YAŞAR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MASTER THESIS

A CONCEPTUAL DESIGN FOR MANAGING INTERNET OF THINGS DEVICES IN EMERGENCY SITUATIONS

BURAK KAYMAZ

THESIS ADVISOR: ASSOC. PROF. AHMET TUNCAY ERCAN

COMPUTER ENGINEERING

PRESENTATION DATE: 08.08.2019

BORNOVA / İZMİR
AUGUST 2019

We certify that, as the jury, we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Jury Members:	**Signature:**
Assoc. Prof. Ahmet Tuncay ERCAN Yasar University	[illegible]
Assoc. Prof. Mehmet Hilal ÖZCANHAN Dokuz Eylül University	[illegible]
Prof. Dr. Mehmet ÜNLÜTÜRK Yasar University	[illegible]

[illegible]

Prof. Dr. Cüneyt GÜZELİŞ
Director of the Graduate School

ABSTRACT

A CONCEPTUAL DESIGN FOR MANAGING INTERNET OF THINGS DEVICES IN EMERGENCY SITUATIONS

Kaymaz, Burak
Msc, Computer Engineering
Advisor: Assoc. Prof. Ahmet Tuncay ERCAN
August 2019

Internet of Things (IoT) concept is used within low cost devices in various forms as a result of technological developments. IoT enabled devices which are commonly used in workspace we need, such as hospitals, buildings, logistics, have been developed into systems, capable of data collection and transmission. In terms of quantity, around 8 billion of sensors are connected to IoT systems as of 2017. IoT devices are integrated into infrastructure systems in large residential areas, making human life easier and becoming a need that is used at every moment of daily life. In addition to the Internet of Things devices operating in ordinary situations, devices that operate in emergencies and provide situational awareness are needed. Climate disasters, fire, flood, earthquake, tsunami, war or terrorism-related nuclear, biological or chemical attacks or conventional attacks, damage infrastructure services either directly or indirectly causing interruption of service which is needed by everyone. This thesis proposes a conceptual design with a functioning prototype which is capable of providing an auxiliary Internet connection for Internet of Things devices in the area of disaster as well as sensing IoT devices with Wi-Fi, Bluetooth/BLE and sub-GHz communication technologies.

Key Words: Internet of Things, Emergency response, network management.

ÖZ

NESNELERİN İNTERNETİNDE ACİL DURUM YÖNETİMİ SAĞLAYAN MOBİL CİHAZ İÇİN KONSEPT TASARIM

Kaymaz, Burak
Yüksek Lisans Tezi, Bilgisayar Mühendisliği
Danışman: Doç.Dr. Ahmet Tuncay ERCAN
Ağustos 2019

Teknolojinin ilerlemesi ve ucuzlaması ile beraber Nesnelerin İnterneti (IoT) gündelik hayatta farklı tiplerde, küçük ve işlem kapasitesi sınırlı sayılabilecek cihazlarda kullanılmakta. Söz konusu cihazlar ihtiyaç duyduğumuz alanlarda - hastaneler, fabrikalar, binalar, lojistikte - veri toplayabilen ve bu veriyi aktarabilen sistemlere evrildi. Sayı bazında incelenecek olursa, 2017 itibariyle yaklaşık olarak 8 milyar sensör Nesnelerin İnterneti'ne bağlandı. IoT cihazları, büyük ölçekteki yerleşim bölgelerindeki altyapı sistemlerine dahil olarak, insan hayatını kolaylaştırıp gündelik hayatın her anında kullanılan bir ihtiyaç haline geliyor. Olağan durumlarda faaliyet gösteren Nesnelerin İnterneti cihazlarına ek olarak, acil durumlarda da faaliyet gösteren ve durumsal farkındalık sağlayan cihazlar ihtiyaç dahilindedir. İklimsel felaketler, yangın, sel, deprem, tsunami, savaş veya terörizm kaynaklı nükleer, biyolojik, kimyasal veya konvansiyonel saldırılar, altyapı sistemlerine doğrudan veya dolaylı yoldan zarar vererek, herkesin ihtiyaç duyduğu hizmetlerde kesintiye yol açar. Bu yüksek lisans tezinde, herhangi bir acil durumda afet bölgesinde bulunan Wi-fi, Bluetooth/BLE teknolojilerine ek olarak 1 GHz altında radyo frekansıyla haberleşen IoT cihazlarını algılayarak tanımlayan, aynı zamanda altyapının hasar görmesiyle internet erişimi kısıtlanmış olan IoT cihazlarına alternatif internet erişim kanalı sunan mobil cihazın konsept dizaynı ve fonksiyonlarını gösteren bir prototip sunulacaktır.

Anahtar Kelimeler: Nesnelerin Interneti, Afet ve acil durum yönetimi, Ag yonetimi.

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Assoc. Prof. Ahmet Tuncay ERCAN for his guidance and patience during this study.

Thinking outside of the box contributed many details during research and development process of this work, thanks to my mentor and beloved friend Graeme Mitchell Hanssen.

I would like to express my enduring love to my parents, who are always supportive, loving and caring to me in every possible way in my life.

Burak Kaymaz

İzmir, 2019

TEXT OF OATH

I declare and honestly confirm that my study, titled “A Conceptual Design for Managing Internet of Things Devices in Emergency Situations” and presented as a Master’s Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Burak Kaymaz

Signature

[illegible]

September 23, 2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS:

BASH	Bourne Again Shell
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
DARPA	Defense Advanced Research Projects Agency
GPIO	General Purpose Input/Output
GPRS	General Packet Radio Service
GPS	Global Positioning System
GUI	Graphical User Interface
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
ISM	Industrial, Scientific and Medical (radio spectrum)
IoT	Internet of Things
MAC	Media Access Control
NIC	Network Interface Card
OS	Operating System
OSI	Open Systems Interconnection
PPP	Point to Point Protocol
RAM	Random Access Memory
RPI	Raspberry Pi
SDN	Software Defined Network
SDR	Software Defined Radio
SSH	Secure Shell
SSID	Service Set Identifier

TCP	Transmission Control Protocol
USB	Universal Serial Bus
WAN	Wide Area Network
WAP	Wireless Access Point
WIFI	Wireless Fidelity
WLAN	Wireless Local Area Network

CHAPTER 1
INTRODUCTION

As Moore stated that while processing power increases and the cost of power decreases, technology becomes more abundant and cheap by the passage of time, in Electronics Magazine issued in 1965. This law can be observed today and leads us to a degree of freedom where any device could be connected to the internet. Internet of things can be considered as a new way of revolution just before the industry 4.0. As the automation replaces labor, internet of things makes us able to connect to the cloud by even our refrigerators. This shift in paradigm provides us with more flexibility in connectivity with our wireless computer networks.

Deployed sensors produce data such as thermal, visual, sonic based and even electromagnetic. Refineries, dams and factories for instance, utilizes these sensors to gain information about critical systems, also decision making systems are dependent on these sensors (Xu et al., 2014).

1.1. IoT

The exact definition of the IoT is a bit blurred just because there are many details which explain this concept. In its core, it can be clarified as connecting the virtual things such as information and data with physical things that we interact with every day. As one can comprehend, there are technologies that include IoT concept within their logic. Eventually these technologies, smart dust, wireless sensor networks, smart grids, smart cities are considered to be in the scope of IoT as well (Uckelmann et al., 2011). It could be better to understand that all these technologies have sensors and internet connection is delivered via secondary modules such as gateways or additional helper nodes depending on the severity of the system. So there are many clues which define the Internet of Things.

The name "IoT" first used by Kevin Aston in 1999, Kevin states that Internet of Things devices provide data which can be useful to maintain and control any system, consequently such systems could be more efficient and reliable. If we assume there is a critical system which provides a continuous public service, a prediction of a maintenance time derived from the data collected by IoT devices of the system. The Internet of Things integrates the digital world of information technology to our physical world with the help of sensors and actuators (Ashton, 2009).

Rob van Kranenburg states that IoT delivers a global connection of dynamic entities via the help of the standard computer network protocols, using sensory interfaces effectively thus, an improvement to data collection is achieved (Kranenburg, 2008).

IoT devices can be utilized in many ways, such as a cluster or a single device for home use. While the quantity increases, communication between these little sensors/actuators must be handled. IoT gateways come into the stage when data needs be sorted or supervised. IoT gateways are in charge of multiple roles, security administration between sensors or actuators, regulation of network communication can be achieved via an IoT gateway. Moreover IoT gateways transfer the data to a designated database for further analysis. In terms of processing power, IoT sensor networks contain a variety of devices. By deploying a simple sensor on the field is more effective in terms of processing power cost. IoT gateways conduct processing and hand over the data to an another IoT gateway or a cloud service. In addition to this, sensors in the IoT network can transfer their data, conducted by an IoT gateway to an end point.

1.2. Emergency Situations

As a conclusion, IoT devices will be around us in the near future, shaping our daily routines and helping our lives in any field. It is essential to note that effective utilization of these devices will be providing the data for both commercial and public use.

Every year, natural disasters occur at random, on different locations. It is a fact that early response in an emergency situation could save lives, while preventing mass panic and disorder. Natural disasters may be significantly colossal in magnitude, resulting aftermath may be perpetuated by the infrastructure which is rendered useless or reduced in utilization. Moreover, infrastructure has limitations even in daily use, and prone to overload while an ongoing aftermath, as a consequence the service quality and reliability may fail in the time of need.

There are many applications of management in emergency situations, differentiated by approach, emergency IoT protocols, auxiliary communication gateways, unmanned drone disaster discovery, emergency data collected by mobile phones, smart detection systems that provide data analysis on the computer network traffic and more.

1.3. Problem Definition

Recovering from a disaster involves diverse elements, for instance search and rescue teams, civilians, devices or systems are immediately incorporated for the purpose. Unfortunately, these elements are usually depend on infrastructure systems or practice. To elaborate, Software Defined Networks grant communication protocols to achieve network optimization, fixing computer network congestion issues, unfortunately SDN oriented solution operates on the infrastructure in order to perform disaster data collection. On the other hand, hardware based solutions such as a wearable hardware, maintaining live data on the vicinity may not be adaptable for frequent modification, supposing it is designed to serve a specific data, as a consequence required to be compliant with an another system. Even though diverse methods for disaster recovery exist, novel approaches would bring diversity for different circumstances. Among alternatives, our approach brings a novel system, designed to provide mobility, ease of use and situational awareness for search and rescue teams while provides a an alternative for communication in case of WAN access is out of service. In this study, a prototype device is proposed to serve during an emergency. In order to grant situational awareness, IoT devices in the vicinity are scanned and enumerated, also a mobile internet connection is established considering the need for an alternative method. In addition to device discovery, an Access Point is served for the IoT devices to maintain data transmission if provided with the settings by an authorized operator.

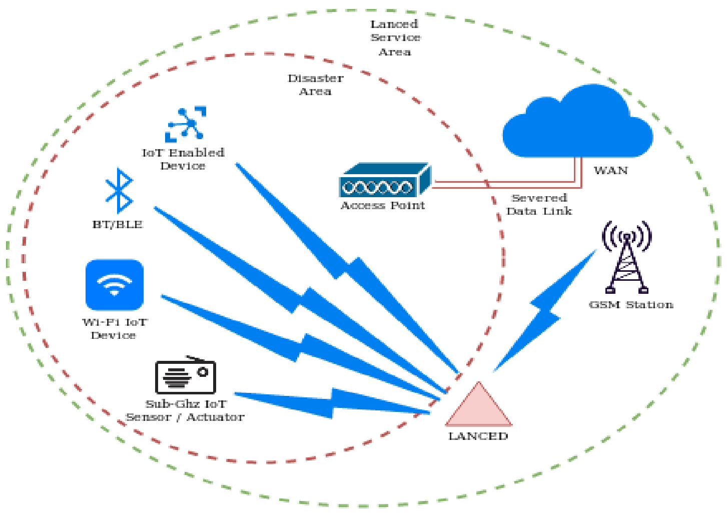


Figure 1.1 : Lanced data collection on emergency

This Master thesis aims to design and implement a device, providing certain functions for disaster recovery and management via help of IoT devices operational on the disaster area. Hardware and software based disaster recovery systems are available since IoT application grows larger by time. Our approach includes both hardware and software components since our goal is to achieve adaptability for any situation also providing mobility, interconnection and ease of use. Creating a scenario for our device is the first step for the design. Our scenario is based on a specific disaster area which is affected by any disaster type, since it is critical to remember that there is no infrastructure available.

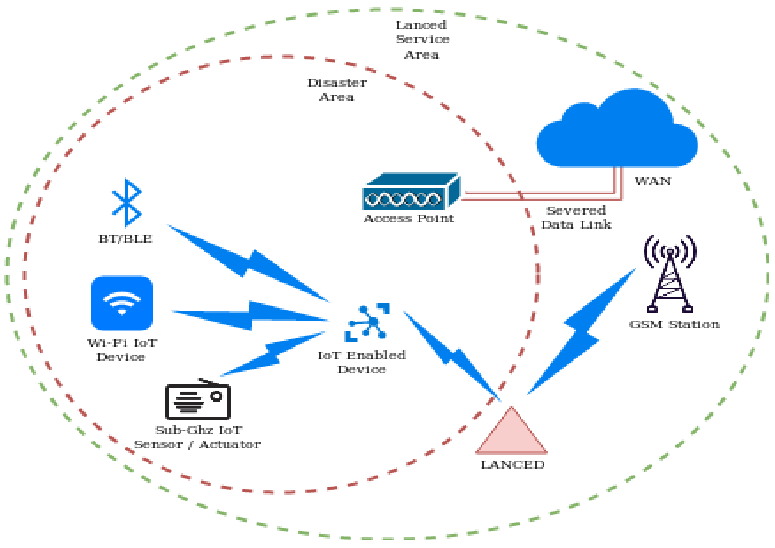


Figure 1.2 : Lanced as an auxiliary WAN connection

1.4. Roadmap

This thesis contain several chapters including this chapter. Organization of the thesis is presented here :

- **Chapter 2 - Background** provides the required content for comprehension of the thesis, history of the Internet before introduction of IoT, Open Systems Interconnection Model and OSI layers compared with IoT layers. Concerns of IoT devices and its uses. Latest research on implementation of IoT in emergency. IoT protocols, with communication and discovery protocols detailed.
- **Chapter 3 - Implementation** section provides details of LANCED prototype including hardware components and required software. System Design comprises the main functionalities, including the physical casing, main and secondary software modules, main and secondary hardware modules, restrictions, operational requirements. Circumstances for our device to operate, IoT devices in the scenario site and other environmental details which also critical for the system design.
- **Chapter 4 - Conclusion and Further Developments** this section provides an overall display of achieved goals and collected data, also future expansions of the device will be presented.

CHAPTER 2
BACKGROUND

Since this thesis aims to propose a device design in this chapter some background information will be presented to have better understanding with the Internet of Things, network protocols and IoT based network protocols, First of all, the term Internet of Things is described, the origin of the Internet of Things concept and applied fields will be presented.

IoT relies on conventional network architecture so briefly the network protocols will be commented here and the OSI model will be reviewed as well. The emergence of the internet as ARPANET, using the first two computers to switching packets are the important milestones of the development of the internet and consequently the Internet of Things (Leiner et al., 2009). IoT concept has new protocols designed to ease the connection and data transfer which is interest of this thesis also. In detail, these protocols will be presented. The IEEE and IETF formatted protocols are the main focus of this chapter, will be covered in detail. It is essential to give some details for the OSI model before detailing the IoT protocols.

2.1. A Brief History of Internet before IoT

Initial form of the internet was a structure aimed to help the researchers to communicate via using electrical lines. This idea indeed goes far back in history, Leonard Kleinrock was the first paper publisher about the modern internet dated in 1961. His approach on the topic was using the packets through line instead of conventional circuitry. Together with Thomas Merill, they created the first computer sending packets over the line and the communicating the two computers was their second goal. Thanks to their research, first tiny model of the internet was established (Kleinrock, 1961). After the first communication between two computers, ARPANET network was proposed by Roberts working for the DARPA, in his paper Multiple Computer Networks and Inter computer Communication in 1967. ARPANET, was the first building stone of the internet era, after the Kleinrock's packet switching model **(**Roberts, 1967).

Developing technologies like internet itself have a time period for to have some standards. This usually triggered by the public interest and use. During the process, public interest had some uses for the internet and consequently demanded products that have standard connection methods. The vendors that has products using the internet technology eventually needed standard for the products they have developed. This demands have lead the producers to have a uniform range of information, a consensus. First community of information trade formed back in 1988 by DARPA researchers (Leiner et al., 2009).

An important milestone of today's internet initiated by the International Standards Organization (ISO) in 1978 (Zimmermann, 1980).A committee on classifying the protocols and connection types decided that Open Systems Interconnection (OSI) abstract model is the official name for required classification for covering all the connections between different devices and systems. Open word in Open Systems Interconnection indicates that any device that has a connection regardless of it's type, is able to communicate through internet using the same standards. OSI model consisted seven layer of abstract fields with each level holds protocols designed with behavioral differences.

2.2. Open Systems Interconnection Model

Eventhough Internet of Things have a different type of approach to communication protocols, base model is the foundation of the newly arisen IoT protocols. Traditional OSI model has the ability to integrate these new additions for protocols, in next section of this thesis, there will be relation presented between OSI and the new protocols.

1. Physical Layer : Foundation of the OSI layers. This layer is the transfer medium for all data stream. A physical link is established between sender and receiver and the data is transferred on this link in signalized form. Connection form is delivered via physical link such as full duplex connection, half duplex or simplex. In addition this layer is able to sever or establish the connection. Network cards, hubs and even simple ethernet cables are elements of physical layer. IoT protocols in this layer contains physical access and network access

functionalities, IEEE 802.15.4 and WiFi, GSM, CDMA and LTE are in this layer.

2. Data-Link Layer : Creates data bundles called frames and ensures reliability while transferring data by providing source and destination MAC. The data passed from the network layer formatted into frames in such a way that in case there is a data loss or modification during the transfer, packet is dropped. This attribute is ensured by the FCS(Frame Check Sequence) part of the frame(IEEE 802.3). Compared with the IoT protocols, this layer is the same as the first layer.

3. Network Layer: This layer provides a wider range related to previous layers. Logical addressing is handled by this layer, destination and source address is encoded into the packet in a fashion which provides navigation between distant sender and receiver nodes. Logical addressing is determined by the Internet Protocol (IP) *RFC HERE* and encoded into frame with four octets of bits for both destination and source. IoT architecture has the same properties of Internet layer in TCP/IP reference model, IPv6 and 6LoWPAN technologies are included.

4. Transport Layer : This layer provides if the data is securely transferred over the physical medium, either reliable transfer TCP or unreliable transfer UDP is chosen. Establishes the reliable connection between the sender and receiver using TCP/IP protocol. Transmitted data may divided into segments, multiple parts also the synchronization of segments' transfer order are handled. IoT Transport Layer is consisted of TCP and UDP.

5. Session Layer : Between multiple nodes, session management and connection is handled by this layer. Starting and ending the session is the main purpose of this layer. This layer is included in the IoT as Application layer, which contains technologies such as MQTT, CoAP, HTTPS, XMPP, AMQP.

6. Presentation Layer : Data format is handled by this layer by translating it, for providing a common format between other nodes on the network. Formats JPG, GIF, MPEG, SSL, TLS are in scope of presentation layer. This layer is included in the IoT as Application layer, which contains technologies such as MQTT, CoAP, HTTPS, XMPP, AMQP.

7. Application Layer : The most outer layer of the OSI layers. Handles the connection based authentication checks, for instance a database connection is handled by this layer. A web browser is a nice example since it is sitting in the application layer. This layer is included in the IoT as Application layer, which contains technologies such as MQTT, CoAP, HTTPS, XMPP, AMQP.

OSI layers are working like top down approach when a request is made from the user, data is created and encapsulated from the above to the lower layers. Protocol Data Unit (PDU) is injected by order with each layer when data reaches the physical layer it is then transformed into a digital signal ready to be transferred through the physical layer transfer medium (Ravali, 2013).

