ADVANCED COMBAT MAN SYSTEM

By

Maj Naveel Rehman
Maj M Junaid Aslam
Capt Sabeeh Bukhari
Capt Hasan Masood

Submitted to the Faculty od Department of Electrical Engineering, Military College of Signals, National University of Science and Technology, Islamabad in partial fulfilment for the requirements of a BE Degree in Electrical Telecom Engineering

2019

This is to certify that the

Final Year Project Titled

ADVANCED COMBAT MAN SYSTEM

Submitted By

Maj Naveel Rehman
Maj M Junaid Aslam
Capt Sabeeh Bukhari
Capt Hasan Masood

Has been accepted towards the requirements
for the undergraduate degree
in
(BETE – 52 A)

Abdul Wakeel
(Asst Prof)

Military College of Signals
National University of Science and Technology
Islamabad, Pakistan

ADVANCED COMBAT MAN SYSTEM

ABSTRACT

With the prevalence of global urbanization, the Pakistan Army will inevitably need to engage in urban operations. The urban environment presents a whole new multi-dimensional battlefield – one which not only requires changes to conventional combat operations and tactics, but also a need to overcome the strategic advantage afforded to an adversary that is concealed and entrenched in an urban environment. The Advanced Combat Man System (ACMS) is an urban fighting system for the Third Generation Army. It is designed to address the challenges faced in urban operations by enhancing command and control (C2), situational awareness, survivability and lethality of the soldier. With the ACMS, soldiers become part of a networked force. Soldiers' situational awareness is thus enhanced, allowing them to engage their targets more effectively. To avoid known and unknown danger, soldiers will also able to navigate accurately through the urban battlefield environment.

DEDICATION

This dissertation is dedicated in thanks to ALLAH ALMIGHTY our Creator who has blessed us with wisdom, knowledge and understanding then to our parents for their direction and their endless support.

Further to our Faculty for their guidance and supervision, without their help and supervision this project would not have been made possible.

<u>DECLARATION</u>

We hereby declare that no content and form of work presented in this thesis has been submitted in support of another award of qualification or degree either in this institution or anywhere else.

ACKNOWLEDGEMENT

The accomplishment of this project would not have been possible without the contribution of different people the names of which cannot be reckoned with. Their assistances are sincerely respected and thankfully recognized. We would like to thank following:

- To Allah Almighty who is the most Powerful and Merciful and everything is in His obedience.
- Asst Prof. Dr. Abdul Wakeel for his support and guidance during the entire project.
- I acknowledge my sincere indebtedness and gratitude to my parents for their love, dream and sacrifice throughout my life. I am really thankful for their sacrifice, patience, and understanding that were inevitable to make this work possible. Their sacrifice had inspired me from the day I learned how to read and write until what I have become now. I cannot find the appropriate words that could properly describe my appreciation for their devotion, support and faith in my ability to achieve my dreams.
- Lastly I would like to thanks any person which contributes to my final year project directly on indirectly. I would like to acknowledge their comments and suggestions, which was crucial for the successful completion of this study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

KEY TO ABBRIVIATIONS

1.	ACMS	Advanced Combat Man System
2.	AES	Advance Encryption System
3.	AP	Access Point
4.	BPM	Beats Per Minute
5.	CPE	Customer Premises Equipment
6.	ESD	Electrostatic Discharge
7.	GPIO	General Purpose Input Output
8.	GPS	Global Positioning System
9.	ICSP	In Circuit Serial Programming
10.	LED	Light Emitting Diode
11.	MAC	Media Access Control
12.	MCU	Multipoint Control Unit
13.	PoE	Power over Ethernet
14.	PSU	Power Supply Unit
15.	PWM	Pulse Width Modulation
16.	RoHS	Restriction of Hazardous Substances
17.	SDIO	Serial Digital Input Output
18.	SPI	Serial Peripheral
19.	SRAM	Static Random Access Memory
20.	SSID	Service Set Identifier
21.	SoC	System on a Chip
22.	UART	Universal Asynchronous Receiver Transmitter
23.	UTP	Universal Twisted Pair
24.	VoIP	Voice over Internet Protocol
25.	WISP	Wireless Internet Service Provider
26.	WPA	Wireless Protected Access
27.	WLAN	Wireless Local Area Network

INTRODUCTION

Advantageous conditions for the soldiers present on ground can be achieved by use of such technological gadgets and aids that are not held by the enemy and at the same time comprise of things ranging from their clothes to their weapons and other tactical and technological aids that help in achievement of better results in the battlefield. The fruitions can be achieved when all these gadgets are combined together to complement each other and integrated at the same time. The men present in the battlefield will not only need upgraded versions of the traditional weapons and equipment but will also require the induction of newest technologies as soon as they are considered compatible and beneficial to the already existing set of things. The ACMS gives the opportunity and the responsibility to a certain number of identified individuals to enjoy greater command and control over the network by using more details and intricacies for the betterment of their men and achievement of the desired target. Every man who is present on the ground functions as a sensor and eyes and ears of the commander and enables the commander to have a better understanding of the ongoing situation and capture the birds' eye view of the battlefield. This enables the commander to make better decisions keeping the complete picture in mind and also enables his men to carry out different scenarios in extremely coordinated manner. All the people who are part of ACMS will be connected with each other. This allows forming of COP and CLP.

A Force Multiplier

Before the dawn of ACMS, the distribution or the transference of data was done manually. This was challenging because there used to be multiple shortcomings using this method. Time which is of the essence on a battlefield, used to be lost and the accuracy while engaging of the targets was sometimes found wanting due to the lack of clarification or miscommunication. The use of graphics while presenting the information in ACMS, provides commander with clearer picture of the situation and precious time is obtained which can be used to make clearer and much accurate decisions which can have mortal effects for the enemy.

.

CHAPTER 1

1.1 OBJECTIVE

We have knowledge that in an urban terrain or the rise of Sub Conventional Warfare, a commander needs to have a vantage point from where he can observe the situation and make educated and informed decisions to influence the battle scenario. ACMS is a tool for provision of the same at technologically advanced, yet easy to use platform. The idea is to provide a wholesome view of the terrain and the situation while considering the difficulties and demands of the kinetic and demanding nature of the SCW.

1.2 PROPOSED SYSTEM

The proposed work of our project is to develop a system that can be supplemented with real-time wireless command & control system which are designed and implemented through microcontroller, GPS, IP networking and are able to record and transmit orders, bio-signals, video between soldier and commander. This aim of this project is to provide a continuous monitoring of all the soldiers (team members) at any place and time and to design a command and control system for team taking part in any operation.

The proposed system will have following features / Applications

1. ACMS provides very high level safety to Soldiers life during operation
2. This system can be effectively used in any environmental conditions.
3. Analysis/report for different situations can be generated via ACMS.
4. Real-time video transmission for clear picture of operation
5. Any military force can easily adopt this system because of ease of operation, low cost and compact design.
6. System can be extended to hundreds of soldiers.
7. Intelligent network connection (Simultaneously with commander and other squad for efficient transmission inside buildings as well)
8. Soldier can communicate with commander or any soldier by using device to device communication feature without dependency of base station.

1.3 ACMS ARCHITECTURE

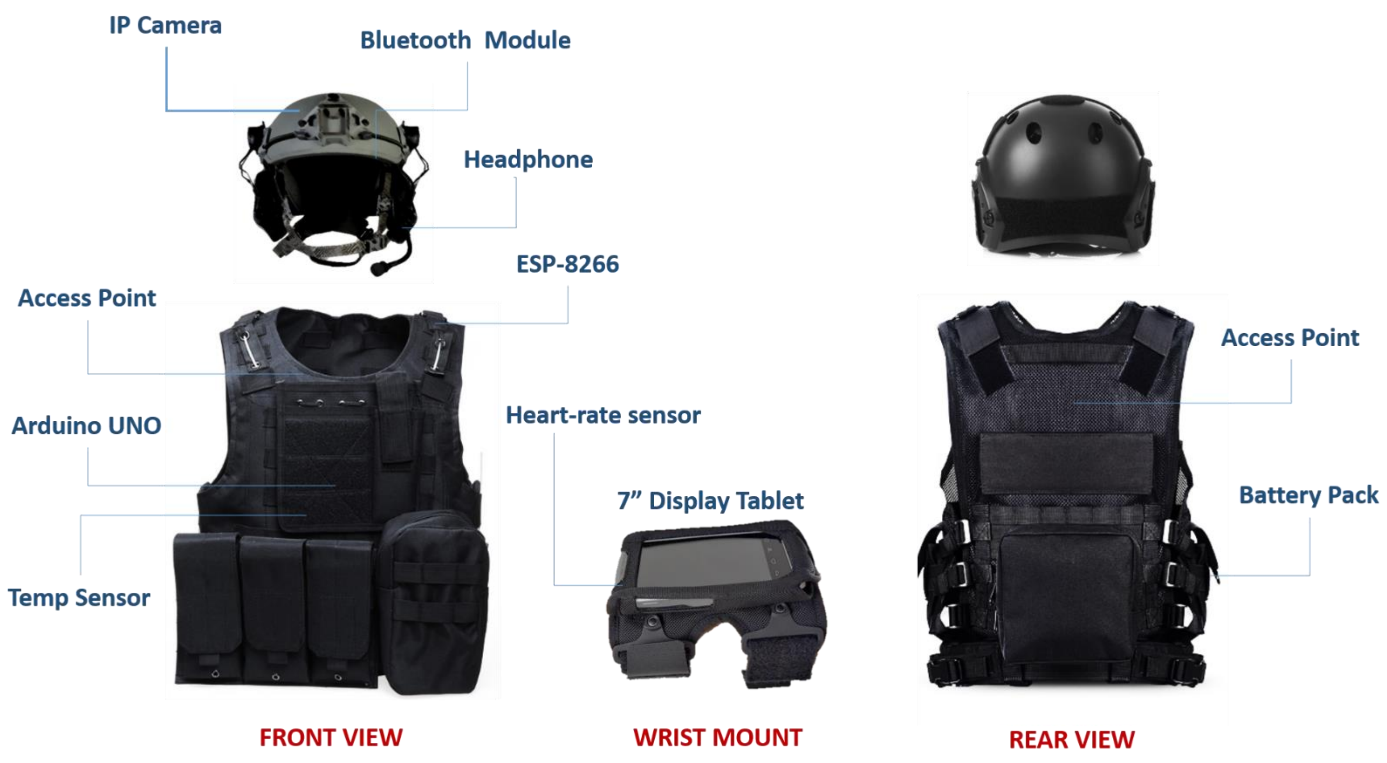


Figure 1.1- ACMS Architecture

1.3.1 TACTICAL HELEMT

The soldier helmet which provides ballistic protection and is equipped soldier communication system headphone and Bluetooth module to connect with VoIP Server running in tablet, an IP Camera along with torch for night operations. IP Camera is connected with Ethernet Switch integrated into soldier's fighting load vest. Headphone is integrated with helmet for 2-way call with commander or with any other soldier at the touch of a button located on tablet for quick access. IP Camera display on tablet shows the video from other soldier camera for clear view and better understanding of situation. Rechargeable search light for night operations has been installed at right side of helmet with power button on it.

1.3.2 BODY ARMR

Soldiers Health Monitoring Module system and network communication system are integrated in soldiers fighting load vest for carrying and interfacing with the sensors and the microcomputer. The soldier is able to adjust the load distribution from shoulders to hips while on the move. 2 x Access Points (TP-Link 5210) are integrated at front-side and back-side of armr vest to provide wireless connection with commander or other syndicate/team members. The tactical helmet is connected and integrated via the usb male/female connector installed on left shoulder of vest.

Arduino microcontroller with temperature and humidity sensor for soldier health monitoring is also integrated inside vest. For power system, 12 Volt 6000mAH rechargeable battery is integrated at side pocket of the vest for easy and quick replacement. The system provides 6 operating hours of power for the sensors, access point and microcontroller. (see figure 2)

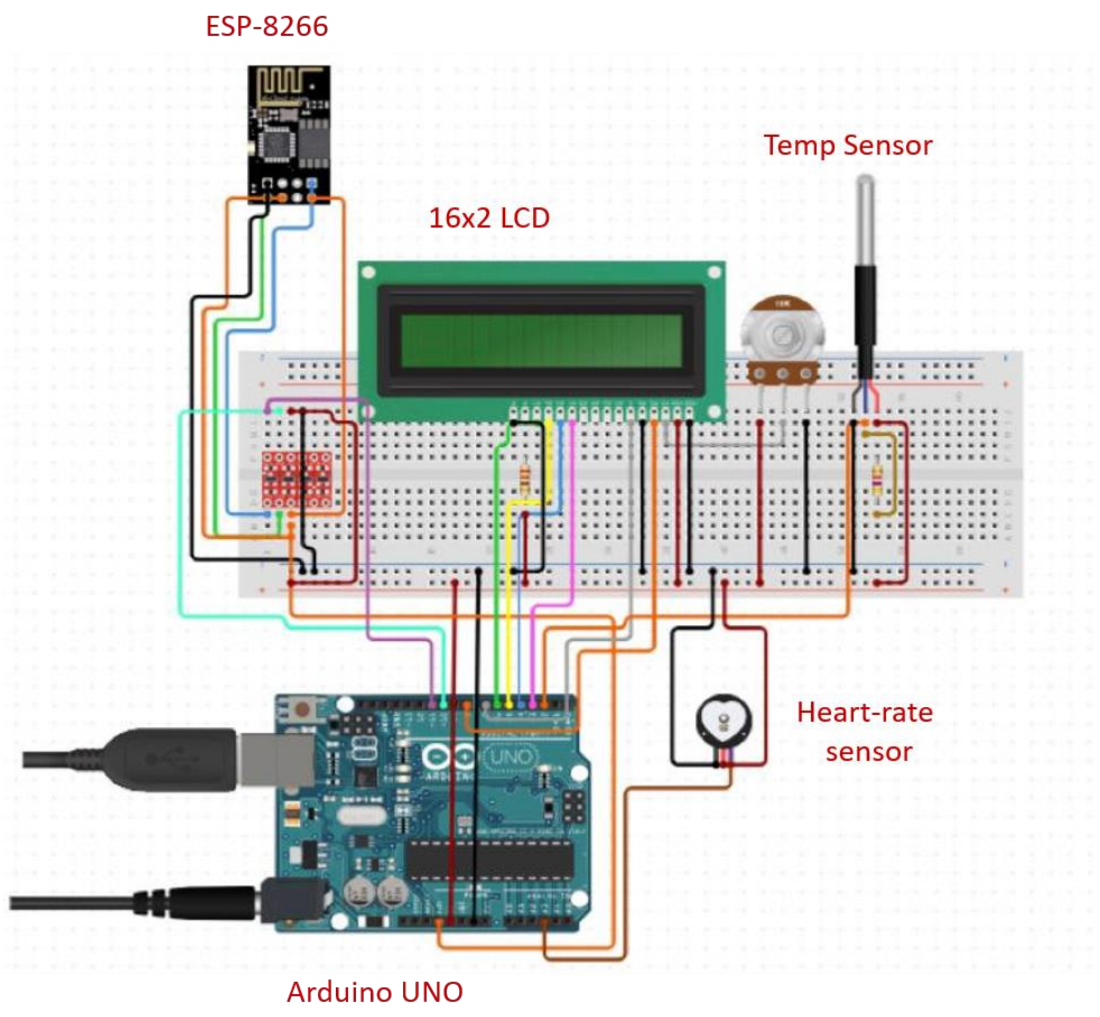


Figure 1.2- Health Monitoring Module Connection Diagram

1.3.3 WRIST TABLET

The Soldier wears a wrist mount tablet for navigation and real time Command and Control of battlefield situation. The display screen is divided into two main parts showing status, map with real time location of complete squad and camera view of any individual.

