THE NATIONAL INSTITUTE OF ENGINEERING

(An Autonomous Institute under VTU, Belagavi)
Mysuru- 570008

Department of Electronics & Communication Engineering

MINOR PROJECT REPORT- EC6C06

A SECURE IoT-BASED HEART MONITORING SYSTEM

Under the guidance of:

Sharmila B S

Assistant Professor

Department of Electronics and Communication

NIE, Mysuru

Submitted by:

Anjani Kumar Srivastava –4NI19EC015

Bhargavi R –4NI19EC123

Nitesh Raj –4NI19EC063

P Uttam – 4NI19EC064

VI Semester B.E

2021-22

TABLE OF CONTENTS

INTRODUCTION

- The rapid advancement in smart object technology has included significant achievements in application development for wireless sensor-based distributed communication architecture. Capitalizing on the contactlessness and efficiency of data retrieval from modern smart objects, various innovative types of on-demand and real-time services have been developed and deployed in daily life.

- The protection of data security and entity privacy are the most critical aspects of IoT-based healthcare systems. As the communication of the BSN is mostly wireless (and insecure) in nature, various attacks may be launched at it as a vulnerability entry, resulting in serious system damage to the entire system.

- Today, the IoT has become one of the most promising communication paradigms, and one in which all the smart objects in our daily life become part of the Internet owing to their communication and computing capabilities. This opportunity brings with it new security challenges for IoT applications. Every smart object (or sensor) in the IoT represents a potential risk in terms of system vulnerability. That is, each intelligent object may become a vulnerable entry point for any malicious attack.

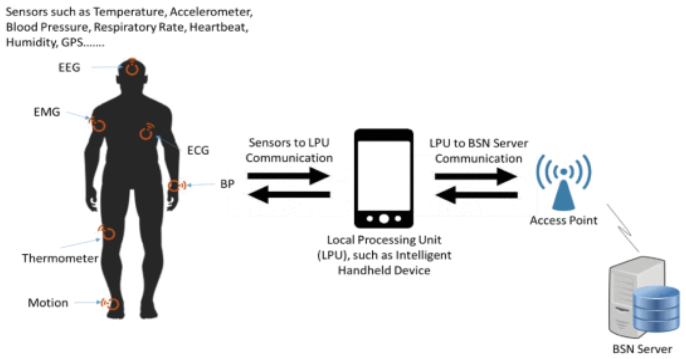
- Two security issues, i.e. (1) physical protection for smart objects, and (2) how to maintain data confidentiality, integrity, and privacy during data collection among smart objects, have thus emerged. Given the novelty and innovative nature of IoT technologies, there seems to be a general expectation for a new and revolutionary security solution tailored specifically to IoT-based objects. This is because traditional security protection mechanisms may not be suitable for smart objects. For example, firewalls containing network management control protocols are able to manage high-level traffic through the Internet.

- However, this application-level solution is not suitable for endpoint devices in IoT applications because these devices usually possess a specific, defined mission with limited resources available to accomplish it. Therefore, the refinement of traditional security solutions to fit the specific security requirements of IoT-based smart objects is one of the most promising ways of securing IoT-based application systems. Normally, security is addressed during the system's life cycle, including secure booting, access control, device authentication, network management, firewall, IDS/IPS and updates, and patches.

LITERATURE SURVEY

(A Secure IoT-Based Healthcare System With Body Sensor Networks - KUO-HUI YEH, (Senior Member, IEEE) - National Dong Hwa University, Hualien 97401, Taiwan)

