ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ψηφιακή Αποθήκευση Βίντεο και Ασφαλής Μετάδοση με Χρήση ESP32

Φίλιππος Μπαραδάκης
161023

Επιβλέπων καθηγητής: Ιωάννης Βογιατζής

Διπλωματική εργασία υποβληθείσα στο Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

ΑΙΓΑΛΕΩ, ΟΚΤΩΒΡΙΟΣ 2022

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

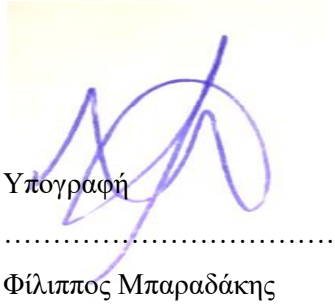
Ο κάτωθι υπογεγραμμένος Φίλιππος Μπαραδάκης του Κωνσταντίνου, με αριθμό μητρώου 711161023 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών:

ΦΙΛΙΠΠΟΣ ΜΠΑΡΑΔΑΚΗΣ



Υπογραφή

……

Φίλιππος Μπαραδάκης

Προπτυχιακός Φοιτητής Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών ΠΑΔΑ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ και ΥΠΟΛΟΓΙΣΤΩΝ

Η παρούσα διπλωματική εργασία παρουσιάστηκε

από τον Φίλιππο Μπαραδάκη (161023) στις 13/10/2022

Επιβλέπων καθηγητής:

Ιωάννης Βογιατζής

Ioannis Voyiatzis

Digitally signed by Ioannis Voyiatzis
Date: 2022.10.31 21:34:00 +02'00'

Μέλος επιτροπής:

Παναγιώτης Γιαννακόπουλος

Panagiotis Giannakopoulos

Digitally signed by Panagiotis Giannakopoulos
Date: 2022.10.31 21:45:19 +02'00'

Μέλος επιτροπής:

Χρήστος Τρούσσας

Christos Troussas

Digitally signed by Christos Troussas
Date: 2022.10.31 21:51:30 +02'00'

ΠΕΡΙΛΗΨΗ

Η ιλιγγιώδης εξέλιξη της τεχνολογίας έχει ως αποτέλεσμα την εύκολη πρόσβαση σε τεχνολογίες που στο παρελθόν ήταν διαθέσιμες μόνο σε κυβερνητικές οργανώσεις και μεγάλες εταιρίες. Στις μέρες μας, η χρήση κάμερας καλύπτει μια μεγάλη γκάμα μη στρατιωτικών εφαρμογών, όπως επιθεώρηση κατασκευών και υποδομών, παρακολούθηση οδικής κυκλοφορίας σε πραγματικό χρόνο, επιχειρήσεις έρευνας και διάσωσης, εντοπισμό και αναγνώριση προσώπων καθώς και απομακρυσμένη παρακολούθηση.

Ένας τομέας χρήσης της είναι τα μη επανδρωμένα αεροσκάφη (Unmanned Aerial Vehicles – UAVs), η όπως αποκαλούνται στις μέρες μας "drones". Η χρήση βίντεο πραγματικού χρόνου μπορεί να αποδειχτεί ιδιαίτερα χρήσιμη σε καταστάσεις που απαιτούν άμεση λήψη αποφάσεων σε πραγματικό χρόνο. Με την κάμερα, ο χειριστής μπορεί να ελέγχει πιο εύκολα και με ασφάλεια το drone, καθώς και να εκτελεί πληθώρα εργασιών μέσω αυτής.

Πέραν όμως της ραγδαίας εξέλιξης αυτών των εφαρμογών, το ηλεκτρονικό έγκλημα είναι ένα φαινόμενο που και αυτό εξελίσσεται με ραγδαίους ρυθμούς και συμβαδίζει με αυτούς της ανάπτυξης της τεχνολογίας. Υπάρχει επομένως αυξημένος κίνδυνος κάθε είδους απάτης, από καλούς γνώστες της τεχνολογίας, οι οποίοι προσπαθούν να πραγματοποιήσουν παράνομες πράξεις, μετατρέποντας την τεχνολογία σε ένα καταστροφικό όπλο. Δυστυχώς, οι δυνατότητες δίωξης τους από τις αρχές είναι περιορισμένες λόγω έλλειψης εκπαίδευσης και εμπειρίας με τις νέες τεχνολογίες που αναπτύσσονται συνεχώς αλλά και ασάφειας σχετικά με την νομοθεσία περί αυτών των θεμάτων.

Στην περίπτωση μας, άτομα με κακόβουλες προθέσεις μπορούν να υποκλέψουν σημαντικά αρχεία βίντεο, και να τα χρησιμοποιήσουν για παράνομους σκοπούς. Τα αρχεία αυτά είναι προσωπικά δεδομένα και πρέπει σε κάθε περίπτωση τα δεδομένα που μεταφέρονται να είναι ασφαλή και μη ανακτήσιμα από κακόβουλα άτομα.

Στην συγκεκριμένη εργασία αναπτύσσεται μια εφαρμογή για ESP32-CAM όπου η χρήση της αφορά μη επανδρωμένα αεροσκάφη αλλά και πληθώρα άλλων εφαρμογών που απαιτείται κάμερα (CCTV Camera). Σκοπός της εφαρμογής είναι η ασφαλής καταγραφή και μετάδοση βίντεο από ένα ESP32-CAM σε έναν ηλεκτρονικό υπολογιστή. Παρακάτω θα γίνει παρουσίαση του τρόπου καταγραφής βίντεο, κρυπτογράφησης, καθώς και αποθήκευσης.

Λέξεις-Κλειδιά:

ESP32, Μη Επανδρωμένα Εναέρια Οχήματα, Ασφαλής Μετάδοση Βίντεο, Παρακολούθηση σε Πραγματικό Χρόνο

ABSTRACT

The dizzying evolution of technology has resulted in easy access to technologies that were previously only available to government organizations and large companies. Nowadays, the use of the camera covers a wide range of civilian applications, such as viewing structures and infrastructure, real-time traffic monitoring, search and rescue operations, face tracking and remote surveillance.

One area of use is unmanned aerial vehicles (UAVs), or as they are called today "drones". Using real-time video can be especially useful in situations that require immediate real-time decision making. With the camera, the pilot can control the drone more easily and safely, as well as perform a variety of tasks through it.

However, apart from the rapid development of these applications, cybercrime is a phenomenon that is also evolving rapidly and keeps pace with the development of technology. There is therefore an increased risk of any kind of fraud, by those who are well versed in technology, who try to carry out illegal acts, turning technology into a destructive weapon. Unfortunately, the possibilities of prosecuting them by the authorities are limited due to lack of training and experience with the new technologies that are constantly evolving but also ambiguity regarding the legislation on these issues.

In our case, people with malicious intent can submit to important videos, and be used for illegal, unlawful purposes. These files are personal data and in any case, their transmission must be created in such a way that the data transferred is secure and not retrieved by malicious individuals.

In this work, an application for ESP32-CAM is developed where its use concerns unmanned aircraft and a variety of other applications that require a camera (CCTV Camera). The purpose of the application is to securely record and transmit video from an ESP32-CAM to a computer. Below will be a presentation of how to record video, encrypt and decrypt, as well as save it.

Keywords

ESP32, Unmanned Aerial Vehicles, Secure Video Transmission, Real-Time Monitoring.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ & ΑΚΡΩΝΥΜΑ

Transmission Control Protocol	TCP
User Datagram Protocol	UDP
Synchronize Sequence Number	SYN
Acknowledgement	ACK
Certificate Authority	CA
HyperText Transfer Protocol	HTTP
HyperText Transfer Protocol Secure	HTTPS
Internet of Things	IoT
Universal Asynchronous Receiver Transmitter	UART
Serial Peripheral Interface	SPI
Inter Integrated Circuit	I2C
Bluetooth Low Energy	BLE
Megabyte	MB
Kilobyte	KB
Pseudo Static Random Access Memory	PSRAM
Light Emitting Diode	LED
General Purpose Input Output	GPIO
Firmware Over The Air	FOTA
Common Collector Voltage	VCC
Ground	GND
Future Technology Devices International Limited	FTDI
Universal Serial Bus	USB
Milliampere Hour	MaH
Uniform Resource Locator	URL
Integrated Development Environment	IDE
Video Graphics Array	VGA
Extensible Markup Language	XML
Wireless Fidelity	Wi-Fi
Open Broadcaster Software	OBS
Matroska Video Files	MKV

Closed Circuit Television	CCTV

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω ιδιαίτερα το κύριο Ιωάννη Βογιατζή για την ευκαιρία που μου έδωσε στο να μπορέσω να εμβαθύνω στο θέμα ασφαλούς καταγραφής και αποθήκευσης βίντεο με χρήση ESP32. Επίσης ευχαριστώ τον κύριο Δημήτριο Ψιλιά για την πολύτιμη βοήθεια που μου πρόσφερε κατά την διάρκεια της διπλωματικής μου εργασίας. Τέλος, θέλω να ευχαριστώ την οικογένεια μου και όλους όσους με στήριξαν και ήταν δίπλα μου όλα αυτά τα δημιουργικά χρόνια φοίτησης μου στο τμήμα.

1. ΕΙΣΑΓΩΓΗ

1.1 ΑΝΤΙΚΕΙΜΕΝΟ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Παρά την συνεχόμενη χρήση κάμερας σε μη επανδρωμένα εναέρια οχήματα, η αβεβαιότητα όσον αφορά την ασφάλεια του βίντεο συνεχίζει να υφίσταται. Αρκετές φορές, η ασφαλής μετάδοση των στοιχείων είναι απαραίτητη ώστε να έρθει σε πέρας κάθε αποστολή που δύναται να εκτελέσει ο χρήστης. Σε πολλές περιπτώσεις, τα στοιχεία που συλλέγει είναι απαραίτητο να είναι κρυπτογραφημένα και να μην μπορούν να κλαπούν, για λόγους προσωπικών δεδομένων (GDPR), και όχι μόνο.

Διαθέτοντας ως εξοπλισμό ένα ESP32-CAM, εφοδιασμένο με κάμερα OV2640 και μια παροχή συνεχούς ρεύματος (DC), θα δείξουμε πως το βίντεο που θα λάβουμε από αυτό μπορεί με ασφαλή τρόπο να μεταφερθεί από την κάμερα σε έναν server, και με αυτό τον τρόπο ο χρήστης να δει το video χωρίς να υπάρχει κίνδυνος απώλειας δεδομένων από κακόβουλα άτομα.

Η προσπάθεια που γίνεται είναι, ο συγκεκριμένος μικροελεγκτής να αποστείλει βίντεο σε έναν server με ασφαλή τρόπο, έτσι ώστε το βίντεο αυτό να είναι μη ανακτήσιμο από μη εξουσιοδοτημένα άτομα.

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής διαθέτει κώδικα C++, JavaScript, HTML, CSS, έχει υλοποιηθεί σε σύστημα Windows και η χρήση της αφορά κυρίως μη επανδρωμένα αεροσκάφη “drones”.

Μπορεί να χρησιμοποιηθεί για οποιαδήποτε άλλη ενέργεια απαιτεί ασφαλή καταγραφή video.

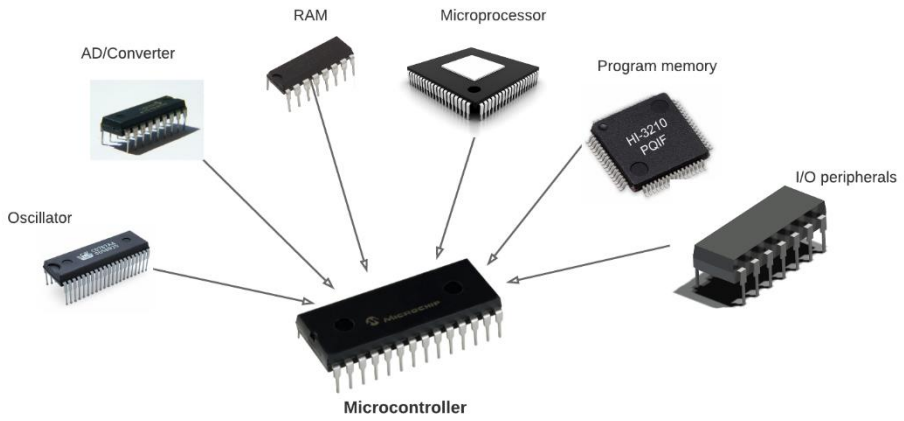
1.2 ΤΙ ΕΙΝΑΙ ΚΑΙ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΟΥΝΤΑΙ ΟΙ ΜΙΚΡΟΕΛΕΓΚΤΕΣ

Ένας μικροελεγκτής είναι ένα συμπαγές ολοκληρωμένο κύκλωμα (IC), σχεδιασμένο να εκτελεί μια συγκεκριμένη λειτουργεία σε ένα ολοκληρωμένο σύστημα. Ένας τυπικός μικροελεγκτής περιλαμβάνει επεξεργαστή, μνήμη, και περιφερειακά εισόδου/εξόδου (I/O), ενσωματωμένα σε ένα ενιαίο τσιπ (chip).

Οι μικροελεγκτές χρησιμοποιούνται σε ένα ευρύ φάσμα συστημάτων. Για παράδειγμα ένα αυτοκίνητο μπορεί να έχει πολλούς μικροελεγκτές που ελέγχουν διάφορα μεμονωμένα συστήματα μέσα, όπως το σύστημα αντί-μπλοκαρίσματος πέδησης (ABS), το σύστημα ελέγχου πρόσφυσης (TRC) και το σύστημα ψεκασμού καυσίμου. Υπάρχουν περιπτώσεις που μικροελεγκτές επικοινωνούν μεταξύ τους για να εκτελεστεί μια λειτουργία και να γίνουν συγχρονισμένα σωστές ενέργειες. Στέλνουν και λαμβάνουν

δεδομένα χρησιμοποιώντας τα περιφερειακά I/O και τα επεξεργάζονται για να εκτελέσουν τις απαιτούμενες ενέργειες που έχουν φτιαχτεί να κάνουν. Η επικοινωνία με το εσωτερικό του μικροελεγκτή γίνεται μέσα από τους ακροδέκτες (pins).

Υπάρχουν αρκετοί διαφορετικοί κατασκευαστές που φτιάχνουν διάφορους τύπους μικροελεγκτών. Συνήθως, ένας κατασκευαστής φτιάχνει μια οικογένεια μικροελεγκτών η οποία έχει ως βάση έναν κοινό πυρήνα, που συνδυάζεται με διαφορετικά περιφερειακά εισόδου/εξόδου (I/O) και/η διαφορετικό μέγεθος μνήμης.



Εικόνα 1.2.1. Παράδειγμα μικροελεγκτή

1.3 NODE.JS

Το Node.js είναι ένα “cross platform runtime” περιβάλλον που αναπτύχθηκε αρχικά το 2009 από τον Ryan Dahl για την ανάπτυξη εφαρμογών από την πλευρά του server. Μπορεί να θεωρηθεί ως ένας JavaScript server. Δημιουργήθηκε για να αντιμετωπιστούν τα προβλήματα απόδοσης χρόνου και επικοινωνίας δικτύου. Είναι μια πλατφόρμα που βασίζεται στην εκτέλεση JavaScript στον Chrome για εύκολη δημιουργία διαδικτυακών εφαρμογών. Χρησιμοποιεί ένα μοντέλο εισόδου-εξόδου βασιζόμενο σε συμβάντα (event-driven) χωρίς αποκλεισμό (non-blocking) που το κάνει "ελαφρύ" και αποτελεσματικό, ιδανικό για εφαρμογές που πραγματοποιούνται σε πραγματικό χρόνο.

Έχει γίνει δημοφιλές καθώς μέσω αυτού, μπορούν να δημιουργηθούν εφαρμογές υψηλής απόδοσης σε πραγματικό χρόνο. Ένα ωφέλιμο χαρακτηριστικό του είναι ότι επιτρέπει την χρήση JavaScript τόσο στον client όσο και στον server. Αρχικά, η java εκτελούνταν μόνο στον web browser, αλλά η ζήτηση της

ήταν μεγάλη και την έφερε στην μεριά του server. Η javascript έχει αναπτυχθεί πάρα πολύ και έχει κυριαρχήσει σε server scripts.

Σε αυτή την διπλωματική εργασία θα χρησιμοποιήσουμε το Node.js για να δημιουργήσουμε έναν ασφαλή Server που θα μπορεί να συνδεθεί το ESP32-CAM για να στείλει το βίντεο.

Εικόνα 1.3.1. Λογότυπο Node.js (https://nodejs.org/en/)

1.4 ΔΙΑΡΘΡΩΣΗ ΔΙΠΛΩΜΑΤΙΚΗΣ

Η διάρθρωση της παρούσας εργασίας καταγράφεται παρακάτω:

Στο 2ο κεφάλαιο γίνεται παρουσίαση της κρυπτογράφησης, εισαγωγή σε αυτήν καθώς και επεξήγηση των κρυπτογραφικών μεθόδων που χρησιμοποιούνται στην εργασία. Αναφέρονται τα βασικά χαρακτηριστικά, ο τρόπος λειτουργίας και το θεωρητικό υπόβαθρο της. Στην συνέχεια γίνεται ανάλυση των πρωτοκόλλων που χρησιμοποιούνται αλλά και της αρχής εκδόσεως πιστοποιητικών.

Στο 3ο κεφάλαιο παρουσιάζεται το σύστημα υλικού που χρησιμοποιείται. Γίνεται αναφορά στα περιφερειακά και μέτρηση συνολικού κόστους υλοποίησης.

Στο 4ο κεφάλαιο γίνεται η περιγραφή υλοποίησης έτσι όπως πραγματοποιήθηκε. Παρουσιάζονται τα γραφικά περιβάλλοντα που γράφτηκε ο κώδικας και αναλύεται ο τρόπος λειτουργίας τους. Επίσης αναλύεται ο κώδικας του server, του μικροελεγκτή, αλλά και ο κώδικας HTML+CSS που έχει προστεθεί. Τέλος, σε αυτό το κεφάλαιο δίνονται οδηγίες σύνδεσης του server, και μεταφοράς κώδικα από τον υπολογιστή, στον μικροελεγκτή.

Στο 5ο κεφάλαιο, παρουσιάζεται το πειραματικό μέρος της εργασίας. Αρχικά παρουσιάζεται ο τρόπος σύνδεσης και εκκίνησης της υλοποίησης, και στην συνέχεια ο τρόπος λειτουργίας της, καθώς ενδεικτικές δοκιμές σε αυτήν.

Στο 6ο κεφάλαιο γίνονται πρακτικές μετρήσεις που αφορούν την εμβέλεια, την θερμοκρασία, κατανάλωση και το βάρος.

Τέλος, στο 7ο κεφάλαιο παρουσιάζονται τα συμπεράσματα και οι μελλοντικές επεκτάσεις της υλοποίησης.

2. ΚΡΥΠΤΟΓΡΑΦΗΣΗ

2.1 ΕΙΣΑΓΩΓΗ

Με τον όρο κρυπτογράφηση, εννοούμε την διαδικασία κωδικοποίησης πληροφοριών. Αυτή η διαδικασία μετατρέπει την αρχική αναπαράσταση μιας πληροφορίας, σε μία εναλλακτική μορφή. Η αρχική μορφή ορίζεται ως κείμενο, ενώ η κρυπτογραφημένη ως κρυπτοκείμενο. Στην βέλτιστη περίπτωση, μόνο εξουσιοδοτημένα μέρη μπορούν να αποκρυπτογραφήσουν ένα κρυπτοκείμενο ώστε να έχουν πρόσβαση στην αρχική του μορφή. Η κρυπτογράφηση δεν αποτρέπει από μόνη της επιθέσεις, αλλά αποκρύπτει το κατανοητό περιεχόμενο που χρειάζεται προστασία, από διάφορους εισβολείς που θέλουν να το υποκλέψουν. Το βασικό χαρακτηριστικό για μία επιτυχημένη κρυπτογράφηση είναι η κατανόηση των κρυπτογραφικών τεχνικών και λειτουργειών που έχουν αναπτυχθεί. Για να μπορέσουμε να εμβαθύνουμε, θα πρέπει να κατανοήσουμε τις βασικές έννοιες κρυπτογράφησης.

Η κρυπτογράφηση χρησιμοποιείται για την διασφάλιση της προστασίας δεδομένων από την έκθεση σε μη εξουσιοδοτημένα άτομα, χρησιμοποιώντας διάφορους τρόπους και μεθόδους. Τα κύρια χαρακτηριστικά της είναι η εμπιστευτικότητα (confidentiality), ακεραιότητα (integrity) και διαθεσιμότητα (availability), που είναι και γνωστά ως CIA τριάδα (CIA Triad).

2.1.1 ΕΜΠΙΣΤΕΥΤΙΚΟΤΗΤΑ (CONFIDENTIALITY)

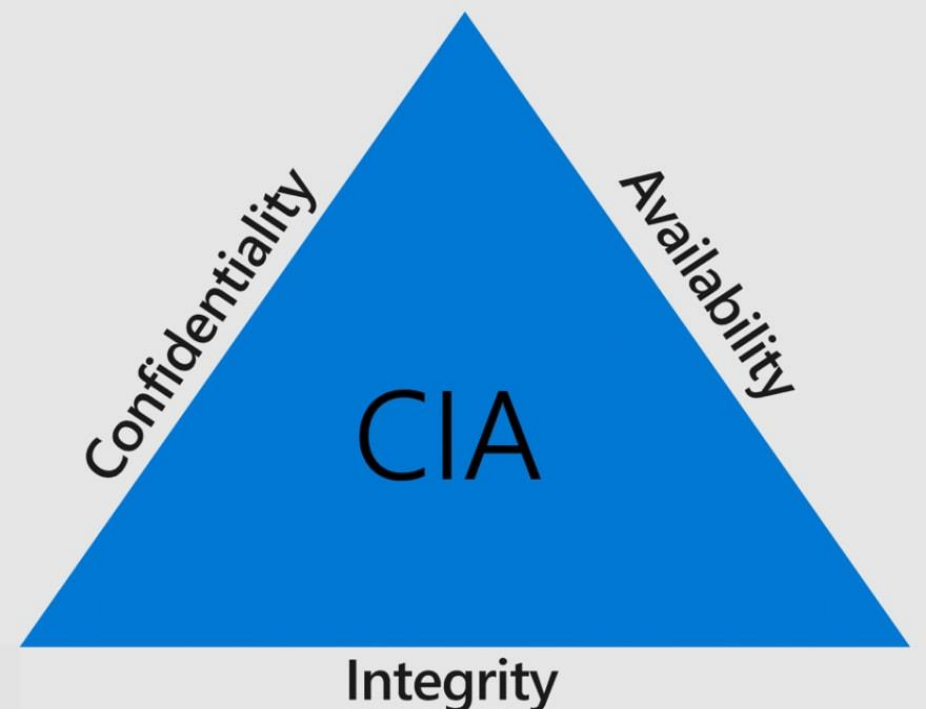
Με τον όρο εμπιστευτικότητα εννοούμε ότι μόνο εξουσιοδοτημένα άτομα η συστήματα μπορούν να έχουν πρόσβαση και δυνατότητα τροποποίησης ευαίσθητων η απόρρητων πληροφοριών. Τα δεδομένα που στέλνονται πρέπει να είναι μη προσβάσιμα από μη εξουσιοδοτημένες οντότητες, είτε αυτές είναι φυσικά πρόσωπα, είτε προγράμματα. Για να επιτευχθεί αυτό, η πρόσβαση στις πληροφορίες πρέπει να ελέγχεται συνεχώς για τυχόν περίπτωση μη εξουσιοδοτημένης κοινής χρήσης τους. Αυτό μπορεί να γίνει σκόπιμα, η τυχαία. Ο πρωταρχικός τρόπος να αποφευχθεί αυτό, είναι η χρήση τεχνικών κρυπτογράφησης για την προστασία των δεδομένων. Με αυτό τον τρόπο, στην περίπτωση που κάποιος αποκτήσει πρόσβαση στα δεδομένα, αυτά τα δεδομένα θα του είναι άχρηστα λόγω του ότι είναι κρυπτογραφημένα και δεν θα μπορούν να αποκρυπτογραφηθούν.

2.1.2 ΑΚΕΡΑΙΟΤΗΤΑ (INTEGRITY)

Η ακεραιότητα, διασφαλίζει ότι τα ευαίσθητα δεδομένα είναι αξιόπιστα και δεν έχουν υποστεί κάποιου είδους μετατροπή από την αρχική τους μορφή. Αυτή η μετατροπή μπορεί να πραγματοποιηθεί από κάποιο κακόβουλο άτομο με σκοπό να εξαπατήσει άλλους ότι τα δεδομένα είναι αυτά που βλέπουν, ενώ στην πραγματικότητα τα δεδομένα έχουν αλλαχτεί από το άτομο αυτό για κακόβουλο σκοπό. Η ακεραιότητα των δεδομένων μπορεί να διατηρηθεί με την προϋπόθεση ότι αυτά είναι ακριβή και αξιόπιστα. Όπως και η εμπιστευτικότητα, έτσι και η ακεραιότητα μπορεί να τεθεί σε κίνδυνο με διάφορους τρόπους. Αυτό μπορεί να συμβεί απευθείας μέσω εισβολής συστημάτων που ανιχνεύουν και τροποποιούν αρχεία, και ως αποτέλεσμα έχουμε την μη εξουσιοδοτημένη τροποποίηση τους.

