(0,0);             // Start at top-left corner
      display.print(F("scanning"));
      display.display();
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
```

```
  }

  // OK success!

  p = finger.image2Tz(2);
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image converted");
      display.clearDisplay();
      display.drawBitmap( 34, 0, FinPr_valid_bits, FinPr_valid_width,
FinPr_valid_height, WHITE);
      display.display();
      break;
    case FINGERPRINT_IMAGEMESS:
      //Serial.println("Image too messy");
      display.clearDisplay();
      display.drawBitmap( 34, 0, FinPr_invalid_bits, FinPr_invalid_width,
FinPr_invalid_height, WHITE);
      display.display();
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint features");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  // OK converted!
  Serial.print("Creating model for #");  Serial.println(id);

  p = finger.createModel();
  if (p == FINGERPRINT_OK) {
    Serial.println("Prints matched!");
    display.clearDisplay();
    display.drawBitmap( 34, 0, FinPr_valid_bits, FinPr_valid_width,
FinPr_valid_height, WHITE);
    display.display();
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
      Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_ENROLLMISMATCH) {
      Serial.println("Fingerprints did not match");
```

```
      display.clearDisplay();
      display.drawBitmap( 34, 0, FinPr_invalid_bits, FinPr_invalid_width,
FinPr_invalid_height, WHITE);
      display.display();
    return p;
  } else {
      Serial.println("Unknown error");
    return p;
  }

  Serial.print("ID "); Serial.println(id);
  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Stored!");
    display.clearDisplay();
    display.drawBitmap( 34, 0, FinPr_valid_bits, FinPr_valid_width,
FinPr_valid_height, WHITE);
    display.display();
    confirmAdding(id);
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }
}

////////////////////////////////////////////////
//      Check if there a Fingerprint ID to add
////////////////////////////////////////////////
void confirmAdding(int id){
  Serial.println("confirm Adding");
  if(WiFi.status() == WL_CONNECTED){
    HTTPClient http;    //Declare object of class HTTPClient
    //GET Data
    getData = "?confirm_id=" + String(id) + "&device_token=" + String(device_token);
// Add the Fingerprint ID to the Post array in order to send it
    //GET methode
    Link = URL + getData;

    http.begin(Link); //initiate HTTP request,
//    Serial.println(Link);
```

```
    int httpCode = http.GET();   //Send the request
    String payload = http.getString();    //Get the response payload
    if(httpCode == 200){
      display.clearDisplay();
      display.setTextSize(1.5);             // Normal 1:1 pixel scale
      display.setTextColor(WHITE);        // Draw white text
      display.setCursor(0,0);             // Start at top-left corner
      display.print(payload);
      display.display();
      Serial.println(payload);
      delay(2000);
    }
    else{
      Serial.println("Error Confirm!!");
    }
    http.end();  //Close connection
  }
}


//////////////////////////////////////////////////
//     connect to the WiFi
//////////////////////////////////////////////////
void connectToWiFi(){
    WiFi.mode(WIFI_OFF);                      //Prevents reconnection issue (taking
too long to connect)
    delay(1000);
    WiFi.mode(WIFI_STA);
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    display.clearDisplay();
    display.setTextSize(1);                   // Normal 1:1 pixel scale
    display.setTextColor(WHITE);              // Draw white text
    display.setCursor(0, 0);                  // Start at top-left corner
    display.print(F("Connecting to \n"));
    display.setCursor(0, 50);
    display.setTextSize(2);
    display.print(ssid);
    display.drawBitmap( 73, 10, Wifi_start_bits, Wifi_start_width, Wifi_start_height,
WHITE);
    display.display();


    uint32_t periodToConnect = 30000L;
    for(uint32_t StartToConnect = millis(); (millis()-StartToConnect) <
periodToConnect;){
      if ( WiFi.status() != WL_CONNECTED ){
```

```
        delay(500);
        Serial.print(".");
      } else{
        break;
      }
    }

    if(WiFi.isConnected()){
      Serial.println("");
      Serial.println("Connected");

      display.clearDisplay();
      display.setTextSize(2);             // Normal 1:1 pixel scale
      display.setTextColor(WHITE);        // Draw white text
      display.setCursor(8, 0);             // Start at top-left corner
      display.print(F("Connected \n"));
      display.drawBitmap( 33, 15, Wifi_connected_bits, Wifi_connected_width,
Wifi_connected_height, WHITE);
      display.display();

      Serial.print("IP address: ");
      Serial.println(WiFi.localIP());  //IP address assigned to your ESP
    }
    else{
      Serial.println("");
      Serial.println("Not Connected");
      WiFi.mode(WIFI_OFF);
      delay(1000);
    }
    delay(1000);
}
////////////////////////////////////////////////
//    END OF CODE
////////////////////////////////////////////////
```

Annex C – Change Bitmap to array

There are many online tools to change images into byte arrays (or your array back into an image) for use with Arduino and (monochrome) displays such as OLEDs. The web page https://javl.github.io/image2cpp/ were used to convert the following images used in this project.

BMP simbol	ID	Use
	Wifi_start	Waiting valid Wi-fi credentials
	Wifi_connected	The IoT device is connected to Wi-fi
	FinPr_start	No finger detected
	FinPr_failed	Didn't find the scanner or there an error
	FinPr_scan	Enroll a Fingerprint
	FinPr_valid	Fingerprint has been detected
	FinPr_invalid	Didn't find a match

https://javl.github.io/image2cpp/

3. Preview

4. Output

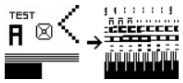
Code output format Arduino code

Adds some extra Arduino code around the output for easy copy-paste into this example. If multiple images are loaded, generates a byte array for each and appends a counter to the identifier.

Identifier/Prefix: epd_bitmap_

Draw mode: Horizontal - 1 bit per pixel

If your image looks all messed up on your display, like the image below, try using a different mode.



Generate code

```
// 'Wifi_connected', 63x49px
const unsigned char epd_bitmap_Wifi_connected [] PROGMEM = {
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0x80, 0x00, 0x00,
        0x00, 0x00, 0x3f, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00,
        0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x3f, 0xff, 0xc0, 0x07, 0xff, 0xf8, 0x00,
        0x00, 0xff, 0xf8, 0x00, 0x00, 0x3f, 0xfe, 0x00, 0x03, 0xff, 0x80, 0x00, 0x00, 0x03, 0xff, 0x80,
        0x07, 0xfe, 0x00, 0x00, 0x00, 0x00, 0xff, 0xc0, 0x1f, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0,
        0x3f, 0xe0, 0x01, 0xff, 0xff, 0x00, 0x0f, 0xf8, 0x7f, 0x80, 0x0f, 0xff, 0xff, 0xe0, 0x03, 0xfc,
        0xff, 0x00, 0x7f, 0xff, 0xff, 0xfc, 0x01, 0xfe, 0xfc, 0x01, 0xff, 0xff, 0xff, 0xff, 0x00, 0x7e,
        0x78, 0x07, 0xff, 0xc0, 0x07, 0xff, 0xc0, 0x3c, 0x00, 0x0f, 0xfc, 0x00, 0x00, 0x7f, 0xe0, 0x00,
```

Annex D – Smart contract – timinglog.sol

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.6.1;
pragma experimental ABIEncoderV2;

contract timinglog {

    uint constant DAY_IN_SECONDS = 86400;
    uint constant YEAR_IN_SECONDS = 31536000;
    uint constant LEAP_YEAR_IN_SECONDS = 31622400;

    uint constant HOUR_IN_SECONDS = 3600;
    uint constant MINUTE_IN_SECONDS = 60;

    uint16 constant ORIGIN_YEAR = 1970;

    function isLeapYear(uint year) pure private returns (bool) {
                if (year % 4 != 0) {
                        return false;
                }
                if (year % 100 != 0) {
                        return true;
                }
                if (year % 400 != 0) {
                        return false;
                }
                return true;
    }

    function toTimestamp(uint year, uint month, uint day) view private returns (uint
timestamp) {
                return toTimestamp(year, month, day, 0, 0, 0);
    }

    function toTimestamp(uint year, uint month, uint day, uint hour, uint minute,
uint second) view private returns (uint timestamp) {
                uint16 i;

                // Year
                for (i = ORIGIN_YEAR; i < year; i++) {
                        if (isLeapYear(i)) {
                                timestamp += LEAP_YEAR_IN_SECONDS;
                        }
                        else {
                                timestamp += YEAR_IN_SECONDS;
                        }
                }

                // Month
                uint8[12] memory monthDayCounts;
                monthDayCounts[0] = 31;
                if (isLeapYear(year)) {
                        monthDayCounts[1] = 29;
                }
                else {
                        monthDayCounts[1] = 28;
                }
                monthDayCounts[2] = 31;
                monthDayCounts[3] = 30;
                monthDayCounts[4] = 31;
                monthDayCounts[5] = 30;
                monthDayCounts[6] = 31;
```

```
                monthDayCounts[7] = 31;
                monthDayCounts[8] = 30;
                monthDayCounts[9] = 31;
                monthDayCounts[10] = 30;
                monthDayCounts[11] = 31;

                for (i = 1; i < month; i++) {
                        timestamp += DAY_IN_SECONDS * monthDayCounts[i - 1];
                }

                // Day
                timestamp += DAY_IN_SECONDS * (day - 1);

                // Hour
                timestamp += HOUR_IN_SECONDS * (hour);

                // Minute
                timestamp += MINUTE_IN_SECONDS * (minute);

                // Second
                timestamp += second;

                return timestamp;
    }

    mapping (uint => mapping (uint => string)) internal comp_date_hash_map;
                                    // comp_date_hash_map[company][date] we get a
string

    function append_log(uint _company, uint year, uint month, uint day, string memory
_log) public returns (bool OK) {
        // check dates
        // check company account
        uint local_date = toTimestamp(year,month,day);
        bytes memory tempEmptyStringTest =
bytes(comp_date_hash_map[_company][local_date]);
        require(tempEmptyStringTest.length == 0); // must be empty the log at this
day
        comp_date_hash_map[_company][local_date] = _log;
        return true;
    }

    function get_log(uint _company, uint year, uint month, uint day) view public
returns (string memory _log) {
        uint local_date = toTimestamp(year,month,day);
        bytes memory tempEmptyStringTest =
bytes(comp_date_hash_map[_company][local_date]);
        require(tempEmptyStringTest.length != 0); // must have data the log at this
day
        _log = comp_date_hash_map[_company][local_date];
        return _log;
    }
}
```