2.3. IoT Implementation

IoT has many uses. Large management systems utilizes IoT, such as logistic firms, factories. Industrial Internet of Things (IIoT) is the term for classifying large actuator systems connected to the internet or control systems. To be more precise Cyber Manufacturing Systems can be utilized with IIoT. The system needs to be handled by centralized data nexus such as a gateway. Household IoT devices are also numerous and by the progression of time they became more common. Additionally, security and surveillance, transportation, healthcare, consumer and home, smart infrastructures are in scope of IoT.

Household or companies could have IoT systems deployed to control temperature, humidity, also smart infrastructure systems such as ventilation, interior lightning, security. Unlike WSNs (Wireless Sensor Networks) IoT systems are consisted of many sensors and some of these sensors are not able to provide internet connection therefore these sensors consume less energy and they are are relatively cheaper. When a tech company needs precise temperature in a system room, sensors are required to push data with a constant rate to a control system. While a household may not be affected by network traffic but a company may suffer congestion issues returns as cost for our company. A system consisted of many sensors and a central node called IoT gateway would ease the traffic and energy consumption. Moreover IoT gateway could handle the data before transmitted in WAN and then to cloud.

Main idea is while a gateway sits behind all these sensors, some aspects are not to be forgotten. While a sensor is continuously transmitting data, certain network traffic is produced.

2.4. Literature Work

Internet of Things based emergency management systems provide a wide scope of benefits. The data derived by devices in various systems then can be analyzed with algorithms, grants leverage for such situations that even a bit of information matters, saving more civilian lives. Since IoT can be applied on different systems, such as healthcare or smart cities, type of the collected data can have differences. These variations grant access on the topic with many angles.

In terms of computer network, congestion is an issue, in case of emergencies. Congestion can be a point of failure for systems which require real time data transmission. Overcoming congestion issue can be arranged via protocol based SDN method. An SDN oriented solution is proposed, ERGID: An efficient routing protocol for emergency response Internet of Things is aimed for real time data transmission control which yields reduced packet loss and energy consumption, the research provides two main techniques which are Delay Iterative Method (DIM) and Residual Energy Probability Choice (REPC). DIM calculates the delays between responses for the nodes then re-configures the neighbors providing a solution for ignored valid paths. REPC is a novel method presented by this research, exchanges data while overseeing the energy of the node compared with other nodes, following with the decision of next forwarding node (Qiu et al., 2016).

Minimal design of IoT devices deliver mobility and accessibility, also serves as wearable technologies. Wearable IoT devices are tailored for the purpose, such as firefighting or healthcare applications. The data collected through sensors placed on the service personnel, maintains a live data feed on the subject of matter. In this scope, researchers proposed a wearable technology named, Wearable IoT sensor based healthcare system for identifying and controlling chikungunya virus. The proposed system provides real time data, collected by the IoT sensor layer then passed through fog layer where classification and alert generation are handled. The final step is cloud layer, which is consisted of storage and indexes for government agencies and hospitals. Since the virus is spread via female mosquitoes, the solution is fast, reliable and provides accuracy since live feed is provided on the site of operation (Sood & Mahajan, 2017).

Another research named Enhanced IoT-based end-to-end emergency and disaster relief system, proposing a wearable technology, named CROW (Critical and Rescue Operations using Wearable Wireless sensors networks). Rescue personnel on the disaster site is connected to the internet, network communication is handled by Optimized Routing Approach for Critical and Emergency Networks routing protocol. CROW system allow users to generate and record live data feed, utilizing different types of hardware which are Raspberry Pi, sensors and smart phones. Decision making is administrated by the command center, concluded after data analysis (Ben et al., 2017).

Hybrid IoT networks are capable of communicating with vast number of devices with various types of sensors mounted. These hybrid systems are able to collect different types of data and derive an approach on the problem with multiple angles. Whistland: An Augmented Reality Crowd-Mapping System for Civil Protection and Emergency Management, proposing a hybrid system to provide emergency management in case of a natural disaster. The scenario is based on a river in danger of flood, the rescue team is mobilized after using the data collected from a server called GeoData Collector which prepares social network data. Augmented reality (AR) is aiding the team via displaying related information about the site such as sensor data in close approximation, while disaster is scaled by mapping which is implemented by Analytics Dashboard (Luchetti et al., 2017).

Smart cities are employing a wide range of IoT devices in infrastructure systems, in a form of sensors and data collection as well. These IoT devices are also can be employed for disaster recovery, the data collected via sensors and actuators would be adjusted for the goal. Algorithms, programs or hardware based systems would help to maintain order when there is a disaster, eventually situational awareness is granted, preventing lack of management and human error. A specific solution for highway tunnels is proposed in the article, named Managing Emergency Situations in the Smart City: The Smart Signal. The proposed system is able to provide automated sensing in times of hazard, following a warning signal pattern for the civilians in the vicinity. The system is fed by multiple data sources such are temperature, pressure, wind, humidity, light sensors which are then processed by a micro-controller. Energy consumption is also balanced via controller (Asensio et al., 2015).

Communication mediums have assorted types, incorporating different devices and methods. Various type of communication mediums provide reliability as well as usage options. It is necessary to note that, during the time of recovery from a disaster, different communication methods would yield better results, considering probable infrastructure system failure in any of these mediums. In this matter, a research named Implementation of Relay-Based Emergency Communication System on Software Defined Radio, proposing a backup for communication in case of a failure induced by infrastructure. Primary feature of the system delivers situational awareness. In addition to this, the system is capable of relaying communication, eventually recovery from severed data communication. The system is monitoring the deployment area with help of sensors for emergency, triggering the backup communication medium if the base station is out of service or overloaded. Integrated relay node is able to boost the signal of the base station, also take over the base station in order to restore data transmission (Lee et al., 2015).

2.5. Recovery Sites

The main goal of the proposed system is to gather data from disaster site by deploying the unit in a safe distance. To have a clear picture, we can generalize uses of IoT devices in various fields.

I. **Healthcare facilities** have various usage of IoT. Critical hardware availability check, tracking public elements such as patient, healthcare personnel or inventory availability, remote monitoring for health status of patients are the most common examples of utilization of IoT in healthcare.

II. **Factories** also have wide scale of use with IoT. Providing a base for industry 4.0, almost entirely automated factories run via help of IoT sensors, management, flow control of actual production, inventory management, security of assets and personnel, product control to ensure quality, optimization of packaging and storing, logistic management of supply chains are general examples.

III. **Educational institutions** are in need of security at all times. Ensuring a safe learning space for students is possible with IoT, using monitoring system for transportation and attendance, emergency warning systems for management.

IV. **Power plants** which use the IoT devices are aimed to sustain for longer. Data collected with IoT nodes provide predictability, adaptability. Safety of employees and environment is also achieved.

V. **Public apartments** are numbered heavily in populated cities which also require IoT devices to provide functioning residences. Most common apartments use heating systems, lightning, safety ensuring sensors such as alarms for gas, fire hazard detection. Security is also ensured with using IoT nodes.

VI. **Smart cities** have wide range of sensors to gather data which can be later utilized after an analysis resulting in less traffic congestion, accurate prediction of time critical services like public transportation, energy efficient buildings, public safety.

2.6. IoT Protocols

After the introduction of concept of Internet of Things, new methods of data transferring need pioneered the new protocols and connection types for the IoT. There are now many of this protocols and communication methods exist some have standards controlled by the IEEE, IETF and more. Before implementing the gateway and the sensor nodes, it is essential to have general knowledge about these new protocols.

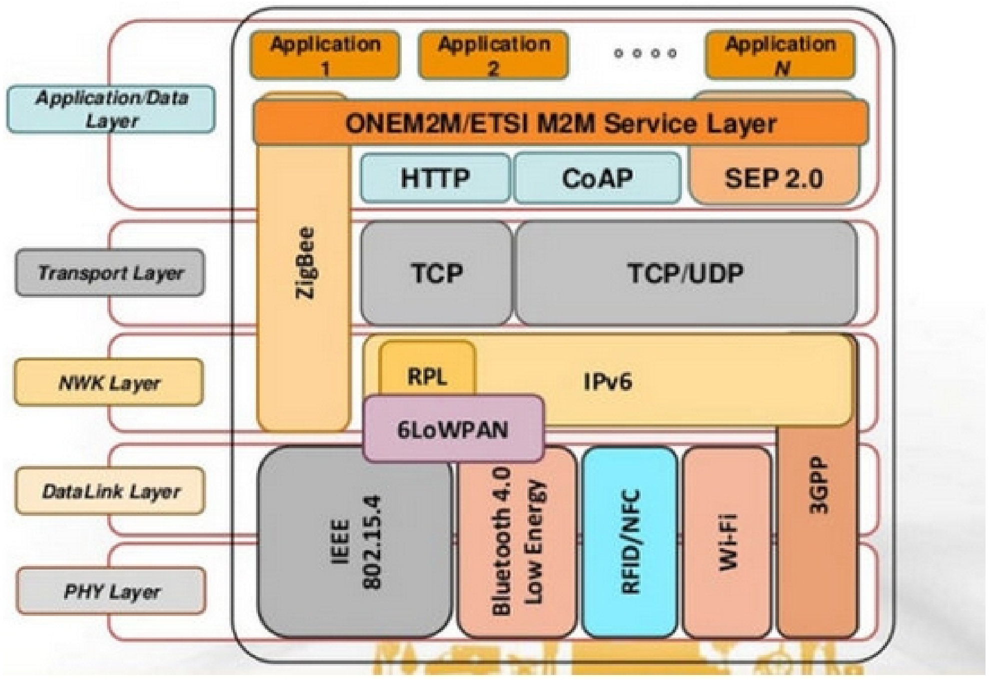


Figure 2.1. OSI and IoT protocols mapping

2.6.1. IoT Data Protocols

I. CoAP (Constrained Application Protocol - RFC 7252) : The most known protocol for the Internet of Things. CoAP is designed for constrained with energy and CPU power. Multicast is supported also machine to machine communication based nodes are in scope of CoAP (Shelby et al., 2014).

II. MQTT (Message Queing Telemetry Transport) : Lightweight message transferring protocol designed for low overhead message like simple binary data, a lightweight and uses TCP/IP connection also is opensource (Andrew et al., 2019).

III. XMPP (Extensible Messaging and Presence Protocol - RFC 3921) : A real time message sender protocol that uses XML and provides TLS (Transport Layer Security). Client and server communication is utilized with XMPP (Saint-Andre, 2004).

IV. AMQP (Advance Message Queuing Protocol) : AMQP allows multiple services to connect each other, organizations or different clients that uses AMQP can work interoperable also an open source protocol available for different platforms (Steve Vinoski, 2006).

V. REST (Representational State Transfer) : REST shows resources between client and server like the conventional internet, using the next possible states. Google search is the best example of REST for instance a simple search results in many options to user to select (Leonard Richardson and Sam Ruby, 2007).

VI. 6LoWPANs (Low Power Personal Area Networks - IETF - RFC 6282) : Used in personal area network nodes which need low energy consumption, also the IPv6 based nodes have a future of large mesh networks that will be implemented for the future. Uses data-link frames for end to end transfer. Mesh network nodes can be extended when needed provided by the protocol, nodes acts as a router (Zach Shelby and Carsten Bormann, 2010).

VII. LLAP (Lightweight Local Automation Protocol) : Short message is used to communicate between devices, this protocol sends the message directly instead of encoding it to a format. Platform independent protocol is easy to understand since it does not encodes the message (Aly Farahat and Ali Ebnenasir, 2012).

2.6.2 IoT discovery protocols

In the previous topic, data transfer protocols are briefly covered. In this section discovery protocols for the nodes will be shown. Some nodes on the IoT systems are easily managed and recognized by other nodes or IoT gateways using these discovery protocols. Traditional discovery methods are used while advertising the nodes.

I. mDNS (multicast Domain Name Service - RFC 6762) : The need of small scaled networks and many nodes on this small networks have raised the need of the discovery of these nodes when there is no classical Domain Name System exist. When there is no presence of Domain Name Service server set up in a network, mDNS can be utilized to resolve the host names (Cheshire, S. and M. Krochmal, 2013).

II. UPnP (Universal Plug and Play) : Manages local sensors and device data and controls the share of the data when or whom to share with in addition to these provides security for the network. UPnP also establishes a discovery between the nodes on the network, supports TCP/IP. Regulated by Open Connectivity Foundation (V. Pehkonen and J. Koivisto, 2010).

III. HyperCaT : Provides a JSON formatted discovery method. Discovery is possible over the internet using the URI (Uniform Resource Information Identifier) based catalogues. Uses HTTPs REST and JSON also provides security also a simple method for the developers (Michalis et al., 2019).

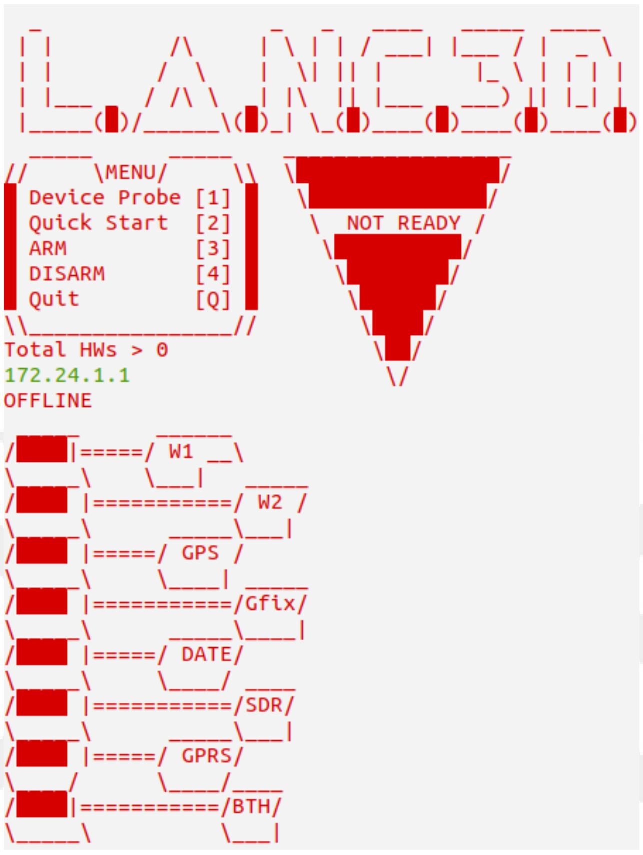
CHAPTER 3
IMPLEMENTATION

The world revolving as we humans develop new technologies and methods for survival. Compared with previous ages, mortality rates are decreasing since developments in healthcare, early warning and recovery systems for disasters resulting in growing population density in cities. In order to maintain a functional society, infrastructure systems are formed to satisfy the needs of populated cities, governments and countries. There are several concerns for managing a healthy population such as men made or natural disasters. As technology have uses of aiding mankind, it also has destructive capabilities also created by men. Such harmful intentions can destroy entire cities even nations. Natural disasters also cause mass panic like war and terrorism, happening any year by anytime regardless of location and population count. Scenario for this thesis is aiming to include not all but most of situations may occur anytime, instead of creating a specific scenario for any of these disasters, a generalized approach is better to avoid situational details.

3.1. L.A.N.C.E.D

L.A.N.C.E.D. (Lightweight Automated Network Component Enumeration Device) will be organizing several modules simultaneously to achieve data collection of IoT devices with Wi-Fi, Bluetooth/BLE and sub-GHz capabilities while providing WAN access as an auxiliary connection for maintaining data access. LANCED device will be using a BASH program for automated installation (lanced_installer) and another one (lanced) for GUI and operation.

The system will be utilizing several open-source programs as well as essential hardware devices for operation. Lanced device will be providing multiple interfaces for the user to operate the device covering monitored and without monitored usage options. BASH language will be used to implement almost all of the program, to complement the capabilities of BASH, Python language may be needed, provides wide range of libraries also easy to implement and use with other programs. Main operations will be handled by BASH, interfacing with hardware and software, storing data in respective folders, also installation will be done by another part of the main program.

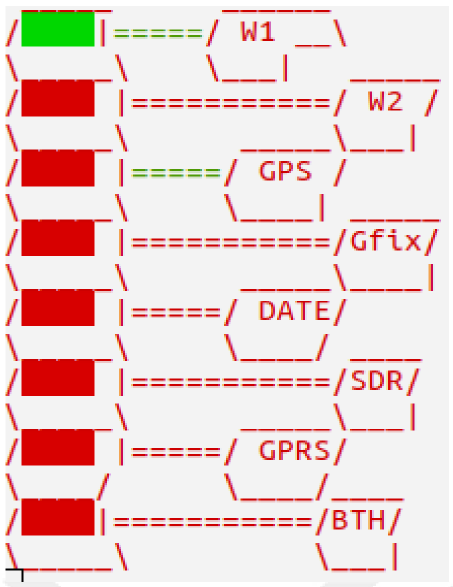


```
Total HWs > 0
172.24.1.1
OFFLINE
```

Figure 3.1. LANCED GUI initialized

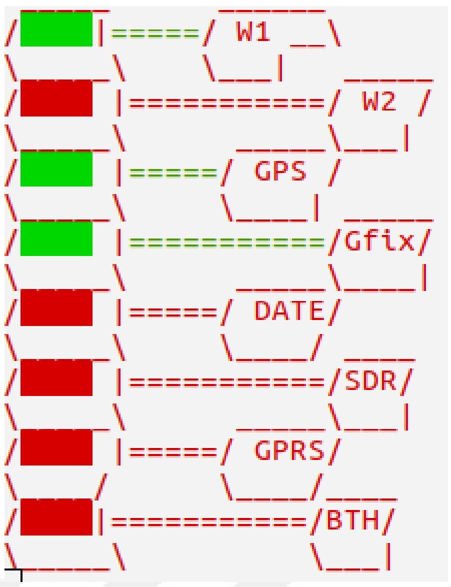
To achieve the goal of creating a stand-alone device that performs wireless spectrum analysis, passive data collecting and logging, ease of use and mobility shall be provided for the user. GUI must provide status of the device included hardware and software modules, also minimal menu keys to reduce response time of the user. Monitoring and interfacing with the device is available via a keyboard and screen or SSH connection with the terminal. Our prototype is capable of using external batteries or alternative power sources such as cars, mobile stations. Connection with an extension cable to keep hardware modules apart or elevated depending on environment and situation is possible as well.

LANCED is designed to handle automation with aim of reducing user interaction resulting in reduced response times that is critical for managing emergency situations. Operation of the device lets the user to have an elevated view of disaster site that contains IoT devices with Wi-Fi, Bluetooth and sub-GHz variations.



```
 _____       ______
/🟩|=====/ W1 __\
\_____\    \___|   _____
/🟥 |===========/ W2 /
\_____\      _____\___|
/🟥 |=====/ GPS /
\_____\     \____| _____
/🟥 |===========/Gfix/
\_____\      _____\____|
/🟥 |=====/ DATE/
\_____\     \____/ ____
/🟥 |===========/SDR/
\_____\      _____\___|
/🟥 |=====/ GPRS/
\____/      \____/____
/🟥|===========/BTH/
\_____\             \___|
```

Figure 3.2. GUI state I



```
 _____       ______
/🟩|=====/ W1 __\
\_____\    \___|   _____
/🟥 |===========/ W2 /
\_____\      _____\___|
/🟩 |=====/ GPS /
\_____\     \____| _____
/🟩 |===========/Gfix/
\_____\      _____\____|
/🟥 |=====/ DATE/
\_____\     \____/ ____
/🟥 |===========/SDR/
\_____\      _____\___|
/🟥 |=====/ GPRS/
\____/      \____/____
/🟥|===========/BTH/
\_____\             \___|
```

Figure 3.3. GUI state II

LANCED menu offers minimal design for the user while allowing operation and diagnostics for the hardware and software modules for troubleshoot.

Operation of the proposed system is starts after powering up the device. There are three modes of operation are available for the user for operation. First method allow user for quick deployment, after the OS boot process, LANCED is able to start the monitoring process.

Second method allow the user to interact with the proposed system via multiple ways. HDMI display port on RPI board is accessible for the user to connect any type of monitor to provide graphical feedback. The user also is able to connect any keyboard and mouse via any of unused USB ports on the proposed system. Desktop variation of the Raspbian OS offers conventional user interaction that is sufficient for any desired modification on the OS. Terminal variation on the other hand is more compact but efficient, allowing user to interact with CLI screen.