2.1.3 ΔΙΑΘΕΣΙΜΟΤΗΤΑ (AVAILABILITY):

Τα δεδομένα πρέπει να είναι διαθέσιμα μόνο σε όσους έχουν την άδεια για πρόσβαση σε αυτά. Επίσης είναι αναγκαίο, τα δεδομένα να είναι διαθέσιμα οποιαδήποτε στιγμή κάποια εξουσιοδοτημένη οντότητα τα χρειάζεται. Η διαθεσιμότητα διασφαλίζει ότι τα δεδομένα θα παρέχονται οποιαδήποτε στιγμή ζητηθούν από μία εξουσιοδοτημένη οντότητα.



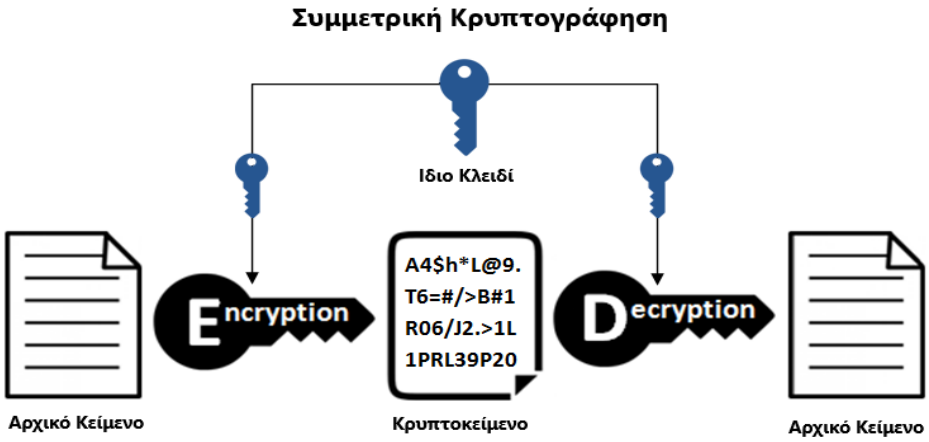
Σχήμα 2.1.1. Τριάδα CIA

2.2 ΣΥΜΜΕΤΡΙΚΗ ΚΡΥΠΤΟΓΡΑΦΗΣΗ

Η κρυπτογράφηση συμμετρικού κλειδιού βασίζεται στην ύπαρξη ενός κλειδιού κρυπτογράφησης. Το κλειδί αυτό έχει διπλό σκοπό. Πρώτον την κρυπτογράφηση και δεύτερον την αποκρυπτογράφηση ενός μηνύματος. Για να υπάρχει ασφάλεια, το κλειδί αυτό πρέπει να είναι γνωστό μόνο στις οντότητες που θέλουν να στείλουν και να λάβουν μηνύματα. Στην απλούστερη περίπτωση ενός αποστολέα και ενός παραλήπτη, το κλειδί αυτό πρέπει να είναι γνωστό μόνο από αυτές τις δύο οντότητες.

Ως παράδειγμα για την κατανόηση λειτουργίας του παραπάνω, θα παρουσιάσουμε την περίπτωση δύο ατόμων που θέλουν να επικοινωνήσουν μεταξύ τους. Η Alice και ο Bob (κύρια ονόματα αποστολέα και παραλήπτη σε παραδείγματα κρυπτογράφησης). Η Alice θέλει να στείλει στον Bob ένα μήνυμα. Αρχικά το βάζει σε ένα κουτί κλειδωμένο με ένα λουκέτο που η Alice έχει το κλειδί για αυτό. Ο Bob για να το ανοίξει, πρέπει με κάποιον τρόπο να έχει το ίδιο κλειδί με της Alice. Μία περίπτωση να το αποκτήσει είναι η Alice να του αποστείλει το κλειδί, η να συναντηθούν και να του το δώσει. Σε κάθε περίπτωση όμως, για να παραδοθεί το αντίγραφο του κλειδιού στον παραλήπτη, υπάρχει κενό ασφαλείας, λόγω του ότι κάποιος μπορεί να υποκλέψει το κλειδί αυτό, και να το χρησιμοποιήσει για την αποκρυπτογράφηση μηνυμάτων.

Χρησιμοποιώντας συμμετρικούς αλγόριθμους κρυπτογράφησης, τα δεδομένα κρυπτογραφούνται έτσι ώστε να μην είναι κατανοητά από οποιονδήποτε δεν διαθέτει το κλειδί για την αποκρυπτογράφηση τους. Μόλις ο παραλήπτης λάβει το μήνυμα, χρησιμοποιεί το κλειδί και ο αλγόριθμος αντιστρέφει την δράση του έτσι ώστε το μήνυμα να επιστρέψει στην αρχική του αναγνώσιμη μορφή. Με την **συμμετρική** κρυπτογράφηση τυπικά το κλειδί είναι 128 αλλά προτιμάται να είναι 256 bit σε μήκος. Οτιδήποτε λιγότερο από 80 bit θεωρείται πλέον μη ασφαλές.



Σχήμα 2.2.1. Μεθοδολογία συμμετρικής κρυπτογράφησης
(https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences)

Η συμμετρική κρυπτογράφηση είναι αποτελεσματική όσον αφορά τον χρόνο υπολογισμού, προκύπτει όμως το εξής πρόβλημα. Αν υποθέσουμε ότι το κανάλι επικοινωνίας μεταξύ αποστολέα και παραλήπτη δεν είναι ασφαλές, τότε ο αποστολέας δεν μπορεί να στείλει το κλειδί ώστε να ληφθεί από τον παραλήπτη και να μπορέσει να αποκρυπτογραφηθεί το κρυπτοκείμενο. Η λύση στο πρόβλημα αυτό δίνεται με την ασύμμετρη κρυπτογράφηση.

2.3 ΑΣΥΜΜΕΤΡΗ ΚΡΥΠΤΟΓΡΑΦΗΣΗ

Το κομμάτι της ασφάλειας στην τεχνολογία της πληροφορίας, έκανε ορισμένα σημαντικά άλματα με την πρόοδο των σύγχρονων ψηφιακών υπολογιστών. Αλγόριθμοι που η παραβίαση τους θεωρούνταν ακατόρθωτη τώρα είναι πια απλό να παραβιαστούν

Ασύμμετρη κρυπτογράφηση η αλλιώς κρυπτογράφηση δημοσίου κλειδιού εκτελείται με έναν τελείως διαφορετικό τρόπο διαχείρισης κλειδιών κρυπτογράφησης από την κρυπτογράφηση συμμετρικού κλειδιού. Η βασική ιδέα της ασύμμετρης κρυπτογράφησης είναι ότι ο αποστολέας και ο παραλήπτης δεν μοιράζονται και χρησιμοποιούν για κρυπτογράφηση και αποκρυπτογράφηση ένα κοινό κλειδί. Αντιθέτως διαθέτουν διαφορετικά κλειδιά με διαφορετικές λειτουργίες.

Έχει χαρακτηριστεί ως η πιο σημαντική σύγχρονη κρυπτογραφική εφεύρεση. Η ασύμμετρη κρυπτογραφία υποστηρίζει την κοινή χρήση πληροφοριών δημόσια (με το δημόσιο κλειδί) που χρησιμοποιείται για την κρυπτογράφηση των δεδομένων και την μεταφορά τους. Τα μεταδιδόμενα δεδομένα μπορούν να αποκρυπτογραφηθούν μόνο από το αντίστοιχο ιδιωτικό κλειδί (private key).

Η ασύμμετρη κρυπτογράφηση λύνει το πρόβλημα ασφαλείας που υπάρχει στους συμμετρικούς αλγορίθμους κρυπτογράφησης. Στην συγκεκριμένη περίπτωση, η κάθε οντότητα έχει στην κατοχή της δύο κλειδιά. Ένα ιδιωτικό κλειδί (private key) και ένα δημόσιο (public key). Το ιδιωτικό κλειδί θα πρέπει να είναι προστατευμένο από οποιονδήποτε και να μην δύναται η δυνατότητα κλοπής του. Από την άλλη μεριά όμως, το δημόσιο κλειδί, μπορεί να είναι διαθέσιμο σε οποιονδήποτε.

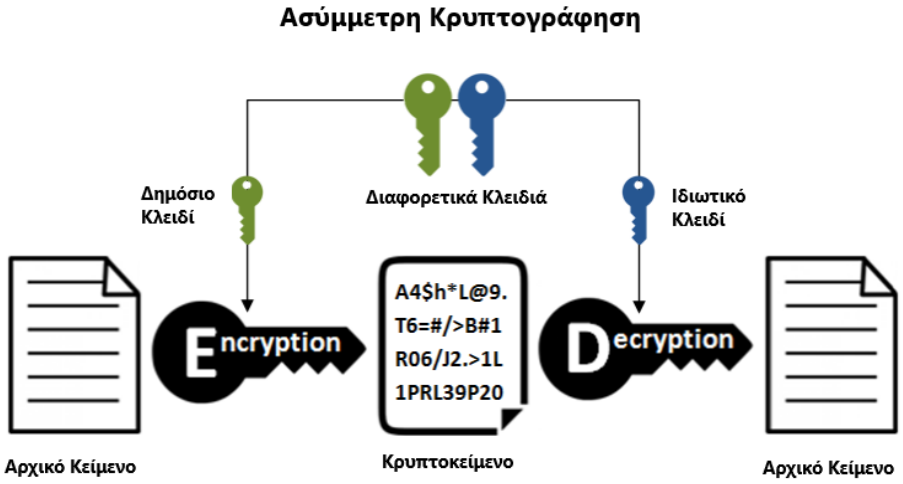
2.3.1 ΤΡΟΠΟΣ ΛΕΙΤΟΥΡΓΙΑΣ

Οι κρυπτογραφικοί αλγόριθμοι δημοσίου κλειδιού γίνονται από ειδικές συναρτήσεις που δέχονται σαν είσοδο έναν τυχαίο, μεγάλο αριθμό. Ως έξοδο παράγουν το ζεύγος κλειδιών που αναφέρθηκε πιο πάνω, ένα δημόσιο και ένα ιδιωτικό κλειδί. Το δημόσιο κλειδί, μπορεί να βρεθεί από το ιδιωτικό, αλλά το

αντίστροφο δεν μπορεί να συμβεί, λόγω του ότι ο υπολογισμός είναι εύκολος στην εκτέλεση προς την μία μεριά και δύσκολη η αντιστροφή του.

Αυτή η ιδιότητα αποτελεί την βάση της ασύμμετρης κρυπτογράφησης. Σε συνδυασμό με την εμπιστευτικότητα που προσφέρει αυτός ο τρόπος κρυπτογράφησης, η ασύμμετρη κρυπτογράφηση μπορεί να εξασφαλίσει τον έλεγχο ταυτότητας των χρηστών αλλά και να διασφαλίσει τις τρείς ιδιότητες που αναλύθηκαν στην εισαγωγή. Όλα αυτά με την προϋπόθεση χρήσης ψηφιακών υπογραφών.

Ένας αποστολέας μπορεί να συνδυάσει ένα μήνυμα μαζί με το ιδιωτικό του κλειδί και με αυτό τον τρόπο να δημιουργήσει μια σύντομη ψηφιακή υπογραφή στο μήνυμα. Οποιοσδήποτε έχει το αντίστοιχο δημόσιο κλειδί του αποστολέα, μπορεί πλέον να συνδυάσει αυτό το μήνυμα με την ψηφιακή υπογραφή. Αν η υπογραφή ταιριάζει με το μήνυμα, τότε επαληθεύεται η προέλευση του και έτσι διασφαλίζεται η τριάδα κρυπτογράφησης. Το μήκος κλειδιού που συνίσταται είναι 1024bit, με προτιμώμενη τιμή τα 2048bit.



Σχήμα 2.3.1. Μεθοδολογία ασύμμετρης κρυπτογράφησης (https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences)

2.4 TRANSPORT LAYER SECURITY (TLS)

2.4.1 ΕΙΣΑΓΩΓΗ

Το πρωτόκολλο **TLS** (Transport Layer Security) εξελίχθηκε από το πρωτόκολλο SSL (Secure Socket Layers) το οποίο αναπτύχθηκε αρχικά το 1994 από την εταιρία Netscape Communications Corporation για την ασφάλεια web sessions. Το πρωτόκολλο SSL (Secure Sockets Layer) 1.0 δεν κυκλοφόρησε ποτέ δημόσια, ενώ το πρωτόκολλο SSL 2.0 αντικαταστάθηκε γρήγορα από το SSL 3.0 , από το οποίο βασίζεται και το TLS.

Είναι ένα κρυπτογραφικό πρωτόκολλο που παρέχει ασφάλεια από άκρο σε άκρο (end-to-end) των δεδομένων που αποστέλλονται μεταξύ εφαρμογών μέσω διαδικτύου. Είναι κυρίως γνωστό στους χρήστες μέσω της χρήσης του στην ασφαλή περιήγηση στον ιστό, και ειδικότερα το εικονίδιο του λουκέτου που εμφανίζεται στα προγράμματα περιήγησης ιστού όταν δημιουργείται μία ασφαλής σύνδεση. Πέραν από αυτή την λειτουργία χρησιμοποιείται για άλλες εφαρμογές, όπως e-mail, μεταφορά αρχείων, βίντεο/βίντεο με ήχο, ανταλλαγή μηνυμάτων, ηχητικά μηνύματα όπως και διαδικτυακές υπηρεσίες (DNS, NTP). Η τελευταία έκδοση του πρωτοκόλλου είναι η 1.3 όπου αυτή χρησιμοποιείται πλέον.

2.4.2 ΛΕΙΤΟΥΡΓΙΑ

Το TLS χρησιμοποιεί ασύμμετρη κρυπτογράφηση, καθώς όπως περιγράφεται παραπάνω, παρέχει έναν καλό συνδυασμό απόδοσης και ασφάλειας κατά την μετάδοση των δεδομένων. Το πλεονέκτημα της ασύμμετρης κρυπτογράφησης που χρησιμοποιείται στο TLS είναι ότι η διαδικασία κοινής χρήσης δημοσίων κλειδιών δεν χρειάζεται να είναι ασφαλής. Οποιοσδήποτε μπορεί να έχει το δημόσιο κλειδί του παραλήπτη και του αποστολέα, αρκεί τα ιδιωτικά τους κλειδιά να είναι ασφαλή και προστατευμένα ώστε να είναι αδύνατη η ανάκτηση τους από κακόβουλα άτομα. Το μήκος κλειδιού που συνίσταται είναι 1024bit, με προτεινόμενη τιμή τα 2048bit.

Το πρωτόκολλο TLS αποτελείται από έναν αριθμό υπό-πρωτόκολλων που από αυτά, τα δύο πιο σημαντικά είναι το πρωτόκολλο χειραψίας (handshake protocol) και το πρωτόκολλο εγγραφής (record protocol).

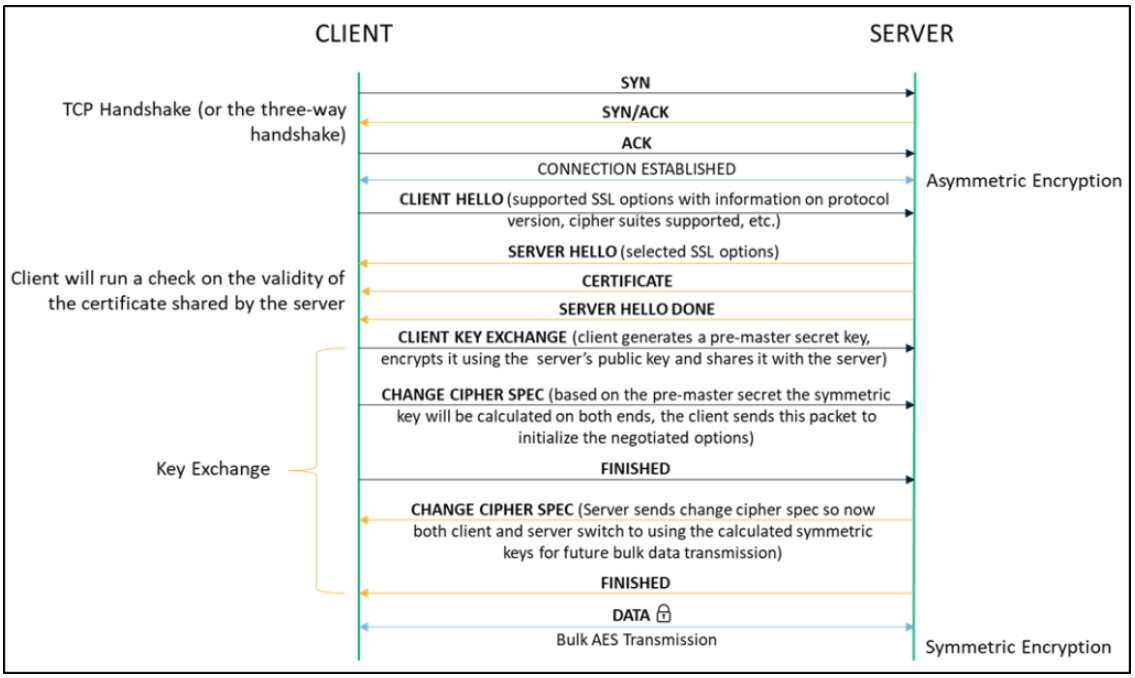
2.4.3 ΥΠΟ ΠΡΩΤΟΚΟΛΛΟ ΧΕΙΡΑΨΙΑΣ (HANDSHAKE PROTOCOL)

Το πρωτόκολλο χειραψίας είναι υπεύθυνο για τον έλεγχο ταυτότητας και την ανταλλαγή των κλειδιών που απαιτούνται για την δημιουργία και συνέχιση μιας ασφαλούς σύνδεσης μεταξύ server και client. Το πρωτόκολλο αρχικά επιλέγει τον τρόπο κρυπτογράφησης που θα χρησιμοποιηθεί σε όλη την ασφαλή σύνδεση. Επίσης είναι υπεύθυνο για την ανταλλαγή κλειδιών. Χρησιμοποιείται στην αρχή κάθε σύνδεσης για την επιλογή μιας έκδοσης πρωτοκόλλου, κρυπτογραφικών παραμέτρων και έλεγχο ταυτότητας των δύο οντοτήτων (client-server). Μπορεί κανείς να σκεφτεί, ότι η χειραψία αποτελείται από τρείς φάσεις. Αρχικά την φάση ανταλλαγής κλειδιών. Στην δεύτερη φάση, ο server στέλνει παραμέτρους που δεν απαιτούνται στην πρώτη φάση. Αυτές περιλαμβάνουν ένα ερώτημα για το αν είναι αναγκαία η αυθεντικοποίηση του client. Στην Τρίτη και τελευταία φάση, ο server (και προαιρετικά ο client) επαληθεύει τον εαυτό του με ένα επιπλέον πιστοποιητικό.

2.4.4 ΥΠΟ ΠΡΩΤΟΚΟΛΛΟ ΕΓΓΡΑΦΗΣ (RECORD PROTOCOL)

Ο σκοπός του πρωτοκόλλου εγγραφής είναι να δημιουργήσει μία ασφαλή σύνδεση μεταξύ server και client. Είναι υπεύθυνο για την ασφάλιση των δεδομένων της εφαρμογής και την επαλήθευση της ακεραιότητας (integrity) και της προέλευσης των δεδομένων που λαμβάνονται. Είναι υπεύθυνο για την διαίρεση των εξερχόμενων μηνυμάτων σε διαχειρίσιμα block και την επανασυναρμολόγηση των εισερχόμενων μηνυμάτων. Το πρωτόκολλο αυτό συμπιέζει τα εξερχόμενα block και αποσυμπιέζει τα εισερχόμενα. Επίσης είναι μέσω του πρωτοκόλλου αυτού γίνεται η κρυπτογράφηση εξερχόμενων και αποκρυπτογράφηση εισερχόμενων μηνυμάτων. Όταν ολοκληρωθεί το πρωτόκολλο εγγραφής, τα εξερχόμενα κρυπτογραφημένα δεδομένα μεταβιβάζονται στο πρωτόκολλο TCP για την μεταφορά τους.

2.4.5 ΔΙΑΔΙΚΑΣΙΑ ΠΡΩΤΟΚΟΛΛΟΥ



Σχήμα 2.4.5.1. Η διαδικασία του πρωτοκόλλου TLS (https://sectigostore.com/blog/port-443-everything-you-need-to-know-about-https-443/)

- Η διαδικασία ξεκινά με μία σύνδεση TCP τριών σταδίων. Στο πρώτο βήμα ο client θέλει να συνδεθεί με τον server οπότε στέλνει ένα SYN (Synchronize Sequence Number) που πληροφορεί τον server ότι ο client θέλει να ξεκινήσει μια σύνδεση με αυτόν.

-Στην συνέχεια ο server απαντά στο αίτημα του client με ένα SYN/ACK (απάντηση). Αυτό σημαίνει ότι ο server έλαβε το SYN που στάλθηκε από τον client και προχωρά στο πρωτόκολλο.

-Σαν τελικό βήμα της TCP σύνδεσης, ο client πιστοποιεί την απάντηση του server και δημιουργείται μια αξιόπιστη σύνδεση με την οποία θα ξεκινήσουν πραγματική μεταφορά δεδομένων.

-Αφού η TCP σύνδεση ήταν επιτυχής και τώρα μπορούν να ανταλλαχθούν πληροφορίες μεταξύ client-server, στην συνέχεια εκτελείται το TLS πρωτόκολλο. Η διαδικασία ξεκινά με την ανταλλαγή μηνυμάτων “hello” μεταξύ του client και του server.

-Μόλις ξεκινήσει η διαπραγμάτευση πρωτοκόλλου κρυπτογράφησης, κοινοποιούνται τα πρότυπα κρυπτογράφησης που υποστηρίζονται και ο server μοιράζεται το πιστοποιητικό του.

-Ο πελάτης τώρα έχει το δημόσιο κλειδί του server και το πιστοποιητικό του. Με αυτό επαληθεύεται η εγκυρότητα του πριν χρησιμοποιηθεί το δημόσιο κλειδί του από τον client για την

δημιουργία ενός “pre-master” μυστικού κλειδιού. Στην συνέχεια το “pre-master” κλειδί κρυπτογραφείται με το δημόσιο κλειδί του server και στέλνεται σε αυτόν.

-Με βάση το “pre-master” κλειδί που έχουν και τα δύο μέρη, υπολογίζουν ανεξάρτητα το συμμετρικό κλειδί που θα χρησιμοποιήσουν για την επικοινωνία τους.

-Ο Server ειδοποιεί τον Client ότι υπολόγισε το συμμετρικό κλειδί.

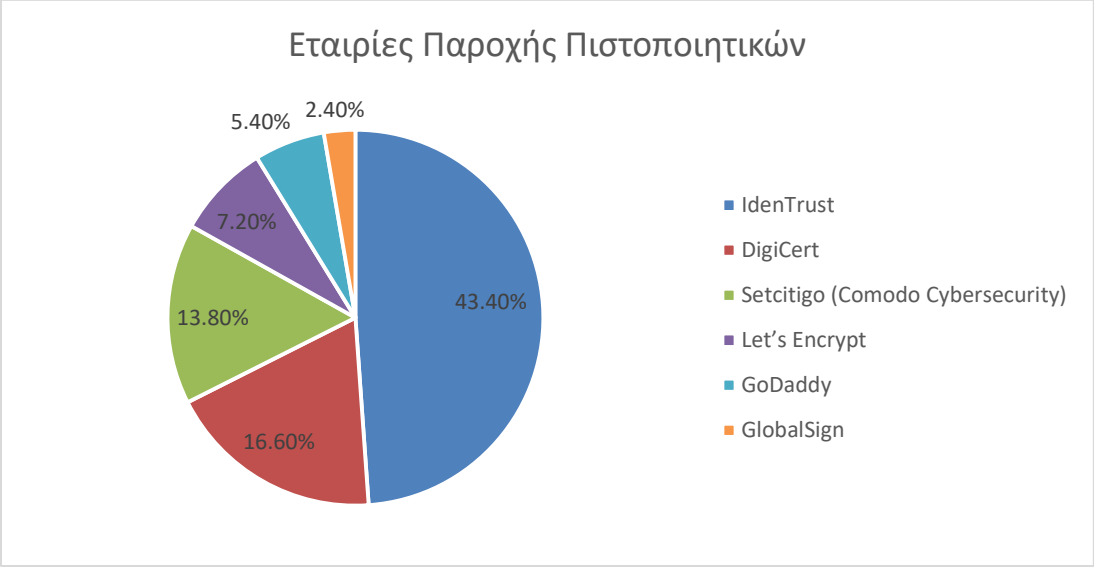
-Ο Client ειδοποιεί τον Server ότι υπολόγισε σωστά το συμμετρικό κλειδί.

-Ξεκινά η ασφαλής συμμετρική κρυπτογράφηση.

2.5 CERTIFICATE AUTHORITY (CA)

Πέραν από την κρυπτογράφηση που παρέχεται για μια ασφαλή σύνδεση, είναι επίσης αναγκαίο ένας client που συνδέεται με τον server, να μπορεί να επικυρώσει την ιδιοκτησία του δημόσιου κλειδιού του server. Αυτό πραγματοποιείται συνήθως χρησιμοποιώντας ένα ψηφιακό πιστοποιητικό X.509 που εκδίδεται από μία αξιόπιστη πηγή που ονομάζεται Αρχή Έκδοσης Πιστοποιητικών (Certificate Authority), και βεβαιώνει την γνησιότητα του δημοσίου κλειδιού του server.

Ο παρακάτω πίνακας αναπαριστά τις κύριες εταιρίες παροχής πιστοποιητικών που υπάρχουν σήμερα.