Annex E – How to deploy a Smart contract

Deploy smart contract means to have it available to users of an Ethereum network. For this project, it was used Ropsten test network. To deploy a smart contract, is send an Ethereum transaction containing the code of the compiled smart contract without specifying any recipients. This means it will need to pay a transaction fee so make sure there is available some ETH.

For the EVM to be able to run your contract it needs to be in bytecode. Compilation turns this:

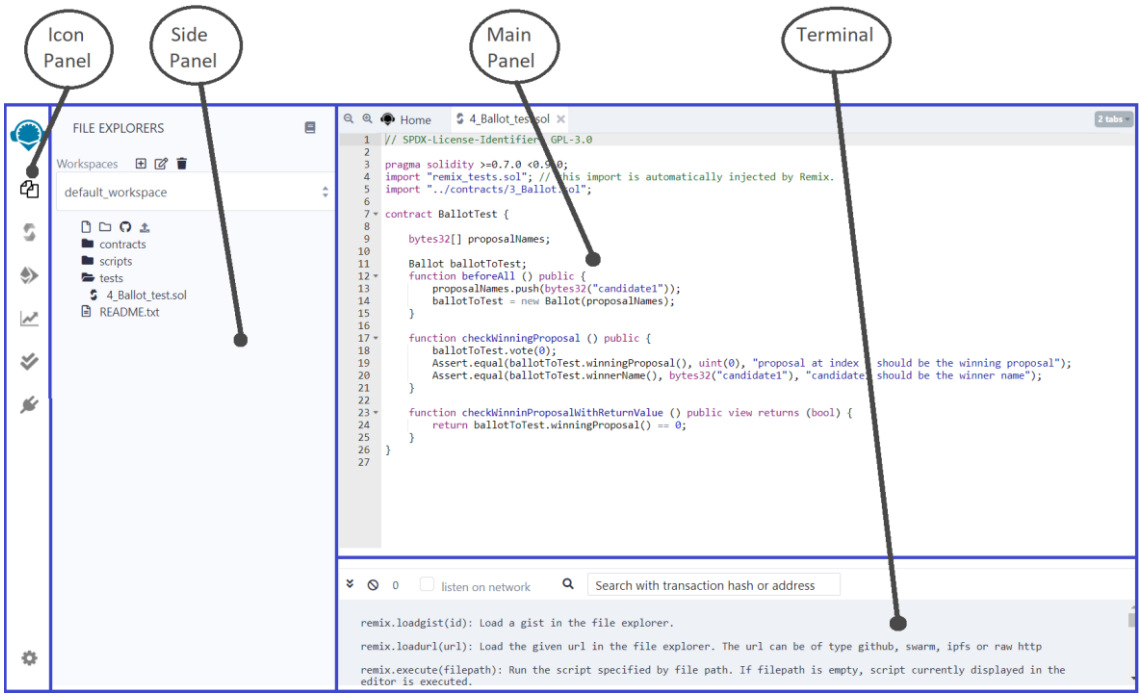
```
pragma solidity 0.4.24;

contract Greeter {
    function greet() public constant returns (string) {
        return "Hello";
    }
}
```

into this

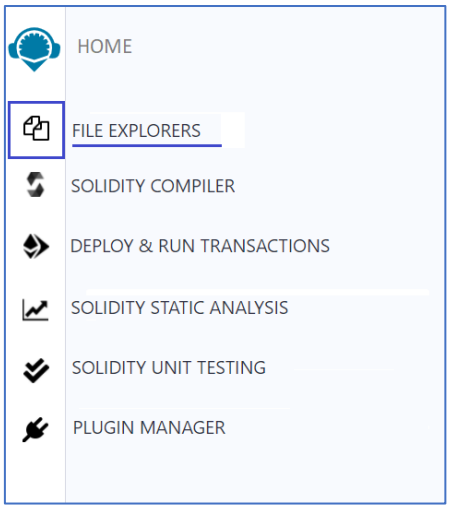
```
PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x4 CALLDATASIZE LT PUSH2 0x41 JUMPI PUSH1 0x0
CALLDATALOAD PUSH29 0x100000000000000000000000000000000000000000000000000000000
SWAP1 DIV PUSH4 0xFFFFFFFF AND DUP1 PUSH4 0xCFAE3217 EQ PUSH2 0x46 JUMPI JUMPDEST
PUSH1 0x0 DUP1 REVERT JUMPDEST CALLVALUE DUP1 ISZERO PUSH2 0x52 JUMPI PUSH1 0x0
DUP1 REVERT JUMPDEST POP PUSH2 0x5B PUSH2 0xD6 JUMP JUMPDEST PUSH1 0x40 MLOAD
DUP1 DUP1 PUSH1 0x20 ADD DUP3 DUP2 SUB DUP3 MSTORE DUP4 DUP2 DUP2 MLOAD DUP2
MSTORE PUSH1 0x20 ADD SWAP2 POP DUP1 MLOAD SWAP1 PUSH1 0x20 ADD SWAP1 DUP1 DUP4
DUP4 PUSH1 0x0 JUMPDEST DUP4 DUP2 LT ISZERO PUSH2 0x9B JUMPI DUP1 DUP3 ADD MLOAD
DUP2 DUP5 ADD MSTORE PUSH1 0x20 DUP2 ADD SWAP1 POP PUSH2 0x80 JUMP JUMPDEST POP
POP POP POP SWAP1 POP SWAP1 DUP2 ADD SWAP1 PUSH1 0x1F AND DUP1 ISZERO PUSH2 0xC8
JUMPI DUP1 DUP3 SUB DUP1 MLOAD PUSH1 0x1 DUP4 PUSH1 0x20 SUB PUSH2 0x100 EXP SUB
NOT AND DUP2 MSTORE PUSH1 0x20 ADD SWAP2 POP JUMPDEST POP SWAP3 POP POP POP PUSH1
0x40 MLOAD DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST PUSH1 0x60 PUSH1 0x40 DUP1 MLOAD
SWAP1 DUP2 ADD PUSH1 0x40 MSTORE DUP1 PUSH1 0x5 DUP2 MSTORE PUSH1 0x20 ADD PUSH32
0x48656C6C6F0000000000000000000000000000000000000000000000000000 DUP2 MSTORE
POP SWAP1 POP SWAP1 JUMP STOP LOG1 PUSH6 0x627A7A723058 KECCAK256 SLT 0xec 0xe
0xf5 0xf8 SLT 0xc7 0x2d STATICCALL ADDRESS SHR 0xdb COINBASE 0xb1 BALANCE 0xe8
0xf8 DUP14 0xda 0xad DUP13 LOG1 0x4c 0xb4 0x26 0xc2 DELEGATECALL PUSH7
0x8994D3E002900
```

Once deployed, your contract will have an Ethereum address like other accounts.