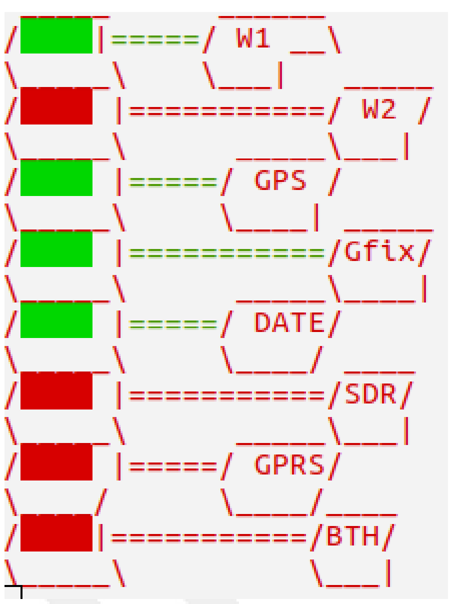


Figure 3.4. GUI state III

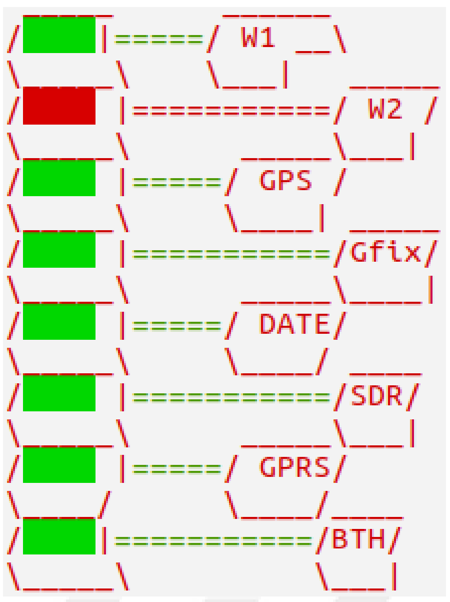
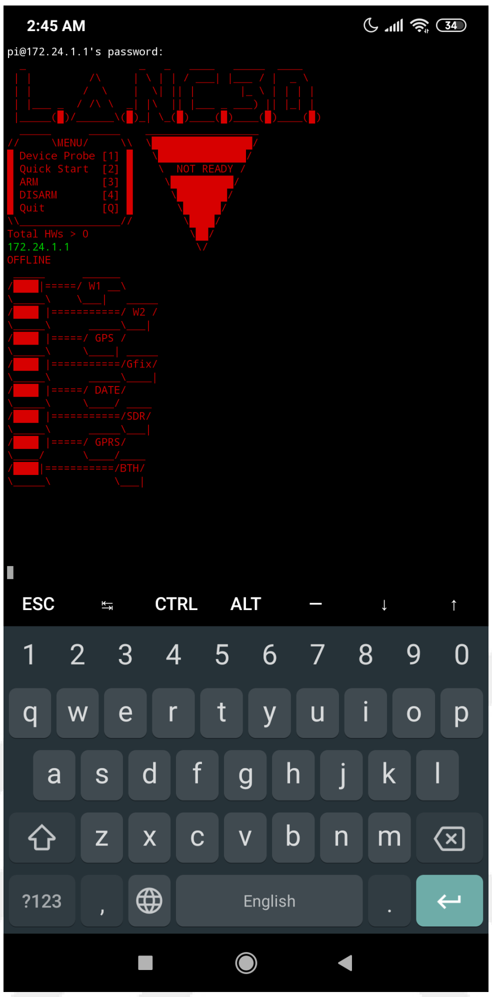


Figure 3.5. GUI state IV

Older versions of terminals were implemented on hardware setup unlike recent versions, instead of hardware like TeleTYpe writer called TTY, software based terminals called terminal emulators are in use for modern computers (Powers et al., 2003). Simulated terminals for instance xterm, is available on a graphical display. Popular Unix based OSes have a variety of terminal types, xterm, gnome, LXterminal, xfce4 are the most general types. Design of the GUI of LANCED is handled by main module called LANCED.sh. Compatibility for certain ANSI characters have constraints in different terminal types, to overcome compatibility issues, Linux TTY terminal supported ANSI characters preferred for user display. Linux tty encoded characters allow LANCED to be displayed with any other terminal consequently GUI is supported by any platform with different types of terminals.

Third method of use is possible with connection via SSH through any mobile phone with installed terminal application (see Figure 3.6), or a laptop with UNIX based OS installed. RPi internal Wi-Fi NIC is used as an AP to provide Wi-Fi connection for one or more users, after connection is established with the LANCED AP, terminal access grants user to run required programs for operation. AP ip is previously defined while setting up Hostapd and is displayed under MENU section (see Figure 3.1). Capability of AP of the proposed system is achieved with program called Hostapd, with parameters set by LANCED installer. Setting of the AP can be modified as desired



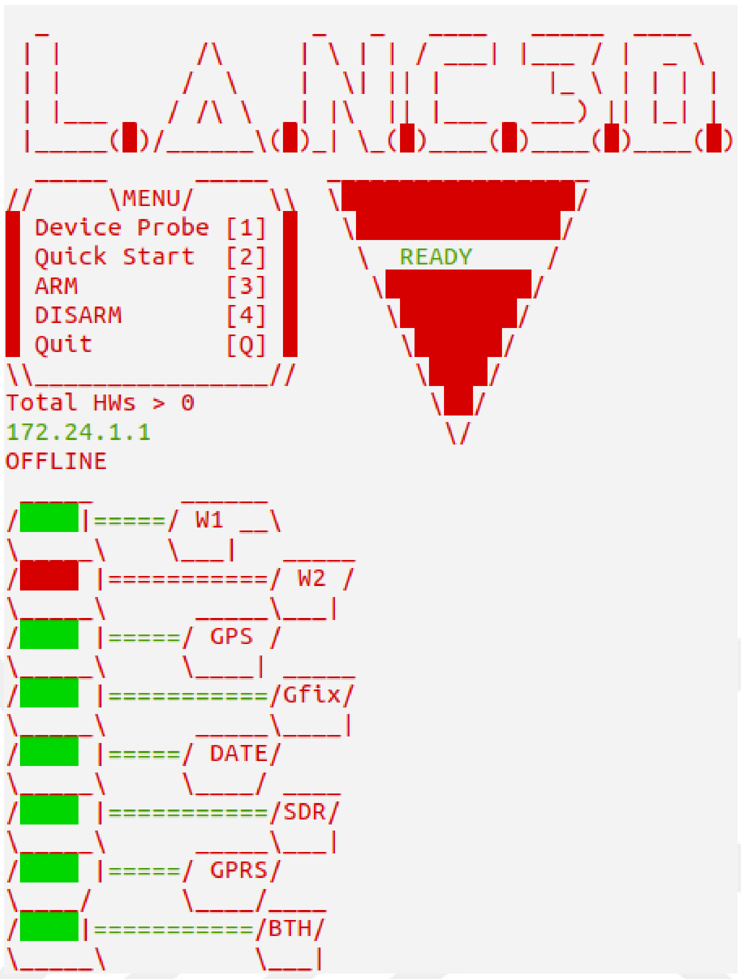
```
pi@172.24.1.1's password:

//      \MENU/      \\
  Device Probe [1]
  Quick Start  [2]
  ARM          [3]
  DISARM       [4]
  Quit         [Q]
\\__________________//
Total HWs > 0
172.24.1.1
OFFLINE

|=====/ W1
|===========/ W2 /
|=====/ GPS /
|===========/Gfix/
|=====/ DATE/
|===========/SDR/
|=====/ GPRS/
|===========/BTH/
```

Figure 3.6. Terminal output with smartphone

after installation of the software modules. Hostapd is a user space daemon for access point and authentication servers, under the terms of BSD license. It implements IEEE 802.11 access point management, IEEE 802.1X/WPA/WPA2/EAP Authenticators, RADIUS client, EAP server, and RADIUS authentication server. The current version supports Linux (Host AP, madwifi, mac80211-based drivers) and FreeBSD (net80211). LANCED device will be providing WiFi connection for managing the device via Hostapd program (HOSTAPD, 2002).



```
//      \MENU/      \\
  Device Probe [1]
  Quick Start  [2]
  ARM          [3]
  DISARM       [4]
  Quit         [Q]
\\________________//
Total HWs > 0
172.24.1.1
OFFLINE
```

Figure 3.7. GUI state V

Organization of the hardware modules and software programs are handled with LANCED.sh (see Figure 3.7). GUI is consisted of sections with minimal design. Ease of use based on minimal user selection with five keys at most displayed in MENU section on GUI (see Figure 3.8). After boot process is completed, user is able to probe the connected devices with pressing key 1. LANCED program then initiates an integrity check for connected interfaces for physical connection followed with software interface capability. Device probe process checks for Wi-Fi NICs while providing support for one or two Wi-Fi NICs. The LANCED system achieves Wi-Fi AP tracking capability with using program called kismet and one or two Wi-Fi NICs. Kismet is an 802.11 wireless network detector, sniffer, and intrusion detection system, under the terms of GPL-2.0 license. Kismet will work with any wireless card which supports raw monitoring mode, and can sniff 802.11b, 802.11a, 802.11g, and 802.11n traffic (devices and drivers permitting). Kismet also sports a plugin architecture allowing for additional non-802.11 protocols to be decoded. Kismet identifies networks by passively collecting packets and detecting networks, which allows it to detect (and given time, expose the names of) hidden networks and the presence of non-beaconing networks via data traffic (KISMET, 2002).

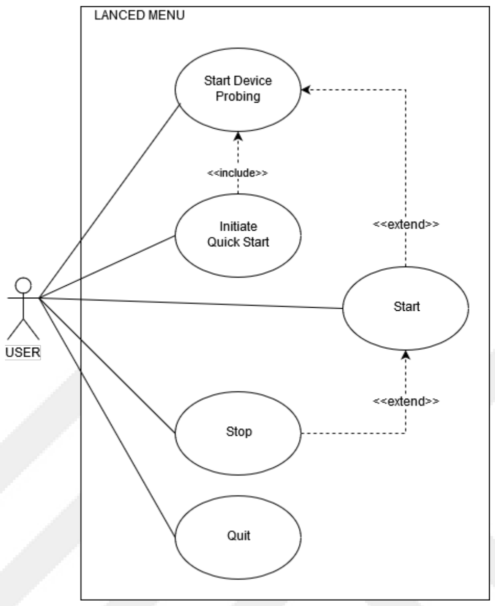


Figure 3.8. LANCED GUI user use case diagram

GPS module if connected via USB interface, following with GPS acquisition. GPS acquisition also provides Raspbian OS time correction since RPi board does not contain a RTC module on the board. GPSD is a service daemon that monitors one or more GPSes or AIS receivers attached to a host computer through serial or USB ports, making all data on the location/course/velocity of the sensors available to be queried on TCP port 2947 of the host computer. Application that presently use gpsd include Kismet, GpsDrive, gpeGPS, roadmap, roadnav, navit, viking, tangogps, foxtrot, odbgpslogger, geohist, LiveGPS, geoclue, qlandkartegt, gpredict, OpenCPN, gpsd-navigator, gpsd-ais-viewer and firefox/mozilla. In addition the Android smartphone operating system (from version 4.0 and later) uses GPSD to monitor the phone's on-board GPS, so every location-aware Android AP is indirectly a GPSD client (GPSD, 1995). GPS data is interfacing with other programs via GPSD program which is able run as a daemon on the Raspbian OS. GPSD program has many abilities such as formatting of GPS data, while is able to establish GPS acquisition with any mobile phone set as GPSD server. LANCED program is able to detect if there is a mobile phone acting as a GPSD server, in case of absence of GPS hardware module, the user is not constrained to use a GPS hardware module connected via USB port. After connection of the module is detected, interface connection bar is displayed in green text is refreshed from red (see Figure 3.2)(see Figure 3.3).

Function name	Tasks/Description
components_chk()	hardware and software module diagnostics is performed for user GUI report
start_ks()	if overall status of the device is ready, start device discovery
stop_ks()	End processes of discovery programs
ln_quit()	End Point-to-Point protocol daemon then power off GPRS module

Table 3.1. : LANCED functions example

SDR module install on the proposed system is probed for connection interface followed by status check, if succeeded with probing then LANCED program indicates the progress on the GUI(see Figure 3.4)(see Figure 3.5). Sub-Ghz devices those are subjected for monitoring on the disaster site are scanned via SDR module that is RTL-SDR v3 for this thesis. Devices on the site broadcasting radio messages with frequency of below 1 GHz are have standards which are not entirely mapped for detection, in some cases some devices are in requirement of RF signal to be mapped. The scope of this thesis therefore must be extended, thus the proposed system is designed to provide RF detection for previously mapped devices with below 1 GHz. Open-source program named rtl_433 under GPL-2.0 license, supplied with required capabilities is able to scan for designated devices. Rtl_433 is a generic data receiver mainly for the 433.92 MHz, 868 MHz, 315 MHz and 915 MHz ISM bands (RTL_433, 2012).

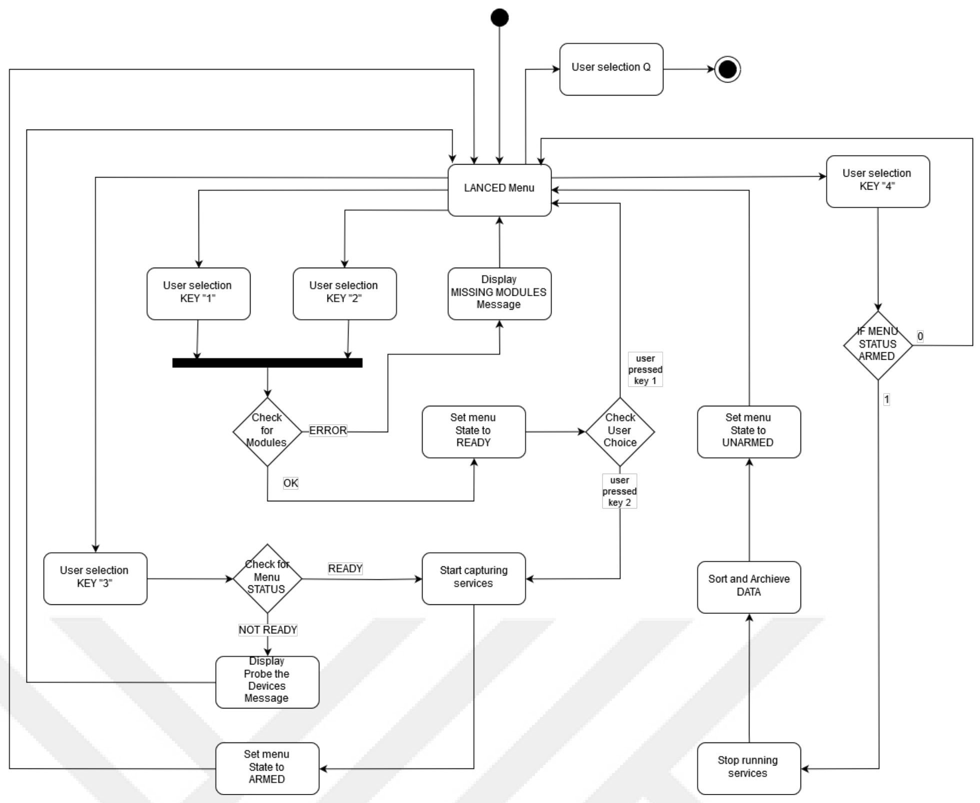


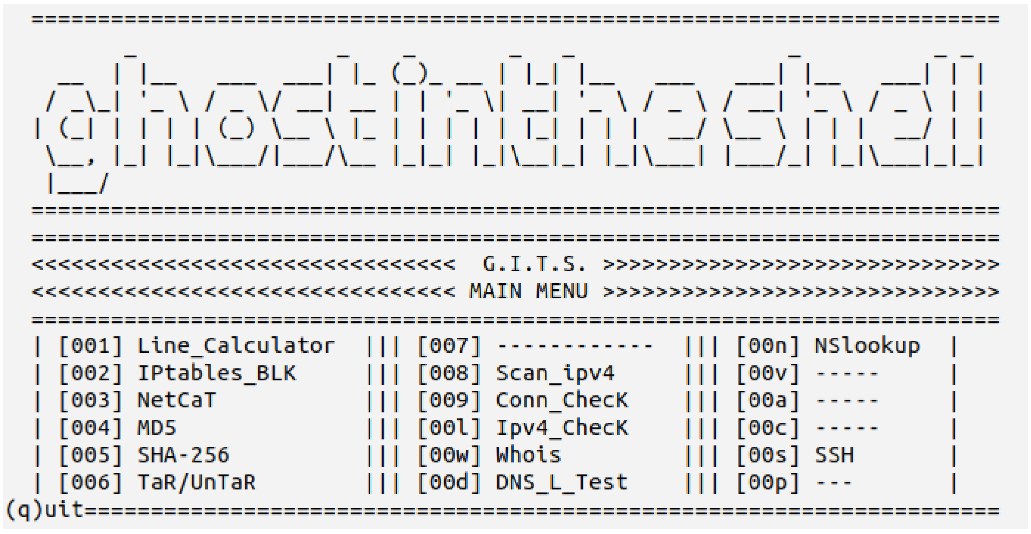
Figure 3.9. LANCED activity diagram

GPRS module installed on the RPI board is interfaced with GPIO pins, and mounted on top of the board. LANCED program initializes the GPRS shield after checking the power status of the GPRS shield. GPIO pins are suitable for this, power status is checked with probing via GPIO 15 with BCM pin specification. Program called gpio supplied within Raspbian OS is able read voltage value of the desired pin, thus if the GPRS shield is not powered, LANCED program initiates a secondary python module for powering the shield. GPIO pin 25 (under BCM) is used for powering the shield if supplied with power with at least two seconds. LANCED program then uses gpio program to read the value of GPIO 15 to sense power status of the GPRS shield. GPRS module on the proposed system interfacing with the SIM module with desired GSM carrier, establish internet connection via program called pppd, which is based on PPP. The Point-to-Point Protocol (PPP) provides a method for transmitting datagrams over serial point-to-point links. PPP is composed of three parts: a method for encapsulating datagrams over serial links, an extensible Link Control Protocol (LCP), and a family of Network Control Protocols (NCP) for establishing and configuring different network-layer protocols. The encapsulation scheme is provided by driver code in the kernel. Pppd provides the basic LCP, authentication support, and an NCP for establishing and configuring the Internet Protocol (IP) (called the IP Control Protocol, IPCP), with under the terms of licenses BSD and GPL-2.0 (PPPD, 2014). Internet connection then is shared with devices connected via AP. Raspbian OS then is able to deliver WAN access via Iptables program which forwards request through ppp0 interface. Internet connection status is displayed under AP IP, with a green ONLINE text for WAN access and red OFFLINE text if connection is not supplied.

Detection of Bluetooth and BLE supported IoT devices are supplied with onboard Bluetooth. LANCED program probes for Bluetooth device status for status indication via GUI. Discovery of Bluetooth and BLE devices depending on the program named BlueHydra, built on bluez library (BLUEHYDRA, 2015), copyrighted by Rapid Focus Security Inc d/b/a Pwnie Express.

GUI indication of overall status indicated in triangular display with green READY message after system integrity is verified. LANCED system is then ready for operation, the user can trigger the monitoring process by pressing 3 (see Figure 3.7). Monitoring is in process until receiving a halt signal from the user by pressing 4, resulting in start of DISARM process. Both ARM and DISARM process regulates run and halt signals to respective programs, stopping or starting them in a proper way.