Tablet is connected with soldier system via WiFi Connection with sodier front Access Point located inside soldier armor vest. Helmet Bluetooth module is connected with tablet for VOIP calls on local LAN with other team members. The tablet display screen will also have SOS/PANIC button for emergency situation.

1.4 PROPOSED BLOCK DIAGRAM

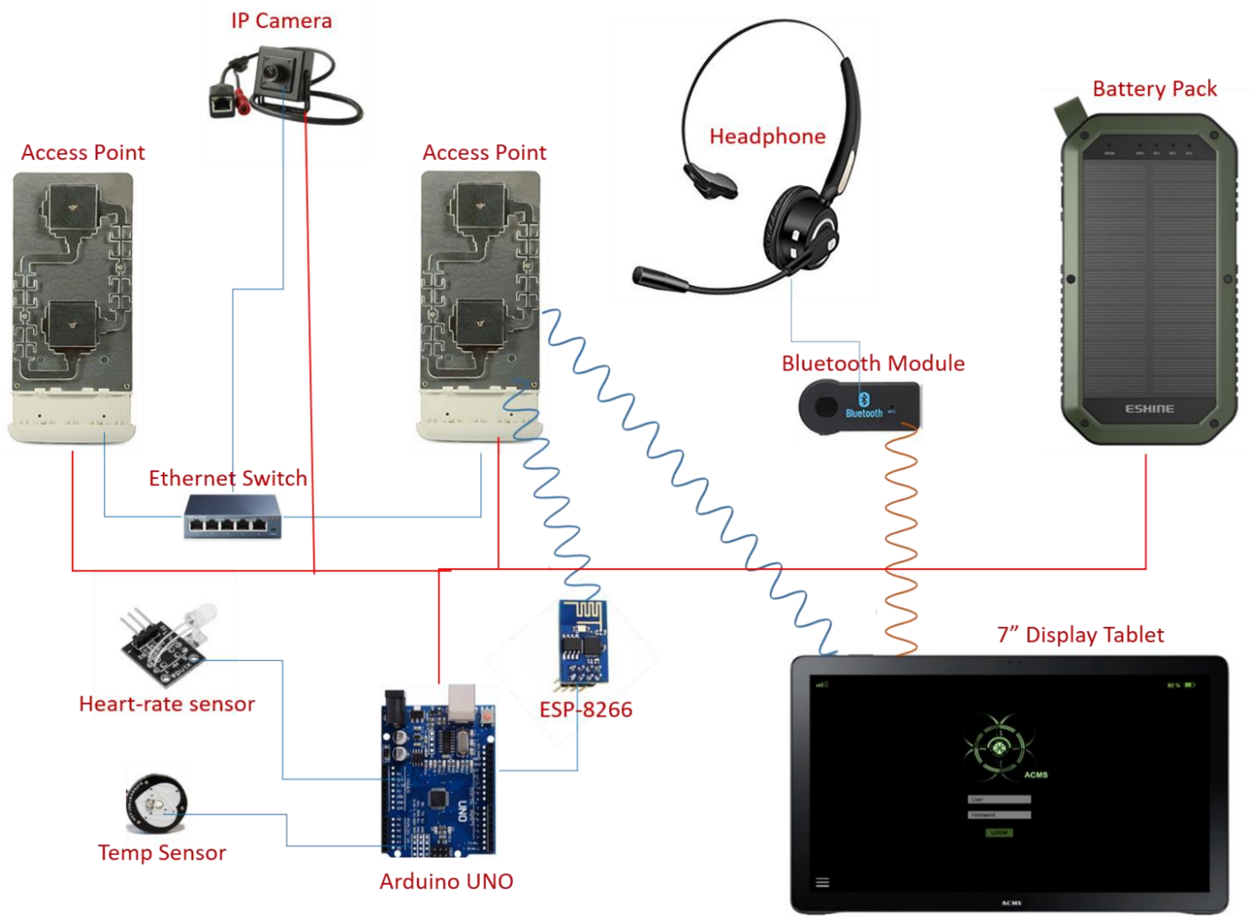


Figure 1.2 - ACMS Equipment Connection Diagram

CHAPTER 2

2.1 OPERATIONAL PROCEDURE

The heart of soldier health monitoring system is microcontroller, In our project we are using arduino microcontroller. To measure the body temperature of soldier temperature sensor is integrated in bullet-proof vest which is connected with Arduino. After every 2 seconds Arduino will upgrade temperature data on self-maintained local webserver. We will use signal conditioning (transducer) to convert output signal of sensor into electrical form. This analog data needs to be converted into digital form from analog form by using ADC inbuilt inside arduino processor . So, the output of the signal conditioner circuit is directly connected to arduino processor. These sensors (LM35 series) are very précised integrated-circuit temperature sensors. There output voltage is linearly proportional to the Celsius (Centigrade) temperature.

Heart beat sensor will give digital output of heart beat of soldier when it is placed over his wrist. Arduino is connected directly with the digital output of sensor to measure Beats per Minute (BPM) rate. It works on the principle of light modulation technique by blood flow through veins at each pulse. ICLM358 chip is used for Heartrate Sensor. It consist of dual low power operational amplifier, super bright green LED and light detector. One LED will work as amplifier while other one will work as comparator. LED needs to be super bright as the light from LED must reflect through vein and need to be detected at receiver. Signal varies with each heart pulse which is detected by receiver, this variation is converted to electrical pulse. Heartrate data is the uploaded to Arduino webserver via ESP8266 Module. (Arduino code is attached with this document)

For Live video broadcasting, a shock proof IP Camera is integrated in front of solder tactical helmet which is further connected to network via Ethernet switch embedded inside jacket. Live video will be transmitted via WLAN to command or any soldier for accurate situation awareness. Soldier can select video of any connected individual via drop-down menu button on tablet.

For communication with commander and other team member, VoIP Server on tablet is installed which works on local LAN Tablet is connected with solder headphone via Bluetooth module.

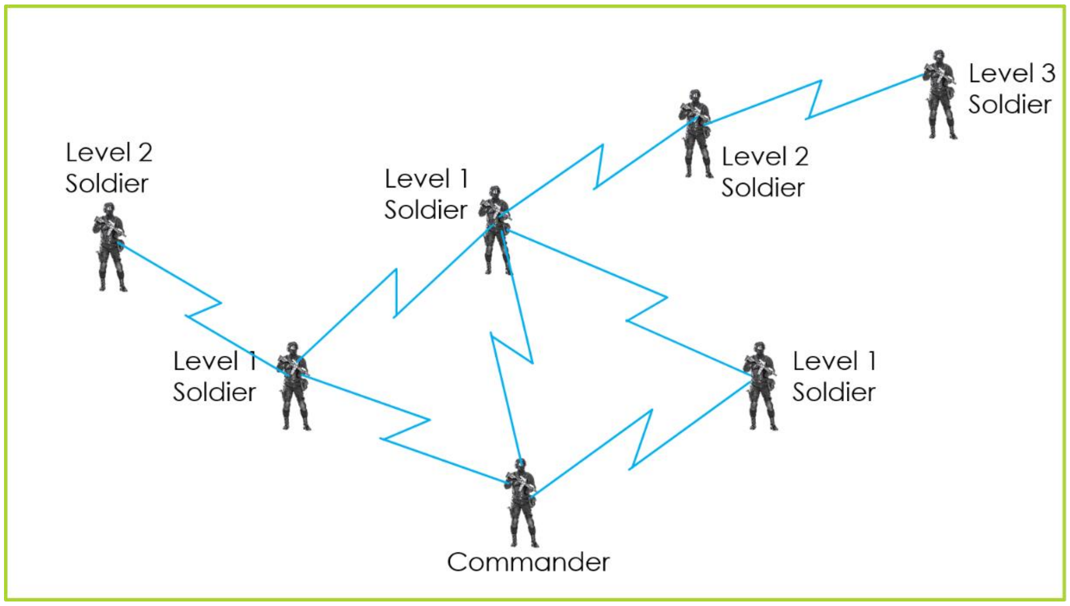


Figure 2.1 - Interconnectivity Diagram

2.2 SOFTWARE WORKING

ACMS Software is designed as a stand-alone software which does not require any backend or server connectivity. Every device running ACMS Software will keep its data in its internal storage and fetch data of other user on regular interval. Main Screen of tablet is divided into 3 main portions. Left half of screen will cover the map of selected area. Maps of different areas can be uploaded into system according to requirement of operation. Right Half of tablet screen is kept for IP Camera View. View of different cameras can be selected by drop-down menu located at top right corner. Soldiers health status is shown at the bottom of screen which include team members name, heart-rate, body temperature along with their GPS coordinates.

Software codes for health monitoring module (Arduino) and ACMS Command and Control Software (Android) is attached as Appendix A and Appendix B respectively.

2.2.1 ACMS Operation Steps

1. Connect Helmet with Vest
2. Power On system using button in armor vest left pocket
3. Switch on ACMS
4. Create an account using allotted Master Password (for first time user only)
5. Enter IP Address in setting tab located at bottom left on tablet screen.
6. Login using username and password.
7. Select Map View Style from dropdown menu
8. Select Camera from Camera Selection menu

9. Check health monitoring tab.
10. Run VoIP Server and check call connection

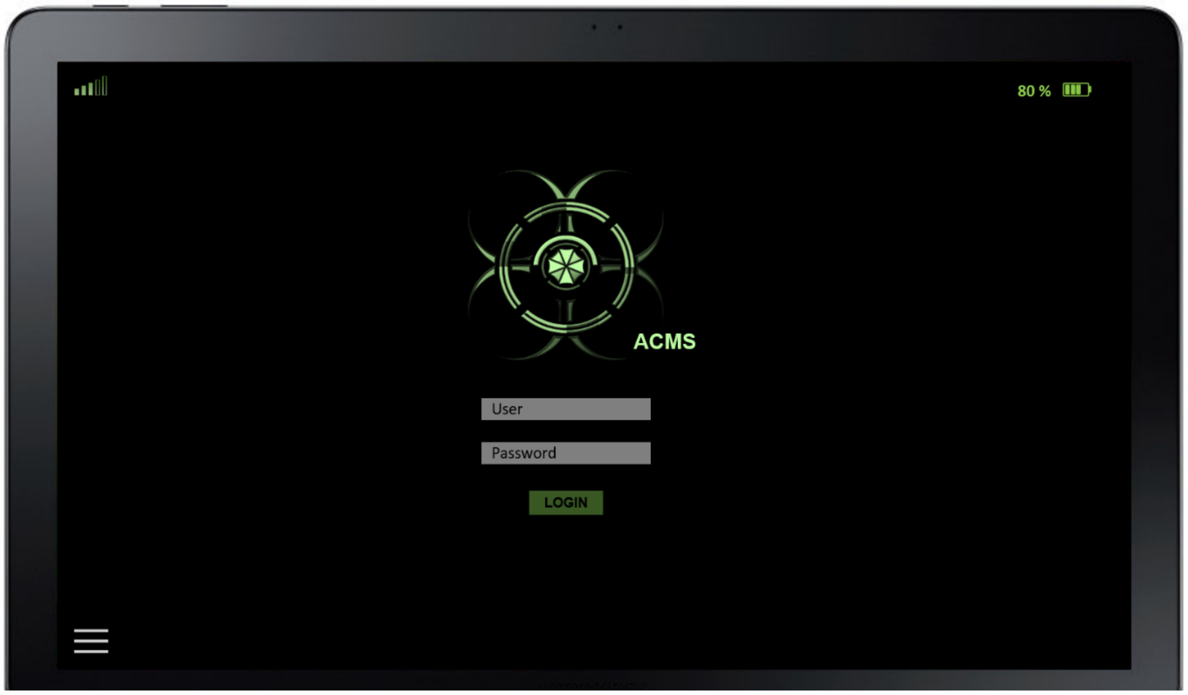
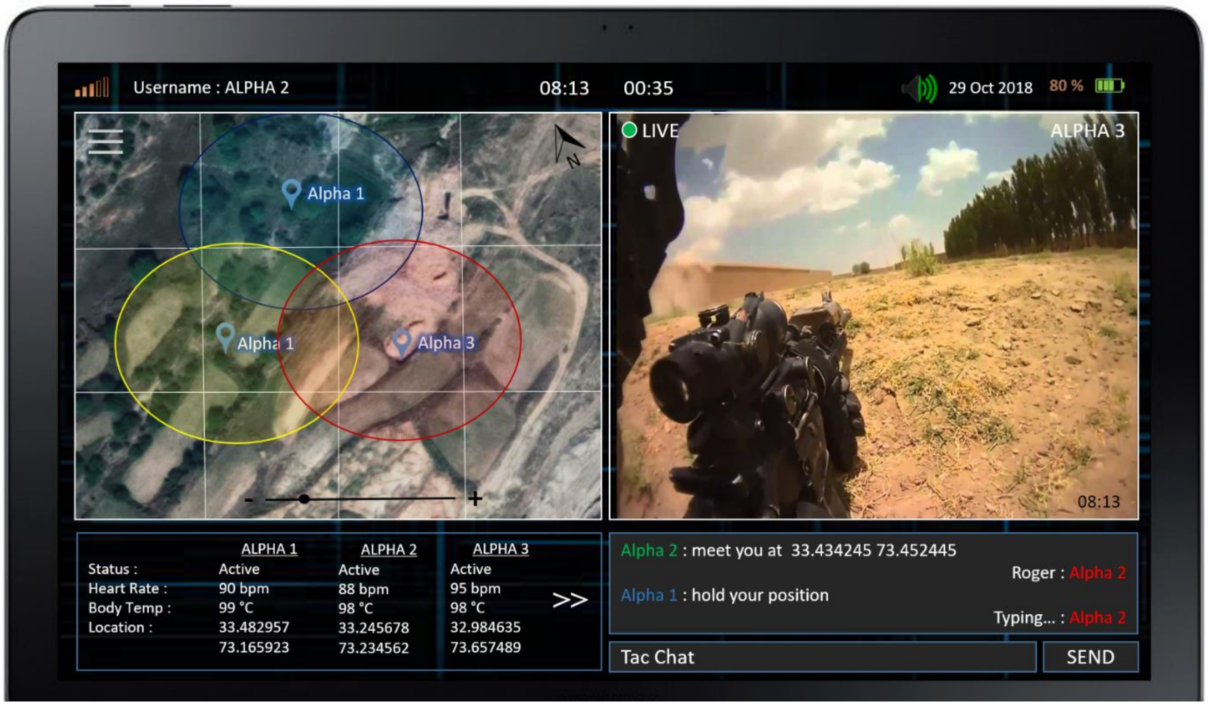