Remix (https://remix.ethereum.org) Remix IDE allows developing, deploying and administering smart contracts for Ethereum like blockchains. As June 2021, Remix web page is divided into four panels:



icon panel: at the left side, it shows the menu, with different activities that can be executed, like file management, compile solidity code or deploy Ethereum contract



Side panel: in the middle, it shows the details

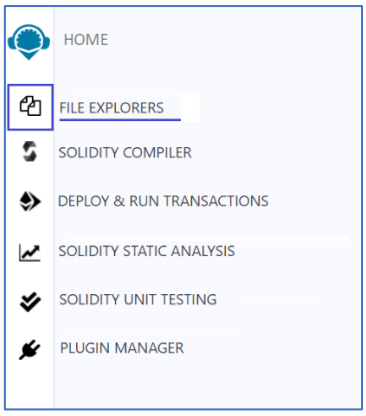
Main panel: at the right side, it is the main windows

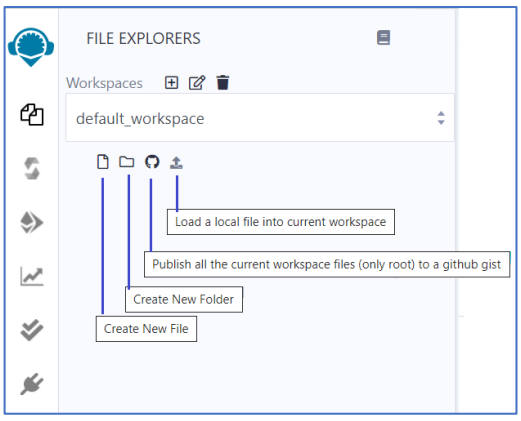
Terminal: at the bottom side, helps to see the result

STEP 1:

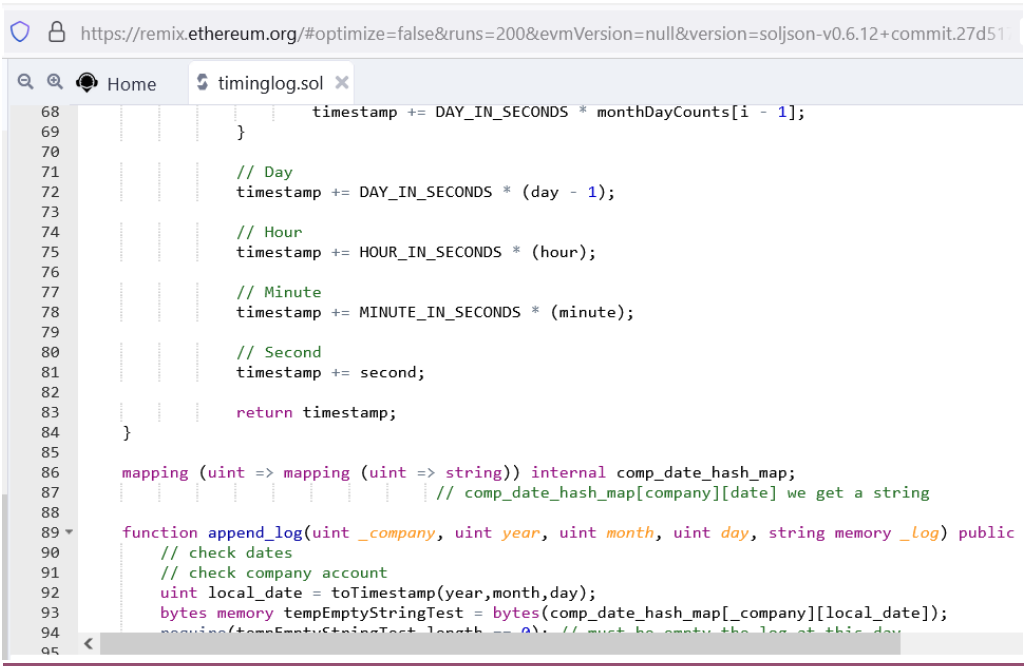
Create a new file with Solidity code. Click on icon panel, file explores icon (two sheets icon).

You can organize different projects in different folder. Click in the option you will open the smart contract: New File, New Folder, Publish to GitHub or Load File.



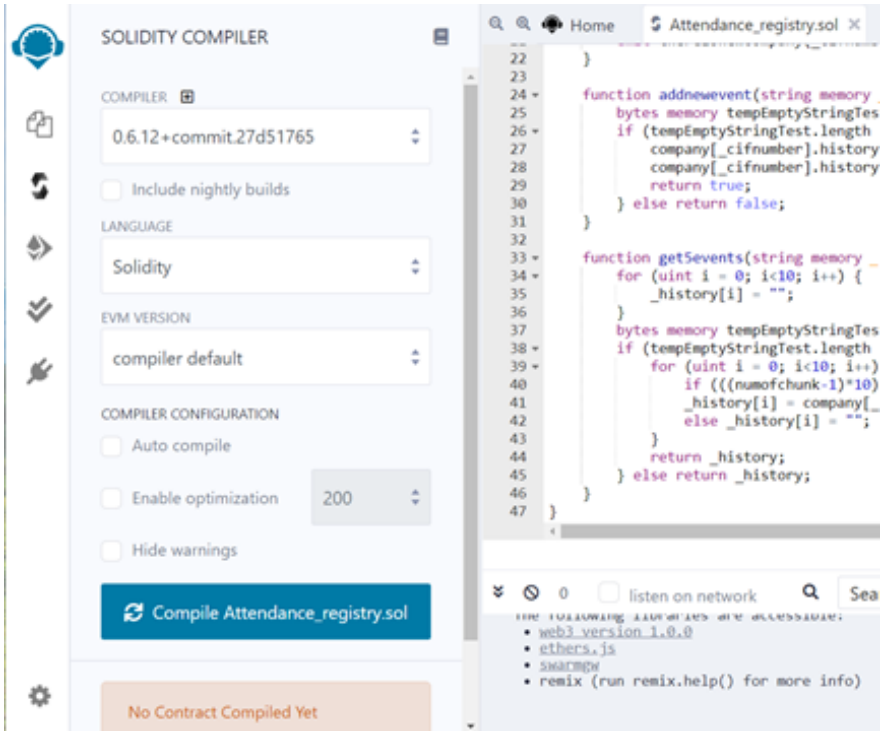


On main panel it will be show a new tab with the contract name, and several program lines

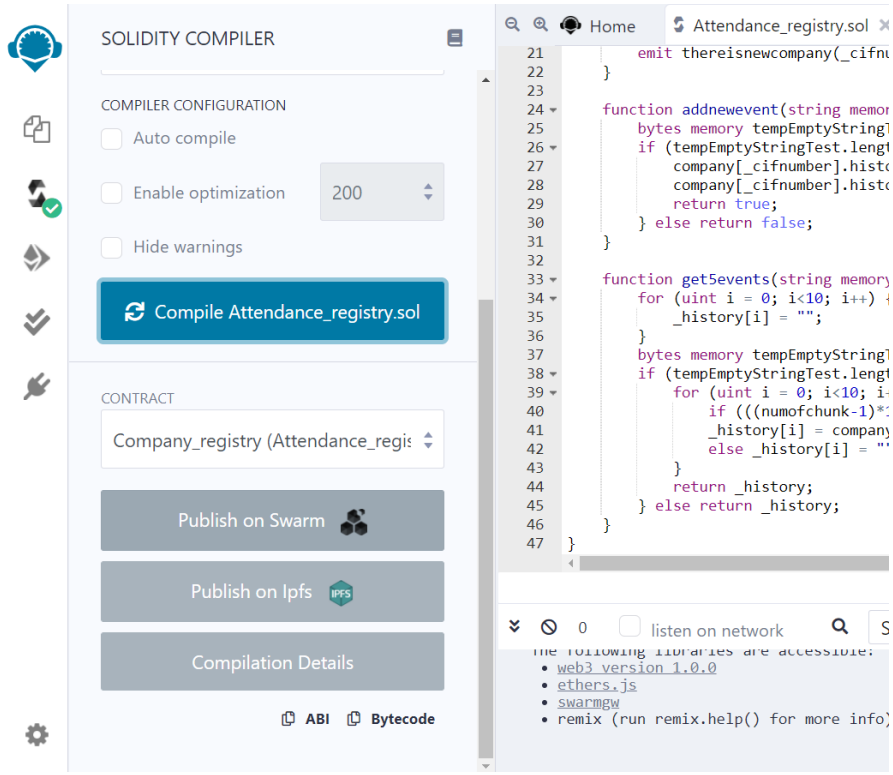


STEP 2:

Compile the contract. Click on the compile icon on the left hand side



If compile is successful, it showed a button for "compilation details" and a green tick on solidity button



Click on ABI, and save ABI code

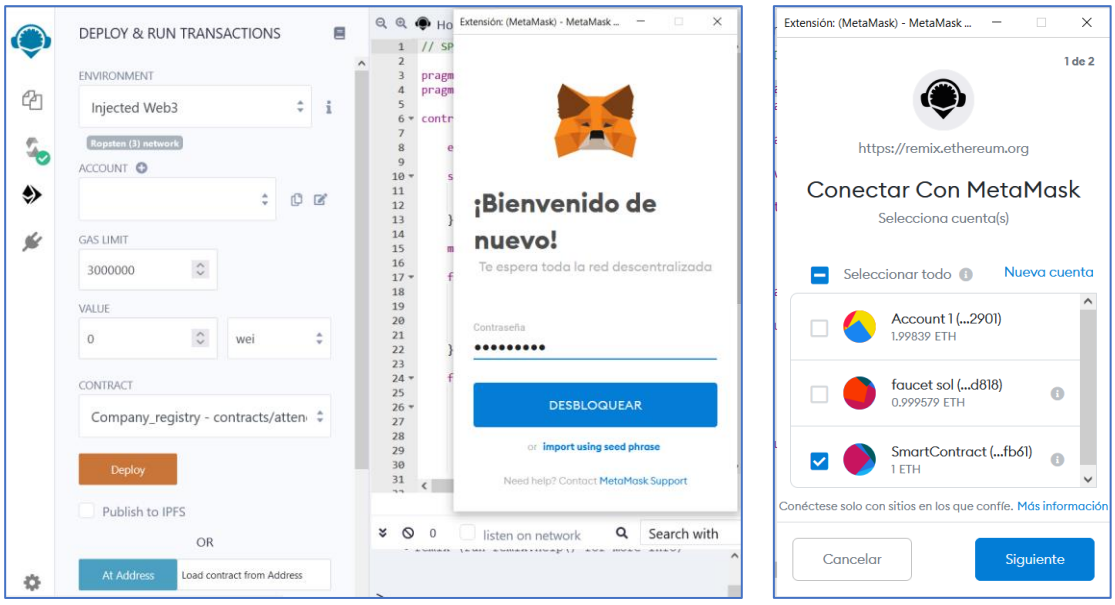

```
[
   {
          "inputs": [
                 {
                        "internalType": "uint256",
                        "name": "_company",
                        "type": "uint256"
                 },
                 {
                        "internalType": "uint256",
                        "name": "year",
                        "type": "uint256"
                 },
                 {
                        "internalType": "uint256",
                        "name": "month",
                        "type": "uint256"
                 },
                 {
                        "internalType": "uint256",
                        "name": "day",
                        "type": "uint256"
                 },
                 {
                        "internalType": "string",
                        "name": "_log",
                        "type": "string"
                 }
          ],
          "name": "append_log",
          "outputs": [
                 {
                        "internalType": "bool",
                        "name": "OK",
                        "type": "bool"
                 }
          ],
          "stateMutability": "nonpayable",
          "type": "function"
   },
   {
          "inputs": [
                 {
                        "internalType": "uint256",
                        "name": "_company",
                        "type": "uint256"
```

```
                },
                {
                        "internalType": "uint256",
                        "name": "year",
                        "type": "uint256"
                },
                {
                        "internalType": "uint256",
                        "name": "month",
                        "type": "uint256"
                },
                {
                        "internalType": "uint256",
                        "name": "day",
                        "type": "uint256"
                }
        ],
        "name": "get_log",
        "outputs": [
                {
                        "internalType": "string",
                        "name": "_log",
                        "type": "string"
                }
        ],
        "stateMutability": "view",
        "type": "function"
   }
]
```

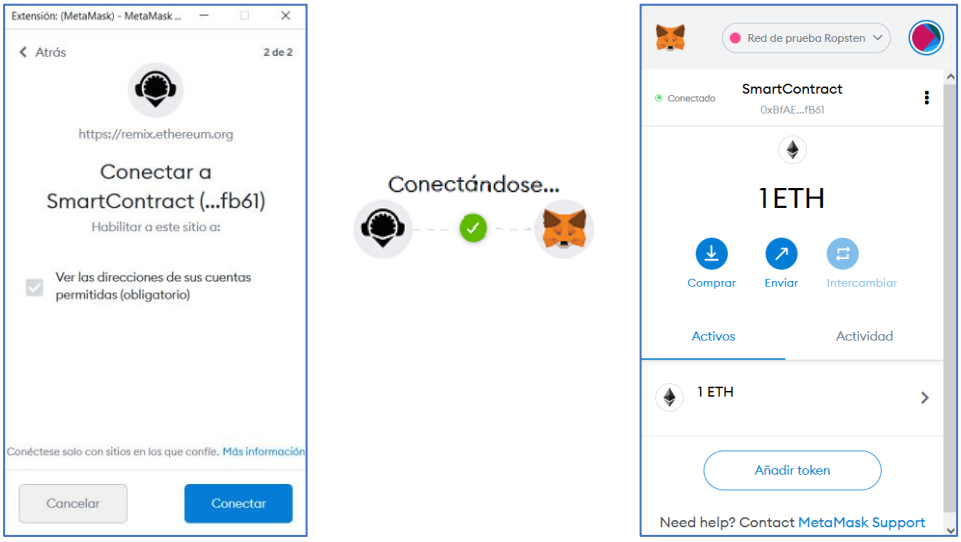
STEP 3:

Deploy contract on Ethereum. Click on Ethereum icon, select Injected Web3 environment. MetaMask plugin will be ask for password. Enter your credentials.

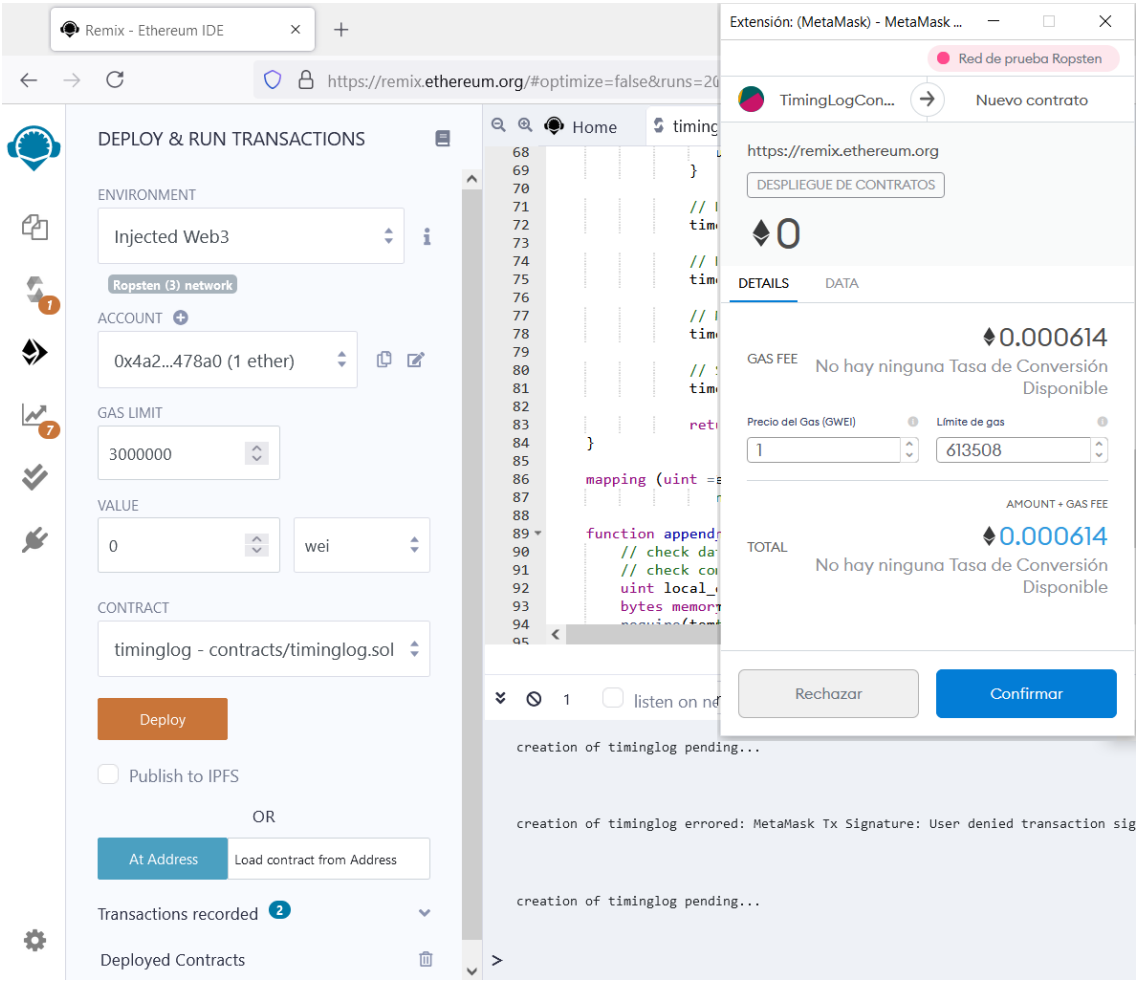
MetaMask will ask permit to connect to Remix.