LANCED program regulates collected IoT device data after each operation process, processed data is saved into respective folder which are created and named by LANCED, also timestamps are used to sign data for duplicated data. LANCED program is also keeping track of unique devices to avoid data duplication. Number of collected devices are indicated after each operation informing the user (see Figure 3.7).



```
  ========================================================================
         _               _   _       _   _                _          _ _
    __ _| |__   ___  ___| |_(_)_ __ | |_| |__   ___   ___| |__   ___| | |
   / _` | '_ \ / _ \/ __| __| | '_ \| __| '_ \ / _ \ / __| '_ \ / _ \ | |
  | (_| | | | | (_) \__ \ |_| | | | | |_| | | |  __/ \__ \ | | |  __/ | |
   \__, |_| |_|\___/|___/\__|_|_| |_|\__|_| |_|\___| |___/_| |_|\___|_|_|
   |___/
  ========================================================================
  ========================================================================
  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  G.I.T.S. >>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< MAIN MENU >>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  ========================================================================
  | [001] Line_Calculator  ||| [007] ------------  ||| [00n] NSlookup  |
  | [002] IPtables_BLK     ||| [008] Scan_ipv4     ||| [00v] -----     |
  | [003] NetCaT           ||| [009] Conn_ChecK    ||| [00a] -----     |
  | [004] MD5              ||| [00l] Ipv4_ChecK    ||| [00c] -----     |
  | [005] SHA-256          ||| [00w] Whois         ||| [00s] SSH       |
  | [006] TaR/UnTaR        ||| [00d] DNS_L_Test    ||| [00p] ---       |
(q)uit====================================================================
```

Figure 3.10. G.I.T.S. GUI

Some functions, used by LANCED are provided via an utility tool called G.I.T.S., stands for Ghost In The Shell. GITS is consisted of collection of various UNIX programs, delivering ease of use to the user with many functions with shortcuts structured via BASH scripts. Internet connection status for instance is checked by a function included in GITS. The user is allowed to use tools included in GITS as a separate program with a GUI with shortcut keys such as SSH program, instead of passing parameters in order, GITS ask the user for parameters then executes the program. Compressing files, internet connection status, file integrity check via MD5 sum, VPN connection if provided with subscription are some of the tools those GITS is offering to the user (see Figure 3.10).

3.1.1. Operating System

IoT devices have many options for OS to be implemented on. In terms of community support and frequency of use, Raspbian OS is great platform, an open-source software with required flexibility for any application to be developed. RPI based hardware is supported by Raspbian OS as well as many IoT modules on the market. Raspbian OS is recommended for beginners and skilled user both, GUI for beginners and lightweight terminal version for skilled for experts.

Raspbian is in the family of Unix-like operating systems. Raspbian OS offers huge advantage compared to other OSes, in terms of flexibility open-source based kernel is possible for any type of configuration. Tailored for the specific needs of the user, documentation for almost any modification is available on the internet supplied in a regular basis from open-source community (Bovet & Cesati, 2005). Based on Debian OS, Raspbian offer more than 35,000 packages, also pre-compiled software.

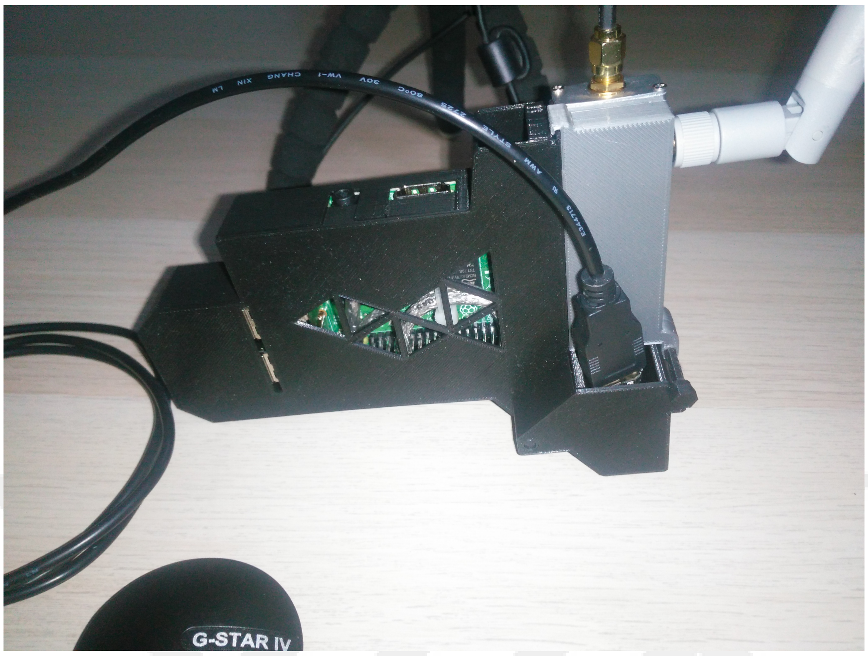


Figure 3.11. L.A.N.C.E.D MKI front view close up

3.2. Hardware Components

3.2.1. Raspberry Pi

Raspberry pi a well known tiny computer, providing a sufficient processing power while keeping a low price. Main market audience of the RPI boards are inventors, hobbyists, students, developers. The board is able deliver any function including web browsing, video playing and such. The Raspberry Pi 3 Model B+ provides 64-bit quad core 1.4 GHz of CPU power, dual-band 2.4 GHz and 5 GHz wireless LAN, Bluetooth 4.2/BLE, 1 Ethernet port and PoE feature, 1 HDMI port, 4 USB ports, 1 audio I/O port.

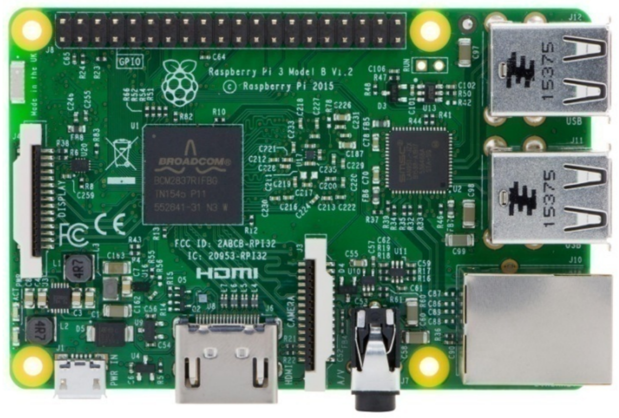


Figure 3.12. Raspberry Pi

Storage unit is not built-in so any 16 GB SD card is available to be used with micro SD card slot. Operating system options are wide since RPI is a preferred device for prototyping and development. Windows 10 IoT core, Ubuntu Mate, Raspbian are the most popular OS options for the RPI, consequently development languages are not constrained by OS.

Eventhough the price of the device is considered cheap, Wi-Fi and Bluetooth technologies are provided. There are many projects available on the internet since Raspberry boards are popular among Do-It-Yourself community consequently a vast number of applications are available to everyone to tinker. These aspects of the device also provides a simple design, makes it suitable for designing mobile or immobile devices.

Raspberry Pi 3 Model B+ is the base board that is used for the design. External NIC, GPS module, SDR modules which are secondary modules are planned to connected via USB ports for interfacing, total of 3 USB ports on the Rpi board is dedicated for hardware modules. RPI GPIO pins are in use as well, providing the interface for GPRS module which is able to nest on the RPi board via connecting the headers (see Figure 3.12).

3.2.2. Network Interface Card

TP-Link TL722WN v1 is a WiFi NIC that uses AR9271 single chip with 802.11b/g/n support. This product operates on 2.4 GHz frequency band with 802.11b/g/n network standards and is able to provide AES, TKIP, WEP hardware encryption. Supported data rates are 11b/g/n which is supplied with USB 2.0 as communications interface. This product is not considered expensive and durable with large antenna, providing required features for developers and tinkers, students, also wireless penetration testers with monitor mode support.

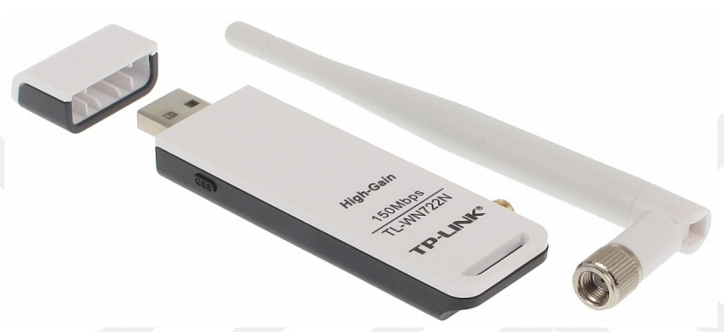


Figure 3.13. TP-Link TL722WN NIC

The Atheros AR9271 single-chip USB is able to provide reliable Wi-Fi connection for any device, gaming consoles, personal computers, home gateways, laptops. This model also supports Ad-Hoc feature. The drivers for the AR971 chip is also easy to use with built-in Linux kernel support (see Figure 3.13).

The system recommends two WiFi NICs, which provides wider range of scanning with different channels and frequencies. These NICs are probed before starting the Kismet server and are not allowed to function other than to be used by kismet server. NICs are interfacing with the system via USB ports. MKI model is mounted with a AR9271 NIC which is sufficient for implementation purpose, if desired an additional NIC can be supplied with extension module casing. The design of the case for external NIC is allowing two modules to be attached on the MKI model. MKI model is bearing an external NIC with gray colored NIC module casing, printed with PLA material (see Figure 3.17).

3.2.3. Software Defined Radio

Radios are consisted of modulators, demodulators and tuners. Implementation of these hardware is possible with the software based applications with the modern computing. Software defined radios are the result of modern computing, is available to everyone providing functionality of a wide band radio scanner. Received signals are filtered with band selection filters with a processor resulting in desired sampled data output (Jondral, 2005). SDRs have many functions, radio scanning, tracking or receiving messages from weather balloons, communicating with amateur radios, watching TV broadcasts, receiving GSM signals, listening to satellites, listening to FM radio, providing a high quality entropy source for random number generation are just the tip of vast number of functionalities.

RTL-SDR RTL2832U v3 is a computer based radio scanner with low price and USB connection support. Based on TV tuner, this product is able to receive radio signals within range of 24MHz to 1.8 GHz. If purchased as a kit, small and large antenna is provided with SMA connector. Magnetic mount with thin cable, two telescopic antennas (larger one) are able to extend 1.5 meters from 20 centimeters.

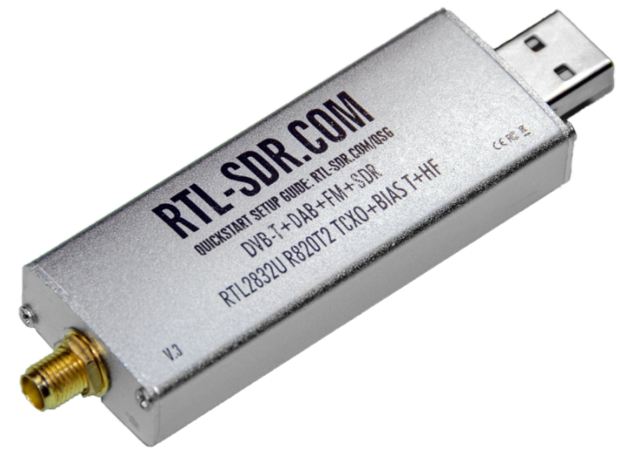

RTL-SDR.COM
QUICKSTART SETUP GUIDE: RTL-SDR.COM/QSG
DVB-T+DAB+FM+SDR
RTL2832U R820T2 TCXO+BIAS T+HF
V3

Figure 3.14. RTL-SDR RTL2832U

Providing compatibility to be used with other SMA supported radio gear, SMA connector standard makes it possible to use the device with handheld trans-receivers. Frequency changes caused from heating of the device called drift, is also handled by the v3. The outer aluminum shell of the v3 is also helping radiating heat. Device is designed to provide robustness, environmental factors such electrostatic discharge caused by air is not a problem for the v3. Compared with similar models, v3 handles the required functions for it's price. Outdoor usage with the mount produce better results. RTL2832U v3 is a great choice for educators, penetration testers, students and tinkers, providing usability for any skill level with many software options for Windows and Linux based devices. This model is a great choice for the LANCED system with features of robustness, portability, low price and reliability (see Figure 3.14).

Discovery of Sub-Ghz devices is possible with the capabilities of the RTL-SDR module described in Chapter 2. Interfacing of the module is handled with USB port connected directly to upper section of the MKI case (see Figure 3.17). Usage of SDR module may vary for user to user, fortunately MKI is also allowing a casing module for SDR board as well as an extension cable for the SDR in case of the module is placed apart from the MKI case.

3.2.4. Global Positioning System

Humans needed navigational assistance since history of mankind, this need is based on discovery of new frontiers. Through the years of development brought us many ways of navigational methods and techniques such as ancient Polynesians, they used measurements of starts to navigate. Utilization of radios in modern world today brought new methods of navigational devices, which are used in complex machines for operation such as planes. In 1973 a group of people started a revolution for the navigation with using the satellites as a radio range measurement node to achieve accurate positioning. First models of the GPS system is designed for military usage, called NNSS (Navy Navigational Satellite System) developed at John Hopkins Applied Physics Laboratory (APL). Broadcast messages from GPS satellites containing one-way range data of the GPS satellite locations are correlated with three or more GPS satellites with the replica signal that is created by the user. Provided four GPS satellites, latitude, longitude, altitude and the correction to the user's clock are determined (Bradford & James, 1996).

GlobalSat G-STAR IV BU-353S4 is a GPS receiver that uses SiRF Star IV GSD4e chipset, running with frequency of 1575.42 MHZ. Supports NMEA0183 and SiRF binary data protocols, also provides IPX6 water resistance. GPS transfer rate is set to 4800 as baud rate with the default setting (see Figure 3.15).

Figure 3.15. G-STAR IV GPS module

BU-353S4 has the support for Windows, Mac and Linux OSes with drivers available. The casing provides a magnet for attaching the module on magnetic surfaces with the cable length of 152 centimeters which is sufficient for placing it outside of the buildings or vehicles. After the connection is provided with the computer, BU-353S4 is usually locked from a cold start in less than a minute. Environmental obstructions must be avoided since signals are absorbed by concrete and surrounding trees. This model is considered cheap with providing multi platform support for students and developers also is used by nautical navigation. Well received as a reliable module with more than one field applications such as mapping, rescue, astronomy and nautical purposes, BU-353S4 is a good choice for IoT applications.

Main functions of this subsystem is coordinate tracking and local time correction. Since Raspberry does not have an RTC module, after rebooting the OS date is inaccurate resulting in incorrect timestamps. GPS module is responsible for geographical location tracking and time correction for the OS that is running on RPI board. GPS are interfacing with the system via USB port (see Figure 3.17).

3.2.5. General Packet Radio Service

GPRS system initiated in 1994, as the standards published in 1997. The European Telecommunications Standards Institute released the drafts for specifications on the subject. GPRS technology is aimed to provide efficiency on the data sources utilized by GSM license holders, ensuring physical resource sharing between GSM services. GPRS is able to use GSM frequency bands while bearing the capability of TDMA for using the GSM time slot structure. GPRS system architecture have the function of creating an end-to-end packet transfer with two different elements compared to GSM. The new logical network node called GPRS support node (GSN), encapsulating the packets within the desired service area, then decapsulation process is completed at the destination GSN, whole process is called tunneling in GPRS (Cai & Goodman, 1997).

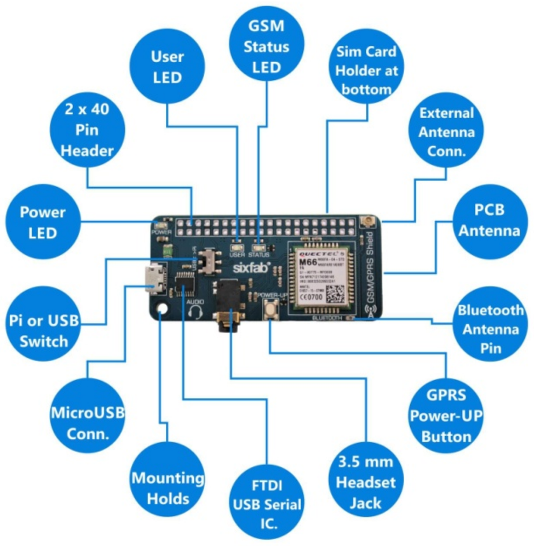


Figure 3.16. Sixfab Raspberry Pi GPRS SHIELD

SIXFAB Raspberry Pi GSM/GPRS Shield is designed for RPI GPIO layout with short and long header options, enabling internet connection via Quectel M66 module on the add-on. SMS and data transmission with audio call is provided GSM/GPRS module based on 2G chipset M66 chip. The add-on is able to connect the RPi with both GPIO or USB connection. Built-in PCB antenna is included for using it without an external antenna, also an external antenna port is available with the design for better results. Supported protocols are TCP/ UDP / PPP / FTP / HTTP / SMTP / CMUX / SSL. The shield is has micro SIM card socket. There are notable applications with the shield for various IoT devices, farming sensors, smart home sensors, environmental monitoring, smart door locks and smart smart lightning are possible with it. Price is about 35 Euros on the Sixfab official site. There are instructions for the shield as well tutorials and setups for various applications on the internet, the official site of Sixfab company is also helpful with providing support for the developers with any skill level (see Figure 3.16).

GRPS module is connected directly to RPI board via GPIO pins. MKI design unfortunately is not allowing a GPRS module, a later design is planned for GPRS extension with a new design allowing space for the module. GPRS module is also able to interface with RPI board via USB ports, an external casing for residing GPRS module is possible for implementation (see Figure 3.17).

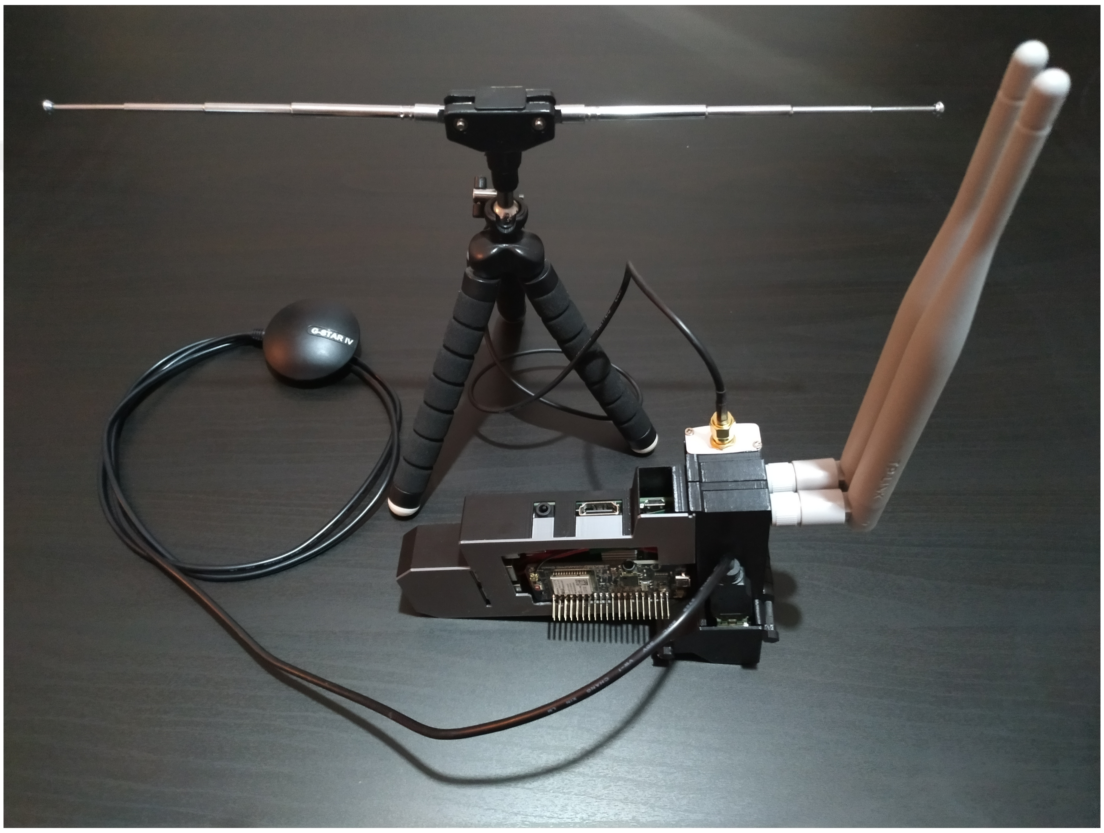


Figure 3.17. L.A.N.C.E.D MKI with GPS/SDR/NIC/GPRS modules

3.2.6. Built-in Wireless Network Interface Card

Raspberry pi 3 models provide built-in wireless NIC. The user is able to communicate with the system via two ways; first is using a external screen and a wireless keyboard. Second method provide WiFi AP via Hostapd software, after connecting the AP of the system, user logins the device using a ssh connection provided with IP and PORT. While tracking process is active, other two Wi-Fi NICs are not able to provide the user a connection (see Figure 3.12).

