Team 13

REMOTE RANGER

Team:
Carter Rice, Rohun Patel, William Archer

Faculty Advisor:
Tania Khanna

I Executive Summary

Our group is working on Remote Ranger because we want to improve the effectiveness of search and rescue operations and develop a product that can be scaled into different areas. When we began sorting through the projects we could work on, we threw several ideas on the drawing board and came to two realizations that molded our ultimate project vision: hiking search and rescue is long overdue for improvement and our project should result in a tangible product at the end of the year. The combination of these two ideas brought us to Remote Ranger, and the work put in throughout the semester has brought us closer to reaching our goals. We determined that the foundation of the project could be done using the SigFox IoT network, because it fit perfectly for our system requirements. Low bandwidth, low power, and long range set the standard for our project that we started from in the semester.

The project itself incorporated several different logistical requirements that were not all required of past projects in our classes. We had several meetings throughout the year to both ensure that progress was being made and to flesh out issues in our project that could cause more problems going forward. Each of these meetings brought us closer to the product that we had at the end of the year, which shows that having periodic discussions about the progress made and progress that needs to be made helps to not only keep things moving, but to inspire the right iterations of the project to improve its final form. The budget constraints on the project did not end up being problematic, and most of the money spent was on multiple different units that did not end up as part of the final product, which is a natural part of the prototype process. The final cost of the development kit and battery gave us confidence that this product can be affordable when brought to scale.

II Overview of Project

Our group is working on Remote Ranger because we want to improve the effectiveness of search and rescue operations and develop a product that can be deployed at scale into different areas for various applications. When we began sorting through the projects we could work on, we threw several ideas on the drawing board and came to two realizations that molded our ultimate project vision: hiking search and rescue is long overdue for improvement and our project should result in a tangible product at the end of the year. The combination of these two ideas brought us to Remote Ranger, and the work put in throughout the year has culminated in a product that achieves this. The project was built using the SigFox IoT network, due to various design advantages that suited our system requirements. Low bandwidth, low power, and long range were the design aspects that our project prioritized.

The current tracking system used by the National Park Service is a physical paper tag given to hikers when they enter the park that keeps some personal information about the hiker, but is not electronic and cannot communicate with the park rangers. Its purpose is to be able to identify hikers in the event of a rescue effort. When hikers do not return after their estimated return time or are reported missing by relatives, friends, or other hikers, the rangers begin their search with only an estimation of the missing hiker's location. Cell coverage is usually weak or non-existent in national parks, so communicating with hikers is difficult. Alternatively, hikers could purchase an expensive and bulky GPS device or satellite phones that would allow them to track their location or call for help when it's needed. GPS-only devices are frequently unable to transmit their location in the case of an emergency, and satellite phones are an expensive and heavy solution that would not financially scale up to handle the number of hikers that visit national parks.

Remote Ranger aims to make this process much more efficient by allowing rangers to track a specific location of a hiker. If the hiker sends a distress signal to the rangers or if a hiker takes an abnormal amount of time to return, the rangers can start their search with a precise location resulting in rangers wasting less time in their efforts.

The widespread innovation that takes place in today's technology marketplaces has a huge impact on consumers and many products aim to make daily life easier for many people across the globe. However, this innovation does not often reach products that do not provide an immediate or probable profit in the future. There are many different areas where a new product could provide positive impact, but that impact may never arrive without a company already in a similar space looking to expand or if one is looking to enter a new space. Remote Ranger seeks to inject innovation into the National Park search and rescue operations to improve on the outdated system described above to provide the positive impact that today's technology is able to provide.

Outside of the specific problem being solved, there is other motivation from the group that led us to this type of project. We started with the goal of creating a tangible product that could have a real impact and that could be used across the country instead of just one specific space. We also wanted to build something that would serve as an extension on the skills learned through our curriculum, primarily our embedded systems backgrounds. We also wanted additional challenges on top of that that we have not necessarily tackled before, such as form factor and power optimization.

The objectives of the project can be summarized by meeting the goals of the problem and group's interests as described above. The main objective was to have a working final product that could be used in the National Parks. At the same time, we hope that this final product will provide a baseline for expanding into other search and rescue operations because the scope of Remote Ranger is currently aimed at National Parks search and rescue, but the impacts it could have are certainly not limited to it.

III Method of Solution

1. *Specification and requirements*

When we set out to build a hiking beacon for Remote Ranger, we set out a few primary goals for our device. The first is that it is a GPS-enabled tracking device for hikers in National Parks that would significantly increase the effectiveness of search and rescue operations.The design objectives we laid out included a long lasting battery life that would survive the duration of a hike, accurate location tracking of a hiker by pinging their location every 10 minutes, and a system that scaled to the capacity of hikers that national parks experience.

We hoped to maximize the range and battery life of the device. Communication methods that were researched included RF frequencies used in walkie-talkies, CB radio communication, and cellular signals. These methods were too power hungry for the range we desired, and had bandwidth and communication speeds that we found unnecessary. They also had infrastructure setup costs that would have been prohibitive. Figure 1 summarizes the different network solutions.

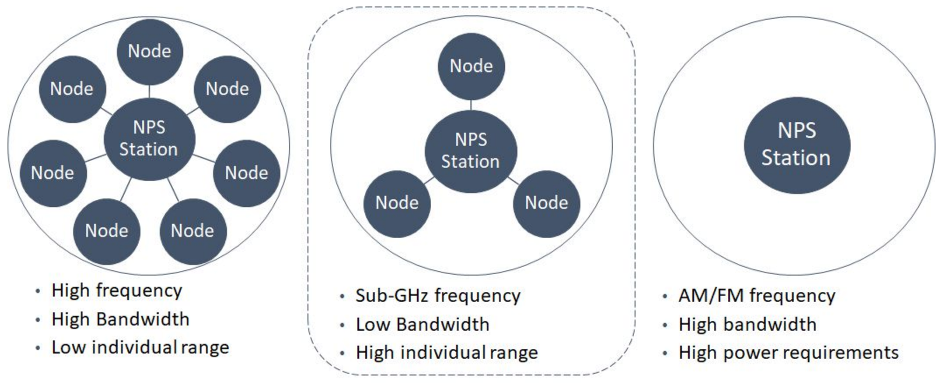


Figure 1: Comparison of infrastructure solutions and network configurations.

Shorter range systems would require too many receiving nodes, and longer range systems that operated with a single receiver were too power hungry. Looking into useful network choices for IoT devices (ones that are both low power requirement and low bandwidth), we decided to use the SigFox network because of a few key design parameters.

SigFox is a company based in France that claims to be the world's largest IoT network with wide coverage in Europe and expanding coverage in the United States. It has a variety of design choices that make it favorable for our specific application. First, it has a lightweight communications protocol that limits the overhead and size of data sent in each packet that is transmitted. The limit for data transmitted is 12 bytes of data. For reference, 1 GPS coordinate requires about 4 bytes of data to send, so the 12 bytes of data could be used to successfully transmit the latitude and longitude of a beacon in one ping. Sigfox has also removed signaling on the network between nodes and relies on pseudo random packet sending from devices to avoid collisions. This allows the network to quickly relay data from receiving SigFox stations to SigFox servers. Lastly, the SigFox network operates at an ultra narrow band. This narrow band results in a high sensitivity for the receiver and low power requirements for a transmitter.

These design parameters gave us a few key benefits that solidified the choice of the SigFox network. The first of these is the ability to have an extremely long range of communication. In urban areas, SigFox claims to get between 5 and 10km of range. In rural areas that range extends up to between 30 and 50km. We have also found claims that SigFox can reach up to 1000 km with direct line of sight. This led us to conservatively estimate a range of 10 to 20 km in most national parks or national forests. In addition, we initially looked into network setup costs for each communication range we considered. Since SigFox is planning on expanding coverage to the full continental United States, the cost of network infrastructure wouldn't be passed on to the parks. The scaling of the network is also a burden that is not placed on the parks, because the network is designed to handle a large capacity of devices. As parks adopt the tracker, there would be no issues with network traffic. Lastly, the low bandwidth requirements and low power requirements for transmission result in a longer battery life for devices on the network. Sigfox also has a low cost subscription that parks could subscribe to for all active devices. The subscription is not based on device traffic, it is instead based on device capacity, that is, the number of active devices registered on the network.

An important part of the design is the battery life of the device. We used a 3.7 V LiPo 1200 mAh battery with the board, which allowed us to have a battery life close to two weeks based on location pings every 10 minutes. Based on our measurements and the specs of the final dev kit we used, GPS takes about 27 mA, Sigfox takes about 54 mA, and the sleep mode uses about 1 mA of current. Based on 15 seconds of GPS mode, 30 seconds of transmission, and the remainder of time sleeping, we arrived at 14 days of battery life for the unit. We can also extrapolate this data to show what the battery life would be with a different amount of time between pings. This data is shown in the graph below:

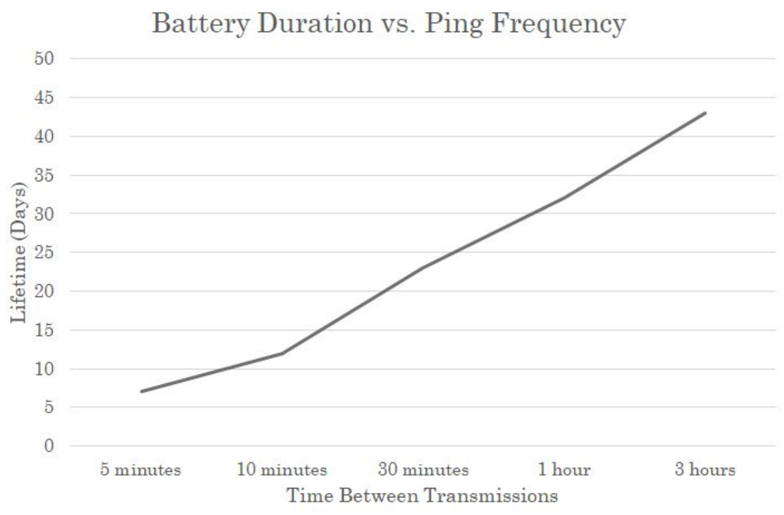


Figure 2: Battery lifetime compared to location sending intervals

Each device's lifetime should last a few years providing they were durably built. This would give an optimal level of device turnover for parks and users. Producing these devices at large volumes should also result in cost savings that put the final cost of the device at a reasonable cost of $26.35 that a park would be able to afford. We also envision this system being modified for an application in search and rescue during natural disasters. It could potentially be deployed by being distributed to people in the path of an impending disaster to aid rescue efforts.