Πίνακας 2.5.1. Παρουσίαση κύριων εταιριών παροχής πιστοποιητικών (https://en.wikipedia.org/wiki/Certificate_authority)

Έχοντας ως μέσο την αρχή έκδοσης πιστοποιητικών για την επιλογή και χρήση δημοσίου κλειδιού, ο client, μπορεί να είναι σίγουρος ότι το δημόσιο κλειδί του server είναι το πραγματικό, και με αυτό τον τρόπο οι ευαίσθητες πληροφορίες να κρυπτογραφηθούν με το σωστό δημόσιο κλειδί, ώστε να μπορέσουν να αποκρυπτογραφηθούν με το αντίστοιχο ιδιωτικό.

2.6 HTTP-HTTPS

2.6.1 ΕΙΣΑΓΩΓΗ

Ο όρος HTTP (Hypertext Transfer Protocol, Πρωτόκολλο Μεταφοράς Υπερκειμένου) επινοήθηκε από τον Ted Nelson το 1965 στο Xanadu Project, το οποίο εμπνεύστηκε από το όραμα που είχε ο Vannevar Bush την δεκαετία του 1930 για ένα σύστημα ανάκτησης και διαχείρισης πληροφοριών που βασίζεται σε μικροφίλμ και περιγράφεται σε δημοσίευση του 1945 με όνομα “As We May Think”.

Το HTTP αποτελεί το σύνολο των κανόνων που απαιτούνται για την μεταφορά αρχείων όπως κείμενο, εικόνες, ήχος, βίντεο κ.α. μεταξύ client και server. Υπάρχουν δύο είδη μηνυμάτων που ανταλλάσσονται, τα HTTP Requests και τα HTTP Responds. Requests στέλνονται από τον Client ώστε να εκτελεστεί κάποια ενέργεια από τον server ενώ responds είναι η απάντηση του server στα requests που του έχουν σταλθεί από τον client.

2.6.2 HTTP REQUESTS

Ένα HTTP Request είναι ο τρόπος με τον οποίο ο client ζητά τις πληροφορίες που χρειάζεται από τον Server. Κάθε HTTP Request που γίνεται φέρει μαζί του κάποια κωδικοποιημένα δεδομένα που μεταφέρουν διαφορετικούς τύπους πληροφοριών. Ένα τυπικό αίτημα περιέχει τον τύπο (version) του HTTP, διεύθυνση URL και την μέθοδο HTTP, HTTP request header.

2.6.3 HTTP RESPONDS

Ένα HTTP Respond είναι η απάντηση που λαβαίνει ο client από έναν server, ως απάντηση από το αίτημα που έχει κάνει σε αυτόν. Αυτή η απάντηση μεταδίδει πολύτιμες πληροφορίες με βάση αυτό που ζητήθηκε

στο αίτημα που είχε γίνει. Ένα τυπικό HTTP Respond περιέχει ένα HTTP Status, HTTP response header, HTTP body.

2.6.4 HTTP METHOD

Μία μέθοδος HTTP που μερικές φορές αναφέρεται και ως HTTP Verb, υποδεικνύει την ενέργεια που αναμένει ο client να λάβει από τον server που δέχθηκε το request. Δύο από τις πιο κοινές μεθόδους είναι η HTTP GET και POST. Το αίτημα GET αναμένει πληροφορίες, ενώ το αίτημα POST συνήθως υποδεικνύει ότι ο client υποβάλει τις πληροφορίες στον server.

2.6.5 HTTPS

Το HTTPS (Hypertext Transfer Protocol Secure) είναι μια επέκταση του HTTPS. Ουσιαστικά η διαφορά που έχει με το HTTPS είναι ότι σε αυτό, προστίθεται η κρυπτογράφηση TLS Που περιγράφηκε πιο πάνω. Με αυτό τον τρόπο προκύπτει ένα πρωτόκολλο που μέσω αυτού μπορεί να γίνει η αμφίδρομη μετάδοση δεδομένων μεταξύ client-server με ασφαλή τρόπο. Με λίγα λόγια αυτό διασφαλίζει ότι οποιοδήποτε άτομο βρίσκεται στο ίδιο δίκτυο δεν μπορεί να δει, να υποκλέψει η να τροποποιήσει μια σύνδεση HTTPS που λαβαίνει χώρα μεταξύ client και server.

2.6.6 PORT 443

Μία θύρα (port) είναι μια εικονικά αριθμημένη διεύθυνση που χρησιμοποιείται ως το τελικό σημείο επικοινωνίας από πρωτόκολλα επιπέδου μεταφοράς όπως το UDP η το TCP. Η standard πόρτα για τον χειρισμό όλων των μη κρυπτογραφημένων συνδέσεων γίνεται στην πόρτα 80. Ενώ για ασφαλείς κρυπτογραφημένες συνδέσεις, η standard πόρτα που χρησιμοποιείται είναι η πόρτα 443. Αυτή η πόρτα θα χρησιμοποιηθεί για την σύνδεση.

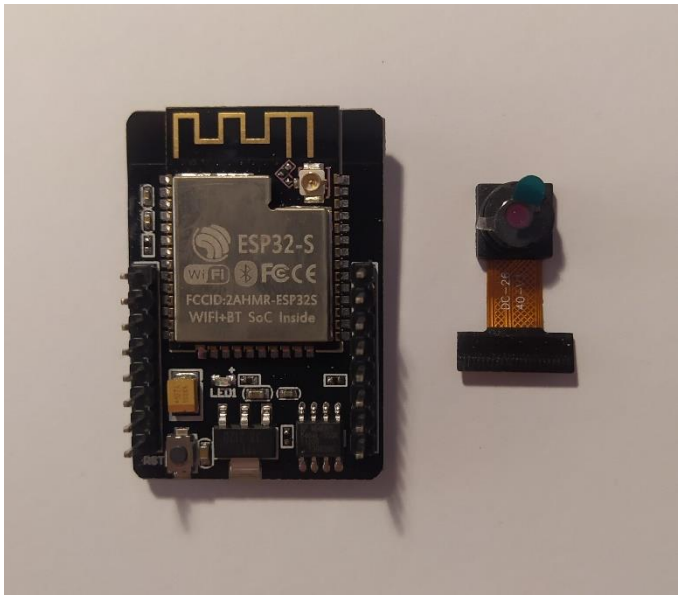
3. ΠΕΡΙΓΡΑΦΗ ΣΥΣΤΗΜΑΤΟΣ ΥΛΙΚΟΥ

3.1 ESP32-CAM

Το ESP32-CAM είναι ένας χαμηλού κόστους μικροελεγκτής με ενσωματωμένα περιφερειακά εισόδου/εξόδου (I/O), Wi-Fi και Bluetooth. Έχει δημιουργηθεί έτσι ώστε να επιτυγχάνεται ελάχιστη κατανάλωση ενέργειας, αλλά ταυτόχρονα να μπορεί να λειτουργήσει σε αφιλόξενο για τον άνθρωπο περιβάλλον (-40°C έως 125°C)

Επίσης η ομάδα κατασκευής του έχει προνοήσει για το βάρος και τον όγκο του, ώστε να μπορεί να χρησιμοποιηθεί σε μικροσυσκευές, και γενικότερα σε υλοποιήσεις που απαιτείται μικρό επιπλέον βάρος (πχ drones).

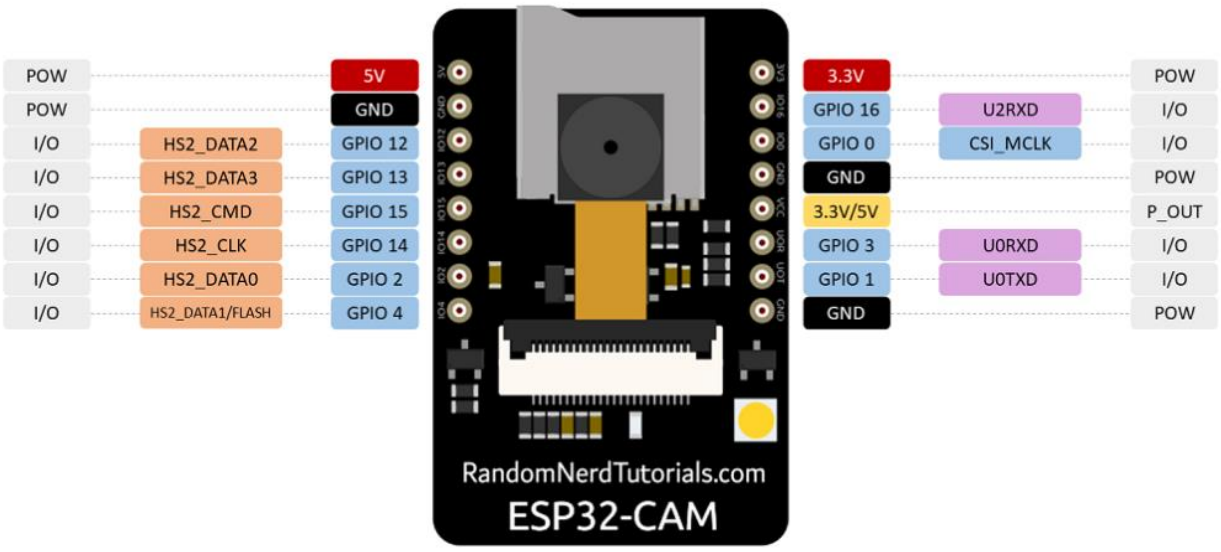
Μπορεί να χρησιμοποιηθεί σε διάφορες εφαρμογές που εκτελούνται με το “διαδίκτυο των πραγμάτων” Internet of Things (IoT).



Εικόνα 3.1.1. ESP32-CAM και OV2640 Camera

Παρακάτω παρατίθενται τα χαρακτηριστικά του:

- 802.11b/g/n Wi-Fi
- Bluetooth 4.2 με BLE
- UART, SPI, I2C και PWM διεπαφές
- Ταχύτητα χρονισμού ως 160 MHz
- Υπολογιστική ισχύ ως 600 DMIPS
- 520 KB SRAM και 4 MB PSRAM
- Υποστήριξη μεταφόρτωση εικόνας μέσω Wi-Fi
- Πολλαπλά προφίλ αδράνειας
- Firmware Over the Air (FOTA) αναβαθμίσιμο
- 10 GPIO πόρτες
- Ενσωματωμένο Flash LED



Σχήμα 3.1.2. Τα pins του ESP32-CAM (https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/)

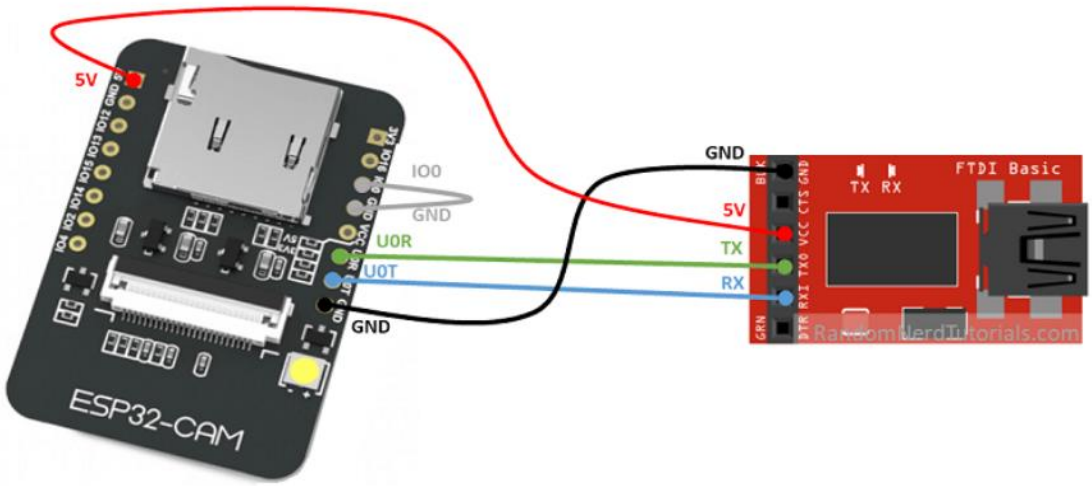
Το ESP32-CAM έρχεται με δύο power pins (χρωματισμένα με κόκκινο στην εικόνα). Η τάση τους είναι 3.3 και 5V. Η εταιρία Espressif, που είναι η εταιρία παραγωγής του ESP32-CAM, συστήνει η τροφοδοσία να γίνεται από την είσοδο των 5V όπου και για αυτό επιλέχθηκε να τροφοδοτηθεί από εκεί. Υπάρχουν τρία pins γείωσης, και στο σχήμα συμβολίζονται με "**GND**". Υπάρχει μία έξοδος με τάση 3.3 ή 5V και στο σχήμα βρίσκεται στο pin με το κίτρινο χρώμα και την ένδειξη "**VCC**". Για τον προγραμματισμό του χρησιμοποιούνται τα GPIO1 και GPIO3 pins. Τα υπόλοιπα GPIO's μπορούν να χρησιμοποιηθούν από τον χρήστη και να τροποποιηθούν στον κώδικα.

3.2 ΠΕΡΙΦΕΡΕΙΑΚΑ ESP32-CAM

Για τον προγραμματισμό της συσκευής χρειάζεται να επιλεχθεί ένας από τους παρακάτω τρόπους.

3.2.1 FTDI PROGRAMMER

Ο πρώτος τρόπος είναι με την χρήση ενός FTDI programmer, ακολουθείται η συνδεσμολογία που φαίνεται στο παρακάτω σχήμα, και στην συνέχεια συνδέεται στο FTDI Programmer ένα micro usb.



Εικόνα 3.2.1.1. Συνδεσμολογία για προγραμματισμό με FTDI (https://randomnerdtutorials.com/program-upload-code-esp32-cam/)

3.2.2 ESP32-CAM MB

Ο δεύτερος και πιο απλός τρόπος περιλαμβάνει την χρήση ενός ESP32-CAM MB. Για την συνδεσμολογία του απαιτείται απλά η τοποθέτηση του πάνω στα PINS του ESP32-CAM, και η σύνδεση ενός Micro-USB σε αυτό (το ESP32-CAM MB). Λόγω της πρακτικότητας που προσφέρεται, επιλέχθηκε ο δεύτερος τρόπος για την υλοποίηση.

Εικόνα 3.2.2.1. ESP32-CAM MB

Εικόνα 3.2.2.2. Τελική όψη μικροελεγκτή

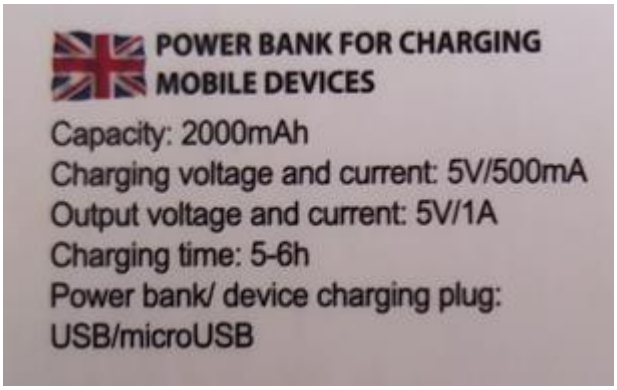
3.3 ΠΕΡΙΦΕΡΕΙΑΚΑ ΥΛΟΠΟΙΗΣΗΣ

Για την δημιουργία του server όπως και για τον προγραμματισμό του ESP32-CAM, του Server αλλά και την χρήση τους, απαιτείται ένας ηλεκτρονικός υπολογιστής. Δεδομένου ότι η διπλωματική εργασία αυτή, τοποθετείται στην χρήση του ESP32-CAM σε DRONE, ο υπολογιστής χρειάζεται να είναι φορητός (laptop) έτσι ώστε να μπορεί να λειτουργήσει σε εξωτερικό περιβάλλον.

Χρειαζόμαστε επίσης μία ηλεκτρονική συσκευή που να έχει την δυνατότητα “Mobile Hotspot”. Η πιο προσιτή και απλή μέθοδος είναι με την χρήση ενός κινητού τηλεφώνου τύπου “Smartphone”. Επισημαίνεται ότι δεν απαιτείται σύνδεση στο internet για την εκτέλεση της εφαρμογής, αλλά μόνο ένα σημείο hotspot. Για την υλοποίηση επιλέχθηκε να χρησιμοποιηθεί το τηλέφωνο Xiaomi Redmi Note 7.

Τροφοδοσία ρεύματος

Για την τροφοδοσία ρεύματος επιλέχθηκε μια συσκευή μπαταρίας τύπου “power bank”. Δεδομένου ότι το ESP32-CAM χαρακτηρίζεται ως συσκευή χαμηλής κατανάλωσης ενέργειας, η φορητή συσκευή τροφοδοσίας είναι 2000mAh. Το μοντέλο που χρησιμοποιήθηκε ανήκει στην εταιρία “Extreme the definitive protection” και είναι το μοντέλο XMP101. Στο κεφάλαιο των μετρήσεων γίνονται αναλυτικές μετρήσεις για την κατανάλωση και το χρονικό διάστημα που μπορεί να λειτουργεί ο μικροελεγκτής με την χρήση αυτού του εξοπλισμού.



Εικόνα 3.3.1. Χαρακτηριστικά συσκευής τροφοδοσίας

Όπως αναφέρει ο κατασκευαστής, το power bank έχει χωρητικότητα 2000mAh ενώ το ρεύμα εξόδου του ανέρχεται στα 5V/1A, που είναι αρκετό για να τροφοδοτήσει το ESP32-CAM

Εικόνα 3.3.2. Συσκευή τροφοδοσίας ρεύματος (power bank)

Το power bank έχει μορφή ορθογώνιου παραλληλόγραμμου με διαστάσεις 9.3cm X 2cm και μαζί με την συσκευασία έρχεται με καλώδιο micro usb μήκους 30 εκατοστών που θα χρησιμοποιηθεί.

3.4 ΚΟΣΤΟΣ ΥΛΟΠΟΙΗΣΗΣ

Από το εξωτερικό αγοράστηκε το ESP32-CAM, ESP32-CAM MB. Το συνολικό κόστος υλοποίησης υπολογίζεται στον παρακάτω πίνακα.

Hardware	**Τιμή (€)**
ESP32-CAM	15.20
ESP32-CAM MB	6.4
Micro USB Καλώδιο	1.7
Power Bank	9,8
Συνολικό Ποσό	33.1

Πίνακας 3.4.1. Συνολικό κόστος υλοποίησης

Ο ηλεκτρονικός Υπολογιστής (laptop) κυμαίνεται στα 400-800 € και το κινητό τηλέφωνο τύπου “smartphone” κυμαίνεται στα 300-500 €. Τα υλικά αυτά δεν προστέθηκαν στα έξοδα λόγω του ότι θεωρείται ότι υπάρχουν ήδη και χρειάζεται μόνο να χρησιμοποιηθούν.

Ερχόμαστε στο συμπέρασμα ότι το κόστος για το συνολικό project ανέρχεται στα 33.1 ευρώ, ενώ λόγω της αυξομείωσης των τιμών στα παραπάνω είδη μπορούμε να πούμε ότι τα υλικά υπολογίζονται σε -10 ως +10 ευρώ, άρα 23,1 ως 43.1 ευρώ.

4. ΠΕΡΙΓΡΑΦΗ ΥΛΟΠΟΙΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

4.1 NODE.JS SERVER

Για την δημιουργία του ασφαλούς server θα χρησιμοποιηθεί η built-in βιβλιοθήκη που υπάρχει στο Node.js και είναι κατασκευασμένη για την δημιουργία και αλληλεπίδραση με HTTPS server.

Πέραν αυτού, θα χρειαστεί ένα Πιστοποιητικό SSL/TLS. Το πιστοποιητικό αυτό μπορεί να υπάρξει με δύο μεθόδους στην υλοποίηση. Η πρώτη μέθοδος είναι η αγορά του από μία Αρχή Έκδοσης Πιστοποιητικών (Certificate Authority) όπως αναλύθηκε στο κεφάλαιο 2.5. Η δεύτερη μέθοδος είναι να χρησιμοποιηθεί ένα Αυτοϋπογεγραμμένο Πιστοποιητικό (Self Signed Certificate). Με την χρήση αυτού του πιστοποιητικού, ουσιαστικά η Αρχή Έκδοσης Πιστοποιητικών γίνεται το άτομο που το δημιουργεί. Για την υλοποίηση επιλέχθηκε ο δεύτερος τρόπος, και η δημιουργία του πιστοποιητικού αναλύεται παρακάτω.

Για την δημιουργία του πιστοποιητικού θα χρησιμοποιήσουμε την βιβλιοθήκη OpenSSL. Η OpenSSL είναι μια βιβλιοθήκη λογισμικού για εφαρμογές προστασίας των επικοινωνιών μεταξύ δικτύων και υπολογιστών από κακόβουλες επιθέσεις και προσπάθειες υποκλοπής δεδομένων. Χρησιμοποιείται ευρέως από servers στο διαδίκτυο συμπεριλαμβανομένων των περισσότερων σελίδων που λειτουργούν με HTTPS. Το OpenSSL χορηγείται με άδεια χρήσης τύπου Apache, πράγμα που ουσιαστικά σημαίνει ότι είμαστε ελεύθεροι να το αποκτήσουμε και να το χρησιμοποιήσουμε για εμπορικούς σκοπούς, με την επιφύλαξη ορισμένων απλών όρων άδειας χρήσης.

Εικόνα 4.1.1. Λογότυπο της OpenSSL

4.1.1 SELF SIGNED CERTIFICATE-PRIVATE KEY

Για την δημιουργία του HTTPS Server θα χρειαστεί πρώτα να παραχθεί ένα ιδιωτικό κλειδί, και στην συνέχεια ένα αυτοϋπογεγραμμένο πιστοποιητικό. Στην επίσημη ιστοσελίδα του Node.js παρέχονται πληροφορίες για την δημιουργία και χρήση ιδιωτικών κλειδιών και αυτοϋπογεγραμμένων πιστοποιητικών

και μέσω αυτών των πληροφοριών θα δημιουργήσουμε τα πιστοποιητικά. Οι εντολές για την δημιουργία γράφονται στο PowerShell του λειτουργικού συστήματος Windows και αναλύονται παρακάτω.

```
> openssl genrsa -out key.pem
```

Σχήμα 4.1.1.1. Δημιουργία ιδιωτικού κλειδιού

Βήμα 1: Δημιουργία ιδιωτικού RSA κλειδιού και αποθήκευση στο αρχείο “key.pem”

```
> openssl req -new -key key.pem -out csr.pem
```

Σχήμα 4.1.1.2. Νέο certificate request

Βήμα 2: Δημιουργία καινούργιου certificate request για το παραπάνω ιδιωτικό κλειδί που μόλις δημιουργήθηκε. Το certificate request αποθηκεύεται στο αρχείο “csr.pem”. Για την ολοκλήρωση της εντολής αυτής, χρειάζεται να εισαχθούν κάποιες πληροφορίες για το πιστοποιητικό που επιθυμούμε να δημιουργηθεί.

```
-----
Country Name (2 letter code) [AU]:GR
State or Province Name (full name) [Some-State]:ATHENS
Locality Name (eg, city) []:ELLINIKO
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UNIWA
Organizational Unit Name (eg, section) []:COMPUTER ENGINEERING
Common Name (e.g. server FQDN or YOUR name) []:FILIPPOS
Email Address []:cs161023@uniwa.gr

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:1234567890
An optional company name []:NONE
```

Σχήμα 4.1.1.3. Εισαγωγή πληροφοριών για δημιουργία πιστοποιητικού

Βήμα 3: Εισαγωγή των πληροφοριών που απαιτούνται για την δημιουργία του αυτοϋπογεγραμμένου πιστοποιητικού. Οι βασικές πληροφορίες που απαιτούνται είναι η χώρα προέλευσης που χαρακτηρίζεται από δύο (2) κεφαλαία λατινικά γράμματα, στην συνέχεια όνομα πολιτείας, πόλη, όνομα οργανισμού, όνομα μονάδας οργανισμού, διεύθυνση ηλεκτρονικού ταχυδρομείου, όνομα κατόχου πιστοποιητικού. Στην συνέχεια ζητούνται κάποιες επιπλέον μεταβλητές. Ο κωδικός “challenge password” είναι ένας κωδικός που ζητείται ως μέρος της δημιουργίας του πιστοποιητικού όταν ένα κλειδί κρυπτογραφείται και χρησιμοποιείται κάθε φορά που ενεργοποιείται το πρωτόκολλο SSL. Ο κωδικός αυτός πρέπει να αποτελείται μόνο από αλφαριθμητικά ψηφία. Τέλος, ζητείται ένα προαιρετικό όνομα. Όλες αυτές οι πληροφορίες αποθηκεύονται στο αρχείο “csr.pem”.

```
> openssl x509 -req -days 9999 -in csr.pem -signkey key.pem -out cert.pem
```

Σχήμα 4.1.1.4. Δημιουργία πιστοποιητικού

Βήμα 4: Στην συνέχεια χρησιμοποιούμε το x509 utility που είναι ένα πρόγραμμα εμφάνισης και υπογραφής πιστοποιητικών. Από προεπιλογή, το πιστοποιητικό αναμένεται στην είσοδο, με την εντολή "–req" εννοούμε ότι γίνεται certificate request. Στην συνέχεια ορίζουμε τον αριθμό ημερών που θα ισχύει το πιστοποιητικό. Η αρχή των ημερών ξεκινά από την ημερομηνία δημιουργίας του. Με το attribute –in διαβάζουμε το πιστοποιητικό από το αρχείο "csr.pem" ενώ με το "–signkey" το πιστοποιητικό γίνεται αυτοϋπογεγραμμένο. Στην συνέχεια αποθηκεύεται στο αρχείο "cert.pem".

```
rm csr.pem
```

Σχήμα 4.1.1.5. Διαγραφή certificate request

Βήμα 5: Έχοντας στην διάθεση μας το πιστοποιητικό ("cert.pem") και το κλειδί ("key.pem"), μπορούμε να διαγράψουμε το certificate request που κάναμε στα προηγούμενα βήματα αφού πλέον δεν είναι αναγκαίο σε κάποια από τις επόμενες ενέργειες που θα πραγματοποιηθούν.

