Deliverable H - Prototype III and Customer Feedback
GNG 1103D

Group #9

March 28, 2021

Group members: Karen Hakko, Elsa Lange, Tri Thai, Jacob

Troop and Sandeep Sinha

Contents

List of Figures

List of Tables

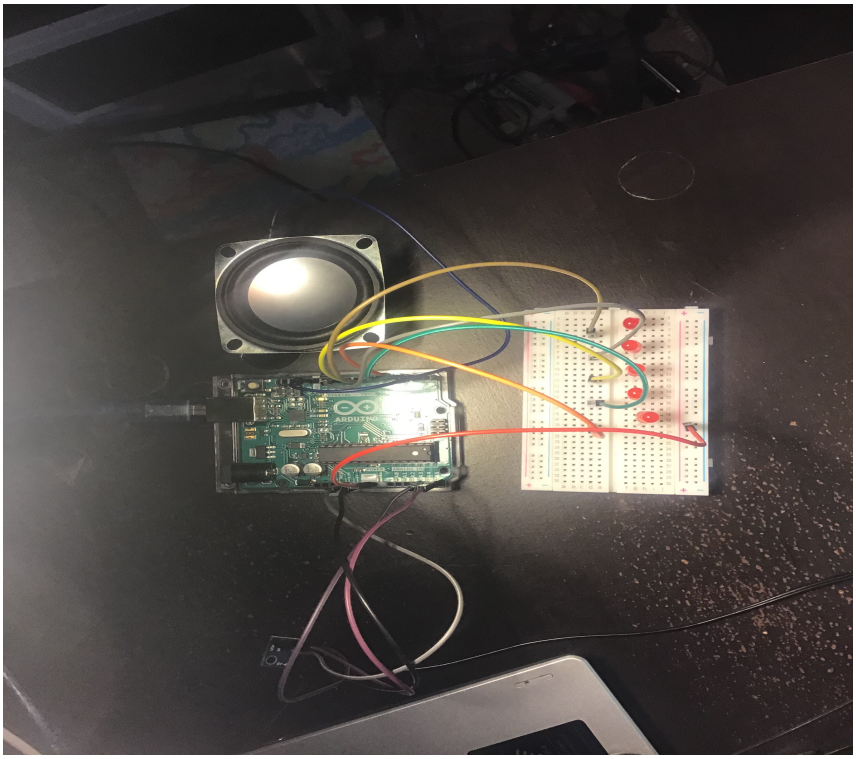
Problem Statement

The JAMZ developers need an emergency beacon that transmits accurate and quick location information about the drone to the operator in live time by interpreting the data received from the sensors as well as alerting nearby citizens of the downed drone with flashing lights and a voice system.

Summary of Previous Deliverable

In our previous deliverable, we provided the testing results for prototype 2 of our emergency beacon for the JAMZ drone. It started out with a recap of the testing done in prototype 1 and the improvements that were made to prototype 1 for the prototype 2 testing. Included in the prototype 2 testing is an analytical model of the various subsystems, including the location, altitude, light and voice subsystems, and an analytical model of the entire system with all the subsystems combined. The previous deliverable also contained the test results for the various subsystems and the circuit designs for each of them. The deliverable finished off with a 3D model of what the case for the emergency beacon would roughly look like, and the Wrike update was added in at the end.

Figure 1- The Complete Emergency Beacon System



The figure above shows the complete emergency beacon system, with the light, voice, and altitude subsystems, all connected to the Arduino, either via a breadboard or directly with jumper cables. The system is being powered with a USB cable, connected to a computer.

Testing, Results and Improvements on Prototype II

On the previous deliverable, the initial model for each of the subsystems was built and we were able to test the code to ensure the wiring on these physical models was correct. This was done for each subsystem but the location subsystem and for a basic overall beacon to ensure the functionality of the system. For this deliverable, those results were improved upon by modifying the code, improving the subsystem layout, completing the location subsystem and making sure the subsystems are capable of sending data to the Arduino. In this round of testing, the light subsystem code was left the same and the only change made was that the lights are now activated when the altitude subsystem determines the drone is too low. Also, the number of LED lights was increased from 4 to 8 in order to increase their visibility and activate lights in a specific order to warn anyone near the drone. The voice subsystem was only tested as a component of the overall subsystem however it was modified to increase the volume of the sound produced by increasing the voltage passed through to 5V and changing the resistors. After multiple tests, an appropriate volume was reached. Also, the automated message that was added to the code in prototype 2 was modified and completed for this prototype. The final changes made to the code allow for the system to be activated and the automated message to be played after the altitude sensor determines a threshold altitude has been crossed. For this prototype, the altitude subsystem was tested only as a part of the overall beacon and it was able to successfully interpret atmospheric pressure when the beacon altitude is too low and then activate the voice and light subsystems. The location subsystem was completed for this prototype after we received the part and we were able to test the subsystem's ability to interpret GPS data to provide an accurate location. It was tested multiple times and functioned very well, however we are still having difficulties connecting this subsystem to the overall beacon in order to send this information to the JAMZ operator. We will need to modify the code and possibly improve upon the systems wiring in order to have it functioning as a part of our overall beacon for design day. The final improvement for this prototype was the addition of a protective case laser cut from acrylic; it was tested by placing all the components of the case to ensure that it is the proper size. We plan to improve upon this case by waterproofing the case and making sure that the speakers can still be heard and the LED's are still highly visible.

Table 1 - Tasks Plan for Prototype 2.5 and 3

Task		Member Responsible	Due date
Fixing of components	1. Location subsystem 2. Altitude subsystem 3. Voice and light subsystem a. Voice b. Lights 4. All-around connections	Sandeep Elsa Karen Jacob Tri	March 15th, 2021
Testing of the physical prototype	1. Test each subsystem 2. Location subsystem 3. Altitude subsystem 4. Voice and light subsystem	Elsa and Karen	March 17th, 2021

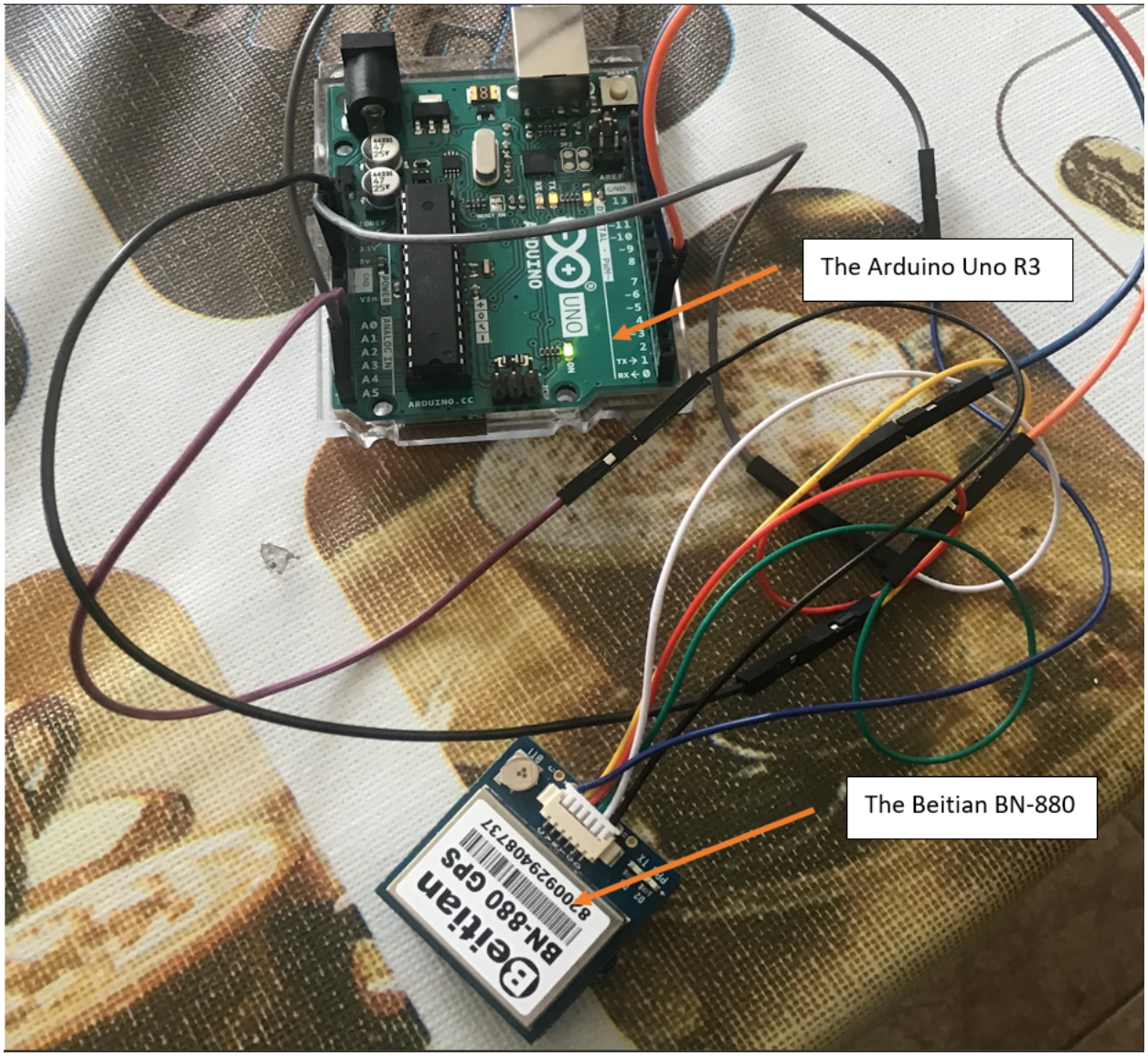
	a. Voice b. Lights 5. All-around connections		
Fixing of components	5. Location subsystem 6. Altitude subsystem 7. Voice and light subsystem a. Voice b. Lights 8. All-around connections	Sandeep Elsa Karen Jacob Tri	March 22nd, 2021
Testing of the physical prototype	1. Test each subsystem 2. Location subsystem 3. Altitude subsystem 4. Voice and light subsystem a. Voice b. Lights 5. All-around connections	Elsa and Karen	March 24th, 2021
Deliverable H	1. Analytic model 2. Formatting 3. Summary of previous deliverable 4. Testing, results and improvement methods of the first prototype 5. Testing results 6. Inkscape design 7. Encasing 8. Presentation 9. Wrike update	Jacob, Elsa and Karen Tri Sandeep Jacob Elsa and Karen Jacob Tri Tri Karen	March 28th, 2021

Prototype 2.5

The main objective of prototype 2.5 was to test different codes while still satisfying design criteria. The subsystems tested were the voice and location subsystem.

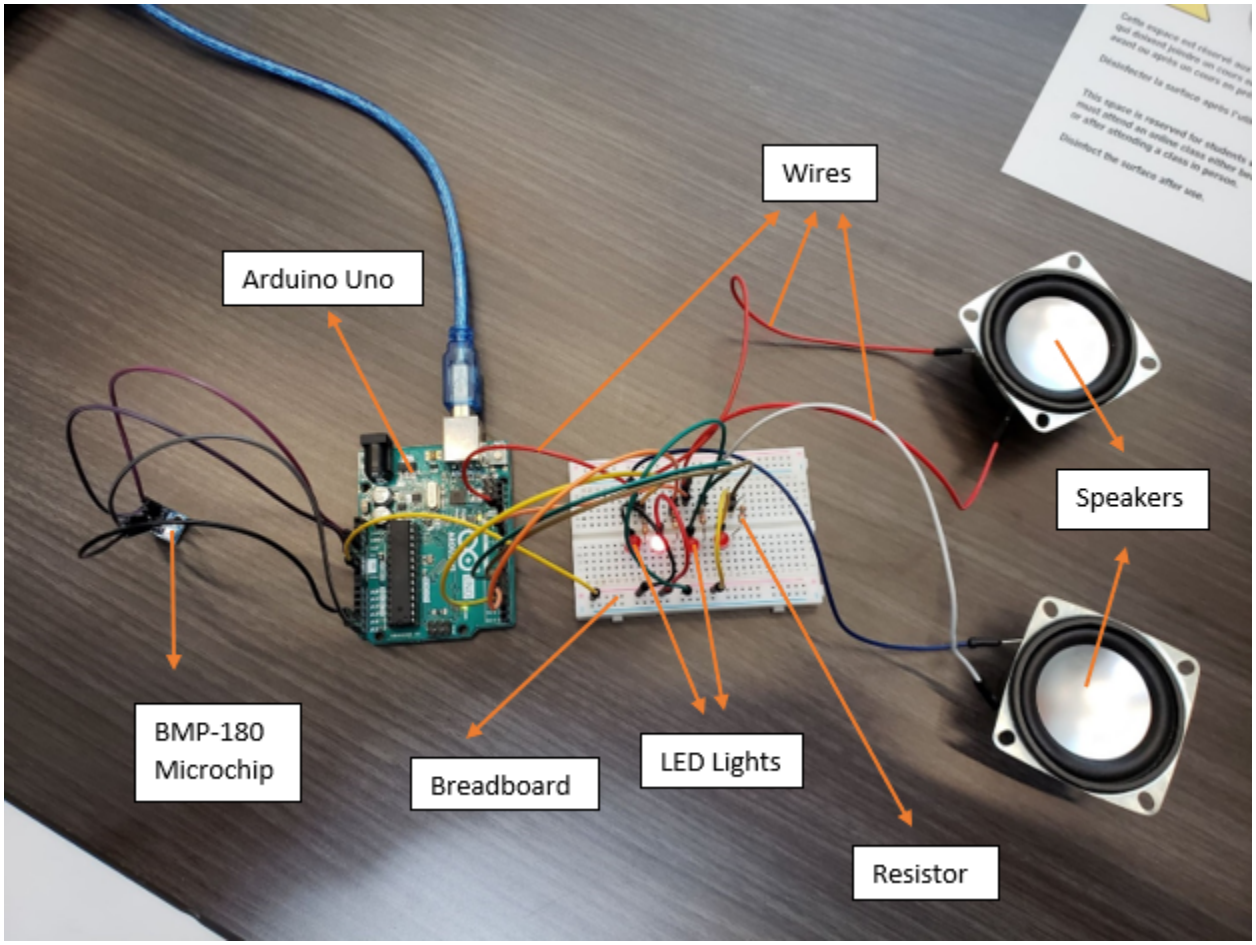
Analytical Model

Figure 2 - Detailed Diagram of the Location Subsystem



The figure above is a detailed diagram of the location subsystem which includes the Beitian BN-880 as well as the Arduino Uno. The Beitian BN-880 is connected through the I2C bus of the Arduino using the SDA and SCL pins as well as pins 3 and 4. The product is also connected to the Arduino in the ground and 5V pins.

Figure 3 - Detailed Diagram of the Voice, Altitude and Light Subsystems



The figure above is a detailed diagram of the integrated system including the voice, light and altitude subsystems. For the voice subsystem, we have the two speakers that are connected to the breadboard and then the Arduino Uno and they display an automated message. For the light subsystem, the figure shows four red LED lights that are connected to the breadboard and their function is to light up, one after the other, in case of an emergency. Lastly, the altitude subsystem, it is made of the BMP-180 Microchip which uses wires to directly connect to the Arduino Uno. Its function is to measure the altitude and send the data to the operator and the other subsystems. Once the altitude drops below a specified threshold, the voice and light subsystems are activated.

Altitude Subsystem

For this prototype, the altitude subsystem was not tested separately, but only in the fully integrated prototype.

Location Subsystem

This section contains testing done for the location subsystem. The first two tests tested if the Beitian BN-880 was functioning properly.

Test 1

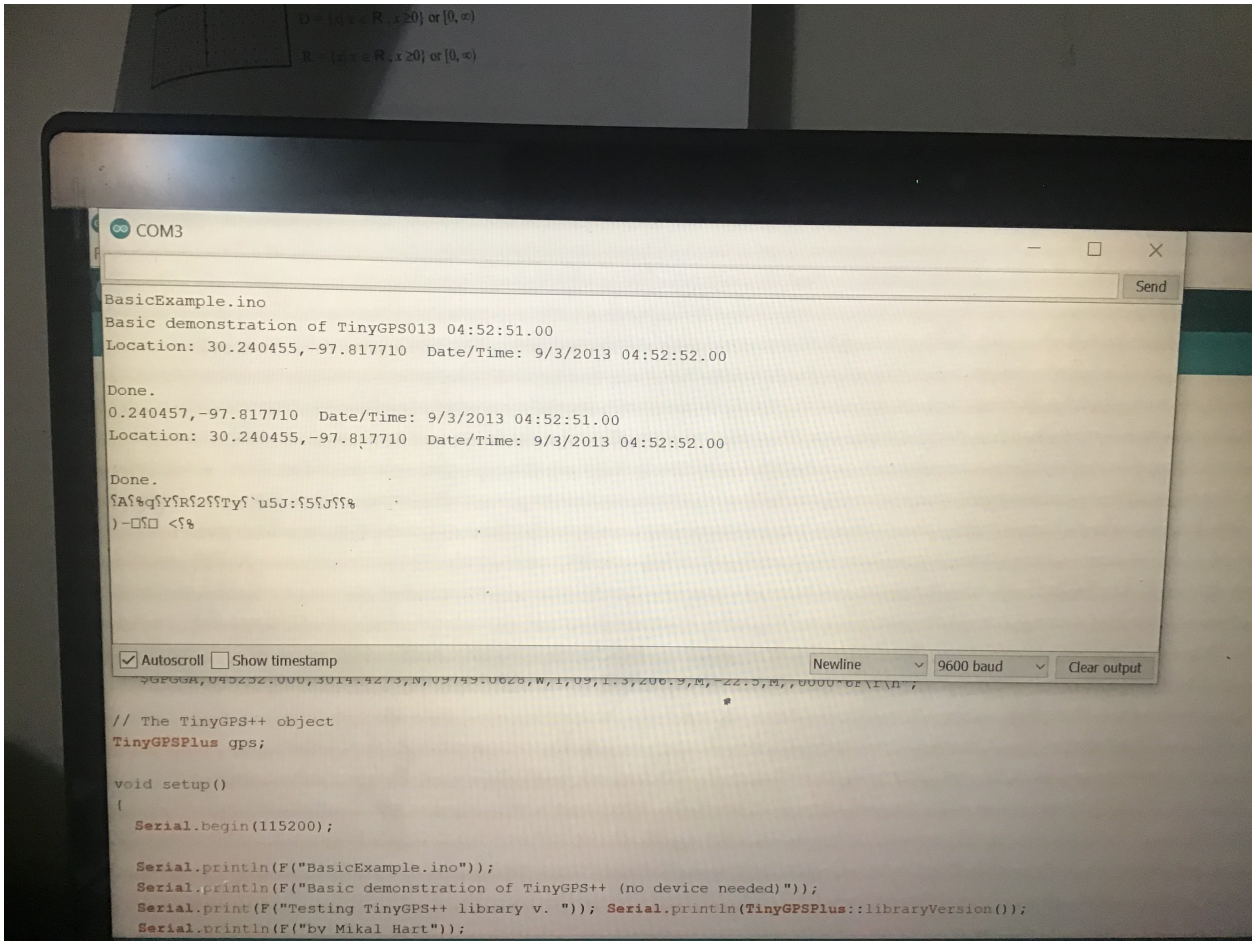
Table 2 - Test Plan 1 for the Location Subsystem for Prototype 2.5

Test ID	Member(s) Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Elsa	To test the functioning of the Beitian BN-880.	The prototype contained the Arduino, the Beitian BN-880 and connecting wires. A code was used to test if the Beitian BN-880 printed the coordinate data to the serial monitor.	The results were a coordinate printed to the serial monitor. In the future, this code can be used as a baseline to check if the product is working.	March 15th, 2021 Test Duration: approximately 2 minutes.	The prototype was deemed satisfactory when it printed coordinates to the serial monitor.