Figure 2.2 - ACMS login Screen



	ALPHA 1	ALPHA 2	ALPHA 3
Status :	Active	Active	Active
Heart Rate :	90 bpm	88 bpm	95 bpm
Body Temp :	99 °C	98 °C	98 °C
Location :	33.482957 73.165923	33.245678 73.234562	32.984635 73.657489

Alpha 2 : meet you at 33.434245 73.452445
Roger : Alpha 2
Alpha 1 : hold your position
Typing... : Alpha 2

Figure 2.3 - ACMS Operational View

2.3 SECURITY FEATURES

2.3.1 Network and Wireless Security

For security of our data over network, we are using AES (Advanced Encryption Standard), which is the latest encryption method available to-date. Our wireless network is protected by WPA-2 (Wireless Protected Access – Version 2) standard. In addition of WPA-2, we have also used MAC Address Filtering to filter out any unauthorized packet of unknown address.
Every device in our network has been allotted a static IP address.

2.3.2 Software Security

To access ACMS software, user must create an account, which can only be done by using Master Key. Without Master Key, ACMS application will not initialize which helps only the authorized users to gain access to system. Team commander will set the Master Key and inform other team member in advance of any operation.

2.3.3 Physical Security

To physically protect the device from getting into hands of opponent, every device is fitted with soldier armor vest. In case any device is compromised, commander can log off the selected user from receiving any update

2.4 OPERATING ENVIRNMENT

1.	Operating Temperature:	-30°C - 70°C (-22°F - 158°F)
2.	Storage Temperature:	-40°C - 70°C (-40°F - 158°F)
3.	Operating Humidity:	10% - 90% (non-condensing)
4.	Storage Humidity:	5% - 95% (non-condensing)

CHAPTER 3

3.1 NETWORK DESCRIPTION – Intelligent Pairing

Each soldier bullet-proof vest is modified to include front and rear interconnected Access Points (AP). These APs can illuminate area about 1 km of radius providing backend connectivity along with level 2 pairing for farthest soldier as well. All Wireless communication is WPA2 encrypted along with MAC-Address filtering, disabled SSID broadcast and Inbound and outbound firewall for enhanced security. Following are the parameters for wireless connection.

Datalink Protocol:	IEEE 802.11b
Network Address:	192.168.1.0
Subnet Mask:	255.255.255.0
Default Gateway:	192.168.1.5
Encryption Algorithm:	WPA2 - PSK
Encryption Type:	AES
MTU Size:	1500 bytes

3.1.1 NETWORK CONNECTIVITY DIAGRAM

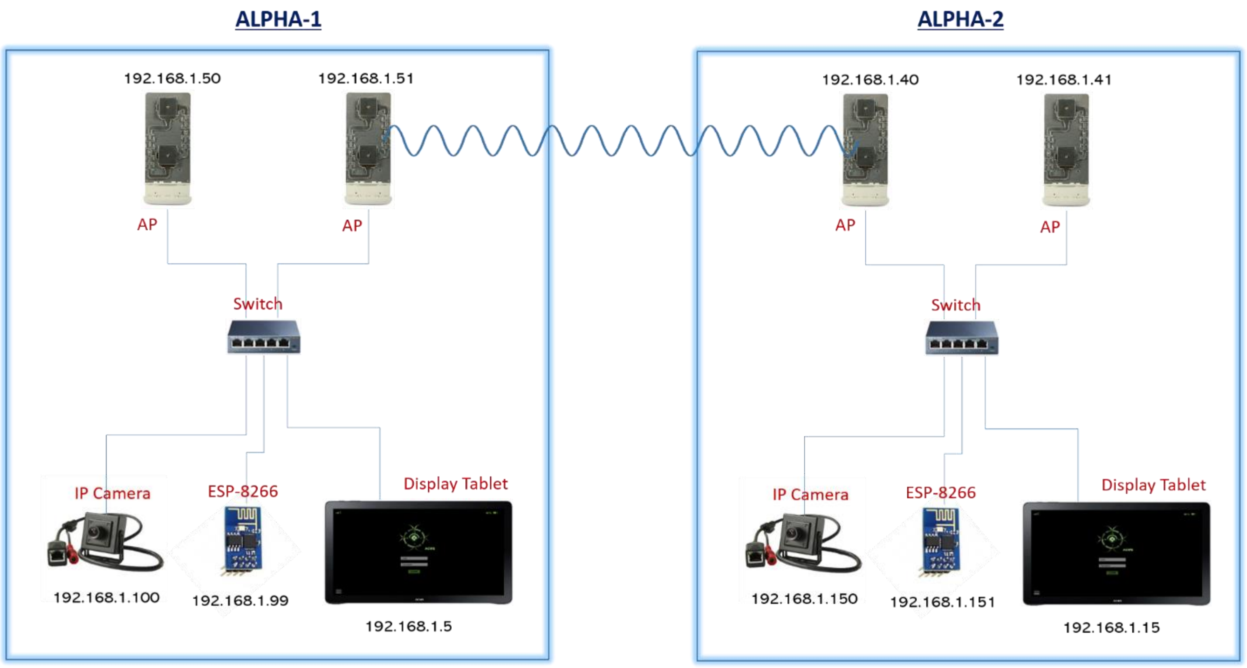


Figure 3.1 - TPLink 5210 (Shop Delta)Figure 3 - Network Topology Diagram

3.3 IP ADDRESS ALLOTMENT

Table 1 - IP Address Allotment

Ser	Equipment	IP Address	Subnet Mask	Remarks
Alpha – 1 (Commander)				
1	Access Point Front	192.168.1.10	255.255.255.0	
2	Access Point Rear	192.168.1.11	255.255.255.0	
3	IP Camera	192.168.1.12	255.255.255.0	
4	VoIP Server	192.168.1.13	255.255.255.0	
5	Control Tab	192.168.1.14	255.255.255.0	
6	ESP-8622	192.168.1.15	255.255.255.0	
7	Admin AP	192.168.1.5	255.255.255.0	
8	Admin PC	192.168.1.6	255.255.255.0	
Alpha – 2 (Team Member 1)				
1	Access Point Front	192.168.1.20	255.255.255.0	
2	Access Point Rear	192.168.1.21	255.255.255.0	
3	IP Camera	192.168.1.22	255.255.255.0	
4	VoIP Server	192.168.1.23	255.255.255.0	
5	Control Tab	192.168.1.24	255.255.255.0	
6	ESP-8622	192.168.1.25	255.255.255.0	
7	Admin AP	192.168.1.3	255.255.255.0	
8	Admin PC	192.168.1.4	255.255.255.0	

CHAPTER 4

4.1 EQUIPMENT REQUIRED

Table 2 - Equipment Required for 1 x System

Ser	Equipment	Model	Quantity	Remarks
1	Control Tablet	Mi One – 7”	1	
2	Arduino Microcontroller	UNO R3	1	
3	Access Point	TPLINK 5210	2	
4	Ethernet Switch	4 Ports 100 Mbps	1	
5	IP Camera	2 MP 10/100 Mbps	1	
6	Headset	Stereo	1	
7	Arduino Wi-Fi Module	ESP-8266	1	
8	Bluetooth Module	-	1	
9	Heart Rate Sensor	-	1	
10	Temperature Sensor	-	1	
11	Tactical Helmet	-	1	
12.	Armor Load Vest	-	1	
13.	Battery	12 V – 5000 mAH	2	1 x Spare
14.	Battery Bank	5V – 5000 mAH	1	
14.	Jumper Cables	Male-Male	2 Sets	
15.	Jumper Cables	Male-Female	2 Sets	
16.	Tablet Wrist Pouch	-	1	
17.	Ethernet Cable	Cat-6	10 meter	
18.	Connector	RJ-45	10	
19.	Arduino LCD Screen	16x2	1	
20.	PoE	-	1	

4.2 POWER CALCULATION

Table 3 - Power requirement of 1 x System

Ser	Equipment	Voltage	AH	Remarks
1	AP	12VDC	1AH	
2	Switch TL-SF1008D	12VDC	0.5AH	
3	IP Camera	12 VDC	0.25AH	
4	Arduino UNO	5 VDC	0.1mAH	
5	Tablet	8 VDC	-	In-built battery
6	Helmet Torch	5 VDC	-	In-built battery
	Total		1.76 AH	

4.3 WEIGHT CALCULATION

Table 4 - Weight Addition of Individual

Ser	Equipment	Qty	Weight	Remarks
1	AP	2	200 gm	100 gm each
2	Switch TL-SF1008D	1	75 gm	
3	IP Camera	1	55 gm	
4	Arduino UNO	1	25 gm	
5	Headphone	1	70 gm	
6	Tablet	1	200 gm	
7	Battery 12 V	1	400 gm	
8	Battery Bank	1	320 gm	
9	Misc	1	50 gm	
		Total	1395 gm	

4.4 COST ESTIMATE

Table 5 - Cost of 1x System

Ser	Equipment	Qty	Price	Amount	Remarks
1	Access Point	2	6000	12000	TP-Link 5210 N
2	Switch	1	1000	1000	TL-SF1008D
3	IP Camera	1	10000	10000	Hik-Vision
4	Armr Vest	1	14000	14000	-
5	Helmet	1	8000	8000	-
5	Tablet	1	12000	12000	-
6	Battery 12 V - 5AH	2	5000	10000	-
7	Battery Bank (5V)	1	8000	8000	
8	Arduino R3	1	800	800	-
9	Heart Rate Sensor	1	1500	1500	-
10	Temp Sensor	1	600	800	-
11	Headphone	1	1000	1000	-
12	PoE	1	1000	1000	-
13	ESP-8266	1	850	850	
14	Bluetooth Module	1	800	800	
15	Cat 6 Ethernet Cable	10 m	50	500	
16	Jumper Cables	4 Sets	200	800	
17	Tablet Wrist Pouch	1	1500	1500	
18	RJ-45 Connectors	10	25	250	
19	16x2 LCD Screen	1	800	800	
20	Misc	-	5000	5000	-
			Total:	90600	

CHAPTER 5

5.1 TPLINK-5210

TP-LINK_5210 is a very powerful piece of equipment that is used as an efficient tool for provision of wireless internet at considerable distances. It is used as a solution to problems posed by such companies or installations that have wireless internet requirements that might be external in nature. Wireless connectivity if every type flourishes in the external setting as there is little possibility of signals being reflected or broken in comparison with the internal or indoor setting. Therefore wireless communication and connectivity tends to flourish in outdoor conditions.

5.1.1 High Power and High Sensitivity

This feature ensures that the signals with low amount of strength can travel longer distances due to the excessive amount of power and the sensitivity to weak signals mean that the weakest signals can be detected. Both of this qualities combined, ensure that even the signals with considerably less strength can travel to the destination and can be detected there ensuring larger data rate at same distances in comparison with similar other devices which lack this combination and feature.

5.1.2 12dBi Dual Polarized Antenna

The amount of power in this antenna ensures the stability and quality of communication. The different polarizations also allow engagement with several wireless devices

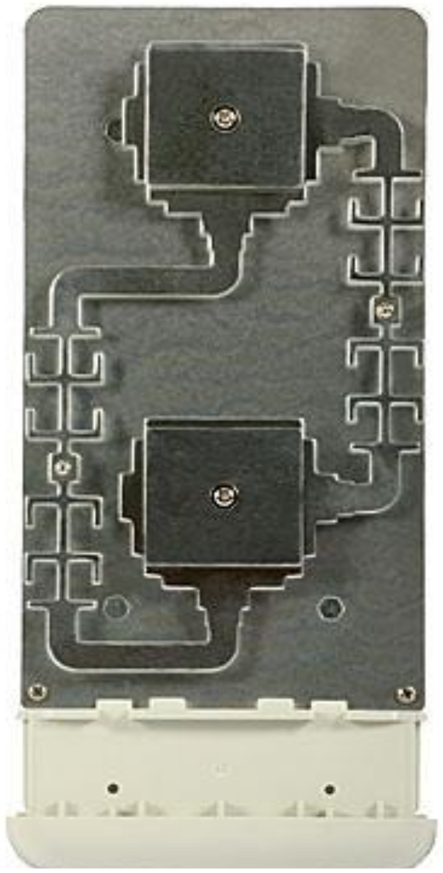


Figure 5.1 - TPLink 5210 (Shop Delta)

5.1.3 Features (TP Link)

• Interface :	1 x 10/100Mbps Auto-Sensing Port (RJ-45)
• Power Supply :	1.0A Linear PSU / 12VDC
• Wireless Standards :	IEEE 802.11g, IEEE 802.11b

Frequency :	2.4 GHz - 2.4835 GHz
Transmit Power :	(Peak Output Power) <27dBm
Certification :	CE, RoHS , FCC
Rx sensitivity:	-76 dBm (54m)
Antenna:	12 dBi (Directional)
ESD protection:	15 kV
Modes:	AP , WISP and AP Router
PoE:	upto 60 m (UTP cable)

Figure 5.2 - TPLink 5210 Sideview (TP Link)

5.2 ESP – 8266

The wireless microchip allows the user to cater for different facets that require attention. It can give the authority to any small device to become part of a wireless network.

