CMPE 220

Topic: Firefighter Dashboard Application

Vatsal Makani: 013731614

Computer Engineering Department,

Davidson College of Engineering,

San Jose State University, San Jose, CA 94303

E-mail: [vatsalkirit.makani@sjsu.edu](mailto:vatsalkirit.makani@sjsu.edu)

—————————— ◆ ——————————

Table of Contents

[1 Introduction 2](#_Toc16527514)

[2 Related work 3](#_Toc16527515)

[3 The Innovation Design 3](#_Toc16527516)

[4 Technical discussion and R&D 5](#_Toc16527517)

[5 Results, Experiment & Evaluation 12](#_Toc16527518)

[6 Conclusions 17](#_Toc16527519)

[7 Read me 17](#_Toc16527520)

[8 References/Citations 18](#_Toc16527521)

# 1 Introduction

In its Research Roadmap for Smart Fire Fighting Summary Report, the National Institute of Standards and Technology (NIST) predicts that in the future, new and emerging technologies will usher in the evolution of “smart fire fighting” in which firefighters will have a whole new arsenal of tools to work with, such as: data-driven, science-based tactic, information-rich decision making, devices that provide for better situational awareness, comprehensive data collection, analysis, and communication in real time, interconnected buildings and equipment to provide better monitoring, data, and control systems, the ability to use devices to perform functions that humans cannot do.The proposed product - Firefighter dashboard shall be used for the firefighter’s safety and precautions against dangerous hazards. The present invention relates to sensor data monitoring which provides an automated alarm system for monitoring multiple parameters during firefighting activities and providing appropriate instructions or indications to a firefighter to inform him of a dangerous situation. The aim of this project is to get all the telemetry data of the firefighter team members and send it to the team lead either using serial communication/network protocol. The product being developed for the firefighters and their team members, keeping their requirements in mind the major market segment of the product aims to be Fire Department, but it can be extended to various communities, military and sport adventurous groups, by adding or subtracting some sensor controllers as per their requirements. In the current scenario, firefighters are not embedded with any sensor device and most of the time, firefighters must restore to manual methods of raising the alarm when they are exposed to dangerous environmental situations. These firefighters are covered with thick insulating uniforms which gives them little to no indication about the temperatures rising above the dangerous limits, the heat may get accumulated in these uniforms without any warning to the firefighters, adding further risk to their lives. Hence it becomes difficult to keep track of all the firefighters by the team lead or the firefighter marshal, so with the help of this product, the team lead, or the firefighter marshal shall be able to monitor the firefighter’s health, position and their surrounding areas. As this product is initially targeted for local firefighter department, so after completion of the product, this prototype will be experimented to monitor the product’s efficiency, we can expand the domain to medical applications where blood pressure sensor, sensor to measure glucose, etc can be integrated and monitored by the doctors. This project provides the following key differentiative features and benefits to the potential customer:

* Firefighter’s location and health monitoring
* Push-to-talk service in extreme situations
* Easy maintenance and low cost
* Interactive Dashboard

# 2 Related work

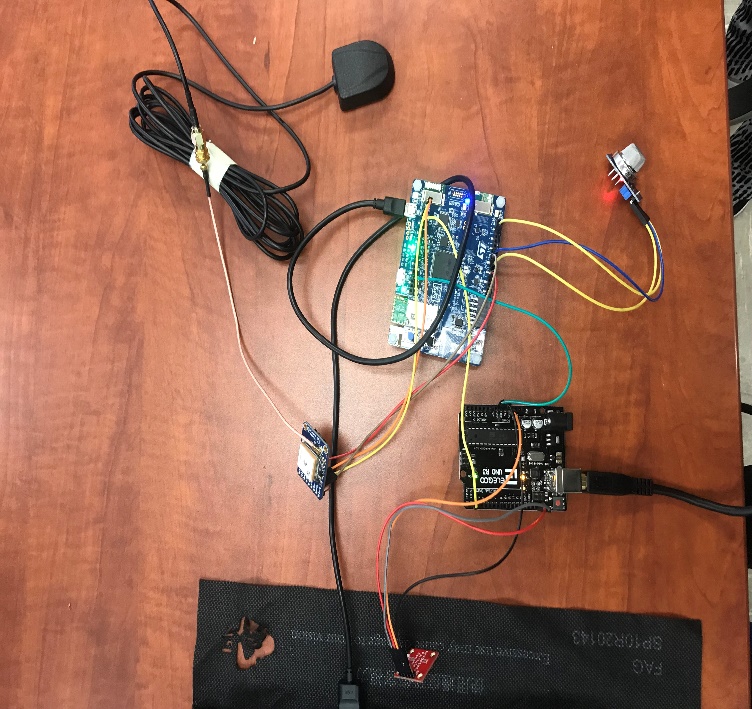
The existing competitive technology allows firefighters to carry the wearable device and an additional walkie-talkie. Sometimes carrying this additional device may seem difficult to manage during the extreme fire situations. To avoid this shortcoming, our proposed product shall implement a built-in push-to-talk feature, which will prevent the firefighters to carry an additional device. This feature will help firefighters to talk to each other and firefighter marshal at ease. Another technological hurdle that these current states of the art products have includes lack of providing the position of the firefighters. Our proposed product will help the firefighter marshal to monitor the GPS coordinates of the team and accordingly provide the instructions to those who are stuck in dangerous surroundings using push-to-talk service. The GPS coordinates can also be monitored over dashboard in real time by the marshal, who guides other firefighters about the whereabouts of each other. In the worst-case scenario, if the firefighter stops responding or acknowledging, the firefighter marshal can summon a rescue team for the firefighter in danger based on their GPS coordinates in real time scenario. Our proposed product involved interfacing STM32 board with Raspberry Pi 3 B+, implementing a push-to-talk service using Raspberry Pi and designing interactive GUI considering QT framework as a platform. Due to such a low scale implementation, our total product cost shall be less than the existing market products. In case of any pitfall in the device, the faulty controllers and sensor modules are easily replaceable aiding low and easy product maintenance. After successful completion of this product, the firefighting team lead shall monitor the telemetry data and accordingly provide instructions to its team members using push-to-talk service preventing any life casualty/injury.