2. What specific classes and knowledge does the project depend on?

There are a few different classes offered within the ESE department that are applicable to this project. The primary two are ESE 350 (Embedded Systems) and ESE 578 (RFIC). Every team member has taken ESE 350 and Carter has taken ESE 578. The system setup for the project can be split into two different components to get a better idea of how each course applies. The central Sigfox module that the device will use interacts with a microcontroller on the board, which has to execute the right steps to access the functioning of the module. Additionally, the C++ programming that was covered in ESE 350 is the same language being used to program the current MCU. Memory, timing, and processing constraints that came into play during the course are also being tested with this board, which makes it very beneficial to come into the project with prior experience. The knowledge gained in ESE 578 applies to the actual wireless signal being used to send information on the network, which came into play earlier in the semester. The relationship between signal strength, power requirements, and antenna size were all touched on in the course and helped the project get narrowed down to the Sigfox network and communications instead of alternatives such as AM/FM or 1+ GHz signals.

IV Self-learning

In order to complete this project, we had to familiarize ourselves with the benefits of various networks. This involved learning bandwidth and power requirements of each type of communication. In addition, we purchased a development kit that we had to learn how to prototype with. It consisted of a debugging board and a SFM20Rx board. The SFM20Rx board contained various peripherals including an NFC, temperature sensor, pressure sensor, GPS antenna, SigFox antenna, Wifi antenna, and BLE antenna. We had to learn how to wire the two boards together and flash the debugger to upload code to the SFM20Rx board. This included becoming familiar with git to access documentation, downloading and learning how to use uVision, the compiler for ARM devices, and the windows powershell to flash the debugger. We also had to familiarize ourselves with the API provided in the documentation. We continued with our self learning and were able to successfully program the board to ping it's GPS coordinates and send them through the SigFox network. Lastly, Will spearheaded the development of the web application, the skills to do which he learned on his own.

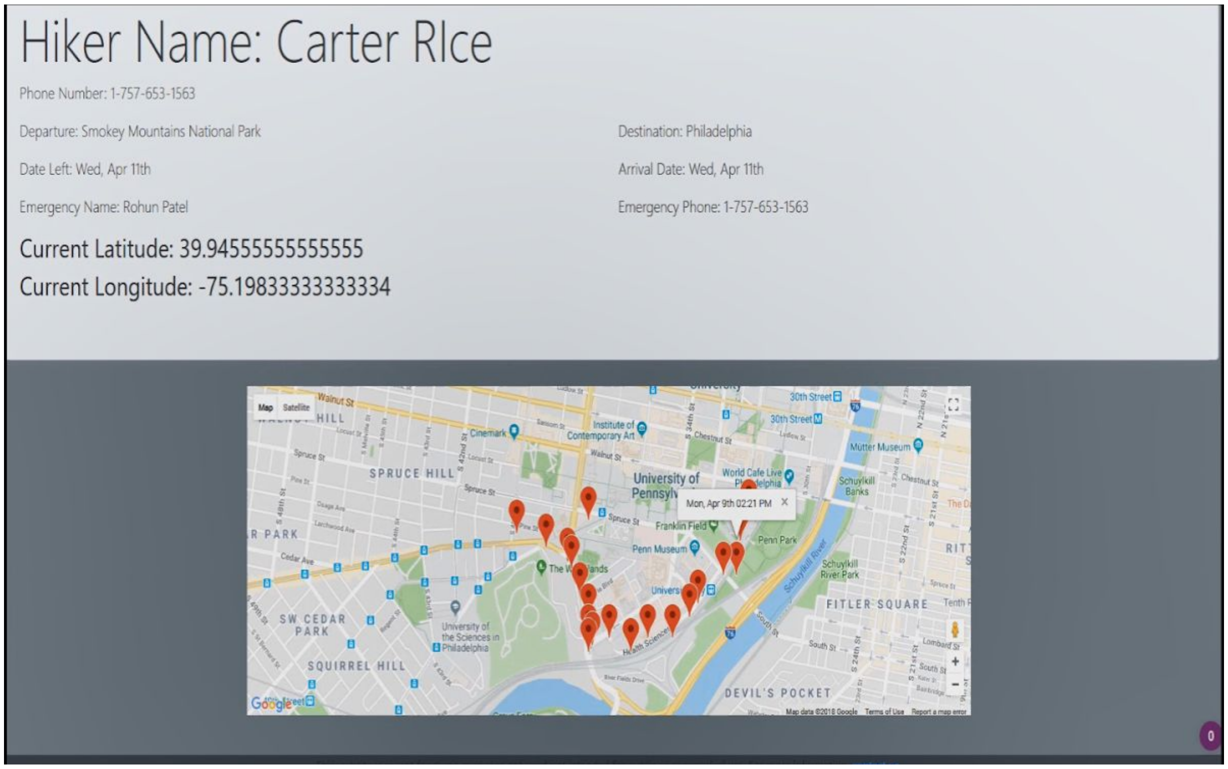
V Design and Iteration

Once we settled on the SigFox network to use, we started our project using a WSSFM20R2 dev kit from Sea Slug labs for initial testing. We were able to successfully configure the unit to get the GPS location and send it over SigFox, but we were having trouble with the SigFox communication in some of the areas surrounding engineering, and nothing worked indoors (including GPS). This led us to our fall demo, which was a proof-of-concept for the GPS and the communication interface.

Moving into second semester, we started with the goal of condensing the GPS payload to a smaller size that would provide higher accuracy as well as extra room for other data that we would like to send, most notably the battery level. In order to test and fix this, we purchased a network emulator that allowed us to quickly test how messages would be received without waiting for them to be processed through the sigfox backend. This allowed us to condense the GPS payload so that we could transmit the most precise coordinates possible. We also acquired a smaller dev kit, the iHere, from Sea Slug labs for this part of the project. One of the notable design differences that was key was the size of each antenna. For the fall demo, our GPS antenna was a little larger than a quarter and the cable was roughly 20 feet long, while the SigFox antenna was about 7 inches long and quite bulky. The iHere used a different GPS antenna (common for PCB mounting) that was a little smaller and was soldered to the board with no extra cord. The SigFox antenna was considerably smaller than the previous version and now took up about one square inch of space on the board with a height of about half an inch, which allowed for the final case design to be considerably smaller and provide a better final product.

For the GPS coordinates, we successfully parsed the NMEA data that we received from the GPS protocol and sent it using 4 bytes for longitude and 4 bytes for latitude This was comprised of one byte to signal north or south and east or west, while the second byte was degrees, the third byte was minutes, and the last byte was seconds. This provided GPS accuracy down to 10 feet, which was ideal for the system design we needed.

∫Throughout fall semester and into the beginning of spring semester, we had callbacks set up on the sigfox backend that sent us emails of the location estimation (done by sigfox) and the actual message (the board's GPS location). To look at the GPS data and see if it was right, we had to take the message and put it in a google sheet that converted each of the bytes into the corresponding part of the GPS coordinates, and we manually mapped the position after that. Needless to say, this would not be efficient for a demo of the product. Using an AWS callback, the data was then sent to our AWS host as well and the web app was created to have a user friendly display with all the information needed in one place. The main dashboard shows each of the registered devices and the information associated with it, as well as an indicator that shows if it has transmitted in the last 30 minutes. The map on the dashboard shows the latest location for each device registered on the app. Each device also has its own page that displays its information as well as a map that shows its location history with timestamps for each location ping, which allows rangers to get a visual for the path that the hiker is taking.



Hiker Name: Carter RIce

Phone Number: 1-757-653-1563

Departure: Smokey Mountains National Park

Destination: Philadelphia

Date Left: Wed, Apr 11th

Arrival Date: Wed, Apr 11th

Emergency Name: Rohun Patel

Emergency Phone: 1-757-653-1563

Current Latitude: 39.94555555555555

Current Longitude: -75.19833333333334

Figure 4: A screenshot of the device overview page for the web application.

The original case design we used for demo day is 4"x2"x2", which is rather bulky, but future iterations would use a case with a 3"x2"x1" design that is much more suitable for hikers to use. The reason for this is that we designed the original case to fit the iHere, battery, booster/charger circuit, and USB cord since we did not have the proper header for the battery to attach to the cord and there was plenty of extra room even with all 3 components. The next design would have a very short connection from the battery to the board with no USB cable or booster board, allowing the case design to be very small.

VI **Societal, global and/or economic impact.**

1. Context of the project

One of the benefits of the target impact of this project is that hiking and search and rescue are not niche activities that are unique to the National Park Service, let alone the U.S. In this way, Remote Ranger can be applied to very similar situations around the world to improve search and rescue operations for hikers and rangers everywhere. While the economic impact of this project may not seem obvious, it is essentially providing an alternative to equipping every hiker with a commercially-available GPS unit. These units can cost hundreds of dollars, making it infeasible for such a rollout to occur. Remote Ranger provides a much more economic alternative to make the concept for rollout. In addition, the cost of devices and a SigFox subscription would be far less than the cost savings that this system is driving. Shorter search and rescue operations result in significantly lower costs for national parks to incur and lower loss of life. An additional note on the project is its ability to scale into different search and rescue operations, most notably hurricane relief in the US and Caribbean. The device could be distributed to people that do not evacuate and could send a distress beacon if they need to be recovered, which can make the job of emergency services much more efficient.

