Project Report 4

Citizen Safety Device

Group 37

Yashvi Pipaliya (AU1841092)

Kesha Bagadia (AU1841011)

Yashvi Gandhi (AU1841033)

Manal Shah (AU1841026)

CHAPTER : 10

Details of Communication Protocols

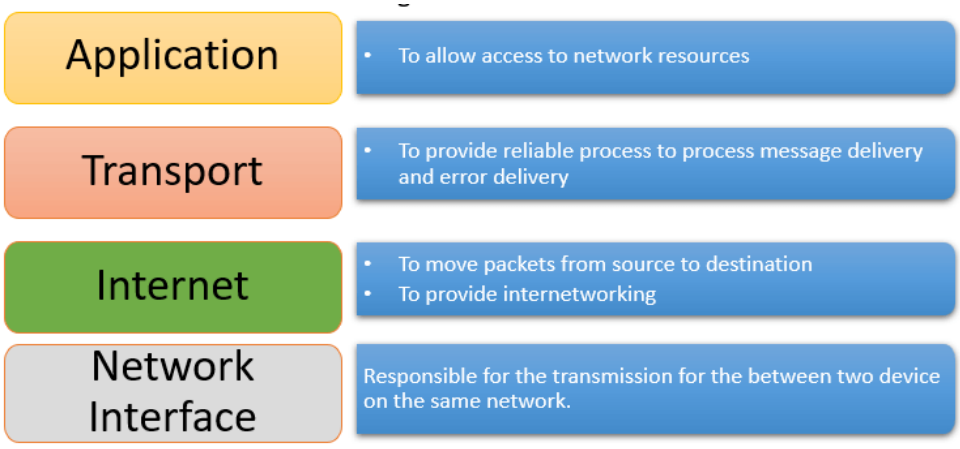
- **TCP/IP PROTOCOL**

  Usage :

  - The GSM Modem SIM900A has an **internal TCP/IP stack** to enable us to connect with the **internet via GPRS.** It is suitable for SMS, Voice as well as DATA transfer applications in the M2M interface.
  - The ESP8266 WiFi Module is a **self contained SOC with integrated TCP/IP protocol stack** that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

  Detail :

  - TCP/IP stands for Transmission Control Protocol/ Internet Protocol. TCP/IP Stack is specifically designed as a model to offer highly reliable and end-to-end byte stream over an unreliable internetwork.
  - TCP/IP Model helps to determine how a specific computer should be connected to the internet and how data should be transmitted between them. It helps to create a virtual network when multiple computer networks are connected together. The purpose of the TCP/IP model is to allow communication over large distances.
  - The functionality of the TCP IP model is divided into four layers, and each includes specific protocols. TCP/IP is a layered server architecture system in which each layer is defined according to a specific function to perform. All these four TCP IP layers work collaboratively to transmit the data from one layer to another.
    - Application Layer
    - Transport Layer
    - Internet Layer
    - Network Interface



Four Layers of TCP/IP model

Advantages :

- It helps to establish/set up a connection between different types of computers.
- It operates independently of the operating system.
- It supports many routing-protocols.
- It enables the internetworking between the organizations.
- TCP/IP model has a highly scalable client-server architecture.
- It can be operated independently.
- Supports a number of routing protocols

Disadvantages :

- TCP/IP is a complicated model to set up and manage.
- The shallow/overhead of TCP/IP is higher-than IPX (Internetwork Packet Exchange).
- In this model the transport layer does not guarantee delivery of packets.
- Replacing protocol in TCP/IP is not easy.
- It has no clear separation from its services, interfaces, and protocols.

- **NMEA PROTOCOL**

  Usage :

  - NMEA, UBX and RTCM protocols are used in the NEO-6M GPS Module.