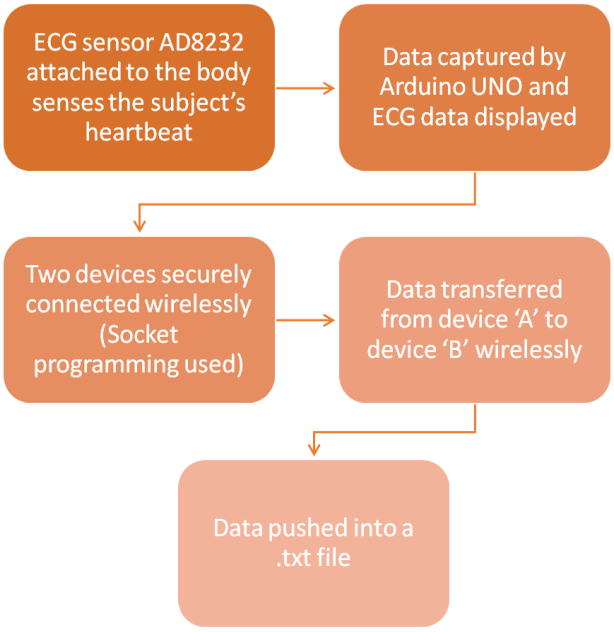
- There are three indispensable components in the IoT-based communication architecture:
    - the wearable body bio-sensors (i.e., the smart objects, in this case, the ECG sensor module)
    - the Local Processing Unit - LPU (the data processing is done as per the instructions given by us)
    - The Body Sensor Networks - BSN server
- The IoT-based biomedical equipment (i.e., body bio-sensors) is adopted (or embedded) by the user for collecting bio-data from the human. All the collected data will be forwarded to the server for data analysis and user-oriented service provision. That is, based on specific bio-data from the user, the system can recognize and satisfy the individual's needs in a faster and more efficient way.
- For instance, by analyzing human bio-data, such as electrocardiography (ECG), electroencephalography (EEG), electromyography (EMG), and blood pressure (BP), a healthcare system in a hospital can provide more individually-tailored and timely services and reduce delays in medical treatment.
- **The need for resistance to man-in-the-middle attack:**
  Resistance to man-in-the-middle attacks is one of the most important security considerations after authentication. A malicious attacker may interrupt transmitted authentication messages and spoof the legal communicating entities into believing that he/she is the other legitimate side via counterfeited and illegal message spoofing. That is, the attacker may pretend that he/she is the legitimate user who is communicating with the server. Spoofing can also be used when the attacker faces a real legitimate user. The attacker may pretend to be the legitimate server to communicate with the legal user. An efficient solution for resisting man-in-the-middle attacks is to embed the identities of all communicating entities into the protocol message for entity authentication.
- **Multiple security and privacy properties must be guaranteed at the same time:**
  The protection of data security and entity privacy are the most critical aspects of IoT-based healthcare systems. As the communication of the BSN is mostly wireless (and insecure) in nature, various attacks may be launched at it as a vulnerability entry, resulting in serious system damage to the entire system. First, mutual authentication among communication entities is required to protect against malicious data access and entity spoofing. Second, the system has to achieve anonymity and untraceability for the biosensors in IoT-based healthcare systems to guard against the disclosure of an individual's personal health status or private information. Third, the resistance against forgery attacks and replay attacks during system operations must be embedded into the IoT-based healthcare system.



PROBLEM STATEMENT

- Consider the scenario: There is a doctor and patient in one room. The patient's heart rate is sensed and output is obtained (from the first device which has the heart rate sensing system) in this room. The value has to be sent to a second device in another room.
- In the other room, there is another doctor. His device receives the data and pushes that data into a file, and displays the ECG graph. Thus, the reports are transmitted wirelessly from one device to another device.
- In this project, we have built a secure heart monitoring system in which the heart rate of an individual is measured, the data is sent to another device through sockets and the graph is displayed.

THE PROPOSED SECURE HEART MONITORING SYSTEM - BLOCK DIAGRAM



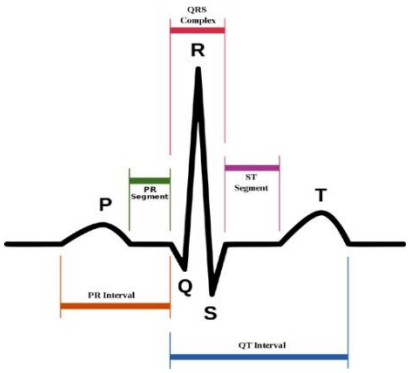
ELECTRO-CARDIO-GRAPH - ECG

Each interval in an ECG signal has an optimal value range, and deviation from that might be linked to a particular disease. Here are the main parts of an ECG signal.

1. P wave - It is the trailing wave on the left of the QRS complex.
2. QRS complex - It is an impulse generated by ventricular contraction.
3. T wave - It is a leading wave right to the QRS complex.
4. U wave - It is not always observed due to its low peak value.