The module has the capability of either being the sole controller of the complete enterprise or can also function as an extension or as part of an already functioning multipoint control unit. The optimal functioning of the cache allows the maximum usage of its memory. It can also be used as a wireless assembly connector.

The low cost Wi-Fi microchip different features and combines them together in such a way that its size does not increase as a result and the circuit extensions need are minimal.

In addition to all the features mentioned above, the microchip combines advanced version of L106 processor with on chip SRAM. It can also be combined with several other devices and the SDK has already embedded codes for various uses and exercises.

Figure 5.3 - ESP 8266 Module (Micro Robotics)

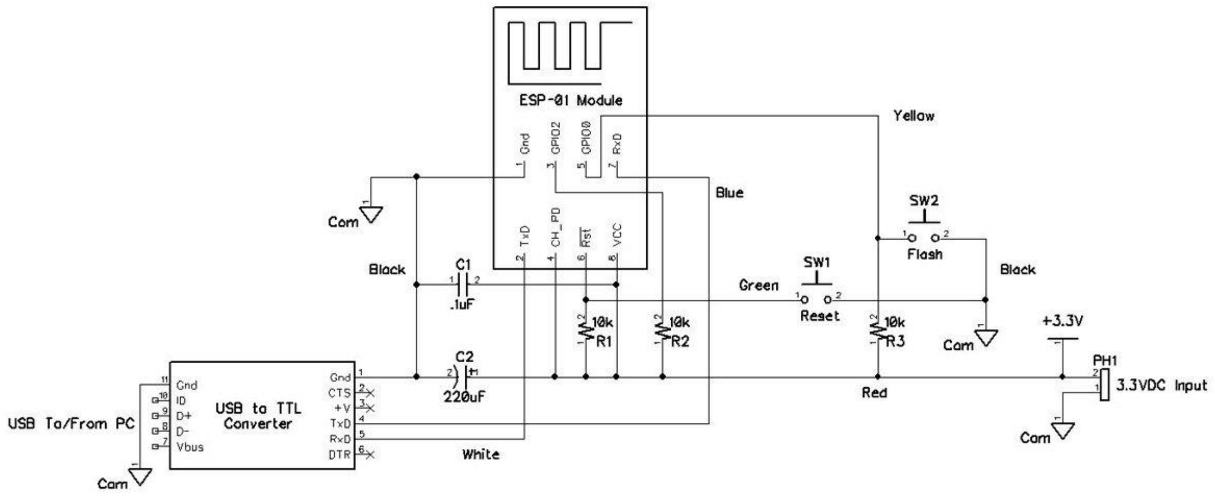


Figure 5.4 - ESP 8266 Flashing Circuit (Micro Robotics, n.d.)

5.3 ARDUINO UNO R3

It is a microcontroller board which can be read and altered to meet the requirements of the user, therefore making it a viable option for several applications as it is prone to change. It can also be combined with other devices to make variants to meet the intended outcome.

5.3.1 Specifications

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage	7-12V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13

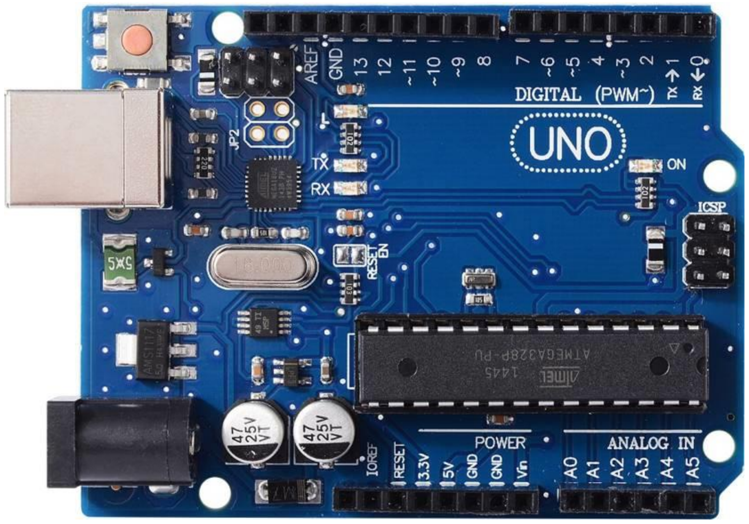


Figure 5.5 - Arduino UNO R3 (Module 143)

5.4 IP CAMERA

For our project, we are using Hik-Vision (DS-2CD2D14WD) Pinhole Network Camera (1.0 MP WDR). IP Camera is mounted on Tactical Helmet to provide live video streaming of area of operation.

5.4.1 Key Features

- Up to 1 Megapixel (1280 x 720) Resolution
- HD720p Real-Time Video
- 100 dB WDR
- 3D DNR

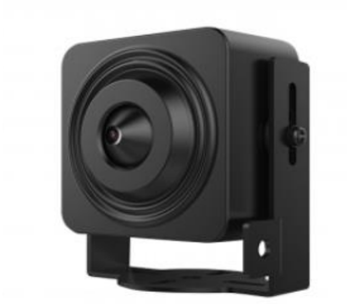


Figure 5.6 - Hik-Vision IP Camera (Hik Vision)

5.4.2 Specifications

Video Compression	H.264/MJPEG
Image Sensor	CMOS - ¼" progressive scan
Network Port	Ethernet
Ethernet Speed	100Mbps
Maximum Image Resolution	1280 x 720
Power Supply	12 VDC ±10%
Weight	55 g (1.94 ozs)

5.4.3 Dimensions

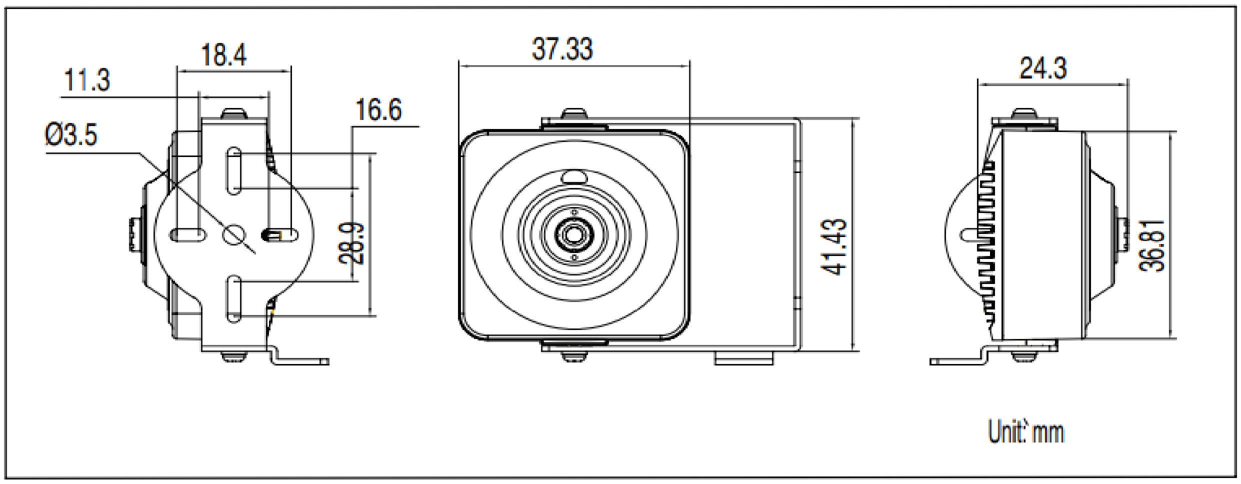


Figure 5.7 - Hik-Vision Network Camera Dimensions (Hik Vision)

5.5 TEMPERATURE SENSOR

The DS18B20 is a digital thermometer which provides temperature (Celsius) readings having an alarm function with capability of retaining data of user-configurable upper and lower trigger points. It requires only one data line to connect with a central microprocessor and communicates over a 1-Wire bus. In addition, it works on the principle of "parasite power" i.e., can derive power directly from the data line, excluding the need for an external power source. It has a distinctive serial code which allows multiple thermometers to operate on the same bus simultaneously, which makes it simple to actuate multiple thermometers spread over a large zone using only a single microprocessor. The protocol which implements bus communication using one control signal is known as "Maxim's exclusive 1-Wire bus protocol". Only a weak pullup resistor is required by the control line to regulate the control signal as all the devices are connected to the bus via a 3-state or open-drain port. Each device on the bus system is identified by the microprocessor using its distinctive code. HVAC environmental control, temperature monitoring systems and process control and monitoring systems are benefiting from this device.

5.5.1 Operation — Measuring Temperature

The temperature sensor (direct - to - digital) is the basic component of our system for this digital thermometer. The resolution of the sensor is programmable and is related to several temperature extensions. The thermometer is powered up with a low power and goes into inoperative domain. An imperative is issued by the thermometer to institute a temperature calculation and for its conversion into digital thermal data. After the conversion, the recorded data is placed in a temperature register located in the memory and the thermometer goes back into inoperative domain. Certain programming codes can not be used to get the transmitting and conversion responses while powered with parasite power whereas, they can be used to get desired responses while powered up with an external source.

5.5.2 Features

- Single bus connection claims only one port for communicating

- EEPROM and the temperature sensor are working together as a unit to reduce the number of parts in the device
    - Wide range of temperature calculations
    - Precise up to ± 0.5°C
    - The resolution of the sensor is user-configurable
    - No exterior parts needed
    - Only 2 pins are used for operating while deriving power from the data line
- Uniforms divided temperature sensing applications with extensibility capacity
- Stretchable user-describable nonvolatile (NV) alarm settings with alarm search is used to recognize the devices with temperatures exceeding the defined levels.

5.5.2 Pin Configuration

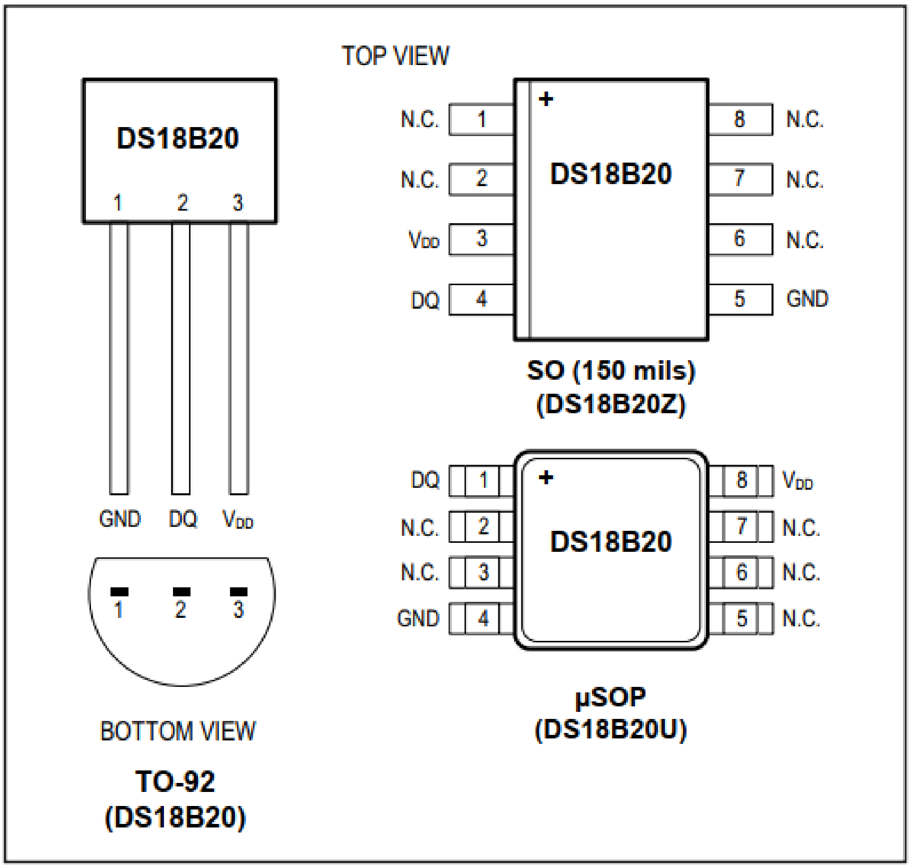


Figure 5.8 - Temperature Sensor Pin Configuration (Maxim Integrated)

5.6 Pulse (Heartrate) Sensor

One of the prominent factors in a soldier's vital signs is heartrate. So calculating the rate precisely and any instant fulfils commander's desire and is beneficial to the complete team in different ways.

The sensor is directly attached with soldier's wrist. It has a component that can convert heart pulses into electrical signals. Reducing unwanted noise and amplifying the electrical pulses is done after the conversion. A real-time analog voltage ranging between 0-5 V is received at the output of the sensor.

The microcontroller receives the output of the sensor component. The analog signal will then go through analog to digital conversion. The converted signal is then processed through the microcontroller configuration, heart rate (BPM) is calculated then communicated to the output pin of the microcontroller along with a contrasting result with 100 beats per minute. The output of the microcontrollers must be in synchronization with the input requisites of the display unit.