  Detail :

  - NMEA 0183 is a combined electrical and data specification for communication between marine electronics such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. It has been defined by, and is controlled by, the National Marine Electronics Association. It replaces the earlier NMEA 0180 and NMEA 0182 standards. In leisure marine applications it is slowly being phased out in favor of the newer NMEA 2000 standard,though NMEA0183 remains the norm in commercial shipping.
  - The electrical standard that is used is EIA-422, although most hardware with NMEA-0183 outputs are also able to drive a single EIA-232 port. Although the standard calls for isolated inputs and outputs, there are various series of hardware that do not adhere to this requirement.
  - The NMEA 0183 standard uses a simple ASCII, serial communications protocol that defines how data is transmitted in a "sentence" from one "talker" to multiple "listeners" at a time. Through the use of intermediate expanders, a talker can have a unidirectional conversation with a nearly unlimited number of listeners, and using multiplexers, multiple sensors can talk to a single computer port.
  - At the application layer, the standard also defines the contents of each sentence (message) type, so that all listeners can parse messages accurately.
  - While NMEA0183 only defines an RS422 transport, there also exists a de facto standard in which the sentences from NMEA0183 are placed in UDP datagrams (one sentence per packet) and sent over an IP network.

- **UBX PROTOCOL**

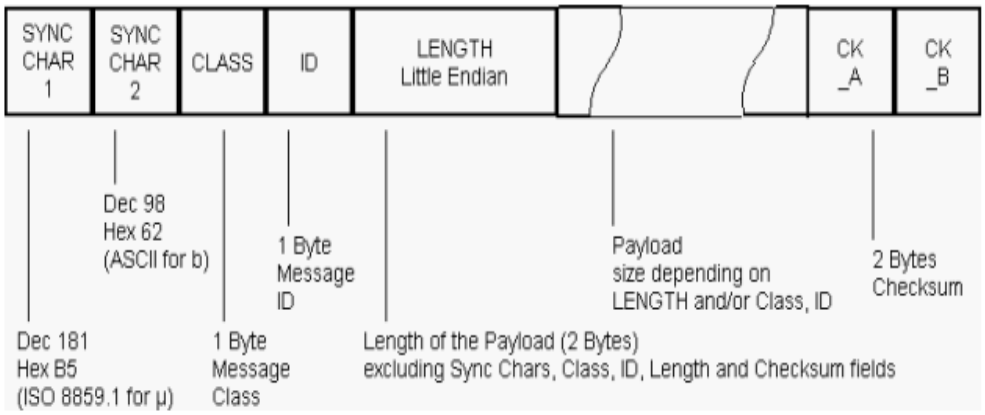
  Usage :

  - NMEA, UBX and RTCM protocols are used in the NEO-6M GPS Module.

  Detail :

  - u-blox GPS receivers use a u-blox proprietary protocol to transmit GPS data to a host computer using asynchronous RS232 ports. This protocol has the following key features:
    - Compact - uses 8 Bit Binary Data.
    - Checksum Protected - uses a low-overhead checksum algorithm
    - Modular - uses a 2-stage message identifier (Class- and Message ID)
  - u-blox receivers are fully configurable with UBX protocol configuration messages (message class UBX-CFG). The configuration used by the u-blox receiver during normal operation is called "Current Configuration". The Current Configuration can be changed during normal operation by sending any UBX-CFG-XXX message to the u-blox receiver over an I/O port. The u-blox receiver will change its Current Configuration immediately after receiving the configuration message. The ublox receiver always uses only the Current Configuration.

  - A basic UBX Packet looks as follows:



- **RTCM PROTOCOL**

  Usage :

  - NMEA, UBX and RTCM protocols are used in the NEO-6M GPS Module.

  Detail :

  - RTCM SC-104, or often simply the RTCM standard, is a communication protocol for sending differential GPS (DGPS) to a GPS receiver from a secondary source like a radio receiver. The format does not define the source of the messages and has been used with systems as varied as longwave marine radio, communications satellite broadcasts, and internet distribution.
  - The first widely used version of the format was released in 1990 and was based on the 30-bit long packet used by the GPS satellites, known as a "frame". Each message started with a standardized two-frame header and then one or more data frames following. The frames were designed to be similar to GPS to make integration in GPS receivers easier, but had the disadvantage of having low channel efficiency and limiting the number of messages that could be sent in a given time.
  - A completely new message format was introduced in 2003 for version 3 of the standard which used a variable-length format to improve efficiency and increase the number of messages that could be sent, which was important for real-time GPS corrections. The new standard also greatly increased the number of possible message types. As part of the standards process, the naming of the standard was changed, and version 3.1 became RTCM Standard 10403.1. As of 2021, the latest version is 3.2, or 10403.2