We can diagnose a lot of cardiac diseases and irregularities. For example:

- Irregular heartbeat – the absence of P-wave –might indicate Atrial Fibrillation
- Resting Heart Rate of more than 100 – might indicate Tachyarrhythmia
- Sawtooth P wave – might indicate Atrial flutter
- Depression of ST-segment – might indicate Ischemia
- Elevation of ST-segment – might indicate Myocardial Infarction



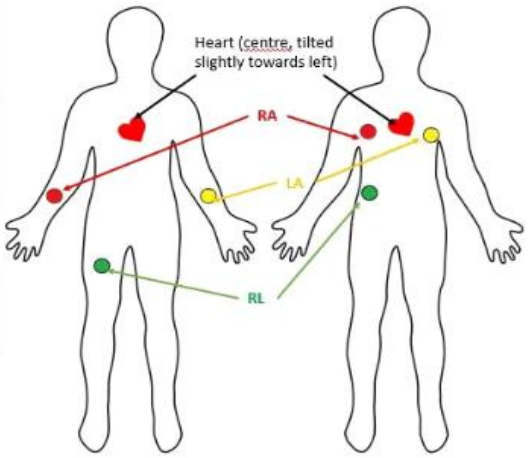
AD8232 ECG Sensor:

- The AD8232 ECG sensor is a commercial board used to calculate the electrical movement of the human heart. The output of this is an analog reading.
- The **working principle of the ECG sensor** is like an operational amplifier to help in getting a clear signal from the intervals simply.
- The main purpose of this chip is to amplify, extract as well as filter biopotential signals which are small in noisy conditions like those formed due to motion.

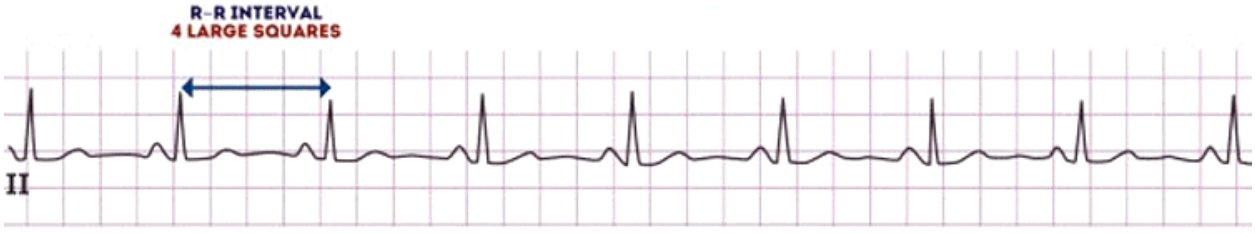
HEART RATE MEASUREMENT FROM ECG GRAPH

- The sensor neither analyzes the ECG, nor checks the presence of the P or T waves, nor will it measure the heartbeat by itself. The calculations must be fed in by us.
- Heart rate = No. of R peaks in 1 minute. This means that it would take 1 entire minute for a single heart rate value to appear.

Electrode Name	Electrode colour	Location
RA	Red	Right Arm
LA	Yellow	Left Arm
RL	Green	Right Leg



- This method is medically incorrect because this assumes that each R-R interval throughout the entire 1 minute is the same. However, this is not the case.
- For an average healthy human, the time taken by 1 entire heartbeat is around 800-850 ms (65-75 beats per minute). This might change depending on whether the person is resting or working. Whatever the case might be, the heartbeat intervals do not vary much for a person. But for a person suffering from Atrial fibrillation or Arrhythmia, the heartbeat varies significantly throughout the ECG, despite the average heartbeat appearing to be normal. The parameter to identify this is called Heart Rate Variability.



HEART RATE = 300 ÷ (NUMBER OF LARGE SQUARES IN ONE R-R INTERVAL)

300 ÷ 4 = 75 BPM

- Calculating Heart Rate Variability (HRV) using AD8232 ECG Sensor:

$$HRV = \frac{HR}{60} - RR \text{ interval}$$

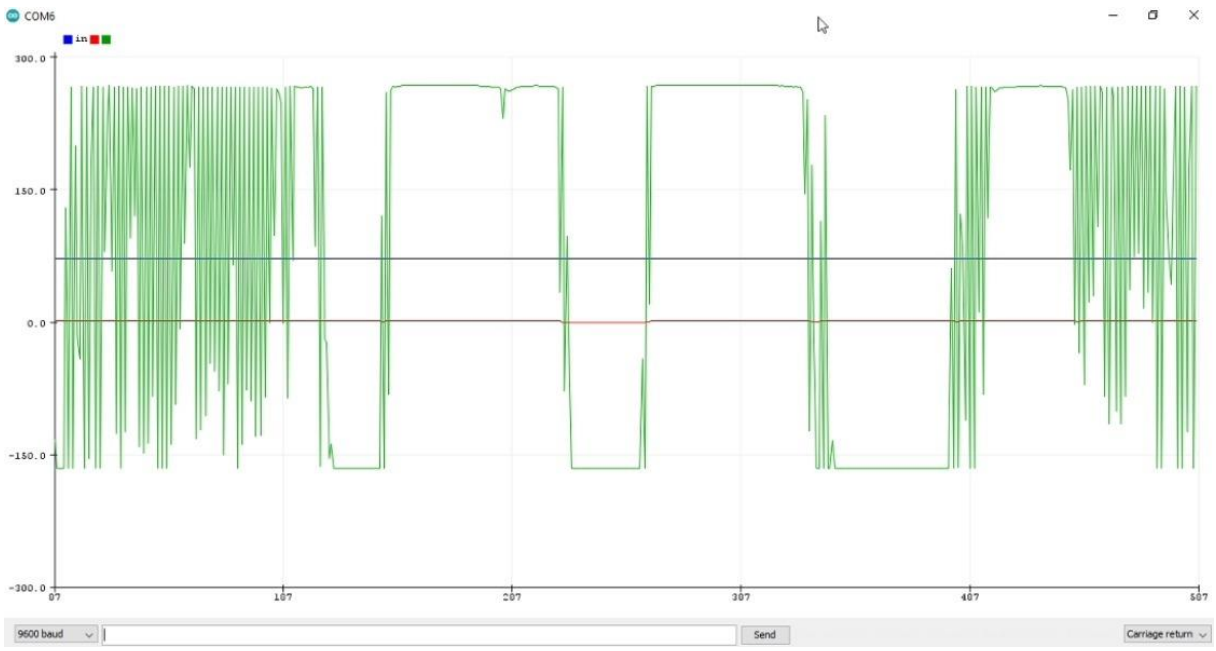
- So, for calculating HRV, we need HR first. But 1 minute is too long. Hence, we use a window of 10 seconds to calculate both parameters.

$$HR = RR\ peaks\ in\ 10\ seconds\ \times 6$$

$$HRV = \frac{HR}{60} - RR\ interval$$

The output of the ECG module:

- The green signal indicates a slightly modified ECG while the blue signal indicates the approximated heartbeat.
- HRV is not visible much on the plotter because it will generally be around -1 to 1 milli secs for a normal human, while the ECG value and heart rate will be around 70. Hence, HRV is seen as a red line on the serial plotter.
- On the serial monitor, the values are displayed as follows: heart rate, HRV, ECG value.



SOCKET PROGRAMMING

- Socket programming is a way of connecting two nodes on a network to communicate with each other.
- One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection.
- The server forms the listener socket while the client reaches out to the server. They are the real backbones behind web browsing.
- A network socket is a software structure within a network node of a computer network that serves as an endpoint for sending and receiving data across the network.
- Socket programming is started by importing the socket library and making a simple socket.

  ```
  import socket

  s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  ```

- Here we made a socket instance and passed two parameters. The first parameter is AF_INET (refers to the address-family ipv4) and the second one is SOCK_STREAM (it means connection-oriented TCP protocol).

Server:

- A server has a bind() method which binds it to a specific IP and port so that it can listen to incoming requests on that IP and port.
- A server has a listen() method which puts the server into listening mode. This allows the server to listen to incoming connections.
- A server has an accept() and close() method. The accept method initiates a connection with the client and the close method closes the connection with the client.
- After that, we bound our server to the specified port. Passing an empty string means that the server can listen to incoming connections from other computers as well.

Client:

- We need something with which a server can interact. We could tenet to the server like this just to know that our server is working.

CODES

Arduino code:

```
/*
 * count: variable to hold count of rr peaks detected in 10 seconds
 * flag: variable that prevents multiple rr peak detections in a single heartbeat
 * hr: HeartRate (initialized to 72)
 * hrv: Heart Rate variability (takes 10-15 seconds to stabilize)
 * instance1: instance when heartbeat first time
 * interval: the interval between second beat and first beat
 * timer: variable to hold the time after which hr is calculated
 * value: raw sensor value of output pin
 * values displayed - hr, hrv, ECG_value
 */
long instance1=0, timer; double hrv =0, hr = 72, interval = 0;
int value = 0, count = 0;  bool flag = 0;
#define shutdown_pin 10
#define threshold 100 // to identify R peak
#define timer_value 10000 // 10 seconds timer to calculate hr
void setup() {
 Serial.begin(9600);
 pinMode(10, INPUT); // Setup for leads off detection LO +
 pinMode(11, INPUT); // Setup for leads off detection LO -
}
void loop() {
 if((digitalRead(8) == 1)||(digitalRead(9) == 1)){
  Serial.println("No input, leads off!");
  digitalWrite(shutdown_pin, LOW); //standby mode
  instance1 = micros();
  timer = millis();
 }
 else {
  digitalWrite(shutdown_pin, HIGH); //normal mode
  value = analogRead(A0);
  value = map(value, 250, 400, 0, 100); //to flatten the ecg values a bit
  if((value > threshold) && (!flag)) {
   count++;
   Serial.println("in");
   flag = 1;
   interval = micros() - instance1; //RR interval
   instance1 = micros();
  }
  else if((value < threshold)) {
   flag = 0;
  }
  if ((millis() - timer) > 10000) {
   hr = count*6;
   timer = millis();
   count = 0;
  }
  hrv = hr/60 - interval/1000000;
  Serial.print(hr);
  Serial.print(",");
  Serial.print(hrv);
  Serial.print(",");
  Serial.println(value);
  delay(10);
 }
}
```

Sender program:

SENDER.py × sendercopy.py* ×

```
# This file is used for sending the file over socket
import os; import socket; import time; from security import encrypt
# Creating a socket.
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client = None
try:
    sock.bind((socket.gethostname(), 22222))
    sock.listen(5)
    # Accepting the connection.
    (clientConn, clientAddr) = sock.accept()
    print("Host Name: ", sock.getsockname())
except Exception as ex:
    print("Exception in bind/listen: ", ex)
# Getting file details.
file_name = input("File Name: "); length = len(file_name)
fnameBuff = file_name; fnameBuff += '_' * (100 - length)
file_size = os.path.getsize(file_name); print(file_size, 'bytes')
# Sending file_name and size details.
fsizeBuff = str(file_size); length = len(fsizeBuff); fsizeBuff += '\0' * (100 - length)
clientConn.send(str(fnameBuff).encode()); clientConn.send(str(fsizeBuff).encode())
# Opening file and sending data.
with open(file_name, "rb") as file:
    c = 0
    # Starting the time capture.
    start_time = time.time()
    # Running loop while c != file_size.
    while c <= file_size:
        data = file.read(1024)
        if not (data):
            break
        print(data)
        enc_data = encrypt(data)
        print(enc_data)
        clientConn.sendall(enc_data)
        c += len(data)
    # Ending the time capture.
    end_time = time.time()
total = end_time - start_time
print("File transfer complete. Total time: ", format((total),".5f"), 'seconds.')
# Closing the socket.
sock.close()
```

Receiver program:

C:\Users\Bhargavi R\Desktop\Final_receiver\receivercopy.py

RECEIVER.py × receivercopy.py* ×

```
# This file will be used for receiving files over socket connection.
import os
import socket
import time
from security import decrypt

def getValidStr_n(string):
    for i in range (len(string)):
        if string[i] == '_':
            string = string.replace('_', ' ')
    return string
def getValidStr_s(string):
    for i in range (len(string)):
        if string[i] == '/0':
            string = string.replace('/0', ' ')
    return string
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = '192.168.1.42'
# Trying to connect to socket.
try:
    sock.connect((host, 22222))
    print("Connected Successfully!")
except Exception as ex:
    print("Unable to connect: ", ex)
    print(host)
# Send file details.
file_name = sock.recv(100).decode()
file_size = sock.recv(100).decode()

file_name = getValidStr_n(file_name)
file_size = getValidStr_s(file_size)
# Opening and reading file.
with open("./" + file_name, "wb") as file:
    c = 0
    # Starting the time capture.
    start_time = time.time()
    # Running the loop while file is being received.
    while c <= int(file_size[:2]):
        data = sock.recv(1024)
        if not (data):
            break
        print(data)
        dec_data = decrypt(data)
        print(dec_data)
        file.write(dec_data)
        c += len(dec_data)
    # Ending the time capture.
    end_time = time.time()
total = end_time - start_time
print("File transfer complete. Total time: ", format((total),".5f"), 'seconds.')
# Closing the socket.
sock.close()
```

Outputs:

C:\Users\B\Desktop\Final_Sender\sendercopy.py

SENDER.py × sendercopy.py ×