# 3 The Innovation Design

My area of work is to implement the sensor communication with STM32 IoT Discovery Board and integrate it with Wifi to send the sensor data over UDP. I have implemented the setup (attached below in the screenshot) for interfacing the sensors. We are using the below list of components:

* STM32 IoT Discovery Kit([B-L475E-IOTA](https://www.mouser.com/ProductDetail/STMicroelectronics/B-L475E-IOT01A2?qs=2m8Gdae5Lr2iZaHS3QKU3w%3D%3D&gclid=CjwKCAjwnMTqBRAzEiwAEF3ndlOikoFIgSTf_jNljQWBZHWZVm-vAP166VQPG2jOLbi8GCzSkrrnMBoClTsQAvD_BwE))
* GPS sensor ([Adafruit Ultimate GPS Breakout - 66 channel w/10 Hz updates - Version 3](https://www.adafruit.com/product/746))
* MAX30105 Particle Sensor ([SparkFun Particle Sensor Breakout - MAX30105](https://www.sparkfun.com/products/14045))
* MQ-135 Air Quality Sensor ([MQ135 Air Quality Sensor Hazardous Gas Detection Module For Arduino AVR](https://www.amazon.com/HiLetgo%C2%AE-Quality-Hazardous-Detection-Arduino/dp/B00LSG5IZ2/ref=asc_df_B00LSG5IZ2/?tag=hyprod-20&linkCode=df0&hvadid=198075593513&hvpos=1o2&hvnetw=g&hvrand=14201395471880539994&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9032166&hvtargid=aud-801381245258:pla-349898399170&psc=1))
* GPS Antenna ([RP SMA Female to uFL/u.FL/IPX/IPEX RF Coaxial Cable Pigtail Adapter Connector](https://www.amazon.com/SaferCCTV-Coaxial-Pigtail-Connector-Transmitter/dp/B06WCZ9JPL/ref=sr_1_4?keywords=GPS+Antenna+PCIe&qid=1557263057&s=gateway&sr=8-4))
* HTS221 On Board Temperature Sensor ([HTS221](https://www.mouser.com/ProductDetail/STMicroelectronics/STEVAL-MKI141V2?qs=T3y1Fp6ghJZnkGm0cL3g1A%3D%3D&gclid=CjwKCAjwnMTqBRAzEiwAEF3ndvOTTj6GOl0tlP4uJkW2AgAz_Xf5CZwo8XHCTypP1NtcF48BlcLLbhoCI2wQAvD_BwE))
* Arduino UNO ([Elegoo UNO](https://www.amazon.com/ELEGOO-Board-ATmega328P-ATMEGA16U2-Compliant/dp/B01EWOE0UU/ref=asc_df_B01EWOE0UU/?tag=hyprod-20&linkCode=df0&hvadid=309751315916&hvpos=1o1&hvnetw=g&hvrand=8960103122754677552&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9032166&hvtargid=aud-801381245258:pla-455309014075&psc=1&tag=&ref=&adgrpid=67183599252&hvpone=&hvptwo=&hvadid=309751315916&hvpos=1o1&hvnetw=g&hvrand=8960103122754677552&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9032166&hvtargid=aud-801381245258:pla-455309014075))

My task was to configure the GPS, Air quality, on board temperature sensors, send the data to STM32 Discovery Kit, Interfacing particle sensor with Arduino UNO and aiding other team members for particle sensor configuration and wifi integration. As mentioned earlier, below is snapshot of my setup with STM32 discovery board interfaced to Arduino UNO and the sensors.



STM32 IoT Discovery Kit with HTS2321 Temperature Sensor

Adafruit GPS Sensor and Antenna

MQ135 Air quality sensor

Arduino UNO

MAX30105 Heart Rate Sensor

Figure 1 Setup

The detailed description of my approach and application of each sensor is covered in next section. I faced few challenges in this project below is the detailed description of my problems and solution

*Challenge 1:* GPS data was not getting received on UART4 of STM32 Discovery Kit. When I started receiving GPS data, I was receiving the following data format

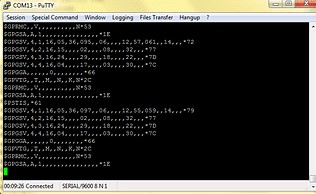


Figure 2 GPS Not locking

*Solution:*  In CubeMX configuration, I did not enable the interrupt mode for UART4. I had to enable the UART4 Global Interrupt in NVIC Settings of UART4. To solve the gps fix problem, I shifted the setup to higher floor (from 2nd floor to 3rd floor) and kept the setup as close as window for proper range.

*Challenge 2:* In FreeRTOS code, I created multiple task for each of the sensor and was executing each task at 100ms delay. When I executed the code, I was just getting UART4 GPS values, it did not execute task for other sensors.

*Solution:*  Increasing the time delay to 400ms and combining the UART\_Receive\_IT() of particle sensor and GPS module in single task solved the problem.