2. Ethical Issues

Hikers could potentially be concerned about their location being tracked while in the park. If the project is clear enough about the frequency of the GPS pings and clear that they are for the purpose of safety in the event of a search and rescue, there should be no issue with the use of the device from the hiker side. The national park service that uses the tracker should have no ethical issues with the proposed system architecture or devices. Since SigFox is responsible for installing infrastructure across the continental United States, the park service should be able to use it without any additional setup. In the worst case if coverage is deemed too sparse, the park could partner with SigFox to build a few extra receiving stations, but this event remains highly unlikely and will likely be of negligible cost.

VII Summary of Meetings

- Meeting with Tania 9/19
    - Attended by all group members
    - Discussed beginning idea of project and motivation
        - Initial desire to aid disaster relief
        - Breaking down the problem into search and rescue
    - Discussed first research steps
        - Look into key parameters of range and battery life
        - Look into various communications options

- Meeting with Sid, Jorge, Neil 9/20
    - Attended by all group members
    - Discussed project motivation, direction, and scope
    - Discussed implementation
        - Preliminary research ideas of communication methods
        - Preliminary discussion of key parameters and device specs

- Meeting w DeHon 9/22
    - Attended by all group members
    - Discussed research on communication methods
        - Piggybacking off of cellular network
        - Piggybacking off of walkie talkie networks or CB radio
        - Creating our own network
    - Further research into IoT networks
    - SigFox was suggested

- Meeting with Max 10/13
    - Attended by all group members
    - Discussed project progress, development kit, milestone development and refinement
    - Discussed additional research into FCC compliance
        - Discussed current understanding and implications of requirements

- Meeting with Tania 12/1
    - Attended by Carter and Rohun
    - Discussed progress with prototype using the development kit and refined device design specs
    - Discussed an appropriate presentation for upcoming fall demo day
    - Discussed appropriate milestones for the spring semester

- Meeting with Sid, Jorge, Neil 12/1
    - Attended by Carter and Rohun

    - Discussed milestone progress with prototype and updated design specs of the device and proposed system
    - Discussed the plan for demo day
    - Discussed milestones for the spring semester

- Meeting with Jorge, Leroy 1/25
    - Attended by Carter, Rohun, and Will
    - Discussed progress from fall demo day and plans for the rest of the semester
    - Discussed plans to buy individual modules and proto-board before pcb

- Meeting with Sid and Jorge 2/20
    - Attended by Carter, Rohun, and Will
    - Discussed need to get pcb out before spring break
    - Discussed plans for testing unit over next month

- Meeting with Leroy and Sid 3/28
    - Attended by Carter, Rohun
    - Discussed standards for project

VIII Final schedule with milestones

Milestone	Carter	Rohun	Will
Establish SigFox as protocol			September
Reach out to NPS/other parks		October	
Order original dev kit	October		
Program dev kit for winter demo	November	November	November
Winter demo	December	December	December
Urban testing		January	
Iterate to iHere	February	February	
Develop PCB			February
Minimize GPS data size		March	March
Rural testing	April		
Finalize AWS/SigFox backend			April
Complete web app front end			April
Final demo	April	April	April

IX **Discussion of teamwork**.

The team members have similar skill sets, with each of us having taken ESE 350 and having a background in hardware development. Fall semester, Will was responsible for determining the best way to develop on the board, helping determine and debug the installation of the IDE and help flash the board. Carter and Rohun took the lead on developing on the board to ping GPS and send the signal through the SigFox network. Rohun and Carter also took the lead on developing the team poster and presentations.

During spring semester, Will took the lead on the GPS parsing issue we were having as well as the web app that we used for the demo, which was very important since it was the method we used to demonstrate the capabilities of the project for a live demo. Carter and Rohun focused on the rest of the software for the board and the testing that was done. Carter took the lead on the logistics for the team and preparing the presentations throughout the semester.

X **Budget**

Our fall semester budget comprised of a WISOL Quad EVK WSSFM20R2 board that cost $130, a Nordic NRF52-DK debugger board that cost $40, and a 10-pin J-link cable that cost $3. These components provided the baseline demo that we needed for this semester in order to present a proof-of-concept for the product design. While these prices are high for individual units, they were not outside our expectations since we were using a development kit and debugger board, both of which have several capabilities that are meant for any developer, but will not be needed for our project.

Spring semester had a higher budget due to the iterations that we will do on the project. Starting from the baseline demo we provided, we purchased another development kit ($130) to assist with field testing, and 3 wisol modules ($26), antenna kits ($30 total), and debuggers ($90 total) in our initial attempt to move off the development kit. Additionally, we purchased a sigfox network emulator costing $160 that saved us a tremendous amount of time by simulating the sigfox network in Detkin lab. We finally settled on ihere development boards ($84 total) for our final demo and proceeded to purchase the batteries and voltage boosting circuits to complete the demo and enable our outdoor testing ($50 total)

XI **Work for Second Semester**

The main deliverables for spring semester were the smaller dev kit and the web app we used for demo day. The smaller dev kit was crucial because it made the design feasible in terms of size, while the original dev kit was too large and awkwardly shaped to reasonably be attached to a backpack. The web app was also critical because it gave a view of the functionality that this project is aiming for. We wanted to be able to show that each device has an ID that can be assigned to a person, and then that person can then be tracked with a map visual that also shows how long it has been since the device has transmitted.

While we intended to make a custom PCB for the project, we did not end up doing so for two reasons. The main reason was the difficulty we experienced tracking down the correct antenna subcomponent to properly impedance match the circuit as well as the need for very small components to fit together on the board. This lead to us running out of time to produce an adequate unit that we could realistically iterate on in time for demo day. The second reason was the smaller dev kit that we found. We did not know that it existed prior to spring semester and it ended up having the vital components we needed, so it worked well enough to program and use in our demo. The smaller antenna size was a huge plus since we were planning on using the bulkier antenna on our PCB. If we were to go forward on this project, we would make a custom PCB tailored to the application's needs (National Parks, natural disasters, etc.) with the smaller PCB antenna found on the smaller dev kit.

XII **Standards and compliance**

Since we are developing on the SigFox network which operates in the unlicensed band between 902 and 928 MHz with a 915MHz center frequency. FCC compliance with transmitting in this band are handled by the Sigfox protocol. Additionally, FCC mandates that our transmitter has to produce an electric field weaker than 50 mV/m at a distance of 3 meters from the transmitting antenna. Since we are such a low power device, this isn't an issue we ran into. In addition, we have to be mindful that the third, fourth, and fifth harmonics fall in restricted bands so we need to also ensure that the transmitting SigFox antenna does not accidentally transmit these frequencies. Regarding the charging, discharging, and safe handling of the lithium ion batteries, we have to comply with IEC 62133, UL 2054, and UL 9990. These standards all ensure that the batteries safely recharge, discharge and handle physical stress and fire in an appropriate way. It also mandates a safe charging cable and safe use in households. We wanted to ensure that the standards and quality maintained by battery manufacturers are not tarnished or lowered by our use of the components. We created a case with the safe storage of these components in mind. The user data from the GPS pings is transmitted securely using Sigfox's encryption and the data is stored and encrypted using HTTPS on Sigfox servers.

XIII **Conclusion**

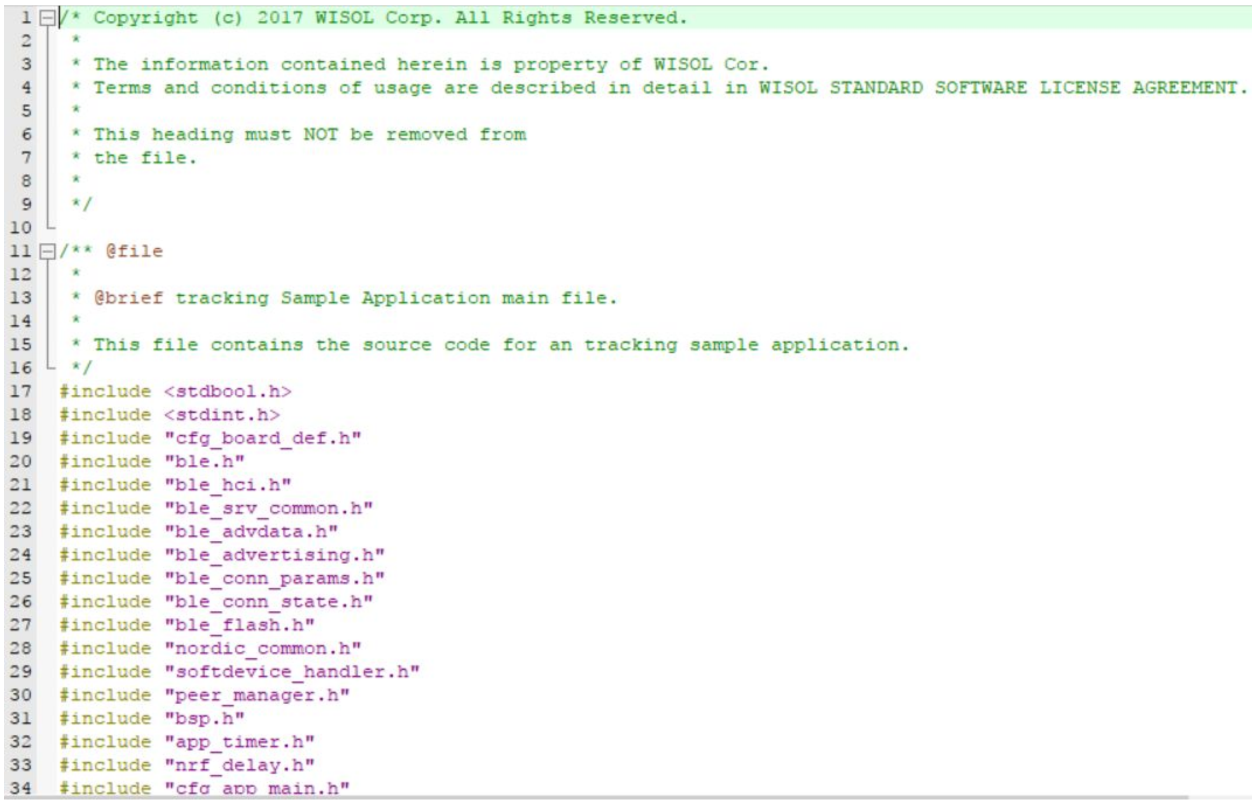
The work that we put in this year culminated in a proof of concept for the remote ranger system that serves as a key stepping stone to a product that we could legitimately market. We began the year by using a development kit with tons of features in order to lock down the functionality of GPS and Sigfox, which allowed us to iterate to a smaller board that served the functions we needed, all while maintaining our goals of low cost and long battery life for the system. Additionally, the small case design that we came up with is perfect for the system since it is lightweight and small enough that it would cause no obstructions for hikers.