```
# This file is used for sending the file over socket
import os; import socket; import time; from security import encrypt
# Creating a socket.
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client = None
try:
    sock.bind((socket.gethostname(), 22222))
    sock.listen(5)
    # Accepting the connection.
    (clientConn, clientAddr) = sock.accept()
    print("Host Name: ", sock.getsockname())
except Exception as ex:
    print("Exception in bind/listen: ", ex)
# Getting file details.
file_name = input("File Name: "); length = len(file_name)
fnameBuff = file_name; fnameBuff += '_' * (100 - length)
file_size = os.path.getsize(file_name); print(file_size, 'bytes')
# Sending file_name and size details.
fsizeBuff = str(file_size); length = len(fsizeBuff); fsizeBuff += '\0' * (100 - length)
clientConn.send(str(fnameBuff).encode()); clientConn.send(str(fsizeBuff).encode())
# Opening file and sending data.
with open(file_name, "rb") as file:
    c = 0
    # Starting the time capture.
    start_time = time.time()
    # Running loop while c != file_size.
    while c <= file_size:
        data = file.read(1024)
        if not (data):
            break
        print(data)
        enc_data = encrypt(data)
        print(enc_data)
        clientConn.sendall(enc_data)
        c += len(data)
    # Ending the time capture.
    end_time = time.time()
total = end_time - start_time
print("File transfer complete. Total time: ", format((total),".5f"), 'seconds.')
# Closing the socket.
sock.close()
```

Source Console Object a

Usage

Here you can get help of any object by pressing

Help Variable Explorer Plots Files

Console 2/A ×

```
584 bytes
b'72.00,1.20,121\r\n72.00,1.20,115\r\n72.00,1.20,118\r\n72.00,1.20,119\r
\n72.00,1.20,114\r\n72.00,1.20,118\r\n\r\n72.00,1.20,121\r
\n72.00,1.20,115\r\n72.00,1.20,118\r\n72.00,1.20,119\r\n72.00,1.20,114\r
\n72.00,1.20,118\r\n\r\n72.00,1.20,121\r\n72.00,1.20,115\r
\n72.00,1.20,118\r\n72.00,1.20,119\r\n72.00,1.20,114\r\n72.00,1.20,118\r\n
\r\n72.00,1.20,121\r\n72.00,1.20,115\r\n72.00,1.20,118\r\n72.00,1.20,119\r
\n72.00,1.20,114\r\n72.00,1.20,118\r\n\r\n72.00,1.20,121\r
\n72.00,1.20,115\r\n72.00,1.20,118\r\n72.00,1.20,119\r\n72.00,1.20,114\r
\n72.00,1.20,118\r\n\r\n72.00,1.20,121\r\n72.00,1.20,115\r
\n72.00,1.20,118\r\n72.00,1.20,119\r\n72.00,1.20,114\r\n72.00,1.20,118'
b'gAAAAABiuyGcoI1XUf5aAz_DXP1rjCVfvmX_Zuxh15p2c3MyQ4WxWDUbYgiIZl7unyBBtydC-
98C9ess28L454gaRGgZ7BbOUlOFRVhCYQYufrY_dxlV_98mQl50ff70XmmJRxRkjdQqYliOKNg2
ylE6rKVDYtEket1u7UN798wrptN8I5oirp1hfDnz12n0bxX46R4J0HnpGmeRJCrxXn6Af7kD6Nw
bXMLHU9KeRx6fV5sR0BbO3cRaiLwxKA5JT6zdL2kZjd_3yzxfOXMTgvw9ZeKaOOtM4EpMT41E-
po9VM-aGJv-WNpCWM-
C9glktIEM0UQlYoxpNuNTsREs69YUrCbWWzYTlCbWeEM0lZilhDOAUEzXXiSxQyMJBqdkaPZKFI
I017stJ3V80rWfErCAmawNlVDy9uX15CME5RTM-kvNiU2TvZO3S71isyjL6z0FNpcG8h1thCHH-
QVILImdMnIpNAiMW97sDGcD9MhmHVhlXgD7UjFAfFyVcANFV7qonQc8SmUwXuvQtCafygQeczsq
9xUTa78Tc3U_Y6w6lXgubaJSbWxGlA9hyI0tlx9de7NSe1FthHFA2hVqPmblljDyvnSpqENa3eW
vZg3SDxqnL6Hak5efrRVl-HZhLWu_rq2_Qt9meuNydhYeZtv0cYkLw7b4--
NmRkUioHbvI_8Q1NbrHQJ3v2o2NZzZ-2J9n2738Cq33dXgk6uRFT8HNbtSJ49SU4-
l8dFlLAPmJuA1ZwfS4L-
yQQNN1SM9-5TW9NxAIxqiDF9PlodgkVwLb6xPt8CmKa7awQDehy0h7fuSyLJH8ttuSu1u4PSjAf
PkEhuil3ADz-2Iyf56CD95D9dpOPZ-7Q==
File transfer complete. Total time:  0.00200 seconds.
```

IPython console History

C:\Users\Bhargavi R\Desktop\Final_receiver\receivercopy.py

RECEIVER.py | receivercopy.py