The first test of this subsystem tested if the Beititan BN-880 was functioning properly. The test was finished once it printed coordinates to the serial monitor. In the future, the test results will be used as a signal to move onto checking if the product satisfies design criteria and as a baseline for the proper execution of the location subsystem.

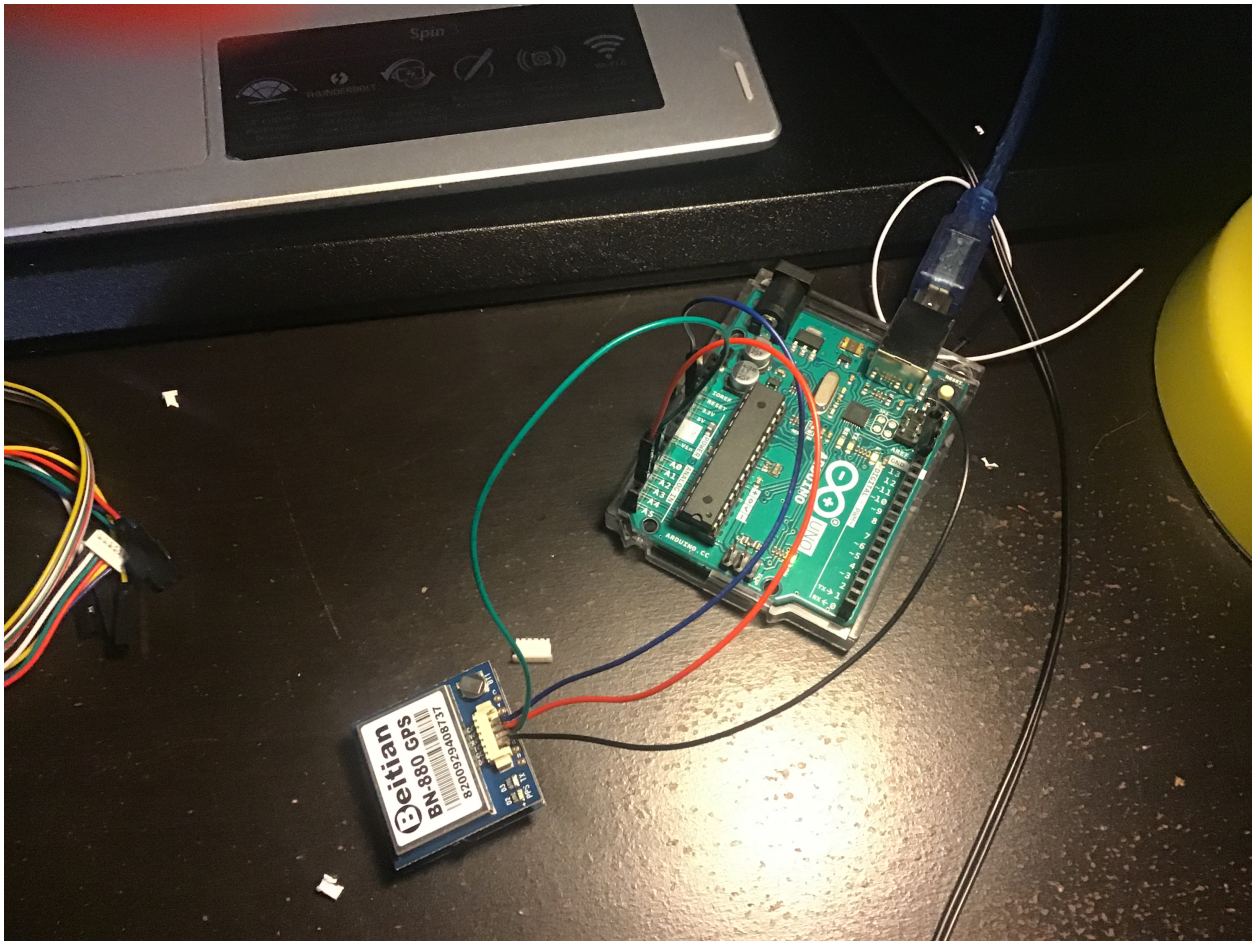
Results

Figure 4 - Code Results for Test 1 of the Location Subsystem



The figure above shows the code results of this test. It was retrieved from here.

Figure 5 - The Location Subsystem



The figure above shows the setup used for the location subsystem.

Test 2

Table 3 - Test Plan 2 for the Location Subsystem for Prototype 2.5

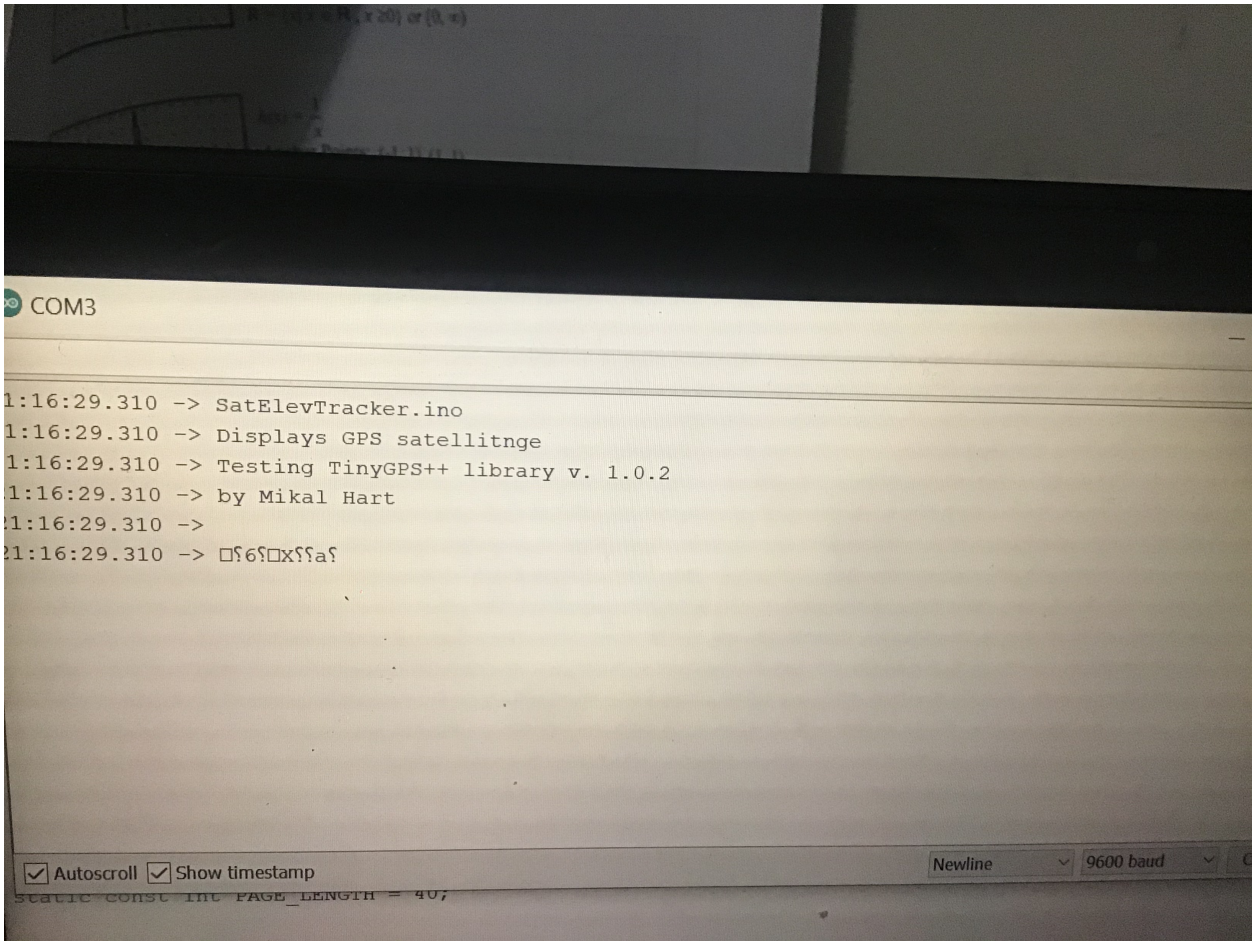
Test ID	Member(s) Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
2	Elsa and Karen	To test if the Beitian BN-880 satisfies the design criteria.	The prototype contained the Arduino, the Beitian BN-880 and connecting wires. A code was used to test if the Beitian BN-880 printed	The test failed as the live coordinates of the product were not printed on the serial monitor.	March 15th, 2021 Test Duration: approximately 2 minutes.	The prototype was deemed satisfactory when it printed the live coordinates

			the coordinate data to the serial monitor.			of the prototype to the serial monitor.

The objective of this test was to test if the Beitian BN-880 could satisfy design criteria. The test failed as it printed checkmarks to the serial monitor instead of coordinates.

Results

Figure 6 - Code Results for Test 2 of the Location Subsystem



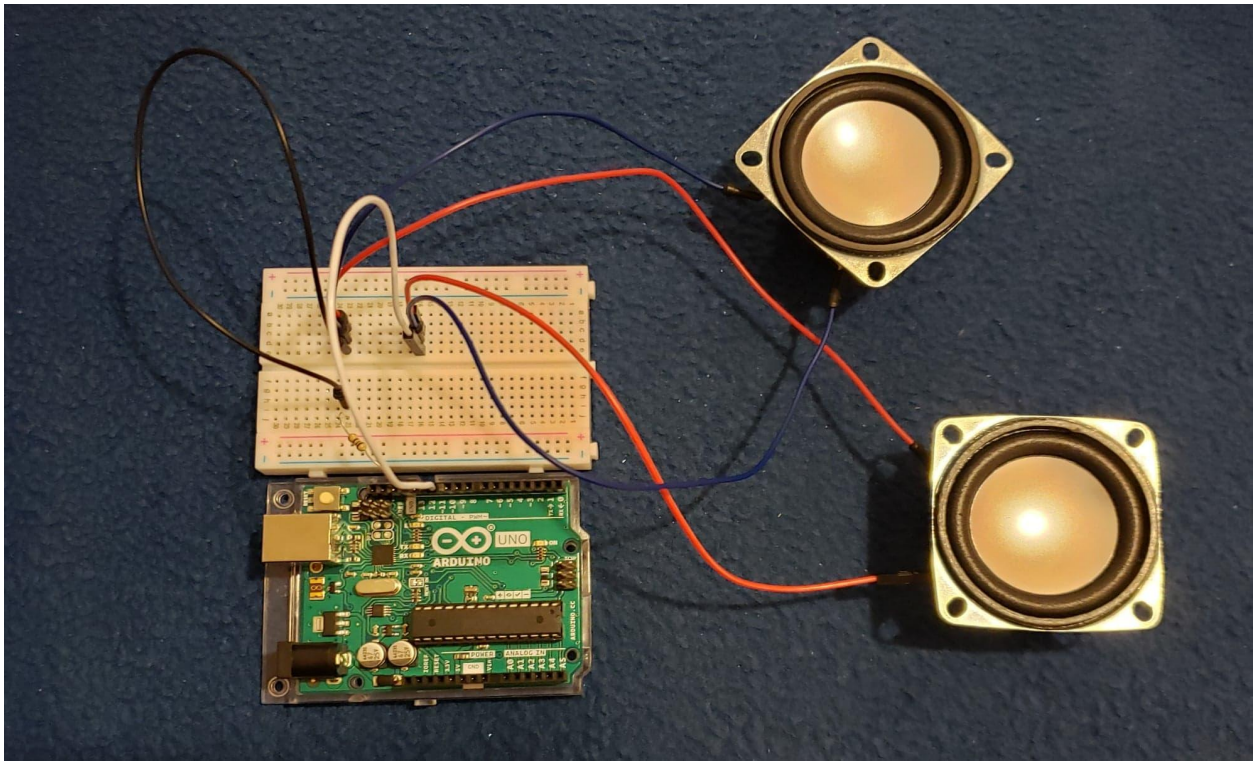
The figure above shows the results of the second test done on the location subsystem. It was retrieved from this file.

This test had the same setup as test 1 for this subsystem.

Voice Subsystem

This section contains testing done for the voice subsystem in prototype III. For this prototype, the final code modification was done to display the automated message needed for the emergency beacon. The code for the voice subsystem was taken and later modified from this website. No other modifications were needed for this prototype, all the wiring was working properly.

Figure 7 - The Voice Subsystem



Test 1

Table 4 - Test Plan 1 for the Voice Subsystem for Prototype 2.5

Test ID	Member(s) Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Karen	To test the functioning of the code to relay an automated message	The prototype contained the Arduino, two speakers, one resistor and connecting wires. A code was executed through the Arduino.	The speakers turned on using a code and were able to play the automated message.	March 15th, 2021 Test Duration: approximately 2 minutes.	The prototype was deemed satisfactory when the speakers played the automated message.

This test focused on having a functional automated message using the “talkie” library. The words chosen for the final automated message play: ‘Danger, operator is on alert ’.

Light Subsystem

This subsystem was not tested separately for this prototype. It was only tested in the fully integrated prototype.

Interconnections

This section contains testing done on the complete prototype. While all the components were connected to the Arduino, only the speakers were tested in the first prototype.e.

Table 5 - Test Plan 1 for the Voice Subsystem for Prototype 2.5

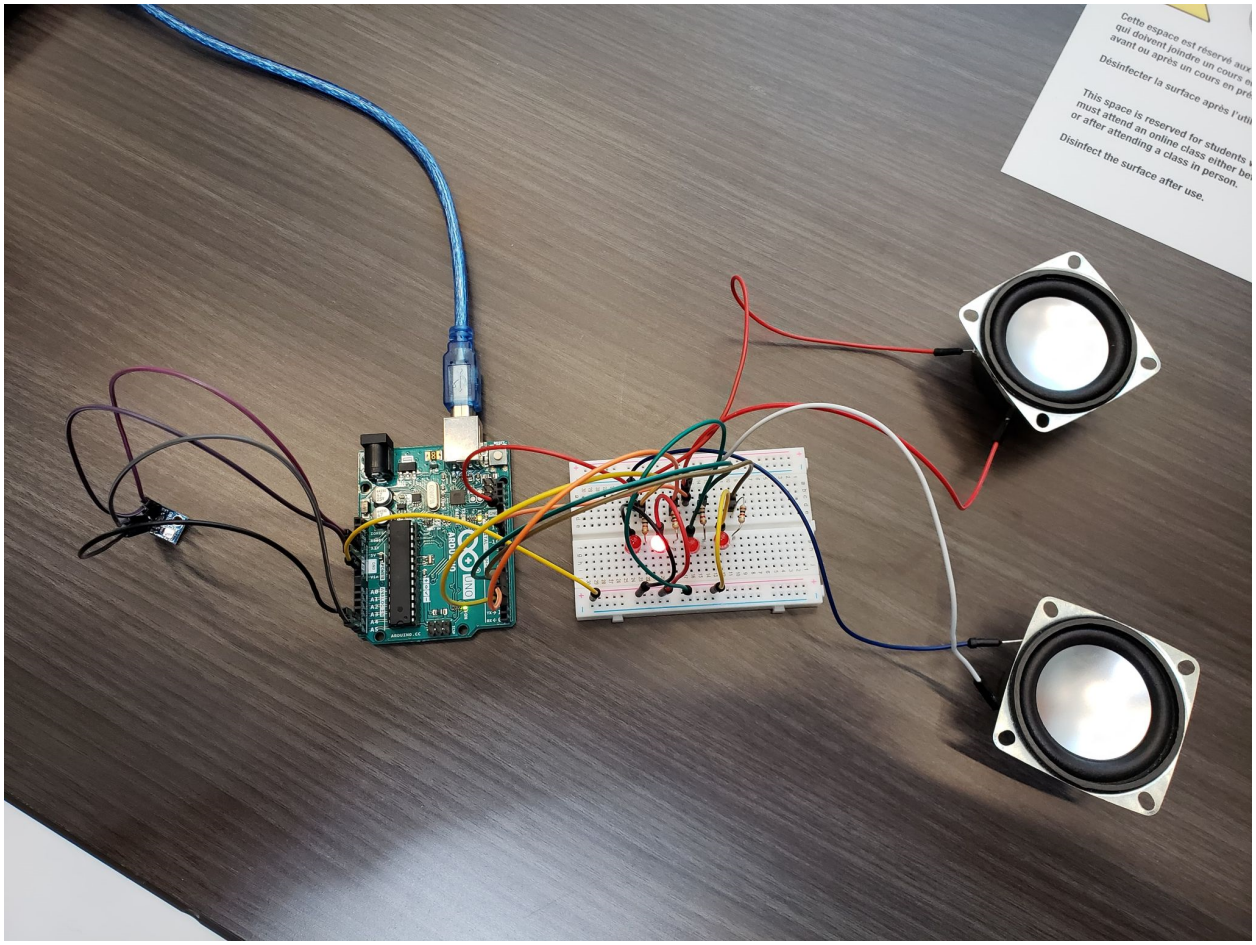
Test ID	Member(s) Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Elsa and Karen	To test the different types of	The prototype contained the Arduino and all	The automated message could be heard from	March 17th, 2021	The prototype was

		automated messages for the voice subsystem.	other subsystems except the location subsystem. A code was used to test the different outputs of the speaker.	the speakers. In the future, it will be decided which type of automated message best satisfies design criteria.	Test Duration: approximately 2 minutes.	deemed satisfactory when the speakers executed the automated message.

This test focused on alternate ways to execute an automated message. In this code, the words executed are included in an Arudino library, contrary to the previous code which was a pre-recorded message. In the future, the code that best satisfies the design criteria will be chosen in the final product.

