EE536 IoT Systems Lab Exam

April 2023

Instructions: You will build the system as a team, but evaluation will be individual. Each team member is expected to know all aspects of the system.

After completion, create a directory named `ee536_2023_team_number`. Add all your code (Arduino, Python) and anything else into this directory. Compress the directory and create a single zip file named as `ee536_2023_team_number.zip` and upload into Moodle. The code will be reviewed after the exam. Note that `team_number = kit_number`.

You must show the output at 5 pm. You can continue to work after that until 6 pm.

Any suspicion of cheating will result in F grade.

Evaluation

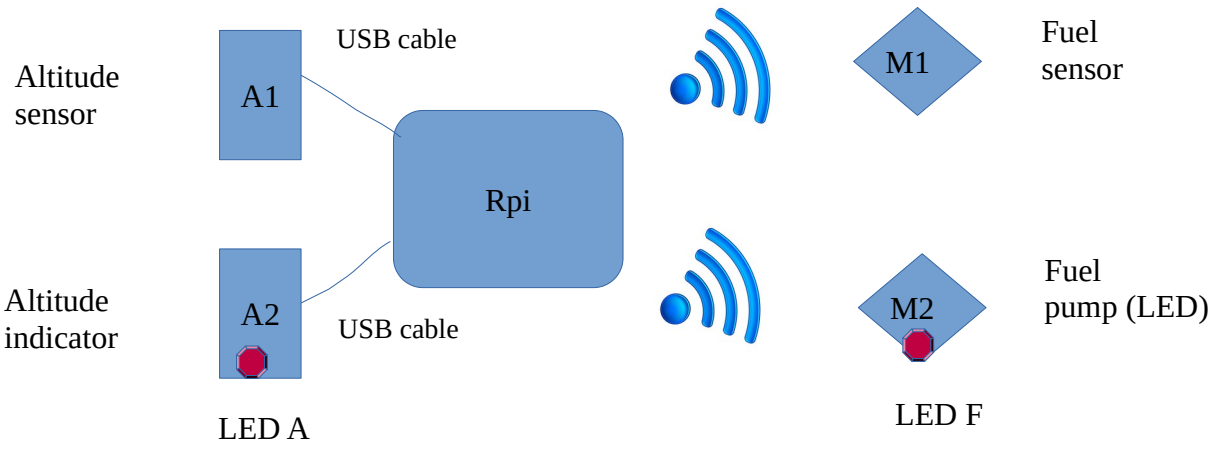
- Demo and understanding: 60 marks
- Code:
  - clear, modular code: 20 marks
  - comments in code: 20 marks
- Individual marks will be given.

Consider a system of sensors and controller in an airplane. We have two sensor-indicator pairs:

Altitude sensor A1 (Arduino 1)	Fuel sensor M1 (nodeMCU 1)
Altitude indicator A2 (Arduino 2 LED)	Fuel pump indicator M2 (nodeMCU 2 LED)

Note that the indicator for both sensors are LEDs.

They are centrally connected via a computer (which is the Rpi.) This are illustrated in the figure below.



1. Part 1
   a. (10 marks) The altitude sensor A1 generates the altitude as per Fig 1. This is communicated to the Rpi via a serial connection. The altitude values are recorded in a CSV file `alti.csv.`
   b. (10 marks) The Rpi sends the altitude values to the indicator LED in A2 via a serial connection. When altitude is < 100, the LED A blinks rapidly. When altitude is >= 100, LED A stays on without blinking.

2. Part 2
   a. (10 marks) The fuel sensor M1 generates the fuel values as per Fig 1. This is communicated to Rpi via a wireless link. The fuel values are recorded in a CSV file `fuel.csv.`
   b. (10 marks) The Rpi sends the fuel values to the LED F in M2. When the fuel is above 20, LED F glows without blinking. When the fuel is below 20, LED F blinks rapidly.

3. Part 3
   (20 marks) If the fuel level is below 20 during level flight, the Rpi prints a warning message. For example, fuel consumption represented by **'Fuel2'** in the plot gives a warning, whereas the consumption represented by **'Fuel1'** is safe.