One of the main challenges we faced was the design of the PCB that we wanted to make for the final demo. While we all had some PCB experience, the components on the board were much smaller than we were used to and we encountered a big problem when trying to design the antenna circuit to impedance match. We also faced a challenge in contacting the national park service to get their input on the project. This isn’t much of a surprise for the team since the premise of the project is based on the NPS being on a rather strict budget, so they don’t necessarily have the resources to look into something like this. All this has taught us the importance of patient and constant iteration on a project since it is very difficult to know everything at the very start and there are challenges along the way that need to be dealt with, and some that are very hard to overcome.

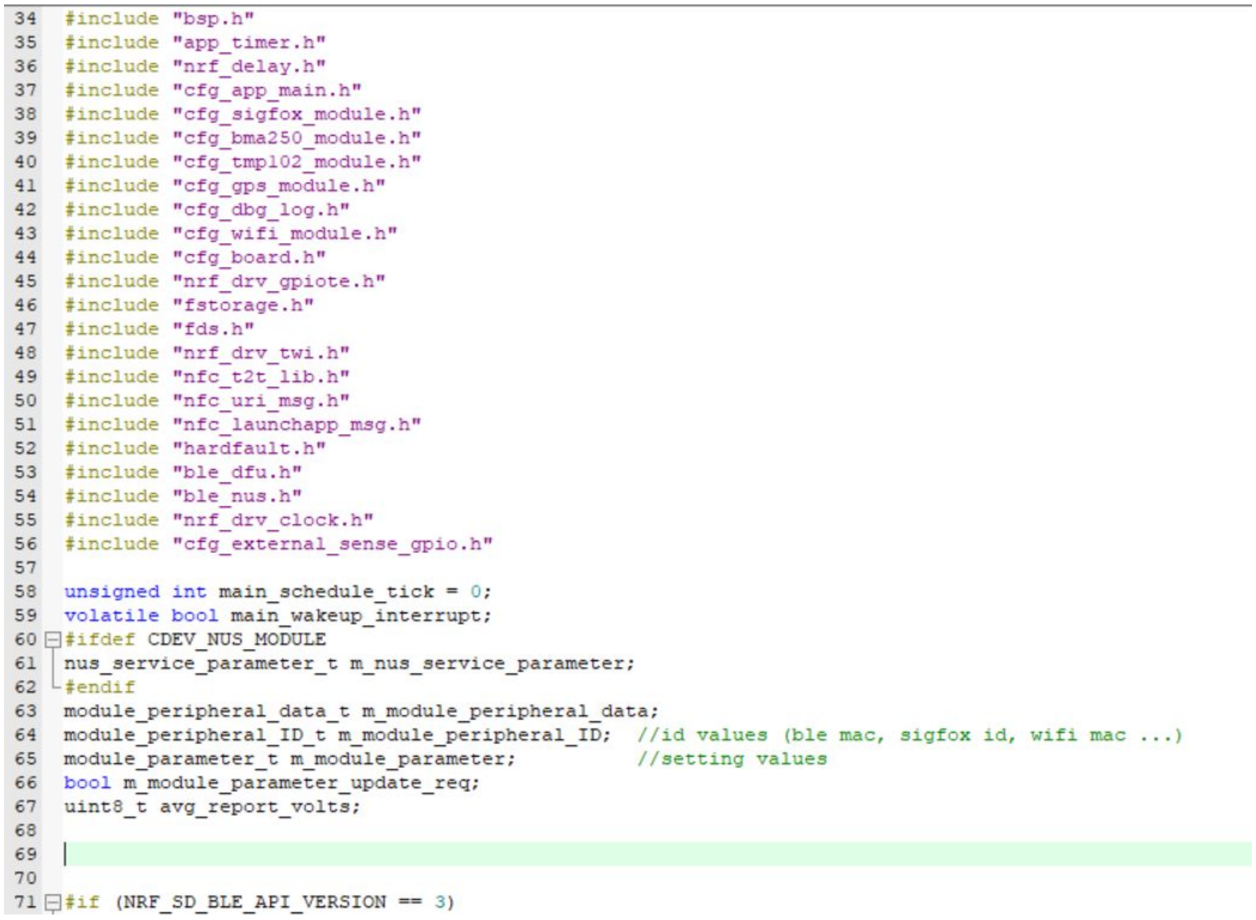
The remote ranger project has a potential future outside of this year. It can be applied to natural disasters to aid in other areas of search and rescue, while it could even make its way into private companies, such as vacation companies that need to track their clients or assets in rural environments. The major challenge that we face is the rollout of the Sigfox network, which is not yet ready for our desired application in the US, but could be used in that fashion in many areas of western Europe. Creating a custom board and bringing the product to scale allows us to have the affordable and effective solution that we sought to create in September.

Appendix A: Code

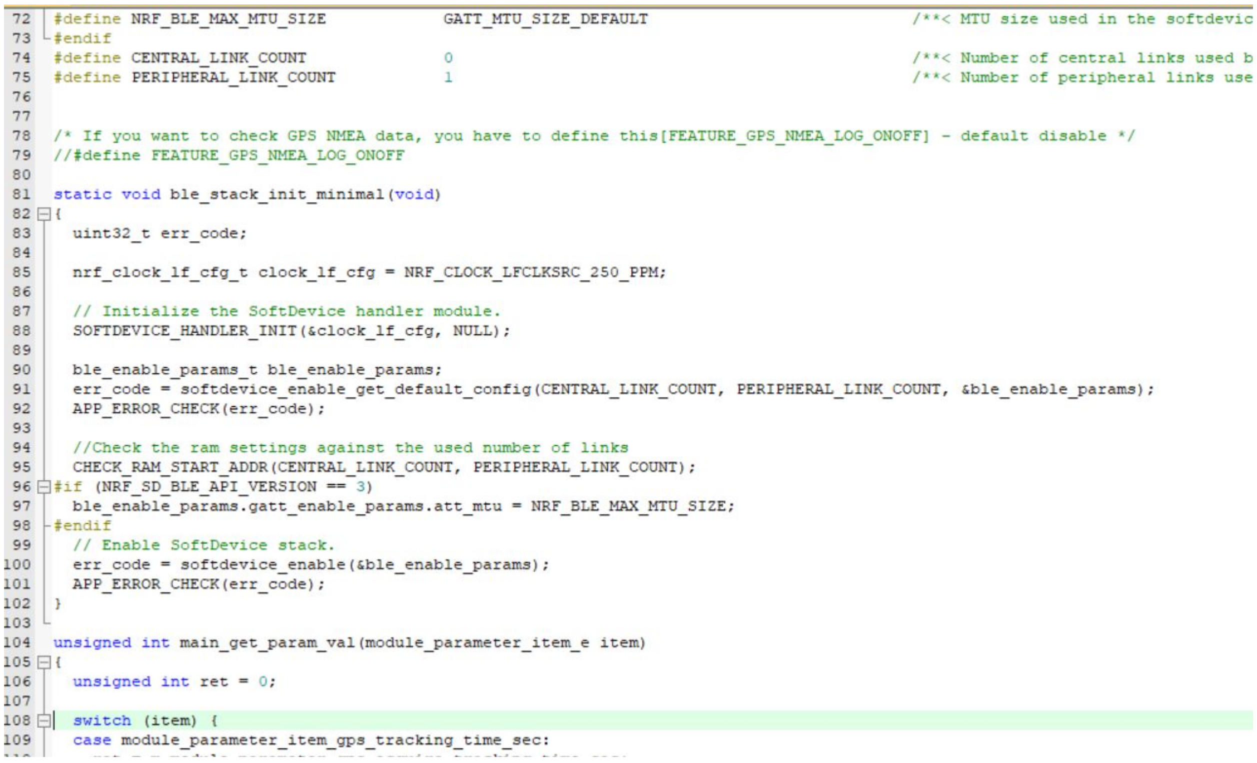
Code uploaded to the iHere board to complete a GPS ping and send the result through the SigFox network.