Results

Figure 8 - The Complete Prototype

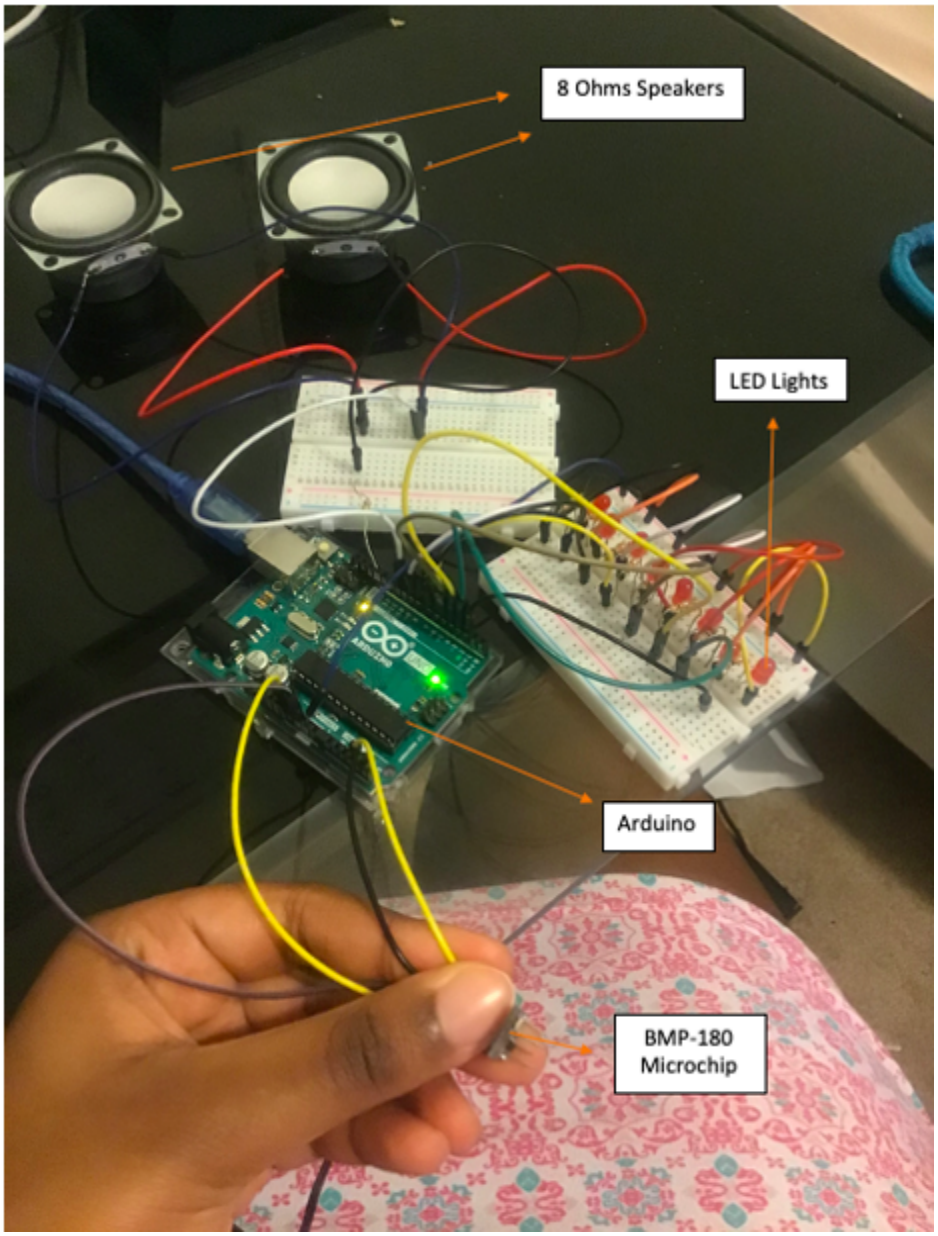


Prototype III

The main objective of the third prototype was to complete the prototype using various steps of integration.

Analytical Model

Figure 9 - Detailed Diagram of Complete Prototype



In the figure above, the speakers are connected to the breadboard and are capable of relaying an automated message that is set in the code. The 8 LED lights are connected to the breadboard and flash one

after another to warn pedestrians to keep away and also the lights increase drone visibility to help the JAMZ team find the drone if it has been downed. The BMP180 microchip is able to measure the atmospheric pressure and when connected to the Arduino, these values can be used to determine the drone's altitude. The Arduino controls all the subsystems, all data and commands are interpreted within the Arduino.

Location Subsystem

This section contains isolated testing done for the location subsystem. The tests were to determine if the Beitian BN-880 could satisfy design criteria using two different codes.

Test 1

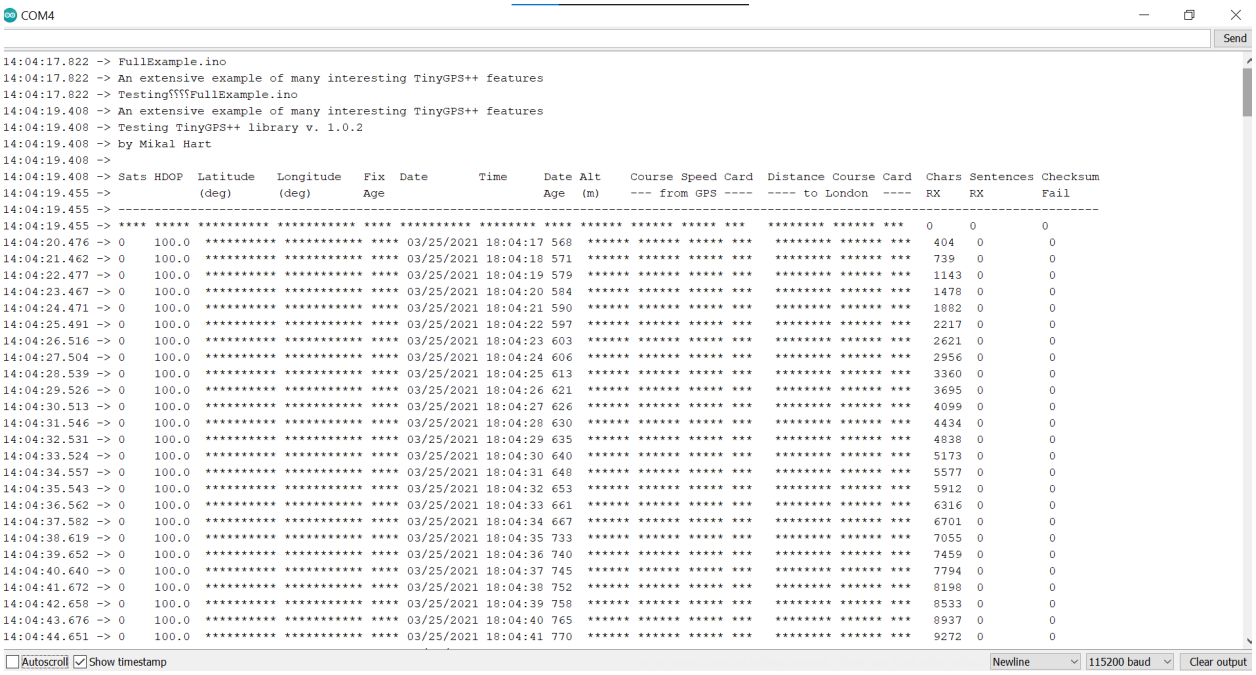
Table 6 - Test Plan 1 for the Location Subsystem for Prototype III

Test ID	Member(s) Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Elsa	To test if the Beitian BN-880 can satisfy design criteria.	The prototype contained the Arduino, the Beitian BN-880 and connecting wires. A code was used to test if the Beitian BN-880 printed the coordinate data to the serial monitor.	The results were coordinate data that was printed to the serial monitor. The data was live as it changed according to the movement. In the future, either this code or the one used in the next cde will be used in the final prototype.	March 25th, 2021 Test Duration: approximately 4 minutes.	The prototype was deemed satisfactory when it printed live coordinates to the serial monitor.

This test focused on printing the live coordinates to the serial monitor. The code also included extra features that may be used in the final prototype. In the future, this code may be included in the final prototype. The code was found in the TinyGPS++ library which can be downloaded from here.

Results

Figure 10 - Code Results of the Location Subsystem

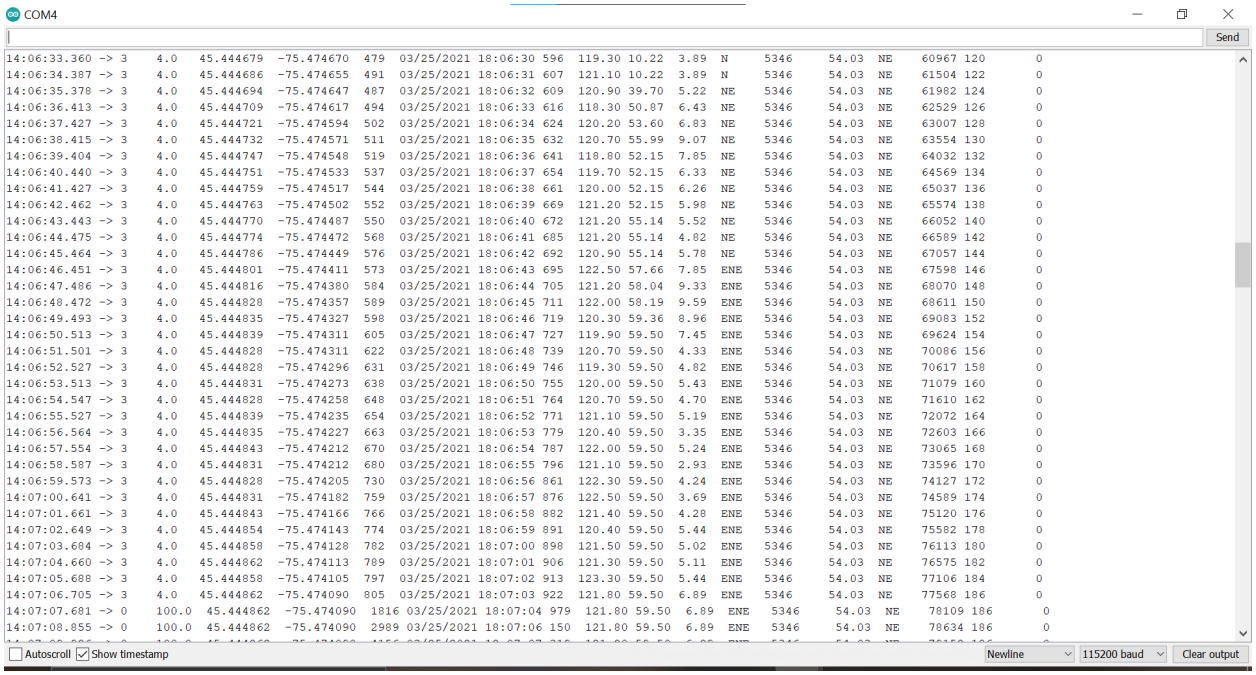


COM4

Send

```
14:04:17.822 -> FullExample.ino
14:04:17.822 -> An extensive example of many interesting TinyGPS++ features
14:04:17.822 -> Testing¶¶¶¶FullExample.ino
14:04:19.408 -> An extensive example of many interesting TinyGPS++ features
14:04:19.408 -> Testing TinyGPS++ library v. 1.0.2
14:04:19.408 -> by Mikal Hart
14:04:19.408 ->
14:04:19.408 -> Sats HDOP  Latitude   Longitude   Fix  Date       Time     Date Alt    Course Speed Card  Distance Course Card  Chars Sentences Checksum
14:04:19.455 ->            (deg)      (deg)      Age                      Age  (m)    --- from GPS ----  ---- to London  ----  RX    RX        Fail
14:04:19.455 -> ----------------------------------------------------------------------------------------------------------------------------------------
14:04:19.455 -> **** ***** ********** ********** **** ********** ******** **** ****** ****** ***** ***   ******** ****** ***   0     0         0
14:04:20.476 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:17 568  ****** ****** ***** ***   ******** ****** ***   404   0         0
14:04:21.462 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:18 571  ****** ****** ***** ***   ******** ****** ***   739   0         0
14:04:22.477 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:19 579  ****** ****** ***** ***   ******** ****** ***   1143  0         0
14:04:23.467 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:20 584  ****** ****** ***** ***   ******** ****** ***   1478  0         0
14:04:24.471 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:21 590  ****** ****** ***** ***   ******** ****** ***   1882  0         0
14:04:25.491 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:22 597  ****** ****** ***** ***   ******** ****** ***   2217  0         0
14:04:26.516 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:23 603  ****** ****** ***** ***   ******** ****** ***   2621  0         0
14:04:27.504 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:24 606  ****** ****** ***** ***   ******** ****** ***   2956  0         0
14:04:28.539 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:25 613  ****** ****** ***** ***   ******** ****** ***   3360  0         0
14:04:29.526 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:26 621  ****** ****** ***** ***   ******** ****** ***   3695  0         0
14:04:30.513 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:27 626  ****** ****** ***** ***   ******** ****** ***   4099  0         0
14:04:31.546 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:28 630  ****** ****** ***** ***   ******** ****** ***   4434  0         0
14:04:32.531 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:29 635  ****** ****** ***** ***   ******** ****** ***   4838  0         0
14:04:33.524 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:30 640  ****** ****** ***** ***   ******** ****** ***   5173  0         0
14:04:34.557 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:31 648  ****** ****** ***** ***   ******** ****** ***   5577  0         0
14:04:35.543 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:32 653  ****** ****** ***** ***   ******** ****** ***   5912  0         0
14:04:36.562 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:33 661  ****** ****** ***** ***   ******** ****** ***   6316  0         0
14:04:37.582 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:34 667  ****** ****** ***** ***   ******** ****** ***   6701  0         0
14:04:38.619 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:35 733  ****** ****** ***** ***   ******** ****** ***   7055  0         0
14:04:39.652 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:36 740  ****** ****** ***** ***   ******** ****** ***   7459  0         0
14:04:40.640 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:37 745  ****** ****** ***** ***   ******** ****** ***   7794  0         0
14:04:41.672 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:38 752  ****** ****** ***** ***   ******** ****** ***   8198  0         0
14:04:42.658 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:39 758  ****** ****** ***** ***   ******** ****** ***   8533  0         0
14:04:43.676 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:40 765  ****** ****** ***** ***   ******** ****** ***   8937  0         0
14:04:44.651 -> 0    100.0 ********** ********** **** 03/25/2021 18:04:41 770  ****** ****** ***** ***   ******** ****** ***   9272  0         0
```