3.3. How to find IoT Devices in the Disaster Area

Wide range of IoT devices are available for the implementation however this thesis' scope is to achieve a conceptual design. Almost any IoT module is compatible for sensing and data extraction, these devices use different protocols and communication mediums. Common communication technologies such as Wi-Fi, Bluetooth and sub-GHz are used in implementation.

I. **Wi-Fi** connected devices are the most common examples of IoT devices, connectivity to internet is achieved through gateways which may or may not be an coordinator node. TP-Link TL722WN v1 is a Wi-Fi NIC that uses AR9271 (Atheros) chip, compatible with Kismet software. Recent versions (v2 and v3) are Using Realtek RTL8188 chip, unfortunately not supported by Kismet.

II. **Bluetooth** can be found everywhere, easy to use and requires low power. This properties allow Bluetooth to be easy to adapt and use. There are wide range of IoT devices with Bluetooth connection capability. Raspberry Pi provides an embedded Bluetooth module, can be supplied with Bluehydra program to scan Bluetooth devices.

III. **Sub-Ghz** modules are usually sensor modules which are the "Things" part of the IoT. These nodes are scattered around via broadcasting in sub-Ghz, providing availability and low power consumption as well. RTL2832U v3 is a computer based radio scanner with low price and USB connection support. Based on a TV tuner, this product is able to receive radio signals within range of 24MHz to 1.8 GHz.

3.4. Scenario

Initial scenario is based on some situational factors. Disaster site is not allowing physical access to the building, structure or construct. Disaster site is not powered by infrastructure so power loss is expected or just happened, resulting in shut down on devices without power supply or auxiliary generator. Disaster site WAN access is severed by any hazardous circumstances. Loss of infrastructure is occurred or will be occurred.

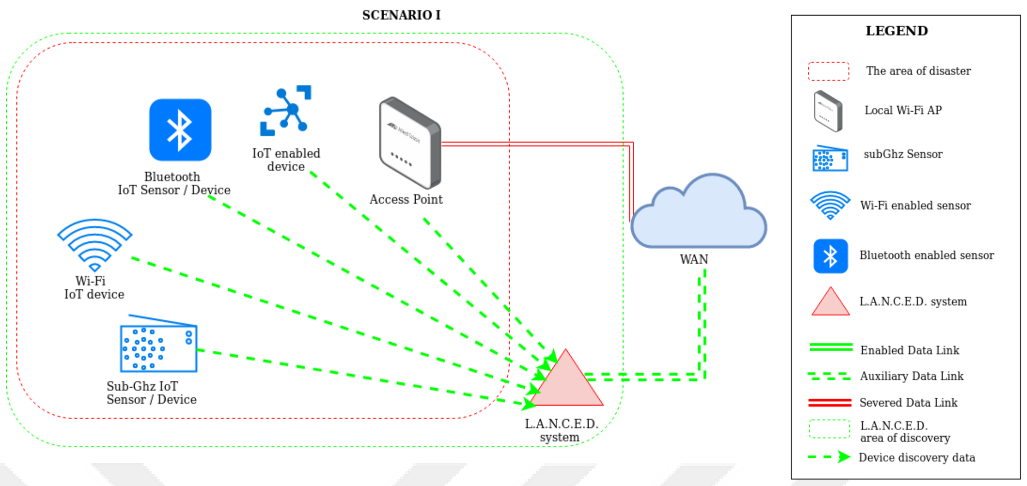


Figure 3.18. Scenario I

The disaster area is assumed to be restricted to access, caused by fire or flood. Infrastructure services are not available. In this scenario LANCED is not authorized previously to serve as an alternative AP. LANCED device is set for scanning for any active IoT device in the area to report general information (see Figure 3.18).

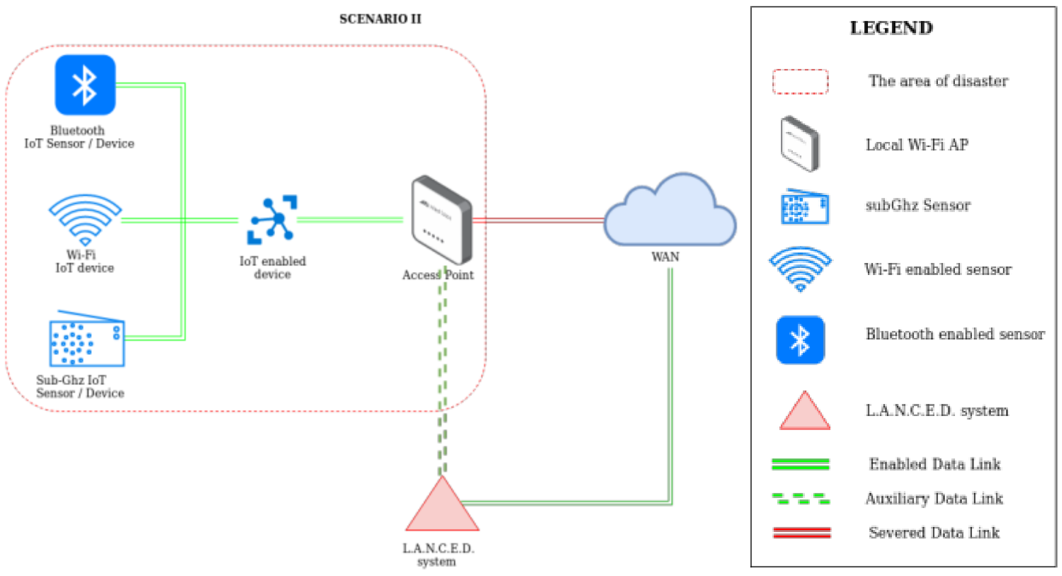


Figure 3.19. Scenario II

This scenario is based on device recovery via fallback logic. The user is already deployed the LANCED system with required parameters to achieve Internet access to the IoT devices in the area. LANCED is run on close proximity, allowing IoT devices on the site to reroute Internet connection via GPRS connection, therefore WAN access is restored for data transmission. This scenario assumes an authorized personnel is already provided credentials for AP (see Figure 3.19).

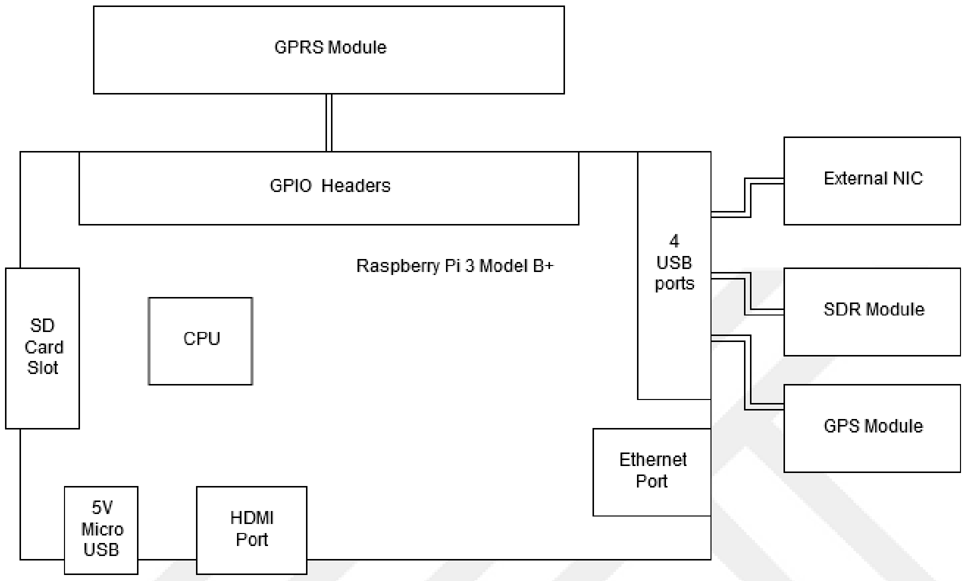


Figure 3.20. Overall layout of hardware modules of LANCED system

3.5. Features

Our prototype detects Wi-Fi devices in close proximity with data collected by analyzing broadcast messages, manufacturer data, signal strength. Bluetooth devices are also in scope of targeted devices so any devices in the subject of matter disaster site shall be recovered via analysis of collected data during our session. The last requirement is sub-Ghz modules which are collection nodes reporting data to other IoT devices in the vicinity. If administrated with AP SSID and password, provide the WAN access using GSM module for auxiliary connection for IoT gateway connections. Provided with MAC address of the previous AP, devices on the site will be connected to the auxiliary AP of the device (see Figure 3.20).

3.5.1. Case

LANCED system casing design is aimed to provide certain capabilities, allowing user to change or replace desired modules or even mix and match with various combination. Simple design for external modules allow the user to integrate desired modules with it's current casing. Mobility is a critical factor for the design, allowing operation while in mobile and immobile states.

Outer casing for the LANCED system is produced with 3D printing. Printing is made by Prusa i3 MK3 by Prusa Printers. MK3 model of Prusa i3 is the latest model of i3 series with reasonable price. Operation of the i3 is not complex, moreover Prusa Printers provide support for both software and hardware. Compared with models within same price range, i3 MK3 surpasses them, offering considerable quality of high price range printers on the market. Software for creating g-code files is supplied by Prusa Printers customized for the model. Slic3rPE software handles material and print options for different materials.

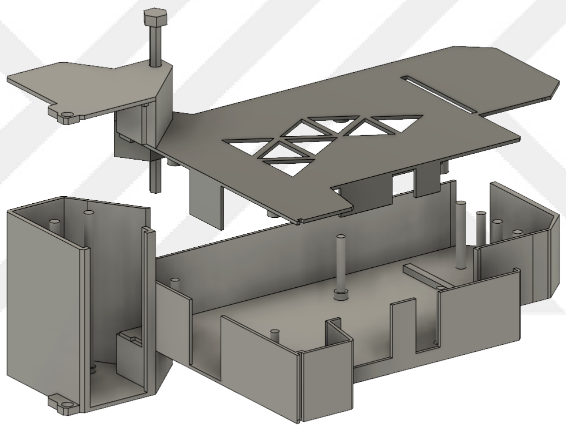


Figure 3.21. L.A.N.C.E.D MK I case before assembly

Design is made with Autodesk Fusion 360. The software is easy to use with a reasonable learning curve, desired functions are provided just as any other design software. Design files are allowed to be stored on Fusion 360 cloud with versions for each iteration on the design. Support for the software is also exist on the official site with interaction of other users as well, a dynamic question and answer flow is ensured which is great for beginners. Materials for 3D printing is also contained within the program.

Earlier versions of LANCED MKI design is printed with using PLA (Poly Lactic Acid) (see Figure 3.22). PLA is a commercialized material with wide range of regional availability. PLA is one the most popular material with bio degradable property. Petrochemical based polymers are widely used and pollution caused by traditional polymers is a critical issue for environment. PLA can be a substitute as a novel type of polymer. Main advantages that PLA includes Ecologic compatibility, bio-compatibility, easy processability and energy efficient in terms of production (Farah et al., 2016).

Final version of the MKI casing is produced with PLA+, consequently a better result is achieved. PLA+ is more robust compared to PLA, rattle between parts are eliminated consequently. Connection of parts are improved with PLA+ as well, desired gap between parts is met while keeping the distance for easy detachment.

The LANCED system MK I is comprised of 6 pieces of interconnecting parts (see Figure 3.21). USB ports are extended via provided extension cables with male and female USB connectors on each side, allowing access for the user to connect. Rerouting of the USB connection ports allow the external modules to connect adjacently (see Figure 3.23). Main casing fits the RPI board perfectly while total sizing is not greater than a standard human hand ensuring easy handling if desired (see Figure 3.22).

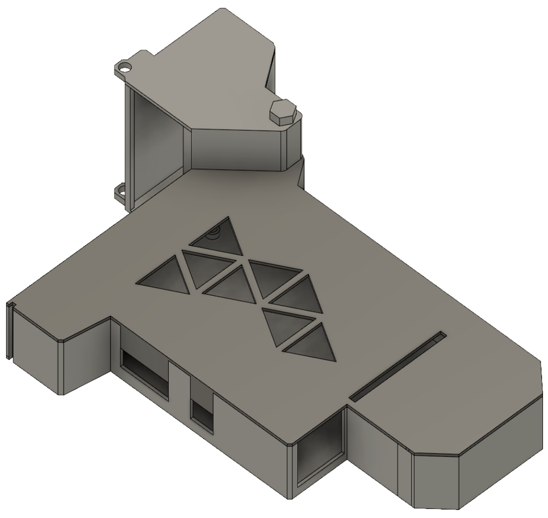


Figure 3.22. L.A.N.C.E.D MK I assembled

Hexagonal connection rods are in use for attaching the upper casing to main casing where RPI board resides. Extension modules are also connected to the upper case via an hexagonal connection rod which can be dismantled for new module attachments, delivering easy customization for the user (see Figure 3.21).

Interlocking parts are locked with using connection pins stored internally, pins are locking RPi board while locking USB extensions, male and female counterparts. Female USB extensions are residing within the upper casing (external module housing) while male counterpart is residing next to Ethernet port of the RPI board (see Figure 3.21). The external Wi-Fi NIC card nesting is designed fit into USB ports with stabilizing rods to ensure stability (see Figure 3.17).

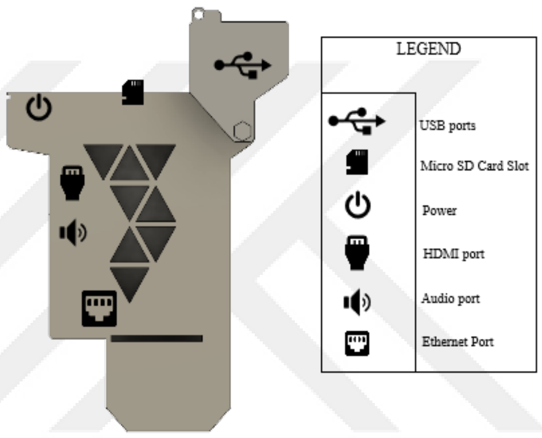
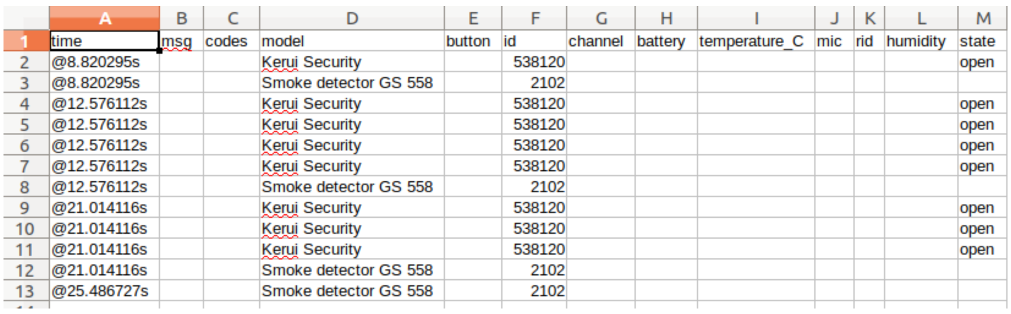


Figure 3.23. L.A.N.C.E.D MKI Interfacing

CHAPTER 4
CONCLUSIONS AND FUTURE RESEARCH

Our prototype LANCED is a multi-purpose smart control box. Its main body is based on Raspberry Pi 3 microcomputer. It includes some additional components required by different functionalities like G-STAR IV GPS module with USB connection, Software Defined Radio module with USB connection, RTL-SDR, Raspberry Pi GSM/GPRS Shield from Sixfab, Wi-Fi Network Interface Card, TP-LINK TL-WN722N. They bring together a functioning prototype which has operational capabilities for different emergency scenarios. They enable auxiliary network connectivity, can detect the IoT devices in use, and collect their data to discover the ownership of different devices in the disaster area.

LANCED has a user-friendly GUI and helps user interaction for managing alternative controls in emergency situations. LANCED operation helps users to have an elevated view of the disaster site from the point of IoT and smart devices. The system utilizes open-source programs as well as essential hardware devices for operation by its generic and integrated design. Even though it is designed to detect Wi-Fi, Bluetooth and Sub-GHz devices, ZigBee and other smart devices working in different Industrial Communications protocols can be covered depending on the interfaces embedded into the system.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	time	msg	codes	model	button	id	channel	battery	temperature_C	mic	rid	humidity	state
2	@8.820295s			Kerui Security		538120							open
3	@8.820295s			Smoke detector GS 558		2102							
4	@12.576112s			Kerui Security		538120							open
5	@12.576112s			Kerui Security		538120							open
6	@12.576112s			Kerui Security		538120							open
7	@12.576112s			Kerui Security		538120							open
8	@12.576112s			Smoke detector GS 558		2102							
9	@21.014116s			Kerui Security		538120							open
10	@21.014116s			Kerui Security		538120							open
11	@21.014116s			Kerui Security		538120							open
12	@21.014116s			Smoke detector GS 558		2102							
13	@25.486727s			Smoke detector GS 558		2102							

Figure 4.1. SDR scanning data

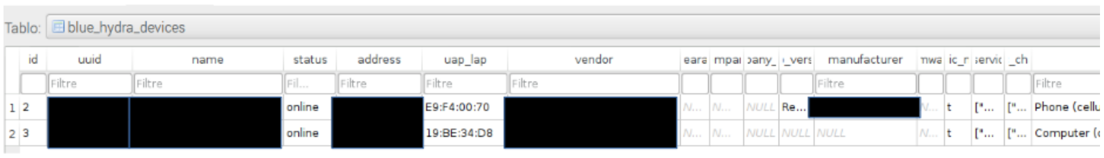
LANCED is designed to handle automation for detecting IoT or smart devices and collecting their ownership/usage information in emergency situations. It reduces user interaction and utilizes critical response times in disaster scenarios. The LANCED menu offers minimal design for the user while allowing operation and diagnostics for the hardware and software modules for troubleshooting. LANCED supports the users, rescue teams in order to have a monitoring capability for IoT devices of Wi-Fi, Bluetooth, and sub-GHz variations (see Figure 4.1)(see Figure 4.2)(see Figure 4.3). The prototype is easy to build with a total price of 260 United States Dollars (see Table 4.1).

Table 4.1. LANCED production cost

Product	Type	Unit	Total Price
Raspberry Pi 3 B+	Microcomputer	1	45 $
Globalsat BU-353 S4	GPS Receiver	1	35 $
TP-LINK TL-WN722N	WiFi NIC	2	86 $
RTL-SDR RTL2832U	SDR	1	30 $
Sixfab GSM/GPRS Shield	GPRS	1	43 $
SanDisk 16GB micro SD	Memory Card	1	6 $
GSM Subscription	SIM Card	1	5 $
USB Female Type-A	USB Connector	4	2 $
USB Male Type-A	USB Connector	4	2 $
GSM Antenna	uFL Antenna	1	5 $
ESUN 3D Printer Filament	PLA	65 grams	1.3 $

LANCED system is designed to achieve a fast response time while ensuring a low cost. Mobility is also delivered via minimal case design, also the system is modular as well, allowing the operator to mix and match between desired hardware modules. Total cost of a rescue operation in an emergency also includes training of the rescue personnel. LANCED system can be operated by anyone regardless of technical experience.

Latest versions Kismet program is able to store WiFi device data in a database, however stable version of Kismet (Kismet-2016-07-R1) does not support a standard data storage. This problem is handled by LANCED program, after each operation WiFi device data is sorted into Comma Seperated Value (CSV) form. Total disk size of a WiFi Network device generated by Kismet is around 1 kilobyte, eventually is reduced to 384 bytes after parsed into CSV form.