Select which account will be connected to Remix, and what network is used (ropsten). To check, only click on MetaMask plugin logo



Deploy contract by click on "Deploy" button at Remix page. MetaMask will ask for Gas fee (1GWEI) and Gas Limit (613508), and show the price, in ETH. This is the price to deploy the contact. Click on "Confirm" button. MetaMask will request to miners to work with this contract. If there is not enough gas, the contract will not be deployed, but the computation cost will be charged to the account

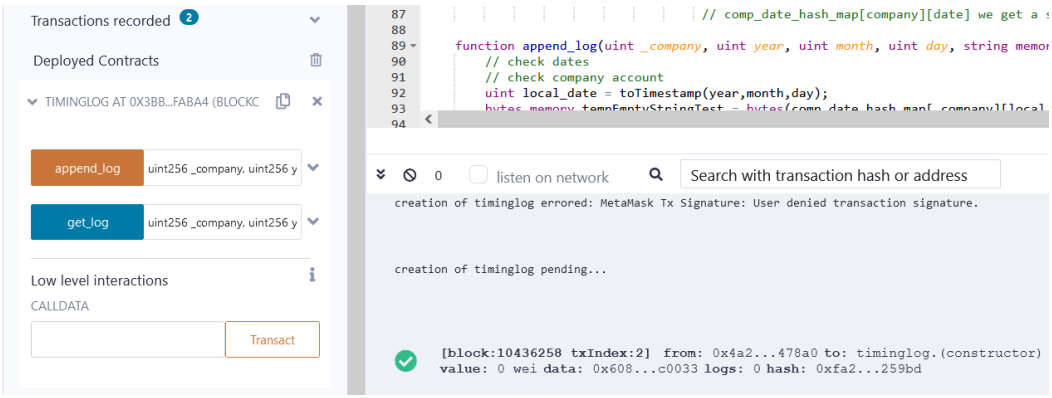


Wait some seconds, as the process need time, until see the confirmation message

STEP 4:

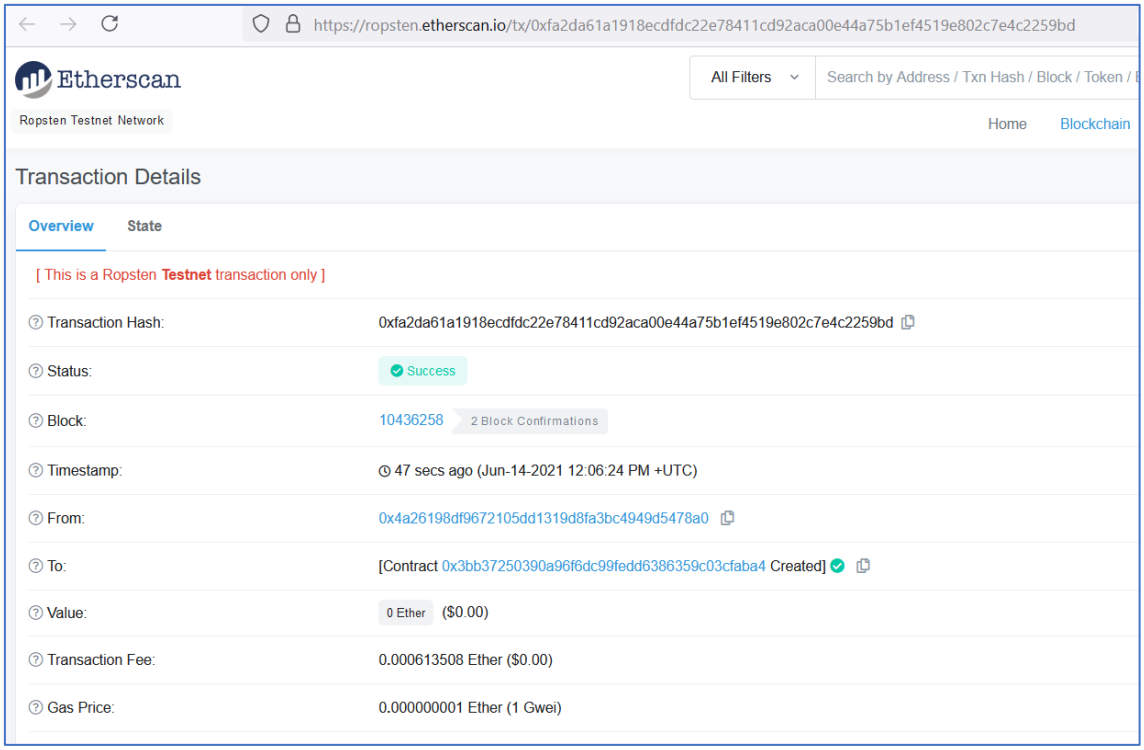
Once the blockchain is created, Remix will notify the transaction was done. To see details of transaction, see debug panel and click on hash.

Copy hash value and contract address



Contract address: 0x3bb37250390a96f6dc99fedd6386359c03cfaba4

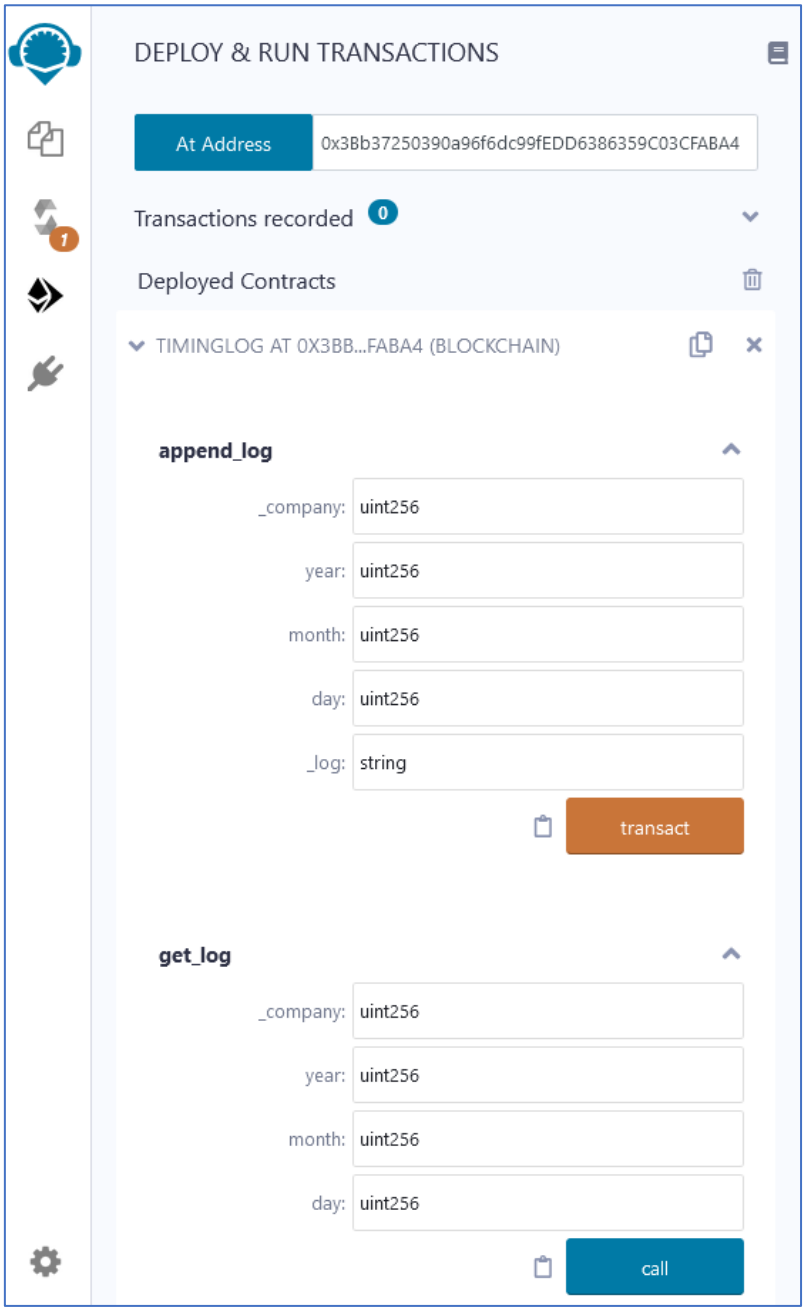
Transactions details available at:
https://ropsten.etherscan.io/tx/0xfa2da61a1918ecdfdc22e78411cd92aca00e44a75b1ef4519e802c7e4c2259bd



STEP 5:

To open any deployed contract, at any other computer, there is need the solidity code, ABI code and contract address.