Autoscroll Show timestamp Newline 115200 baud Clear output

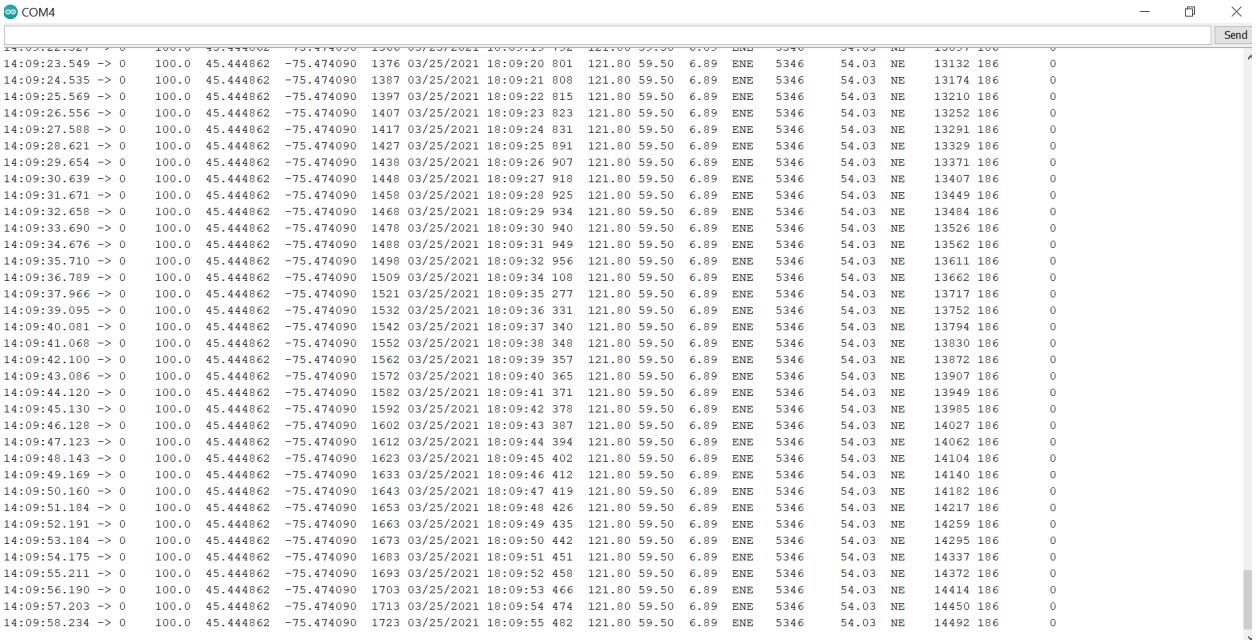


COM4

Send

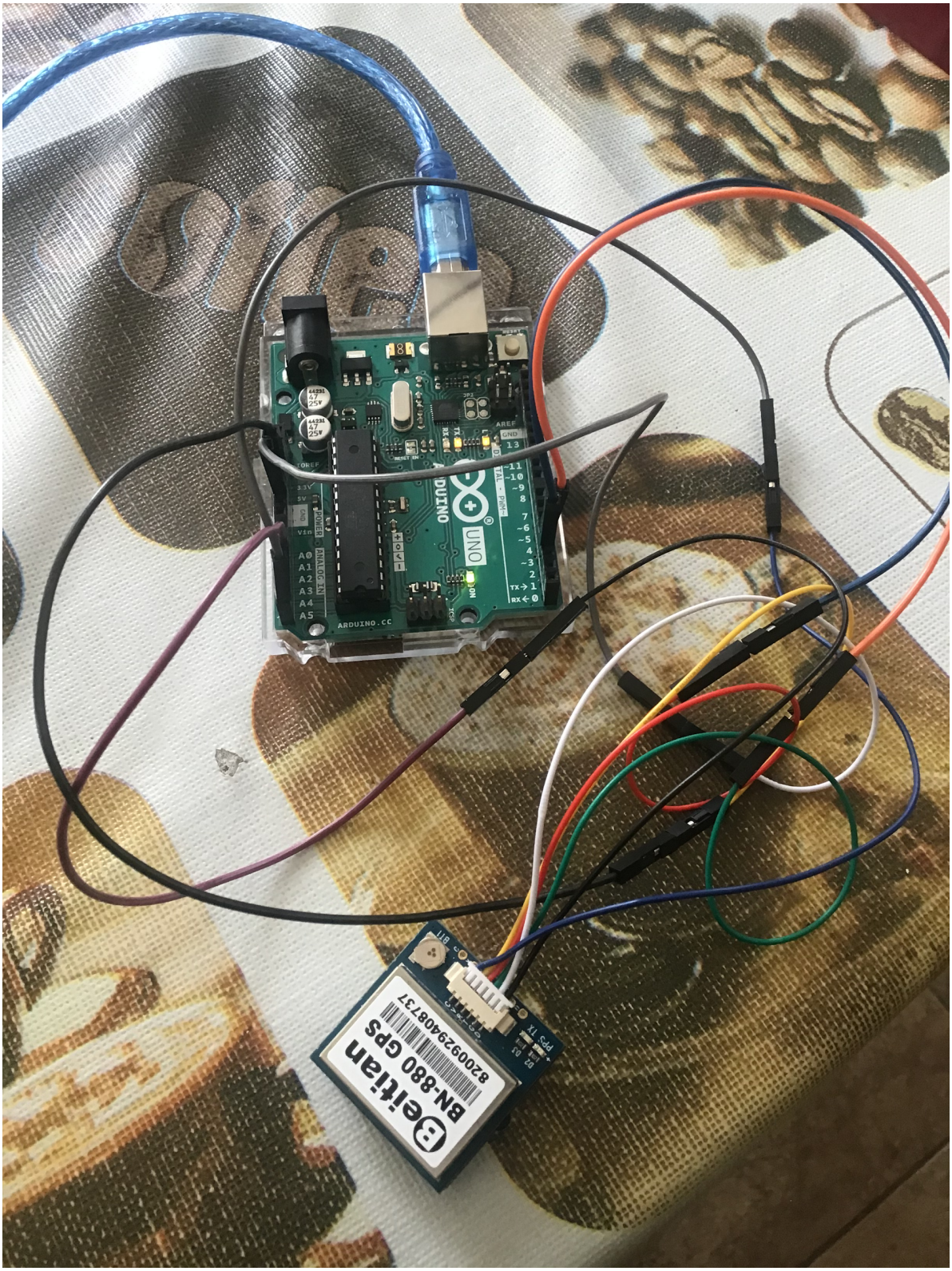
```
14:06:33.360 -> 3    4.0   45.444679  -75.474670  479  03/25/2021 18:06:30 596  119.30 10.22  3.89  N     5346     54.03  NE    60967 120       0
14:06:34.387 -> 3    4.0   45.444686  -75.474655  491  03/25/2021 18:06:31 607  121.10 10.22  3.89  N     5346     54.03  NE    61504 122       0
14:06:35.378 -> 3    4.0   45.444694  -75.474647  487  03/25/2021 18:06:32 609  120.90 39.70  5.22  NE    5346     54.03  NE    61982 124       0
14:06:36.413 -> 3    4.0   45.444709  -75.474617  494  03/25/2021 18:06:33 616  118.30 50.87  6.43  NE    5346     54.03  NE    62529 126       0
14:06:37.427 -> 3    4.0   45.444721  -75.474594  502  03/25/2021 18:06:34 624  120.20 53.60  6.83  NE    5346     54.03  NE    63007 128       0
14:06:38.415 -> 3    4.0   45.444732  -75.474571  511  03/25/2021 18:06:35 632  120.70 55.99  9.07  NE    5346     54.03  NE    63554 130       0
14:06:39.404 -> 3    4.0   45.444747  -75.474548  519  03/25/2021 18:06:36 641  118.80 52.15  7.85  NE    5346     54.03  NE    64032 132       0
14:06:40.440 -> 3    4.0   45.444751  -75.474533  537  03/25/2021 18:06:37 654  119.70 52.15  6.33  NE    5346     54.03  NE    64569 134       0
14:06:41.427 -> 3    4.0   45.444759  -75.474517  544  03/25/2021 18:06:38 661  120.00 52.15  6.26  NE    5346     54.03  NE    65037 136       0
14:06:42.462 -> 3    4.0   45.444763  -75.474502  552  03/25/2021 18:06:39 669  121.20 52.15  5.98  NE    5346     54.03  NE    65574 138       0
14:06:43.443 -> 3    4.0   45.444770  -75.474487  550  03/25/2021 18:06:40 672  121.20 55.14  5.52  NE    5346     54.03  NE    66052 140       0
14:06:44.475 -> 3    4.0   45.444774  -75.474472  568  03/25/2021 18:06:41 685  121.20 55.14  4.82  NE    5346     54.03  NE    66589 142       0
14:06:45.464 -> 3    4.0   45.444786  -75.474449  576  03/25/2021 18:06:42 692  120.90 55.14  5.78  NE    5346     54.03  NE    67057 144       0
14:06:46.451 -> 3    4.0   45.444801  -75.474411  573  03/25/2021 18:06:43 695  122.50 57.66  7.85  ENE   5346     54.03  NE    67598 146       0
14:06:47.486 -> 3    4.0   45.444816  -75.474380  584  03/25/2021 18:06:44 705  121.20 58.04  9.33  ENE   5346     54.03  NE    68070 148       0
14:06:48.472 -> 3    4.0   45.444828  -75.474357  589  03/25/2021 18:06:45 711  122.00 58.19  9.59  ENE   5346     54.03  NE    68611 150       0
14:06:49.493 -> 3    4.0   45.444835  -75.474327  598  03/25/2021 18:06:46 719  120.30 59.36  8.96  ENE   5346     54.03  NE    69083 152       0
14:06:50.513 -> 3    4.0   45.444839  -75.474311  605  03/25/2021 18:06:47 727  119.90 59.50  7.45  ENE   5346     54.03  NE    69624 154       0
14:06:51.501 -> 3    4.0   45.444828  -75.474311  622  03/25/2021 18:06:48 739  120.70 59.50  4.33  ENE   5346     54.03  NE    70086 156       0
14:06:52.527 -> 3    4.0   45.444828  -75.474296  631  03/25/2021 18:06:49 746  119.30 59.50  4.82  ENE   5346     54.03  NE    70617 158       0
14:06:53.513 -> 3    4.0   45.444831  -75.474273  638  03/25/2021 18:06:50 755  120.00 59.50  5.43  ENE   5346     54.03  NE    71079 160       0
14:06:54.547 -> 3    4.0   45.444828  -75.474258  648  03/25/2021 18:06:51 764  120.70 59.50  4.70  ENE   5346     54.03  NE    71610 162       0
14:06:55.527 -> 3    4.0   45.444839  -75.474235  654  03/25/2021 18:06:52 771  121.10 59.50  5.19  ENE   5346     54.03  NE    72072 164       0
14:06:56.564 -> 3    4.0   45.444835  -75.474227  663  03/25/2021 18:06:53 779  120.40 59.50  3.35  ENE   5346     54.03  NE    72603 166       0
14:06:57.554 -> 3    4.0   45.444843  -75.474212  670  03/25/2021 18:06:54 787  122.00 59.50  5.24  ENE   5346     54.03  NE    73065 168       0
14:06:58.587 -> 3    4.0   45.444831  -75.474212  680  03/25/2021 18:06:55 796  121.10 59.50  2.93  ENE   5346     54.03  NE    73596 170       0
14:06:59.573 -> 3    4.0   45.444828  -75.474205  730  03/25/2021 18:06:56 861  122.30 59.50  4.24  ENE   5346     54.03  NE    74127 172       0
14:07:00.641 -> 3    4.0   45.444831  -75.474182  759  03/25/2021 18:06:57 876  122.50 59.50  3.69  ENE   5346     54.03  NE    74589 174       0
14:07:01.661 -> 3    4.0   45.444843  -75.474166  766  03/25/2021 18:06:58 882  121.40 59.50  4.28  ENE   5346     54.03  NE    75120 176       0
14:07:02.649 -> 3    4.0   45.444854  -75.474143  774  03/25/2021 18:06:59 891  120.40 59.50  5.44  ENE   5346     54.03  NE    75582 178       0
14:07:03.684 -> 3    4.0   45.444858  -75.474128  782  03/25/2021 18:07:00 898  121.50 59.50  5.02  ENE   5346     54.03  NE    76113 180       0
14:07:04.660 -> 3    4.0   45.444862  -75.474113  789  03/25/2021 18:07:01 906  121.30 59.50  5.11  ENE   5346     54.03  NE    76575 182       0
14:07:05.688 -> 3    4.0   45.444858  -75.474105  797  03/25/2021 18:07:02 913  123.30 59.50  5.44  ENE   5346     54.03  NE    77106 184       0
14:07:06.705 -> 3    4.0   45.444862  -75.474090  805  03/25/2021 18:07:03 922  121.80 59.50  6.89  ENE   5346     54.03  NE    77568 186       0
14:07:07.681 -> 0    100.0 45.444862  -75.474090  1816 03/25/2021 18:07:04 979  121.80 59.50  6.89  ENE   5346     54.03  NE    78109 186       0
14:07:08.855 -> 0    100.0 45.444862  -75.474090  2989 03/25/2021 18:07:06 150  121.80 59.50  6.89  ENE   5346     54.03  NE    78634 186       0
```

Autoscroll Show timestamp Newline 115200 baud Clear output



The figure above shows the results of this test on the location subsystem.

Figure 11 - The Location Subsystem



The figure above shows the setup of the location subsystem.

Test 2

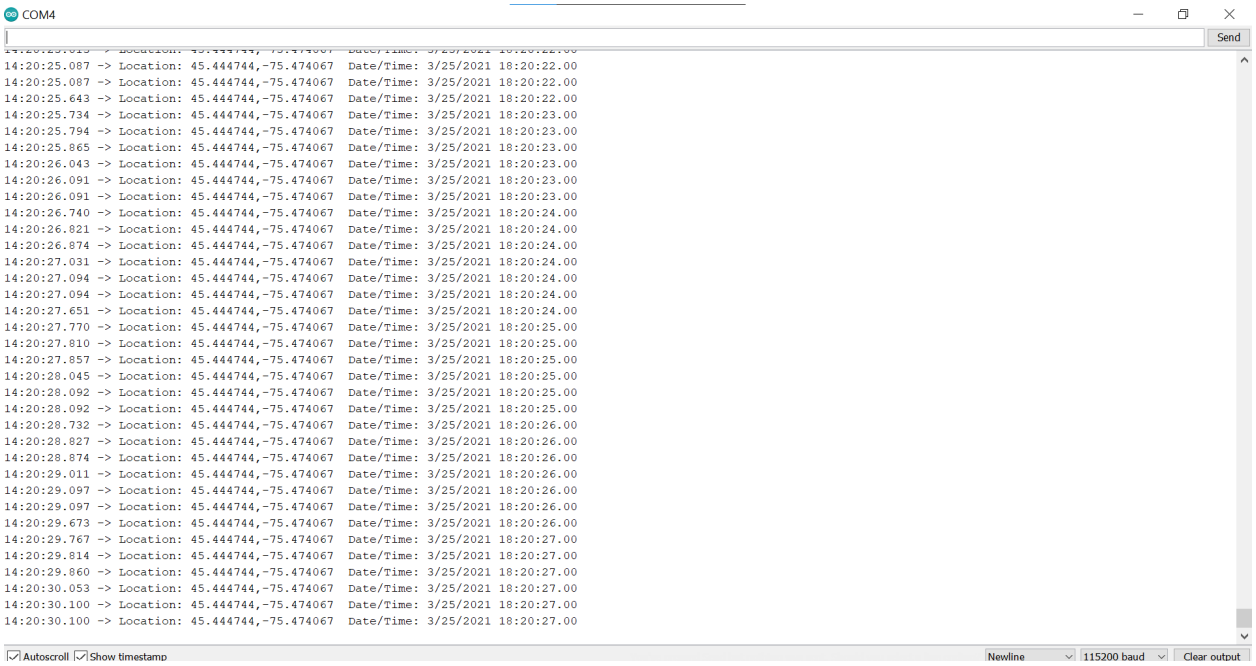
Table 7 - Test Plan 2 for the Location Subsystem for Prototype III

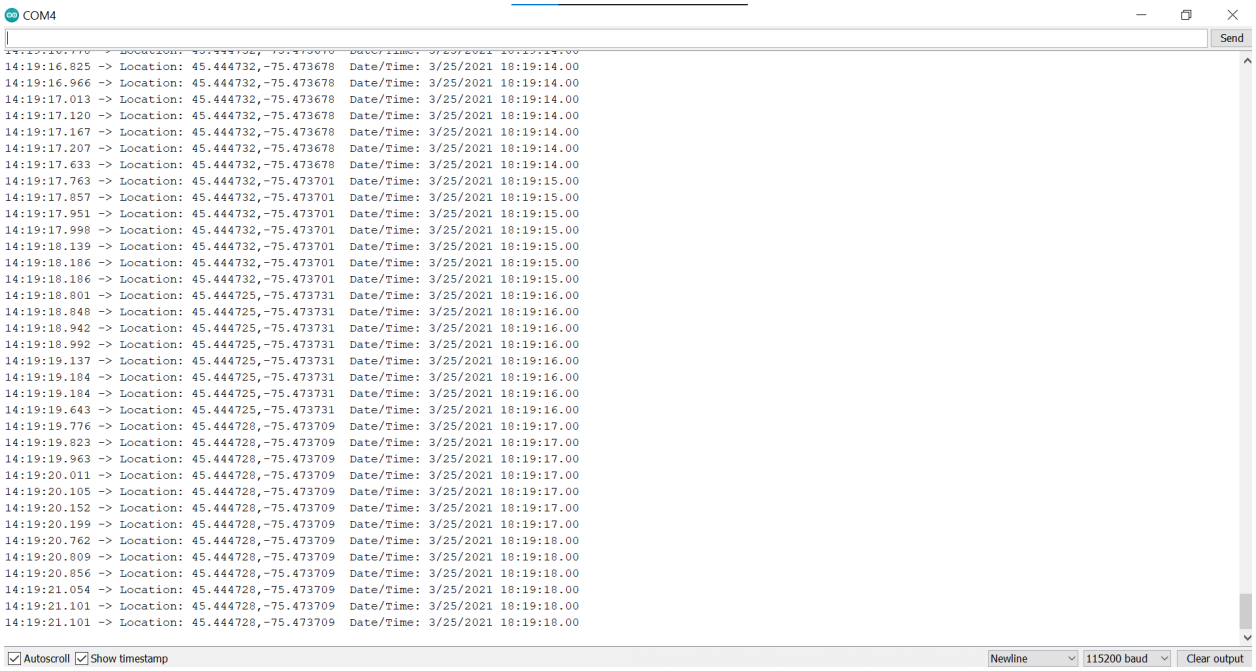
Test ID	Member(s) Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
2	Elsa	To test if the Beitian BN-880 can satisfy design criteria.	The prototype contained the Arduino, the Beitian BN-880 and connecting wires. A code was used to test if the Beitian BN-880 printed the coordinate data to the serial monitor.	The results were coordinate data that was printed to the serial monitor. The data was live as it changed according to the movement. In the future, either this code or the one used in the next cde will be used in the final prototype	March 25th, 2021 Test Duration: approximately 3 minutes.	The prototype was deemed satisfactory when it printed live coordinates to the serial monitor.

The second test tested the same as the previous test for this subsystem. This code had less features than the previous test but still satisfies design criteria. In the future, this code may be used in the final prototype. The code was found in the TinyGPS++ library. This library can be downloaded from here.

Results

Figure 12 - Code Results of the Location Subsystem





The figure above shows the code results of the second test of the location subsystem.

The setup was the same as the previous test.

Interconnections

This section contains testing done on the interconnected subsystems. The first two tests tested only two subsystems while the final test was done on the voice, light and altitude subsystems.

Light and Altitude Subsystem

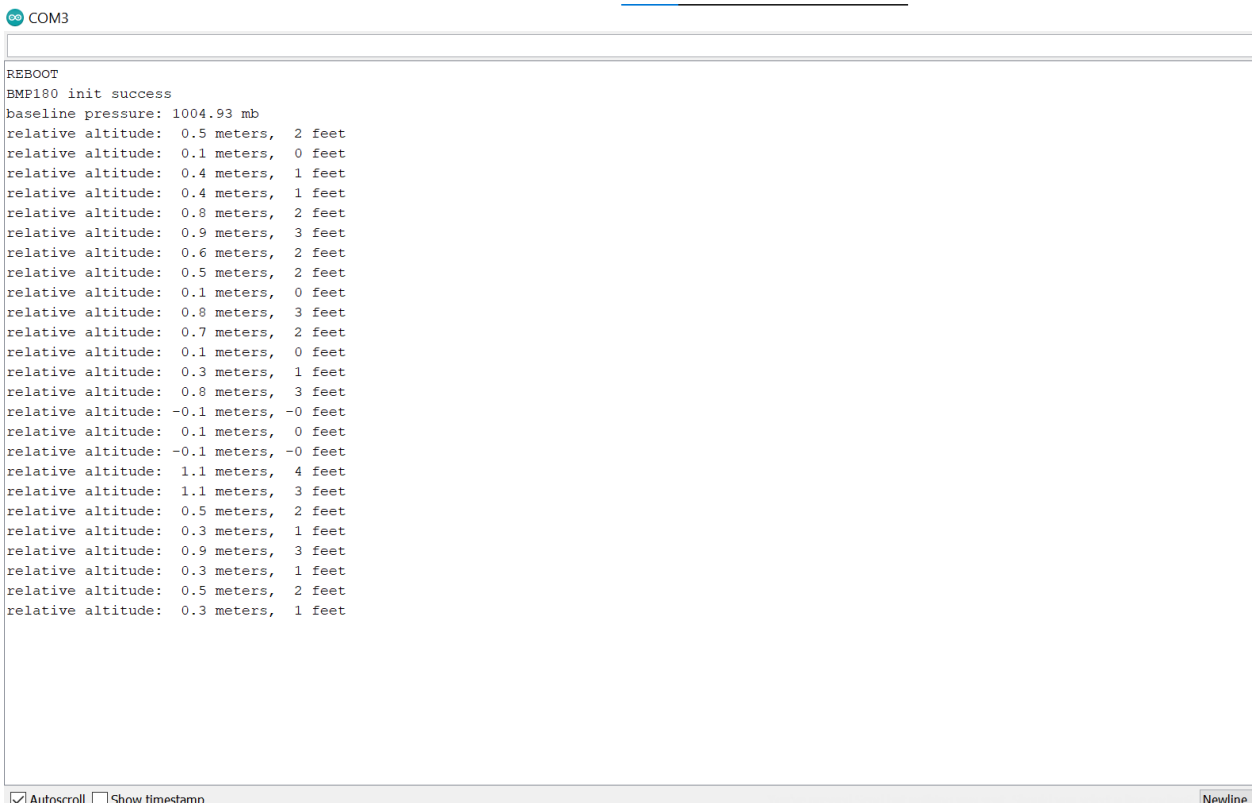
Table 8 - Test Plan 1 for the Light and Altitude Subsystem for Prototype III

Test ID	Member(s) Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
2	Elsa	The test was to ensure that these two subsystems could work together and not if they satisfied their design criteria.	The prototype contained the LED lights, the 8 resistors, the BMP180 and the connecting wires. An integrated code was used using the codes from previous prototypes.	The results were both subsystems working properly. In the future, these results will be used to form the complete prototype.	March 25th, 2021 Test Duration: approximately 5 minutes.	The code was deemed satisfactory when the subsystems functioned properly.

In this test, an integrated code created from the two isolated codes for these subsystems were used to test if these subsystems could work together in one Arduino. When the lights turned on and the BMP180 printed to the serial monitor, the test was stopped. These results will be used to connect to the other subsystems at the end.

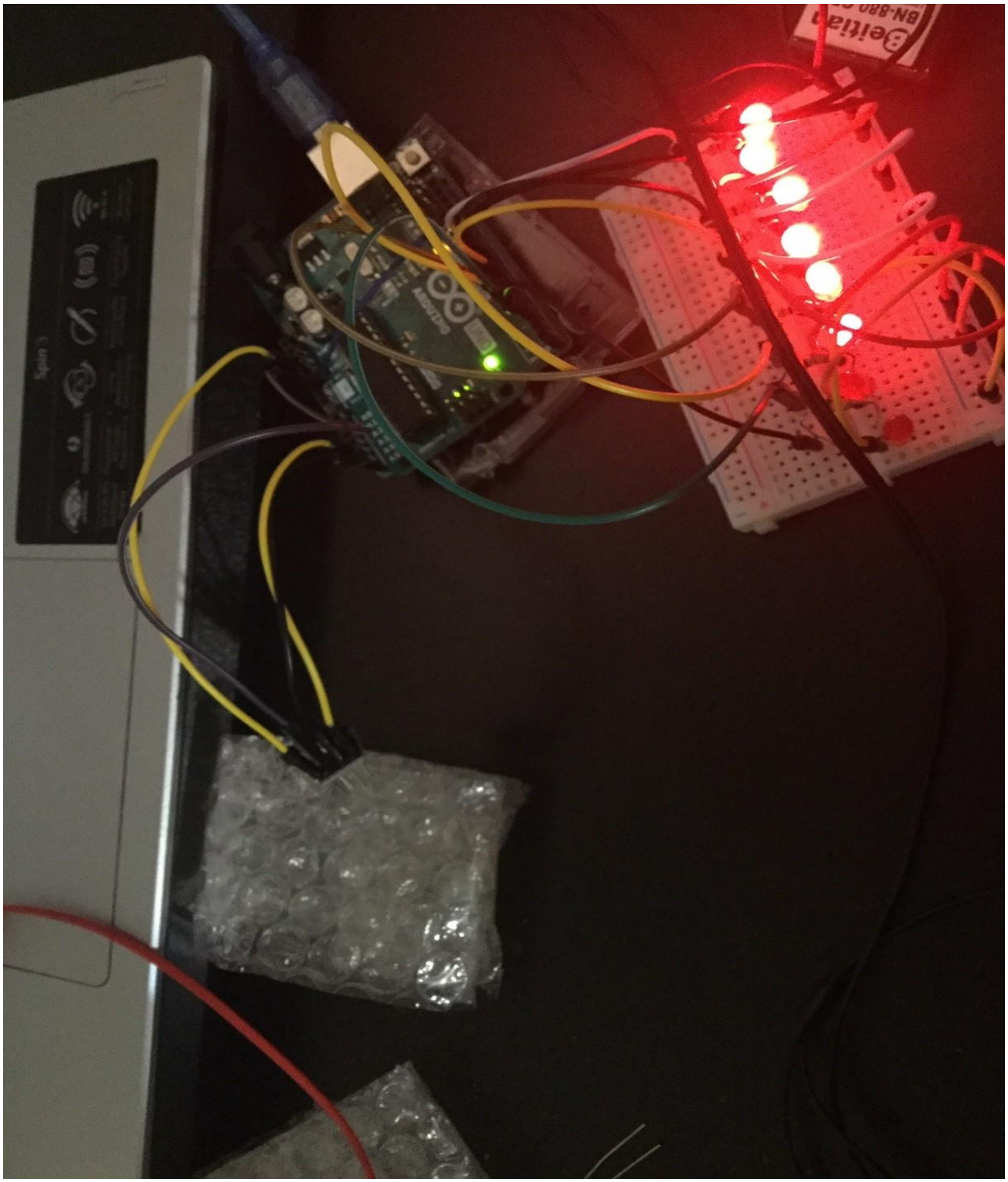
Results

Figuru 13 - Code Results of the Light and Altitude Subsystem



The figure above shows the code results of the light and altitude subsystem.