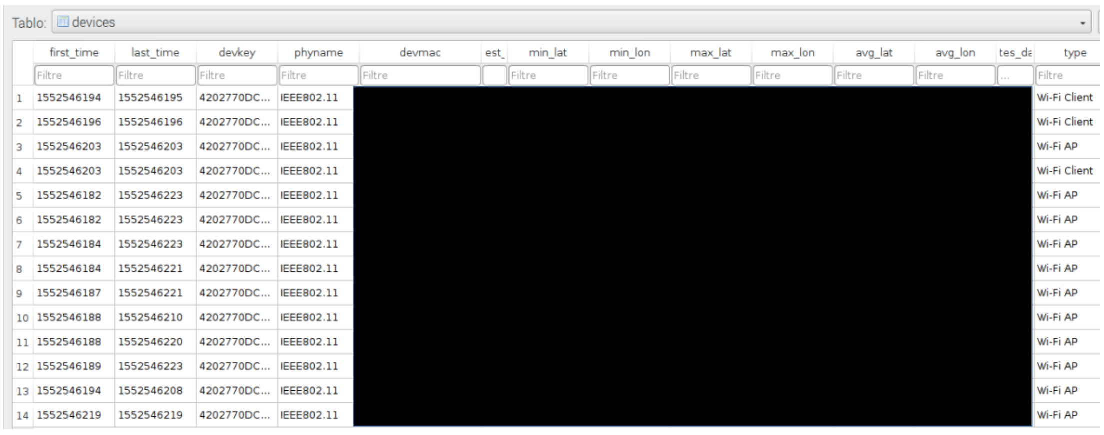

Tablo: blue_hydra_devices

	id	uuid	name	status	address	uap_lap	vendor	eara	mpai	pany_	_vers	manufacturer	nwa	ic_r	servic	_ch	
		Filtre	Filtre	Fil...	Filtre	Filtre	Filtre					Filtre					Filtre
1	2	[redacted]	[redacted]	online	[redacted]	E9:F4:00:70	[redacted]	N...	N...	NULL	Re...	[redacted]	N...	t	["...	["...	Phone (cellu
2	3	[redacted]	[redacted]	online	[redacted]	19:BE:34:D8	[redacted]	N...	N...	NULL	NULL	NULL	N...	t	["...	["...	Computer (c

Figure 4.2. Bluetooth scanning data

4.1. Limitations

Lanced system is capable of scanning WiFi, Bluetooth and Sub-Ghz devices, unfortunately the system is based on a Raspberry Pi microcomputer, therefore RAM and CPU are limited. Primary programs of scanning are not CPU exhaustive, on the contrary RAM usage is intensive. The total operation time is limited with maximum of 20 minutes assuming the system is set to monitor with all programs otherwise OS limits the user if the cache memory is overloaded, killing essential processes to prevent them from turning to a zombie process, consequently a system halt occurs. Avoiding an unexpected termination of essential programs, sessions are automatically stopped for data storage.


Tablo: devices

	first_time	last_time	devkey	phyname	devmac	est_	min_lat	min_lon	max_lat	max_lon	avg_lat	avg_lon	tes_da	type
	Filtre	Filtre	Filtre	Filtre	Filtre		Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	...	Filtre
1	1552546194	1552546195	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi Client
2	1552546196	1552546196	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi Client
3	1552546203	1552546203	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
4	1552546203	1552546203	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi Client
5	1552546182	1552546223	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
6	1552546182	1552546223	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
7	1552546184	1552546223	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
8	1552546184	1552546221	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
9	1552546187	1552546221	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
10	1552546188	1552546210	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
11	1552546188	1552546220	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
12	1552546189	1552546223	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
13	1552546194	1552546208	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP
14	1552546219	1552546219	4202770DC...	IEEE802.11	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	Wi-Fi AP

Figure 4.3. Kismet scanning data

Provided with power to LANCED device, the prototype is operational under 60 seconds, however a GPS acquisition may cause a bottleneck. Nearby structures like buildings or trees may be avoided to achieve a faster activation.

LANCED system is delivering Internet connection via 2G cellular connection with 114 Kbits per second for download and 20 Kbits per second for upload. As a result, LANCED system is limited by 2G cellular technology if the prototype is used as a backup AP. Considering a basic Internet connection is switched with LANCED AP, a dramatic data transfer is inevitable.

4.2. Future Work

LANCED system has drawbacks of Internet connection speed. This problem can be tackled via implementing latest cellular connection technologies. Cellular technologies like 4G is capable of delivering 100 Mbits per second communication rate. Unfortunately latest cellular technologies are expensive in hardware costs.

LANCED system is also affected via limited RAM. Successor of Raspberry Pi 3, Raspberry pi 4 model is now delivering 4 GB RAM with 1.5 GHz CPU. LANCED system also is possible to be deployed on any Linux based board with improved hardware specifications.

In addition to device discovery, exact geographical locations of the devices could be reported via signal triangulation if provided with at least three LANCED units with adequate calculation algorithm. Triangulation of any RF signal source is then possible via using any commercial SDR.

REFERENCES

Aly Farahat and Ali Ebnenasir. 2012. A Lightweight Method for Automated Design of Convergence in Network Protocols. *ACM Trans. Auton. Adapt. Syst.* 7, 4, Article 38 (December 2012), 36 pages. DOI=http://dx.doi.org/10.1145/2382570.2382574

Asensio, Á., Blanco, T., Blasco, R., Marco, Á., & Casas, R. (2015). Managing emergency situations in the smart city: The smart signal. Sensors, 15(6), 14370-14396.

Ashton, Kevin. (2009). That 'Internet of Things' Thing. RFiD Journal. 22. 97-114.

Ben Arbia, D., Alam, M. M., Kadri, A., Ben Hamida, E., & Attia, R. (2017). Enhanced IoT-Based End-To-End Emergency and Disaster Relief System. Journal of Sensor and Actuator Networks, 6(3), 19.

Bovet, D. P., & Cesati, M. (2005). *Understanding the Linux Kernel: from I/O ports to process management*. " O'Reilly Media, Inc.".

Cai, J., & Goodman, D. J. (1997). General packet radio service in GSM. IEEE Communications Magazine, 35(10), 122-131.

C. A. Mack, "Fifty Years of Moore's Law," in *IEEE Transactions on Semiconductor Manufacturing*, vol. 24, no. 2, pp. 202-207, May 2011.

Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

Farah, S., Anderson, D. G., & Langer, R. (2016). Physical and mechanical properties of PLA, and their functions in widespread applications—A comprehensive review. *Advanced drug delivery reviews*, *107*, 367-392.

GPSD (1995). from source: http://catb.org/gpsd/

HOSTAPD (2002). from source: https://w1.fi/hostapd/

Jondral, F. K. (2005). Software-defined radio: basics and evolution to cognitive radio. EURASIP journal on wireless communications and networking, 2005(3), 275-283.

KISMET (2002). from source : https://www.kismetwireless.net/

Kleinrock, L. (1961). Information Flow in Large Communication Nets, Ph.D. Thesis Proposal. Unpublished doctoral dissertation , Massachusetts Institute of Technology.

Kranenburg, R. V. (2008). The Internet of Things: A critique of ambient technology and the all-seeing network of RFID.

Lee, C. H., Orikumhi, I., Leow, C. Y., Malek, M. A. B., & Rahman, T. A. (2015, December). Implementation of relay-based emergency communication system on software defined radio. In Parallel and Distributed Systems (ICPADS), 2015 IEEE 21st International Conference on (pp. 787-791). IEEE.

Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., ... & Wolff, S. (2009). A brief history of the Internet. *ACM SIGCOMM Computer Communication Review*, *39*(5), 22-31.

Leonard Richardson and Sam Ruby. 2007. *Restful Web Services* (First ed.). O'Reilly.

Luchetti, G., Mancini, A., Sturari, M., Frontoni, E., & Zingaretti, P. (2017). Whistland: An augmented reality crowd-mapping system for civil protection and emergency management. ISPRS International Journal of Geo-Information, 6(2), 41.

Michalis Georgiou, Ilias Tachmazidis, and Grigoris Antoniou. 2019. Hypercat JSON-LD: A Semantically Enriched Catalogue Format for IoT. In *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics* (WIMS2019). ACM, New York, NY, USA, Article 13, 12 pages. DOI: https://doi.org/10.1145/3326467.3326477

MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 07 March 2019. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.

Qiu, Tie & Lv, Yuan & Xia, Feng & Chen, Ning & Wan, Jiafu & Tolba, Amr. (2016). ERGID: An Efficient Routing Protocol for Emergency Response Internet of Things. Journal of Network and Computer Applications. 72. 104-112. 10.1016/j.jnca.2016.06.009.

Parkinson, B. W., Enge, P., Axelrad, P., & Spilker Jr, J. J. (Eds.). (1996). *Global positioning system: Theory and applications, Volume II*. American Institute of Aeronautics and Astronautics.

Powers, S., Peek, J., O'reilly, T., Loukides, M., & Loukides, M. K. (2003). *UNIX power tools*. " O'Reilly Media, Inc.".

PPPD (2014).from source:https://www.freebsd.org/cgi/man.cgi?query=pppd&sektion=8&manpath=FreeBSD+4.7-RELEASE

Ravali, P. (2013). A comparative evaluation of OSI and TCP/IP models. *International Journal of Science and Research*, *4*(7), 514-521.

Roberts, L.G. (1967). Multiple computer networks and intercomputer communication.

RTL_433 (2012). from source : https://github.com/merbanan/rtl_433

Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, October 2004.

Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

Steve Vinoski. 2006. Advanced Message Queuing Protocol. *IEEE Internet Computing* 10, 6 (November 2006), 87-89. DOI=http://dx.doi.org/10.1109/MIC.2006.116

Sood, S. K., & Mahajan, I. (2017). Wearable IoT sensor based healthcare system for identifying and controlling chikungunya virus. Computers in Industry, 91, 33-44.

Uckelmann, D., Harrison, M., & Michahelles, F. (2011). An Architectural Approach Towards the Future Internet of Things. Architecting the Internet of Things.

V. Pehkonen and J. Koivisto, "Secure Universal Plug and Play network," *2010 Sixth International Conference on Information Assurance and Security*, Atlanta, GA, 2010, pp. 11-14. doi: 10.1109/ISIAS.2010.5604189

Zach Shelby and Carsten Bormann. 2010. *6LoWPAN: The Wireless Embedded Internet*. Wiley Publishing.

Zimmermann, H. (1980). OSI reference model--The ISO model of architecture for open systems interconnection. *IEEE Transactions on communications*, *28*(4), 425-432.

Xu, L., He, W., & Li, S. (2014). Internet of Things in Industries: A Survey. IEEE Transactions on Industrial Informatics, 10, 2233-2243.

