ADEL ATEF	3960
RIMON ADEL	4441
BAVLI GEORES	3756
YOUSSEF KISHK	3812

INTRODUCTION

- Real world applications Networks (socket) programming is used in many applications that we use day to day; the most obvious applications might be Facebook and other chatting/social media applications. This is not just it; industry and gaming also use socket programming heavily. The purpose of the multi-player game and the IoT project is to get an idea of how sockets are used in other domains.

- Socket Programming provides the ability to implement real time analytic, instant messaging, binary streaming, and document collaboration. Socket.IO controls the connection transparently. Socket.IO is a custom real time transport protocol. Socket.IO needs the use of socket.IO libraries on both client and server side. It is also used to make real time apps feasible in every browser and mobile devices and its open source too.

- Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors. In Unix, every I/O action is done by writing or reading a file descriptor. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.

PEER DISCOVERY MECHANISM

- The multi-player game and the IoT project must implement a peer-discovery mechanism, that is, any node that enters a network will be able to find other nodes running the same software automatically, this is done using UDP broadcasting (link). Broadcasting allows the node to send packets to everybody in the network using a special IP address. Broadcasting is used in DHCP as the device that just entered the network doesn't know the IP of the router/switch to obtain IP from. Note that the broadcasting should stop after a device is connected. You'll use broadcasting to share certain information by which other peers receiving that information will use it to establish a TCP connection with the device.

- A high-level conversation overview example…

```
[UDP] Device 0 -> *** : Hello everybody I am device-Larry, you can connect
to me via TCP using port 6758
[UDP] Device 1-> *** : Hello everybody I am device-Alpha, you can connect
to me via TCP on port 8768
[TCP] Device 0 connects to port 8768 of device 1
[TCP] Device 0 -> Device 1: READ Temperature
[TCP] Device 1 -> Device 0: 24
```

- In a nutshell, you broadcast your name and how people can reach you to the network so TCP connection can be established without prior knowledge or their port and address.

IOT DESCRIPTION

- In this project our target is to control and read values from a slave (a device that doesn't have an operating system) which can be an Arduino or a NodeMCU or whatever microcontroller you want using sockets.

- The input value will be a push button attached to the controller and the output value can be an LED -Our project will do more complicated stuff than on an LED.

- The master device can be a PC or an Android device, anything that has an operating system (RPi included).
  It will be able to read data from and send data to the salve.

IOT & SOFTWARE FUNCTIONALITY

- The general architecture of the project is that there is a master device and a slave device. The master device controls the slave device using UDP and TCP packets.

- The master device code is written in python while the slave device code is written in C so that it can run on a microcontroller (Arduino in our case).

- First both the master and slave devices listen for UDP broadcast messages. When the master discovers a slave (using Peer discover mechanism), it extracts the IP of the slave from the broadcast message and initializes a TCP connection with this device for data transmission. When the TCP connection is created between the slave and the master device, the master can now send control packets to control the slave.

- These control messages can be simple TCP messages with a specific format previously agreed on by both the master and the slave or can be done using the MQTT protocol. In case of the MQTT, the master and slave both subscribe to the same topics and the master publish control messages with the know format of the MQTT publish function which is *publish (topic, message).*

- Our Peer implementation supports multiple devices, as a single peer can broadcast UDP message, and can be either a TCP client or TCP server.

- Our Peer implementation can transmit different types of data from devices e.g. integers and strings and handle that data correctly on the master.

BONUS 1 – CONTROLLING MORE SOPHISTICATED

- Our IOT project is designed to do more sophisticated task than lightening ON/OFF a LED. It can record voice and send it to the master Peer (Software).

- When the embedded slave receives turns on, it broadcast itself telling that it is on network.

- The master (software) peer respond by passing a post signal, when the embedded slave record sound an transmit it to the master PC.

BONUS 2 – USING MQTT PROTOCOL

- We used MQTT protocol to control your devices over the internet.
- **cloudmqtt** is one of the free MQTT brokers link.

THANK YOU