Figure 14 - The Light and Altitude Subsystem
*



The figure above shows the setup of the light and altitude subsystem.

Voice and Location Subsystem

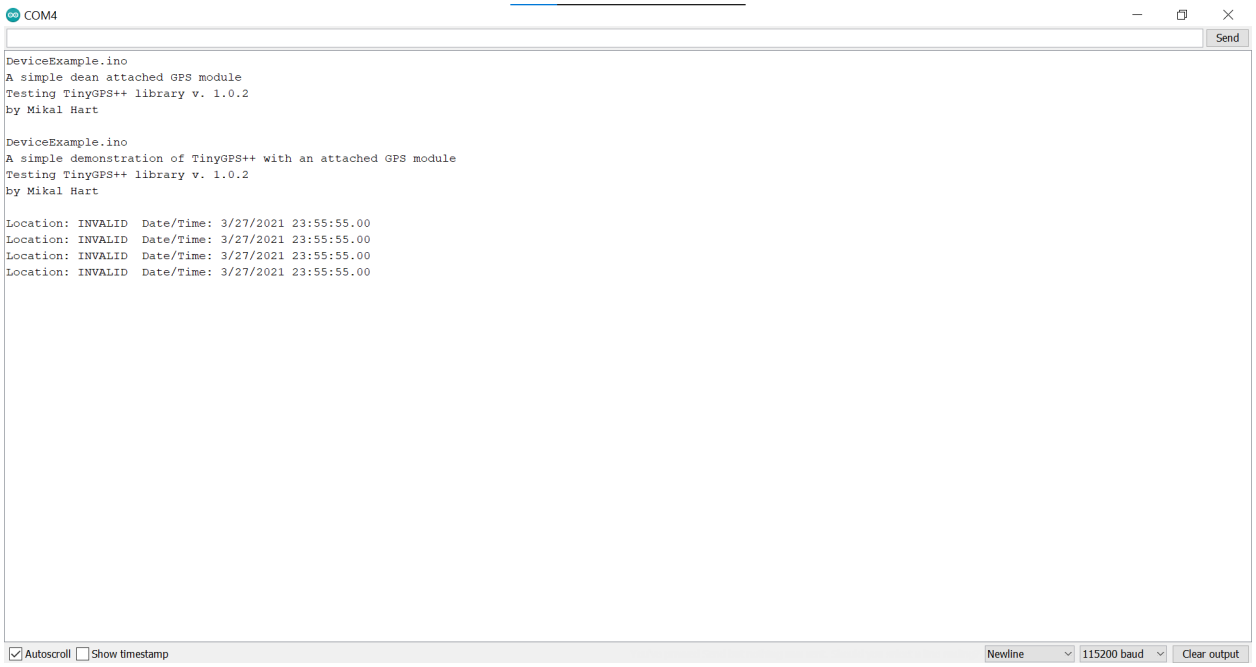
Table 9 - Test Plan 1 for the Voice and Location Subsystems for Prototype III

Test ID	Member(s) Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
2	Elsa	This test was to determine if the location and voice subsystems could work together.	The prototype included an Arduino, the two speakers, the Beitian BN-880 and the connecting wires. An integrated code was used to test if the subsystems worked properly.	The subsystems functioned properly. These results will be used in the final prototype to setup and integrate into one code.	March 25th, 2021 Test Duration: approximately 5 minutes.	The results were deemed satisfactory when the subsystems function properly.

In this test, an integrated code created from the two isolated codes for these subsystems were used to test if these subsystems could work together in one Arduino. When the speakers relayed the message and the Beitian BN-880 printed to the serial monitor, the test was stopped. These results will be used to connect to the other subsystems at the end.

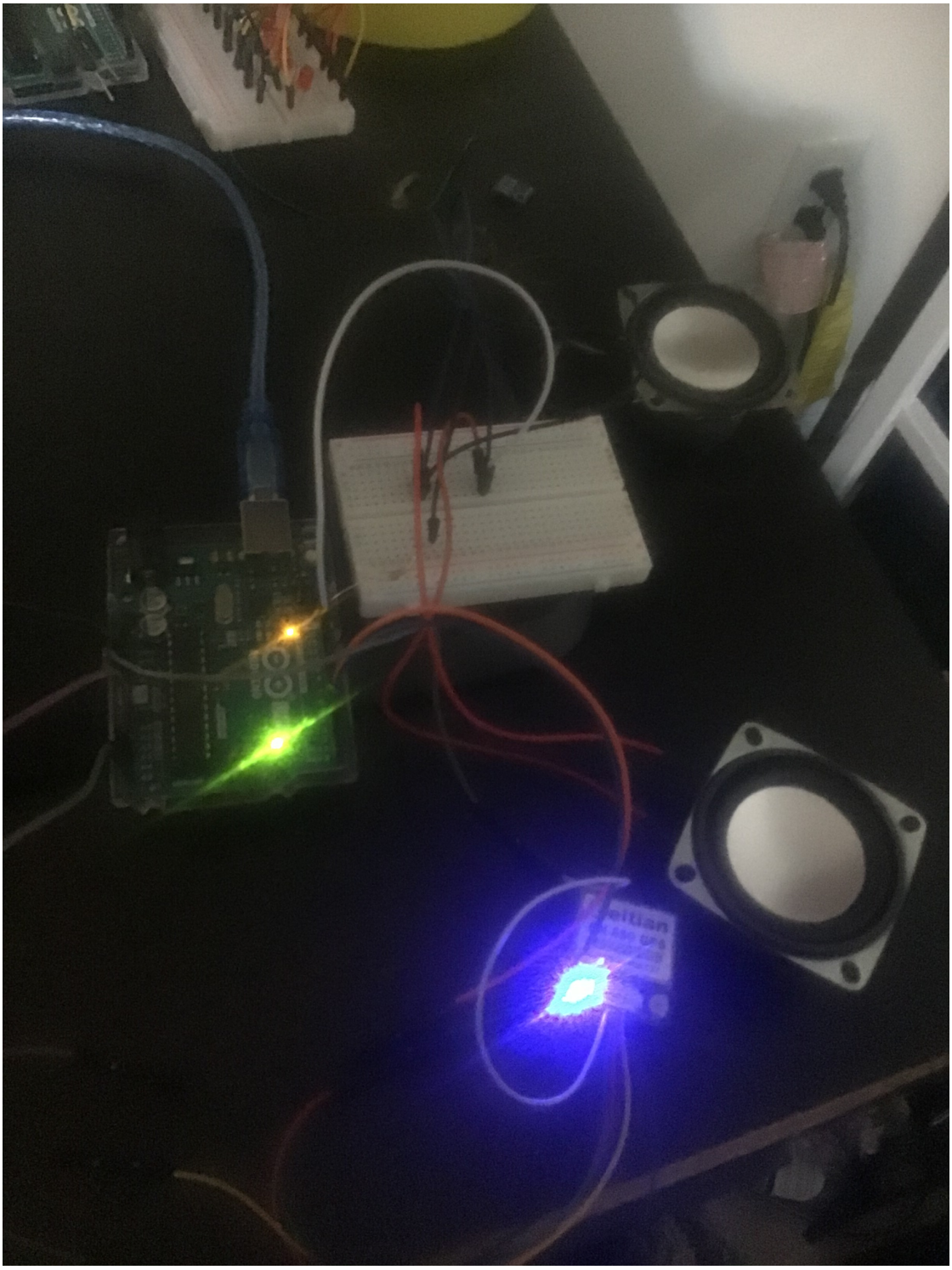
Results

Figure 15 - Code Results of the Voice and Location Subsystem



The figure above shows the code results of this test. As the test was to determine if the Beitian BN-880 could work with the voice subsystem, it was tested indoors (it is for this reason that it printed “invalid”).

Figure 16 - The Voice and Location Subsystem



The figure above shows the setup of these two subsystems.

Final Prototype

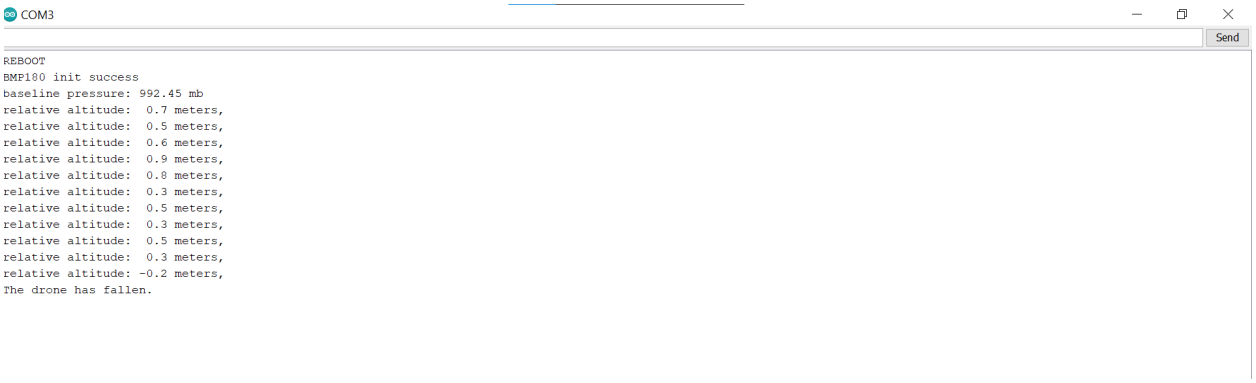
Table 10 - Test Plan for the Complete Prototype (Without the Location Subsystem) for Prototype III

Test ID	Member(s) Responsible	Test Objective	Description of Prototype Used and of Basic Test Method	Descriptions of Results and How These Results Will Be Used	Estimated Test Duration and Planned Start Date	Stopping criteria
1	Elsa	This test was to determine if the voice and light subsystem could be dependent on the altitude subsystem.	A code using a threshold value was used to test if the voice and light subsystems would turn on after the altitude subsystem went below the threshold value.	The results were the voice and light subsystem turning on only after the altitude subsystem went below the threshold value. In the future, the location subsystem will be integrated in the prototype.	March 27th, 2021 Test Duration: approximately 4.5 minutes.	The test was complete after the voice and light subsystem turned on after the altitude subsystem went below the threshold value.

This test tested if the voice and light subsystems could be reliant on the altitude subsystem. The results were the voice subsystem relaying the automated message and the lights turning on after the altitude subsystem fell below the threshold value. In the future, the location subsystem will be integrated in this final prototype.

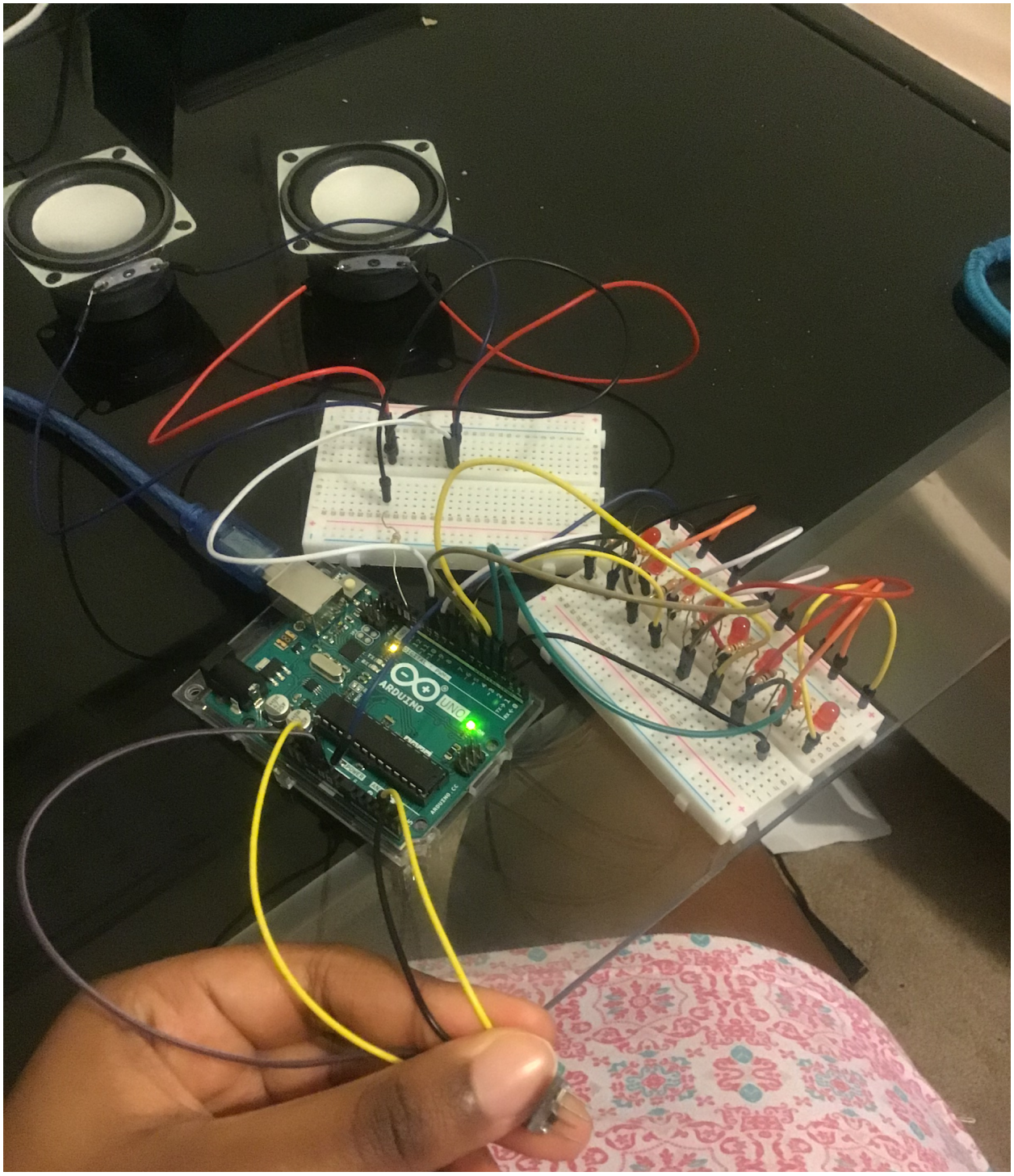
Results

Figure 17 - Code Results of the Complete Prototype



The figure above shows the code results of the complete prototype.

Figure 18 - The Complete Prototype



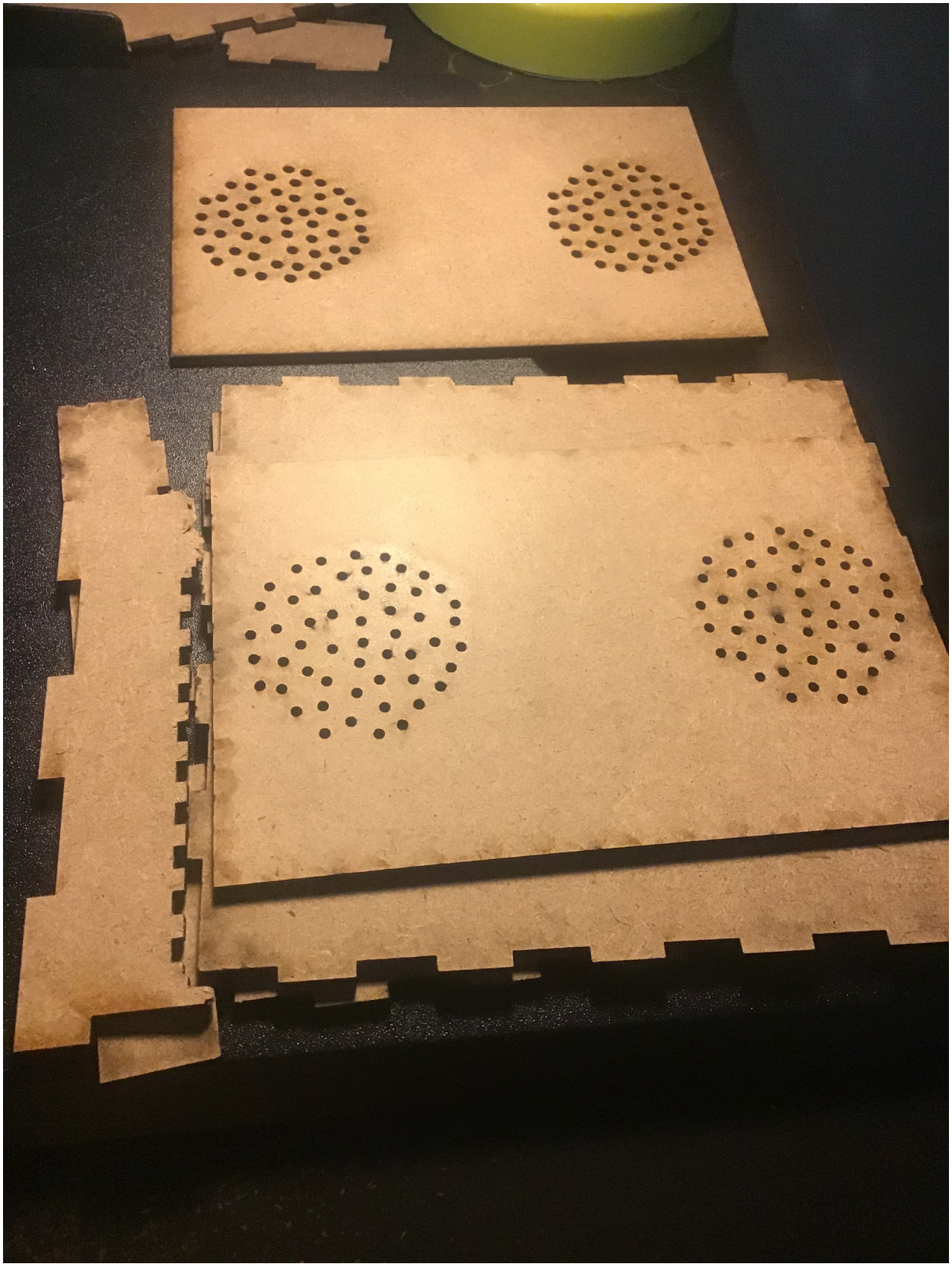
The figure above shows the setup of the complete prototype.