*Challenge 3:* While integrating wifi module with sensor, the UDP connection was getting established and data was getting sent once. When we kept the wifi connection code in the task, it will establish the data and send the data only ONCE, whereas it should keep on sending data

*Solution:* There is need for closing the client connection after the data is being sent. So, the sequence of data shall be Open Client🡪Sent data over Wifi🡪Close Client🡪Repeat

# 4 Technical discussion and R&D

The system block diagram of the project is shown below:

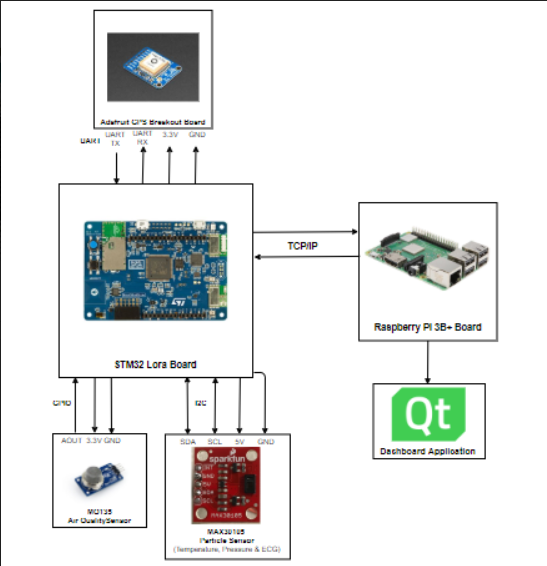


Figure 3 System Block Diagram

Our project consists of the air quality sensor to measure the gas concentration around the firefighter, where STM32 controller shall take the concentration value and warn about the hazards to the firefighter marshal, if the data exceeds certain limits. Temperature and ECG values are obtained from particle sensor to measure temperature around firefighter and heartbeat of the firefighter. The controller shall notify the firefighter marshal about the temperature exceeding above the dangerous limits. These data will be monitored by the marshal with the help of QT application, which gets the data over a network. The push-to-talk service shall be implemented directly using a microphone. In summary, our project highlights 2 new transformative and original feature compared to current state of the art products available in the market:

1. Our project incorporated a push-to-talk feature which helps firefighters communicate with each other at ease. This push-to-talk feature works on pressing a switch which could prove better than handling an additional device such as walkie talkie.
2. GPS Positioning and real-time tracking on the dashboard helps marshal to get the position of the other firefighters, who can provide appropriate instructions, thus ensuring firefighter’s safety.

Air quality sensor (MQ-135)

* MQ135 breakout board consists of MQ135 gas sensor and LM393 differential comparator.
* The gas sensitive material used with the MQ135 gas sensor is tin dioxide (SnO2), which has a lower conductivity in clean air. composed by micro AL2O3 ceramic tube, Tin Dioxide (SnO2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless-steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-135 have 6 pin ,4 of them are used to fetch signals, and other 2 are used for

providing heating current [[1].](#_8_References/Citations)

* It is suitable for the detection of harmful gases such as ammonia, aromatics, sulfides, benzene vapor, smoke and other harmful gas detection devices, gas sensor testing range of concentration: 10 to 1000ppm
* In order to get the data from MQ-135, we need to calibrate the sensor using “Gironi’s Formulation” [[2]](#_8_References/Citations) which converts the output of sensor to the related ppm characteristics for the gas under test. From the sensitivity graph of MQ135 , the resistance ratio of the sensor (RS/RO) and the gas concentration (ppm) are related as a power function:

y = a\*x^b...…..(1)

* The calibration of RScan be done in clean air once stable readings are being received from the sensor,

RS = RO \* sqrt(a/ppm, b)...…..(2)

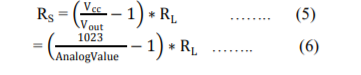
* The concentration scope of a particulargas was identified using datasheet leading to the computation of the limit for RS/RO,

RS/RO\_limit = (ppm/a) ^ (1/b) …… (3)

* STM32 reads and produces output in analog value which is not a very useful parameter for gas concentration reading. This data must be converted to PPM (parts per million) values. First of all, conversion of the analog values (0-4095) to corresponding voltage values(Vout)(0-5V) is done using:

Vout =AnalogValue\*5/1023 ………. (4)

* Resistance of sensor (RS) is defined in the datasheet[1] of MQ135 as (It is 4095 instead of 1023):



* Replacing value of x as (RS/RO) and y as ppm in equation (1), we get the below equation for computing ppm

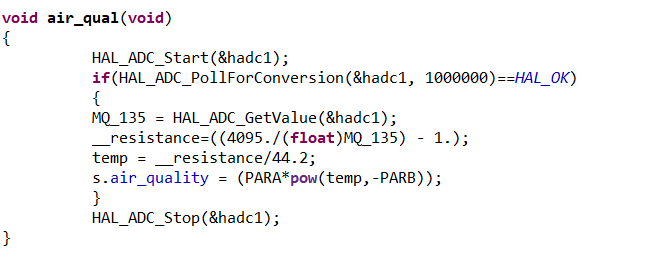
ppm = a\*(RS/RO) ^b ….. (7)

* Power Regression Analysis on data using Excel, gives the values of a andb as: a = 116.6020 and b = -2.76903
* The general sensitivity is roughly same for all the gases detected by MQ135. CO2 is the fourth most abundant trace gas in the earth’s atmosphere with approximately 400 ppm concentration, therefore it is safe to assume that in a normal atmosphere the sensor mostly detects CO2.
* After the Burn-in time (24-48 hours) of sensor, RO is calibrated using the following equation:

RO = RS\*pow(a/ppm, 1/b) …….. (8)

* STM32 uses ADC1 channel to take the reading from MQ-135. We configiured ADC1 channel at pin A3 of STM32 IoT discovery kit in **MX\_ADC1\_Init(void)** in main.c
* Implementation of the algorithm in code is mentioned as below:



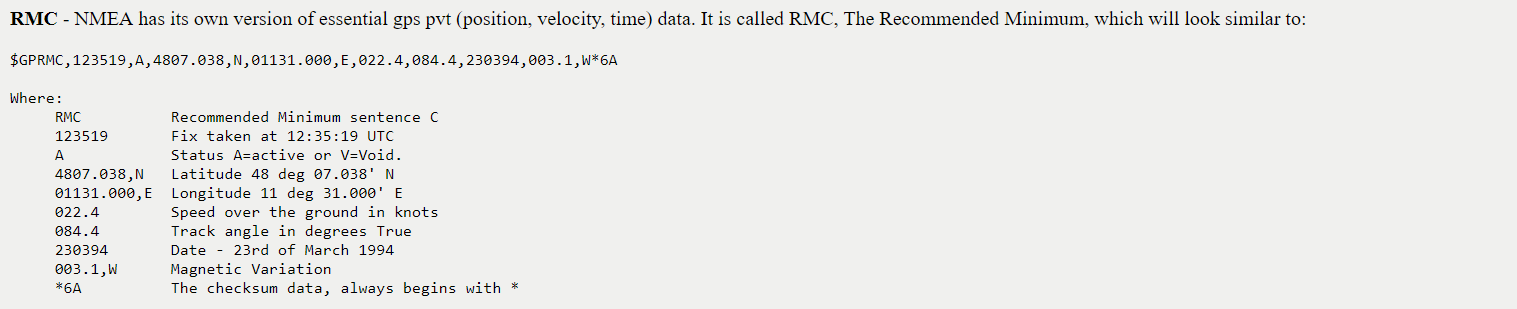


Adafruit GPS Sensor breakout board

* The breakout is built around the MTK3339 chipset, a no-nonsense, high-quality GPS module that can track up to 22 satellites on 66 channels, has an excellent high-sensitivity receiver. It can do up to 10 location updates a second for high speed, high sensitivity logging or tracking. Power usage is incredibly low, only 20 mA during navigation [[3]](#_8_References/Citations).
* It consists of a ultra-low dropout 3.3V regulator, 5V level safe inputs, ENABLE pin so you can turn off the module using any microcontroller pin or switch, a footprint for optional CR1220 coin cell to keep the RTC running and allow warm starts and a tiny bright red LED. The LED blinks at about 1Hz while it's searching for satellites and blinks once every 15 seconds when a fix is found to conserve power. If you want to have an LED on all the time, we also provide the FIX signal out on a pin so you can put an external LED on [[3]](#_8_References/Citations).
* Two features of version 3 MTK3339-based module is the external antenna functionality and the the built-in data-logging capability. The module has a standard ceramic patch antenna that gives it -165 dBm sensitivity, but it was not enough for our application, so we used an external GPS anternna by 3M.
* It makes use of MTK3339 commands which can enable and log on the set of output such as GPRMC, GPGGA, GPGNA, etc which are known as PMTK command packet.
* For our application we used the below 3 commands that will enable GPRMC data, the rate at which the position is echoed and set the update rate from once a second (1 Hz) to 10 times a second (10Hz).



* We connected the GPS module to a GPS antenna with RX and TX pins connected to UART4 pins (PA0 to RX and PA1 to TX). The module is communicating at the baud rate of 9600 bps.
* After setting the baud rate, we get GPRMC data over the output. The data format for GPRMC is as follows:



* From the format, we can see that latitude and longitude is the only information that needs to be parsed. The location value is in this format is combination of actual latitude and longitude as well as the time. So, we implemented a parsing logic that scans the values as per format, stores in variable and converts to actual latitude and longitude

**void** **Uart4\_RX\_GPS\_Int\_Handler**(**void**)

{

**if** (i == 750 || i == -1)

{

**sscanf**(gps\_line, "$GPRMC,%[^,],%2[^,],%lf,%2[^,],%lf", timeStamp, valid\_invalid, &tempLat, south\_or\_north, &tempLon);

s.latitude = (tempLat - 100\*(**int**)(tempLat/100))/60.0;

s.latitude += (**int**)(tempLat/100);

s.longitude = (tempLon - 100\*(**int**)(tempLon/100))/60.0;

s.longitude += (**int**)(tempLon/100);

s.longitude = -s.longitude;

HAL\_UART\_Transmit(&huart1, (uint8\_t \*)aTxBuffer, string\_size, 100);

i = 0;

}

**else**

{

HAL\_UART\_Receive\_IT(&huart4, &gps\_rx, 1);

gps\_line[i] = gps\_rx;

i++;

}

}

* The above lines of code represent the GPS parsing logic in UART4 interrupt callback function.
* We define a buffer of 750 and collect the characters and store in character array till the buffer is full. Once it is full, we scan the array to find the characters starting with “$GPRMC” and stores the values, converts to int and extracts the actual gps latitude and longitude.

HTS221 On Board Temperature Sensor

* The HTS221 is an ultra-compact sensor for relative humidity and temperature. It includes a

sensing element and a mixed signal ASIC to provide the measurement information through

digital serial interfaces.

* The sensing element consists of a polymer dielectric planar capacitor structure capable of

detecting relative humidity variations and it is manufactured using a dedicated ST process.