4.1.2 ΚΩΔΙΚΑΣ ΤΟΥ SERVER

```
//System Modules Required
const fs = require('fs');
const https = require('node:https');
const path = require('node:path');
const express = require('express');
const app = express();
const WebSocket = require('ws');

//Key and Certificate from folder
const keyAndCert = {
    key: fs.readFileSync('keyAndCert/key.pem'),
    cert: fs.readFileSync('keyAndCert/cert.pem')
  };

//Creating the Server
const secureServer = https.createServer(keyAndCert, app);
const webSocketServer = new WebSocket.Server({ server: secureServer });
```

Σχήμα 4.1.2.1. Πρώτο block κώδικα στον server

Για την υλοποίηση του server απαιτούνται κάποια modules που χρησιμοποιούνται στην συνέχεια. Τα modules αυτά είναι "fs", για αλληλεπίδραση με αρχεία που βρίσκονται στον υπολογιστή που θα τρέξει ο server, "https" για δημιουργία και χρήση ενός https server στην εφαρμογή, "path" για χρήση directory

paths (επειδή ο server τρέχει σε λειτουργικό Windows, χρησιμοποιούνται τα Windows-style paths), "express" για χρήση URL και "ws" για χρήση web sockets. Στην συνέχεια, αφού έχουν δημιουργηθεί, όπως παρουσιάστηκε παραπάνω, το κλειδί και το πιστοποιητικό, γίνεται ανάκτηση τους από τον φάκελο που βρίσκονται και αποθηκεύονται στην μεταβλητή **keyAndCert**. Τέλος, γίνεται χρήση τους για την δημιουργία του Server που βρίσκεται στην μεταβλητή webSocketServer.

```
let webClients = [];
webSocketServer.on('connection', function connection(ws)
{
    console.log("Connection Established");
    ws.on("message", function incoming(data)  {
        if (data.indexOf("webClient") !== -1)
        {
            webClients.push(ws);
            console.log("A new Web Client was added");
            return;
        }

        webClients.forEach((ws, i) => {
            if (webClients[i] == ws && ws.readyState === ws.OPEN)
            {
                ws.send(data);
            }
            else
            {
                webClients.splice(i, 1);
                console.log("A Client was Deleted");
            }
        });
    });

    //Error Handling
    ws.on("error", (type) => {
        console.error("An Error occurred on WebSocket ", type);
    });
});
```

Σχήμα 4.1.2.2. Δεύτερο block κώδικα στον server

Αφού οριστούν τα απαιτούμενα modules και οι σχετικές ρυθμίσεις, η ενέργεια που ακολουθεί στον κώδικα είναι η δημιουργία του Websocket Server. Χρησιμοποιείται ένας πίνακας με όνομα webClients, που περιέχει τους clients που είναι συνδεδεμένοι εκείνη την στιγμή στην σελίδα. Κάθε φορά που συνδέεται ένας client ή ένα ESP32-CAM, εμφανίζεται το ενημερωτικό μήνυμα "Connection Established" στο command prompt του powershell. Στην περίπτωση που συνδεθεί ένας WebClient εμφανίζεται επίσης το μήνυμα "A new Web Client was Added". Στην περίπτωση που ένας WebClient συνδεθεί στον server, τότε αυτός προστίθεται στον πίνακα με τους webClients που έχει δημιουργηθεί. Στην συνέχεια, τα δεδομένα

του ESP32-CAM που έχει συνδεθεί, στέλνονται σε κάθε webClient που υπάρχει στον πίνακα αυτό ώστε να μπορούν να εμφανιστούν. Όσο υπάρχουν διαθέσιμοι clients που μπορούν να λάβουν τα δεδομένα, αυτά αποστέλλονται σε αυτούς, ενώ στην περίπτωση που κάποιος client αποχωρεί, διαγράφεται από τον πίνακα των ενεργών client και εμφανίζεται το ενημερωτικό μήνυμα στο command prompt “A Client was Deleted”. Για τον server έχει γραφτεί κώδικας διαχείρισης σφαλμάτων. Στην περίπτωση κάποιου error, επιστρέφεται ο τύπος του λάθους και εμφανίζεται στο command prompt το μήνυμα “An Error occurred on WebSocket”, και στην συνέχεια τον κωδικό του σφάλματος.

```
app.use(express.static("."));
app.get("/secureClient", (req, res) => res.sendFile(path.resolve(__dirname, "./secureClient.html")));
secureServer.listen(443, () =>
    console.log(`HTTPS server is listening at secure port ${443}`));
```

Σχήμα 4.1.2.3. Τρίτο block κώδικα στον server

Αφού δοθούν τα παραπάνω χαρακτηριστικά, χρησιμοποιούμε την ασφαλή πόρτα 443 για να “ακούει” (listen) ο server από εκεί. Σε αυτή την πόρτα θα συνδεθούν οι clients και το ESP32-CAM. Πιο συγκεκριμένα, το ESP32-CAM θα συνδεθεί στην πόρτα 443 ενώ οι clients συνδέονται στην πόρτα 443 αλλά στο “/secureClient”. Προς ενημέρωση του χειριστή του server, εμφανίζεται το μήνυμα «HTTPS server is listening at secure port» και την πόρτα που ακούει, δηλαδή την 443. Για το γραφικό περιβάλλον που εμφανίζεται στον χρήστη, χρησιμοποιείται το path “/secureClient” που είναι κώδικας HTML και εξηγείται παρακάτω.

4.2 ARDUINO CODE

4.2.1 ΠΕΡΙΓΡΑΦΗ

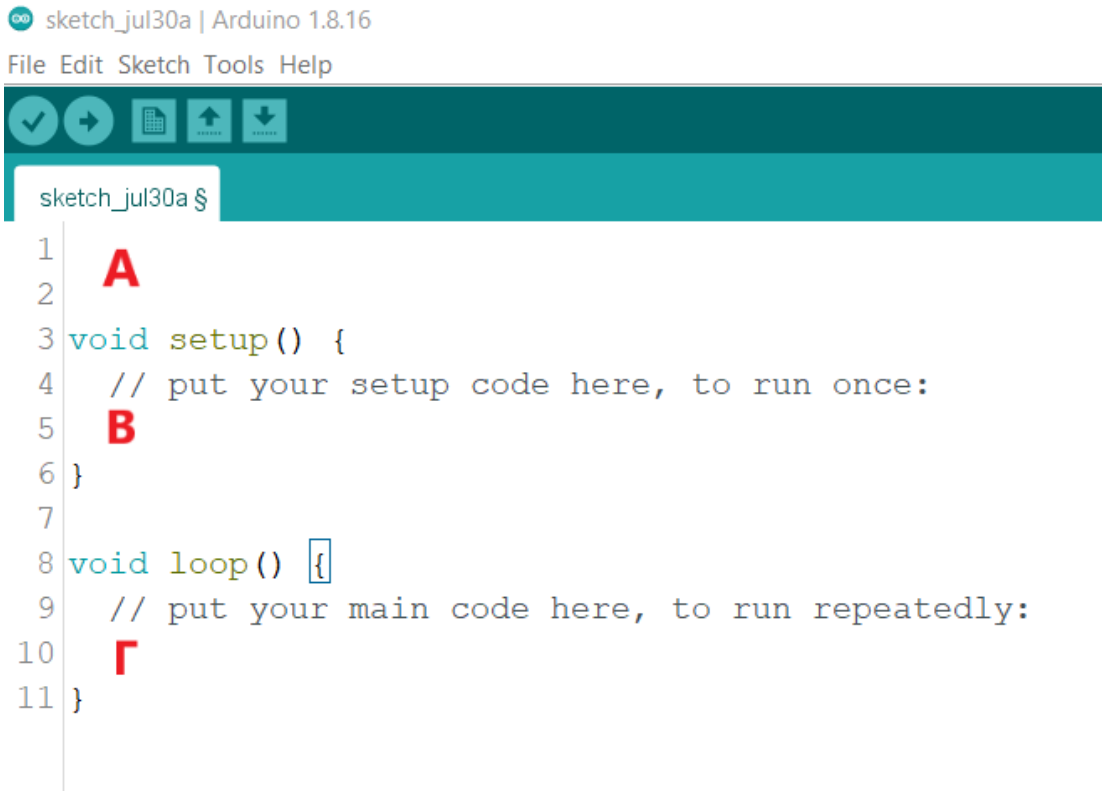
Στην ενότητα αυτή παρουσιάζεται η υλοποίηση του κώδικα που χρησιμοποιήθηκε στο ESP32-CAM. Για την ομαλή συγγραφή και μεταφορά του κώδικα, χρησιμοποιήθηκε το γραφικό περιβάλλον Arduino IDE (Integrated Development Environment)

Το Arduino IDE είναι ένα πρόγραμμα με το οποίο πραγματοποιείται η συγγραφή και μεταφορά του κώδικα από τον υπολογιστή, στον μικροελεγκτή που χρειάζεται. Διατίθεται δωρεάν από την επίσημη ιστοσελίδα του Arduino και παρέχεται σε όλα τα λειτουργικά συστήματα (Windows,Linux,Mac OS). Η

γλώσσα προγραμματισμού που χρησιμοποιείται σε αυτό είναι η C/C++ με την διαφορά των επιπλέον βιβλιοθηκών “libraries” που παρέχονται και των συναρτήσεων I/O. Επίσης, η “main”, ορίζεται ως “loop”.

Το IDE αυτό, προσφέρει διάφορα εργαλεία για την παρακολούθηση των διαδικασιών που υλοποιούνται και είναι αρκετά απλό στην χρήση και λειτουργία του. Μέσω των συναρτήσεων της κλάσης Serial, υπάρχει δυνατότητα σειριακής επικοινωνίας του μικροελεγκτή με τον υπολογιστή, με απώτερο σκοπό την ενημέρωση του προγραμματιστή για τυχόν σφάλματα αλλά και ελέγχου λειτουργίας του προγράμματος που υλοποιείται.

Εικόνα 4.2.1.1. Λογότυπο Arduino (https://el.m.wikipedia.org/wiki/%CE%91%CF%81%CF%87%CE%B5%CE%AF%CE%BF:Arduino_Logo.svg)



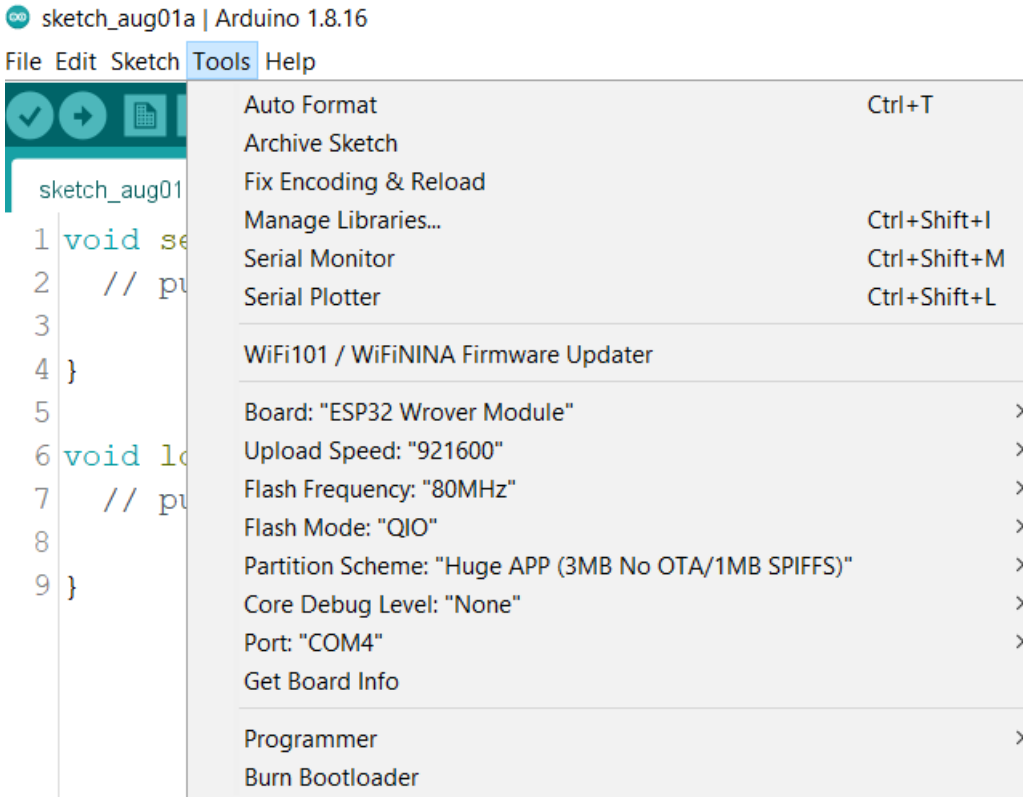
Εικόνα 4.2.1.2. Περιοχές εισαγωγής κώδικα

Περιοχή Α: Στην περιοχή αυτή γίνεται ο ορισμός των καθολικών μεταβλητών και σταθερών που θα χρησιμοποιηθούν στο πρόγραμμα. Εδώ ορίζονται και οι βιβλιοθήκες που θα χρησιμοποιηθούν παρακάτω.

Περιοχή Β (setup): Στην περιοχή αυτή, ορίζεται η λειτουργία των pin που θα χρησιμοποιηθούν (πχ pin εισόδου, εξόδου IO/pins), εδώ θα οριστούν οι ρυθμίσεις της κάμερας του ESP32-CAM. Επίσης σε αυτό το μέρος καλούνται οι συναρτήσεις που μπορεί να χρησιμοποιήσουν εξωτερικές βιβλιοθήκες που ορίζονται παραπάνω. Η συνάρτηση αυτή είναι void και καλείται για μία μόνο φορά στην αρχή της λειτουργίας του μικροελεγκτή. Οι περιπτώσεις αρχής λειτουργίας του μικροελεγκτή είναι δύο. Να ενεργοποιηθεί θέτοντας τον σε λειτουργεία μέσω παροχής ρεύματος, ή να πατηθεί εκούσια το κουμπί επαναφοράς (reset) που βρίσκεται πάνω του.

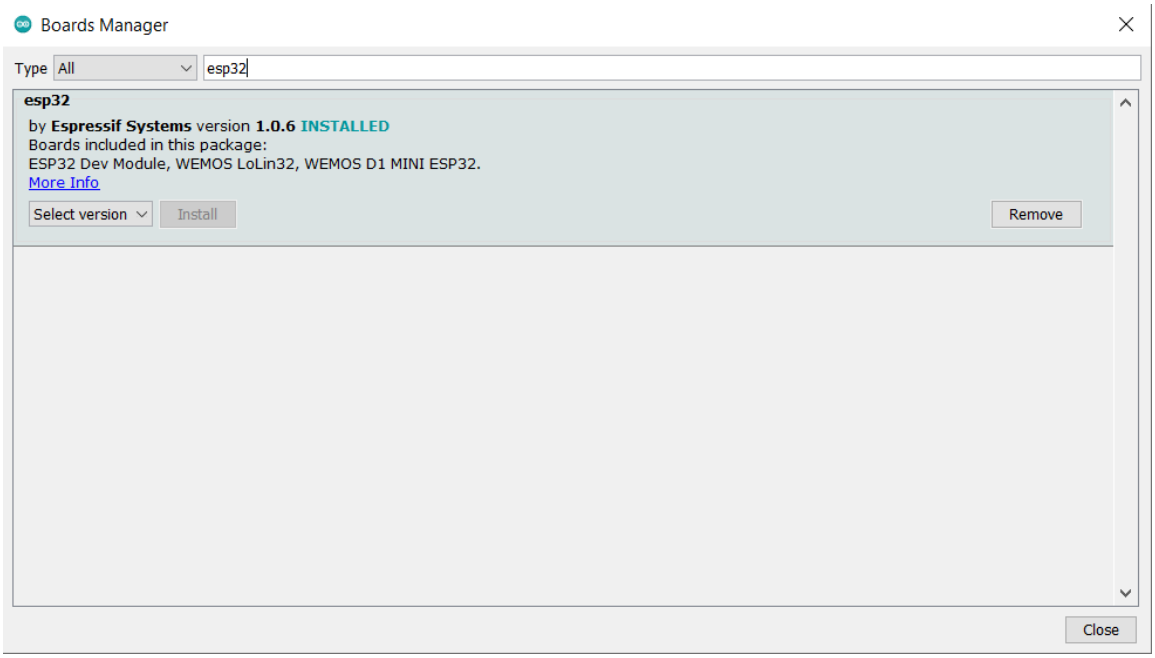
Περιοχή Γ (loop): Ο κώδικας που γράφεται εδώ, εκτελείται σε επανάληψη για όση διάρκεια λαβαίνει ρεύμα ο μικροελεγκτής. Μπορεί να γίνει ορισμός μεταβλητών και κλήση συναρτήσεων, όπως ακριβώς γίνεται σε ένα πρόγραμμα C/C++.

4.2.2 SETUP



Εικόνα 4.2.2.1. Ρυθμίσεις για επιτυχή σύνδεση

Ως βασικό παράθυρο ρυθμίσεων θα χρησιμοποιηθεί το Tools. Για τις ρυθμίσεις του Arduino IDE θα χρειαστεί αρχικά να γίνει εγκατάσταση του λογισμικού για το ESP32. Αυτό γίνεται μέσω της επιλογής Tools -> Board-> Boards Manager. Πέραν από την εγκατάσταση του λογισμικού, με αυτό τον τρόπο εγκαθιστούμε τις απαραίτητες βιβλιοθήκες που θα χρησιμοποιηθούν στον κώδικα. Η βιβλιοθήκη που απαιτείται είναι η “WebSocketsClient.h” και βρίσκεται στο πακέτο “arduinoWebSockets” (Εικόνα 4.2.2.1).



Εικόνα 4.2.2.2. Εισαγωγή βιβλιοθηκών

Έπειτα πρέπει να επιλεχθεί μέσω του Tools/Board ο μικροελεγκτής που θα χρησιμοποιηθεί. Στην περίπτωση μας επιλέγουμε το "ESP32 Wrover Module". Στη συνέχεια επιλέγουμε την ταχύτητα ανεβάσματος του κώδικα στον μικροελεγκτή "Upload Speed". Η ταχύτητα που επιλέχθηκε είναι "921600" και είναι η μεγαλύτερη δυνατή. Έχοντας συνδεδεμένο το ESP32-CAM σε μία θύρα USB του υπολογιστή μας, επιλέγουμε το Port που θα χρησιμοποιηθεί για να ανέβει ο κώδικας σε αυτό. Το Port που θα χρησιμοποιηθεί είναι το COM4.

4.2.3 ΚΩΔΙΚΑΣ

Για την συγγραφή του κώδικα χρησιμοποιήθηκε ως βάση το παράδειγμα που δίνεται από την βιβλιοθήκη του ESP32-CAM με όνομα "CameraWebServer". Το παράδειγμα αυτό μπορεί να βρεθεί με τον εξής τρόπο. File → Examples → ESP32 → Camera →CameraWebServer.

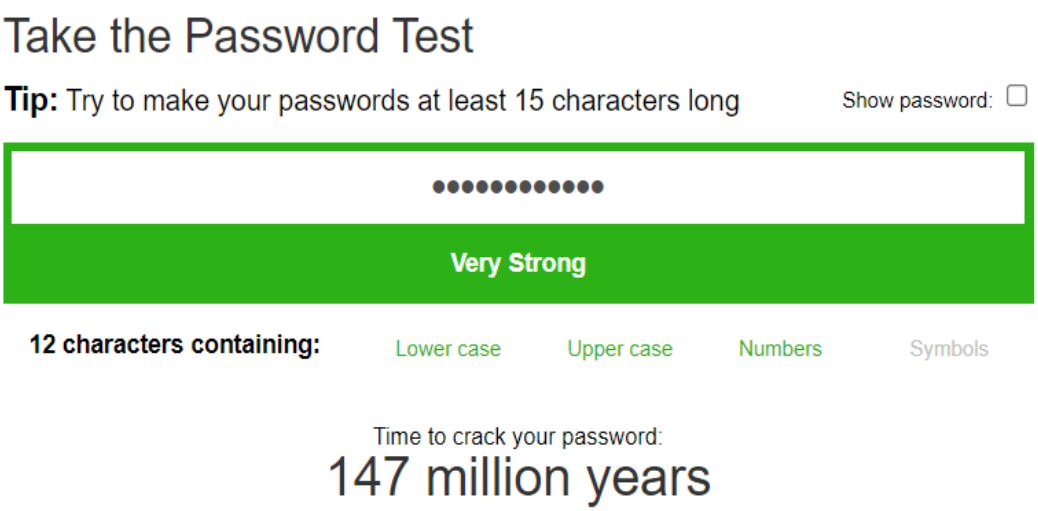
```
#include "esp_camera.h"
#include <WiFi.h>
#include <WebSocketsClient.h>
#define CAMERA_MODEL_AI_THINKER
#include "camera_pins.h"

const char* ssid = "*****";//WiFi Name
const char* password = "*****";//WiFi Password
const char* ipAddress = "*****";//IP Address
int securePort = 443;//Secure Port

WebSocketsClient client;
```

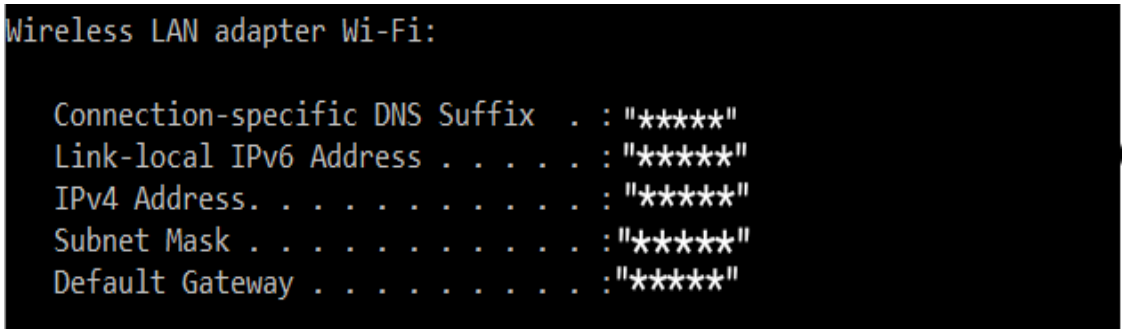
Σχήμα 4.2.3.1. Πρώτο block κώδικα στον μικροελεγκτή

Στην περιοχή Α, ορίζονται οι βιβλιοθήκες που θα χρησιμοποιηθούν. Οι βιβλιοθήκες “esp_camera.h” και “camera_pins.h” επιτρέπουν την αλληλεπίδραση με την κάμερα του ESP32. Επίσης με αυτές θα γίνει ο ορισμός των ρυθμίσεων της κάμερα που γίνεται παρακάτω. Η βιβλιοθήκη “WiFi.h” μας δίνει την δυνατότητα να συνδεθούμε σε ένα δίκτυο Wi-Fi. Το δίκτυο που θα γίνει η σύνδεση θα είναι μέσω του φορητού σημείου πρόσβασης hotspot από κινητή συσκευή. Θα χρησιμοποιηθούν τα WebSockets που παρέχονται από την βιβλιοθήκη “WebSocketsClient.h” ώστε να γίνει σύνδεση στον server. Στην συνέχεια ορίζονται οι μεταβλητές που δίνουν το όνομα του Wi-Fi, τον κωδικό, την IP διεύθυνση και την πόρτα σύνδεσης. Το όνομα και ο κωδικός ορίζονται από το smartphone.



Εικόνα 4.2.3.1. Υπολογισμός ασφάλειας κωδικού

Για την χρήση του φορητού σημείου πρόσβασης επιλέχθηκε να χρησιμοποιηθεί κωδικός που περιέχει τουλάχιστον 12 χαρακτήρες που αποτελείται από αλφαριθμητικούς χαρακτήρες (πεζούς η κεφαλαίους) και νούμερα. Σύμφωνα με την ιστοσελίδα https://www.passwordmonster.com/, ένας μέσος υπολογιστής χρειάζεται 147 εκατομμύρια χρόνια για να καταφέρει να βρει τον κωδικό αυτό.