Table 1: RTCM-3 Message Structure (RTCM Standard 10403.1, 2006)

Preamble	Reserved	Message Length	Data Message	Checksum
8 bits	6 bits	10 bits	n bits	24 bits
0xD3	Not defined	Message Length in bytes	Variable length in bytes	QualComm definition CRC-24Q

Comparison Chart of Wi-Fi, Bluetooth and Zigbee

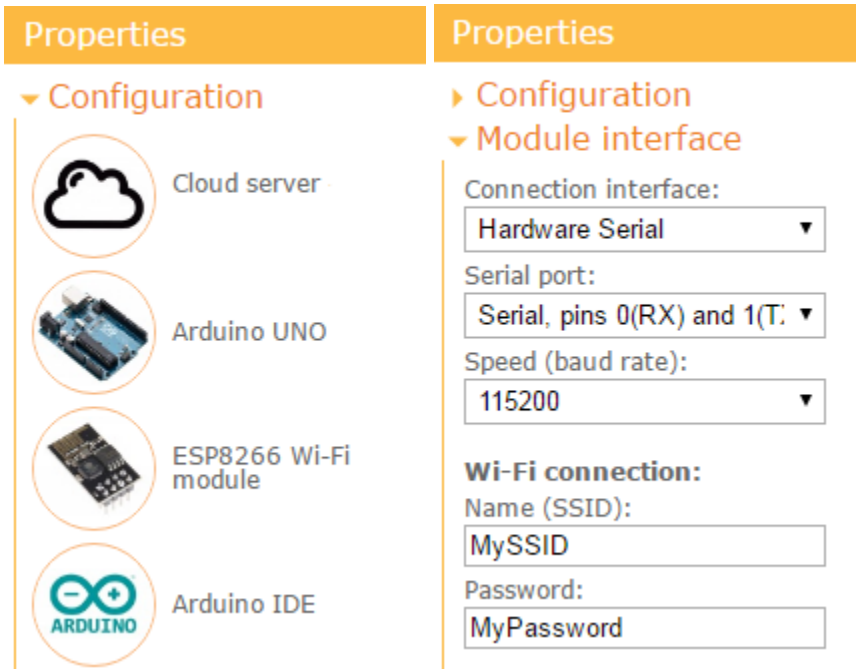
	Wi-Fi (IEEE 802 11B)	**Bluetooth (IEEE 802.15.1)**	**Zigbee (IEEE 802.15.4)**
Radio	DSSS	FHSS	DSSS
Data Rate	11 Mbps	1 Mbps	250kbps
Nodes per Master	32	7	64000
Data Type	Video, audio, graphics, pictures, files	Audio, graphics, pictures, files	Small data packet
Range (M)	100	10	70
Extendibility	Roaming possible	No	Yes
Battery Type	AAA battery power	Li-ion batteries	Alkaline batteries
Slave enumeration latency	Upto 3s	Upto 10s	30 ms
Complexity	Complex	Very complex	Simple
Frequency	2.4 GHz	2.4 GHz	850-930 MHz
Security	Authentication Service Set ID (SSID)	64 bit, 128 bit	128 bit AES and application layer user defined
Operating Environment	32℉-104℉	32℉-122℉	32℉-131℉
Battery Life	Hours	1 Week	> 1 Year
Success Metrics	Speed, flexibility	Compatible, easy to use	Reliability, power,cost

CHAPTER : 11

Details of supporting tools

Technical Details

The ESP8266 module is configured as a client to connect to a WiFi access point. The access point must have an Internet connection. The RemoteXY library ensures registration of the device on the cloud server.



The mobile application connects to the cloud server, and not to the device directly. Thus, the device will be available from anywhere on the Internet.

The application will fetch data from the sensors real time and update their live status on its interface for the close contacts of the central user. The details will only be accessible when the safety mode is on in accordance with the system flow. When active, the pulse rate and finger press status will determine the imminent danger the central user is in and update the alert.