```python
# This file will be used for receiving files over socket connection.
import os
import socket
import time
from security import decrypt

def getValidStr_n(string):
    for i in range (len(string)):
        if string[i] == '_':
            string = string.replace('_', ' ')
    return string
def getValidStr_s(string):
    for i in range (len(string)):
        if string[i] == '/0':
            string = string.replace('/0', ' ')
    return string
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = '192.168.1.42'
# Trying to connect to socket.
try:
    sock.connect((host, 22222))
    print("Connected Successfully!")
except Exception as ex:
    print("Unable to connect: ", ex)
    print(host)
# Send file details.
file_name = sock.recv(100).decode()
file_size = sock.recv(100).decode()

file_name = getValidStr_n(file_name)
file_size = getValidStr_s(file_size)
# Opening and reading file.
with open("./" + file_name, "wb") as file:
    c = 0
    # Starting the time capture.
    start_time = time.time()
    # Running the loop while file is being received.
    while c <= int(file_size[:2]):
        data = sock.recv(1024)
        if not (data):
            break
        print(data)
        dec_data = decrypt(data)
        print(dec_data)
        file.write(dec_data)
        c += len(dec_data)
    # Ending the time capture.
    end_time = time.time()
total = end_time - start_time
print("File transfer complete. Total time: ", format((total),".5f"), 'seconds.')
# Closing the socket.
sock.close()
```

Source | Console | Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the

Help | Variable Explorer | Plots | Files | Code Analysis

Console 1/A