APPENDIX 1 – LANCED

```
#####  INSTALLER  #####
### APPENDIX ###



#####  INSTALLER  #####
====================================================================
==================================
====================================================================
==================================
====================================================================
==================================

#!/bin/bash
    #title          :installersuidrootLANCED.sh
    #description    :installer tool for LANCED
    #author         :rektosauruz
    #date           :20181127
    #version        :v0.1
    #usage          :./installersuidrootLANCED.sh
    #notes          :lanced version update for thesis
    #bash_version   :4.4-5
    #=============================================

#File Declarations
#/home/pi/lanced_logs/                    [raw data files are saved
here from kismet_server.]
#/home/pi/lanced_arch/                    [files are transferred
after each run to this location to be processed.]
#/home/pi/lanced_arch/{date}/             [a dated folder is created
for that day.]
#/home/pi/lanced_arch/BSSID.list          [unique MACs are held here
for counting and comparison for uniqueness.]
#/home/pi/lanced_arch/datapool.txt        [datapool.txt holds the
unique data, populated after each run, a simple database file holds
raw data.]
#/home/pi/lanced_arch/temp.list           [for each run MACs in the
respective .nettxt file are passed to temp.list for comparison with
BSSID.list]
#/home/pi/lanced_arch_processed/          [processed files are saved
here under the same respective dates.]
#/home/pi/lanced_arch_processed/{date}/   [dated folders are
directly transferred under processed section after the sequence.]
#/etc/kismet/timechk             [timechk file is created after the
first date correction, at the end of each run, this file is
removed.]

##For both GUI and CLI, automation script
##will automatically install and configure lanced system to latest
version.
##For debugging, an echo can be return if a specific process is
failed.


#Color Declerations
ESC="#["
```

```
RESET=$ESC"39m"
RED=$ESC"31m"
GREEN=$ESC"32m"
LYELLOW=$ESC"36m"
YELLOW=$ESC"34m"
YELLOW=$ESC"33m"
      RB=$ESC"48;5;160m"
RESET1=$ESC"0m"

##check for root
if [[ "${EUID}" -ne 0 ]]; then
  echo -e "${GREEN}run${RESET} ${RED}installer.sh${RESET} $
{GREEN}with${RESET} ${RED}root !${RESET}"
  exit 1
fi

##fix locale
#if [ "`locale | tail -1`" == "LC_ALL=" ];then
#  export LC_ALL=C
#  sudo dpkg-reconfigure locales
#fi

####FIRST SECTION####
##initialization of required folders used by LANCED.sh
sudo mkdir /home/pi/lanced_logs
sudo mkdir /home/pi/lanced_arch
sudo mkdir /home/pi/lanced_arch/processed
sudo touch /home/pi/lanced_arch/BSSID.list
sudo touch /home/pi/lanced_arch/datapool.txt



#initialize v{1-20}
for i in `seq 1 23`;do
  c="`sed ""$i"q;d" /home/pi/LANCED/lanced_handler/dep.list`"
  eval "v${i}=${RED}[${c}]${RESET}"
done


###Loading bar indicates while download and initialization
progress###
ledger () {
clear
cat <<-ENDOFMESSAGE
${RB}                                       ${RESET1}
      ${RB} ${RESET1}$v21   $v22   $v23${RB} ${RESET1}
      ${RB} ${RESET1}$v1   $v2     $v3${RB} ${RESET1}
      ${RB} ${RESET1}$v4 $v16${RB} ${RESET1}
      ${RB} ${RESET1}$v6  $v7   $v8${RB} ${RESET1}
      ${RB} ${RESET1}$v9  $v10  $v13${RB} ${RESET1}
      ${RB} ${RESET1}$v11$v12$v14  $v20${RB} ${RESET1}
      ${RB} ${RESET1}$v15    $v5${RB} ${RESET1}
      ${RB} ${RESET1}$v17   $v18  $v19${RB} ${RESET1}
${RB}                                       ${RESET1}
ENDOFMESSAGE

}

ledger

###using the dep.list, this iteration handles the apt-get
```

```
update/upgrade/dist-upgrade
for i in `seq 21 23`; do
      a="`sed ""$i"q;d" /home/pi/LANCED/lanced_handler/dep.list`"
      sed ""$i"q;d" /home/pi/LANCED/lanced_handler/dep.list | xargs
apt-get -y > /dev/null 2>&1 && eval "v${i}=${GREEN}[${a}]${RESET}"
||
      echo -e "$a" >> /home/pi/LANCED/lanced_handler/error.list
      ledger
done

###using the dep.list this iteration hadnles the apt-get install
echo -e "update - upgrade - dist-upgrade DONE" >>
/home/pi/LANCED/lanced_handler/progress.list

for i in `seq 1 19`; do
      a="`sed ""$i"q;d" /home/pi/LANCED/lanced_handler/dep.list`"
      sed ""$i"q;d" /home/pi/LANCED/lanced_handler/dep.list | xargs
apt-get install -y > /dev/null 2>&1 && eval "v${i}=${GREEN}[${a}]$
{RESET}" ||
      echo -e "$a" >> /home/pi/LANCED/lanced_handler/error.list
  ledger
done

#echo -e "essential files DONE" >>
/home/pi/LANCED/lanced_handler/progress.list

#sudo apt-get install locate
#sudo apt-get install libpcre3 libpcre3-dev
sudo wget https://www.kismetwireless.net/code/kismet-2016-07-
R1.tar.xz > /dev/null 2>&1
sudo tar -xf kismet-2016-07-R1.tar.xz
cd kismet-2016-07-R1/

sudo ./configure > /dev/null 2>&1

sudo make dep > /dev/null 2>&1

sudo make > /dev/null 2>&1

sudo make suidinstall > /dev/null 2>&1

sudo usermod -a -G kismet pi

v20=${GREEN}[kismet]${RESET}
ledger



#echo -e "kismet DONE" >>
/home/pi/LANCED/lanced_handler/progress.list

####SECOND SECTION####

##git folder
#/home/pi/LANCED/lanced_handler/
#/etc/kismet/
#/etc/kismet/kismet.conf
#/etc/kismet/oui2.txt
#/etc/kismet/correct_date.py
```

```
##/etc/kismet/ folder contained file operations are handled here
#sudo mv /usr/local/etc/kismet_drone.conf
/usr/local/etc/kismet_drone.conf.orig
sudo rm /usr/local/etc/kismet.conf
sudo cp
/home/pi/LANCED/config_files/suidins/{kismet.conf,oui2.txt,correct_d
ate.py} /usr/local/etc/
#sudo chmod 777 /etc/kismet/kismet.conf
#sudo chmod 777 /etc/kismet/oui2.txt
sudo chmod 777 /usr/local/etc/correct_date.py


      ##get mac address of wlan0 then pass it for kismet.conf for
      filtering the AP
##filter_tracker=BSSID(!--:--:--:--:--:--) is the defined format
holder="`sudo ip -o link | awk '{print $2,$(NF-2)}' | grep wlan0 |
cut -d' ' -f2`"
sed -i "198s/.*/filter_tracker=BSSID(!$holder)/"
/usr/local/etc/kismet.conf

#echo -e "/etc/kismet/ files done" >>
/home/pi/LANCED/lanced_handler/progress.list


##/etc/default/gpsd
##/etc/default/hostapd
##/etc/default/ folder file operations are handled in this part
sudo mv /etc/default/gpsd /etc/default/gpsd.origin
sudo mv /etc/default/hostapd /etc/default/hostapd.origin
sudo cp /home/pi/LANCED/config_files/{gpsd,hostapd} /etc/default/
#sudo chmod 777 /etc/default/gpsd
#sudo chmod 777 /etc/default/hostapd

#echo -e "gpsd - hostapd files done" >>
/home/pi/LANCED/lanced_handler/progress.list

#/etc/network/interfaces
sudo mv /etc/network/interfaces /etc/network/interfaces.origin
sudo cp /home/pi/LANCED/config_files/interfaces /etc/network/

#echo -e "interfaces done" >>
/home/pi/LANCED/lanced_handler/progress.list


####/etc/hostapd/###
#/etc/hostapd/hostapd.conf
if [ -e /etc/hostapd/hostapd.conf ]; then
  sudo mv /etc/hostapd/hostapd.conf /etc/hostapd/hostapd.conf.origin
fi
sudo cp /home/pi/LANCED/config_files/hostapd.conf /etc/hostapd/

#echo -e "hostapd.conf done" >>
/home/pi/LANCED/lanced_handler/progress.list

#allow ssh
#change the default port if needed
####/etc/ssh/###
#/etc/ssh/sshd_config

#allow ssh
#change timezone
```

```
#sudo raspi-config

#echo -e "raspi-config done" >>
/home/pi/LANCED/lanced_handler/progress.list

####/etc/####

#/etc/dnsmasq.conf file operations are handled in this part
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.origin
sudo cp /home/pi/LANCED/config_files/dnsmasq.conf /etc/


#/etc/dhcpcd.conf file operations are handled in this part
sudo mv /etc/dhcpcd.conf /etc/dhcpcd.conf.origin
sudo cp /home/pi/LANCED/config_files/dhcpcd.conf /etc/


      #/etc/sysctl.conf file operations are handled in this part
      sudo mv /etc/sysctl.conf /etc/sysctl.conf.origin
      sudo cp /home/pi/LANCED/config_files/sysctl.conf /etc/


#ipv4 port forward section
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"

#/etc/rc.local
sudo mv /etc/rc.local /etc/rc.local.origin
sudo cp /home/pi/LANCED/config_files/rc.local /etc/

sudo systemctl disable dhcpcd.service > /dev/null 2>&1

##transfer the latest a.sh file to /home/pi by renaming the script
and set priviliges.
sudo mv /home/pi/LANCED/lanced_handler/asuidroot.sh
/home/pi/asuidroot.sh
sudo chmod 777 /home/pi/asuidroot.sh

sleep 5
echo "${RED}Shutting down!${RESET}"
sudo shutdown +0

#echo -e "/etc/ done" >>
/home/pi/LANCED/lanced_handler/progress.list

#echo -e "FINALIZED" >> /home/pi/LANCED/lanced_handler/progress.list

===


====================================================================
==================================
====================================================================
==================================
====================================================================
==================================
###### LANCED ######
====================================================================
==================================
====================================================================
==================================
====================================================================
==================================
#!/bin/bash
    #title          :a.sh
    #description    :Providing powerful yet simle UI for both cli an
desktop, this script handles automation  via bash scripting.
    #author         :rektosauruz
    #date           :20181005
    #version        :v2.5
    #usage          :#pi./a.sh
    #notes          :Single module usage is implemented for v2.0.
Cell phone gps usage is implemented in addition to pl2303 gps usage.
    #bash_version   :4.4-5
    ===============================



      #File Declarations
      #/home/pi/lanced_logs/                    [raw data files are
      saved here from kismet_server.]
#/home/pi/lanced_arch/                    [files are transferred
after each run to this location to be processed.]
#/home/pi/lanced_arch/{date}/             [a dated folder is created
for that day.]
#/home/pi/lanced_arch/BSSID.list          [unique MACs are held here
for counting and comparison for uniqueness.]
#/home/pi/lanced_arch/datapool.txt        [datapool.txt holds the
unique data, populated after each run, a simple database file holds
raw data.]
#/home/pi/lanced_arch/temp.list           [for each run MACs in the
respective .nettxt file are passed to temp.list for comparison with
BSSID.list]
#/home/pi/lanced_arch_processed/          [processed files are saved
here under the same respective dates.]
#/home/pi/lanced_arch_processed/{date}/   [dated folders are
directly transferred under prcessed section after the sequence.]
#/home/pi/etc/kismet/timechk             [timechk file is created
after the first date correction, at the end of each run, this file
is removed.]

# colors
ESC="#["
RESET=$ESC"39m"
RED=$ESC"31m"
GREEN=$ESC"32m"
LYELLOW=$ESC"36m"
YELLOW=$ESC"34m"
YELLOW=$ESC"33m"
RB=$ESC"48;5;160m"
RESET1=$ESC"0m"
RESETU=$ESC"24m"
GB=$ESC"48;5;40m"
```

```
wst1="${RB}    ${RESET1}"
wst2="${RB}    ${RESET1}"
gpsm="${RB}    ${RESET1}"
gpfx="${RB}    ${RESET1}"
dat0="${RB}    ${RESET1}"

wst11="${RED}=====${RESET}"
wst22="${RED}===========${RESET}"
gpsmm="${RED}=====${RESET}"
gpfxx="${RED}===========${RESET}"
dat00="${RED}=====${RESET}"


ledger () {

cat <<-ENDOFMESSAGE
 ${RED}_____      ______${RESET}
${RED}/${RESET}$wst1${RED}|${RESET}$wst11${RED}/ W1 __\ ${RESET}
${RED}\_____\    \___|   _____${RESET}
${RED}/${RESET}$wst2 ${RED}|${RESET}$wst22${RED}/ W2 /${RESET}
${RED}\_____\      _____\___|${RESET}
${RED}/${RESET}$gpsm ${RED}|${RESET}$gpsmm${RED}/ GPS /${RESET}
${RED}\_____\     \____| _____${RESET}
${RED}/${RESET}$gpfx ${RED}|${RESET}$gpfxx${RED}/Gfix/${RESET}
${RED}\_____\      _____\____| ${RESET}
${RED}/${RESET}$dat0 ${RED}|${RESET}$dat00${RED}/ DATE/${RESET}
      ${RED}\____/      \____/${RESET}
      ENDOFMESSAGE

      }

if [ -n "`sudo pidof gpsd`" ]; then
sudo pkill gpsd
fi


##use these later
#FOR THE GREEN BACKGORUND BLOCK
#echo -e "\e[48;5;40m \e[0m"
#above line prints a single block of green on tty

#init var(1-4)
for i in `seq 1 4`; do
  eval "var${i}=${RED}X${RESET}"
done

var5="${RED}NOT READY${RESET}"
var6="${RED}X${RESET}"
timeloc=/usr/local/etc/timechk
kiss_state="`sudo pidof kismet_server`"
chk1="${RED}[Wlan1]${RESET}"
chk2="${RED}[Wlan2]${RESET}"
chk3="${RED}[GPS]${RESET}"
chk4="${RED}[GPSfix]${RESET}"
chk5="${RED}[Date]${RESET}"
reset

##check for the kismet server if the a.sh is armed then closed,
after a rerun state is fixed to armed.
```

```
if [ -n "$kiss_state" ]; then
      var5="${RED}ARMED${RESET}"
fi

#refresh() {
#if [ -n "`ls -A /home/pi/lanced_logs/`" ]; then
#   tempcalc="`ls /home/pi/lanced_logs/ | grep .nettxt`"
#   kk="${GREEN}`cat /home/pi/lanced_logs/"$tempcalc" | grep Network
| uniq | wc -l`${RESET}"
#else
#   kk="${GREEN}0${RESET}"
#fi
#}




count() {
kk="${RED}`wc -l /home/pi/lanced_arch/BSSID.list | cut -d' ' -f1`$
{RESET}"
}

count




##check if hostapd is active or not, this option is for monitored
runs and while no hostapd is needed.
##also ip address is printed in the LANCED GUI for easy usage.
apdip() {

if [ -z "`pidof hostapd`" ]; then
      var7="${RED}X${RESET}"
      var8="${RED}IP${RESET}    ${RED}>${RESET} ${RED}X${RESET}"
else
      var7="${GREEN}OK${RESET}"
      var8="${GREEN}`ifconfig wlan0 | grep "inet " | cut -d't' -f2 |
cut -d'n' -f1 | xargs`${RESET}"
fi

}

apdip


##this is data sorter function for the LANCED. After every use, data
is collected under lanced_logs is first transferred to lanced_arch
d_sorter() {

##declare
datum="`date +%Y%m%d`"
locA=/home/pi/lanced_logs/
locB=/home/pi/lanced_arch/
locC=/home/pi/lanced_arch/processed/

#check for dated folder /home/pi/lanced_arch/ . Multiple runs in the
same day are collected under the same folder such as 20180513
if [ ! -d "$locB""$datum" ];then
```

```
    sudo mkdir "$locB""$datum"
    sudo chmod 777 "$locB""$datum"/
fi

#check for dated folder /home/pi/lanced_arch/processed/
if [ ! -d "$locC""$datum" ];then
    sudo mkdir "$locC""$datum"
    sudo chmod 777 "$locC""$datum"/
fi

##create data count list for different dates
sudo touch "$locB"datac.list
sudo chmod 777 "$locB"datac.list

##check for different dated files then pass it to datac.list
echo "`ls "$locA" | grep "$datum"`" >> "$locB"datac.list

##pass the dated files from lanced_logs to respective dated folder
for i in $(cat /home/pi/lanced_arch/datac.list);do
    #copy option
   sudo cp "$locA"$i "$locB""$datum"
done
sudo rm -r "$locA"*


##clear the datac.list
      sudo rm "$locB"datac.list


      #check for datapool file, create at /home/pi/lanced_arch/ if
      needed
if [ ! -f "$locB""$datum"/datapool.txt ];then
    sudo touch "$locB"datapool.txt
    sudo chmod 777 "$locB"datapool.txt
fi


####check for temporary list, this list is used to compare with
BSSID.list to keep track of unique MACs for the run.
if [ ! -f "$locB""$datum"/templ.list ];then
   sudo touch /home/pi/lanced_arch/templ.list
   sudo chmod 777 /home/pi/lanced_arch/templ.list
fi

##this line gets the exact name of the .nettxt file then picks the
MACs then pushed it to temprary list named templ.list
filenom="`ls "$locB""$datum" | grep ".nettxt"`"
echo -e "`grep "Network " "$locB""$datum"/"$filenom" | cut -d' ' -
f4`" >> "$locB"templ.list

##comparison is made in this for loop. For every MAC address located
in templ.list, BSSID.list is checked, if no match then the MAC is
unique.
##unique MACs then passed to datapool.txt file with certain
lines(line 125). Also BSSID list is populated with new unique MACs.
for i in $(cat /home/pi/lanced_arch/templ.list);do
      if [ "`grep "$i" "$locB"BSSID.list`" == "" ];then
            test1="`grep "Network " "$locB""$datum"/"$filenom" |
grep "$i"`"
                sed -n "/$test1/,/Network /{/$test1/{p};/Network
/{d};p}" "$locB""$datum"/"$filenom" >> "$locB"datapool.txt
```

```
            echo -e "$i" >> /home/pi/lanced_arch/BSSID.list

    fi
done

##temporary list is cleared.
sudo rm /home/pi/lanced_arch/templ.list

##moving the processed dated folder under lanced_arch to processed
folder under lanced_arch
sudo mv -v /home/pi/lanced_arch/"$datum"/*
/home/pi/lanced_arch/processed/"$datum"/ 1>/dev/null

count

}


##device probe function
components_chk() {
clear
ledger
if [ "`iwconfig 2>&1 | sed -n -e 's/wlan1     //p'| cut --bytes=1`"
== "I" ]; then
#if [ "`iwconfig wlan1 | cut -d' ' -f1 | xargs`" == "wlan1" ]; then
#if [ "`iwconfig wlan1: | cut -d':' -f1 | cut -d' ' -f1 | xargs`" ==
"wlan1" ]; then
#     chk1="${GREEN}[Wlan1]${RESET}"
#     echo -ne "$chk1$chk2$chk3$chk4$chk5\r"
  wst11="${GREEN}=====${RESET}"
  clear
  ledger
  sleep 1
  wst1="${GB}    ${RESET1}"
        clear
        ledger
        var1="${GREEN}OK${RESET}"
elif [ "`iwconfig 2>&1 | sed -n -e 's/wlan1     //p'| cut --
bytes=1`" != "I" ];then
#     echo -ne "$chk1$chk2$chk3$chk4$chk5\r"
      var1="${RED}X${RESET}"

fi

if [ "`iwconfig 2>&1 | sed -n -e 's/wlan2     //p' | cut --bytes=1`"
== "I" ]; then
#if [ "`iwconfig wlan2 | cut -d' ' -f1 | xargs`" == "wlan2" ]; then
#if [ "`iwconfig wlan2: | cut -d':' -f1 | cut -d' ' -f1 | xargs`" ==
"wlan2" ]; then
#     chk2="${GREEN}[Wlan2]${RESET}"
#     echo -ne "$chk1$chk2$chk3$chk4$chk5\r"
  clear
  ledger
  wst22="${GREEN}===========${RESET}"
  sleep 1
  wst2="${GB}    ${RESET1}"
  clear
  ledger
      var2="${GREEN}OK${RESET}"
elif [ "`iwconfig 2>&1 | sed -n -e 's/wlan2     //p'| cut --
bytes=1`" != "I" ];then
```

```
#     echo -ne "$chk1$chk2$chk3$chk4$chk5\r"
      var2="${RED}X${RESET}"
fi

if [ "`dmesg | grep "pl2303 converter now attached to ttyUSB0" | cut
-d':' -f2 | xargs`" == "pl2303 converter now attached to ttyUSB0" ];
then
#     chk3="${GREEN}[GPS]${RESET}"
#     echo -ne "$chk1$chk2$chk3$chk4$chk5\r"
  clear
  ledger
  gpsmm="${GREEN}=====${RESET}"
  gpfxx="${GREEN}===========${RESET}"
  dat00="${GREEN}===========${RESET}"
  clear
  ledger
  sleep 1
  clear
  ledger
  gpsm="${GB}    ${RESET1}"
      var3="${GREEN}OK${RESET}"

else
#     echo -ne "$chk1$chk2$chk3$chk4$chk5\r"
      var3="${RED}X${RESET}"
#     gpsd -b -n tcp://172.24.1.150:50000
#     return 1
fi

testv=`ping -c 1 -w 1 172.24.1.150 | grep ttl`
if [ -n "$testv" ];
    then
    clear
    ledger
    gpsmm="${GREEN}=====${RESET}"
    clear
    ledger
    sleep 1
    gpsm="${GB}    ${RESET1}"
    clear
    ledger
    sleep 1
          gpfxx="${GREEN}===========${RESET}"
    clear
    ledger
    sleep 1
    dat00="${GREEN}=====${RESET}"
    clear
    ledger
    sleep 1
    gpsd -b -n tcp://172.24.1.150:50000
    var3="${GREEN}OK${RESET}"
else
    return 1
fi

if [ "`timeout 6 gpspipe -w -n 5 | cut -d',' -f3 | grep mode | cut -
d':' -f2`" != "3" ]; then
    return 1
fi
```

```
while true ; do
   if [ "`gpspipe -w -n 5 | cut -d',' -f3 | grep mode | cut -d':' -
f2`" == "3" ]; then
 #     chk3="${GREEN}[GPS]${RESET}"
 #     chk4="${GREEN}[GPSfix]${RESET}"
 #     echo -ne "$chk1$chk2$chk3$chk4$chk5\r"
#      gpsm="${GB}    ${RESET1}"
      var3="${GREEN}OK${RESET}"
      var4="${GREEN}OK${RESET}"
      gpfx="${GB}    ${RESET1}"
      clear
      ledger
      sleep 1
      if [ ! -f "$timeloc"  ]; then
           #correcttime
           correcttime > /dev/null 2>&1
       #  chk5="${GREEN}[Date]${RESET}"
   #        echo -ne "$chk1$chk2$chk3$chk4$chk5\r"

           dat0="${GB}    ${RESET1}"
           clear
           ledger
           sleep 1

           var5="${GREEN}READY${RESET}"
         sudo touch "$timeloc"
        else
       #  chk5="${GREEN}[Date]${RESET}"
   #        echo -ne "$chk1$chk2$chk3$chk4$chk5\r"
           dat0="${GB}    ${RESET1}"
           clear
           ledger
           sleep 1
           var5="${GREEN}READY${RESET}"
        fi
      break
   else
      chk4="${RED}[GPSfix]${RESET}"
   #   echo -ne "$chk1$chk2$chk3$chk4$chk5\r"
   fi
done

##check WLAN1 WLAN2 GPS GPSfix
if [ "$var1" == "${GREEN}OK${RESET}" ] && [ "$var2" == "${GREEN}OK$
{RESET}" ] && [ "$var3" == "${GREEN}OK${RESET}" ] && [ "$var4" == "$
{GREEN}OK${RESET}" ]; then
      var5="${GREEN}READY${RESET}"
elif [ "$var1" == "${GREEN}OK${RESET}" ] && [ "$var2" == "${RED}X$
{RESET}" ] && [ "$var3" == "${GREEN}OK${RESET}" ] && [ "$var4" == "$
{GREEN}OK${RESET}" ]; then
      var5="${GREEN}READY${RESET}"

else
       var5="${RED}MISSING MODULE(s)${RESET}"
       echo -e "\n${RED}reprobe needed${RESET}"
       sleep 1
       return 1
fi
if [ -z "`pidof hostapd`" ]; then
      var7="${RED}X${RESET}"
      var8="${RED}IP${RESET}    ${RED}>${RESET} ${RED}X${RESET}"
```

```
else
      var7="${GREEN}OK${RESET}"
      #ifconfig wlan0 | grep "inet " | cut -d't' -f2 | cut -d'n' -f1
| xargs