* On the STM32L4 Discovery kit for IoT node, the I2C2 bus from STM32L475VG is used.
* STM32Cube\_FW\_L4\_V1.12.0 contains the sample project that measures the temperature from onboard HTS221. Since it uses I2C, slave address for HTS221 to read the temperature is

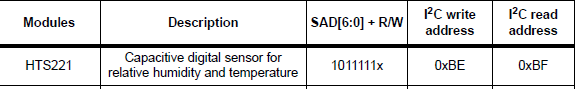
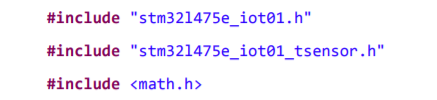


Figure 4 Slave address of HTS221

* As per application sheet [4], we need to configure as per the steps mentioned for reading temperature from HTS221. Step-by-step configuration is covered in the next section.
* In code, we need to include the following files:



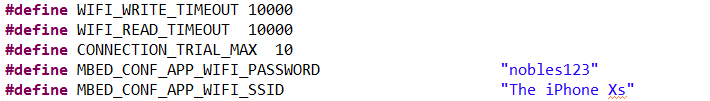
* Initialize the temperature module using **BSP\_TSENSOR\_Init();**
* Take the temperature readings using **BSP\_TSENSOR\_ReadTemp();** placed in while(1) loop, which is our case is the task.We included the sensor task with the highest priority and heap size of 1024 to take readings from all the 4 sensors.

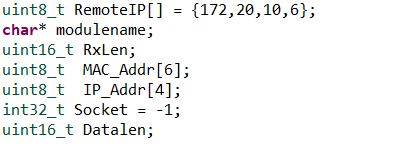


Figure 5 Sensor Task

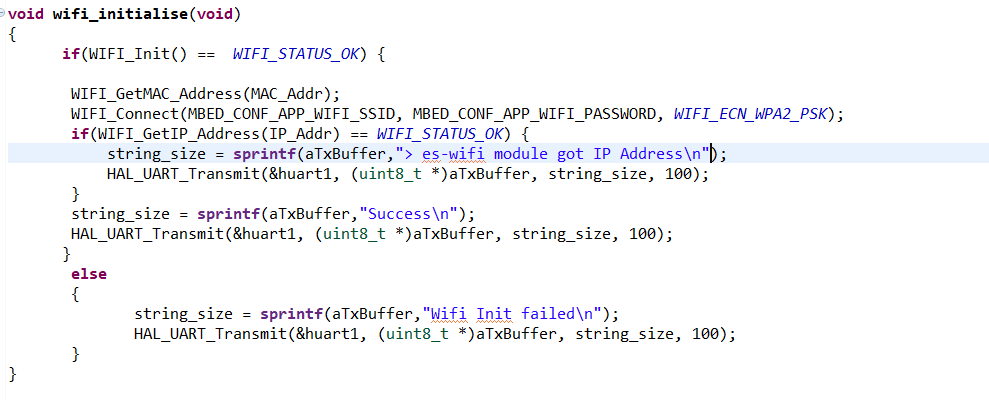
Wifi Module Integration with sensors

* For wifi communication, I assisted Halak in integrating the wifi module with my sensor code. We are using UDP as the communication protocol to send the data to Raspberry Pi 3B+
* First approach is to implement the exisiting STM32 wifi API to send the data. We define the following variables and parameters for wifi communication in main.c file





* We need to add the credentials of the hotspot that STM32 board will connect to. In our case, we used Iphone Xs’s hotspot. After raspberry pi is connected to the same hotspot, we can get the IPv4 address from raspberry Pi and update it in RemoteIP[] array, in our case it was 172.20.10.6.
* We can add the wifi\_init () code that shall connect to the hotspot.



* In sensor task, we can get the data and send it to the network by first connecting to client, sending the data over the wifi and closing the client connection. When the task executes again, we perform the same operation. We are making the structure of sensor data and we define it as firefighter\_sensor\_t struct as below:

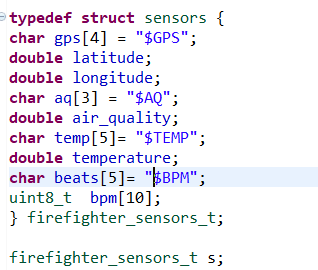


Figure 6 Sensor Structure

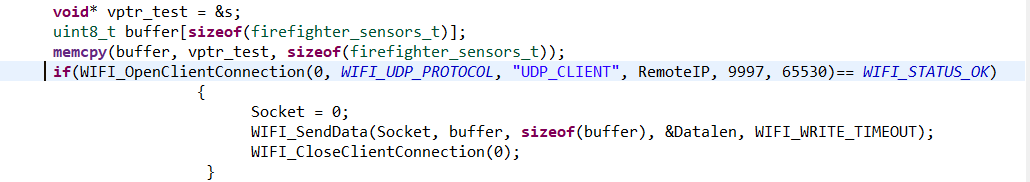


Figure 7 Sending data over Wifi in sensor task

MAX30105 Particle Sensor Overview

* The SparkFun MAX30105 Particle Sensor is a flexible and powerful sensor enabling sensing of distance, heart rate, particle detection, even the blinking of an eye. We use it to measure the heartrate from the IR sensor.
* Arduino UNO is used to communicate with MAX30105 using I2C with help of SparkFun MAX30105 Arduino Library. The instructions to hookup and implement the libraries is mentioned in the link [[5]](#_8_References/Citations).
* The basic working principle is that hemoglobin reflects IR light really well, and the MAX3015 is capable of detecting such small changes in IR reflectance that it can detect blood flowing through your finger at different rates.
* Humans are bad at applying consistent pressure to a thing. Without a rubber band the pressure varies enough to cause the blood in your finger to flow differently which causes the sensor readings to go wonky. It is best to attach the sensor to your finger using a rubber band or other tightening device. This example runs a filter called the PBA or Penpheral Beat Amplitude algorithm on the IR data. This algorithm can pull out the blips from all the noise and calculate the time between blips to get a heart rate. The output is your instantaneous heart rate and your average heart rate (BPM).