Open Remix. Under file tab upload solidity code. Under compile tab click on compile. Go to Deploy tab, select "Injected Web3" environment. Connect to MetaMask. Instead click on Deploy (not needed because the contract is already deployed), write the address at the section "At Address". You will see the details of Smart Contract.



Annex F: DApp: index.html

```
<html>

<head>
  <meta charset="UTF-8" />
  <title>DAppTimeLog</title>
  <link rel="icon" type="image/svg" href="metamask-fox.svg" />

  <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=
swap" rel="stylesheet" />
  <link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.4.1/css/bootstrap.min.css" rel="stylesheet" />
  <link href="https://cdnjs.cloudflare.com/ajax/libs/mdbootstrap/4.14.1/css/mdb.min.c
ss" rel="stylesheet" />

  <link rel="stylesheet" href="metamask.css" />
</head>

<body>
  <main class="container-fluid">
    <header>
      <div id="logo-container">
        <h1 id="logo-text" class="text-center">
          Employee Attendance Management in Ethereum
        </h1>

        <img id="mm-logo" src="metamask-fox.svg">
      </div>
    </header>

    <!-- Part 1 Setting up Basic Actions and Status-->
    <section>
      <div class="row d-flex justify-content-center">
        <div class="col-xl-4 col-lg-6 col-md-12 col-sm-12 col-12">
          <div class="card">
            <div class="card-body">
              <h4 class="card-title">
                Basic Actions
              </h4>

              <button class="btn btn-primary btn-lg btn-block mb-
3" id="connectButton">
                --- Connect to MetaMask ---
              </button>

            </div>
          </div>
```

```
        </div>
      </div>
    </section> <br>
    <!-- /Part 1 -->

    <section>
      <div class="row d-flex justify-content-center">

        <!-- Append Log => add attendance registries-->
        <div class="col-xl-4 col-lg-6 col-md-12 col-sm-12 col-12 d-flex align-items-
stretch">
          <div class="card full-width">
            <div class="card-body">
              <h4 class="card-title">
                Store Attendance Report
              </h4>
              <label>Company CIF: </label>
              <input type="text" id="company_cif">
              </input><br>
              <label for="type_event">Day of record: </label>

              <input type="date" id="registry_date" value="2021-05-30" </input><br>

              <label>
                <br>File to upload: <br>
              </label><br>

              <input type="file" id="file_event"></input><br>
              <button class="btn btn-primary btn-lg btn-block mb-3" id="addEvent">
                Add Attendance Report
              </button>
            </div>
          </div>
        </div>

        <!-- Get log => Reading attendance registries -->
        <div class="col-xl-4 col-lg-6 col-md-12 col-sm-12 col-12 d-flex align-items-
stretch">
          <div class="card full-width">
            <div class="card-body">
              <h4 class="card-title">
                Read attendance report
              </h4>
              <label>Company CIF: </label>
              <input type="text" id="company_cif">
              </input><br>
              <label>Day of record: </label>

              <input type="date" id="registry_date" value="2021-05-30" </input><br>
```

```
97
98                <button class="btn btn-primary btn-lg btn-block mb-3" id="getFile">
99                  Get Attendance Report
100               </button>
101             </div>
102           </div>
103         </div>
104
105       </div>
106     </section>
107
108     <section>
109       <div id="contenedor">
110       </div>
111     </section> <br>
112
113   </main>
114
115   <script src="https://cdn.ethers.io/lib/ethers-
116 5.0.umd.min.js" type="application/JavaScript">
117   </script>
118   <script src="node_modules/@metamask/onboarding/dist/metamask-
119 onboarding.bundle.js" defer>
120   </script>
121   <script src="https://cdn.jsdelivr.net/npm/ipfs-http-client/dist/index.min.js">
122   </script>
123   <script src="contract.js" defer>
124   </script>
125
126 </body>
127
128 </html>
```

Annex G: DApp - metamask.css

```
section {
  margin: 20px 0 20px 0;
}

.info-text {
  font-size: 1.1em;
}

.full-width {
  width: 100%;
}

.card {
  margin-bottom: 20px;
}

/* Logo & Header */
header {
  display: flex;
  justify-content: center;
  align-items: center;
}

#logo-container {
  display: inline-flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  margin-bottom: 20px;
}

#logo-text {
  padding-top: 15px;
  padding-bottom: 5px;
}

#mm-logo {
  width: 20%;
}

/* Encrypt/Decrypt */

#encryptMessageInput {
  margin-bottom: 1rem;
}
```

Annex H: DApp - contract.js

```
/*
The `TimeLogContract` is compiled from:
  project folder (C:\TFM\solidity
    include
    *  TimeLogContract.sol
    *  ContractAccountAddress.txt
    *  ABI.txt
*/

// * ES6 module: `import MetamaskOnboarding from '@metamask/onboarding'`
// * ES5 module: `const MetamaskOnboarding = require('@metamask/onboarding')`
// import MetaMaskOnboarding from '@metamask/onboarding'

// Address for web server (nodejs) that contains all packages
const onboarding = new MetamaskOnboarding({ forwarderOrigin })

// Web Page Variables definitions

// Basic Actions Section
const onboardButton = document.getElementById('connectButton');
const accountsDiv = document.getElementById('accounts')

// Store Attendance Report Section
//const company_cif = document.getElementById('company_cif');
//const type_event = document.getElementById('type_event');
//const addEventButton = document.getElementById('addEvent');

// Read Attendance Section
//const company_cif         = document.getElementById('company_cif');
//const append_logButton = document.getElementById('append_log');

// Smart contract where is deployed TIMELOG contract
const ContractAccountAddress = '0x3Bb37250390a96f6dc99fEDD6386359C03CFABA4';

// Smart contract binary interface ABI
// deploy solidity contract and copy ABI result

const ABI = [
  {
    "inputs": [
      {
        "internalType": "uint256",
        "name": "_company",
        "type": "uint256"
      },
      {
```

```
        "internalType": "uint256",
        "name": "year",
        "type": "uint256"
      },
      {
        "internalType": "uint256",
        "name": "month",
        "type": "uint256"
      },
      {
        "internalType": "uint256",
        "name": "day",
        "type": "uint256"
      },
      {
        "internalType": "string",
        "name": "_log",
        "type": "string"
      }
    ],
    "name": "append_log",
    "outputs": [
      {
        "internalType": "bool",
        "name": "OK",
        "type": "bool"
      }
    ],
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [
      {
        "internalType": "uint256",
        "name": "_company",
        "type": "uint256"
      },
      {
        "internalType": "uint256",
        "name": "year",
        "type": "uint256"
      },
      {
        "internalType": "uint256",
        "name": "month",
        "type": "uint256"
      },
      {
```

```
        "internalType": "uint256",
        "name": "day",
        "type": "uint256"
      }
    ],
    "name": "get_log",
    "outputs": [
      {
        "internalType": "string",
        "name": "_log",
        "type": "string"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  }
]

/*
*********************************
     Connect to MetaMask
*********************************
*/

/*
const initialize = async () => {
    //You will start here

    let onboarding
    try {
        onboarding = new MetaMaskOnboarding({ forwarderOrigin })
    } catch (error) {
        console.error(error)
    }


    const onClickConnect = async () => {
        try {
            //Will open the MetaMask UI
            // You should disable this button while the request is pending!
            await ethereum.request({ method: 'eth_requestAccounts' });
            onboardButton.disabled = true;
        } catch (error) {
            console.error(error);
        }
    }


}
```

```
*/

const initialize = () => {
  //You will start here

  //This is to check if the brower has the Metamask extension installed
  const isMetaMaskInstalled = () => {
    const { ethereum } = window;
    return Boolean(ethereum && ethereum.isMetaMask);
  };

  // Show ether account in case of Metamask is connected
  let accounts

  const isMetaMaskConnected = () => accounts && accounts.length > 0

  //This will start the onboarding proccess, to MetaMask installation page
  const onClickInstall = () => {
    onboardButton.innerText = 'Onboarding in progress';
    onboardButton.disabled = true;
    onboarding.startOnboarding();
  };

  //Will open small windows: the MetaMask UI
  // You should disable this button while the request is pending!
  const onClickConnect = async () => {
    try {
      const newAccounts = await ethereum.request({
        method: 'eth_requestAccounts',
      })
      handleNewAccounts(newAccounts)
    } catch (error) {
      console.error(error)
    }
  }

  //Now we check to see if MetaMask is installed
  const MetamaskClientCheck = async () => {
    if (!isMetaMaskInstalled()) {
      //If it isn't installed we ask the user to click to install it
      onboardButton.innerText = 'Click here to install MetaMask!';
      onboardButton.onclick = onClickInstall;
      onboardButton.disabled = false;
    } else if (isMetaMaskConnected()) {
      //If it is installed we change our button text
      onboardButton.innerText = 'Connected'
      //When the button is clicked we call this function to connect the users MetaMas
k Wallet
      onboardButton.disabled = true
```

```
      if (onboarding) {
        onboarding.stopOnboarding()
      }
    } else {
      onboardButton.innerText = 'Connect'
      onboardButton.onclick = onClickConnect
      onboardButton.disabled = false
    }
  };

  MetamaskClientCheck();



  function handleNewAccounts (newAccounts) {
    accounts = newAccounts
    accountsDiv.innerHTML = accounts
    if (isMetaMaskConnected()) {
      initializeAccountButtons()
    }
    updateButtons()
  }

};









window.addEventListener('DOMContentLoaded', initialize); //Lanza la inicialización se
gún cargue toda la página HTML

//Se instancian el provider  y el "firmador" para poder trabajar con el contrato
//window.ethereum se refiere al wallet Metamask que es una extensión del navegador
const provider = new ethers.providers.Web3Provider(window.ethereum);
//El signer se obtiene para poder hacer métodos de alteración de estado (o transferen
cias)
const signer = provider.getSigner();

//Se instancia un objeto del contrato y se firma con signer
const myContract = new ethers.Contract(ContractAccountAddress, ABI, provider); //Prox
y del contrato en el usuario
const myContractWithSignature = myContract.connect(signer); //Proxy anterior con la f
uncionalidad de firmar
```

```
244
245 var fileData = ''; //Metainformación del archivo del usuario
246
247 //Se trata el fichero al pulsar en el elemento de explorador de archivos
248 const fileSelector = document.getElementById('file_event');
249 fileSelector.addEventListener('change', (evento) => {
250   fileData = evento.target.files[0];
251   console.log(fileData);
252 });
253
254 //Se instancia un objeto IPFS con un nodo de infura (es gratuito por ahora) y se pone
255  el puerto 5001
256 //La librería IpfsHttpClient ha sido previamente importado en el HTML
257 const ipfs = window.IpfsHttpClient({ host: 'ipfs.infura.io', port: 5001 })
258
259
```