Encasing

As a part of the overall system, the group has created an acrylic box via laser cutting. The box plays a role as a protective case for the electronics contained inside the box. Our TAs recommended that the box have speaker holes on the top surface as well as a separate piece with congruent holes to be placed on top of the speaker holes. In between, there is plastic that will prevent debris from entering inside the box. The holes will allow sound from the speakers to go through so that pedestrians will be able to hear it. The box has a height of 54.372 mm, a width of 271.920 mm, and a length of 152.585 mm. The acrylic box can accommodate the prototype while still ensuring that the weight and cost are minimized.

Prototype I

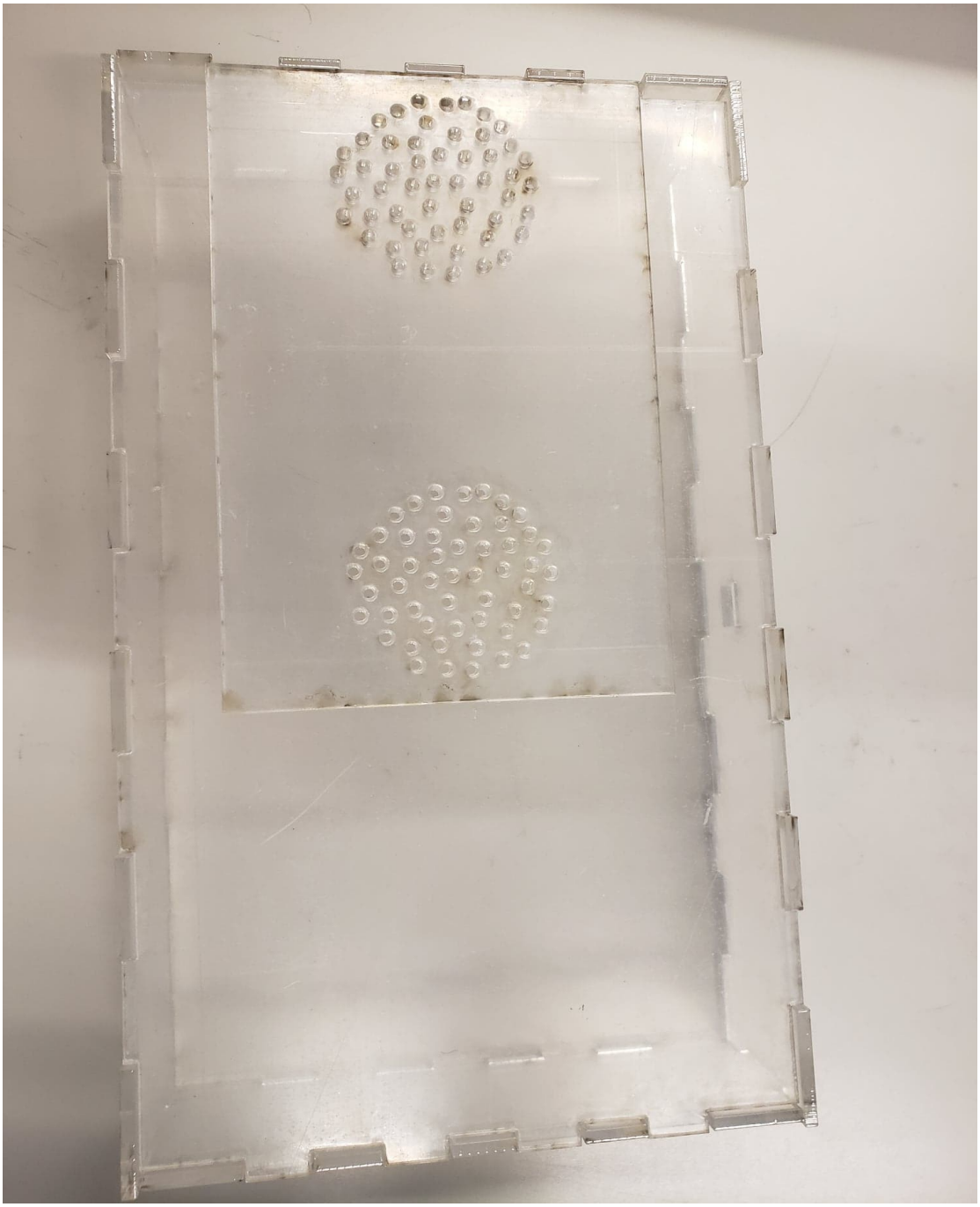
Figure 19 - The Initial Prototype Encasing



The figure above demonstrated the initial encasing for the prototype. In this step, we did not laser cut all the parts for the case because it was unnecessary but was only used to confirm our dimensions. Luckily, we cut this using MDF before using our acrylic because we found out that our final prototype did not fit using these dimensions. Consequently, we made the necessary adjustments.

Prototype II

Figure 20 - The Final Prototype Encasing

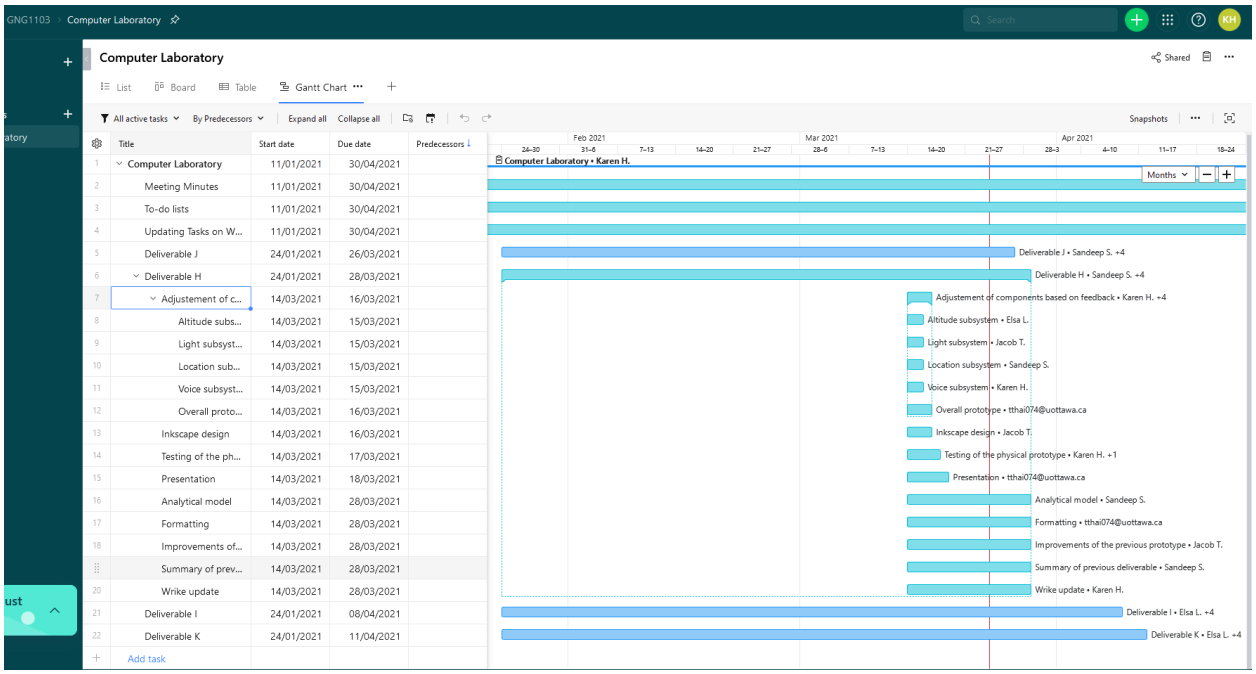


The figure above shows the final case design that will be used to protect our final prototype. We used a laser cutter to get a detailed cut and to ensure that no water gets inside and damages our prototype. It was made a little bigger than needed to guarantee that all the components of our prototype can fit easily. The only openings made on the encasing is to ensure the voice from the speakers come out. To prevent debris from entering our case, we have laser cut an extra acrylic part with congruent holes as the top face of the box to protect our prototype.

Future Work

Before Design Day, the location subsystem will be added to the integrated prototype and tested again. Additionally, we need to test the prototype again after soldering to ensure that nothing was damaged in the process. Lastly, we need to ensure that our prototype is waterproof as per client's request.

Wrike update

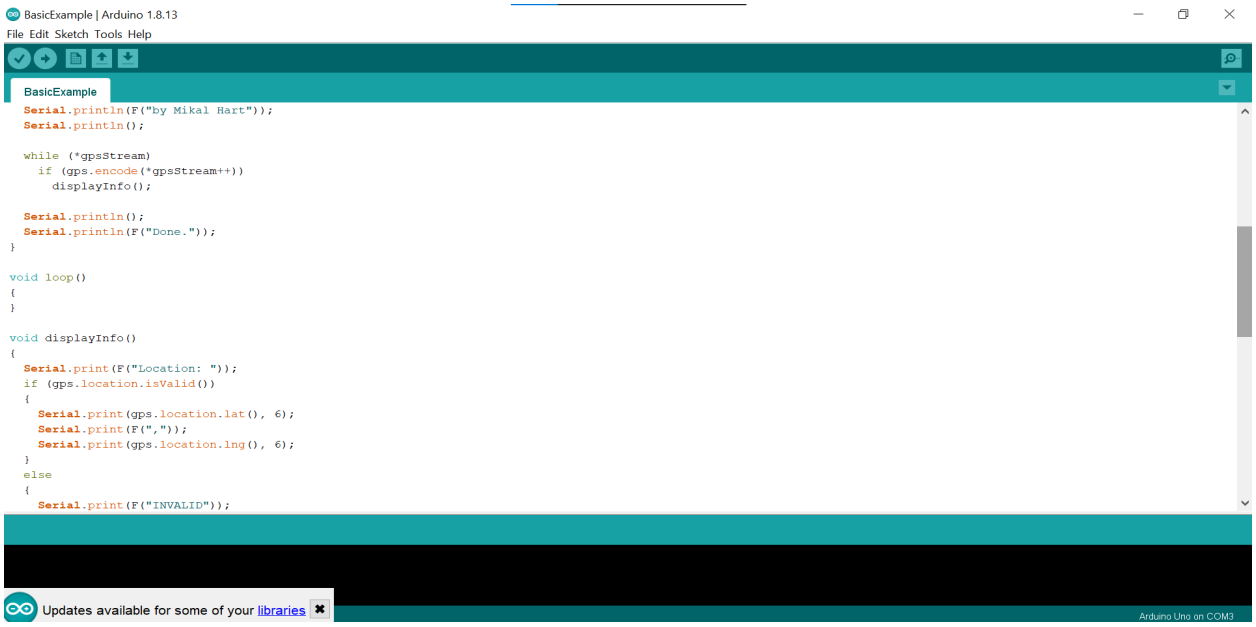


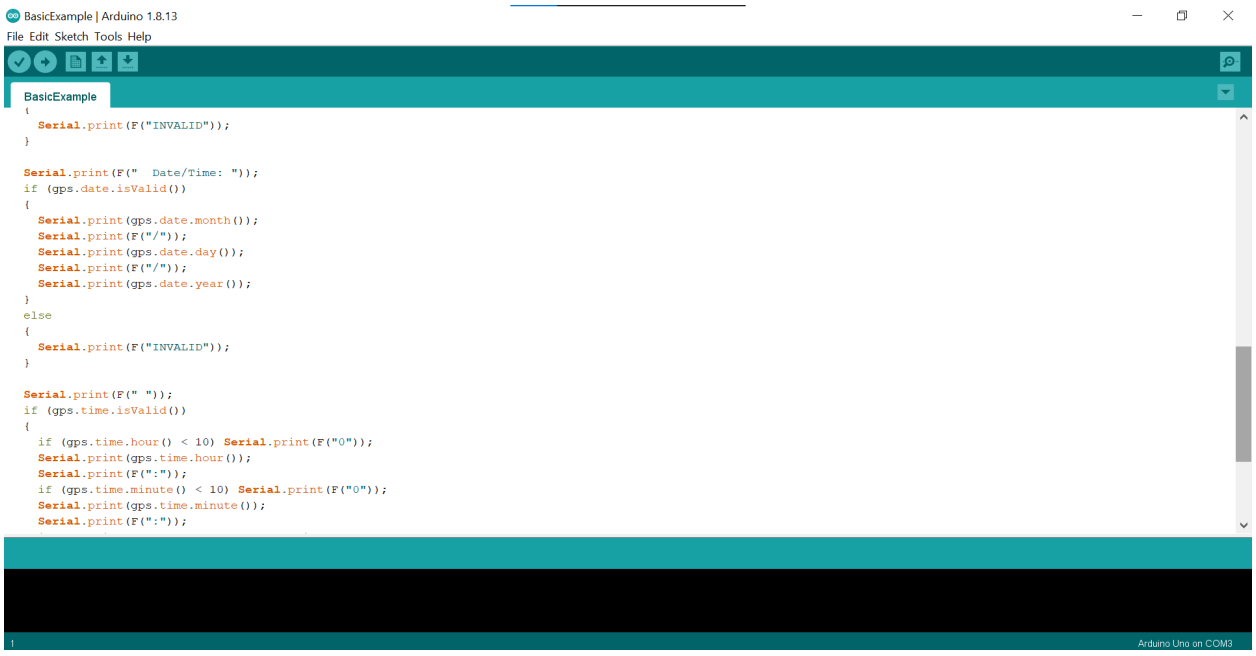
Appendices of Prototype 2.5

Appendix I

Code for the Location Subsystem

Test 1


BasicExample | Arduino 1.8.13
File Edit Sketch Tools Help
BasicExample
Serial.println(F("by Mikal Hart"));
Serial.println();
while (*gpsStream)
if (gps.encode(*gpsStream++))
displayInfo();
Serial.println();
Serial.println(F("Done."));
}
void loop()
{
}
void displayInfo()
{
Serial.print(F("Location: "));
if (gps.location.isValid())
{
Serial.print(gps.location.lat(), 6);
Serial.print(F(","));
Serial.print(gps.location.lng(), 6);
}
else
{
Serial.print(F("INVALID"));
Updates available for some of your libraries
Arduino Uno on COM3


BasicExample | Arduino 1.8.13
File Edit Sketch Tools Help
BasicExample
{
Serial.print(F("INVALID"));
}
Serial.print(F(" Date/Time: "));
if (gps.date.isValid())
{
Serial.print(gps.date.month());
Serial.print(F("/"));
Serial.print(gps.date.day());
Serial.print(F("/"));
Serial.print(gps.date.year());
}
else
{
Serial.print(F("INVALID"));
}
Serial.print(F(" "));
if (gps.time.isValid())
{
if (gps.time.hour() < 10) Serial.print(F("0"));
Serial.print(gps.time.hour());
Serial.print(F(":"));
if (gps.time.minute() < 10) Serial.print(F("0"));
Serial.print(gps.time.minute());
Serial.print(F(":"));
1
Arduino Uno on COM3

BasicExample | Arduino 1.8.13

File Edit Sketch Tools Help

BasicExample §

```
#include <TinyGPS++.h>
/*
   This sample sketch should be the first you try out when you are testing a TinyGPS++
   (TinyGPSPlus) installation.  In normal use, you feed TinyGPS++ objects characters from
   a serial NMEA GPS device, but this example uses static strings for simplicity.
*/

// A sample NMEA stream.
const char *gpsStream =
  "$GPRMC,045103.000,A,3014.1984,N,09749.2872,W,0.67,161.46,030913,,,A*7C\r\n"
  "$GPGGA,045104.000,3014.1985,N,09749.2873,W,1,09,1.2,211.6,M,-22.5,M,,0000*62\r\n"
  "$GPRMC,045200.000,A,3014.3820,N,09748.9514,W,36.88,65.02,030913,,,A*77\r\n"
  "$GPGGA,045201.000,3014.3864,N,09748.9411,W,1,10,1.2,200.8,M,-22.5,M,,0000*6C\r\n"
  "$GPRMC,045251.000,A,3014.4275,N,09749.0626,W,0.51,217.94,030913,,,A*7D\r\n"
  "$GPGGA,045252.000,3014.4273,N,09749.0628,W,1,09,1.3,206.9,M,-22.5,M,,0000*6F\r\n";

// The TinyGPS++ object
TinyGPSPlus gps;

void setup()
{
  Serial.begin(115200);

  Serial.println(F("BasicExample.ino"));
  Serial.println(F("Basic demonstration of TinyGPS++ (no device needed)"));
  Serial.print(F("Testing TinyGPS++ library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
  Serial.println(F("by Mikal Hart"));
```

1 Arduino Uno on COM3

The code was retrieved from this folder.