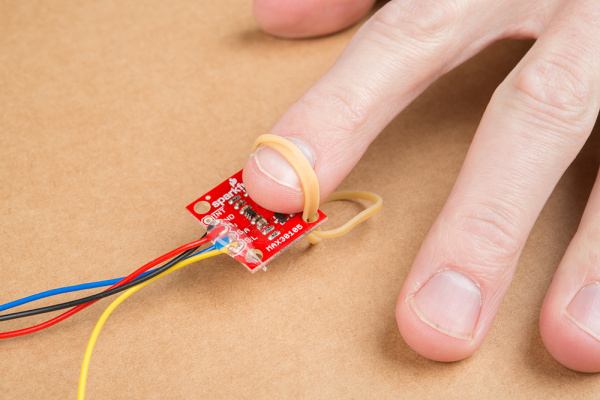


Figure 8 MAX30105 for heartrate measurement

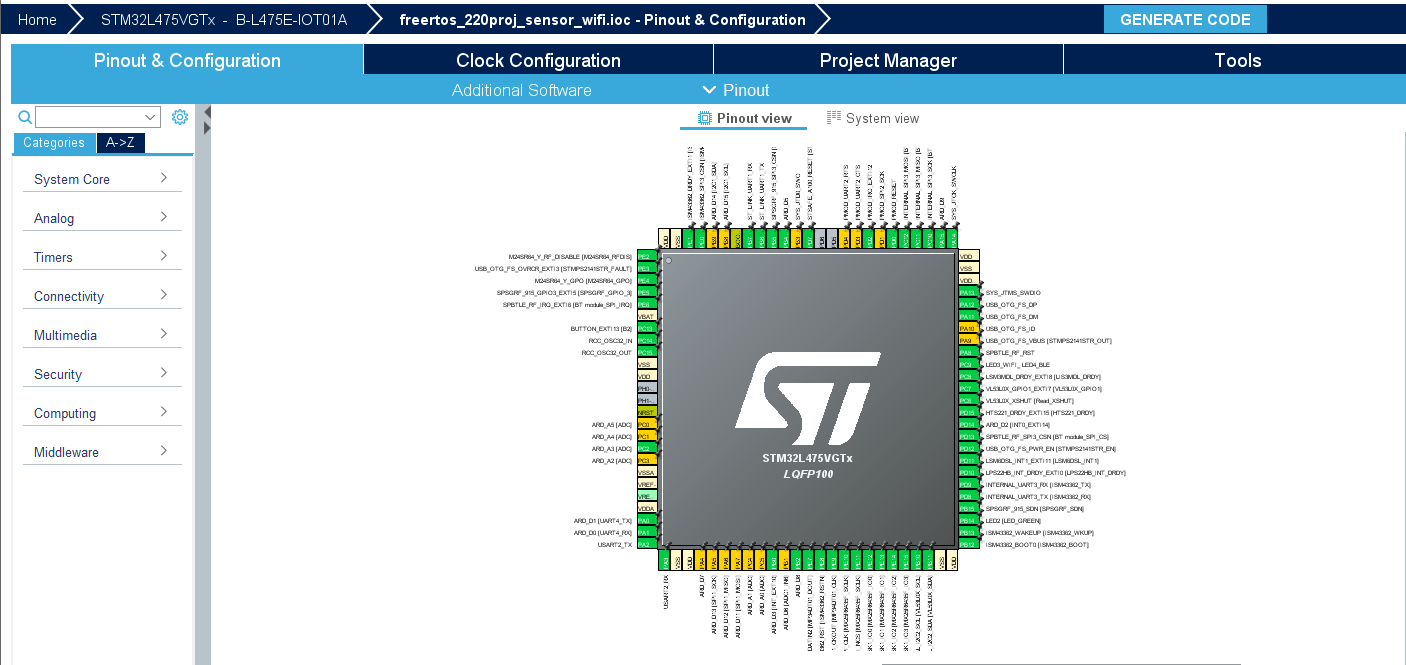
* The BPM data is sent to STM32 discovery kit over UART2 by Arduino TX which is hookedup to ARD\_D4 pin (UART2 RX)

# 5 Results, Experiment & Evaluation

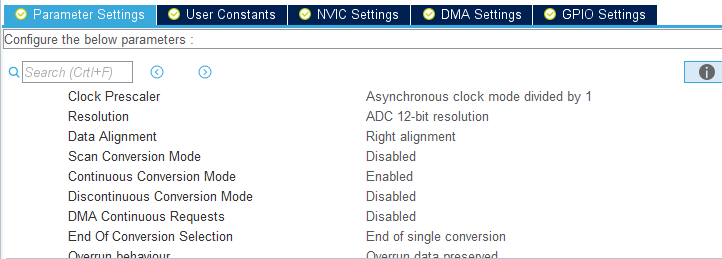
* We set up the sensor modules and their peripherals using STM32 CubeMX. We describe the step-by-step implementation and configuration in STM32 CubeMX and Ac6 System Workbench