Photo

Complete Code of Application

```
*/

//////////////////////////////////////////////
//        RemoteXY include library          //
//////////////////////////////////////////////

// RemoteXY select connection mode and include library
#define REMOTEXY_MODE__ESP8266_SOFTSERIAL_CLOUD
#include <SoftwareSerial.h>

#include <RemoteXY.h>

// RemoteXY connection settings
#define REMOTEXY_SERIAL_RX 2
#define REMOTEXY_SERIAL_TX 3
#define REMOTEXY_SERIAL_SPEED 9600
#define REMOTEXY_WIFI_SSID "YOUR_SSID"
#define REMOTEXY_WIFI_PASSWORD "YOUR_PASS"
#define REMOTEXY_CLOUD_SERVER ""
#define REMOTEXY_CLOUD_PORT 6376
#define REMOTEXY_CLOUD_TOKEN ""


// RemoteXY configurate
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
  { 255,0,0,21,0,99,0,13,8,1,
  68,17,21,42,20,20,8,36,129,0,
  18,64,26,5,99,80,117,108,115,101,
  32,82,97,116,101,0,65,112,40,25,
  9,9,67,4,14,83,21,12,36,16,
  11,69,0,38,84,10,10,1,129,0,
  5,7,36,6,99,83,97,102,101,116,
  121,32,77,111,100,101,0,129,0,7,
  28,31,5,99,70,105,110,103,101,114,
  32,80,114,101,115,115,0,70,19,44,
  5,9,9,26,12,0 };

// this structure defines all the variables and events of your control
```

```
interface
struct {
    // output variables
  float onlineGraph_1;
  uint8_t led_2_r; // =0..255 LED Red brightness
  uint8_t led_2_g; // =0..255 LED Green brightness
  uint8_t led_2_b; // =0..255 LED Blue brightness
  char text_1[11];  // string UTF8 end zero
  int16_t sound_1; // =0 no sound, else ID of sound, =1001 for example,
look sound list in app
  uint8_t led_1; // led state 0 .. 1
    // other variable
  uint8_t connect_flag;  // =1 if wire connected, else =0

} RemoteXY;
#pragma pack(pop)

/////////////////////////////////////////////
//           END RemoteXY include          //
/////////////////////////////////////////////

void setup()
{
  RemoteXY_Init ();
  // TODO you setup code

}

void loop()
{
  RemoteXY_Handler ();
  // TODO you loop code
  // use the RemoteXY structure for data transfer
  // do not call delay()
}
```

Appendix B

Programming Review

Points to compare	C	C++	Java	Python
Paradigms	Procedural	Object Oriented	Object Oriented	Multi-Paradigm
Platform Dependency	Dependent	Dependent	Independent	Independent
Keywords	32	63	50 defined (goto, const unusable)	33
Preprocessor directives	Supported (#include, #define)	Supported (#include, #define)	Not Supported	Not Supported
Header files	Supported	Supported	Use Packages (import)	Use Packages (import)

Inheritance	No Inheritance	Supported	Multiple Inheritance Not Supported	Supported
Overloading	No Overloading	Supported	Operator Overloading Not Supported	Operator Overloading Not Supported
Pointers	Supported	Supported	No Pointers	No Pointers
Multi-threading and Interfaces	Not Supported	Not Supported	Supported	Supported
Database Connectivity	Not Supported	Not Supported	Supported	Supported
Code Translation	Compiled	Compiled	Interpreted	Interpreted
Exception Handling	No Exception handling	Supported	Supported	Supported

Appendix C

Trouble-shooting

1. **Ublox NEO-6M GPS Module**

   Problem: Not able to get any data from Neo-6M (GY-GPS6MV2) connected to Arduino

   Connections: GND - GND, TX - D3, RX - D2, VCC - 5V

   Code:

```
#include <SoftwareSerial.h>


// choose pins that connect to UBlox Neo 6m
int RXPin = 2;
int TXPin = 3;
int GPSBaud = 4800;

SoftwareSerial gpsSerial(RXPin, TXPin);


void setup() {
  // put your setup code here, to run once:
  // define pin modes for tx, rx:
  pinMode(RXPin, INPUT);
  pinMode(TXPin, OUTPUT);
  Serial.begin(9600);
  gpsSerial.begin(9600);
  Serial.write("Welcome to the GPS Show");
}

void loop() {
  // put your main code here, to run repeatedly:
  while(gpsSerial.available() > 0) {
    Serial.write(gpsSerial.read());
  }
}
```

Debugging & Troubleshooting:

After referring the data sheet (http://www.u-blox.com/images/downloads/Product_Docs/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf), we found that the baud rate should be set to 9600 for the module to work and GSM module's operating voltage is between 2.7V-3.6V, so changed the supply voltage to 3.3V.