Εικόνα 4.2.3.2. Εύρεση διεύθυνσης IP

Για την εύρεση της διεύθυνσης IP χρησιμοποιούμε την γραμμή εντολών CMD. Εκεί πληκτρολογείται η εντολή “ipconfig” που μέσω αυτής, βρίσκουμε την IP διεύθυνση που θα τρέξει ο server. Λόγω του ότι χρησιμοποιείται ασύρματο δίκτυο, η διεύθυνση βρίσκεται στον τομέα του Wireless LAN adapter Wi-Fi. Τα δεδομένα αποκρύφτηκαν για λόγους προσωπικών δεδομένων. Η διεύθυνση βρίσκεται στο μέρος IPv4 Address.

```
void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 15000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 40;
  config.fb_count = 2;
```

Σχήμα 4.2.3.2. Δεύτερο block κώδικα στον μικροελεγκτή

Στην περιοχή Β γίνεται ο ορισμός ρυθμίσεων της κάμερας του ESP32. Ορίζονται όλα τα Pins που υπάρχουν σε αυτό. Λόγω του ότι το μοντέλο ESP32-CAM που χρησιμοποιούμε είναι το AI_THINKER και έχει psram (pseudostatic random access memory), εξ ορισμού, για το μοντέλο που χρησιμοποιείται το frame size πρέπει να είναι τύπου VGA, η ποιότητα jpeg να είναι μεγέθους 40 και οι frame buffers να είναι 2.

```
if(psramFound()){
  Serial.println("Psram Found");
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 40;
  config.fb_count = 2;
} else {
  Serial.println("Psram Not Found");
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// camera initialisation
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK)
{
  Serial.printf("Camera init failed with error 0x%x", err);
  return;
}
```

Σχήμα 4.2.3.3. Τρίτο block κώδικα στον μικροελεγκτή

Στην συνέχεια γίνεται έλεγχος αν υπάρχει psram στον μικροελεγκτή που χρησιμοποιείται. Στην περίπτωση μας υπάρχει, όμως στην περίπτωση που κάποιος χρήστης χρησιμοποιήσει άλλο μοντέλο ESP32-CAM θα χρειαστεί να τροποποιηθεί το frame size, jpeg quality και το frame buffer count. Αυτές οι ρυθμίσεις εκτελούνται αυτόματα και εξ ορισμού από την αρχή υλοποίησης του προγράμματος. Μετά τον έλεγχο γίνεται αρχικοποίηση της κάμερας και έλεγχος αν η διαδικασία ολοκληρώθηκε. Στην περίπτωση που υπάρχει πρόβλημα τότε ο κωδικός του προβλήματος εμφανίζεται στο serial monitor του Arduino IDE.

```
    Serial.println("Attempting WiFi Connection");
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED)//Wait untill connected
    {
      delay(1000);
      Serial.print(" . ");
    }
    Serial.println("\n");
    Serial.println("WiFi connected");

    client.beginSSL(ipAddress, securePort, "/");
    client.setReconnectInterval(5000);
    client.enableHeartbeat(15000, 3000, 2);
}
```

Σχήμα 4.2.3.4. Τέταρτο block κώδικα στον μικροελεγκτή

Απόπειρα σύνδεσης του μικροελεγκτή με το φορητό σημείο πρόσβασης (hotspot). Στην περίπτωση που δεν γίνει απευθείας σύνδεση τότε γίνεται καθυστέρηση κατά ένα (1) δευτερόλεπτο (1000ms) και εμφάνιση μιας κουκίδας “.” ανά δευτερόλεπτο στο serial monitor. Αυτή η διαδικασία εκτελείται σε επανάληψη μέχρις ότου να πραγματοποιηθεί σύνδεση στο δίκτυο. Μόλις γίνει σύνδεση, εμφανίζεται το μήνυμα “WiFi Connected” προς ενημέρωση του χρήστη. Αφού επιτευχθεί σύνδεση στο δίκτυο, γίνεται σύνδεση στον ασφαλή server και στην πόρτα που έχει οριστεί στην αρχή του κώδικα. Στην περίπτωση που δεν επιτευχθεί σύνδεση, γίνεται επανάληψη της προσπάθειας κάθε πέντε (5) δευτερόλεπτα (5000ms). Έπειτα εκτελείται η εντολή heartbeat. Η εντολή αυτή στέλνει ping στον server κάθε δεκαπέντε (15) δευτερόλεπτα (150000ms) και αναμένει να λάβει απάντηση στο ping από τον server (pong) μέσα σε τρία (3) δευτερόλεπτα (3000ms). Στην περίπτωση που η απάντηση από τον server (pong) δεν ληφθεί παραπάνω από δύο (2) φορές, η σύνδεση θεωρείται ότι έχει λήξει και ότι δεν υπάρχει επικοινωνία με τον server.

```
void loop() {

  client.loop();
  camera_fb_t *fb = esp_camera_fb_get();
  if(!fb){
    Serial.println("Camera failed to capture");
    esp_camera_fb_return(fb);
    return;
  }

  //Setting up frame buffer
  fb->buf[12] = 0x01;

  client.sendBIN(fb->buf, fb->len);

  Serial.println("Image sent\n");
  Serial.print("Buffer Size: ");
  Serial.println((int)fb->buf);

  Serial.print("Buffer Length: ");
  Serial.println((int)fb->len);
  esp_camera_fb_return(fb);
}
```

Σχήμα 4.2.3.5. Πέμπτο block κώδικα στον μικροελεγκτή

Στην περιοχή Γ ως τελικό βήμα του κώδικα, ορίζεται ο frame buffer που θα περιέχει τα δεδομένα του βίντεο. Γίνεται έλεγχος για την περίπτωση που προκύψει κάποιο σφάλμα στον frame buffer και ενημέρωση μέσω του serial monitor. Από την στιγμή που έχει ρυθμιστεί ο frame buffer και δεν έχει εμφανιστεί κάποιο σφάλμα, αρχίζει η μετάδοση δεδομένων που αποστέλλονται σε δυαδική μορφή στον server. Τα δεδομένα που αποστέλλονται είναι ο frame buffer και το μέγεθος του. Προς ενημέρωση του χρήστη, τα δεδομένα αυτά εμφανίζονται στο serial monitor. Τέλος, ο frame buffer επιστρέφεται πίσω στον driver για να χρησιμοποιηθεί ξανά στην επόμενη επανάληψη.

4.2.4 ΕΛΕΓΧΟΣ ΚΑΙ ΜΕΤΑΦΟΡΑ ΚΩΔΙΚΑ ΣΤΟΝ ΜΙΚΡΟΕΛΕΓΚΤΗ

Για να πραγματοποιηθεί αυτή η διαδικασία, χρειάζεται αρχικά να χρησιμοποιηθεί ένα καλώδιο USB σε Micro-USB. Η αρσενική θύρα USB συνδέεται στον υπολογιστή, ενώ η αρσενική θύρα Micro-USB

συνδέεται στον μικροελεγκτή (ESP32-CAM). Όπως αναφέρθηκε παραπάνω, γίνεται η επιλογή του COM4 για την μεταφορά του κώδικα.

4.2.5 VERIFY-COMPILATION

Πριν την μεταφορά του κώδικα στον μικροελεγκτή είναι αναγκαίο να γίνει επαλήθευση και compilation του κώδικα.

Εικόνα 4.2.5.1. Κουμπιά verify-compilation

Μέσω της επιλογής verify, γίνεται η επαλήθευση και compilation του κώδικα. Ο μέσος χρόνος επαλήθευσης που μετρήθηκε από το πάτημα του κουμπιού μέχρι την πλήρη εκτέλεση επαλήθευσης είναι δεκαέξι (16) δευτερόλεπτα.

Εικόνα 4.2.5.2. Ενημερωτικά στοιχεία εκτέλεσης (compilation window)

Μετά την εκτέλεση του compilation μας επιστρέφονται κάποια ενημερωτικά στοιχεία. Τα bytes που χρησιμοποιήθηκαν στην μνήμη, η μνήμη που δεσμεύουν οι global μεταβλητές καθώς και το υπολειπόμενο ποσό μνήμης που είναι διαθέσιμο.

4.2.6 UPLOAD

Εικόνα 4.2.6.1. Επιλογή upload

Μέσω της επιλογής upload, ο κώδικας που έχει γραφεί γίνεται πρώτα verify και στην συνέχεια μεταφέρεται στον μικροελεγκτή μέσω του port που έχει οριστεί (COM4). Η αρχική διαδικασία που εκτελείται είναι η ίδια με την επιλογή Verify, μόνο που στην συνέχεια γίνεται η μεταφορά του κώδικα. Ο μέσος χρόνος εκτέλεσης της εντολής Upload από την στιγμή που εκτελείται μέχρι την στιγμή που έχει γίνει μεταφορά του κώδικα στον μικροελεγκτή ανέρχεται στα 34,5 δευτερόλεπτα.

Εικόνα 4.2.6.2. Μήνυμα επιτυχούς μεταφοράς κώδικα

Μόλις ληφθεί το παραπάνω μήνυμα, διαπιστώνεται ότι ο κώδικας έχει μεταφερθεί επιτυχώς στον μικροελεγκτή και είναι έτοιμος προς χρήση.

4.3 HTML + CSS

4.3.1 ΠΕΡΙΓΡΑΦΗ

Για την αλληλεπίδραση του χρήστη με την εφαρμογή, επιλέχθηκε να χρησιμοποιηθεί κώδικας HTML και για την περεταίρω μορφοποίηση του, CSS. Η υλοποίηση αυτή επιτρέπει στον χρήστη να δει το βίντεο που προβάλλεται αλλά και να αποθηκεύσει στιγμιότυπο του στον υπολογιστή. Εντός του κώδικα HTML έχει προστεθεί κώδικας javascript για την αλληλεπίδραση με τον server. Η υλοποίηση του έχει γίνει με χρήση του Visual Studio Code και ο κώδικας παρουσιάζεται παρακάτω.

4.3.2 HTML

```
<html>

<head>
    <title>Encrypted Video Streaming</title>
    <link rel="stylesheet" href="/serverStyles.css">
</head>

<body>
    <h1>ESP32-CAM ENCRYPTED VIDEO</h1>
        <div class="main">
            <img id="ESP32" src="" />
            <h2><b>Live When Dot Blinking <span class="sqr"
                id="redSqr" style="visibility: hidden;"></span></b></h2>
            <button class="button1" id="startStop" onclick="buttonStartStop()">Start</button>
        </div>
```

Σχήμα 4.3.2.1. Πρώτο block κώδικα HTML

Ως head στο HTML αρχείο, ορίζεται ο τίτλος που θα φαίνεται στο browser tab. Επίσης ορίζεται το link που οδηγεί στο αρχείο CSS που θα παρουσιαστεί παρακάτω. Στο body του HTML δίνεται το παράθυρο που θα εμφανίζεται το βίντεο που θα λαβαίνει ο server από τον μικροελεγκτή. Στην συνέχεια ορίζονται τα h1 και h2 tags. Το 1ο ορίζει τον αρχικό τίτλο που φαίνεται στον browser, ενώ στο 2ο χρησιμοποιείται ένα κόκκινο τετράγωνο που μέσω αυτού μπορεί να διακριθεί αν το βίντεο από τον μικροελεγκτή αποστέλλεται στον server και ο server μπορεί να το εμφανίσει. Τέλος, έχουν προστεθεί δύο κουμπιά “buttons”. Μέσω του 1ου (button1) ξεκινά και σταματά το βίντεο σε στιγμιότυπο του, ενώ με το 2ο (button2) μπορεί να γίνει αποθήκευση στιγμιότυπου του βίντεο σε εικόνα τύπου “.jpeg”. Στο 1ο κουμπί έχει δοθεί η δυνατότητα να σταματά το βίντεο για να έχει την ευκαιρία ο χρήστης να επιλέξει το στιγμιότυπο που επιθυμεί να αποθηκεύσει.

```
<script>
    const imageSequence = document.getElementById("ESP32");
    const WS_URL = 'wss://' + window.location.hostname;
    const ws = new WebSocket(WS_URL);
    let url;

    window["startStop"] = false;


    function buttonStartStop()
    {
        var x = document.getElementById("startStop");
        if (x.innerHTML === "Start")
            {
                x.innerHTML = "Stop";
                window["startStop"] = true;
            }
        else
            {
                x.innerHTML = "Start";
                window["startStop"] = false;
            }
    }
```

Σχήμα 4.3.2.2. Δεύτερο block κώδικα HTML

Στο αρχικό μέρος του script που βρίσκεται μέσα στον κώδικα HTML ορίζονται οι σταθερές που θα χρησιμοποιηθούν παρακάτω. Η μεταβλητή imageSequence θα περιέχει το κάθε frame του video που θα αποστέλλεται. Στην συνέχεια ορίζεται το websocket url και μέσω αυτού, δημιουργείται νέο WebSocket, η μεταβλητή url θα χρησιμοποιηθεί παρακάτω. Ως πρώτη συνάρτηση ορίζεται η συνάρτηση buttonStartStop που μέσω αυτής, κάθε φορά που πατιέται το κουμπί “button1”, αλλάζει η ένδειξη που αναγράφεται σε αυτό από “Start” σε “Stop” και αντίστροφα. Στην περίπτωση που το κουμπί έχει την ένδειξη “Start”, τότε η ένδειξη αυτή μεταβάλλεται σε “Stop”, ενώ αντίστροφα όταν το κουμπί έχει την ένδειξη “Stop”, η ένδειξη αυτή μεταβάλλεται σε “Start”

```
function hideRedSqr()
{
    document.getElementById("redSqr").style.visibility = "hidden";
}

setInterval(hideRedSqr, 1000);
```

Σχήμα 4.3.2.5. Τρίτο block κώδικα HTML

Η συνάρτηση hideRedSqr παρέχει την δυνατότητα να σβήνεται ο το κόκκινο τετράγωνο που μέσω αυτού ο χρήστης αναγνωρίζει την σύνδεση του μικροελεγκτή στον server. Μέσω της εντολής setInterval,

η συνάρτηση αυτή εκτελείται κάθε ένα (1) δευτερόλεπτο (1000ms). Με αυτό τον τρόπο δίνεται στον χρήστη η οπτική ότι το κόκκινο τετράγωνο αναβοσβήνει.

```
        ws.onopen = () =>
        {
            ws.send("webClient");
        };

        ws.onmessage = async (message) =>
        {
            var binData = new Blob([message.data]);
            document.getElementById("redSqr").style.visibility = "visible";
            if (!window["startStop"]) return;

            if (url)
            {
                URL.revokeObjectURL(url);
            }
            url = URL.createObjectURL(binData);
            imageSequence.src = url;
        }
    </script>
</body>

</html>
```

Σχήμα 4.3.2.6. Τέταρτο block κώδικα HTML

Μέσω της μεθόδου send, τα δεδομένα του βίντεο αποστέλλονται στον server από την websocket σύνδεση που έχει οριστεί παραπάνω. Η συνέχεια του κώδικα ενεργοποιείται στην περίπτωση που ληφθούν δεδομένα μέσω του websocket. Τα δεδομένα βρίσκονται στην μεταβλητή message, στην συνέχεια μετατρέπονται σε δυαδική μορφή στη μεταβλητή binData για χρήση τους παρακάτω. Λόγω του ότι έχουν ληφθεί δεδομένα, ενεργοποιείται ο κόκκινος κύκλος προς ένδειξη στον χρήστη ότι υπάρχει επιτυχία σύνδεσης και ομαλή λειτουργία. Στην περίπτωση που έχει δημιουργηθεί ήδη από προηγούμενη κλήση της .onmessage, με την χρήση της στατικής μεθόδου .revokeObjectURL, απελευθερώνεται η υπάρχουσα διεύθυνση URL του αντικειμένου που υπήρχε. Στην συνέχεια δημιουργείται νέο url με τα νέα δεδομένα που έχουν ληφθεί παραπάνω. Ο κώδικας javascript αλλά και HTML σταματά εδώ.

4.3.3 CSS

Το ακρώνυμο CSS (Cascading Style Sheets) είναι μία γλώσσα σχεδιασμού που κάνει μία εφαρμογή να φαίνεται πιο ελκυστική σε σχέση με απλά κομμάτια κειμένου που θα παρουσιάζονταν χωρίς την χρήση

της. Η γλώσσα αυτή καθορίζει την οπτική δομή, διάταξη και αισθητική. Η προσθήκη CSS έγινε για την αύξηση εμπειρίας του χρήστη στην εφαρμογή και την διευκόλυνση του στις βασικές λειτουργίες της.

```css
html {
    font-size: 16px;
    background-color: #0385e9;
}

h1 {
    font-family: 'Times New Roman';
    text-align: center;
}

body {
    color: rgb(255, 255, 255);
    padding: 1rem;
}
```

Σχήμα 4.3.3.1. Πρώτο block κώδικα CSS

Ως αρχή στο CSS αρχείο, ορίζουμε τις βασικές ρυθμίσεις για τον HTML κώδικα. Το αρχικό font-size ορίζεται σε 16px ενώ το χρώμα του background ορίζεται σε μπλε με κωδικό #0385e9. Το πρώτο tag (h1) ορίζεται να απεικονιστεί στο κέντρο της οθόνης με γραμματοσειρά “Times New Roman”. Το body ορίζεται σε άσπρο χρώμα με padding 1rem. Λόγω του ότι έχει οριστεί σε άσπρο και βρίσκεται πίσω από τα tags h1 και h2, αυτά απεικονίζονται σε άσπρο χρώμα.

```css
.main {

  display: flex;
  justify-content: center;
  align-items: center;
  text-align: center;
  min-height: 100vh;
  max-width: 1400px;
  margin: 0 auto;
}
```

Σχήμα 4.3.3.2. Δεύτερο block κώδικα CSS

Το στοιχείο (element) main αναπαριστά την κλάση “main” που βρίσκεται στον κώδικα HTML. Η κλάση αυτή περιέχει μέσα της το h2 tag, την κλάση “sqr” αλλά και τις δύο κλάσεις των buttons. Είναι η κύρια κλάση και μέσα σε αυτήν υπάρχει επίσης το τετράγωνο που αναπαριστά το βίντεο που προβάλλεται στον χρήστη από το ESP32-CAM.

```css
.button1 {
    background-color: #f32718;
    display: inline-block;
    padding: 12px 33px;
    border: dotted dashed ;
    color: white;
    text-align: center;
    text-decoration: none;
    font-size: 16px;
}
```

Σχήμα 4.3.3.3. Τρίτο block κώδικα CSS

Οι κλάσεις button1 και button2 που αναπαριστούν το Play/Pause και Save Button αντίστοιχα, για το χρώμα του Play/Pause ορίζεται το κόκκινο με κωδικό #f32818, ενώ το χρώμα του Save Button ορίζεται σε πράσινο με κωδικό #0ff01b. Το χρώμα της γραμματοσειράς που υπάρχει πάνω στα κουμπιά ορίζεται σε άσπρο. Οι υπόλοιποι ορισμοί γίνονται για τον ορισμό της αναπαράστασης των κουμπιών στην σελίδα αλλά και την εξωτερική τους εμφάνιση. Τα κουμπιά έχουν το ίδιο μέγεθος 12x33, δηλαδή ορθογώνιου σχήματος.

```css
.sqr {
    height: 23px;
    width: 23px;
    background-color: #f32718;
    display: inline-block;
}
```

Σχήμα 4.3.3.4. Τέταρτο block κώδικα CSS

Στο τελευταίο κομμάτι του κώδικα βρίσκεται το στοιχείο sqr και αναπαριστά το κόκκινο τετράγωνο που αναβοσβήνει και χρησιμοποιείται προς ενημέρωση του χρήστη ότι έχει γίνει σύνδεση του μικροελεγκτή και υπάρχει βίντεο που μπορεί να προβληθεί. Το ύψος και το πλάτος έχουν οριστεί σε 23px που σημαίνει ότι πρόκειται για τετράγωνο. Ως χρωματική απόχρωση έχει οριστεί η ίδια με του κουμπιού Play/Pause.

5. ΔΟΚΙΜΕΣ

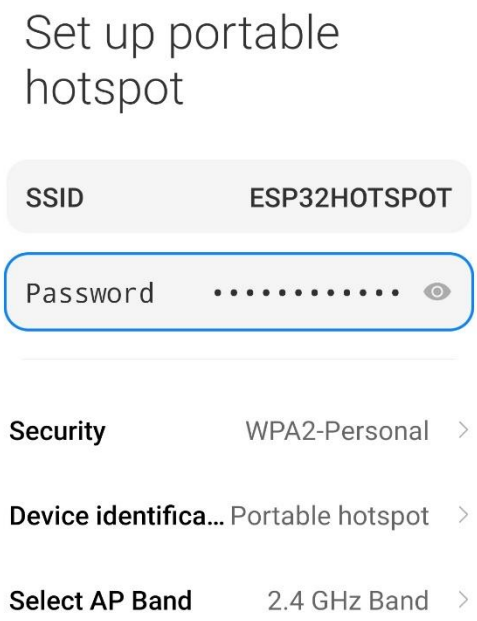
5.1 ΕΙΣΑΓΩΓΗ

Για την εκκίνηση χρήσης της εφαρμογής είναι απαραίτητο να εκτελεστούν κάποιες βασικές ενέργειες. Αρχικά χρειάζεται να γίνει η σύνδεση του υπολογιστή που θα εκκινήσει ο server στο δίκτυο hotspot. Στην συνέχεια πρέπει να γίνει εκκίνηση του server και έπειτα σύνδεση του μικροελεγκτή με τον server, τέλος να γίνει σύνδεση του χρήστη στην IP διεύθυνση που γίνεται η προβολή του video. Αυτές οι ενέργειες παρουσιάζονται με βήματα, αναλυτικά σε αυτό το κεφάλαιο. Στην συνέχεια γίνονται οι δοκιμές της εφαρμογής.

5.2 ΕΚΚΙΝΗΣΗ ΛΕΙΤΟΥΡΓΙΑΣ

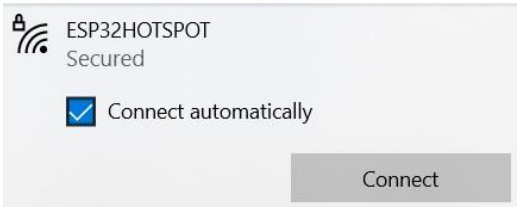
5.2.1 ΣΥΝΔΕΣΗ ΦΟΡΗΤΟΥ ΣΗΜΕΙΟΥ ΠΡΟΣΒΑΣΗΣ (HOTSPOT)

Για την σύνδεση του hotspot θα γίνει χρήση κινητού τηλεφώνου τύπου smartphone. Το τηλέφωνο που χρησιμοποιήθηκε για την υλοποίηση είναι το Xiaomi Redmi Note 7.

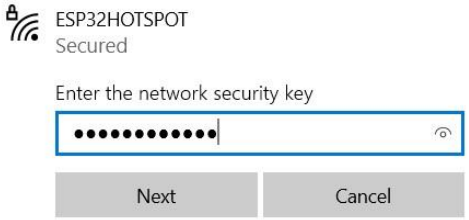


Εικόνα 5.2.1.1. Ρυθμίσεις για δημιουργία hotspot

Μέσω των ρυθμίσεων του τηλεφώνου επιλέγουμε όνομα και κωδικό. Το όνομα και ο κωδικός πρέπει να γραφούν στον κώδικα Arduino που θα μεταφερθεί στον μικροελεγκτή. Όπως αναλύθηκε στο κεφάλαιο 4.2, ο κωδικός που χρησιμοποιείται αποτελείται από δώδεκα αλφαριθμητικούς χαρακτήρες (πεζούς η κεφαλαίους) και νούμερα. Το ποσοστό χαρακτήρων και νούμερων που υπάρχουν ανάμεσα σε αυτούς τους δώδεκα χαρακτήρες είναι τυχαίο.

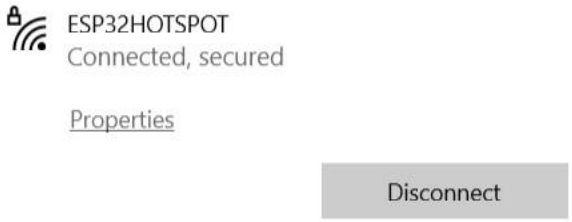


Εικόνα 5.2.1.2. Σύνδεση σε Wi-Fi



Εικόνα 5.2.1.3. Εισαγωγή κωδικού

Στην συνέχεια απαιτείται να γίνει σύνδεση του υπολογιστή με το φορητό σημείο πρόσβασης που μόλις ρυθμίστηκε. Μέσω των ρυθμίσεων δικτύου Wi-Fi των Windows, αναζητείται και βρίσκεται το hotspot (αριστερή εικόνα). Αφού το δίκτυο αναγνωρίζεται από το σύστημα, γίνεται απόπειρα σύνδεσης με την εισαγωγή του δωδεκαψήφιου κωδικού που έχει οριστεί στο δίκτυο hotspot (εικόνα δεξιά).



Εικόνα 5.2.1.4. Επιτυχία σύνδεσης

Με την παραπάνω ένδειξη στο σύστημα διαπιστώνεται ότι έχει γίνει επιτυχής σύνδεση στο δίκτυο και έχει δοθεί IP διεύθυνση. Για την εύρεση αυτής υπάρχει αναλυτική περιγραφή στο κεφάλαιο 4.2.

5.2.2 ΕΚΚΙΝΗΣΗ ΤΟΥ SERVER

Για την εκκίνηση του θα χρησιμοποιηθεί το terminal που παρέχεται στο visual studio code. Εκτός αυτού, η εκκίνηση του μπορεί να γίνει και από το powershell των Windows, με την προϋπόθεση το command line του powershell να είναι ανοιχτό στον φάκελο που βρίσκεται ο server. Αυτό γίνεται αυτόματα με το terminal του visual studio code, που μεταφέρει τον προγραμματιστή απευθείας στον φάκελο που βρίσκεται οποιοδήποτε αρχείο επιλεχθεί.

```
node server
```

Σχήμα 5.2.2.1. Εντολή εκκίνησης σε powershell

Με την παραπάνω εντολή στο powershell, εκκινεί ο server την λειτουργία του, για τον έλεγχο ορθής εκκίνησης εμφανίζεται το μήνυμα που αναλύεται στο παρακάτω σχήμα.

```
HTTPS server is listening at secure port 443
```

Σχήμα 5.2.2.2. Πρώτη απάντηση από τον server

Ως απάντηση από τον server έπειτα από την εντολή εκκίνησης λαβαίνουμε το παραπάνω μήνυμα. Από την στιγμή που αυτό ληφθεί σημαίνει ότι ο server έχει εκκινήσει και λειτουργεί ομαλά. Είναι έτοιμος να δεχτεί την σύνδεση του μικροελεγκτή.