Annex I: DApp - Debugging code in VSC

Create package.json file

A package.json file: lists the packages your project depends on. Also specifies versions of a package that your project can use using semantic versioning rules and makes your build reproducible, and therefore easier to share with other developers

In the project file, open CMD window and run "npm init" command. The program will ask about parameters and will create an updated version of package.json file

```
npm init
  package name: (dapptimelog)
  version: (1.0.0)
  description: Building DApp using MetaMask
  entry point: (contract.js)
  test command: npm run lint
  git repository:
  keywords:
  author: MetaMask
  license: (ISC) MIT
About to write to C:\asignaturas\DAppTimeLog\package.json:
```

After the operation was successful, it will be create a *package.json* file that will manage all configuration for our project. Next step is to modified section, adding this lines:

```
script:{
    "deploy": "./deploy.sh",
    "lint": "eslint . --ext js,json",
    "lint:fix": "eslint . --fix --ext js,json",
    "serve": "static-server . --port 9011",
  "test": "npm run lint"
}
```

add working files

```
  "files": [
    "/contract.js",
    "/index.html"
  ],
```

in this file are stored specific dependencies, that will be filled in the next steps

- "dependencies": Packages required by your application in production.
- "devDependencies": Packages that are only needed for local development and testing.

The difference between these two, is that *devDependencies* are modules which are only required during development, while *dependencies* are modules which are also required at runtime. To save a dependency as a *devDependency* on installation we need to do an npm install --save-dev , instead of just an npm install --save.

```
  "name": "metamask-simple-dapp",
  "version": "1.0.0",
  "description": "Building a simple dapp used in MetaMask e2e tests.",
```

Update version in package.json file, information available at https://www.npmjs.com/

```
    "@metamask/onboarding": "^0.2.1",
    "eth-sig-util": "^2.5.3",
    "ethers": "^5.0.21",
    "ipfs-core": "^0.8.0",
    "ipfs-http-client": "^48.1.1",
    "web3": "^1.3.0"
```

```
    "@metamask/onboarding": "^1.0.1",
    "eth-sig-util": "^3.0.1",
    "ethers": "^5.3.1",
    "ipfs-core": "^0.8.0",
    "ipfs-http-client": "^50.1.2",
    "web3": "^1.3.6"
```

Install Lintern

ESLint is a JavaScript tool that checks your code for potential errors and bad code practices. It helps you enforce a code standard and style guide in your codebase. You can add ESLint in any of your JavaScript Code. It's not only limited to React Projects. You can use it with Vue.js, Node.js, or even vanilla JavaScript Projects. It's pluggable and highly configurable. ESLint supports multiple plugins and extensions to extend and enhance its functionality.

[Documentation - ESLint - Pluggable JavaScript linter https://eslint.org/docs/user-guide/formatters/]

ESLINT is pluggable linting utility for JavaScript and JSX, it help to discover possible errors.

ESLint is a *linter* -a tool that analyzes your code and flags potential errors. This is very helpful to avoid common mistakes that are made while you're coding (using undefined variables, syntactic errors, etc).

lintern https://eslint.org/docs/user-guide/getting-started

Steps to install ESLINT: install and init

```
npm install eslint --save-dev
```

On new folders you might also need to create a .eslintrc configuration file.

You should then set up a configuration file, and the easiest way to do that is to use the --init flag. Note: --init assumes you have a package.json file already. If you don't, make sure to run npm init

```
$ npx eslint --init
```

it will be opened severals questions:

check syntax and problems:

```
√ How would you like to use ESLint? · problems
√ What type of modules does your project use? · none
√ Which framework does your project use? · none
√ Does your project use TypeScript? · No / Yes
√ Where does your code run? · browser
√ What format do you want your config file to be in? · JavaScript
```

```
Successfully created .eslintrc.js file in C:\asignaturas\DAppTimeLog
```

Add Lintern as a extension of Visual Studio Code

VS Code is one of the top editor for development it was developed and maintain by Microsoft it help to improve productivity and also comes with many features

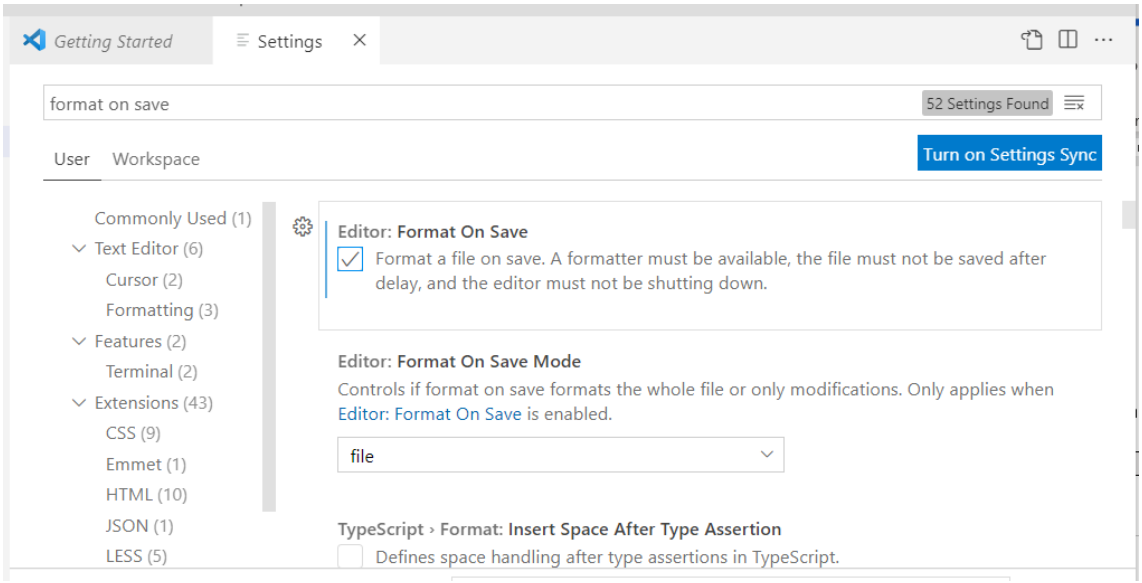
let try to install ESLINT extension on vs code editor

extension ESlint

view - extension

((this extension is enable globally))

succeeded



Run all programs.

