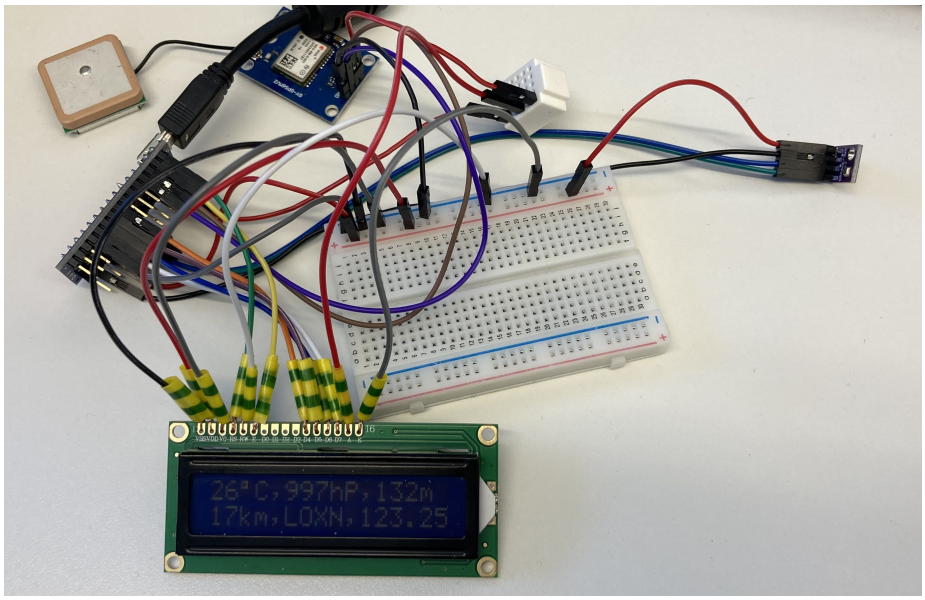
Glider Assistant

Rositsa Ivanova

24.06.21



1 Idea

The glider assistant is a rather small device, which aims to give support to a pilot when flying a glider. The information it provides is the *current temperature*, the *current air pressure*, the *current altitude*, the *distance to the closest airfield*, the *ICAO abbreviation of the closest airfield*, and the *frequency of the closest airfield*. This information is displayed on a simple LCD display.

2 Required Components

For the building of the device, we use the following components:

- Arduino Nano
- Mini USB - USB cable
- LCD display 16x2
- BME 280 sensor
- NEO-6M sensor
- Regular-sized breadboard
- Mini breadboard
- Male-Male jumper cables
- Female-Female jumper cables
- Male-Female jumper cables

We decided to use an Arduino Nano as it is small, yet sufficient for the purpose at hand. The power supply is delivered via a cable from a computer. The current temperature, air pressure, and altitude are collected using the BME 280 sensor. The NEO-6M module delivers the current GPS coordinates of the device.

3 Connection of Components

For this project we connect the components using jumper cables, a small breadboard and a regular-sized breadboard. Figure 1 shows the connections. Due to the fact that the BME 280 sensor is connected to the Arduino Nano using the I2C connectors, we do not use the hat of the LCD display, which also uses I2C, but instead directly wire it to the Arduino Nano[1]. We clearly separate the 3.3 V and the 5 V power supply, but putting them on two individual breadboards. For the NEO-6M module we use the bigger version of the standard antena.

4 Software

We compile and upload the source code to the Arduino Nano using the official Arduino Software[2]. For the BME 280 sensor we download the BlueDot BME280 library. The most fitting library for the NEO-6M module in our case was TinyGPS[3]. For the LCD display we adapted sample code provided in a tutorial[4]. Lastly, for the calculation of the distance between 2 sets of GPS coordinates, we made use of source code provided by GeoDataSource[5].

[1] https://create.arduino.cc/projecthub/najad/interfacing-lcd1602-with-arduino-764ec4
[2] https://www.arduino.cc/en/software
[3] https://github.com/mikalhart/TinyGPS
[4] www.diyusthad.com
[5] https://www.geodatasource.com/developers/c

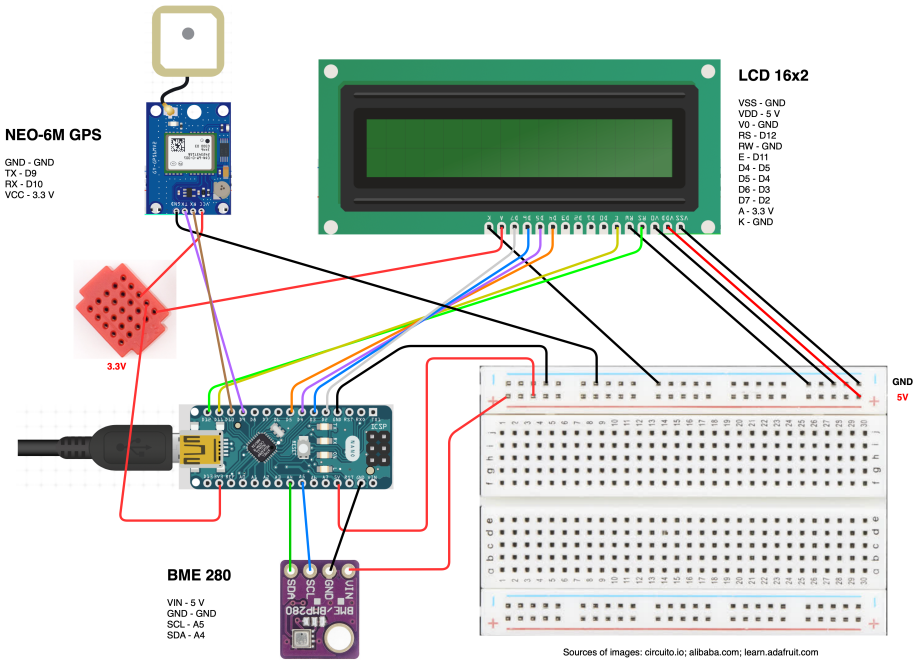


Figure 1: Connection of Components

The source code of the device is written in C and can be found at: https://github.com/therosko/Glider_Assistant.

5 Limitations

Despite all components working together successfully, the setup does have certain limitations. The LCD display size is compact and can be used for little information without much distraction. However, if one decided to use a display with one more line, additional information could be fit without overloading the user with too much content. Further, the GPS module requires relatively more power, despite it needing 3.3 V and not 5 V. In the current setup, the power is just enough to display content on the LCD display, yet the display is not reaching its highest brightness, when the GPS module is plugged into the same power source. One potential solution is the use of an external power source for the GPS module. One more shortcoming of the NEO-6M module is the antenna. In our implementation, we used the bigger version of the antenna, however was challenging to receive good enough signal[6]. There are alternative GPS modules or higher-end antennas that could be considered with a bigger budget. Lastly, for the connection of the LCD display we did not have a soldering tool, therefore

[6]It was impossible to receive signal indoors. Only try outside for results.

we used an alternative method. First, we cut off one end of a jumper cable and removed a section of the isolation cable around the wires. Then, we twisted the wires and wrapped them through and around the respective holes of the LCD display (i.e. where usually we would have pins). We made sure that only one cable runs through each hole and isolated the open wires of the cables in order to avoid a short-circuit. In the end, we connected the other ends of the cables to the correct pins of the Arduino and the breadboard(s).