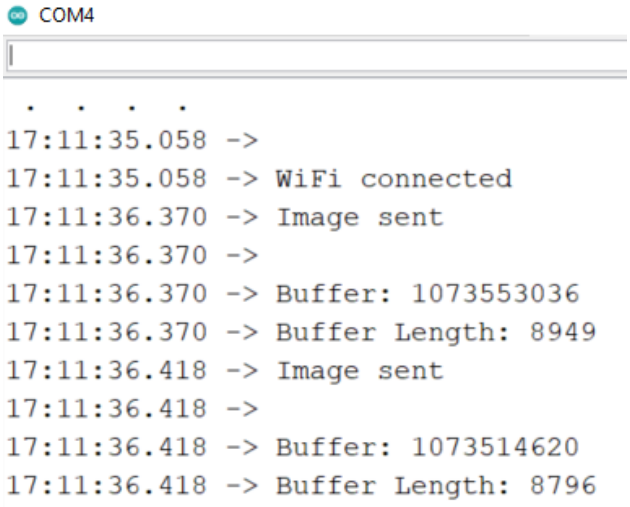
5.2.3 ΣΥΝΔΕΣΗ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ ΣΤΟΝ SERVER

Εικόνα 5.2.3.1. Σύνδεση μικροελεγκτή σε power bank

Για την σύνδεση του μικροελεγκτή με τον server απαιτείται να δοθεί τροφοδοσία ρεύματος σε αυτόν. Για τον λόγο του ότι η μελέτη έχει γίνει για την χρήση του σε Drone, και όπως αναφέρθηκε στο κεφάλαιο κόστους υλοποίησης, η τροφοδοσία του θα γίνει μέσω ενός power bank χωρητικότητας 2000mAh. Το μόνο που χρειάζεται για την αρχή τροφοδοσίας του μικροελεγκτή είναι η σύνδεση του με καλώδιο micro USB στο power bank. Παρατηρούμε ότι το power bank παρέχει ρεύμα στον μικροελεγκτή λόγω του ότι είναι αναμμένο το led φως μπλε χρώματος. Επίσης από την στιγμή που παρατηρείται ότι το κόκκινο φως του μικροελεγκτή είναι αναμμένο, διαπιστώνεται ότι δέχεται ρεύμα και λειτουργεί.

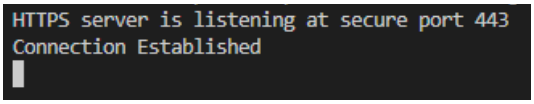
5.2.4 SERIAL MONITOR

Για την παρακολούθηση της ομαλής λειτουργίας του ESP32-CAM και του τρόπου σύνδεσης του στον server, θα γίνει σύνδεση σε υπολογιστή, ώστε μέσω του Serial Monitor από το Arduino IDE να υπάρξει αναπαράσταση δεδομένων από αυτό. Αυτό γίνεται μόνο για σκοπούς ανάλυσης και δεν αποτελεί μέρος της υλοποίησης λόγω του ότι ο αρχικός σχεδιασμός του είναι η χρήση του σε Drone.



Σχήμα 5.2.4.1. COM4 output

Από την στιγμή που γίνεται σύνδεση σε πηγή ρεύματος, αρχίζει η διαδικασία σύνδεσης στο φορητό σημείο πρόσβασης. Τα σύμβολα "." Αναπαριστούν την προσπάθεια σύνδεσης σε αυτό. Όπως αναλύθηκε στο κομμάτι του κώδικα Arduino, κάθε τελεία, απέχει ένα (1) δευτερόλεπτο (1000ms) από την επόμενη. Με αυτό συμπεραίνεται ότι απαιτήθηκαν περίπου τέσσερα δευτερόλεπτα από την στιγμή που έγινε η εκκίνηση του μικροελεγκτή μέχρι να συνδεθεί στο δίκτυο. Αμέσως μετά γίνεται η αρχή μετάδοσης των δεδομένων.



Σχήμα 5.2.4.2. Δεύτερη απάντηση από τον server

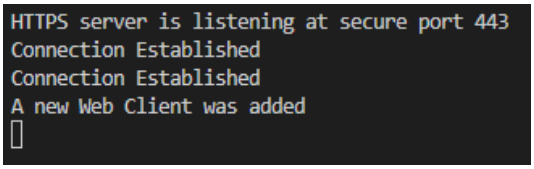
Την ίδια στιγμή που λαβαίνουμε το μήνυμα Wi-Fi connected από το serial monitor του Arduino IDE, λαβαίνουμε επίσης την ενημέρωση από τον server ότι έχει γίνει μία σύνδεση σε αυτόν.

5.2.5 ΣΥΝΔΕΣΗ CLIENT ΣΤΟΝ SERVER

Η σύνδεση του client μπορεί να γίνει από οποιονδήποτε browser υπάρχει διαθέσιμος (Google Chrome, Mozila,Microsoft Internet Explorer κλπ.). Για την επίδειξη της υλοποίησης επιλέχθηκε να γίνει σύνδεση με Mozila Firefox.

Για την σύνδεση του client, απαιτείται να γραφτεί στο URL, το πρωτόκολλο https που θα πραγματοποιηθεί, η IP διεύθυνση που έχει οριστεί και βρεθεί μέσω της γραμμής εντολών windows (command prompt, κεφάλαιο 4.2). Έπειτα από την διεύθυνση είναι απαραίτητο να γραφεί το path που θα

οδηγήσει τον client στον html κώδικα που έχει οριστεί. Για να γίνει αυτό, έπειτα από την IP διεύθυνση, προστίθεται στο URL το "/secureClient". Ως σύνολο, η όψη του URL που θα γραφεί είναι η "https://IP_ΔΙΕΥΘΥΝΣΗ/secureClient".

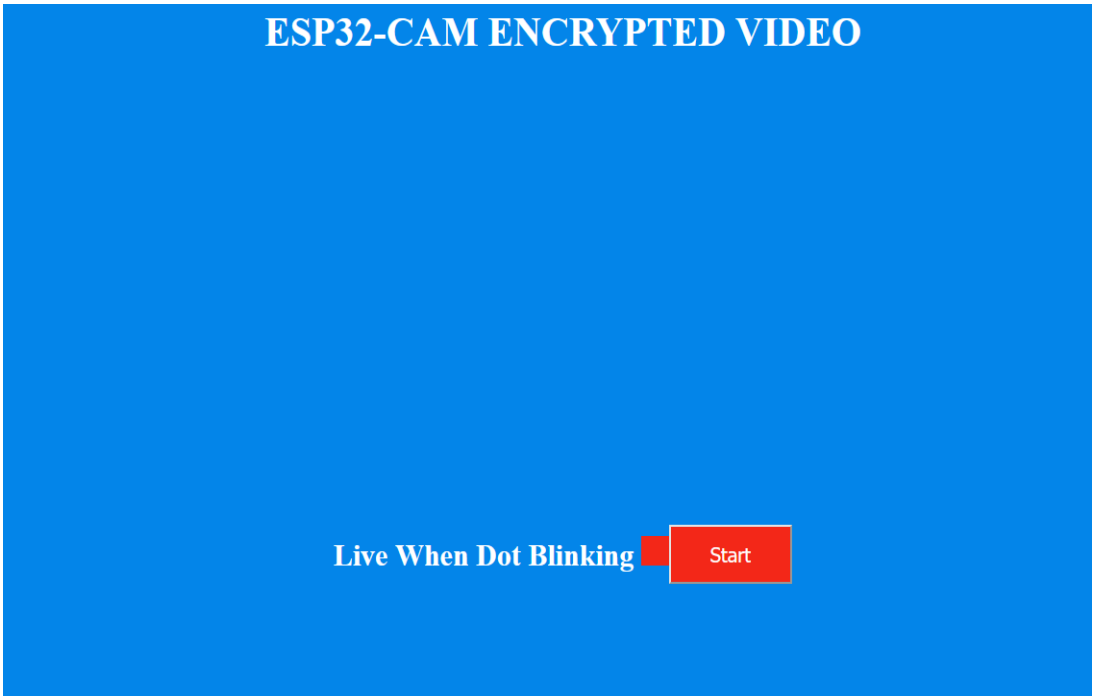


Σχήμα 5.2.5.1. Επιτυχία σύνδεσης client

Για να διαπιστωθεί αν έχει γίνει επιτυχής σύνδεση του client, αναμένεται ένα δεύτερο μήνυμα "Connection Established" και στην συνέχεια το μήνυμα "A new Web Client was added". Όπως διαπιστώνεται η σύνδεση πραγματοποιήθηκε επιτυχώς από μεριάς του client.

5.3 ΣΤΟΙΧΕΙΑ HTML/CSS

5.3.1 ΑΡΧΙΚΗ ΣΕΛΙΔΑ



Εικόνα 5.3.1.1. Αρχική σελίδα προβολής βίντεο

Αφού η σύνδεση έχει γίνει επιτυχώς, εμφανίζεται η βασική σελίδα HTML μαζί με τον κώδικα CSS που δημιουργήθηκε. Το κόκκινο τετράγωνο που βρίσκεται στο κάτω μέρος της, αναβοσβήνει, και αυτό σημαίνει ότι ο μικροελεγκτής έχει συνδεθεί επιτυχώς και στέλνει δεδομένα στον server. Βρισκόμαστε ουσιαστικά σε ζωντανή σύνδεση, για αυτό και προστέθηκε το μήνυμα “Live When Dot Blinking”. Όλα κυλούν ομαλά και η εκτέλεση της υλοποίησης συνεχίζεται στο επόμενο βήμα.

5.3.2 ΠΡΟΒΟΛΗ ΒΙΝΤΕΟ (START)

Εικόνα 5.3.2.1. Προβολή Βίντεο “Start”

Για την προβολή των δεδομένων που στέλνει ο μικροελεγκτής στον server, πρέπει να επιλεχθεί με το ποντίκι το κουμπί Start. Κατά την επιλογή του, εμφανίζεται το βίντεο. Το βίντεο αυτό είναι συνεχές. Επίσης παρατηρείται ότι τώρα, το κουμπί “Start”, έχει μετατραπεί σε κουμπί “Stop”. Αυτό δίνει την δυνατότητα στον χρήστη να σταματήσει το βίντεο στο τρέχον στιγμιότυπο.

5.4 ΒΙΝΤΕΟΣΚΟΠΗΣΗ ΚΑΙ ΑΠΟΘΗΚΕΥΣΗ

5.4.1 ΕΙΣΑΓΩΓΗ

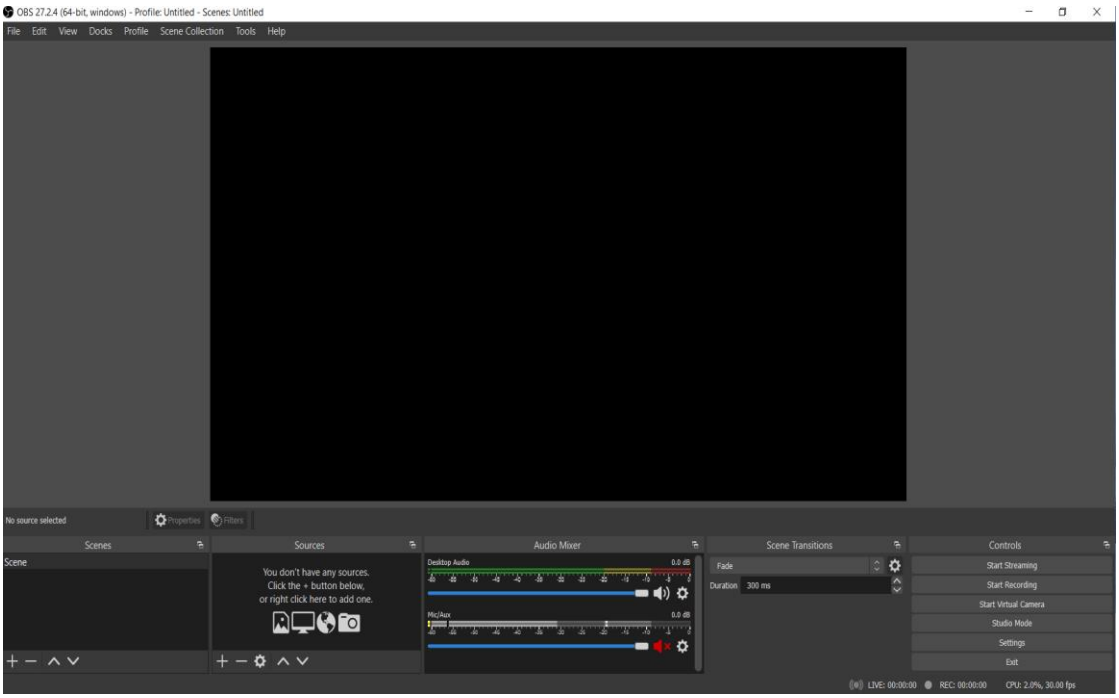
Για την βιντεοσκόπηση και αποθήκευση των όσων καταγράφονται από τον μικροελεγκτή, επιλέχθηκε να χρησιμοποιηθεί, πρόγραμμα καταγραφής βίντεο μέσω του υπολογιστή. Με αυτή την μέθοδο, καθίσταται πιο δύσκολη η υποκλοπή του, αφού το βίντεο καταγράφεται και αποθηκεύεται κατευθείαν στον υπολογιστή, χωρίς να υπάρχει ανάγκη να αποσταλεί κάπου και να μεταδοθεί.

Το πρόγραμμα που επιλέχθηκε είναι το OBS Studio (Open Broadcaster Studio). Είναι ένα δωρεάν πρόγραμμα τύπου open source γραμμένο σε C/C++ και επιτρέπει την καταγραφή οθόνης του υπολογιστή σε πραγματικό χρόνο. Το πρόγραμμα αυτό είναι διαθέσιμο και στα τρία λειτουργικά συστήματα (Windows, Linux, Mac OS), κάτι που το κάνει ακόμη πιο προσιτό και εύχρηστο.

Εικόνα 5.4.1.1. Λογότυπο OBS Studio
(https://obsproject.com/)

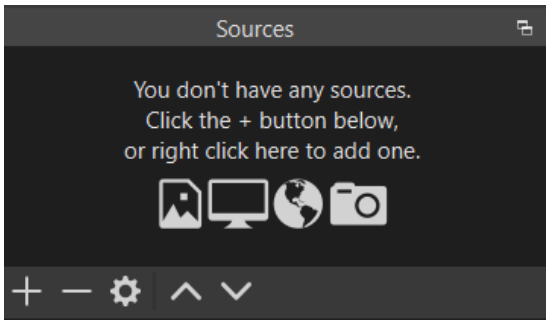
5.4.2 ΡΥΘΜΙΣΕΙΣ ΕΦΑΡΜΟΓΗΣ

Μετά την επιτυχή λήψη και εγκατάσταση του προγράμματος είναι απαραίτητη η εκτέλεση ρυθμίσεων για να μπορέσει να βιντεοσκοπηθεί το κομμάτι της οθόνης που μας ενδιαφέρει. Στην περίπτωση αυτή, το κομμάτι που απαιτείται να βιντεοσκοπηθεί είναι το μέρος που εμφανίζεται το βίντεο στην σελίδα HTML (βλέπε Εικόνα 5.3.2.1).

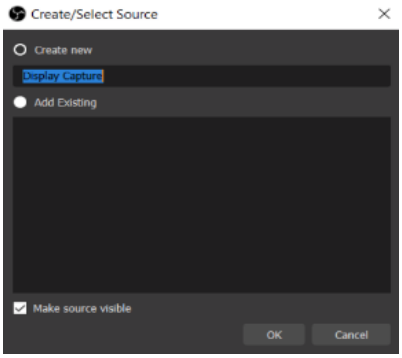


Εικόνα 5.4.2.1. Αρχική οθόνη OBS Studio

Στην παραπάνω εικόνα απεικονίζεται η αρχική οθόνη του προγράμματος. Για την βιντεοσκόπηση απαιτείται να εκτελεστούν οι ρυθμίσεις που επιδεικνύονται παρακάτω.

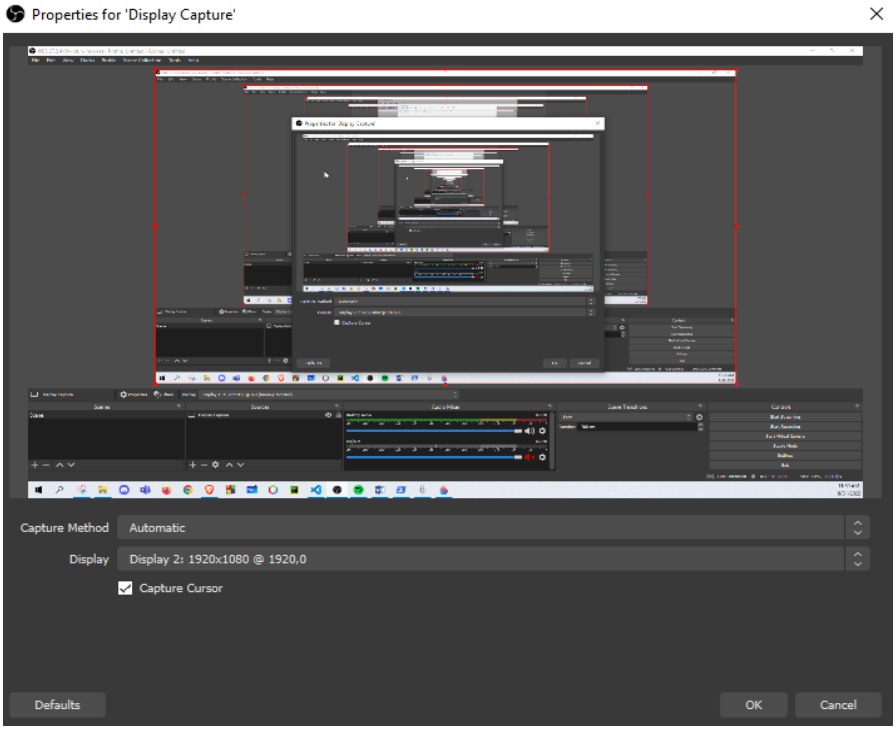


Εικόνα 5.4.2.2. Παράθυρο “Sources”



Εικόνα 5.4.2.3. Display Capture

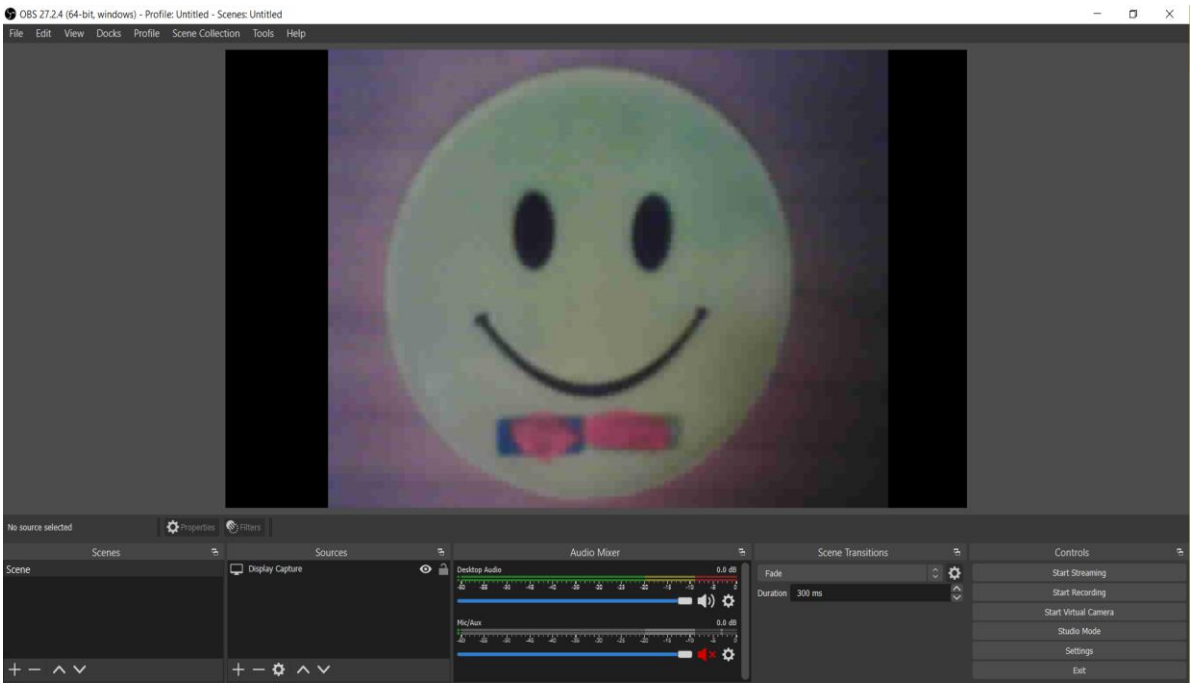
Μέσω του παραθύρου “Sources”, που βρίσκεται στο κάτω μέρος της αρχικής οθόνης, και του συμβόλου “+”, επιλέγεται ένα νέο display capture. Δίνεται όνομα σε αυτό και δημιουργείται με το πάτημα του κουμπιού “OK”.



Εικόνα 5.4.2.4. Συνέχεια ρυθμίσεων βιντεοσκόπησης

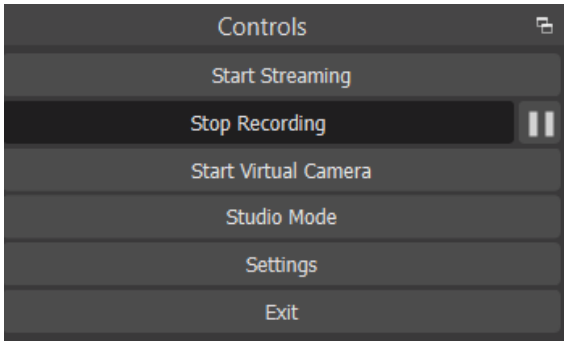
Στην συνέχεια των ρυθμίσεων ο χρήστης έχει την επιλογή να διαλέξει ποια οθόνη θα χρησιμοποιήσει ώστε να γίνει βιντεοσκόπηση. Στην περίπτωσή μας έχουμε δύο οθόνες και επιλέχθηκε η δεύτερη. Το τελευταίο βήμα που απαιτείται είναι να επιλεχθεί το κομμάτι της οθόνης που χρειάζεται θα βιντεοσκοπηθεί. Κρατώντας παρατεταμένα το κουμπί Alt, έχουμε την δυνατότητα να επιλέξουμε μόνο το κομμάτι που επιθυμούμε μέσα σε όλη την οθόνη. Το αποτέλεσμα φαίνεται παρακάτω.

5.4.3 ΒΙΝΤΕΟΣΚΟΠΗΣΗ



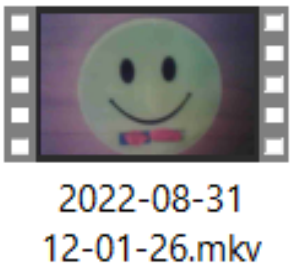
Εικόνα 5.4.3.1. Επιλογή μέρους οθόνης για βιντεοσκόπηση

Αφού εκτελέστηκε το κομμάτι των ρυθμίσεων και έχει επιλεχθεί το μέρος της οθόνης που θα βιντεοσκοπηθεί, το πρόγραμμα βρίσκεται σε θέση να ξεκινήσει την βιντεοσκόπηση. Αυτό γίνεται με το πάτημα του κουμπιού “Start” που βρίσκεται στο κάτω δεξιά μέρος του προγράμματος και στην ενότητα “Controls”



Εικόνα 5.4.3.2. Διακοπή βιντεοσκόπησης

Μετά το πάτημα του κουμπιού “Start”, παρατηρείται ότι έχει ξεκινήσει η βιντεοσκόπηση. Για να σταματήσει, πατιέται το κουμπί Stop Recording, που πριν την εκκίνηση βιντεοσκόπησης είχε την ονομασία “Start”



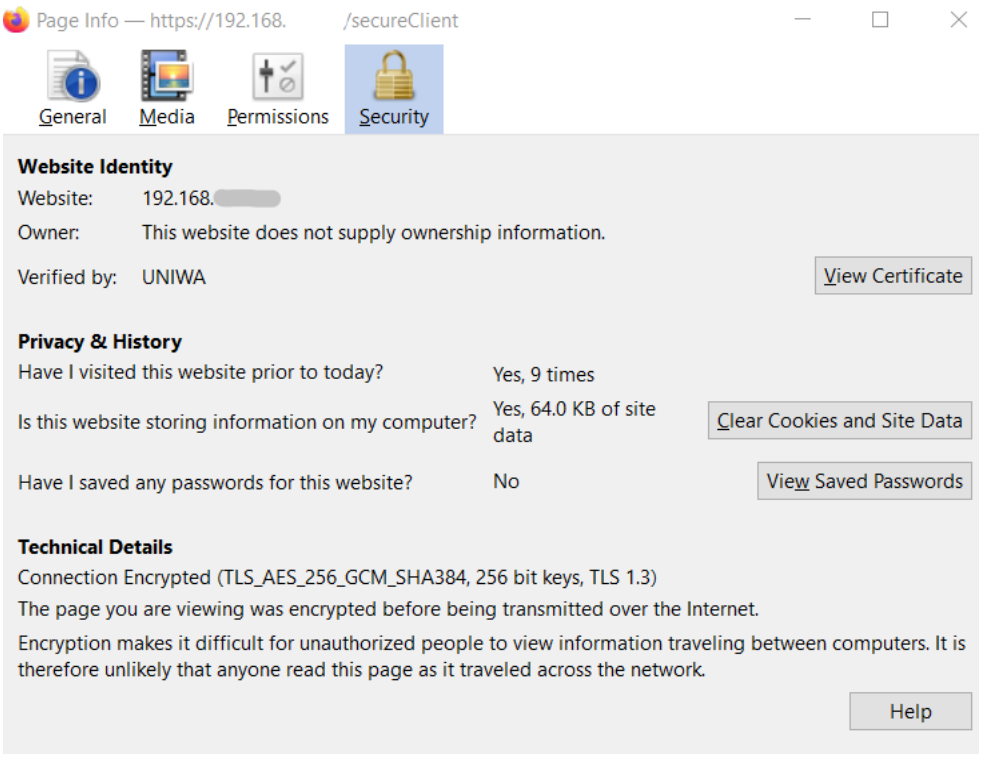
Εικόνα 5.4.3.3. Αποθηκευμένο αρχείο βιντεοσκόπησης

Έπειτα από το τέλος βιντεοσκόπησης, ο χρήστης μπορεί να βρει το αρχείο στον φάκελο που έχει ορίσει για αποθήκευση. Στην περίπτωση μας το αρχείο βρίσκεται στον φάκελο “Videos”. Η ονομασία του γίνεται αυτόματα από το πρόγραμμα και το όνομα που λαβαίνει αναπαρίσταται με τον παρακάτω τρόπο “Έτος-Μήνας-Ημέρα-Ώρα-Λεπτά-Δευτερόλεπτα”. Ο τύπος αρχείου που επιλέχθηκε να αποθηκευτεί το βίντεο είναι “.mkv”. Ο λόγος που επιλέχθηκε αυτός είναι η ασφάλεια που παρέχεται στην περίπτωση βλάβης συστήματος. Σε περίπτωση που το OBS Studio σταματήσει να λειτουργεί ενώ γίνεται βιντεοσκόπηση, η προκύψει κάποιο πρόβλημα στον υπολογιστή, το αποθηκευμένο βίντεο σε μορφή “.mkv” θα συνεχίσει να υπάρχει μέχρι την στιγμή που προκύπτει το πρόβλημα. Στην περίπτωση που χρησιμοποιούνταν μορφή “.mp4”, υπάρχει μεγάλη πιθανότητα το αρχείο να ορίζεται ως “corrupted” και να μην μπορεί να γίνει η ανάκτηση αυτού.