Every time npm is executed, NPM generates a ‘package-lock.json’ file. The package-lock.json file is a little more complex due to a trade-off between determinism and simplicity. Due to this complexity, the package-lock will generate the same node_modules folder for different npm versions. Every dependency will have an exact version number associated with it in the package-lock file.

1.- Open cdm terminal. To install all dependencies needed for the project, run "npm install" command. Once the installation is finished, it will be created a new folder named “node_modules” with several files inside

```
> npm install
```

2.- *save ether* : This package is used to make transactions and make calls to the contract deployed in Ethereum. It will be necessary to open a command console in the project directory and write:

```
> npm install --save ethers
```

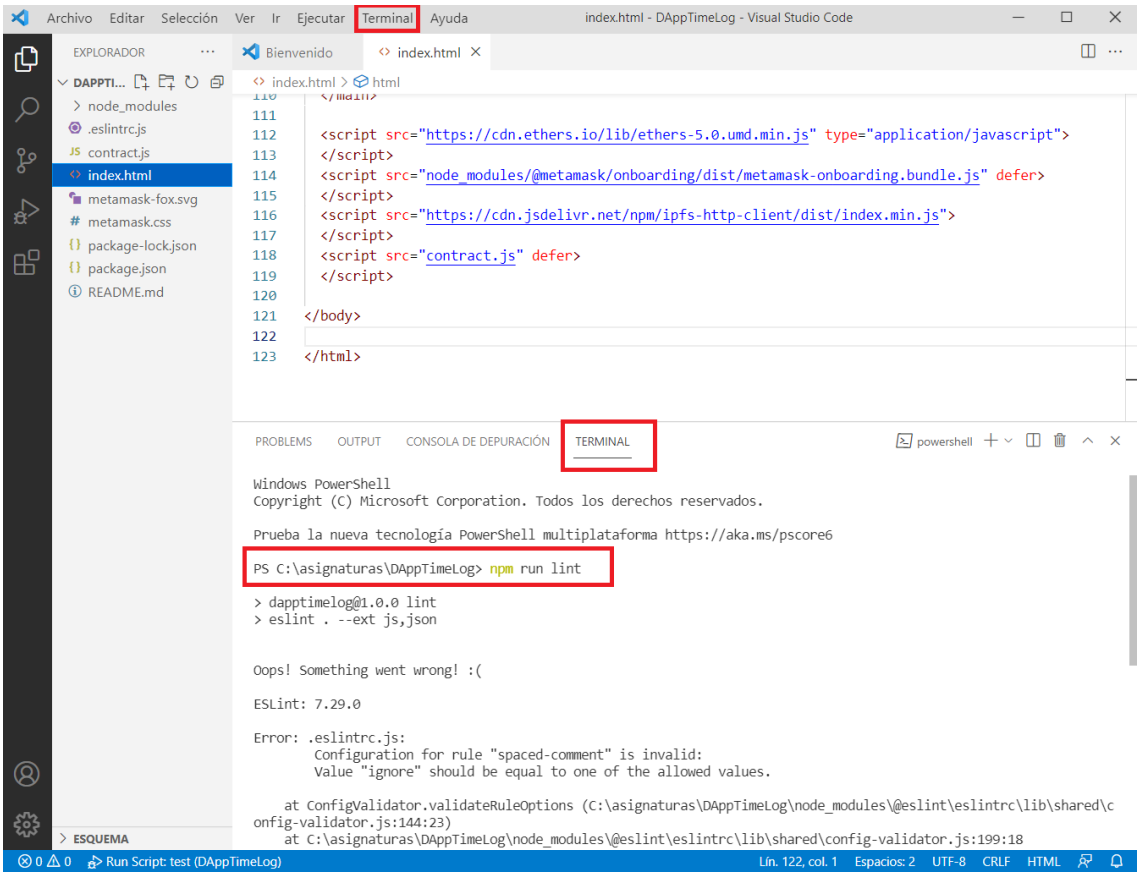
3.- *install ipfs* : This package is the one that will upload to the Inter Planetary File System the file that the client uploads as a vehicle event. A command console is opened in the project and typed:

```
> npm install ipfs-core
```

4.- on console or using terminal inside VSC, run debugger to check sintax and errors.

- file - Open folder (C:/asignaturas/DAppTimeLog
- open index.html file
- open console terminal
- in terminal write the command

```
> npm run lint
```



see console: get code suggestions, error checking, and warning highlights

Fixing error

the result for debug was as follow:

```
C:\asignaturas\DAppTimeLog\contract.js
   39:7   error  'ContractAccountAddress' is assigned a value but
never used  no-unused-vars
  147:1   error  Mixed spaces and tabs        no-mixed-spaces-and-tabs
  148:13  error  'ethereum' is not defined      no-undef
  161:14  error  'isMetaMaskConnected' is not defined       no-undef
  167:1   error  Mixed spaces and tabs      no-mixed-spaces-and-tabs
  184:22  error  'ethers' is not defined     no-undef
  189:24  error  'ethers' is not defined        no-undef
  189:40  error  'contract' is not defined       no-undef
```

```
  223:1   error  Mixed spaces and tabs        no-mixed-spaces-and-tabs
  224:1   error  Mixed spaces and tabs    no-mixed-spaces-and-tabs
  225:1   error  Mixed spaces and tabs    no-mixed-spaces-and-tabs
  226:1   error  Mixed spaces and tabs     no-mixed-spaces-and-tabs
  227:1   error  Mixed spaces and tabs      no-mixed-spaces-and-tabs
  228:1   error  Mixed spaces and tabs     no-mixed-spaces-and-tabs
  229:1   error  Mixed spaces and tabs        no-mixed-spaces-and-tabs
  230:1   error  Mixed spaces and tabs        no-mixed-spaces-and-tabs
  231:1   error  Mixed spaces and tabs      no-mixed-spaces-and-tabs
  232:1   error  Mixed spaces and tabs       no-mixed-spaces-and-tabs
  233:1   error  Mixed spaces and tabs       no-mixed-spaces-and-tabs
  248:1   error  Mixed spaces and tabs       no-mixed-spaces-and-tabs
  249:1   error  Mixed spaces and tabs      no-mixed-spaces-and-tabs
  249:21  error  'chunk_read' is not defined          no-undef
  250:1   error  Mixed spaces and tabs      no-mixed-spaces-and-tabs
  251:1   error  Mixed spaces and tabs       no-mixed-spaces-and-tabs
  252:1   error  Mixed spaces and tabs        no-mixed-spaces-and-tabs
  253:1   error  Mixed spaces and tabs       no-mixed-spaces-and-tabs
  254:1   error  Mixed spaces and tabs       no-mixed-spaces-and-tabs
  255:1   error  Mixed spaces and tabs      no-mixed-spaces-and-tabs
  256:1   error  Mixed spaces and tabs     no-mixed-spaces-and-tabs
  257:1   error  Mixed spaces and tabs       no-mixed-spaces-and-tabs
  258:1   error  Mixed spaces and tabs no-mixed-spaces-and-tabs
  259:1   error  Mixed spaces and tabs  no-mixed-spaces-and-tabs
  260:1   error  Mixed spaces and tabs     no-mixed-spaces-and-tabs
```

as we already has the old version of package.json and .eslintcr.js, it can be added lines, with caution, because it can appears 500 errors instead of 500 warnings. The first errors was related to indent, because in some parts of the program it was used two spaces, in anothers 4 spaces and others use tab.

1) write indent conditions in .eslint

```
"rules": {
    "indent": [1, 2, {"SwitchCase": 1}]
```

```
}
```

first numbers refers how to mark this error:

"off" or 0 - turn the rule off

"warn" or 1 - turn the rule on as a warning (doesn’t affect exit code)

"error" or 2 - turn the rule on as an error (exit code is 1 when triggered)

According to documentation, second '2' is for number of spaces used for indent (can be set to 4 or 6 or "tab").

2) add missing dependencies, if it is showed errors:

ESLint couldn't find the config "@metamask/eslint-config" to extend from.

solution

```
npm install @metamask/eslint-config
```

Could not find a declaration file for module '@metamask/onboarding'.

```
npm i --save-dev @types/metamask__onboarding
```

3) do it iterative until no error appears