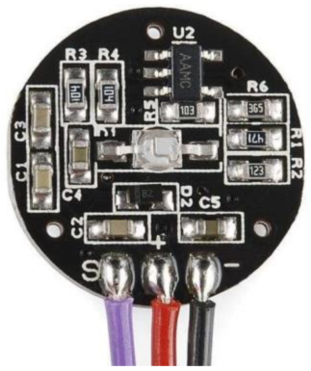


Figure 5.9 - Pulse Sensor (Back) (Spark Fun)

Figure 5.10 - Pulse Sensor (Front) (Spark Fun)

CONCLUSION

From the above mentioned details and the design of the venture, a deduction can be made that every soldier acts as a sensor thus making him a transmitter of live situation to the commander. This enables the commander to make decisions based on the data received from every soldier, which is received via wireless access point. All the components form part of the system therefore making it easy to keep a track of it at all times. Live situation awareness is achieved due to these functions. Further progress can be made by choosing better devices that can make the system more efficient, durable and

proficient. The routing can always be improved as the scope is immense. The complete working of the system can be enhanced because the scope and evolution of technology make it so. The project was supposed to be of less cost as it is a very pertinent point in any technological initiative therefore; tradeoffs were made keeping this very point in mind. The resultant provides a cost effective solution to the Army which has huge scope of improvement and evolution due to the rapid pace of scientific progressions.

FUTURE TECHNOLOGIES

The induction of the ACMS in the Army is a new facet in meeting the future soldier requirements of the Army as well as making it competitive with different other technologically advanced adversaries. Scientific advancements are never ending phenomena; therefore the need of the hour is to identify the potential technologies that can improve the system which at the same time meet all the other limitations which encapsulate the whole process. Therefore, below are mentioned some of the advancements that can be inculcated in the future into the ACMS after weighing their pros and cons and which can evolve the system into becoming more efficient and in tandem with the ongoing developments:

a) **Wearable Technologies** - There are different new gadgets that can be inculcated into the system such as smart watches and glasses. This is an area which can be tapped into to maximize the potential of this endeavor. Figure 4 points some potential areas which can be treaded into by using wearable technologies, developments and advancements.

b) **Simultaneous Localization and Mapping (SLAM)** –This advancement allows a soldier to map an area which has not been mapped already. This is applicable specifically to SCW, in which internal mapping of any building or an apartment might be of extreme help in fulfillment of the objective. As the person is passing through an unknown environment, its map can be developed for use by all by using a SLAM enabled unmanned ground vehicle. This technology is still at the very basic steps of its evolution but its integration in future can be used to exciting outcomes.

c) **Voice and Gesture Recognition Applications** - Embedding this technology into the ACMS can provide the soldier on the ground with the capability to take control of sensor operated devices. Moreover, the smartphone or the tablet which is being used can be told to carry out certain actions thus stealing time and leading to quicker and stealthier reactions.

d) **Wireless Charging** - This technology can be used to provide comfort to the process of recharging of devices. Now instead of manually plugging and unplugging the devices, the soldiers being in close proximity to hot-spots for wireless charging can recharge their devices. This will save quite a bit of hassle whereas special vehicles can be used as hot-spots.

e) **Energy Harvesting** - Energy is very much a sticking point in this system. The soldier while carrying the burden of all the gear he already has will have to carry extra devices thus increasing the weight to be carried which can limit his mobility. Now instead of carrying batteries such techniques can be used in which energy is yielded from other facets. One such example is biochemical harvesting in which power is produced by movements of the body the user.

BIBLOGRAPHY

[1] *Hik Vision*. Retrieved from Hik Vision Web site: https://us.hikvision.com/en/products/cameras/network-camera/value-series/specialty/pin-hole/10-mp-cmos-wdr-mini-network-camera-ds-2cd2d14wd

[2] *Maxim Integrated*.. Retrieved from https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[3] *Micro Robotics*. Retrieved from www.robotics.org: https://www.robotics.org.za/ESP-01

[4] *Module 143*. Retrieved from https://www.module143.com/arduino/arduino-uno-r3-atmega328p-microcontroller-board

[5] *Shop Delta*.. Retrieved from www.shopdelta.eu: https://shopdelta.eu/access-point-tl-wa5210g-2-4-ghz_l2_p5604.html

[6] *Spark Fun*. Retrieved from https://www.sparkfun.com/products/11574

[7] *TP Link*.. Retrieved from www.tp-ink.com: https://www.tp-link.com/lk/business-networking/outdoor-radio/tl-wa5210g/

[8] GEN George Casey (2010). Future Solider 2030 Initiative. U.S Army Natick Solider Rdge Center Wearable, *Computers Research* 2010.

[9] Smith, R. R. (1980). The contract net protocol: High-level communication and control in a distributed problem solver *IEEE Trans. Comput* 29 (1) 1104-1113.

Appendix A

HEALTH MONITORING MODULE (Arduino Code)

```
#include <OneWire.h>                                   //loading libraries
#include <DallasTemperature.h>
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>

#define DEBUG true
#define ONE_WIRE_BUS 9
#define SSID "AlphaAP1"                       //Setting Front AP SSID
#define PASS "alphaap1"                       //Setting Front AP Password

OneWire oneWire(9);                                   //Initializing temperature sensor
DallasTemperature sensors(&oneWire);

LiquidCrystal lcd(7,6,5,4,3,2);
SoftwareSerial Serial1(13,12);                        //setting Tx and Rx pins

int connectionId;
float temp;
float temp2;
float temp3;
double alpha=0.75;
int period=20;
double refresh=0.0;

void setup()
{   pinMode(A4,INPUT);
    sensors.begin();
    lcd.begin(16, 2);
    lcd.print("     ACMS      ");
    lcd.setCursor(0,1);
    lcd.print("  INITIALIZING.   ");
    delay(1000);

  Serial.begin(9600);                                 //For Serial monitor
  Serial1.begin(115200);                      //ESP Baud rate

  sendData("AT+RST\r\n",2000,DEBUG);          // reset module
  sendData("AT+CWMODE=1\r\n",1000,DEBUG);    // configure as access point
  String cmd="AT+CWJAP=\"";
  cmd+=SSID;
  cmd+="\",\"";
  cmd+=PASS;
  cmd+="\"";
  Serial.println(cmd);
  Serial1.println(cmd);
  delay(5000);

  lcd.setCursor(0,0);
  lcd.print("   CONNECTING     ");
  lcd.setCursor(0,1);
  lcd.print("----------------");
  delay(5000);

lcd.setCursor(0,0);
  lcd.print("   VERIFYING    ");
  lcd.setCursor(0,1);
  lcd.print("  CREDENTIALS    ");
  delay(5000);

  lcd.setCursor(0,0);
  lcd.print("     SYSTEM    ");
  lcd.setCursor(0,1);
  lcd.print(" ACCESS GRANTED ");
  delay(5000);

  lcd.setCursor(0,0);
  lcd.print("--- CHECKING ---");
  lcd.setCursor(0,1);
  lcd.print(" SENSOR MODULE  ");
  delay(2000);

  sendData("AT+CIFSR\r\n",1000,DEBUG);                    // get ip address
  delay(1000);
  sendData("AT+CIPMUX=1\r\n",1000,DEBUG);                 //for multiple connections
  sendData("AT+CIPSERVER=1,80\r\n",1000,DEBUG);            // turn on server on port 80
}


void loop()
{
////////////// CHECKING TEMPERATURE AND HEART RATE//////////////
    lcd.setCursor(0,0);
    lcd.print(" Temp  -  Pulse ");
    lcd.setCursor(0,1);
    lcd.print("      F-     BPM");
    lcd.setCursor(5,1);
    lcd.print((char)223);

    static double oldValue=0;
    static double oldrefresh=0;
    int beat=analogRead(A4);
    double value=alpha*oldValue+(0-alpha)*beat;
    refresh=value-oldValue;
```

```
 lcd.setCursor(9,1);
 lcd.print(beat/10);

 sensors.requestTemperatures();
 temp=((sensors.getTempCByIndex(0) * 9.0) / 5.0 + 32.0);
 if(temp<98)
 { temp2=temp+10;
   if (temp2>100)
   {  temp3=99.8;
    }
 }
 lcd.setCursor(0,1);
 lcd.print(temp3);
 delay(500);

 oldValue=value;
 oldrefresh=refresh;
 delay(period*10);

 if(Serial1.available())
 {
        if(Serial1.find("+IPD,"))
     {
         delay(10);
         connectionId = Serial1.read()-48;
     }
         ////////////////////Sending data to browser/////////////////

         String first = "HTTP/1.1 200 OK\nContent-type:text/html\n\n";
         String two = String(temp3);
         two+="&#x2103";
         String three = String(beat/10);
         three+=" BPM";
          String four=first+"<h1>"+two+"</h1><h2>"+three+"</h2><br>/n";
         espsend(four);

         String closeCommand = "AT+CIPCLOSE=";
            closeCommand+=connectionId; // append connection id
         closeCommand+="\r\n";
         sendData(closeCommand,3000,DEBUG);

          lcd.setCursor(0,0);
          lcd.print(" SERVER UPDATED  ");
          lcd.setCursor(0,1);
          lcd.print("  SUCCESSFULLY  ");
          delay(500);
   }
}
```

```
 ////////////////////////////Sends data from ESP to webpage////////////////////////

void espsend(String d)
       {
          String cipSend = " AT+CIPSEND=";
          cipSend += connectionId;
          cipSend += ",";
          cipSend +=d.length();
          cipSend +="\r\n";
          sendData(cipSend,1000,DEBUG);
          sendData(d,1000,DEBUG);
       }

/////////////gets the data from esp and displays in serial monitor/////////////////////
String sendData(String command, const int timeout, boolean debug)
          {
             String response = "";
             Serial1.print(command);
             Serial.print(command);
             long int time = millis();
             while( (time+timeout) > millis())
             {
                while(Serial1.available())
                  {  //Serial.print(command);
                     //Serial.print("aval");
                     char c = Serial1.read();               // read the next character.
                     response+=c;
                  }
             }

             if(debug)
                 {
                 Serial.print(response);                    //displays the esp response messages
                 }
             return response;
          }
```

Appendix B

ACMS Command and Control Software (Android Based)