Test 2

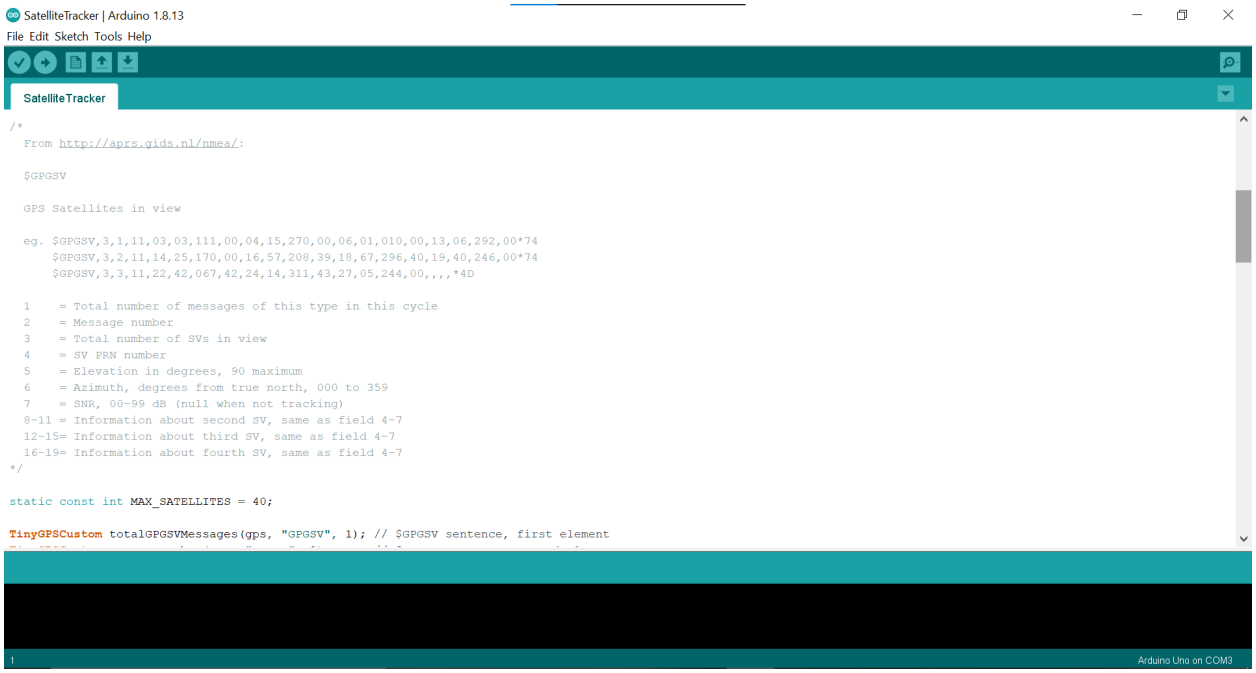
SatelliteTracker | Arduino 1.8.13

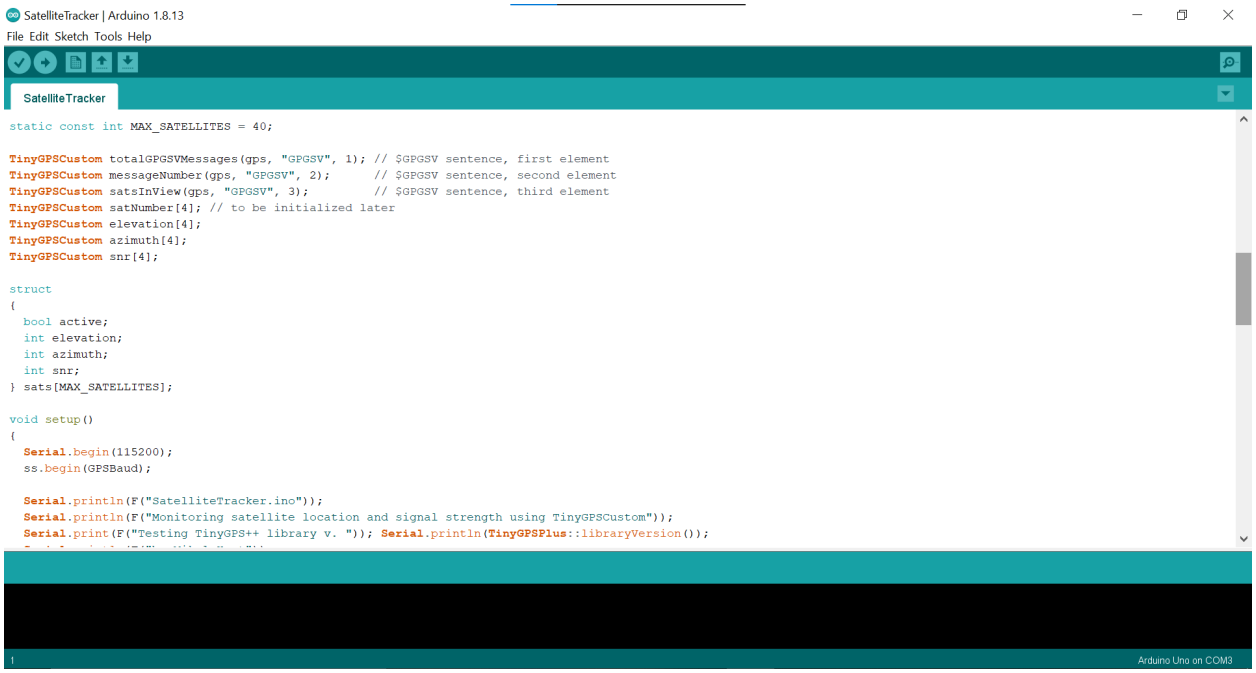
File Edit Sketch Tools Help

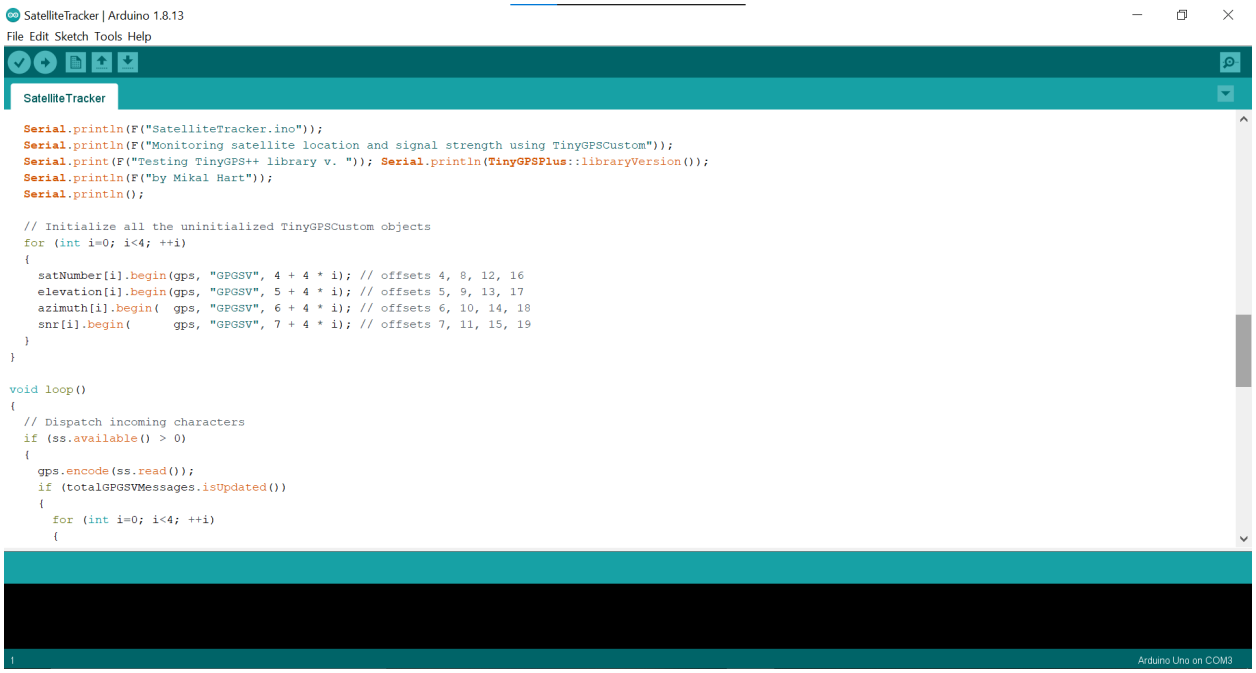
SatelliteTracker

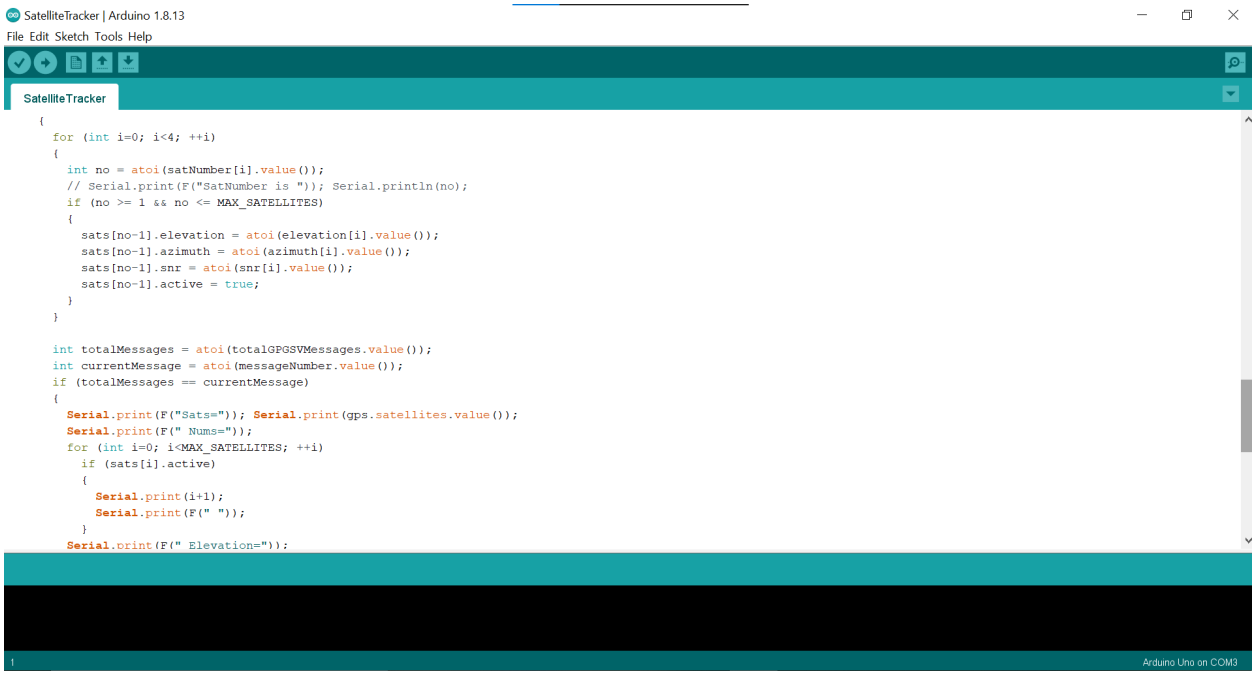
```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
/*
   This sample code demonstrates how to use an array of TinyGPSCustom objects
   to monitor all the visible satellites.

   Satellite numbers, elevation, azimuth, and signal-to-noise ratio are not
   normally tracked by TinyGPS++, but by using TinyGPSCustom we get around this.

   The simple code also demonstrates how to use arrays of TinyGPSCustom objects,
   each monitoring a different field of the $GPGSV sentence.

   It requires the use of SoftwareSerial, and assumes that you have a
   4800-baud serial GPS device hooked up on pins 4(RX) and 3(TX).
*/
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 4800;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

/*
  From http://aprs.gids.nl/nmea/:
```

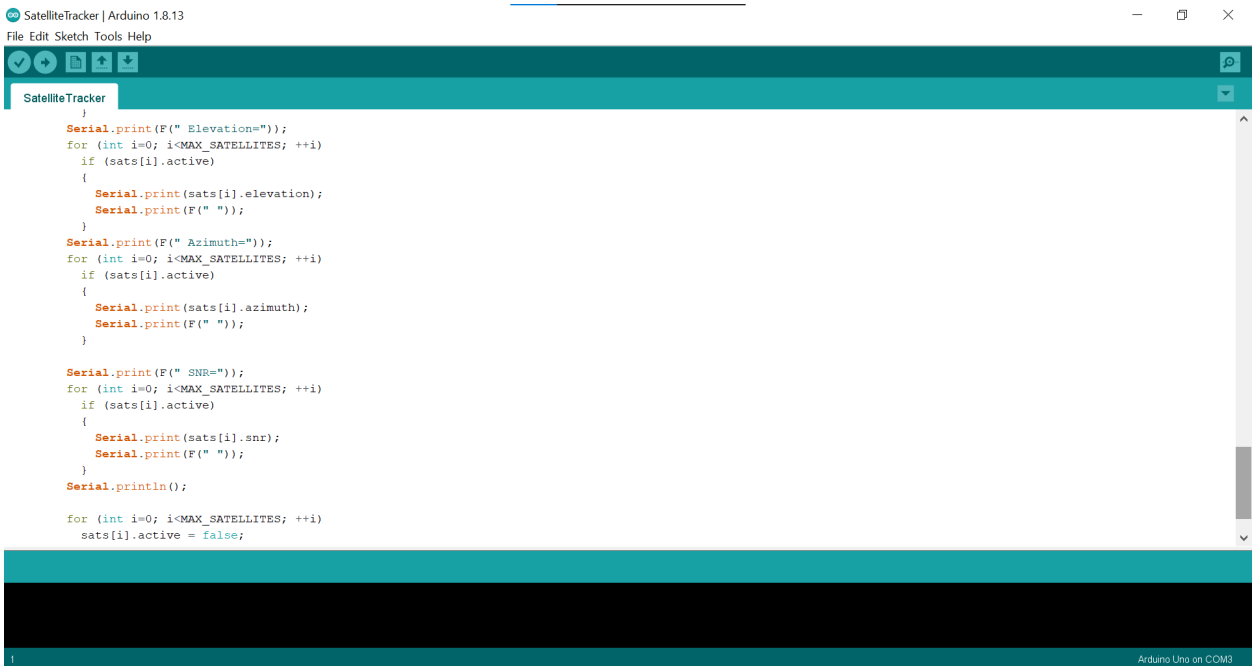
1 Arduino Uno on COM3

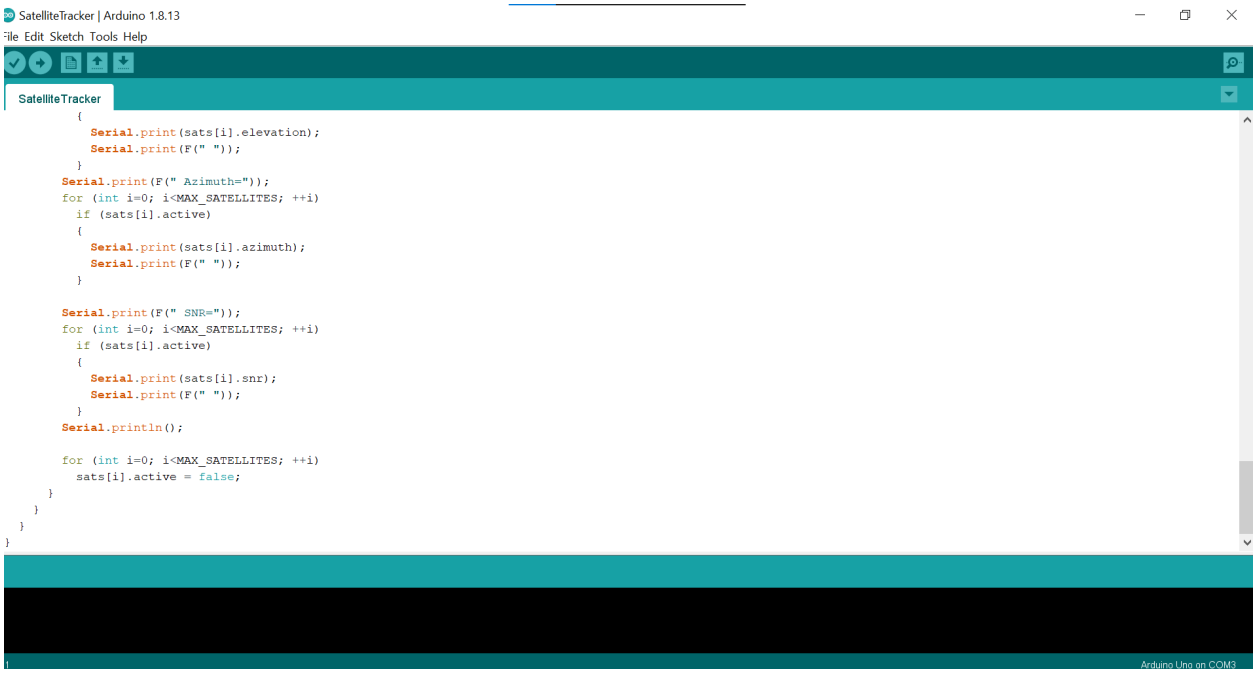

SatelliteTracker | Arduino 1.8.13
File Edit Sketch Tools Help
SatelliteTracker
/*
From http://aprs.gids.nl/nmea/:
$GPGSV
GPS Satellites in view
eg. $GPGSV,3,1,11,03,03,111,00,04,15,270,00,06,01,010,00,13,06,292,00*74
$GPGSV,3,2,11,14,25,170,00,16,57,208,39,18,67,296,40,19,40,246,00*74
$GPGSV,3,3,11,22,42,067,42,24,14,311,43,27,05,244,00,,,,*4D
1 = Total number of messages of this type in this cycle
2 = Message number
3 = Total number of SVs in view
4 = SV PRN number
5 = Elevation in degrees, 90 maximum
6 = Azimuth, degrees from true north, 000 to 359
7 = SNR, 00-99 dB (null when not tracking)
8-11 = Information about second SV, same as field 4-7
12-15= Information about third SV, same as field 4-7
16-19= Information about fourth SV, same as field 4-7
*/
static const int MAX_SATELLITES = 40;
TinyGPSCustom totalGPGSVMessages(gps, "GPGSV", 1); // $GPGSV sentence, first element
1
Arduino Uno on COM3


SatelliteTracker | Arduino 1.8.13
File Edit Sketch Tools Help
SatelliteTracker
static const int MAX_SATELLITES = 40;
TinyGPSCustom totalGPGSVMessages(gps, "GPGSV", 1); // $GPGSV sentence, first element
TinyGPSCustom messageNumber(gps, "GPGSV", 2); // $GPGSV sentence, second element
TinyGPSCustom satsInView(gps, "GPGSV", 3); // $GPGSV sentence, third element
TinyGPSCustom satNumber[4]; // to be initialized later
TinyGPSCustom elevation[4];
TinyGPSCustom azimuth[4];
TinyGPSCustom snr[4];
struct
{
bool active;
int elevation;
int azimuth;
int snr;
} sats[MAX_SATELLITES];
void setup()
{
Serial.begin(115200);
ss.begin(GPSBaud);
Serial.println(F("SatelliteTracker.ino"));
Serial.println(F("Monitoring satellite location and signal strength using TinyGPSCustom"));
Serial.print(F("Testing TinyGPS++ library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
1
Arduino Uno on COM3


SatelliteTracker | Arduino 1.8.13
File Edit Sketch Tools Help
SatelliteTracker
Serial.println(F("SatelliteTracker.ino"));
Serial.println(F("Monitoring satellite location and signal strength using TinyGPSCustom"));
Serial.print(F("Testing TinyGPS++ library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
Serial.println(F("by Mikal Hart"));
Serial.println();
// Initialize all the uninitialized TinyGPSCustom objects
for (int i=0; i<4; ++i)
{
satNumber[i].begin(gps, "GPGSV", 4 + 4 * i); // offsets 4, 8, 12, 16
elevation[i].begin(gps, "GPGSV", 5 + 4 * i); // offsets 5, 9, 13, 17
azimuth[i].begin(gps, "GPGSV", 6 + 4 * i); // offsets 6, 10, 14, 18
snr[i].begin(gps, "GPGSV", 7 + 4 * i); // offsets 7, 11, 15, 19
}
}
void loop()
{
// Dispatch incoming characters
if (ss.available() > 0)
{
gps.encode(ss.read());
if (totalGPGSVMessages.isUpdated())
{
for (int i=0; i<4; ++i)
{
1
Arduino Uno on COM3


SatelliteTracker | Arduino 1.8.13
File Edit Sketch Tools Help
SatelliteTracker
{
for (int i=0; i<4; ++i)
{
int no = atoi(satNumber[i].value());
// Serial.print(F("SatNumber is ")); Serial.println(no);
if (no >= 1 && no <= MAX_SATELLITES)
{
sats[no-1].elevation = atoi(elevation[i].value());
sats[no-1].azimuth = atoi(azimuth[i].value());
sats[no-1].snr = atoi(snr[i].value());
sats[no-1].active = true;
}
}
int totalMessages = atoi(totalGPGSVMessages.value());
int currentMessage = atoi(messageNumber.value());
if (totalMessages == currentMessage)
{
Serial.print(F("Sats=")); Serial.print(gps.satellites.value());
Serial.print(F(" Nums="));
for (int i=0; i<MAX_SATELLITES; ++i)
if (sats[i].active)
{
Serial.print(i+1);
Serial.print(F(" "));
}
Serial.print(F(" Elevation="));
1
Arduino Uno on COM3





The code was retrieved from this folder.