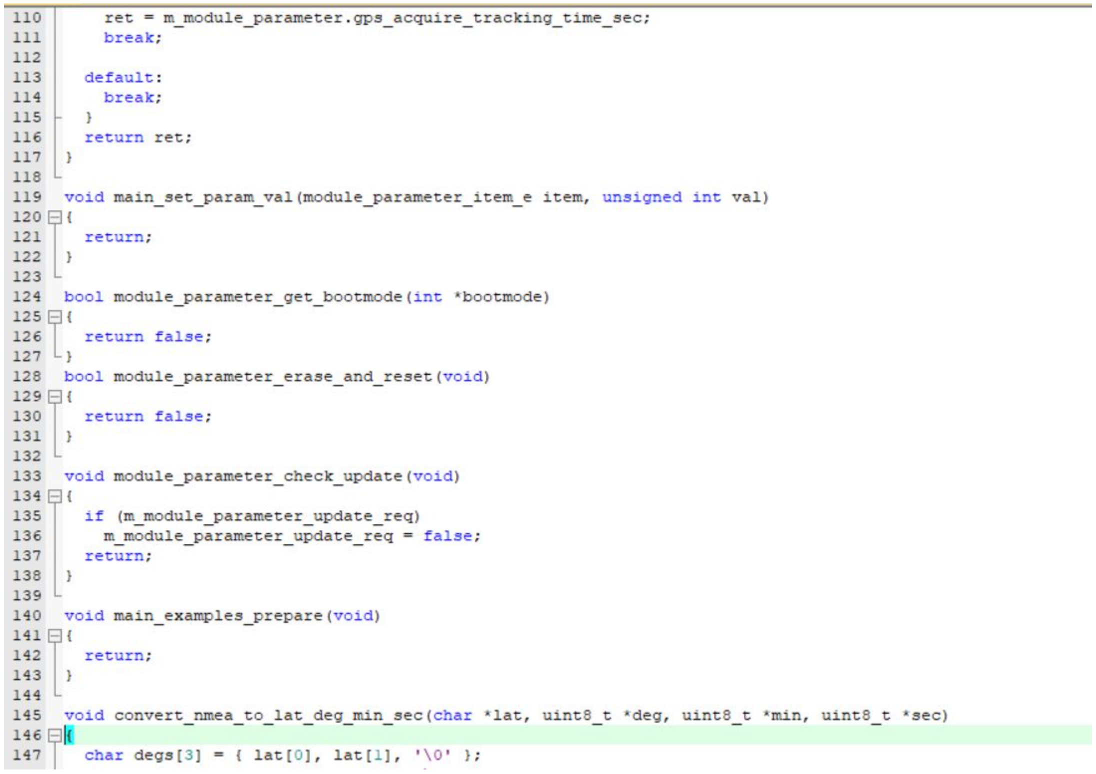
```
/* Copyright (c) 2017 WISOL Corp. All Rights Reserved.
 *
 * The information contained herein is property of WISOL Cor.
 * Terms and conditions of usage are described in detail in WISOL STANDARD SOFTWARE LICENSE AGREEMENT.
 *
 * This heading must NOT be removed from
 * the file.
 *
 */

/** @file
 *
 * @brief tracking Sample Application main file.
 *
 * This file contains the source code for an tracking sample application.
 */
#include <stdbool.h>
#include <stdint.h>
#include "cfg_board_def.h"
#include "ble.h"
#include "ble_hci.h"
#include "ble_srv_common.h"
#include "ble_advdata.h"
#include "ble_advertising.h"
#include "ble_conn_params.h"
#include "ble_conn_state.h"
#include "ble_flash.h"
#include "nordic_common.h"
#include "softdevice_handler.h"
#include "peer_manager.h"
#include "bsp.h"
#include "app_timer.h"
#include "nrf_delay.h"
#include "cfg_app_main.h"
```



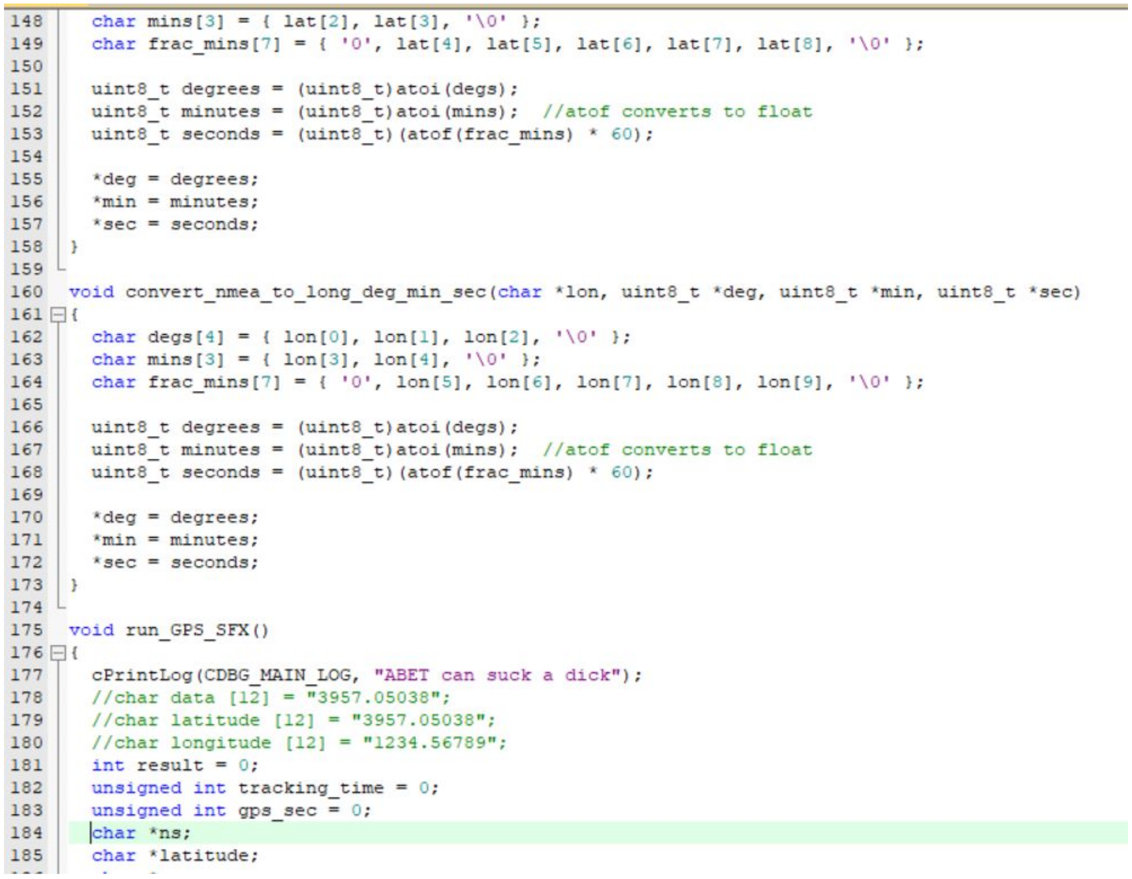
```c
#include "bsp.h"
#include "app_timer.h"
#include "nrf_delay.h"
#include "cfg_app_main.h"
#include "cfg_sigfox_module.h"
#include "cfg_bma250_module.h"
#include "cfg_tmp102_module.h"
#include "cfg_gps_module.h"
#include "cfg_dbg_log.h"
#include "cfg_wifi_module.h"
#include "cfg_board.h"
#include "nrf_drv_gpiote.h"
#include "fstorage.h"
#include "fds.h"
#include "nrf_drv_twi.h"
#include "nfc_t2t_lib.h"
#include "nfc_uri_msg.h"
#include "nfc_launchapp_msg.h"
#include "hardfault.h"
#include "ble_dfu.h"
#include "ble_nus.h"
#include "nrf_drv_clock.h"
#include "cfg_external_sense_gpio.h"

unsigned int main_schedule_tick = 0;
volatile bool main_wakeup_interrupt;
#ifdef CDEV_NUS_MODULE
nus_service_parameter_t m_nus_service_parameter;
#endif
module_peripheral_data_t m_module_peripheral_data;
module_peripheral_ID_t m_module_peripheral_ID;  //id values (ble mac, sigfox id, wifi mac ...)
module_parameter_t m_module_parameter;          //setting values
bool m_module_parameter_update_req;
uint8_t avg_report_volts;



#if (NRF_SD_BLE_API_VERSION == 3)
```