      var8="${GREEN}`ifconfig wlan0 | grep "inet " | cut -d't' -f2 |
cut -d'n' -f1 | xargs`${RESET}"
fi



}

#clock () {

#while [1];do ledger;printf "\33[A";sleep 1;done

#}
#var5="${RED}READY${RESET}"
##LANCED menu
while :
do
    #clear
    reset
    cat<<EOF
`echo -e "${RED}  _                  _   _   ____   _____  ____ $
{RESET}"`
`echo -e "${RED} | |         /\      | \ | | / ___| |___ / |  _ \ $
{RESET}"`
`echo -e "${RED} | |        /  \     |  \| || |       |_ \ | | | | $
{RESET}"`
`echo -e "${RED} | |___ _  / /\ \  _| |\  || |___ _ ___) || |_| |$
{RESET}"`
`echo -e "${RED} |_____(${RESET}${RB} ${RESET1}${RED})/______\($
{RESET}${RB} ${RESET1}${RED})_| \_(${RESET}${RB} ${RESET1}$
{RED})____(${RESET}${RB} ${RESET1}${RED})____(${RESET}${RB} $
{RESET1}${RED})____(${RESET}${RB} ${RESET1}${RED})${RESET}"`
 ${RED} _____      _____ ${RESET}   ${RED}__________________${RESET}
${RED}//     \\${RESET}${RED}MENU${RESET}${RED}/     \\\\${RESET}  $
{RED}\\${RESET}${RB}                ${RESET1}${RED}/${RESET}
${RB} ${RESET1} ${RED}Device Probe${RESET} ${RED}[1]${RESET} ${RB} $
{RESET1}   ${RED}\\${RESET}${RB}              ${RESET1}${RED}/$
{RESET}
${RB} ${RESET1} ${RED}Quick Start${RESET}  ${RED}[2]${RESET} ${RB} $
{RESET1}    ${RED}\\${RESET}  `printf "%-20s\n" "$var5"`${RED}/$
{RESET}
${RB} ${RESET1} ${RED}ARM${RESET}          ${RED}[3]${RESET} ${RB} $
{RESET1}     ${RED}\\${RESET}${RB}          ${RESET1}${RED}/${RESET}
${RB} ${RESET1} ${RED}DISARM${RESET}       ${RED}[4]${RESET} ${RB} $
{RESET1}      ${RED}\\${RESET}${RB}        ${RESET1}${RED}/${RESET}
${RB} ${RESET1} ${RED}Quit${RESET}         ${RED}[Q]${RESET} ${RB} $
{RESET1}       ${RED}\\${RESET}${RB}      ${RESET1}${RED}/${RESET}
${RED}\\${RESET}${RED}\\${RESET}${RED}________________/${RESET}$
{RED}/${RESET}        ${RED}\\${RESET}${RB}    ${RESET1}${RED}/$
{RESET}
      ${RED}Total APs >${RESET} `printf "%-20s\n" "$kk"`       $
      {RED}\\${RESET}${RB}  ${RESET1}${RED}/${RESET}
`printf "%-31s\n" "$var8"`         ${RED}\\/${RESET}
`ledger`
```

```
EOF

##while [ 1 ];do date;printf "\33[A";sleep 1;done

##start function uses kismet_server
start_ks() {
#refresh
#create a file named by YearMonthDay
#sudo mkdir /home/pi/lanced_logs/"`date +%Y%m%d`"
if [ "$var5" == "${GREEN}READY${RESET}" ];then
/usr/local/bin/kismet_server --daemonize > /dev/null 2>&1
var5="${RED}ARMED${RESET}"
sleep 2
else
      echo "${RED}!!!probe the devices!!!${RESET}"
      return 1
fi

}


##stop function kills kismet_server then after 3 seconds sorts the
data using d_sorter function
stop_ks() {

sudo killall kismet_server
sleep 3
if [ -z "$(ls -A /home/pi/lanced_logs)" ]; then
      return 1
else
    d_sorter
    sleep 5
fi
if [ -n "`pidof gpsd`" ]; then
        sudo pkill gpsd
fi
var5="${GREEN}UNARMED${RESET}"

}


##start function for selection 2. first the components are probed,
then the kismet_server is run
q_start() {
components_chk
start_ks
return 1
}


    read -n1 -s
    case "$REPLY" in
    "1")  components_chk ;;
    "2")  q_start  ;;
    "3")  start_ks ;;
    "4")  stop_ks  ;;
#     "r")  refresh ;;
    "Q")  sudo rm "$timeloc" 2>/dev/null
```

```
        reset
          exit
                  ;;
    "q")  sudo rm "$timeloc" 2>/dev/null
        reset
        exit
              ;;
     * )  echo "invalid option"  ;;
    esac
    sleep 1
done

====================================================================
==================================
====================================================================
==================================
====================================================================
==================================



###SETTING blue_hydra ###
====================================================================
==================================
====================================================================
==================================
====================================================================
==================================

sudo git clone https://github.com/pwnieexpress/blue_hydra
echo "ok!"

sudo apt-get install bluez -y
sudo apt-get install bluez-test-scripts -y
sudo apt-get install pyhton-bluez -y
sudo apt-get install python-dbus -y
sudo apt-get install sqlite3 -y
sudo apt-get install libsqlite3-dev -y
sudo apt-get install ruby-dev bundler -y

echo "ok!"


cd blue_hydra/
sudo bundler
sudo bundle install

====================================================================
==================================
====================================================================
==================================
====================================================================
==================================
##SETTING ppp0 AS BRIDGED CONNECTION###
====================================================================
==================================

sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"

#setting ipv4 forward option to 1 in here
sudo nano /etc/sysctl.conf

#iptables rulez
sudo iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
sudo iptables -A FORWARD -i ppp0 -o wlan0 -m state --state
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o ppp0 -j ACCEPT

##to save iptables rules we set to run for each reboot
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"

##change rc.local file
sudo nano /etc/rc.local
above 0 add this line
iptables-restore < /etc/iptables.ipv4.nat


sudo service hostapd start
sudo service dnsmasq start

====================================================================
```

```
###GQRX SETUP for RPi3###
====================================================================
==================================

 sudo wget
https://github.com/csete/gqrx/releases/download/v2.11.5/gqrx-sdr-
2.11.5-linux-rpi3.tar.xz
sudo tar -xf gqrx-sdr-2.11.5-linux-rpi3.tar.xz
sudo apt install gnuradio libvolk1-bin libusb-1.0-0 gr-iqbal
sudo apt install qt5-default libqt5svg5 libportaudio2
cd gqrx-sdr-2.11.5-linux-rpi3/
sudo cp udev/*.rules /etc/udev/rules.d/
##done here run gqrx for testing
./gqrx

====================================================================
#!/bin/bash
#author : rektosauruz
#version : v3.2
#definition : G.I.T.S. - Ghost In The Shell - A shell framework
##ssh manager included

### | Declerations          |
====================================================#
# Color Declerations
ESC="#["
RESET=$ESC"39m"
RED=$ESC"31m"
GREEN=$ESC"32m"
LBLUE=$ESC"36m"
BLUE=$ESC"34m"
BLACK=$ESC"30m"
YELLOW=$ESC"33m"

#[*] Status Indicator with different colors.
RLS=${RED}"[*]"${RESET}
BLS=${BLUE}"[*]"${RESET}
GLS=${GREEN}"[*]"${RESET}
RES=${RED}"[!]"${RESET}



##################!  MENU  !##################
while :
do
    clear
    cat<<EOF
    `echo -e "${BLUE}==========================${RESET}$
{RED}=========================${RESET}$
{YELLOW}======================${RESET}"`
    `echo -e "       ${BLUE}_               _${RESET}    ${RED}_
_   _                ${YELLOW}_          _ _${RESET} "`
    `echo -e "  ${BLUE}__  | |__   ___  ___| |_${RESET} ${RED}(_)_
__ | |_| |__   ___   ${YELLOW}___| |__   ___| | |${RESET}"`
    `echo -e " ${BLUE}/ _\_| '_ \ / _ \/ __| __${RESET} ${RED}| | '_
\| __| '_ \ / _ \ ${YELLOW}/ __| '_ \ / _ \ | |${RESET}"`
    `echo -e "${BLUE}| (_| | | | | (_) \__ \ |_${RESET} ${RED}| | |
| | |_| | | |  __/ ${YELLOW}\__ \ | | |  __/ | |${RESET}"`
    `echo -e " ${BLUE}\__, |_| |_|\___/|___/\__${RESET} ${RED}|_|_|
|_|\__|_| |_|\___|${YELLOW} |___/_| |_|\___|_|_|${RESET}"`
    `echo -e " ${BLUE}|___/${RESET}

             "`
    `echo "${BLUE}==========================${RESET}$
{RED}=========================${RESET}$
{YELLOW}======================${RESET}"`
    $
{GREEN}=============================================================
============${RESET}
    `echo -e "${YELLOW}<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<${RESET}$
{RED}  ${BLUE}G.${RESET}${RED}I.T.${RESET}${YELLOW}S.${RESET}$
{YELLOW} >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>${RESET}"`
    `echo -e "${YELLOW}<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<${RED} MAIN
```

```
MENU${RESET} ${YELLOW}>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>${RESET}"`
    $
{GREEN}=====================================================================
============${RESET}
    |${GREEN} [001] Line_Calculator${RESET}  ||| ${GREEN}[007]
CryptoPaRseR${RESET}  ||| ${GREEN}[00n] NSlookup${RESET}  |
    |${GREEN} [002] IPtables_BLK${RESET}     ||| ${GREEN}[008]
Scan_ipv4${RESET}     ||| ${GREEN}[00v] Rec_V${RESET}     |
    |${GREEN} [003] NetCaT${RESET}           ||| ${GREEN}[009]
Conn_ChecK${RESET}    ||| ${GREEN}[00a] Rec_A${RESET}     |
    |${GREEN} [004] MD5${RESET}              ||| ${GREEN}[00l]
Ipv4_ChecK${RESET}    ||| ${GREEN}[00c] Rec_C${RESET}     |
    |${GREEN} [005] SHA-256 ${RESET}         ||| ${GREEN}[00w] Whois
${RESET}    ||| ${GREEN}[00s] SSH${RESET}       |
    |${GREEN} [006] TaR/UnTaR${RESET}        ||| ${GREEN}[00d]
DNS_L_Test${RESET}    ||| ${GREEN}[00p] Vpn${RESET}       |
  ${RED}(q)uit${RESET}$
{GREEN}=====================================================================
========${RESET}

EOF
##################!  MENU  !##################





md5() {

echo "please give the string to be converted to md5 format / Q to
quit"
read u_input
      if [ "$u_input" == "Q" ];
            then
            return 1
      else
            echo "md5 of $u_input : `echo -n "$u_input" | md5sum`"
            sleep 1
            echo "md5 of $u_input : `echo -n "$u_input" | md5sum`"
>> "$default_out"md5.txt
    fi

}



sha256() {

echo "please give the string to be converted to md5 format / Q to
quit"
read u_input
      if [ "$u_input" == "Q" ];
            then
            return 1
      else
            echo "sha256 of $u_input : `echo -n "$u_input" |
sha256sum`"
            sleep 1
            echo "sha256 of $u_input : `echo -n "$u_input" |
sha256sum`" >> "$default_out"sha256.txt
```

```
      fi
}



nc_manager() {

echo  "<SEND>     1"
echo  "<RECEIVE>  2"
echo  "<QUIT>     Q"
read user_c

case "$user_c" in
   "1") echo "ip / port / file"
        read ipaddr
        read port
        read file
        nc -nv $ipaddr $port < $file
   ;;
   "2") echo "port / file"
        read port
        read file
        nc -nvlp $port > $file
   ;;
   "Q") return 1 #exit 0
   ;;
   "q") return 1 #exit 0
   ;;
   *) echo "use defined values, ending the program now."
   ;;
esac

}



line_calc() {

#get the first parameter if not exist then ask for the directory
if [ -z "$1" ]; then
    echo "please give the path to directory / Enter "Q" to abort"
    read user_input
    path=$user_input
else
    path=$1
fi

if [ "$user_input" = "Q" ]; then
    #exit 0
    return 1
fi

#create the temporary file for listing the sub folders and files
touch /root/filenames.txt
chmod 755 /root/filenames.txt
temp=/root/filenames.txt
find $path | cat >> $temp ##/root/filenames.txt

#define a counter for the line count
inc=0
sum=0
```

```
total=0

#loop for the wc -l command to read from the temporary file
for ccb in $(cat $temp); do
    fl_count=$(wc -l $ccb 2> /dev/null | cut -d ' ' -f1)
    sum=$fl_count
    total=$(( sum + total + 1 ))
done

echo "total number of lines in $path = $total"
rm $temp
sleep 2
      }



iptables_blk() {

#first line is to get the address to be blocked.
echo "please input the IP address to be blocked / Enter "Q" to
abort"
read blk_addr
if [ "$blk_addr" = "Q" ]; then
    #exit 0
    return 1
fi

iptables -I INPUT -s $blk_addr -j DROP
iptables -I OUTPUT -s $blk_addr -j DROP
iptables -I FORWARD -s $blk_addr -j DROP
iptables-save > /etc/iptables.conf

}



tar_archiever() {

echo  "<PacK>     1"
echo  "<UnPacK>   2"
echo  "<QUIT>     Q"
read user_c

case "$user_c" in
   "1") echo "name the packed file"
        read file_p1
        echo "path/file_name"
        read file_p2
        tar -czvf /root/Desktop/"$file_p1".tar.gz --directory
"$file_p2" .

   ;;
   "2") echo "path/filename"
        read file_p3
        tar -xzvf "$file_p3" -C /root/Desktop
   ;;
   "Q") return 1 #exit 0
   ;;
   *) echo "use defined values, ending the program now."
   ;;
esac
```

```
}



function scan_last_two() {

for ipv4_4 in $(seq 1 255); do
      ping -c 1 192.168.1.$ipv4_4 | grep "ttl=" | cut -d" " -f4 &
done
sleep 4

}




test_connection() {

testv=`ping -c 1 -w 1 8.8.8.8 | grep ttl`
    if [ -z "$testv" ];
        then
        echo "Internet Connection is ${RED}DOWN${RESET}"
        sleep 1
        return 1
    else
        echo "Internet Connection is ${GREEN}UP${RESET}"
        sleep 1
        return 1
    fi

}



Ipv4_chk() {
echo "your IP is ${GREEN}`ifconfig wlan0 | grep "inet " | cut -d't'
-f2 | cut -d'n' -f1`${RESET}"
sleep 1
return 1
}







#nslookup function will be added laterz
lookup () {

echo -e "input the address / Q to exit"
read que
      if [ "$que" == "Q" ];
            then
            return 1
      else
            nslookup "$que" >> "$default_out"nslookup.txt
      fi

}
```

```
who_is() {

echo -e "IP addr lookup / Q to exit"
read iaddr
      if [ "$iaddr" == "Q" ];
            then
            return 1
      else
          whois "$iaddr" >> "$default_out"whois.txt
      fi

}



ssh_connect () {

echo -e "[hostname/ip/port] / q to exit"
echo -e "hostname ?"
read uissh1
            if [ "$uissh1" == "q" ];
            then
               return 1
          fi
echo -e "ip ?"
read uissh2
            if [ "$uissh2" == "q" ];
            then
               return 1
            fi
echo -e "port ?"
read uissh3
            if [ "$uissh3" == "q" ];
            then
               return 1
            fi

ssh "$uissh1"@"$uissh2" -p "$uissh3"

}
```

```
    read -n1 -s
    case "$REPLY" in
#    "1")  sh /Desktop/GITS/linecalculator_v05.sh ;;
    "1")  line_calc ;;
#   "2")  sh /Desktop/GITS/fw_entry.sh ;;
    "2")  iptables_blk ;;
```

```
#     "3")  sh /Desktop/GITS/nc_manager.sh ;;
    "3")  nc_manager ;;
#     "4")  sh /Desktop/GITS/md5c.sh ;;
    "4")  md5 ;;
#     "5")  sh /Desktop/GITS/sha256c.sh ;;
    "5")  sha256 ;;
    "6")  tar_archiever ;;
    "8")  scan_last_two  ;;
    "9")  test_connection ;;
    "l")  Ipv4_chk ;;
    "w")  who_is ;;
    "s")  ssh_connect ;;
    "n")  lookup ;;
    "q")  exit                    ;;
#     "q")  echo "case sensitive!!"   ;;
     * )  echo "invalid option"     ;;
    esac
    sleep 1
done


====================================================================



#!/bin/bash


#/home/pi/lncd_arch/datapool_db.txt


network_log() {

BSSID="`grep "Network" <<< "$1" | cut -d' ' -f4`"
manuf="`grep "Manuf" <<< "$1" | cut -d':' -f2 | xargs`"
first_seen="`grep -m 1 "First" <<< "$1" | cut -d':' -f2-4 | xargs`"
last_seen="`grep -m 1 "Last" <<< "$1" | cut -d':' -f2-4 | xargs`"
type="`grep -m 1 "Type" <<< "$1" | cut -d':' -f2 | xargs`"
SSID="`grep -m 1 "SSID       :" <<< "$1" | cut -d':' -f2 | xargs`"
Beacon="`grep -m 1 "Beacon" <<< "$1" | cut -d':' -f2 | xargs`"
packets="`grep -m 1 "Packets" <<< "$1" | cut -d':' -f2 | xargs`"
WPS="`grep -m 1 "WPS" <<< "$1" | cut -d':' -f2 | xargs`"
encryption="`grep -m 1 "Encryption" <<< "$1" | cut -d':' -f2 |
xargs`"
WPA_ver="`grep -m 1 "WPA Version" <<< "$1" | cut -d':' -f2 | xargs`"
channel="`grep -m 1 "Channel" <<< "$1" | cut -d':' -f2 | xargs`"
frequency="`grep -m 1 "Frequency" <<< "$1" | cut -d' ' -f5,9 |
xargs`"
max_seen_packets="`grep -m 1 "Max Seen" <<< "$1" | cut -d':' -f2 |
xargs`"
LLC="`grep -m 1 "LLC" <<< "$1" | cut -d':' -f2 | xargs`"
data="`grep -m 1 "Data" <<< "$1" | cut -d':' -f2 | xargs`"
crypt="`grep -m 1 "Crypt" <<< "$1" | cut -d':' -f2 | xargs`"
fragments="`grep -m 1 "Fragments" <<< "$1" | cut -d':' -f2 | xargs`"
retries="`grep -m 1 "Retries" <<< "$1" | cut -d':' -f2 | xargs`"
total="`grep -m 1 "Total" <<< "$1" | cut -d':' -f2 | xargs`"
datasize="`grep -m 1 "Datasize" <<< "$1" | cut -d':' -f2 | xargs`"
min_position="`grep -m 1 "Min Pos" <<< "$1" | cut -d':' -f2 |
xargs`"
max_position="`grep -m 1 "Max Pos" <<< "$1" | cut -d':' -f2 |
xargs`"
```

```
peak_position="`grep -m 1 "Peak Pos" <<< "$1" | cut -d':' -f2 |
xargs`"
avg_position="`grep -m 1 "Avg Pos" <<< "$1" | cut -d':' -f2 |
xargs`"
last_BSSTS="`grep -m 1 "Last BSSTS" <<< "$1" | cut -d':' -f2-4 |
xargs`"
seen_by="`grep -m 1 "Seen By" <<< "$1" | cut -d' ' -f8 | xargs`"

echo "$BSSID,$manuf,$first_seen,$last_seen,$type,$SSID,$Beacon,
$packets,$WPS,$encryption,$WPA_ver,$channel,$frequency,
$max_seen_packets,$LLC,$data,$crypt,$fragments,$retries,$total,
$datasize,$min_position,$max_position,$peak_position,$avg_position,
$last_BSSTS,$seen_by"

}


client_log() {

c_network_id="`head -1 <<< "$test1" | cut -d' ' -f4`"
c_mac="`grep "Client" <<< "$1" | cut -d' ' -f5`"
c_manuf="`grep "Manuf" <<< "$1" | cut -d':' -f2 | xargs`"
c_first_seen="`grep -m 1 "First" <<< "$1" | cut -d':' -f2-4 |
xargs`"
c_last_seen="`grep -m 1 "Last" <<< "$1" | cut -d':' -f2-4 | xargs`"
c_type="`grep -m 1 "Type" <<< "$1" | cut -d':' -f2 | xargs`"
c_channel="`grep -m 1 "Channel" <<< "$1" | cut -d':' -f2 | xargs`"
c_frequency="`grep -m 1 "Frequency" <<< "$1" | cut -d' ' -f6,10 |
xargs`"
c_max_seen="`grep -m 1 "Max Seen" <<< "$1" | cut -d':' -f2 | xargs`"
c_carrier="`grep -m 1 "Carrier" <<< "$1" | cut -d':' -f2 | xargs`"
c_encoding="`grep -m 1 "Encoding" <<< "$1" | cut -d':' -f2 | xargs`"
c_LLC="`grep -m 1 "LLC" <<< "$1" | cut -d':' -f2 | xargs`"
c_data="`grep -m 1 "Data" <<< "$1" | cut -d':' -f2 | xargs`"
c_crypt="`grep -m 1 "Crypt" <<< "$1" | cut -d':' -f2 | xargs`"
c_fragments="`grep -m 1 "Fragments" <<< "$1" | cut -d':' -f2 |
xargs`"
c_retries="`grep -m 1 "Retries" <<< "$1" | cut -d':' -f2 | xargs`"
c_total="`grep -m 1 "Total" <<< "$1" | cut -d':' -f2 | xargs`"
c_datasize="`grep -m 1 "Datasize" <<< "$1" | cut -d':' -f2 | xargs`"
c_min_position="`grep -m 1 "Min Pos" <<< "$1" | cut -d':' -f2 |
xargs`"
c_max_position="`grep -m 1 "Max Pos" <<< "$1" | cut -d':' -f2 |
xargs`"
c_peak_position="`grep -m 1 "Peak Pos" <<< "$1" | cut -d':' -f2 |
xargs`"
c_avg_position="`grep -m 1 "Avg Pos" <<< "$1" | cut -d':' -f2 |
xargs`"
c_seen_by="`grep -m 1 "Seen By" <<< "$1" | cut -d' ' -f9 | xargs`"


echo "$c_network_id,$c_mac,$c_manuf,$c_first_seen,$c_last_seen,
$c_type,$c_channel,$c_frequency,$c_max_seen,$c_LLC,$c_carrier,
$c_encoding,$c_data,$c_crypt,$c_fragments,$c_retries,$c_total,
$c_datasize,$c_min_position,$c_max_position,$c_peak_position,
$c_avg_position,$c_seen_by"
}
```

```
clientparser() {

loop_count="`grep -n "Client" <<< $test1 | wc -l`"
segment_Client_line_index="`grep -n "Client" <<< $test1 | cut -d':'
-f1`"
l_index="`tail -1 <<< $segment_Client_line_index`"


      if [ "$loop_count" == "1" ]; then

            condition_client_p="`tail --
lines=+"$segment_Client_line_index" <<< $test1`"
            client_log "$condition_client_p" >>
/home/pi/Desktop/clients.txt

      else
            loop_count=$((loop_count-1))

            for i in `seq 1 $loop_count`; do

                  f_line="`sed "${i}q;d" <<<
$segment_Client_line_index`"
                  let i++
                  s_line="`sed "${i}q;d" <<<
$segment_Client_line_index`"
                  client_p="`awk -v s="$f_line" -v e="$s_line"
'NR>=s&&NR<e' <<< $test1`"
                  client_log "$client_p" >>
/home/pi/Desktop/clients.txt

            done

            client_p_last="`tail --lines=+"$l_index" <<< $test1`"
            client_log "$client_p_last" >>
/home/pi/Desktop/clients.txt

      fi

}



f_index_line="`cat /home/pi/lncd_arch/datapool_db.txt | grep
"Network " | cut -d':' -f1 | cut -d' ' -f2 | sed '1q;d'`"
l_index_line="`cat /home/pi/lncd_arch/datapool_db.txt | grep
"Network " | cut -d':' -f1 | cut -d' ' -f2 | wc -l`"
l_index_line=$((l_index_line-1))


for i in `seq 1 $l_index_line`; do

val1="`cat /home/pi/lncd_arch/datapool_db.txt | grep "Network " |
cut -d':' -f1 | sed "${i}q;d"`"
let "i++"
val2="`cat /home/pi/lncd_arch/datapool_db.txt | grep "Network " |
cut -d':' -f1 | sed "${i}q;d"`"
test1="`cat /home/pi/lncd_arch/datapool_db.txt | sed -n "/$
{val1}:/,/${val2}:/{/${val2}:/b;p}"`"
network_cid="`head -1 <<< "$test1" | cut -d' ' -f4`"
network_p="`sed -n "/Network /,/Client /{/Client /b;p}" <<<
```

```
"$test1"`"
network_log "$network_p" >> /home/pi/Desktop/networks.txt
clientparser

done


last_network_index="`cat /home/pi/lncd_arch/datapool_db.txt | grep -
n "Network " | tail -1 | cut -d':' -f1`"
network_p_last="`cat /home/pi/lncd_arch/datapool_db.txt | tail --
lines=+"$last_network_index"`"
last_network_p="`sed -n "/Network /,/Client /{/Client /b;p}" <<<
"$network_p_last"`"
network_log "$network_p_last" >> /home/pi/Desktop/networks.txt

test1=$network_p_last
clientparser
```