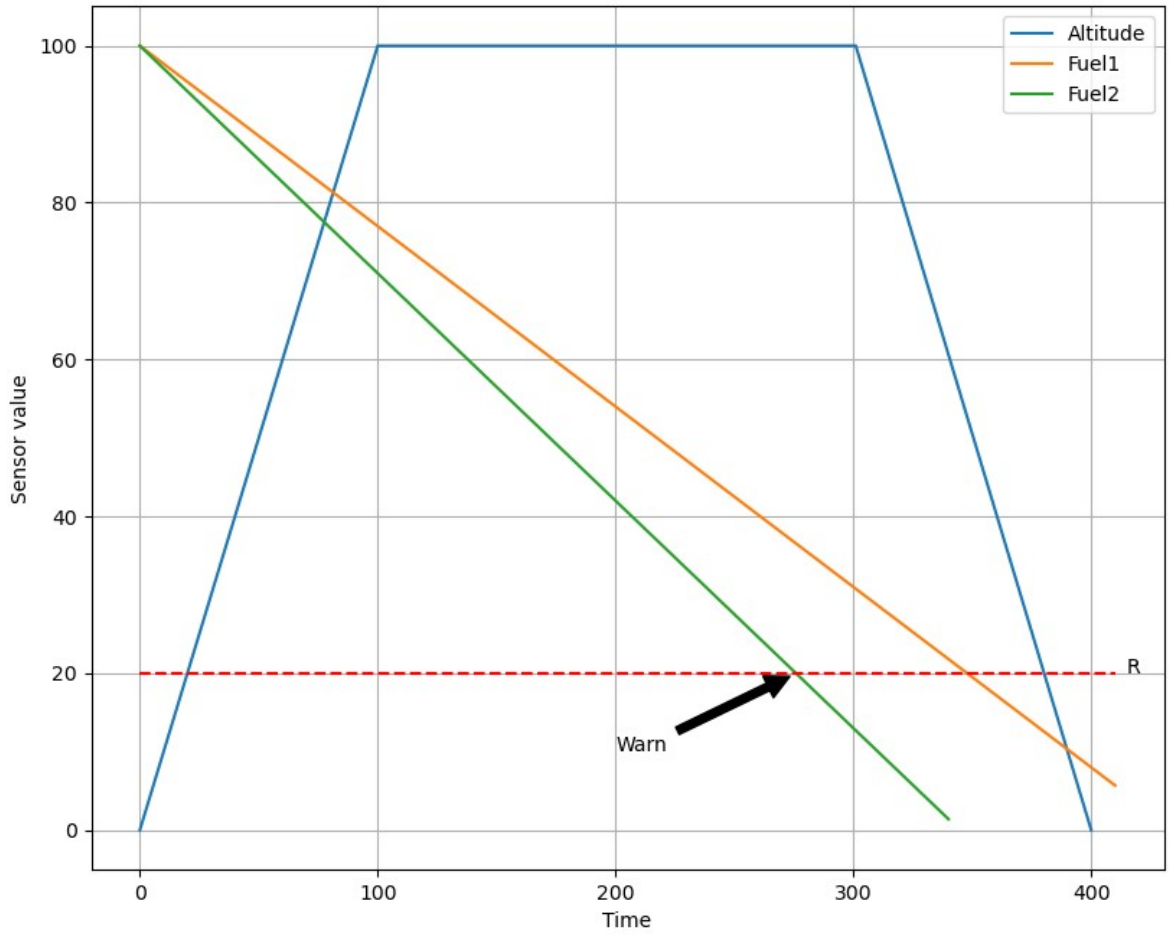


Figure 1: Plot showing altitude and two types of fuel consumption. R represents the fuel reserve level.

Notes:

1. **Fuel1** can be represented by the line `f = -0.23t+100`, where as **Fuel2** is represented by `f = -0.29t+100`, where `t` is the time.
2. Level flight means the altitude is 100. At other times, the altitude is < 100, and `a = t`, where `a` is altitude and `t` is the time (use this to generate the altitude curve.)
3. If the fuel level goes below 20 during level flight, a warning has to be generated at the Rpi. (black arrow in the figure.)
4. You must be able to demonstrate simulation of **Fuel1** flight or **Fuel2** flight (by reloading sensor code as needed.) The demo must finish in less than 2 minutes, so plan accordingly. If needed, other fuel and altitude profiles may also need to be simulated.
5. Once started, the simulation should go without any manual intervention for a complete flight cycle.
6. While doing the development, it may be helpful to increment the time count by 5, so that the simulation finishes rapidly. Also, you may experience some issues with buffering, so the this will help prevent that.
7. Code for a UDP receiver is provided (`udp_receiver.ino.`) You can modify and use that if needed.

8. **Optional (bonus marks)**: Add an mechanism so that the Rpi can start both altitude and fuel subsystems (for example, by accepting the user input ‘`S`’. This will make the synchronization easy.