5.5 ΠΙΣΤΟΠΟΙΗΤΙΚΑ ΣΕ BROWSERS

5.5.1 ΠΡΟΒΟΛΗ ΠΙΣΤΟΠΟΙΗΤΙΚΟΥ

Μέσω της επιλογής Ctrl+I από το πληκτρολόγιο, στον Mozila Firefox, υπάρχει η δυνατότητα ο χρήστης να δει τις πληροφορίες της σελίδας που έχει επισκεφτεί. Στην περίπτωση μας θέλουμε να δούμε την ασφάλεια που παρέχεται σε αυτήν. Με την επιλογή “Security”, μπορούμε να δούμε τα παρακάτω.

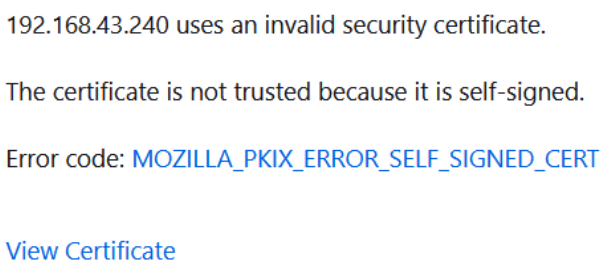


Εικόνα 5.5.1.1. Πληροφορίες σύνδεσης

Στην παραπάνω εικόνα παρατηρούμε της πληροφορίες ασφαλείας. Όπως βλέπουμε στο κομμάτι “Website Identity”, εμφανίζεται η διεύθυνση της ιστοσελίδας, που είναι και η IP διεύθυνση που χρησιμοποιήθηκε. Στην συνέχεια αναφέρεται το “Verified by: UNIWA” που αυτή η πληροφορία εισήχθη στο κεφάλαιο 4.1 κατά την δημιουργία του αυτοϋπογεγραμμένου πιστοποιητικού.

Στο μέρος “Technical Details”, παρατηρούμε την δήλωση ότι η σύνδεση είναι κρυπτογραφημένη. Πέραν αυτού όμως, ενημερωνόμαστε ότι όλη η κρυπτογράφηση γίνεται πριν μεταδοθεί στο διαδίκτυο. Στην υλοποίηση μας, δεν γίνεται ποτέ σύνδεση στο διαδίκτυο, καθώς αυτή θα έκανε πιο ευάλωτη την χρήση της.

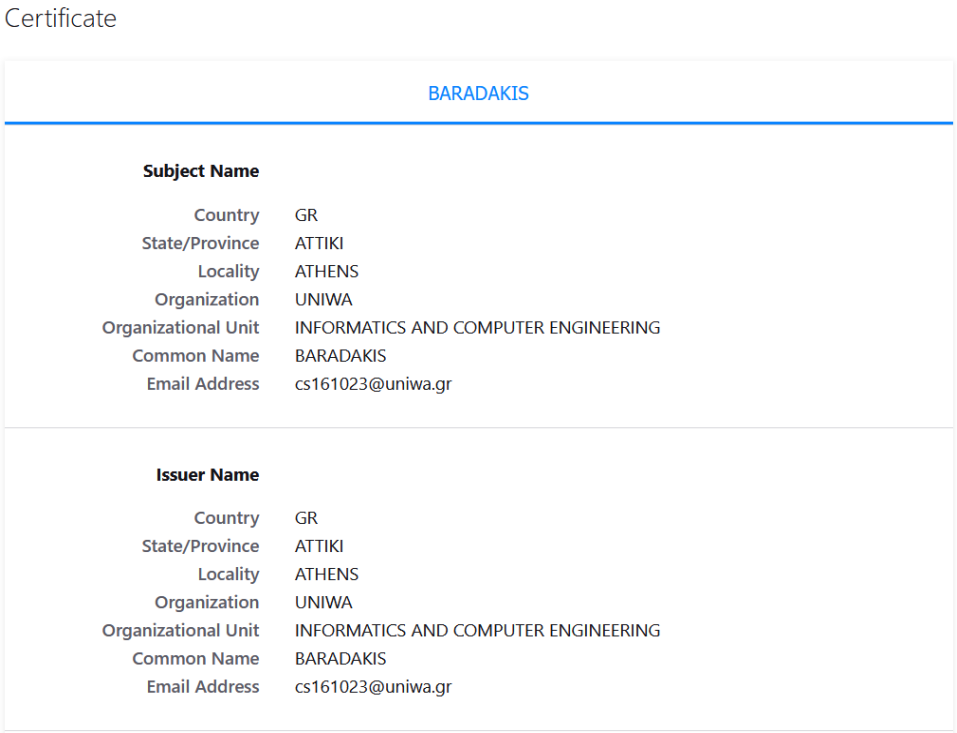
5.5.2 EXCEPTIONS ΣΕ BROWSERS



Εικόνα 5.5.2.1. Exceptions

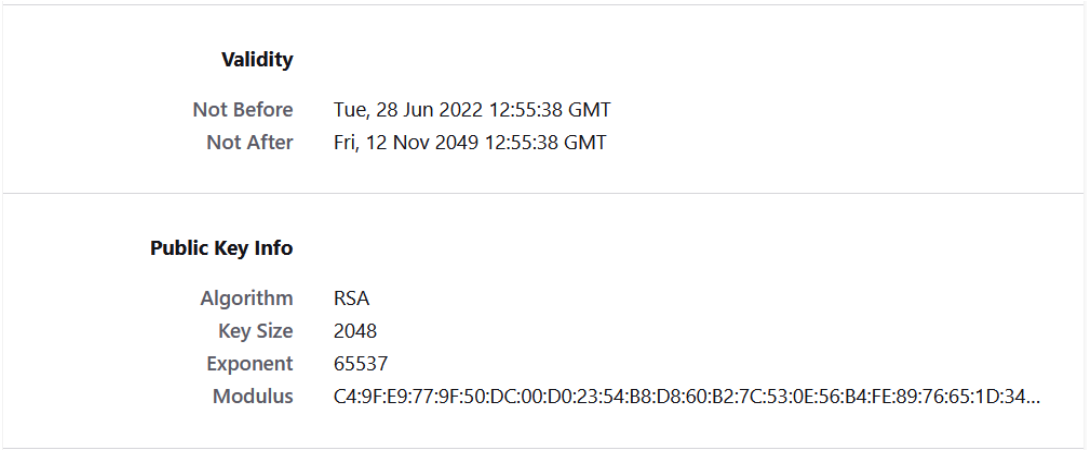
Λόγω των αυξημένων αναγκών ασφαλείας, οι περισσότεροι Browsers, χαρακτηρίζουν μη ασφαλή, μία σύνδεση που σε αυτήν, το πιστοποιητικό είναι αυτοϋπογεγραμμένο. Αυτή η δήλωση όμως, δεν σημαίνει ότι δεν γίνεται κρυπτογράφηση, ή ότι το αυτοϋπογεγραμμένο πιστοποιητικό είναι άκυρο. Όπως είδαμε στην εικόνα 5.4.1.1 , υπάρχει κρυπτογράφηση TLS σε όλη την διάρκεια λειτουργίας του Server. Σε περίπτωση που επιθυμεί ο χρήστης να εξαλειφθεί το μήνυμα αυτό, είναι αναγκαίο να γίνει αγορά πιστοποιητικού από μία αρχή έκδοσης πιστοποιητικών (Certificate Authority), όπως εξηγήθηκε στο κεφάλαιο 2.5.

Με το κλικ στην επιλογή View Certificate, μπορεί να προβληθεί αναλυτικά το αυτοϋπογεγραμμένο πιστοποιητικό που δημιουργήθηκε στο κεφάλαιο 4.1.



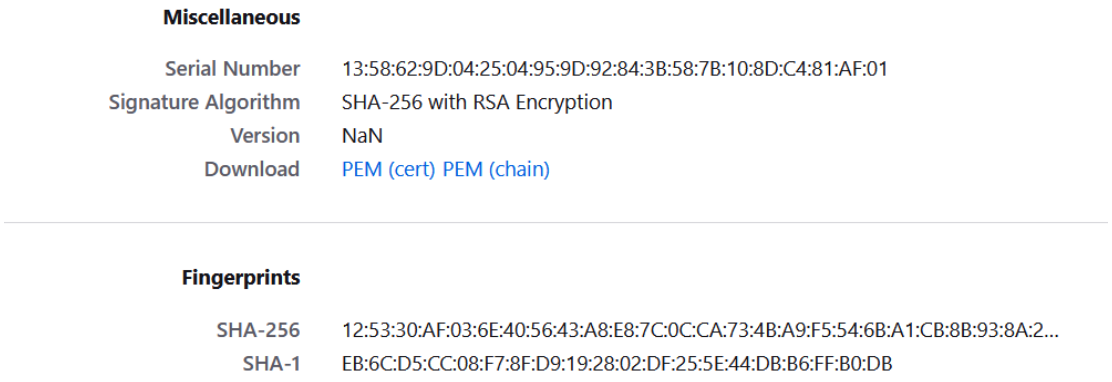
Εικόνα 5.5.2.2. Πρώτο μέρος στοιχείων πιστοποιητικού

Παρατηρούμε ότι φαίνονται όλα τα στοιχεία που είχαν εισαχθεί κατά την δημιουργία του πιστοποιητικού, όπως ακριβώς υπήρχαν σε αυτό.



Εικόνα 5.5.2.3. Δεύτερο μέρος στοιχείων πιστοποιητικού

Στην συνέχεια βλέπουμε την εγκυρότητα του, με αυτή να ξεκινά από 28/6/2022 και να τερματίζει στις 12/11/2049. Έπειτα παρέχονται πληροφορίες για το δημόσιο κλειδί που δημιουργήθηκε.



Εικόνα 5.5.2.4. Τρίτο μέρος στοιχείων πιστοποιητικού

Τέλος, παρατηρούμε τον σειριακό αριθμό του πιστοποιητικού καθώς και τον αλγόριθμο υπογραφής του. Με την επιλογή PEM(cert) ή PEM(chain) υπάρχει η δυνατότητα για λήψη του αυτοϋπογεγραμμένου πιστοποιητικού που έχει δημιουργηθεί.

5.6 ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΕ WIRESHARK

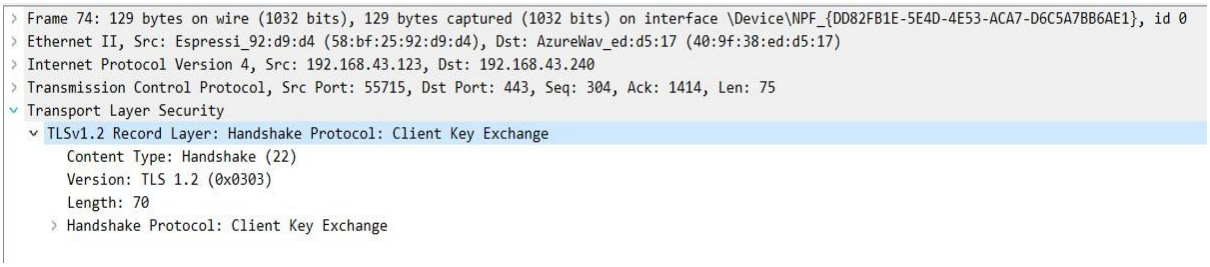
5.6.1 ΕΙΣΑΓΩΓΗ

Το Wireshark είναι ελεύθερο και ανοιχτού κώδικα λογισμικό ανάλυσης πρωτοκόλλων δικτύου υπολογιστών. Χρησιμοποιείται για την ανάλυση και παρακολούθηση του δικτύου, εντοπισμό και αντιμετώπηση προβλημάτων στα δίκτυα και για εκπαίδευση. Είναι γραμμένο στις γλώσσες Lua, C, C++.

Με την χρήση του, θα παρατηρήσουμε τα κρυπτογραφημένα πακέτα και δεδομένα που στέλνονται στην παραπάνω υλοποίηση.

Εικόνα 5.6.1.1 Λογότυπο WireShark

5.6.2 ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΠΑΚΕΤΩΝ

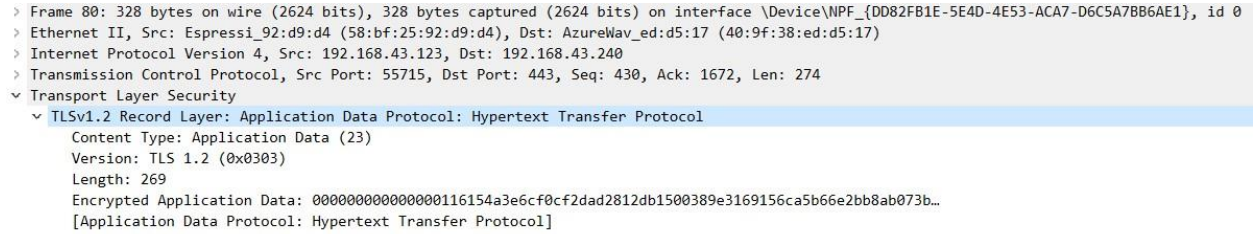


Εικόνα 5.6.2.1. Έναρξη handshake πρωτοκόλλου

Αρχικά, μέσω του Wireshark και του φίλτρου “ssl” που εφαρμόστηκε, παρατηρείται η έναρξη του handshake πρωτοκόλλου που ξεκινά κατά την αρχή λειτουργίας της υλοποίησης. Γίνεται η ανταλλαγή κλειδιού του client.

Εικόνα 5.6.2.2. Handshake πρωτόκολλο

Το handshake πρωτόκολλο συνεχίζεται με την αποστολή ενός κρυπτογραφημένου handshake μηνύματος. Παρατηρείται επίσης ότι και στις δύο εικόνες με τα παραπάνω πακέτα που παρουσιάστηκαν, το πρωτόκολλο TLS έχει οριστεί.



Εικόνα 5.6.2.3 Κρυπτογραφημένα δεδομένα

Παρατηρούμε ότι μετά την επιτυχή σύνδεση, τα δεδομένα είναι κρυπτογραφημένα. Υπό την κανονική τους μορφή, τα δεδομένα είναι τύπου Binary, όμως με την χρήση κρυπτογράφησης, αυτή η μορφή παύει πλέον να υπάρχει και αντικαθίσταται από τα binary δεδομένα σε κρυπτογραφημένη μορφή.

6. ΜΕΤΡΗΣΕΙΣ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

6.1 ΜΕΤΡΗΣΗ ΕΜΒΕΛΕΙΑΣ

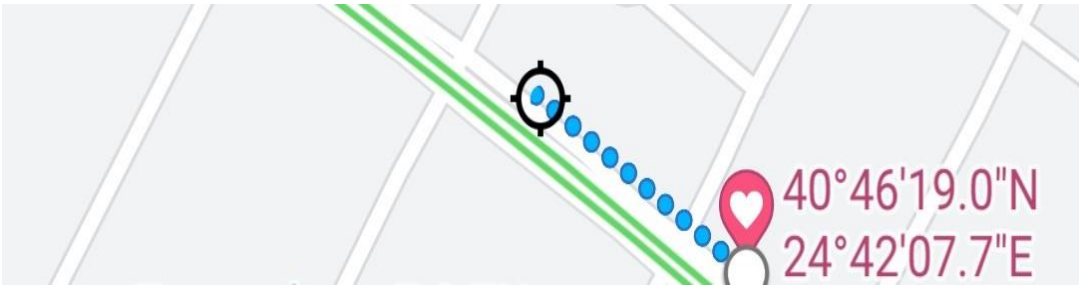
6.1.1 ΕΙΣΑΓΩΓΗ

Για την μέτρηση εμβέλειας, ο χώρος που επιλέχθηκε είναι ένας ανοιχτός, ευρύς και εξωτερικός χώρος. Αυτό μας δίνει την δυνατότητα να έχουμε τον μικροελεγκτή σε ευθεία απόσταση με το laptop και το φορητό σημείο πρόσβασης. Επίσης δεν υπάρχουν αντικείμενα που να μπορούν να παρεμποδίσουν την μετάδοση του σήματος Wi-Fi, αφού ο μικροελεγκτής και τα υπόλοιπα περιφερειακά έχουν έναν “line of sight” τύπο μετάδοσης δεδομένων. Για αυτό τον λόγο, η μετάδοση αυτών γίνεται με πιο ακριβή τρόπο.

6.1.2 ΜΕΘΟΔΟΣ ΜΕΤΡΗΣΗΣ

Για την μέτρηση χρησιμοποιήθηκε το εργαλείο μέτρησης αποστάσεων Google Maps. Επίσης απαιτήθηκε χρήση ενός τηλεφώνου τύπου Smartphone και ένα επιπλέον άτομο. Το laptop τρέχει ο server παραμένει σε σταθερό σημείο και οι συντεταγμένες του σημείο αυτού, αποθηκεύονται στο smartphone του ατόμου που θα πάρει την μέτρηση.

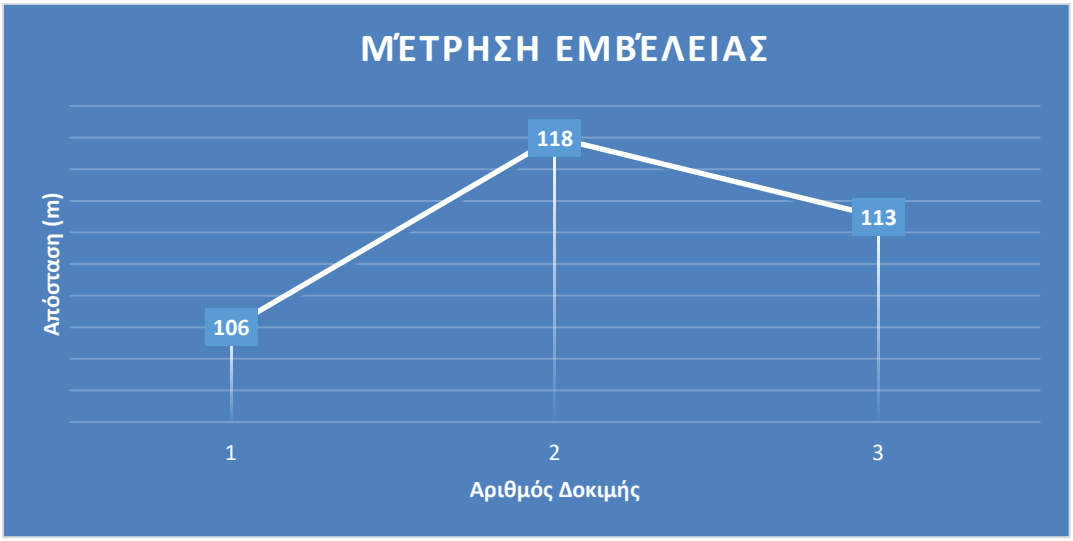
Στην συνέχεια ένα άτομο κρατά τον μικροελεγκτή και κατευθύνεται προς μία πορεία ενώ το δεύτερο άτομο παρατηρεί το video που στέλνεται από τον μικροελεγκτή στον server. Την στιγμή που υπάρχει κάποια αλλοίωση-καθυστέρηση στο video, ο παρατηρητής του server ενημερώνει το άτομο που προχωρά ώστε να σταματήσει. Τέλος, μέσω του smartphone, το άτομο που κρατά τον μικροελεγκτή, μετρά την απόσταση του, από τις συντεταγμένες που έχει ο server.



Εικόνα 6.1.2.1. Συντεταγμένες Αφετηρίας Μέτρησης Απόστασης

Παραπάνω δίνονται οι συντεταγμένες που του σημείου που βρισκόταν σε λειτουργία ο server κατά την διάρκεια των δοκιμών. Η ευθεία μπλε γραμμή αναπαριστά την απόσταση που διανύει το δεύτερο άτομο για την μέτρηση της απόστασης μεταξύ αυτού και του server.

6.1.3 ΑΠΟΤΕΛΕΣΜΑΤΑ



Πίνακας 6.1.3.1. Αποτελέσματα μέτρησης εμβέλειας

Έπειτα από τρείς μετρήσεις, συμπεραίνουμε ότι ο μέσος όρος τριών μετρήσεων απόστασης είναι τα 112.3 μέτρα. Σε αυτή την απόσταση η λειτουργία της υλοποίησης γίνεται ομαλά, χωρίς καθυστερήσεις και αλλοιώσεις στο βίντεο.

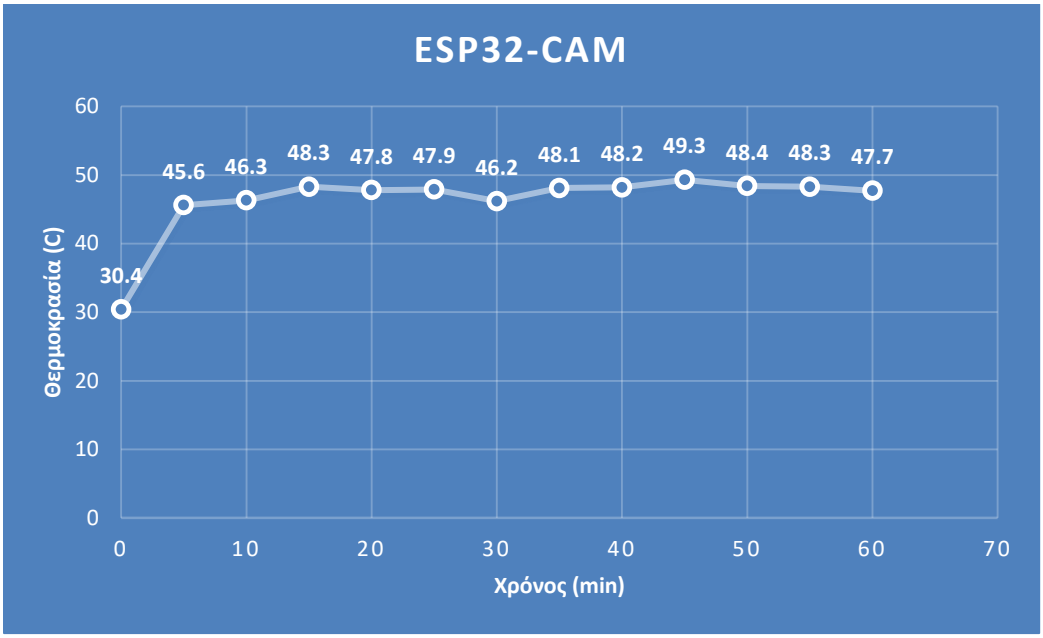
6.2 ΜΕΤΡΗΣΗ ΘΕΡΜΟΚΡΑΣΙΑΣ

6.2.1 ΕΙΣΑΓΩΓΗ

Για την μέτρηση θερμοκρασίας του μικροελεγκτή χρησιμοποιήθηκε θερμόμετρο υπέρυθρων ακτινών. Το μοντέλο που χρησιμοποιήθηκε είναι το HD-E-01 και οι μετρήσεις του γίνονται ανέπαφα. Σύμφωνα με τον κατασκευαστή, το θερμόμετρο αυτό έχει ακρίβεια ±0.3C ενώ η απόσταση μέτρησης πρέπει να γίνεται σε απόσταση 5 έως 15 εκατοστά από το αντικείμενο.

Ο συνολικός χρόνος των μετρήσεων ήταν μία ώρα λειτουργίας ενώ η δειγματοληψία γινόταν κάθε 5 λεπτά. Η δειγματοληψία έγινε για τον μικροελεκγτή και για το power bank. Η θερμοκρασία περιβάλλοντος που πραγματοποιήθηκαν οι μετρήσεις είναι κατά μέσο όρο 30 βαθμοί κελσίου.