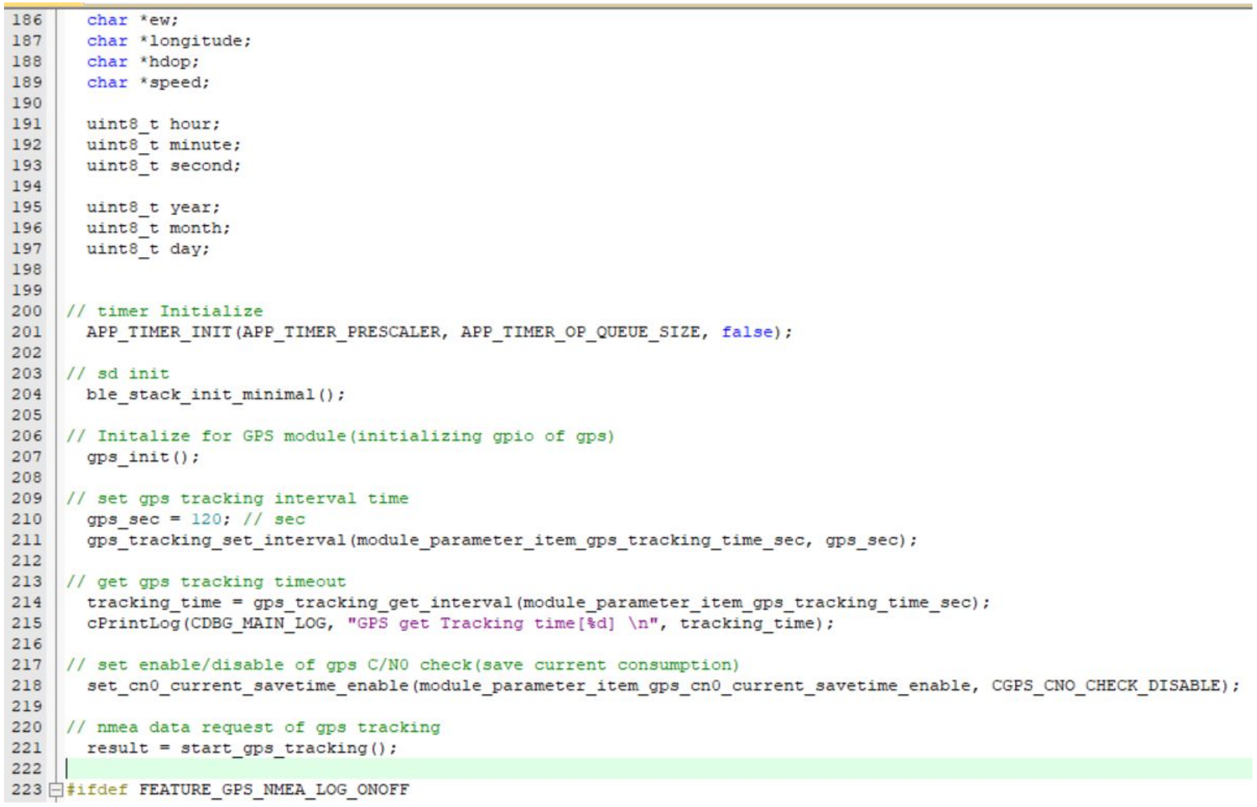
```c
#define NRF_BLE_MAX_MTU_SIZE            GATT_MTU_SIZE_DEFAULT                              /**< MTU size used in the softdevic
#endif
#define CENTRAL_LINK_COUNT              0                                                  /**< Number of central links used b
#define PERIPHERAL_LINK_COUNT           1                                                  /**< Number of peripheral links use


/* If you want to check GPS NMEA data, you have to define this[FEATURE_GPS_NMEA_LOG_ONOFF] - default disable */
//#define FEATURE_GPS_NMEA_LOG_ONOFF

static void ble_stack_init_minimal(void)
{
  uint32_t err_code;

  nrf_clock_lf_cfg_t clock_lf_cfg = NRF_CLOCK_LFCLKSRC_250_PPM;

  // Initialize the SoftDevice handler module.
  SOFTDEVICE_HANDLER_INIT(&clock_lf_cfg, NULL);

  ble_enable_params_t ble_enable_params;
  err_code = softdevice_enable_get_default_config(CENTRAL_LINK_COUNT, PERIPHERAL_LINK_COUNT, &ble_enable_params);
  APP_ERROR_CHECK(err_code);

  //Check the ram settings against the used number of links
  CHECK_RAM_START_ADDR(CENTRAL_LINK_COUNT, PERIPHERAL_LINK_COUNT);
#if (NRF_SD_BLE_API_VERSION == 3)
  ble_enable_params.gatt_enable_params.att_mtu = NRF_BLE_MAX_MTU_SIZE;
#endif
  // Enable SoftDevice stack.
  err_code = softdevice_enable(&ble_enable_params);
  APP_ERROR_CHECK(err_code);
}

unsigned int main_get_param_val(module_parameter_item_e item)
{
  unsigned int ret = 0;

  switch (item) {
  case module_parameter_item_gps_tracking_time_sec:
```



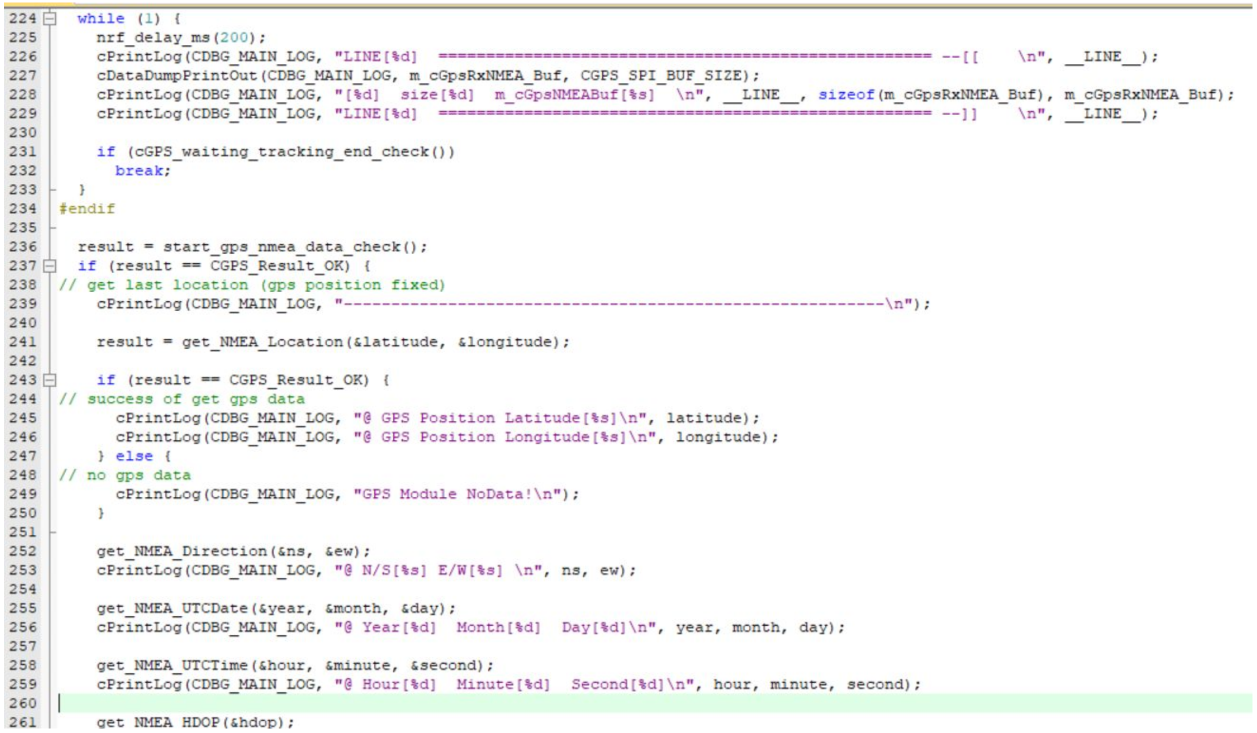
```
    ret = m_module_parameter.gps_acquire_tracking_time_sec;
    break;

  default:
    break;
  }
  return ret;
}

void main_set_param_val(module_parameter_item_e item, unsigned int val)
{
  return;
}

bool module_parameter_get_bootmode(int *bootmode)
{
  return false;
}
bool module_parameter_erase_and_reset(void)
{
  return false;
}

void module_parameter_check_update(void)
{
  if (m_module_parameter_update_req)
    m_module_parameter_update_req = false;
  return;
}

void main_examples_prepare(void)
{
  return;
}

void convert_nmea_to_lat_deg_min_sec(char *lat, uint8_t *deg, uint8_t *min, uint8_t *sec)
{
  char degs[3] = { lat[0], lat[1], '\0' };
```



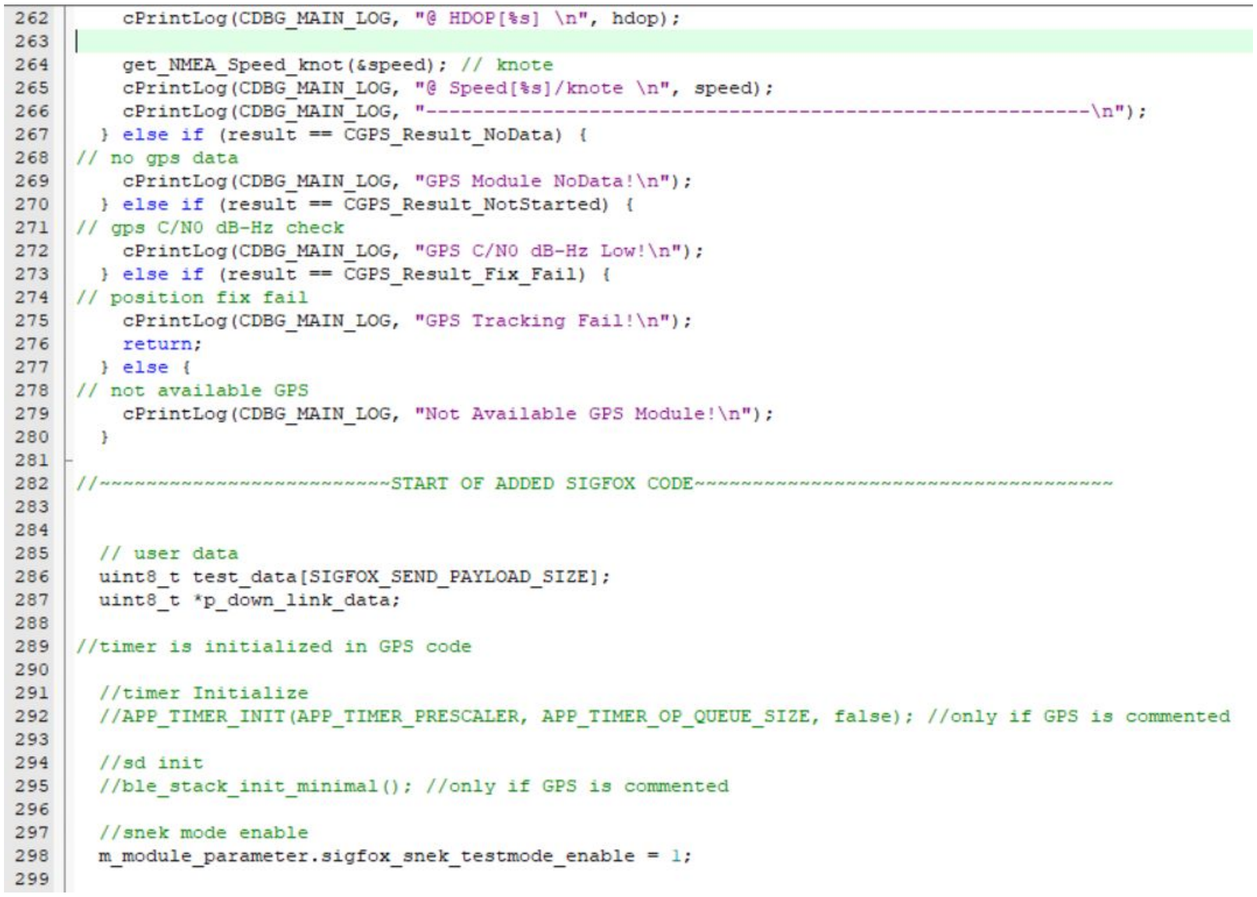
```
  char mins[3] = { lat[2], lat[3], '\0' };
  char frac_mins[7] = { '0', lat[4], lat[5], lat[6], lat[7], lat[8], '\0' };

  uint8_t degrees = (uint8_t)atoi(degs);
  uint8_t minutes = (uint8_t)atoi(mins);  //atof converts to float
  uint8_t seconds = (uint8_t)(atof(frac_mins) * 60);

  *deg = degrees;
  *min = minutes;
  *sec = seconds;
}

void convert_nmea_to_long_deg_min_sec(char *lon, uint8_t *deg, uint8_t *min, uint8_t *sec)
{
  char degs[4] = { lon[0], lon[1], lon[2], '\0' };
  char mins[3] = { lon[3], lon[4], '\0' };
  char frac_mins[7] = { '0', lon[5], lon[6], lon[7], lon[8], lon[9], '\0' };

  uint8_t degrees = (uint8_t)atoi(degs);
  uint8_t minutes = (uint8_t)atoi(mins);  //atof converts to float
  uint8_t seconds = (uint8_t)(atof(frac_mins) * 60);

  *deg = degrees;
  *min = minutes;
  *sec = seconds;
}

void run_GPS_SFX()
{
  cPrintLog(CDBG_MAIN_LOG, "ABET can suck a dick");
  //char data [12] = "3957.05038";
  //char latitude [12] = "3957.05038";
  //char longitude [12] = "1234.56789";
  int result = 0;
  unsigned int tracking_time = 0;
  unsigned int gps_sec = 0;
  char *ns;
  char *latitude;
```