Main Activity

```
package com.productions.rebel.acms;
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.Looper;
import android.os.SystemClock;
import android.provider.Settings;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.DefaultItemAnimator;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.LinearSmoothScroller;
import android.support.v7.widget.RecyclerView;
import android.text.format.Formatter;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.VideoView;
import com.esri.arcgisruntime.data.TileCache;
import com.esri.arcgisruntime.geometry.Point;
import com.esri.arcgisruntime.geometry.SpatialReferences;
import com.esri.arcgisruntime.layers.ArcGISTiledLayer;
import com.esri.arcgisruntime.mapping.ArcGISMap;
import com.esri.arcgisruntime.mapping.Basemap;
import com.esri.arcgisruntime.mapping.Viewpoint;
import com.esri.arcgisruntime.mapping.view.Graphic;
import com.esri.arcgisruntime.mapping.view.GraphicsOverlay;
import com.esri.arcgisruntime.mapping.view.LocationDisplay;
import com.esri.arcgisruntime.symbology.SimpleMarkerSymbol;
import com.google.android.gms.common.util.ArrayUtils;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.productions.rebel.acms.DB_Utils.DatabaseHelper;
import com.productions.rebel.acms.DB_Utils.User_Ip_List;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.math.BigInteger;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Locale;
import java.util.Timer;
import java.util.TimerTask;
import static com.google.android.gms.location.LocationServices.getFusedLocationProviderClient;
public class MainActivity extends AppCompatActivity {
   //Permissions
   public static final int MULTIPLE_PERMISSIONS = 10; // code you want.
   public String heart, temp;
   String[] permissions = new String[]{
         Manifest.permission.WRITE_EXTERNAL_STORAGE,
         Manifest.permission.READ_EXTERNAL_STORAGE,
         Manifest.permission.INTERNET,
         Manifest.permission.ACCESS_NETWORK_STATE,
         Manifest.permission.ACCESS_FINE_LOCATION ,
         Manifest.permission.ACCESS_WIFI_STATE};
   double initialZoom = 8.0;
   long timeInMilliseconds = 0L;
   long timeSwapBuff = 0L;
   long updatedTime = 0L;
   VideoView livefeed;
   TextView user_name,
         mission_time,
         system_date, exit;
   String user_lat, user_long;
   String usernameFromIntent = null;
   String operationName;
   RecyclerView recyclerView;
   userDataAdapter mAdapter;
   String arduinoIP, cameraIP, cameraUserName, cameraPassword;
   String videoURL = "";
   LocationManager locationManager;
   Context mContext;
   double latitude, longitude;
```

```
    String loc = "", ip = "192.168.10.1";
    String currentIp;
    //Arcgis Var
    private com.esri.arcgisruntime.mapping.view.MapView mMapView;
    //Timer VARS
    private long startTime = 0L;
    private Handler customHandler = new Handler();
    //Recycler VIew VARS
    private List<userDataHelper> userDataHelperList = new ArrayList<>();
    List<User_Ip_List> user_ip_lists = new ArrayList<>();
    private DatabaseHelper databaseHelper;
    Timer timer;
    TimerTask timerTask;
    Handler handler = new Handler();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        user_name = findViewById(R.id.userName_value);
        mMapView = findViewById(R.id.mapView);
        mission_time = findViewById(R.id.missionTime);
        system_date = findViewById(R.id.date);
        exit = findViewById(R.id.exit);
        livefeed = findViewById(R.id.live_feed);
        recyclerView = findViewById(R.id.recycler_view);
        //Other Users IPs
        databaseHelper = new DatabaseHelper(getApplicationContext());
        user_ip_lists = databaseHelper.getAllUserIp();
        //Setting username via EXTRAS from Login Activity
        Bundle bundle = getIntent().getExtras();
        if (bundle != null) {
            usernameFromIntent = bundle.getString("USER_NAME");
            operationName = bundle.getString("OP_NAME");
            arduinoIP = bundle.getString("arduinoIP");
            cameraIP = bundle.getString("cameraIP");
            cameraUserName = bundle.getString("cameraUserName");
            cameraPassword = bundle.getString("cameraPassword");
        }
        user_name.setText(usernameFromIntent);
        checkPermissions();
        getCurrentIp();
        videoURL = "rtsp://" + cameraUserName + ":" + cameraPassword + "@" + cameraIP +
":554/Streaming/Channels/101/";
        //Set the path of Video or URI
        livefeed.setVideoURI(Uri.parse(videoURL));
        livefeed.requestFocus();
        livefeed.start();
        getCurrentDate(system_date);
        startTime = SystemClock.uptimeMillis();
        customHandler.postDelayed(updateTimerThread, 0);
        mAdapter = new userDataAdapter(userDataHelperList);
        RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext(),
LinearLayoutManager.HORIZONTAL, false);
        recyclerView.setLayoutManager(mLayoutManager);
        recyclerView.setItemAnimator(new DefaultItemAnimator());
        recyclerView.setAdapter(mAdapter);
        prepareUserListData(usernameFromIntent,"","","0","0",videoURL,currentIp);
        for(int i=0; i<user_ip_lists.size(); i++) {
            User_Ip_List currentX = user_ip_lists.get(i);
            prepareUserListData("","","","0","0","",currentX.getIp_address());
        }
        //Arcgis setup
        if (checkPermissions())
        //  permissions  granted.
        {
            getLocation();
            setupOfflineMap();
        }
        exit.setOnClickListener(v -> {
            startActivity(new Intent(getApplicationContext(), LoginActivity.class));
            finish();
        });
        Thread serverThread = new Thread(new Server());
        serverThread.start();
        startTimer();
    }
    //Check Permissions
    private boolean checkPermissions() {
        int result;
        List<String> listPermissionsNeeded = new ArrayList<>();
        for (String p : permissions) {
            result = ContextCompat.checkSelfPermission(getApplicationContext(), p);
            if (result != PackageManager.PERMISSION_GRANTED) {
                listPermissionsNeeded.add(p);
            }
        }
        if (!listPermissionsNeeded.isEmpty()) {
            ActivityCompat.requestPermissions(this, listPermissionsNeeded.toArray(new String[0]),
MULTIPLE_PERMISSIONS);
            return false;
        }
        return true;
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissionsList, @NonNull int[]
grantResults) {
        if (requestCode == MULTIPLE_PERMISSIONS) {
            if (grantResults.length > 0) {
                StringBuilder permissionsDenied = new StringBuilder();
                for (String per : permissionsList) {
                    if (grantResults[0] == PackageManager.PERMISSION_DENIED) {
                        permissionsDenied.append("\n").append(per);
                    }
                }
            } else {
                Toast.makeText(getApplicationContext(), "Permission Denied", Toast.LENGTH_LONG).show();
            }
        }
    }
    private void getLocation() {
        LocationRequest lr = new LocationRequest();
        lr.setInterval(10000);
        lr.setFastestInterval(5000);
        lr.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(getApplicationContext(), "Please Grant Location Permissions",
Toast.LENGTH_LONG).show();
        }
        getFusedLocationProviderClient(this).requestLocationUpdates(lr, new LocationCallback() {
                    @Override
                    public void onLocationResult(LocationResult locationResult) {
                        onLocationChanged(locationResult.getLastLocation());
                    }
                },
                Looper.myLooper());
    }
    public void onLocationChanged(Location location) {
        if (location == null)//  Filtering out null values
            return ;
        latitude = location.getLatitude();
        longitude = location.getLongitude();
        user_lat = String.valueOf(latitude);
        user_long = String.valueOf(longitude);
    }
    //Current Device IP
    public void getCurrentIp(){
        WifiManager wm = (WifiManager) getApplicationContext().getSystemService(WIFI_SERVICE);
        WifiInfo wifiInfo = wm.getConnectionInfo();
        int ipInt = wifiInfo.getIpAddress();
        try {
            currentIp = InetAddress.getByAddress(
                    ByteBuffer.allocate(4).order(ByteOrder.LITTLE_ENDIAN).putInt(ipInt).array())
                    .getHostAddress();
            Log.d("Ip","ip: "+currentIp);
        } catch (UnknownHostException e) {
            e.printStackTrace();
        }
    }
    //TIMER CODE
    private Runnable updateTimerThread = new Runnable() {
        public void run() {
            timeInMilliseconds = SystemClock.uptimeMillis() - startTime;
            updatedTime = timeSwapBuff + timeInMilliseconds;
            int secs = (int) (updatedTime / 1000);
            int mins = secs / 60;
            int hours = mins / 60;
            secs = secs % 60;
            int milliseconds = (int) (updatedTime % 1000);
            mission_time.setText(String.format(Locale.ENGLISH, "%d:%d:%s", hours, mins, secs));
            customHandler.postDelayed(this, 0);
        }
    };
    @Override
    protected void onPause() {
        mMapView.pause();
        super.onPause();
    }
    @Override
    protected void onResume() {
        super.onResume();
        mMapView.resume();
      // isLocationEnabled();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        mMapView.dispose();
    }
    //Arcgis map code
    /**
     * Load local tile cache.
     */
    private void loadTileCache() {
        // use local tile package for the base map
        TileCache rawalpindiTileCache;
        ArcGISMap map;
        if (!Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
            rawalpindiTileCache = new TileCache(Environment.getDataDirectory().getAbsolutePath()+
getString(R.string.bahriatpk));
            ArcGISTiledLayer tiledLayer = new ArcGISTiledLayer(rawalpindiTileCache);
            map = new ArcGISMap(new Basemap(tiledLayer));
            mMapView.setMap(map);
            // create an initial viewpoint with a point and scale
            Point point = new Point(latitude, longitude, SpatialReferences.getWebMercator());
            Viewpoint vp = new Viewpoint(point, 7500);
            // set initial map extent
            map.setInitialViewpoint(vp);
            LocationDisplay locationDisplay = mMapView.getLocationDisplay();
            locationDisplay.setAutoPanMode(LocationDisplay.AutoPanMode.RECENTER);
            locationDisplay.setInitialZoomScale(initialZoom);
            locationDisplay.setShowPingAnimation(true);
            locationDisplay.getLocation();
            locationDisplay.startAsync();
        } else {
            rawalpindiTileCache = new TileCache(Environment.getExternalStorageDirectory().getAbsolutePath()+
getString(R.string.bahriatpk));
            ArcGISTiledLayer tiledLayer = new ArcGISTiledLayer(rawalpindiTileCache);
            map = new ArcGISMap(new Basemap(tiledLayer));
            mMapView.setMap(map);
            // create an initial viewpoint with a point and scale
            Point point = new Point(latitude, longitude, SpatialReferences.getWebMercator());
            Viewpoint vp = new Viewpoint(point, 7500);
            // set initial map extent
            map.setInitialViewpoint(vp);
            LocationDisplay locationDisplay = mMapView.getLocationDisplay();
            locationDisplay.setAutoPanMode(LocationDisplay.AutoPanMode.RECENTER);
            locationDisplay.setInitialZoomScale(initialZoom);
            locationDisplay.setShowPingAnimation(true);
            locationDisplay.getLocation();
            locationDisplay.startAsync();
        }
        for(int i=0; i<userDataHelperList.size(); i++) {
          userDataHelper currenty = userDataHelperList.get(i);
createMarkers(Double.parseDouble(currenty.getUserLng()),Double.parseDouble(currenty.getUserLat()));
        }
    }
    private void createMarkers(double x, double y){
        // create a new graphics overlay and add it to the mapview
        GraphicsOverlay graphicsOverlay = new GraphicsOverlay();
        mMapView.getGraphicsOverlays().add(graphicsOverlay);
        //create a simple marker symbol
        SimpleMarkerSymbol symbol = new SimpleMarkerSymbol(SimpleMarkerSymbol.Style.CIRCLE, Color.RED,
12); //size 12, style of circle
        //add a new graphic with a new point geometry
        Point graphicPoint = new Point(x, y, SpatialReferences.getWebMercator());
        Graphic graphic = new Graphic(graphicPoint, symbol);
        graphicsOverlay.getGraphics().add(graphic);
    }
    private void setupMap() {
        if (mMapView != null) {
          loadTileCache();
        }
    }
    private void setupOfflineMap() {
        if (mMapView != null) {
          setupMap();
        }
    }
    public void prepareUserListData(String name, String heart_rate, String body_temp, String lat, String lng, String
camera_ip,String userIPaddress) {
        userDataHelper userDataHelper = new userDataHelper(name, "", heart_rate, body_temp, lat, lng,
camera_ip,userIPaddress);
        userDataHelperList.add(userDataHelper);
        mAdapter.notifyDataSetChanged();
    }
    //Changing Video Url onClick
    public void changeVideoUrl(){
        livefeed.stopPlayback();
        if (mAdapter.clickedVideoUrl != null) {
          String clickedUrl = mAdapter.clickedVideoUrl;
          livefeed.setVideoURI(Uri.parse(clickedUrl));
          livefeed.requestFocus();
          livefeed.start();
        }
    }
    //Getting Current Date
    public void getCurrentDate(View view) {
        Calendar calendar = Calendar.getInstance();
        SimpleDateFormat mdformat = new SimpleDateFormat("dd MMM yyyy", Locale.getDefault());
        String strDate = "" + mdformat.format(calendar.getTime());
        displayDate(strDate);
    }
    ///Displaying Current Date
    private void displayDate(String num) {
        system_date.setText(num);
    }
    //Retrieve Data
    private class getArdinoData extends AsyncTask<Void, Void, Void> {
        @Override
        protected Void doInBackground(Void... voids) {
            try {
                Document document = Jsoup.connect("http://" + arduinoIP).get();
                temp = document.select("h1").text();
                heart = document.select("h2").text();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return null;
        }
        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute(aVoid);
            int selectedIndex = -1;
            for(int j=0; j<userDataHelperList.size(); j++)
            {
                userDataHelper userData = userDataHelperList.get(j);
                if(currentIp.equals(userData.getUserIPaddress()))
                    selectedIndex = j;
            }
            if(selectedIndex > -1){
                userDataHelper setuserdata = new userDataHelper(usernameFromIntent, "ACTIVE", heart, temp, user_lat,
user_long, videoURL,currentIp);
                userDataHelperList.set(selectedIndex,setuserdata);
                mAdapter.notifyItemChanged(selectedIndex);
            }
            for(int i=0; i<user_ip_lists.size(); i++) {
                BackGroundTask b = new BackGroundTask();
                User_Ip_List currentY = user_ip_lists.get(i);
b.execute(currentY.getIp_address(),currentIp,usernameFromIntent,temp,heart,user_lat,user_long,videoURL);
            }
        }
    }
    public void startTimer() {
        timer = new Timer();
        initializeTimerTask();
        timer.schedule(timerTask, 0, 5000); //
    }

    public void initializeTimerTask() {
        timerTask = new TimerTask() {
            public void run() {
                handler.post(new Runnable() {
                    public void run() {
                        new getArdinoData().execute();
                    }
                });
            }
        };
    }

    //CLient Server Code
       class Server implements Runnable{
        ServerSocket serverSocket;
        Socket socket;
        ObjectInputStream objectInputStream;
        String[] dataRecieved = new String[8];
        Handler handler = new Handler();
        String
receivedIP="",receivedUserName="",receivedTemp="",receivedHeart="",receivedLat="",receivedLong="",received
VideoUrl="";
        @Override
        public void run() {
            try
            {
                serverSocket = new ServerSocket(9700);
                handler.post(new Runnable() {
                    @Override
                    public void run() {
                    }
                });
                while (true)
                {
                    socket = serverSocket.accept();
                    objectInputStream = new ObjectInputStream(socket.getInputStream());
                    dataRecieved = (String[]) objectInputStream.readObject();
                    handler.post(new Runnable() {
                        @Override
                        public void run() {
                            receivedIP = dataRecieved[1];
                            receivedUserName  = dataRecieved[2];
                            receivedTemp = dataRecieved[3];
                            receivedHeart  = dataRecieved[4];
                            receivedLat = dataRecieved[5];
                            receivedLong  = dataRecieved[6];
                            receivedVideoUrl = dataRecieved[7];
                            int selectedIndex = -1;
                            for(int j=0; j<userDataHelperList.size(); j++)
                            {
                                userDataHelper userData = userDataHelperList.get(j);
                                if(receivedIP.equals(userData.getUserIPaddress()))
                                    selectedIndex = j;
                            }
                            if(selectedIndex > -1){
                                userDataHelper setuserdata = new userDataHelper(receivedUserName, "ACTIVE",
receivedHeart, receivedTemp, receivedLat, receivedLong, receivedVideoUrl,receivedIP);
                                userDataHelperList.set(selectedIndex,setuserdata);
                                mAdapter.notifyItemChanged(selectedIndex);
                            }
                            Log.d("Names","Data at 0 = " + dataRecieved[0] + "  Data at 1 = " + dataRecieved[1]);
                        }
                    });
                }
            }catch (IOException e)
            {
                e.printStackTrace();
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
        }
    }
    @SuppressLint("StaticFieldLeak")
    class BackGroundTask extends AsyncTask<String,Void,String> {
        Socket socket;
        ObjectOutputStream objectOutputStream;
        String ipAddress;
        String[] data = new String[8];
        @Override
        protected String doInBackground(String... strings) {
            ipAddress = strings[0];
            data[0] = strings[0];
            data[1] = strings[1];
            data[2] = strings[2];
            data[3] = strings[3];
            data[4] = strings[4];
            data[5] = strings[5];
            data[6] = strings[6];
            data[7] = strings[7];
            try{
                socket = new Socket(ipAddress,9700);
                objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
                objectOutputStream.writeObject(data);
                objectOutputStream.close();
                socket.close();
            }catch (Exception e)
            {
                e.printStackTrace();
            }
            return null;
        }
    }
}
```