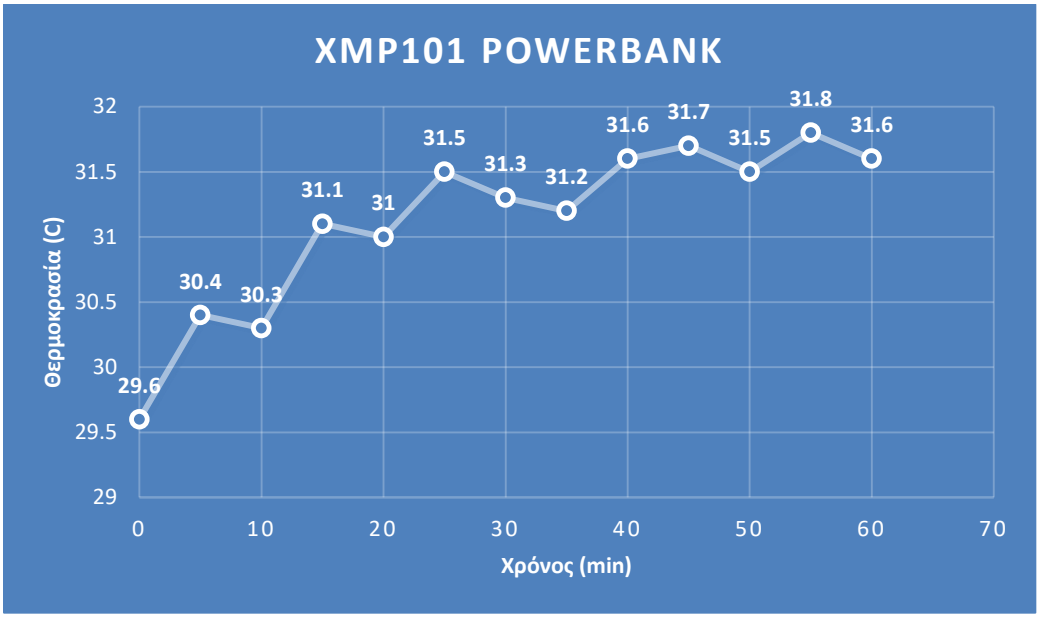
6.2.2 ΜΕΤΡΗΣΗ ΘΕΡΜΟΚΡΑΣΙΑΣ ΜΙΚΡΟΕΛΕΓΚΤΗ



Πίνακας 6.2.2.1. Αποτελέσματα μέτρησης θερμοκρασίας μικροελεγκτή

Στον παραπάνω πίνακα απεικονίζονται οι μετρήσεις θερμοκρασίας για το ESP32-CAM. Όπως βλέπουμε η αρχική του θερμοκρασία είναι ίδια με την θερμοκρασία περιβάλλοντος. Από τα 5 πρώτα λεπτά χρήσης, η θερμοκρασία έχει αυξηθεί κατά 15.5 βαθμούς. Παρατηρείται ότι η ελάχιστη θερμοκρασία λειτουργίας από τα 5 λεπτά και μετά είναι οι 45.6 βαθμοί ενώ η μέγιστη οι 49.3, ενώ διαπιστώνουμε ότι από τα 5 λεπτά λειτουργίας μέχρι το τέλος των μετρήσεων, η θερμοκρασία παραμένει σταθερή, με μέση τιμή τους 47.6 βαθμούς κελσίου. Σε γενικές γραμμές η θερμοκρασία παραμένει σε σταθερή τιμή κατά την λειτουργία της συσκευής.

6.2.3 ΜΕΤΡΗΣΗ ΘΕΡΜΟΚΡΑΣΙΑΣ ΜΠΑΤΑΡΙΑΣ



Πίνακας 6.2.3.1. Αποτελέσματα μέτρησης θερμοκρασίας XMP101 power bank

Στον παραπάνω πίνακα απεικονίζονται οι μετρήσεις θερμοκρασίας που έγιναν για το power bank που χρησιμοποιείται. Όπως και στον μικροελεγκτή, έτσι και σε αυτό, η αρχική του θερμοκρασία είναι όμοια με τη θερμοκρασία περιβάλλοντος. Μετά τα πρώτα 5 λεπτά λειτουργίας, η θερμοκρασία έχει αυξηθεί μόνο κατά 0.8 βαθμούς κελσίου. Ενώ παρατηρείται ότι η μεταβολή της θερμοκρασίας σε όλη την διάρκεια των δοκιμών είναι μηδαμινή. Η ελάχιστη τιμή από τα 5 λεπτά λειτουργίας και μετά είναι οι 30.5 βαθμοί ενώ η μέγιστη τιμή οι 31.8. Ο μέσος όρος θερμοκρασίας κατά τις δοκιμές από τα 5 λεπτά και μετά είναι 31.3 βαθμοί κελσίου.

6.3 ΜΕΤΡΗΣΗ ΚΑΤΑΝΑΛΩΣΗΣ

6.3.1 ΕΙΣΑΓΩΓΗ

Η χρήση power bank στον μικροελεκγτή του συστήματος, τον καθιστά μεν αυτόνομο, αλλά αυξάνει το βάρος του μέρους που θα τοποθετηθεί στο drone. Ως εκ τούτου, η χαμηλή κατανάλωση ενέργειας για τη διατήρηση χαμηλού βάρους, καθίσταται απαραίτητη. Επιπλέον είναι σημαντική η συνεχής

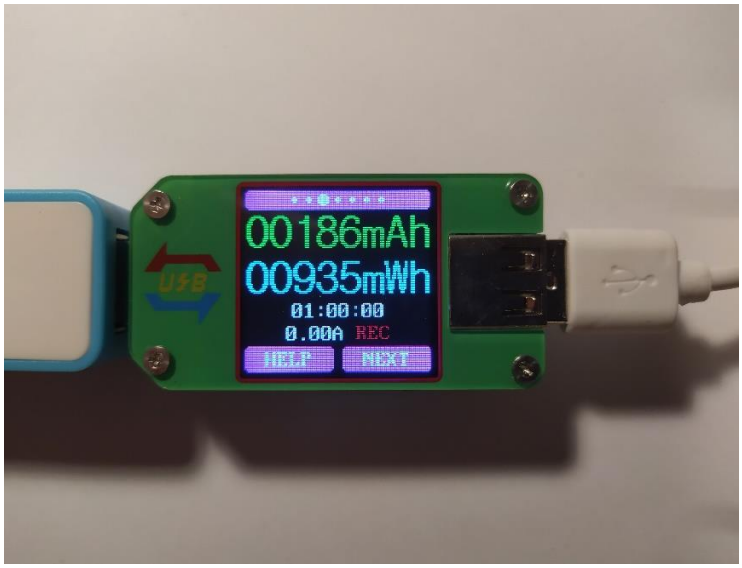
παρακολούθηση της στάθμης των μπαταριών που θα χρησιμοποιηθούν, ώστε να υπάρχει έγκαιρη ενημέρωση για την επαναφόρτιση/αντικατάσταση τους.

6.3.2 ΜΕΤΡΗΣΗ ΚΑΤΑΝΑΛΩΣΗΣ ΜΕ ΚΡΥΠΤΟΓΡΑΦΗΣΗ

Εικόνα 6.3.2.1. Μέτρηση κατανάλωσης

Για την μέτρηση κατανάλωσης, χρησιμοποιήθηκε η συσκευή ελέγχου ορθής λειτουργίας RuiDeng UM24C. Η συσκευή αυτή έχει την δυνατότητα μέτρησης της κατανάλωσης στον μικροελεγκτή. Αποτελείται από ένα αρσενικό (male) και ένα θηλυκό (female) USB-A. Το Αρσενικό συνδέεται στην συσκευή τροφοδοσίας, ενώ το θηλυκό στην συσκευή κατανάλωσης ρεύματος. Το αρσενικό μέρος θα συνδεθεί στο power bank ενώ το θηλυκό στον μικροελεγκτή, μέσω του Micro USB. Η εξίσωση για την μέτρηση διάρκειας λειτουργίας είναι η παρακάτω:

$$Διάρκεια\ Λειτουργίας(Ώρες) = \frac{Χωρητικότητα\ Μπαταρίας(mAh) * Τάση\ Λειτουργίας(V)}{Κατανάλωση\ Ανά\ Ώρα(mWh)}$$



Εικόνα 6.3.2.2. Αποτέλεσμα μέτρησης κατανάλωσης (μία ώρα χρήσης)

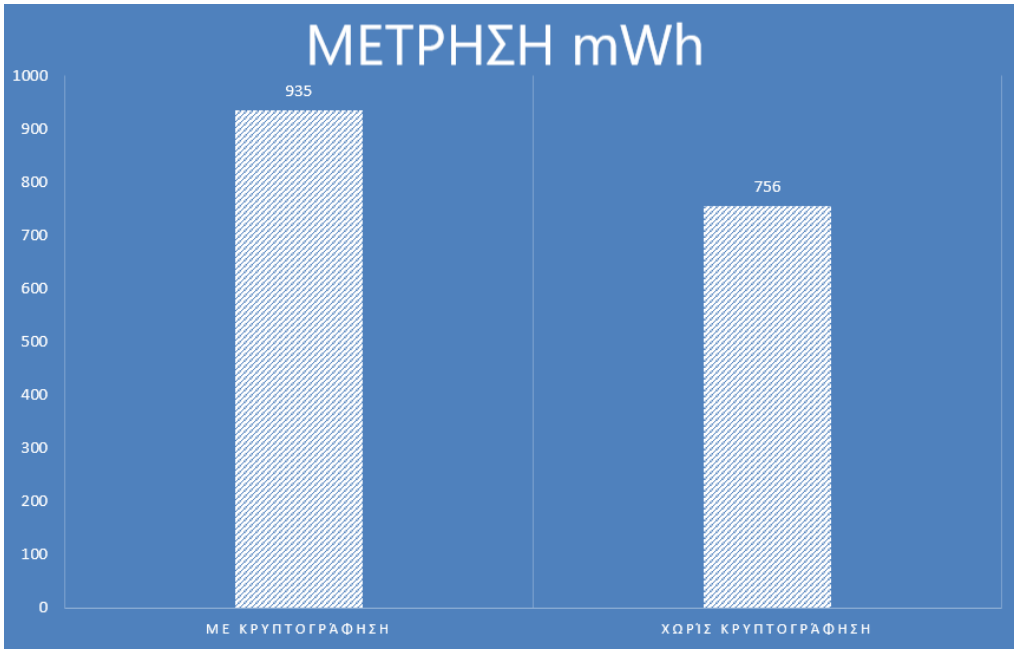
Αντικαθιστώντας τις τιμές που δίνονται από την συσκευή στον παραπάνω τύπο μέτρησης διάρκειας ζωής μπαταρίας, μπορούμε να βρούμε τον χρόνο στον οποίο ο μικροελεγκτής λειτουργεί συνεχώς. Δεδομένου ότι η μπαταρία που βρίσκεται στο power bank έχει τάση 3.7 Volts, η χωρητικότητα της είναι 2000mAh και μέσα σε μία ώρα καταναλώθηκαν 935mWh, ερχόμαστε στο συμπέρασμα ότι με αυτή την πηγή ρεύματος, εξασφαλίζονται σχεδόν 8 ώρες λειτουργίας (474 λεπτά).

6.3.3 ΣΥΓΚΡΙΣΗ ΚΑΤΑΝΑΛΩΣΗΣ ΜΕ ΜΗ ΚΡΥΠΤΟΓΡΑΦΗΜΕΝΗ ΕΚΔΟΣΗ

Για την σύγκριση αυτή έγιναν δοκιμές στην μη κρυπτογραφημένη έκδοση της υλοποίησης. Οι δοκιμές αυτές είχαν διάρκεια μίας ώρας και μετρήθηκε η συνολική ενέργεια που απαιτήθηκε για την λειτουργία της έκδοσης αυτής. Παρακάτω παρατηρούμε τα αποτελέσματα.

Εικόνα 6.3.3.1. Αποτέλεσμα μέτρησης κατανάλωσης χωρίς χρήση κρυπτογράφησης

Ως αποτέλεσμα έπειτα από μία ώρα λειτουργίας παρατηρούμε ότι η κατανάλωση στην μη κρυπτογραφημένη έκδοση απαιτεί 756mWh. Αντικαθιστώντας τις τιμές αυτές στον τύπο που παρουσιάστηκε, ερχόμαστε στο συμπέρασμα ότι η μη κρυπτογραφημένη έκδοση έχει χρόνο λειτουργίας σχεδόν 10 ώρες (587 λεπτά).



Σχήμα 6.3.3.1. Σύγκριση μετρήσεων κατανάλωσης

Με τα παραπάνω δεδομένα ερχόμαστε στο συμπέρασμα ότι η χρήση κρυπτογράφησης στην υλοποίηση, έχει αρνητική επίδραση στην κατανάλωση ρεύματος. Πιο συγκεκριμένα η κρυπτογραφημένη έκδοση απαιτεί 23.6% περισσότερη ενέργεια (179 mWh) σε σχέση με την μη κρυπτογραφημένη έκδοση. Επίσης ο χρόνος λειτουργίας της εφαρμογής με κρυπτογράφηση σε σχέση με την μη κρυπτογραφημένη έκδοση μειώνεται κατά 19.2% (113 λεπτά).

6.4 ΣΥΓΚΡΙΣΗ ΡΟΗΣ ΔΕΔΟΜΕΝΩΝ

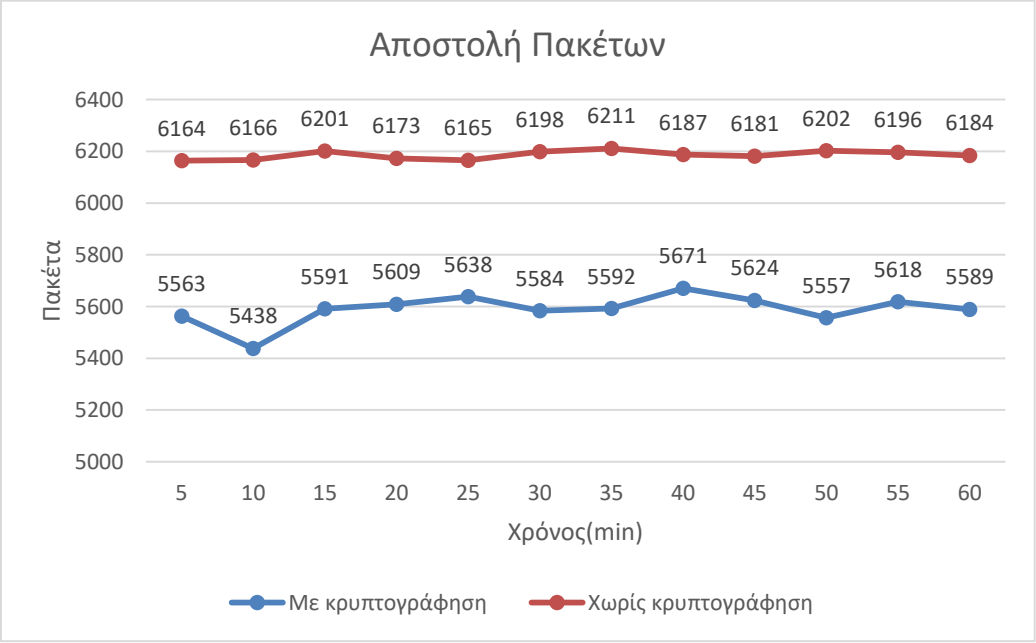
6.4.1 ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό θα γίνει σύγκριση ροής των κρυπτογραφημένων δεδομένων που αποστέλλονται από τον μικροελεγκτή στον server, σε σχέση με τα δεδομένα που αποστέλλονται χωρίς την χρήση κρυπτογράφησης. Για την επίτευξη αυτής της μέτρησης χρησιμοποιείται το serial monitor του περιβάλλοντος Arduino IDE, όπως παρουσιάστηκε στην ενότητα 5.2.4.

Μέσω του timestamp που δίνεται σε κάθε αποστολή δεδομένων, παρέχεται η δυνατότητα μέτρησης αποστολής δεδομένων. Οι δοκιμές που εκτελέστηκαν στην κρυπτογραφημένη, και μη κρυπτογραφημένη έκδοση, είχαν συνολική διάρκεια χρόνου την μία ώρα.

6.4.2 ΑΠΟΤΕΛΕΣΜΑΤΑ

Παρακάτω παρουσιάζονται και αναλύονται τα αποτελέσματα που προέκυψαν



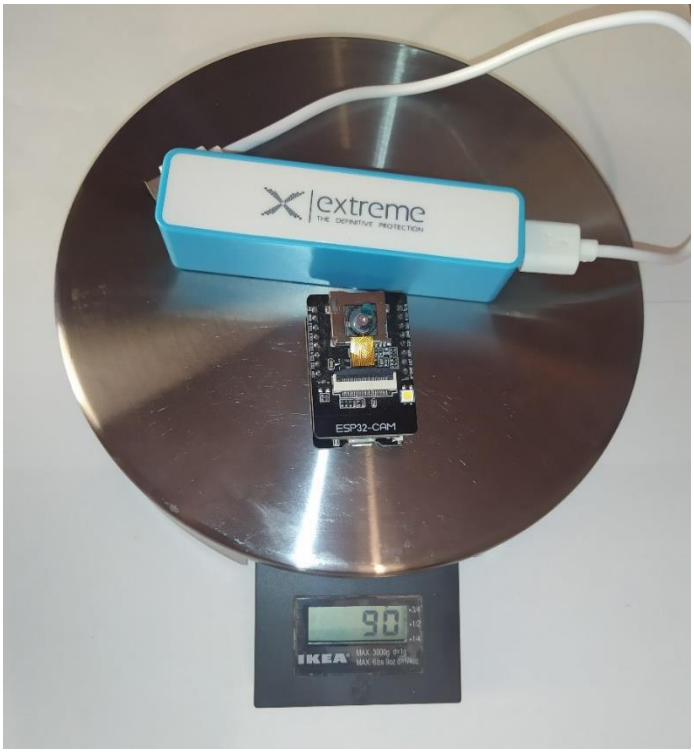
Σχήμα 6.4.2.1. Στοιχεία δεδομένων αποστολής πακέτων ανά ώρα

Στην κρυπτογραφημένη έκδοση, ο μέσος όρος αποστολής πακέτων ανά λεπτό ανέρχεται στα 1117.9. Ενώ τα συνολικά πακέτα που στάλθηκαν σε μία ώρα δοκιμών είναι 67074. Στην μη κρυπτογραφημένη έκδοση, ο μέσος όρος αποστολής πακέτων ανά λεπτό ανέρχεται στα 1237.1, ενώ το σύνολο των πακέτων που στάλθηκαν σε μία ώρα είναι 74228. Με τα παραπάνω δεδομένα ερχόμαστε στο συμπέρασμα ότι η κρυπτογράφηση παρουσιάζει μείωση του ρυθμού αποστολής δεδομένων κατά 9.64% σε σχέση με την μη κρυπτογραφημένη έκδοση.

Η μείωση αυτή είναι αναμενόμενη, δεδομένου ότι για την διαδικασία κρυπτογράφησης απαιτείται επιπλέον χρόνος. Παρ όλα αυτά, η υλοποίηση δεν παρουσιάζει προβλήματα καθυστέρησης βίντεο όταν βρίσκεται εντός του ορίου εμβέλειας που παρουσιάστηκε στο κεφάλαιο 6.1.

6.5 ΜΕΤΡΗΣΗ ΒΑΡΟΥΣ

Όπως είναι λογικό, το βάρος σε οποιοδήποτε εξάρτημα προστεθεί σε ένα drone παίζει πολύ σημαντικό ρόλο. Στην υλοποίηση, τα υλικά που απαιτείται να τοποθετηθούν σε drone είναι ο μικροελεγκτής, το power bank, και το καλώδιο σύνδεσης τους. Για τις μετρήσεις του συνολικού βάρους χρησιμοποιήθηκε η ζυγαριά ORDNING 20047.



Εικόνα 6.5.1. Μέτρηση βάρους

Όπως παρατηρούμε στην μέτρηση, το συνολικό βάρος που απαιτείται να προσαρτηθεί πάνω στο Drone είναι τα 90 γραμμάρια. Για ένα drone μέσης χρήσης και τιμής, το βάρος αυτό είναι πολύ προσιτό, καθώς επιτρέπει στο drone να απογειωθεί αλλά και να εκτελέσει τις απαιτούμενες ενέργειες που έχει σκοπό να διεκπεραιώσει.

7. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Η χρήση κάμερας αποτελεί ένα εξαιρετικά χρήσιμο εργαλείο σε πληθώρα τομέων. Είναι ένα μέσο το οποίο μπορεί είτε να χρησιμοποιηθεί αποκλειστικά για την καθοδήγηση ενός drone, ενώ δύναται να χρησιμοποιηθεί για πληθώρα άλλων έργων, αναγκαίων για την σύγχρονη σημερινή κοινωνία.

Μία μελλοντική κατεύθυνση που θα μπορούσε να υλοποιηθεί για βελτίωση του συστήματος είναι η εφαρμογή κώδικα πάνω στο video της κάμερας με απώτερο σκοπό την αναγνώριση προτύπων μέσω μηχανικής μάθησης. Η κάμερα θα μπορούσε να χρησιμοποιηθεί για εντοπισμό ατόμων που βρίσκονται σε κίνδυνο, εντοπισμό αυτοκίνητων, πλοίων και άλλου είδους κινούμενων αντικειμένων, επιχειρήσεις έρευνας και διάσωσης καθώς και εντοπισμό πυρκαγιών.

Λόγω της ευχρηστίας και μεγάλης αντοχής της κάμερας στις θερμοκρασίες (-40 έως +125 κελσίου), μπορεί να γίνει τροποποίηση της για χρήση σε διάφορα άλλα μέσα εκτός των drone, κάποια από αυτά θα ήταν η χρήση της σε πλοία, ερευνητικές επιχειρήσεις σε πολικά κλίματα η κλίματα με αυξημένη θερμοκρασία.

Μία επιπλέον μελλοντική επέκταση που μπορεί να γίνει είναι η αλλαγή κεραίας Wi-Fi στον μικροελεγκτή. Με ισχυρότερη κεραία σε σχέση με την εργοστασιακή, οι αποστάσεις λειτουργίας επεκτείνονται και αυξάνονται, η προσθήκη αυτή επιτρέπει στον χειριστή να λειτουργήσει το drone σε μεγαλύτερες αποστάσεις, και να έχει συνεχή ροή δεδομένων παρά τις δυσκολίες αυτές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. National Institute of Standards and Technology. An Introduction to Computer Security: The NIST Handbook. Special Publication 800-12. October 1995.

2. William Stallings, Cryptography and Network Security Principles and Practice, Fifth Edition, 2011 Pearson Education Inc., publishing as Prentice Hall, 1 Lake Street, Upper Saddle River, NJ 07458.

3. Johnson, Leighton (2016), "Security Component Fundamentals for Assessment", Security Controls Evaluation, Testing, and Assessment Handbook, Elsevier.

4. Douglas E. Comer. Internetworking with TCP/IP, volume 1. Pearson, Harlow, Essex, UK, sixth edition, 2013.

5. J. Postel. DoD standard Transmission Control Protocol. RFC 761, January 1980. Obsoleted by RFC 793.

6. Pouyan Sepehrdad, Serge Vaudenay, and Martin Vuagnoux. Discovery and exploitation of new biases in RC4. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, SAC 2010: 17th Annual International Workshop on Selected Areas in Cryptography, volume 6544 of Lecture Notes in Computer Science.

7. Lawrence, Scott; Khare, Rohit (May 2000). "Upgrading to TLS Within HTTP/1.1". Internet Engineering Task Force. Retrieved December 15, 2018.

8. V. Patriciu , "Design Aspects in a Public Key Infrastructure for Network Applications Security", August 2000.

9. Cooper, S Santesson, S.Farrell, S.Boeyen, R. ousley, T.Polk. “Internet . public Key Infrastructure certificate and certificate revocation list (CR) profile”. RFC5280, IETF, May 2008

10. The Apache Software Foundation, "About the Apache HTTP Server Project", http://www .apache.org/ ABOUT _AP ACHE.html, February 1999.

11. World Wide Web Consortium, "HTTP - HyperText Transfer Protocol", http://www. w3c.org/Protocols/Overview .html, v 1.186 2000/07/06.

12. "Encrypting the Web". Electronic Frontier Foundation. Archived from the original on 18 November 2019. Retrieved 19 November 2019.

13. Konigsburg, Eitan; Pant, Rajiv; Kvochko, Elena (13 November 2014). "Embracing HTTPS". The New York Times. Archived from the original on 8 January 2019. Retrieved 20 October 2018

14. KUROSE|ROSS Computer Networking A Top-Down Approach

15. HTTP/2 Protocol Overview. RFC 7540, Hypertext Transfer Protocol Version 2 (HTTP/2)

16. https://randomnerdtutorials.com/ [Accessed June 2022]

17. https://github.com/Links2004/arduinoWebSockets [Accessed March 2022]
18. https://nodejs.org/en/knowledge/HTTP/servers/how-to-create-a-HTTPS-server/ [Accessed June 2022]
19. https://www.passwordmonster.com/ [Accessed June 2022]
20. https://github.com/0015/ThatProject/tree/master/ESP32CAM_Projects [Accessed January 2022]
21. https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol [Accessed June 2022]
22. https://el.wikipedia.org/wiki/HTTPS [Accessed June 2022]
23. https://en.wikipedia.org/wiki/Transport_Layer_Security [Accessed June 2022]
24. https://obsproject.com/ [Accessed July 2022]
25.https://www.researchgate.net/publication/357402723_AES_Encrypted_Real-Time_Video_Stream_and_Image_Transmission_from_ESP32-CAM [Accessed June 2022]
26. https://www.wireshark.org/ [Accessed October 2022]