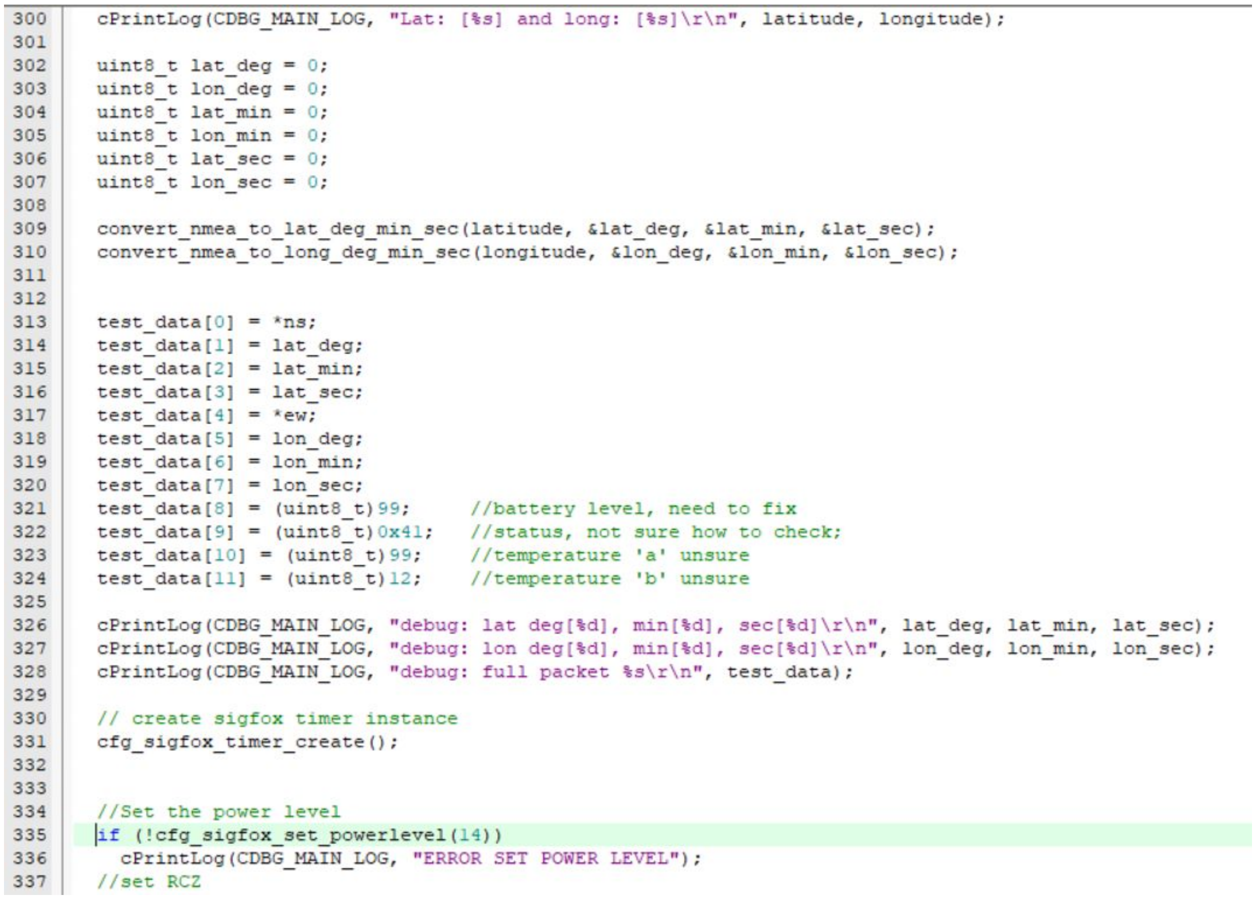
```c
  char *ew;
  char *longitude;
  char *hdop;
  char *speed;

  uint8_t hour;
  uint8_t minute;
  uint8_t second;

  uint8_t year;
  uint8_t month;
  uint8_t day;


// timer Initialize
  APP_TIMER_INIT(APP_TIMER_PRESCALER, APP_TIMER_OP_QUEUE_SIZE, false);

// sd init
  ble_stack_init_minimal();

// Initalize for GPS module(initializing gpio of gps)
  gps_init();

// set gps tracking interval time
  gps_sec = 120; // sec
  gps_tracking_set_interval(module_parameter_item_gps_tracking_time_sec, gps_sec);

// get gps tracking timeout
  tracking_time = gps_tracking_get_interval(module_parameter_item_gps_tracking_time_sec);
  cPrintLog(CDBG_MAIN_LOG, "GPS get Tracking time[%d] \n", tracking_time);

// set enable/disable of gps C/N0 check(save current consumption)
  set_cn0_current_savetime_enable(module_parameter_item_gps_cn0_current_savetime_enable, CGPS_CNO_CHECK_DISABLE);

// nmea data request of gps tracking
  result = start_gps_tracking();

#ifdef FEATURE_GPS_NMEA_LOG_ONOFF
```