VCC - 3.3V

int GPSBaud = 9600;

2. AE GSM MODEM SIM900A

Problem: The program is not loaded successfully.

Connections old: GND - GND, TX - 0, RX - 1, VCC - 5V, GSM Module Vol - 12V

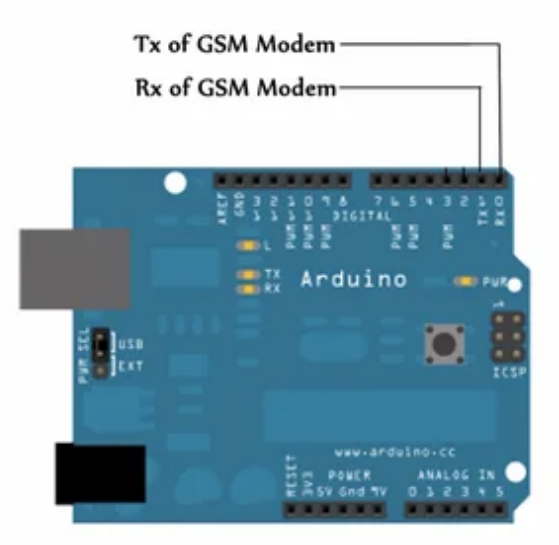
Connections new: GND - GND, TX - 9, RX - 10, VCC - 5V, GSM Module Vol - 12V

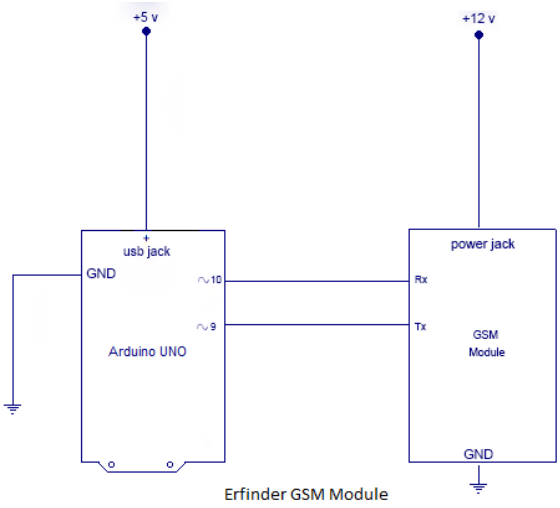
Debugging & Troubleshooting:

There are two ways of connecting a GSM module to an arduino. In any case, the communication between Arduino and GSM modules is serial. So we used serial pins of Arduino (Rx and Tx). So with this method, we connected the Tx pin of the GSM module to the Rx pin of the Arduino and the Rx pin of the GSM module to the Tx pin of Arduino.

Then, we loaded the programs to communicate with the gsm module to make it work.

The problem with this connection came while programming. Arduino uses serial ports to load programs from the Arduino IDE. If these pins are used in wiring, the program will not be loaded successfully to Arduino. So we had to disconnect wiring in Rx and Tx each time we ran the program. Once the program is loaded successfully, we can reconnect these pins and have the system working! To avoid this difficulty, We came up with an alternate method in which two digital pins of the arduino are used for serial communication. We need to select two PWM enabled pins of the arduino for this method. So we chose pins 9 and 10 (which are PWM enabled pins). This method is made possible with theSoftwareSerial Library of Arduino. SoftwareSerial is a library of Arduino which enables serial data communication through other digital pins of Arduino. The library replicates hardware functions and handles the task of serial communication.





Erfinder GSM Module

3. Heart Rate Pulse Sensor

Problem: Sensor not working properly

Debugging & Troubleshooting: According to the datasheet given, the operating current should be 3-4mA and it seemed that the current it received was more for some reason due to which it got short circuited.