Overview

You have been chosen to implement the height sensing subsystem (HSS) onboard a Lunar lander for the first Canadian-led soft landing on the Moon. This subsystem will be engaged during the last 50 meters of the descent towards the Lunar surface and is needed to determine the current spacecraft altitiude and to signal to other subsystems when we have landed. The spacecraft has a laser altimeter sensor that can be used to compute the height of the spacecraft, and as such the HSS should read the raw altimeter measurements and calculate the height.

Due to the complex nature of the Lander flight software, each subsystem is executed in a distributed context where each communicates with the others through the use of a message bus called MoonWire. Subsystems send data by sending specially formatted UDP packets onto the bus and receive data by listening for specially formatted UDP packets on this bus from other subsystems. You can use the provided MoonWire specification to determine how to read and send data from the message bus.

Task Description

Use the provided spacecraft simulator to create the height sensing subsystem. For each LASER_ALTIMETER message recieved you should send a HEIGHT message containing the current spacecraft height in centimeters. Once you detect a landing event, send an ENGINE_CUTOFF message to indicate to all other subsystems that we have touched down. The spacecraft simulator will run on startup and also contains the message bus router.

Use the provided MoonWire specification for more information on how to send and receive data from other subsystems. Some notes about MoonWire:

- The HSS should be bound to port 12778
- The message bus router is bound to port 12777
- For any given message type, all messages are sent in order.
- Each data field in a message is byte-aligned.
- Each message is broadcast to all subsystems regardless of its type.

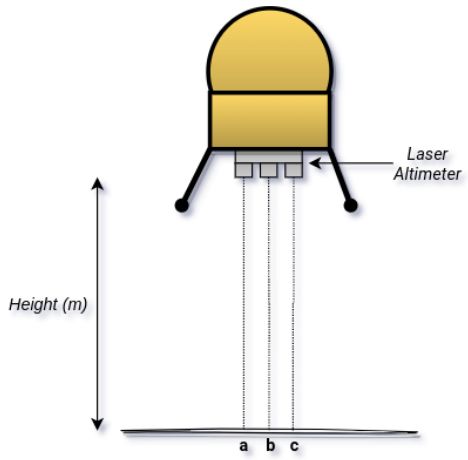

Figure 1: Landing Diagram

The lander system has the following properties:

- The laser altimeter is mounted vertically to the underside of the lander spacecraft.
- When landed, the laser altimeter is approximately 40 cm above the ground.
- The altimeter instrument has 3 sensing heads on it and will therefore produces 3 measurements each time it is sampled.

Keep in mind that laser altimeter readings are generated as unsigned 16-bit integers and must be scaled to determine the true height in meters. The laser altimiter has a max range of 1000 meters and will produces readings from 0 to 65535 (UINT16_MAX).

To implement the height sensing susbsytem, you may use any suitable language/runtime. You should not use any external libraries in your implementation (POSIX APIs are allowed). The flight computer runs Linux on an x86_64 processor and the subsystem must run on this target. You're expected to state all assumptions and justify design decisions. Examples include your choices of language, program structure, and algorithms. Also include any documentation required to setup your software.

You can deliver your source code and supporting documentation as a .zip or .tar archive.

Appendix

Using the simulator

Provided with these instructions is a file named `simulator`. This file is a 64-bit executable intended to run on Linux and simulate the spacecraft for the purpose of testing the HSS. To use the simulator, simply start your HSS program, and then start the simulator with `./build/simulator`. The simulator will send UDP messages to your subsystem which simulate the flight in real-time.

Connecting to the message bus

In C, you can bind a UDP socket using the POSIX socket API:

```c
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>

// Create a UDP datagram socket
int fd = socket(AF_INET, SOCK_DGRAM, 0);
if (fd < 0)
{
    exit(EXIT_FAILURE);
}

sockaddr_in addr;
memset(&addr, 0, sizeof(addr));

addr.sin_family = AF_INET;          // use IPv4
addr.sin_addr.s_addr = INADDR_ANY; // bind to all interfaces
addr.sin_port = htons(port);        // the port we want to bind

// Bind to the port specified above
if (bind(sockfd, (const struct sockaddr *)&addr, sizeof(addr)) < 0)
{
    exit(EXIT_FAILURE);
}

// Listen for data on our port (this is blocking)
char buffer[4096];
int n = recvfrom(socket, buffer, 4096, MSG_WAITALL, NULL, NULL);
```