Login Activity

```
package com.productions.rebel.acms;

import android.content.Intent;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.PopupMenu;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.Toast;
import com.productions.rebel.acms.DB_Utils.DatabaseHelper;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
public class LoginActivity extends AppCompatActivity {
    EditText userName, appPassword, operationName;
    Button loginBtn;
    ImageView applogo;
    ImageButton menu_btn;
    private DatabaseHelper databaseHelper;
    String opName = "";
    String arduinoIP="",cameraIP="",cameraUserName="",cameraPassword="";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        //Data from Settings
        Bundle settingsBundle = getIntent().getExtras();
        if (settingsBundle!=null){
            //Settings Data
            arduinoIP = settingsBundle.getString("arduinoIP");
            cameraIP = settingsBundle.getString("cameraIP");
            cameraUserName = settingsBundle.getString("cameraUserName");
            cameraPassword = settingsBundle.getString("cameraPassword");
        }
        userName = findViewById(R.id.userName);
        appPassword = findViewById(R.id.appPassword);
        operationName = findViewById(R.id.operationName);
        applogo = findViewById(R.id.applogo);
        databaseHelper = new DatabaseHelper(getApplicationContext());
        loginBtn = findViewById(R.id.loginBtn);
        loginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                verifyFromSQLite();
            }
        });
        menu_btn = findViewById(R.id.menu_btn);
        menu_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Creating the instance of PopupMenu
                PopupMenu popup = new PopupMenu(LoginActivity.this, menu_btn);
                //Inflating the Popup using xml file
                popup.getMenuInflater().inflate(R.menu.popup_menu, popup.getMenu());
                //registering popup with OnMenuItemClickListener
                popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                    public boolean onMenuItemClick(MenuItem item) {
                        // Handle item selection
                        switch (item.getItemId()) {
                            case R.id.register_menu:
                                startActivity(new Intent(getApplicationContext(), RegisterActivity.class));
                                break;
                            case R.id.setting_menu:
                                Intent loginToSettings = new Intent(getApplicationContext(), Settings_Class.class);
                                Bundle tosettingsBundle = new Bundle();
                                tosettingsBundle.putString("arduinoIP", arduinoIP);
                                tosettingsBundle.putString("cameraIP", cameraIP);
                                tosettingsBundle.putString("cameraUserName", cameraUserName);
                                tosettingsBundle.putString("cameraPassword", cameraPassword);
                                loginToSettings.putExtras(tosettingsBundle);
                                startActivity(loginToSettings);
                                break;
                            case R.id.history_menu:
                                startActivity(new Intent(getApplicationContext(), History.class));
                                break;
                        }
                        return true;
                    }
                });

                popup.show();//showing popup menu
            }
        });
    }
    /**
     * This method is to validate the input text fields and verify login credentials from SQLite
     */
    private void verifyFromSQLite() {
        if (databaseHelper.checkUser(userName.getText().toString().trim()
                , appPassword.getText().toString().trim()) &&
                !arduinoIP.equals("") && !cameraIP.equals("") && !cameraUserName.equals("") &&
    !cameraPassword.equals("")) {
            Intent mainIntent = new Intent(getApplicationContext(), MainActivity.class);
            opName = operationName.getText().toString();
            Bundle bundle = new Bundle();
            bundle.putString("USER_NAME", userName.getText().toString().trim());
            bundle.putString("OP_NAME", opName);
            bundle.putString("arduinoIP", arduinoIP);
            bundle.putString("cameraIP", cameraIP);
            bundle.putString("cameraUserName", cameraUserName);
            bundle.putString("cameraPassword", cameraPassword);
            emptyInputEditText();
            mainIntent.putExtras(bundle);
            startActivity(mainIntent);
            finish();
        } else {
            // Toast to show success message that record is wrong
            Toast.makeText(getApplicationContext(), "Problem with UserName or Password or Settings",
    Toast.LENGTH_LONG).show();
        }
    }

    /**
     * This method is to empty all input edit text
     */
    private void emptyInputEditText() {
        userName.setText(null);
        appPassword.setText(null);
    }
    }
```

--
Register Activity

```
package com.productions.rebel.acms;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.productions.rebel.acms.DB_Utils.DatabaseHelper;
import com.productions.rebel.acms.DB_Utils.User;
public class RegisterActivity extends AppCompatActivity {
    EditText enter_username,enter_password,enter_confirmpassword,enter_passcode;
    Button register;
    private DatabaseHelper databaseHelper;
    private User user;
    private  static String code = "1234";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        databaseHelper = new DatabaseHelper(getApplicationContext());
        user = new User();
        enter_username = findViewById(R.id.enter_username);
        enter_password = findViewById(R.id.enter_password);
        enter_confirmpassword = findViewById(R.id.enter_confirmpassword);
        enter_passcode = findViewById(R.id.enter_passcode);
        register = findViewById(R.id.register);
        register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (enter_passcode.getText().toString().equals(code)) {
                    postDataToSQLite();
                    startActivity(new Intent(getApplicationContext(), LoginActivity.class));
                    finish();
                }else {
                    Toast.makeText(getApplicationContext(),"Please enter your assigned
code",Toast.LENGTH_LONG).show();
                }
            }
        });
    }
    /**
     * This method is to validate the input text fields and post data to SQLite
     */
    private void postDataToSQLite()
        if (!databaseHelper.checkUser(enter_username.getText().toString().trim())) {
            user.setName(enter_username.getText().toString().trim());
            user.setPassword(enter_password.getText().toString().trim());
            databaseHelper.addUser(user);
            // Snack Bar to show success message that record saved successfully
            Toast.makeText(getApplicationContext(),"Success",Toast.LENGTH_LONG).show();
            emptyInputEditText();
        } else {
            // Snack Bar to show error message that record already exists
            Toast.makeText(getApplicationContext(),"Error",Toast.LENGTH_LONG).show();
        }
    }
    /**
     * This method is to empty all input edit text
     */
    private void emptyInputEditText() {
        enter_username.setText(null);
        enter_password.setText(null);
        enter_confirmpassword.setText(null);
    }
}
```

--

Settings Activity

--

```
package com.productions.rebel.acms;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.DefaultItemAnimator;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.productions.rebel.acms.DB_Utils.DatabaseHelper;
import com.productions.rebel.acms.DB_Utils.User;
import com.productions.rebel.acms.DB_Utils.User_Ip_List;
import java.util.ArrayList;
import java.util.List;
public class Settings_Class extends AppCompatActivity {
    EditText arduinoIPSettings,cameraIPSettings,camerUserNameSettings,cameraPasswordSettings;
    Button saveSettings;
    String arduinoIP,cameraIP,cameraUserName,cameraPassword;
    //UserList vars
    RecyclerView settingsRecyclerView;
    usersListHelper usersListHelper;
    usersListAdapter usersListAdapter;
    ArrayList<usersListHelper> usersListHelperArrayList = new ArrayList<>();
    Button addBtn,deleteBtn;
    EditText ipEntryRecyclerView;
    String enteredUserIp = "";
    private DatabaseHelper databaseHelper;
    private User_Ip_List useriplist;
    public List<User_Ip_List> arrayUserIp = new ArrayList<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings__class);
        arduinoIPSettings = findViewById(R.id.arduinoIPSettings);
        cameraIPSettings = findViewById(R.id.cameraIPSettings);
        camerUserNameSettings = findViewById(R.id.cameraUserNameSettings);
        cameraPasswordSettings = findViewById(R.id.cameraPasswordSettings);
        saveSettings = findViewById(R.id.saveSettings);
        databaseHelper = new DatabaseHelper(getApplicationContext());
        useriplist = new User_Ip_List();
        Bundle settingsBundle = getIntent().getExtras();
        if (settingsBundle!=null){
            //Settings Data
            arduinoIP = settingsBundle.getString("arduinoIP");
            arduinoIPSettings.setText(arduinoIP);
            cameraIP = settingsBundle.getString("cameraIP");
            cameraIPSettings.setText(cameraIP);
            cameraUserName = settingsBundle.getString("cameraUserName");
            camerUserNameSettings.setText(cameraUserName);
            cameraPassword = settingsBundle.getString("cameraPassword");
            cameraPasswordSettings.setText(cameraPassword);
        }
        //Setup recyclerView and Adapter
        settingsRecyclerView = findViewById(R.id.users_list_recyclerview);
        RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getApplicationContext());
        settingsRecyclerView.setLayoutManager(mLayoutManager);
        settingsRecyclerView.setItemAnimator(new DefaultItemAnimator());
//        usersListAdapter = new usersListAdapter(getApplicationContext(), usersListHelperArrayList);
//        settingsRecyclerView.setAdapter(usersListAdapter);
        ipEntryRecyclerView = findViewById(R.id.user_ip_entry_recyclerview);
        addBtn = findViewById(R.id.add_ip_to_usersList);
        addBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (ipEntryRecyclerView != null) {
                    enteredUserIp = ipEntryRecyclerView.getText().toString();
                    Toast.makeText(getApplicationContext(), "ip: " + enteredUserIp, Toast.LENGTH_SHORT).show();
                    addUserIPtoList(enteredUserIp);
                    ipEntryRecyclerView.setText("");
                } else {
                    Toast.makeText(getApplicationContext(), "Please Enter IP", Toast.LENGTH_SHORT).show();
                }
            }
        });
        deleteBtn = findViewById(R.id.delete_ip_from_usersList);
        deleteBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               int clickedId= usersListAdapter.id_ip;
               if(clickedId > -1)
               {
                   useriplist.setIp_id(clickedId);
                   databaseHelper.deleteUserIp(useriplist);
                   if(arrayUserIp != null)
                   {
                       arrayUserIp.clear();
                   }
                   getUserIpList();
               }
            }
        });
        saveSettings.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                    arduinoIP = arduinoIPSettings.getText().toString();
                    cameraIP = cameraIPSettings.getText().toString();
                    cameraUserName = camerUserNameSettings.getText().toString();
                    cameraPassword = cameraPasswordSettings.getText().toString();

                if (arduinoIP.equals("") || cameraIP.equals("") || cameraUserName.equals("") ||
cameraPassword.equals("")){
                    Toast.makeText(getApplicationContext(),"Please Enter Valid Values",Toast.LENGTH_LONG).show();
                }else {
                    Intent settingsToLogin = new Intent(getApplicationContext(),LoginActivity.class);
                    Bundle settingsBundle = new Bundle();
                    settingsBundle.putString("arduinoIP", arduinoIP);
                    settingsBundle.putString("cameraIP", cameraIP);
                    settingsBundle.putString("cameraUserName", cameraUserName);
                    settingsBundle.putString("cameraPassword", cameraPassword);
                    settingsToLogin.putExtras(settingsBundle);
                    startActivity(settingsToLogin);
                    finish();
                }
            }
        });
        if(arrayUserIp != null)
        {
            arrayUserIp.clear();
        }
        getUserIpList();
    }

    public void addUserIPtoList(String entered_user_ip){
        useriplist.setIp_address(entered_user_ip);
        databaseHelper.addUserIp(useriplist);
        if(arrayUserIp != null)
        {
            arrayUserIp.clear();
        }
        getUserIpList();
//        usersListHelper  = new usersListHelper(entered_user_ip);
//        usersListHelperArrayList.add(usersListHelper);
//        usersListAdapter.notifyDataSetChanged();
    }
    private void getUserIpList(){
        arrayUserIp = databaseHelper.getAllUserIp();
        usersListAdapter = new usersListAdapter(getApplicationContext(), arrayUserIp);
        settingsRecyclerView.setAdapter(usersListAdapter);
    }
}
```

--
userDataAdapter Class
--

```
package com.productions.rebel.acms;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.RelativeLayout;
import android.widget.TextView;
import java.util.List;
public class userDataAdapter extends RecyclerView.Adapter<userDataAdapter.MyViewHolder> {
    private List<userDataHelper> userDataHelperList;
    public String clickedVideoUrl;
    public class MyViewHolder extends RecyclerView.ViewHolder
    {
        public TextView userName, statusValue, heartRateValue,bodyTempValue,userLat,userLong;
        public RelativeLayout dataLayout;
        public MyViewHolder(View view) {
            super(view);
            userName = view.findViewById(R.id.listUserName);
            statusValue = view.findViewById(R.id.status_value);
            heartRateValue = view.findViewById(R.id.heartrate_value);
            bodyTempValue = view.findViewById(R.id.bodytemp_value);
            userLat = view.findViewById(R.id.lat_value);
            userLong = view.findViewById(R.id.long_value);
            dataLayout = view.findViewById(R.id.dataLayout);
        }
    }
    public userDataAdapter(List<userDataHelper> userDataHelperList) {
        this.userDataHelperList = userDataHelperList;
    }
    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.userdata_list, parent, false);
        return new MyViewHolder(itemView);
    }
    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position) {
        userDataHelper userDataHelper = userDataHelperList.get(position);
        holder.userName.setText(userDataHelper.getUserName());
        holder.statusValue.setText(userDataHelper.getUserStatus());
        holder.heartRateValue.setText(userDataHelper.getUserHeartRate());
        holder.bodyTempValue.setText(userDataHelper.getUserBodyTemp());
        holder.userLat.setText(userDataHelper.getUserLat());
        holder.userLong.setText(userDataHelper.getUserLng());
        holder.dataLayout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                clickedVideoUrl = userDataHelperList.get(holder.getLayoutPosition()).getCameraIP();
            }
        });
    }
    @Override
    public int getItemCount() {
        return userDataHelperList.size();
    }
}
```

--

userDataHelper Class

```
package com.productions.rebel.acms;
public class userDataHelper {
    private String userName;
    private String userStatus;
    private String userHeartRate;
    private String userBodyTemp;
    private String userLat;
    private String userLng;
    private String cameraIP;
    private String userIPaddress;
    public userDataHelper(String userName, String userStatus, String userHeartRate, String userBodyTemp, String
userLat, String userLng, String cameraIP,String userIPaddress) {
        this.userName = userName;
        this.userStatus = userStatus;
        this.userHeartRate = userHeartRate;
        this.userBodyTemp = userBodyTemp;
        this.userLat = userLat;
        this.userLng = userLng;
        this.cameraIP = cameraIP;
        this.userIPaddress = userIPaddress;
    }
    public userDataHelper() {
    }
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public String getUserStatus() {
        return userStatus;
    }
    public void setUserStatus(String userStatus) {
        this.userStatus = userStatus;
    }
    public String getUserHeartRate() {
        return userHeartRate;
    }
    public void setUserHeartRate(String userHeartRate) {
        this.userHeartRate = userHeartRate;
    }
    public String getUserBodyTemp() {
        return userBodyTemp;
    }
    public void setUserBodyTemp(String userBodyTemp) {
        this.userBodyTemp = userBodyTemp;
    }
    public String getUserLat() {
        return userLat;
    }
    public void setUserLat(String userLat) {
        this.userLat = userLat;
    }
    public String getUserLng() {
        return userLng;
    }
    public void setUserLng(String userLng) {
        this.userLng = userLng;
    }
    public String getCameraIP() {
        return cameraIP;
    }
    public void setCameraIP(String cameraIP) {
        this.cameraIP = cameraIP;
    }
    public String getUserIPaddress() {
        return userIPaddress;
    }
    public void setUserIPaddress(String userIPaddress) {
        this.userIPaddress = userIPaddress;
    }
}
```