```
Connected Successfully!
```

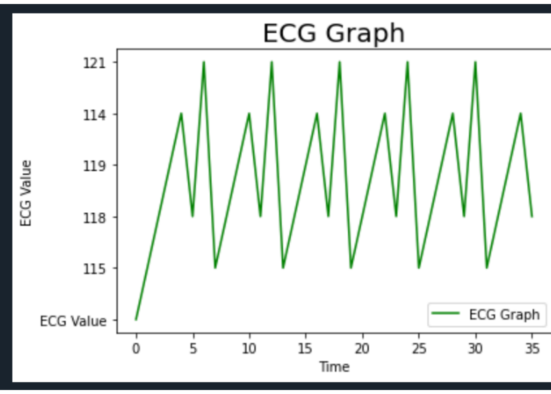

```
b'gAAAAABiuyGcoI1XUf5aAz_DXP1rjCVfvmX_Zuxh15p2c3MyQ4WxWDUbYgiIZl7unyB
BtydC-98C9ess28L454gaRGgZ7BbOUlOFRVhCYQYufrY_dxlV_98mQl50ff70XmmJRxRk
jdQqYliOKNg2ylE6rKVDYtEket1u7UN798wrptN8I5oirp1hfDnz12n0bxX46R4J0HnpG
meRJCrxXn6Af7kD6NwbXMLHU9KeRx6fV5sR0BbO3cRaiLwxKA5JT6zdL2kZjd_3yzxfOX
MTgvw9ZeKaOOtM4EpMT41E-po9VM-aGJv-WNpCWM-
C9glKtIEM0UQlYoxpNuNTsREs69YUrCbWWzYTlCbWeEM0lZilhDOAUEzXXiSxQyMJBqdk
aPZKFII017stJ3V80rWfErCAmawNlVDy9uX15CME5RTM-
kvNiU2TvZO3S71isyjL6z0FNpcG8h1thCHH-
QVILImdMnIpNAiMW97sDGcD9MhmHVhlXgD7UjFAfFyVcANFV7qonQc8SmUwXuvQtCafyg
Qeczsq9xUTa78Tc3U_Y6w6lXgubaJSbWxGlA9hyI0tlx9de7NSe1FthHFA2hVqPmblljD
yvnSpqENa3eWvZg3SDxqnL6Hak5efrRVl-
HZhLWu_rq2_Qt9meuNydhYeZtv0cYkLw7b4--
NmRkUioHbvI_8Q1NbrHQJ3v2o2NZzZ-2J9n2738Cq33dXgk6uRFT8HNbtSJ49SU4-
lBdFlLAPmJuA1ZwfS4L-
yQQNN1SM9-5TW9NxAIxqiDF9PlodgkVwLb6xPt8CmKa7awQDehy0h7fuSyLJH8ttuSu1u
4PSjAfPKEhuil3ADz-2Iyf56CD95D9dpOPZ-7Q=='
b'72.00,1.20,121\r\n72.00,1.20,115\r\n72.00,1.20,118\r
\n72.00,1.20,119\r\n72.00,1.20,114\r\n72.00,1.20,118\r\n\r
\n72.00,1.20,121\r\n72.00,1.20,115\r\n72.00,1.20,118\r
\n72.00,1.20,119\r\n72.00,1.20,114\r\n72.00,1.20,118\r\n\r
\n72.00,1.20,121\r\n72.00,1.20,115\r\n72.00,1.20,118\r
\n72.00,1.20,119\r\n72.00,1.20,114\r\n72.00,1.20,118\r\n\r
\n72.00,1.20,121\r\n72.00,1.20,115\r\n72.00,1.20,118\r
\n72.00,1.20,119\r\n72.00,1.20,114\r\n72.00,1.20,118\r\n\r
\n72.00,1.20,121\r\n72.00,1.20,115\r\n72.00,1.20,118\r
\n72.00,1.20,119\r\n72.00,1.20,114\r\n72.00,1.20,118\r\n\r
\n72.00,1.20,121\r\n72.00,1.20,115\r\n72.00,1.20,118\r
\n72.00,1.20,119\r\n72.00,1.20,114\r\n72.00,1.20,118'
```


```
File transfer complete. Total time:  0.00150 seconds.
```

IPython console | History

LSP Python: ready | conda (Python 3.9.12) | Line 6, Col 1 | ASCII | CRLF | RW | Mem 53%

```python
# txt file to csv file
dataframe1 = pd.read_csv("Data1.txt")
dataframe1.columns = ['Heart rate', 'Heart rate variable', 'ECG Value']
dataframe1.to_csv('Data1.csv', index = None)
x = []
with open('Data1.csv','r') as csvfile:
    lines = csv.reader(csvfile, delimiter=',')
    for column in lines:
        x.append(column[2])
plt.plot(x, color = 'g',label = "ECG Graph")
plt.xlabel('Time'); plt.ylabel('ECG Value')
plt.title('ECG Graph', fontsize = 20)
plt.legend(); plt.show()
```



Security code:

```python
# Importing the cryptograpy module for using encryption and decryption method.
from base64 import urlsafe_b64encode
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives.hashes import SHA256
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

# Generating the Secret Key
secret_key = "VerySecretKey"
kdf = PBKDF2HMAC(
    algorithm=SHA256(),
    length=32,
    salt="Pass Salt Here".encode(),
    iterations=100000,
)
key = urlsafe_b64encode(kdf.derive(secret_key.encode()))

# Assign the Method to a vaiable and pass the key
token = Fernet(key)

def encrypt(data):
    return token.encrypt(data)

def decrypt(data):
    return token.decrypt(data)
```

FUTURE SCOPE

- This can be made more accurate by using a higher-end ECG monitoring device
- A more complex security code can be used to make the system more secure.
- More clients can be connected to the server, thus making the network a bigger one.

CONCLUSION

- This project thus demonstrates a very efficient heart monitoring system embedded into a secure network.
- It gives an idea of how security can be implemented in the most basic way.

REFERENCES

- https://ieeexplore.ieee.org/document/7779108 - A Secure IoT-Based Healthcare System With Body Sensor Networks - KUO-HUI YEH, (Senior Member, IEEE) - National Dong Hwa University, Hualien 97401, Taiwan
- https://how2electronics.com/ecg-monitoring-with-ad8232-ecg-sensor-arduino/
- https://www.youtube.com/watch?v=01y_Vu_sAQU
- https://www.geeksforgeeks.org/socket-programming-python/