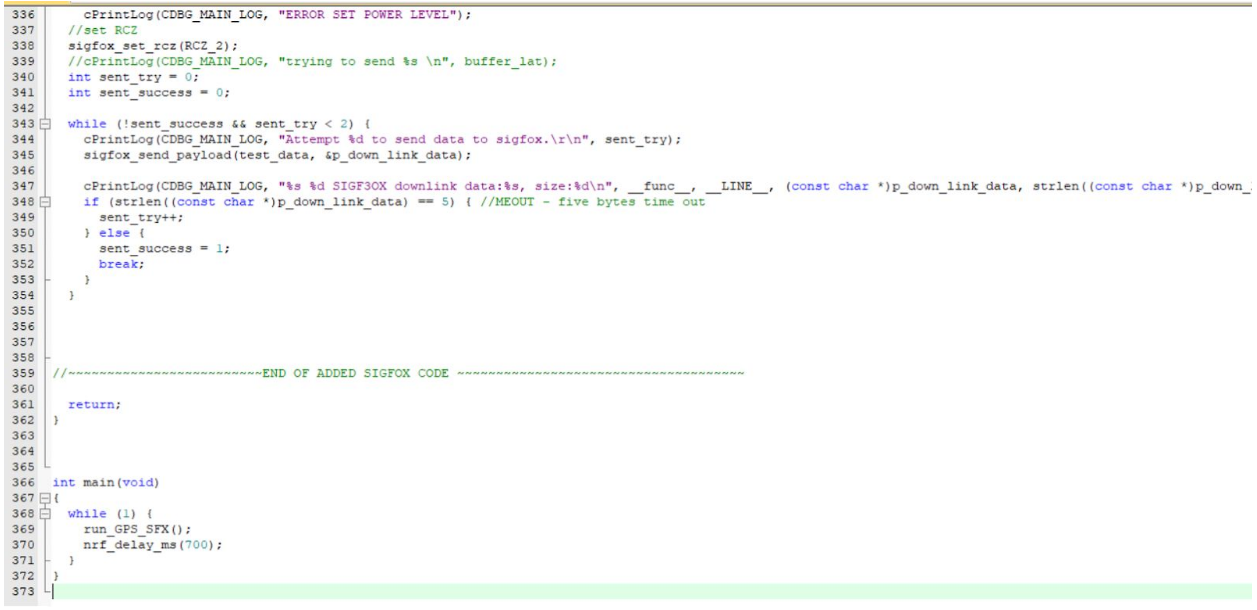
```c
  while (1) {
    nrf_delay_ms(200);
    cPrintLog(CDBG_MAIN_LOG, "LINE[%d]  ==================================================== --[[    \n", __LINE__);
    cDataDumpPrintOut(CDBG_MAIN_LOG, m_cGpsRxNMEA_Buf, CGPS_SPI_BUF_SIZE);
    cPrintLog(CDBG_MAIN_LOG, "[%d]  size[%d]  m_cGpsNMEABuf[%s]  \n", __LINE__, sizeof(m_cGpsRxNMEA_Buf), m_cGpsRxNMEA_Buf);
    cPrintLog(CDBG_MAIN_LOG, "LINE[%d]  ==================================================== --]]    \n", __LINE__);

    if (cGPS_waiting_tracking_end_check())
      break;
  }
#endif

  result = start_gps_nmea_data_check();
  if (result == CGPS_Result_OK) {
// get last location (gps position fixed)
    cPrintLog(CDBG_MAIN_LOG, "---------------------------------------------------------\n");

    result = get_NMEA_Location(&latitude, &longitude);

    if (result == CGPS_Result_OK) {
// success of get gps data
      cPrintLog(CDBG_MAIN_LOG, "@ GPS Position Latitude[%s]\n", latitude);
      cPrintLog(CDBG_MAIN_LOG, "@ GPS Position Longitude[%s]\n", longitude);
    } else {
// no gps data
      cPrintLog(CDBG_MAIN_LOG, "GPS Module NoData!\n");
    }

    get_NMEA_Direction(&ns, &ew);
    cPrintLog(CDBG_MAIN_LOG, "@ N/S[%s] E/W[%s] \n", ns, ew);

    get_NMEA_UTCDate(&year, &month, &day);
    cPrintLog(CDBG_MAIN_LOG, "@ Year[%d]  Month[%d]  Day[%d]\n", year, month, day);

    get_NMEA_UTCTime(&hour, &minute, &second);
    cPrintLog(CDBG_MAIN_LOG, "@ Hour[%d]  Minute[%d]  Second[%d]\n", hour, minute, second);

    get_NMEA_HDOP(&hdop);
```



```
    cPrintLog(CDBG_MAIN_LOG, "@ HDOP[%s] \n", hdop);

    get_NMEA_Speed_knot(&speed); // knote
    cPrintLog(CDBG_MAIN_LOG, "@ Speed[%s]/knote \n", speed);
    cPrintLog(CDBG_MAIN_LOG, "--------------------------------------------------------\n");
  } else if (result == CGPS_Result_NoData) {
// no gps data
    cPrintLog(CDBG_MAIN_LOG, "GPS Module NoData!\n");
  } else if (result == CGPS_Result_NotStarted) {
// gps C/N0 dB-Hz check
    cPrintLog(CDBG_MAIN_LOG, "GPS C/N0 dB-Hz Low!\n");
  } else if (result == CGPS_Result_Fix_Fail) {
// position fix fail
    cPrintLog(CDBG_MAIN_LOG, "GPS Tracking Fail!\n");
    return;
  } else {
// not available GPS
    cPrintLog(CDBG_MAIN_LOG, "Not Available GPS Module!\n");
  }

//~~~~~~~~~~~~~~~~~~~~~~~~START OF ADDED SIGFOX CODE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


  // user data
  uint8_t test_data[SIGFOX_SEND_PAYLOAD_SIZE];
  uint8_t *p_down_link_data;

//timer is initialized in GPS code

  //timer Initialize
  //APP_TIMER_INIT(APP_TIMER_PRESCALER, APP_TIMER_OP_QUEUE_SIZE, false); //only if GPS is commented

  //sd init
  //ble_stack_init_minimal(); //only if GPS is commented

  //snek mode enable
  m_module_parameter.sigfox_snek_testmode_enable = 1;

```

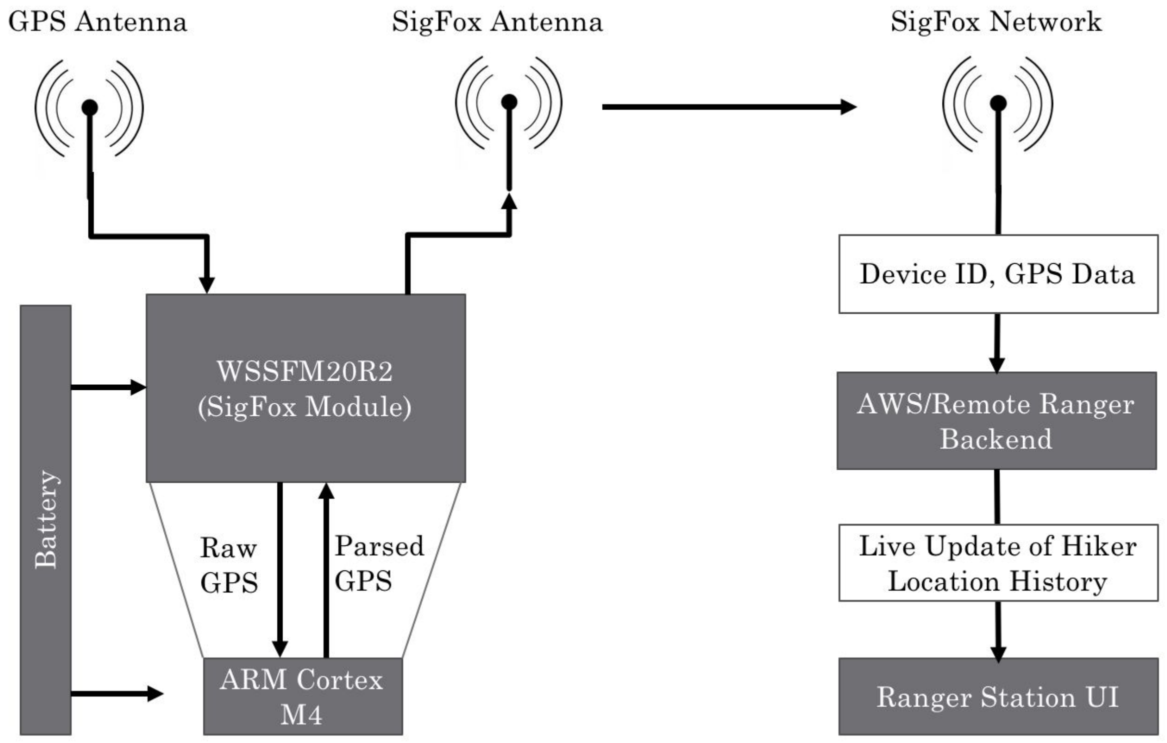


```c
  cPrintLog(CDBG_MAIN_LOG, "Lat: [%s] and long: [%s]\r\n", latitude, longitude);

  uint8_t lat_deg = 0;
  uint8_t lon_deg = 0;
  uint8_t lat_min = 0;
  uint8_t lon_min = 0;
  uint8_t lat_sec = 0;
  uint8_t lon_sec = 0;

  convert_nmea_to_lat_deg_min_sec(latitude, &lat_deg, &lat_min, &lat_sec);
  convert_nmea_to_long_deg_min_sec(longitude, &lon_deg, &lon_min, &lon_sec);


  test_data[0] = *ns;
  test_data[1] = lat_deg;
  test_data[2] = lat_min;
  test_data[3] = lat_sec;
  test_data[4] = *ew;
  test_data[5] = lon_deg;
  test_data[6] = lon_min;
  test_data[7] = lon_sec;
  test_data[8] = (uint8_t)99;     //battery level, need to fix
  test_data[9] = (uint8_t)0x41;   //status, not sure how to check;
  test_data[10] = (uint8_t)99;    //temperature 'a' unsure
  test_data[11] = (uint8_t)12;    //temperature 'b' unsure

  cPrintLog(CDBG_MAIN_LOG, "debug: lat deg[%d], min[%d], sec[%d]\r\n", lat_deg, lat_min, lat_sec);
  cPrintLog(CDBG_MAIN_LOG, "debug: lon deg[%d], min[%d], sec[%d]\r\n", lon_deg, lon_min, lon_sec);
  cPrintLog(CDBG_MAIN_LOG, "debug: full packet %s\r\n", test_data);

  // create sigfox timer instance
  cfg_sigfox_timer_create();


  //Set the power level
  if (!cfg_sigfox_set_powerlevel(14))
    cPrintLog(CDBG_MAIN_LOG, "ERROR SET POWER LEVEL");
  //set RCZ
```



```c
    cPrintLog(CDBG_MAIN_LOG, "ERROR SET POWER LEVEL");
  //set RCZ
  sigfox_set_rcz(RCZ_2);
  //cPrintLog(CDBG_MAIN_LOG, "trying to send %s \n", buffer_lat);
  int sent_try = 0;
  int sent_success = 0;

  while (!sent_success && sent_try < 2) {
    cPrintLog(CDBG_MAIN_LOG, "Attempt %d to send data to sigfox.\r\n", sent_try);
    sigfox_send_payload(test_data, &p_down_link_data);

    cPrintLog(CDBG_MAIN_LOG, "%s %d SIGF3OX downlink data:%s, size:%d\n", __func__, __LINE__, (const char *)p_down_link_data, strlen((const char *)p_down_
    if (strlen((const char *)p_down_link_data) == 5) { //MEOUT - five bytes time out
      sent_try++;
    } else {
      sent_success = 1;
      break;
    }
  }




//~~~~~~~~~~~~~~~~~~~~~~~~~END OF ADDED SIGFOX CODE ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  return;
}



int main(void)
{
  while (1) {
    run_GPS_SFX();
    nrf_delay_ms(700);
  }
}

```

Appendix B: System Architecture


GPS Antenna
SigFox Antenna
SigFox Network
WSSFM20R2 (SigFox Module)
Battery
Raw GPS
Parsed GPS
ARM Cortex M4
Device ID, GPS Data
AWS/Remote Ranger Backend
Live Update of Hiker Location History
Ranger Station UI