--
userListAdapter Class
--

```
package com.productions.rebel.acms;
import android.content.Context;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.productions.rebel.acms.DB_Utils.User_Ip_List;
import java.util.List;
public class usersListAdapter extends RecyclerView.Adapter<usersListAdapter.MyViewHolder> {
    private List<User_Ip_List> usersListHelperList;
    private User_Ip_List user_ip_list;
    private Context context;
    int id_ip = -1;
    public usersListAdapter(Context context, List<User_Ip_List> usersListHelperList) {
        this.usersListHelperList = usersListHelperList;
        this.context = context;
    }
    @NonNull
    @Override
    public usersListAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
        View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.users_list_item, viewGroup, false);
        return new MyViewHolder(view);
    }
    @Override
    public void onBindViewHolder(@NonNull usersListAdapter.MyViewHolder myViewHolder, int i) {
        user_ip_list = usersListHelperList.get(i);
        myViewHolder.userIPinList.setText(user_ip_list.getIp_address());
        myViewHolder.userIPinList.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                id_ip = usersListHelperList.get(myViewHolder.getLayoutPosition()).getIp_id();
            }
        });
    }
    @Override
    public int getItemCount() {
        return usersListHelperList.size();
    }
    public class MyViewHolder extends RecyclerView.ViewHolder {
        TextView userIPinList;
        public MyViewHolder(View view) {
            super(view);
            userIPinList = view.findViewById(R.id.users_ip_recyclerview);
        }
    }
}
```

--
userListHelper Class
--

```
package com.productions.rebel.acms;
public class usersListHelper {
    private String user_ip;
    public usersListHelper(String user_ip) {
        this.user_ip = user_ip;
    }
    public String getUser_ip() {
        return user_ip;
    }
    public void setUser_ip(String user_ip) {
        this.user_ip = user_ip;
    }
}
```

--
DatabaseHelper Class
--

```
package com.productions.rebel.acms.DB_Utils;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import java.util.ArrayList;
import java.util.List;
public class DatabaseHelper extends SQLiteOpenHelper {
    // Database Version
    private static final int DATABASE_VERSION = 1;
    // Database Name
    private static final String DATABASE_NAME = "UserManager.db";
    // User table name
    private static final String TABLE_USER = "user";
    private static final String TABLE_OPERATIONS = "operations";
    private static final String TABLE_IPLIST = "iplist";
    // User Table Columns names
```

```
private static final String COLUMN_USER_ID = "user_id";
private static final String COLUMN_USER_NAME = "user_name";
private static final String COLUMN_USER_PASSWORD = "user_password";
//Operations Table Columns names
private static final String COLUMN_OPERATION_ID = "operation_id";
private static final String COLUMN_OPERATION_NAME = "operation_name";
private static final String COLUMN_OPERATION_TIME = "operation_time";
private static final String COLUMN_RUNNING_TIME = "running_time";
private static final String COLUMN_HEART_RATE = "heart_rate";
private static final String COLUMN_TEMPRATURE = "temprature";
private static final String COLUMN_LAT = "lat";
private static final String COLUMN_LONG = "long";
private static final String COLUMN_F_USER_ID = "f_user_id";
// IpList Table Columns names
private static final String COLUMN_IP_ID = "ip_id";
private static final String COLUMN_IP_ADDRESS = "ip_address";
// create table sql query
private String CREATE_USER_TABLE = "CREATE TABLE " + TABLE_USER
        + "("
        + COLUMN_USER_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_USER_NAME + " TEXT,"
        + COLUMN_USER_PASSWORD + " TEXT"
        + ")";
private String CREATE_OPERATIONS_TABLE = "CREATE TABLE " + TABLE_OPERATIONS
        + "("
        + COLUMN_OPERATION_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_OPERATION_NAME + " TEXT,"
        + COLUMN_OPERATION_TIME + " TEXT,"
        + COLUMN_RUNNING_TIME + " TEXT,"
        + COLUMN_HEART_RATE + " TEXT,"
        + COLUMN_TEMPRATURE + " TEXT,"
        + COLUMN_LAT + " TEXT,"
        + COLUMN_LONG + " TEXT,"
        + COLUMN_F_USER_ID + " INTEGER"
        +")";
private String CREATE_IPLIST_TABLE = "CREATE TABLE " + TABLE_IPLIST
        + "("
        + COLUMN_IP_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + COLUMN_IP_ADDRESS + " TEXT"
        + ")";
// drop table sql query
private String DROP_USER_TABLE = "DROP TABLE IF EXISTS " + TABLE_USER;
private String DROP_OPERATIONS_TABLE = "DROP TABLE IF EXISTS " + TABLE_OPERATIONS;
private String DROP_IPLIST_TABLE = "DROP TABLE IF EXISTS " + TABLE_IPLIST;
/**
 * Constructor
 *
 * @param context
 */
public DatabaseHelper(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
}
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_USER_TABLE);
    db.execSQL(CREATE_OPERATIONS_TABLE);
    db.execSQL(CREATE_IPLIST_TABLE);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    //Drop User Table if exist
    db.execSQL(DROP_USER_TABLE);
    db.execSQL(DROP_OPERATIONS_TABLE);
    db.execSQL(DROP_IPLIST_TABLE);
    // Create tables again
    onCreate(db);
}
/**
 * This method is to create user record
 *
 * @param user
 */
public void addUser(User user) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_USER_NAME, user.getName());
    values.put(COLUMN_USER_PASSWORD, user.getPassword());
    // Inserting Row
    db.insert(TABLE_USER, null, values);
    db.close();
}
/**
 * This method is to fetch all user and return the list of user records
 *
 * @return list
 */
public List<User> getAllUser() {
    // array of columns to fetch
    String[] columns = {
            COLUMN_USER_ID,
            COLUMN_USER_NAME,
            COLUMN_USER_PASSWORD
    };
    // sorting orders
    String sortOrder =
            COLUMN_USER_NAME + " ASC";
    List<User> userList = new ArrayList<User>();
    SQLiteDatabase db = this.getReadableDatabase();
    // query the user table
    Cursor cursor = db.query(TABLE_USER, //Table to query
            columns,    //columns to return
            null,        //columns for the WHERE clause
            null,        //The values for the WHERE clause
            null,       //group the rows
            null,       //filter by row groups
            sortOrder); //The sort order
    // Traversing through all rows and adding to list
    if (cursor.moveToFirst()) {
        do {
            User user = new User();
user.setId(Integer.parseInt(cursor.getString(cursor.getColumnIndex(COLUMN_USER_ID))));
user.setName(cursor.getString(cursor.getColumnIndex(COLUMN_USER_NAME)));
user.setPassword(cursor.getString(cursor.getColumnIndex(COLUMN_USER_PASSWORD)));
                // Adding user record to list
                userList.add(user);
            } while (cursor.moveToNext());
        }
        cursor.close();
        db.close();
        // return user list
        return userList;
    }
    /**
     * This method to update user record
     *
     * @param user
     */
    public void updateUser(User user) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_USER_NAME, user.getName());
        values.put(COLUMN_USER_PASSWORD, user.getPassword());
        // updating row
        db.update(TABLE_USER, values, COLUMN_USER_ID + " = ?",
                new String[]{String.valueOf(user.getId())});
        db.close();
    }
    /**
     * This method is to delete user record
     *
     * @param user
     */
    public void deleteUser(User user) {
        SQLiteDatabase db = this.getWritableDatabase();
        // delete user record by id
        db.delete(TABLE_USER, COLUMN_USER_ID + " = ?",
                new String[]{String.valueOf(user.getId())});
        db.close();
    }
    /**
     * This method to check user exist or not
     *
     * @param user_name
     * @return true/false
     */
    public boolean checkUser(String user_name) {
        // array of columns to fetch
        String[] columns = {
                COLUMN_USER_ID
        };
        SQLiteDatabase db = this.getReadableDatabase();
        // selection criteria
        String selection = COLUMN_USER_NAME + " = ?";
        // selection argument
        String[] selectionArgs = {user_name};
        // query user table with condition
    Cursor cursor = db.query(TABLE_USER, //Table to query
            columns,                    //columns to return
            selection,                  //columns for the WHERE clause
            selectionArgs,              //The values for the WHERE clause
            null,                       //group the rows
            null,                       //filter by row groups
            null);                      //The sort order
    int cursorCount = cursor.getCount();
    cursor.close();
    db.close();
    if (cursorCount > 0) {
        return true;
    }
    return false;
}
/**
 * This method to check user exist or not
 *
 * @param user_name
 * @param password
 * @return true/false
 */
public boolean checkUser(String user_name, String password) {
    // array of columns to fetch
    String[] columns = {
            COLUMN_USER_ID
    };
    SQLiteDatabase db = this.getReadableDatabase();
    // selection criteria
    String selection = COLUMN_USER_NAME + " = ?" + " AND " + COLUMN_USER_PASSWORD + " = ?";
    // selection arguments
    String[] selectionArgs = {user_name, password};
    // query user table with conditions
    Cursor cursor = db.query(TABLE_USER, //Table to query
            columns,                    //columns to return
            selection,                  //columns for the WHERE clause
            selectionArgs,              //The values for the WHERE clause
            null,                       //group the rows
            null,                       //filter by row groups
            null);                      //The sort order
    int cursorCount = cursor.getCount();
    cursor.close();
    db.close();
    return cursorCount > 0;
}
public void addUserIp(User_Ip_List userip) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(COLUMN_IP_ADDRESS, userip.getIp_address());
    // Inserting Row
    db.insert(TABLE_IPLIST, null, values);
    db.close();
}
public List<User_Ip_List> getAllUserIp() {
    // array of columns to fetch
    String[] columns = {
```

```
                COLUMN_IP_ID,
                COLUMN_IP_ADDRESS
        };
        // sorting orders
        String sortOrder =
                COLUMN_IP_ID+ " ASC";
        List<User_Ip_List> userIpList = new ArrayList<User_Ip_List>();
        SQLiteDatabase db = this.getReadableDatabase();
        // query the user table
        Cursor cursor = db.query(TABLE_IPLIST, //Table to query
                columns,    //columns to return
                null,        //columns for the WHERE clause
                null,        //The values for the WHERE clause
                null,       //group the rows
                null,       //filter by row groups
                sortOrder); //The sort order
        // Traversing through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                User_Ip_List userip = new User_Ip_List();
userip.setIp_id(Integer.parseInt(cursor.getString(cursor.getColumnIndex(COLUMN_IP_ID))));
userip.setIp_address(cursor.getString(cursor.getColumnIndex(COLUMN_IP_ADDRESS)));
                // Adding user record to list
                userIpList.add(userip);
            } while (cursor.moveToNext());
        }
        cursor.close();
        db.close();
        // return user list
        return userIpList;
    }
    public void deleteUserIp(User_Ip_List userip) {
        SQLiteDatabase db = this.getWritableDatabase();
        // delete user record by id
        db.delete(TABLE_IPLIST, COLUMN_IP_ID + " = ?",
                new String[]{String.valueOf(userip.getIp_id())});
        db.close();
    }
}
```

OperationsHelper Class

```
package com.productions.rebel.acms.DB_Utils;
public class Operations_Helper {
    private int operation_id;
    private String operation_name;
    private String operation_time;
    private String running_time;
    private String heart_rate;
    private String temprature;
    private String lat;
    private String lng;
    private int f_user_id;
    public int getOperation_id() {
        return operation_id;
    }
    public void setOperation_id(int operation_id) {
        this.operation_id = operation_id;
    }
    public String getOperation_name() {
        return operation_name;
    }
    public void setOperation_name(String operation_name) {
        this.operation_name = operation_name;
    }
    public String getOperation_time() {
        return operation_time;
    }
    public void setOperation_time(String operation_time) {
        this.operation_time = operation_time;
    }
    public String getRunning_time() {
        return running_time;
    }
    public void setRunning_time(String running_time) {
        this.running_time = running_time;
    }
    public String getHeart_rate() {
        return heart_rate;
    }
    public void setHeart_rate(String heart_rate) {
        this.heart_rate = heart_rate;
    }
    public String getTemprature() {
        return temprature;
    }
    public void setTemprature(String temprature) {
        this.temprature = temprature;
    }
    public String getLat() {
        return lat;
    }
    public void setLat(String lat) {
        this.lat = lat;
    }
    public String getLng() {
        return lng;
    }
    public void setLng(String lng) {
        this.lng = lng;
    }
    public int getF_user_id() {
        return f_user_id;
    }
    public void setF_user_id(int f_user_id) {
        this.f_user_id = f_user_id;
    }
}
```

--
User Modal Class
--

```
package com.productions.rebel.acms.DB_Utils;
public class User {
   private int id;
   private String name;
   private String password;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getPassword() {
      return password;
   }
   public void setPassword(String password) {
      this.password = password;
   }
}
```

--
User_Ip_List Modal Class
--

```
package com.productions.rebel.acms.DB_Utils;
public class User_Ip_List {
   private int ip_id;
   private String ip_address;
   public int getIp_id() {
      return ip_id;
   }
   public void setIp_id(int ip_id) {
      this.ip_id = ip_id;
   }
   public String getIp_address() {
      return ip_address;
   }
   public void setIp_address(String ip_address) {
      this.ip_address = ip_address;
   }
}
```