Setup and Configurations

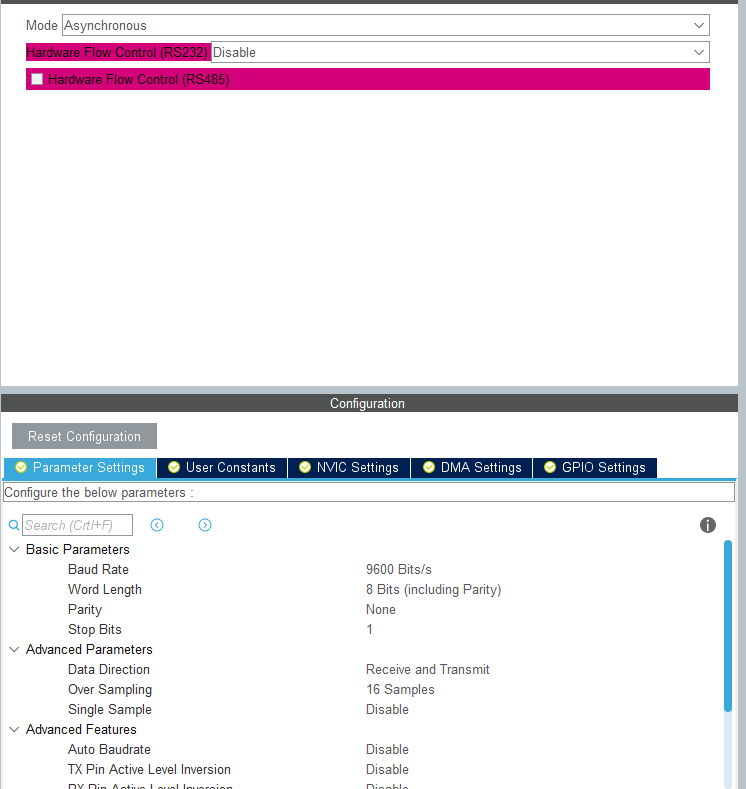
* Open CubeMX software, click on “Board selector”, search B-L475E-IOTA in search box and select the board. Click on “Start Project”
* It will open the new window similar to the below screenshot:



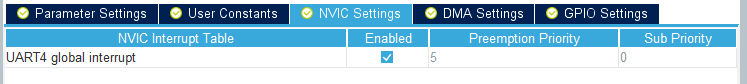
* Go to Analog🡪ADC1🡪IN3🡪Change to Single Ended. Go to the parameter settings. Keep the ADC Resolution at 12 bits. Enable the continuous scanning mode as mentioned in the below snapshot



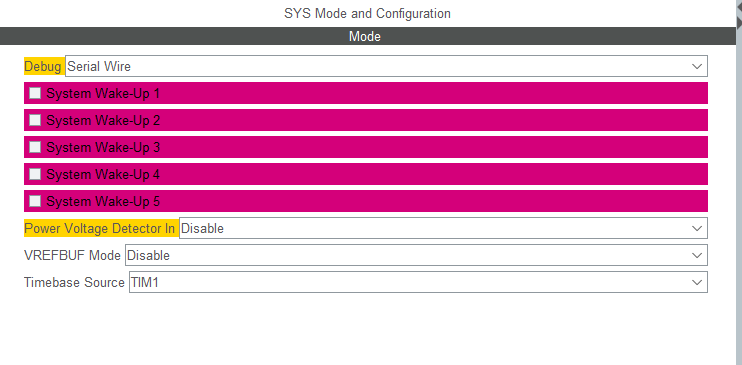
* Go to connectivity on the left panel and click on UART4. Change the mode from disabled to “Asynchronous”. Go to parameter settings and change the baud rate to 9600 and keep other settings same.



* Go to NVIC settings and enable UART4 Global Interrupt



* Go to pin diagram. Search for pin D10 and change it to USART2\_TX and pin D4 to USART2\_RX. Go to connectivity🡪USART2🡪Change the mode to Asynchronous and keep the baud rate at 115200. Enable the USART2 Global Interrupt
* Go to Middleware🡪FreeRTOS🡪Select CMMIS V1
* Go to System Core🡪 SYS🡪Change the Time base source from SysClock to TIM1



* Go to Project Manager🡪 give a name and location to be saved. Change the toolchain to SW4STM32 and open the project.
* To setup wifi files, take the folder named “DISCO\_L475\_IOTA\_wifi” and move it to Src. Add the build path in project properties as mentioned in the below screenshot
* For setting up the files for reading temperature from HTS221, we need to do the changes as mentioned in application sheet 4.
  1. Copy the *STM32CubeL4/Drivers/BSP/B-L475E-IOT01 folder from STM32\_FW\_L4\_V1.13.0 package*
  2. In the generated project, create a folder *L4\_IOT\_Sensors/Drivers/BSP*. Paste the copied folder there.
  3. Copy the *STM32CubeL4/Drivers/BSP/Components folder*. Paste it under *L4\_IOT\_Sensors/Drivers/BSP*.
  4. Optional cleanup of working directory: as only HTS221 temperature sensor is used, some other files and folders already copied to the working directory may be removed
  5. Under *L4\_IOT\_Sensors\Drivers\BSP\B-L475E-IOT01*, keep only the following files:



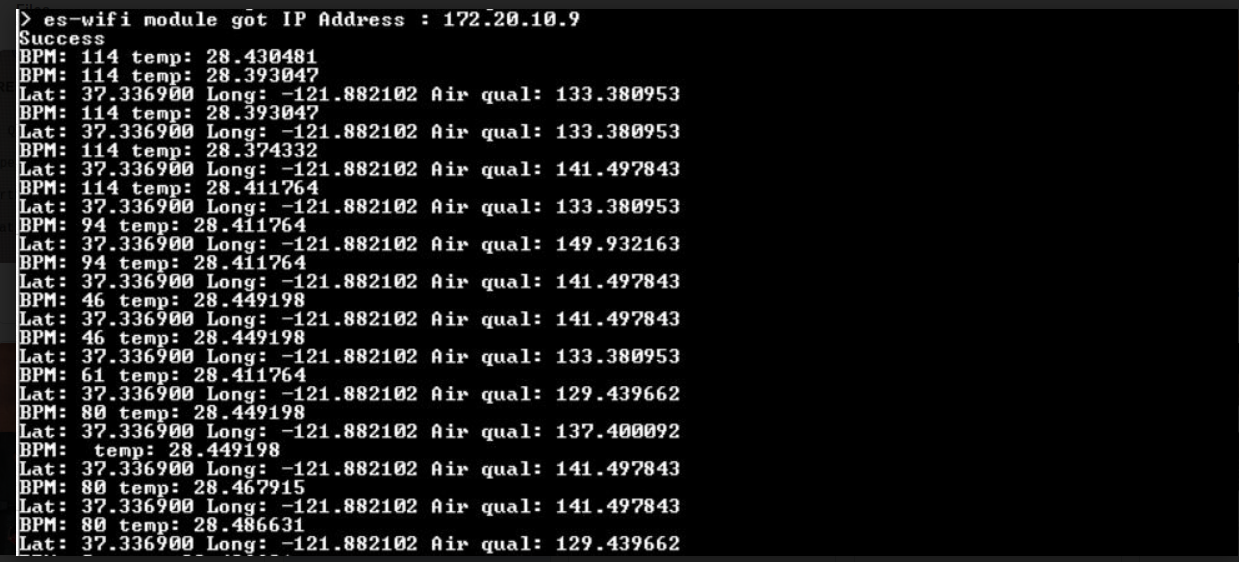
* 1. Under *L4\_IOT\_Sensors\Drivers\BSP\Components*, keep only the following folders:



* 1. From Project menu or File menu, go to Properties > C/C++ Build > Settings > Tool Settings > C Compiler > Directories. Add “../Drivers/BSP/B-L475E-IOT01” and “../Drivers/BSP/Components/hts221” paths.



Results & Evaluations



* We added printf statements in the task and function calls to monitor the sensor data. We successfully extracted and displayed the proper GPS location
* The air quality was varying between 100-500 ppm since we have kept it for powered on for 24-48 hours, so the correct environmental air quality value saturates at 400 ppm
* Recored temperature was providing room temperature values between 26-29 degrees.
* The heartrate was at the optimum range between 50-100 beats per minute.
* Wifi UDP connection is the first step that takes place which takes roughly about 5 seconds to connect to the mobile hotspot.
* When I performed a loopback test and checked the output at terminal, I was able to get the sent data in form of characters which concluded that data was successfully sent using UDP

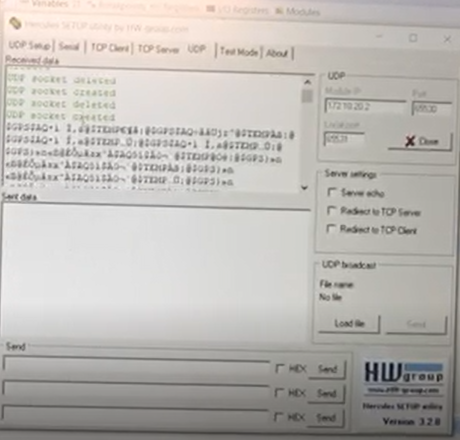


Figure 9 Loopback test in Hercules

Learnings

* With this project, I gained knowledge about using STM32 CubeMX and generating the code for custom applications
* I also got the clear understanding on working of device drivers and communication protocols like UART, I2C and Wifi UDP
* Understanding of ADC parameters and calibration of air quality sensor
* Learned how to read datasheets and use it to develop code for embedded applications
* Gained the conceptual understanding and basics of FreeRTOS task creation, execution and delays

# 6 Conclusions

* Safety of firefighters is our priority. This can be done by equipped firefighters with necessaryelectronics. But at the same time this should have less weight.
* We have developed the product for the firefighters and their team members, keeping their requirements in mind the major market segment of the product aims to be Fire Department, but it can be extended to various communities, military and sport adventurous groups, by adding or subtracting some sensor controllers as per their requirements. Also, weight of this project is very less.
* This product consists real-time location facility, because of that firefighter marshal will be able to monitor all its team members' location. Also, this product is embedded with sensors like temperature, heartbeat sensor and air quality sensor, so if situation near any firefighter becomes harsh, firefighter marshal will come to know and guide its team members, evacuate the area.
* We can expand the domain to medical applications where blood pressure sensor, sensor to measure glucose, etc can be integrated and monitored by the doctors.
* Currently, the project is delicate for firefighters to carry due to many components and wiring and sometimes GPS doesn’t get locked in indoors conditions, this project can be improved by scaling down and using a reliable GPS module.
* The push-to-talk service can be given a cloud support where it could maintain the history of recorded data which can be used by fire department for investigation purposes.

# 7 Read me

* Sensor configuration code: <https://drive.google.com/drive/u/1/folders/1QE9_Px2GQo1oCQJwJzYojijmYih71whT>
* Output Screenshots and Video: <https://drive.google.com/drive/u/1/folders/11SZLQJmdAh7eOEnA8qV28_tYKQn2EdpC>
* Wifi code: <https://drive.google.com/drive/u/1/folders/1Ccg56wygCxr5Ruv1X05HTCV6udJ3a3FJ>

# 8 References/Citations

[1] MQ-135 Datasheet: <https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf>

[2] MQ-135 Calibration: <https://www.researchgate.net/publication/328875972_Influence_of_Temperature_and_Humidity_on_the_Output_Resistance_Ratio_of_the_MQ-135_Sensor>

[3] Adafruit Ultimate GPS Breakout - 66 channel w/10 Hz updates - Version 3: <https://www.adafruit.com/product/746>

[4] Use of sensors on B-L475E-IOTA: <https://www.st.com/content/ccc/resource/training/technical/product_training/group0/05/4b/02/4c/27/64/45/09/STM32-Education_step4/files/STM32-Education_step4.pdf/jcr:content/translations/en.STM32-Education_step4.pdf>

[5] MAX30105 Particle and Pulse Ox Sensor Hookup Guide: <https://learn.sparkfun.com/tutorials/max30105-particle-and-pulse-ox-sensor-hookup-guide/all>