Appendix II

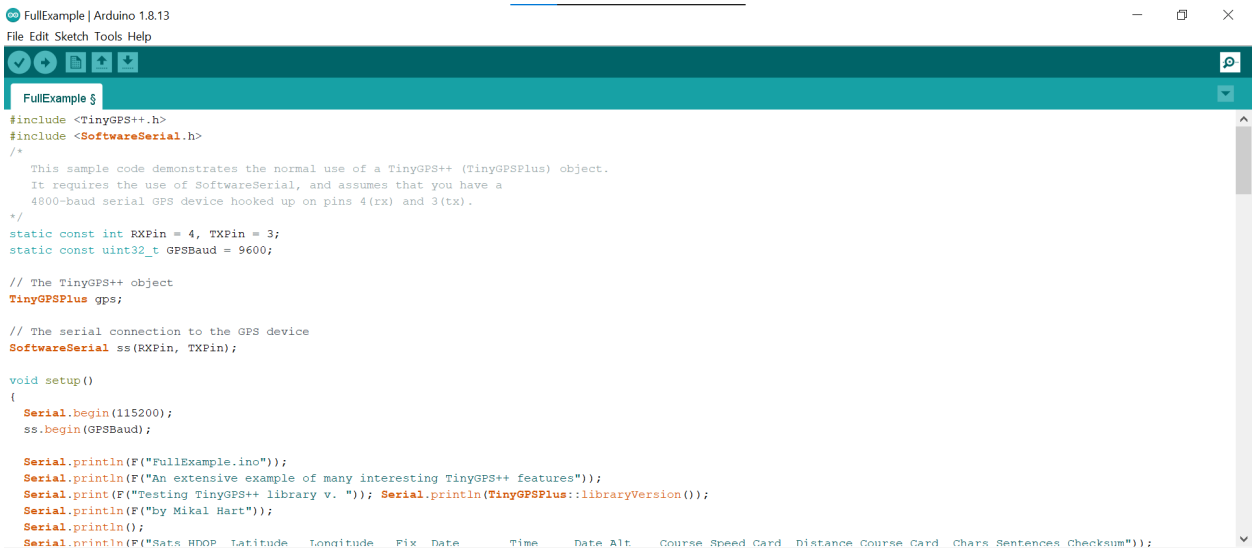
Code for the Voice Subsystem

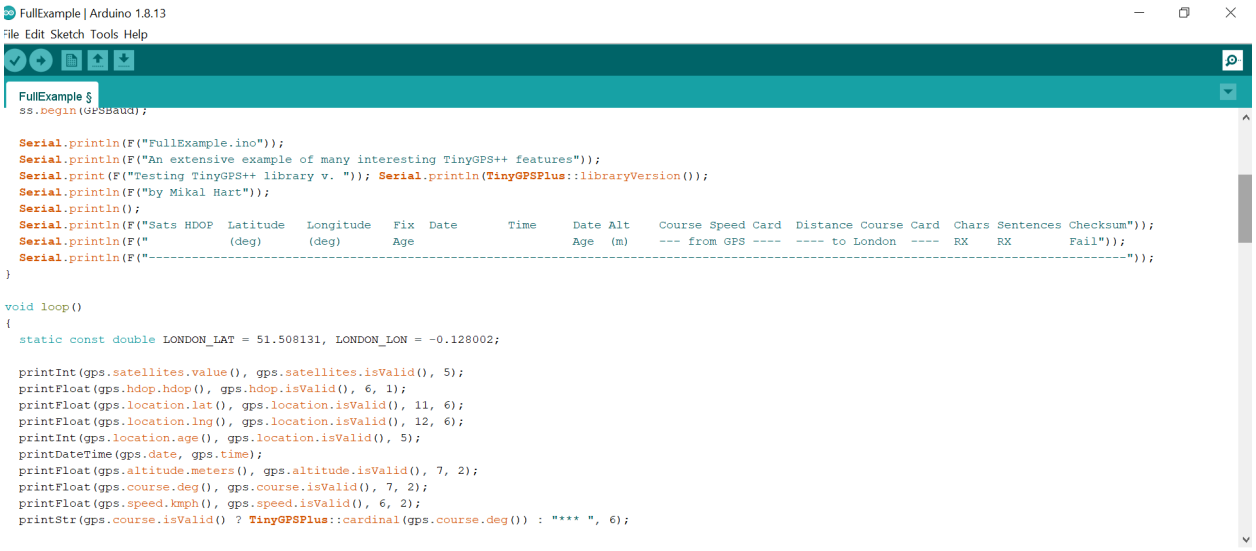
```
#include "Talkie.h"
#include "Vocab_US_Large.h"
#include "Vocab_Special.h"

Talkie voice;

void setup() {
}
void loop() {
    voice.say(spPAUSE2);
    voice.say(sp2_DANGER);
    voice.say(sp2_DANGER);
    voice.say(sp2_MOVE);
    voice.say(sp2_OPERATOR);
    voice.say(sp2_IS);
    voice.say(sp2_ON);
    voice.say(sp2_ALERT);
}
```

Appendices of Prototype III

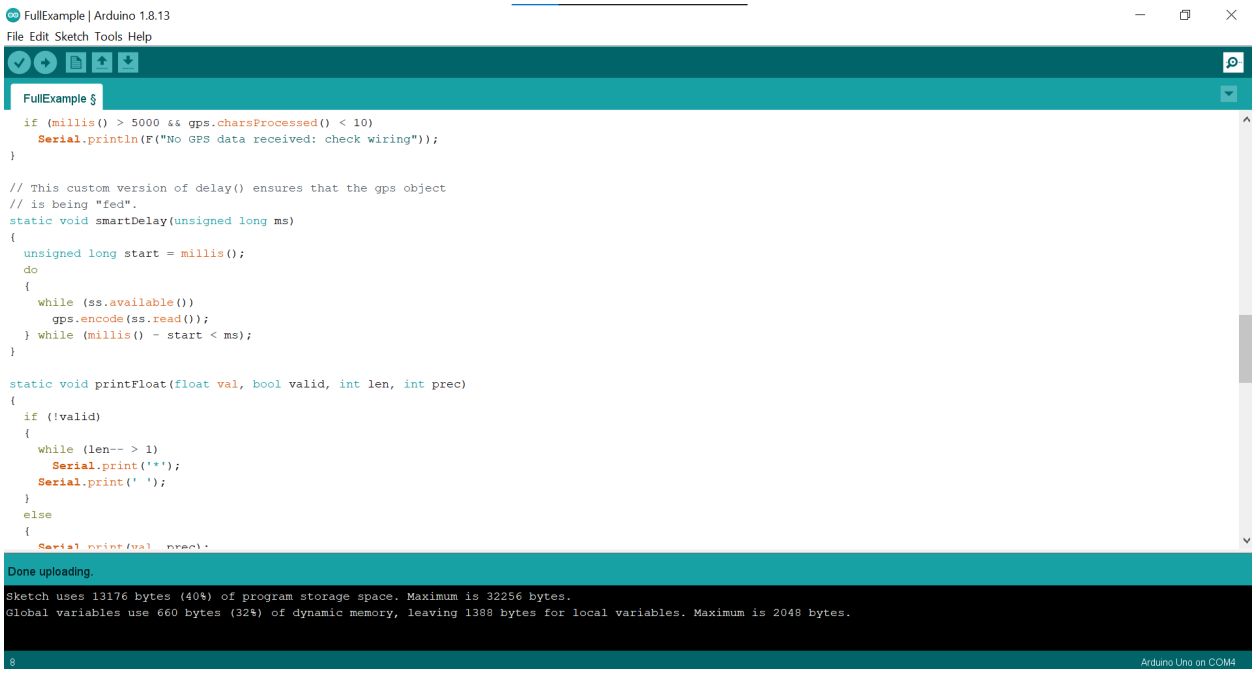
Appendix I

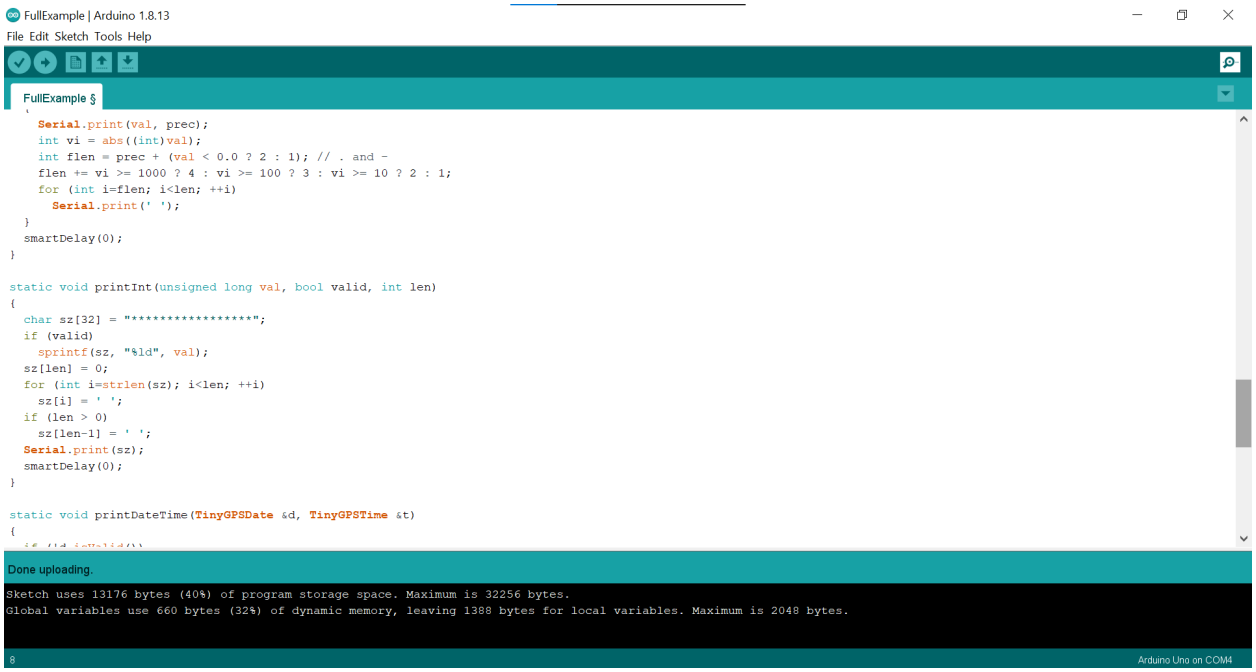
Code for Test 1 of the Location Subsystem

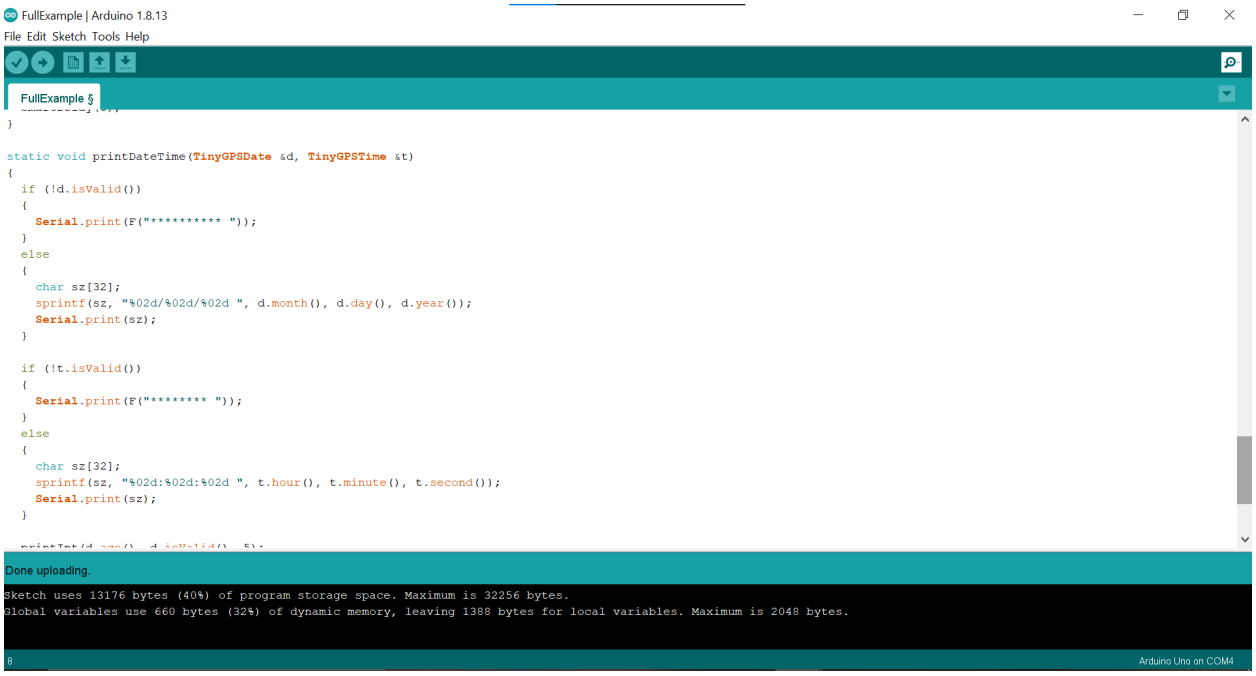

FullExample | Arduino 1.8.13
File Edit Sketch Tools Help
FullExample §
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
/*
This sample code demonstrates the normal use of a TinyGPS++ (TinyGPSPlus) object.
It requires the use of SoftwareSerial, and assumes that you have a
4800-baud serial GPS device hooked up on pins 4(rx) and 3(tx).
*/
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;
// The TinyGPS++ object
TinyGPSPlus gps;
// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);
void setup()
{
Serial.begin(115200);
ss.begin(GPSBaud);
Serial.println(F("FullExample.ino"));
Serial.println(F("An extensive example of many interesting TinyGPS++ features"));
Serial.print(F("Testing TinyGPS++ library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
Serial.println(F("by Mikal Hart"));
Serial.println();


FullExample | Arduino 1.8.13
File Edit Sketch Tools Help
FullExample §
ss.begin(GPSBaud);
Serial.println(F("FullExample.ino"));
Serial.println(F("An extensive example of many interesting TinyGPS++ features"));
Serial.print(F("Testing TinyGPS++ library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
Serial.println(F("by Mikal Hart"));
Serial.println();
Serial.println(F("Sats HDOP Latitude Longitude Fix Date Time Date Alt Course Speed Card Distance Course Card Chars Sentences Checksum"));
Serial.println(F(" (deg) (deg) Age Age (m) --- from GPS ---- ---- to London ---- RX RX Fail"));
Serial.println(F("----------"));
}
void loop()
{
static const double LONDON_LAT = 51.508131, LONDON_LON = -0.128002;
printInt(gps.satellites.value(), gps.satellites.isValid(), 5);
printFloat(gps.hdop.hdop(), gps.hdop.isValid(), 6, 1);
printFloat(gps.location.lat(), gps.location.isValid(), 11, 6);
printFloat(gps.location.lng(), gps.location.isValid(), 12, 6);
printInt(gps.location.age(), gps.location.isValid(), 5);
printDateTime(gps.date, gps.time);
printFloat(gps.altitude.meters(), gps.altitude.isValid(), 7, 2);
printFloat(gps.course.deg(), gps.course.isValid(), 7, 2);
printFloat(gps.speed.kmph(), gps.speed.isValid(), 6, 2);
printStr(gps.course.isValid() ? TinyGPSPlus::cardinal(gps.course.deg()) : "*** ", 6);


FullExample | Arduino 1.8.13
File Edit Sketch Tools Help
FullExample §
unsigned long distanceKmToLondon =
(unsigned long)TinyGPSPlus::distanceBetween(
gps.location.lat(),
gps.location.lng(),
LONDON_LAT,
LONDON_LON) / 1000;
printInt(distanceKmToLondon, gps.location.isValid(), 9);
double courseToLondon =
TinyGPSPlus::courseTo(
gps.location.lat(),
gps.location.lng(),
LONDON_LAT,
LONDON_LON);
printFloat(courseToLondon, gps.location.isValid(), 7, 2);
const char *cardinalToLondon = TinyGPSPlus::cardinal(courseToLondon);
printStr(gps.location.isValid() ? cardinalToLondon : "*** ", 6);
printInt(gps.charsProcessed(), true, 6);
printInt(gps.sentencesWithFix(), true, 10);
printInt(gps.failedChecksum(), true, 9);
Serial.println();
Done uploading.
Sketch uses 13176 bytes (40%) of program storage space. Maximum is 32256 bytes.
Global variables use 660 bytes (32%) of dynamic memory, leaving 1388 bytes for local variables. Maximum is 2048 bytes.
8
Arduino Uno on COM4


FullExample | Arduino 1.8.13
File Edit Sketch Tools Help
FullExample §
if (millis() > 5000 && gps.charsProcessed() < 10)
Serial.println(F("No GPS data received: check wiring"));
}
// This custom version of delay() ensures that the gps object
// is being "fed".
static void smartDelay(unsigned long ms)
{
unsigned long start = millis();
do
{
while (ss.available())
gps.encode(ss.read());
} while (millis() - start < ms);
}
static void printFloat(float val, bool valid, int len, int prec)
{
if (!valid)
{
while (len-- > 1)
Serial.print('*');
Serial.print(' ');
}
else
{
Done uploading.
Sketch uses 13176 bytes (40%) of program storage space. Maximum is 32256 bytes.
Global variables use 660 bytes (32%) of dynamic memory, leaving 1388 bytes for local variables. Maximum is 2048 bytes.
8
Arduino Uno on COM4


FullExample | Arduino 1.8.13
File Edit Sketch Tools Help
FullExample §
Serial.print(val, prec);
int vi = abs((int)val);
int flen = prec + (val < 0.0 ? 2 : 1); // . and -
flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;
for (int i=flen; i<len; ++i)
Serial.print(' ');
}
smartDelay(0);
}
static void printInt(unsigned long val, bool valid, int len)
{
char sz[32] = "******************";
if (valid)
sprintf(sz, "%ld", val);
sz[len] = 0;
for (int i=strlen(sz); i<len; ++i)
sz[i] = ' ';
if (len > 0)
sz[len-1] = ' ';
Serial.print(sz);
smartDelay(0);
}
static void printDateTime(TinyGPSDate &d, TinyGPSTime &t)
{
Done uploading.
Sketch uses 13176 bytes (40%) of program storage space. Maximum is 32256 bytes.
Global variables use 660 bytes (32%) of dynamic memory, leaving 1388 bytes for local variables. Maximum is 2048 bytes.
8
Arduino Uno on COM4


FullExample | Arduino 1.8.13
File Edit Sketch Tools Help
FullExample §
}
static void printDateTime(TinyGPSDate &d, TinyGPSTime &t)
{
if (!d.isValid())
{
Serial.print(F("********** "));
}
else
{
char sz[32];
sprintf(sz, "%02d/%02d/%02d ", d.month(), d.day(), d.year());
Serial.print(sz);
}
if (!t.isValid())
{
Serial.print(F("******** "));
}
else
{
char sz[32];
sprintf(sz, "%02d:%02d:%02d ", t.hour(), t.minute(), t.second());
Serial.print(sz);
}
Done uploading.
Sketch uses 13176 bytes (40%) of program storage space. Maximum is 32256 bytes.
Global variables use 660 bytes (32%) of dynamic memory, leaving 1388 bytes for local variables. Maximum is 2048 bytes.
8
Arduino Uno on COM4

FullExample | Arduino 1.8.13

File Edit Sketch Tools Help

FullExample §

```
    char sz[32];
    sprintf(sz, "%02d/%02d/%02d ", d.month(), d.day(), d.year());
    Serial.print(sz);
  }

  if (!t.isValid())
  {
    Serial.print(F("******** "));
  }
  else
  {
    char sz[32];
    sprintf(sz, "%02d:%02d:%02d ", t.hour(), t.minute(), t.second());
    Serial.print(sz);
  }

  printInt(d.age(), d.isValid(), 5);
  smartDelay(0);
}

static void printStr(const char *str, int len)
{
  int slen = strlen(str);
  for (int i=0; i<len; ++i)
    Serial.print(i<slen ? str[i] : ' ');
  smartDelay(0);
}
```

Done uploading.

```
Sketch uses 13176 bytes (40%) of program storage space. Maximum is 32256 bytes.
Global variables use 660 bytes (32%) of dynamic memory, leaving 1388 bytes for local variables. Maximum is 2048 bytes.
```

8 Arduino Uno on COM4

The code was found in the TinyGPS++ library. This library can be downloaded from here.

Code for Test 2 of the Location Subsystem

DeviceExample | Arduino 1.8.13

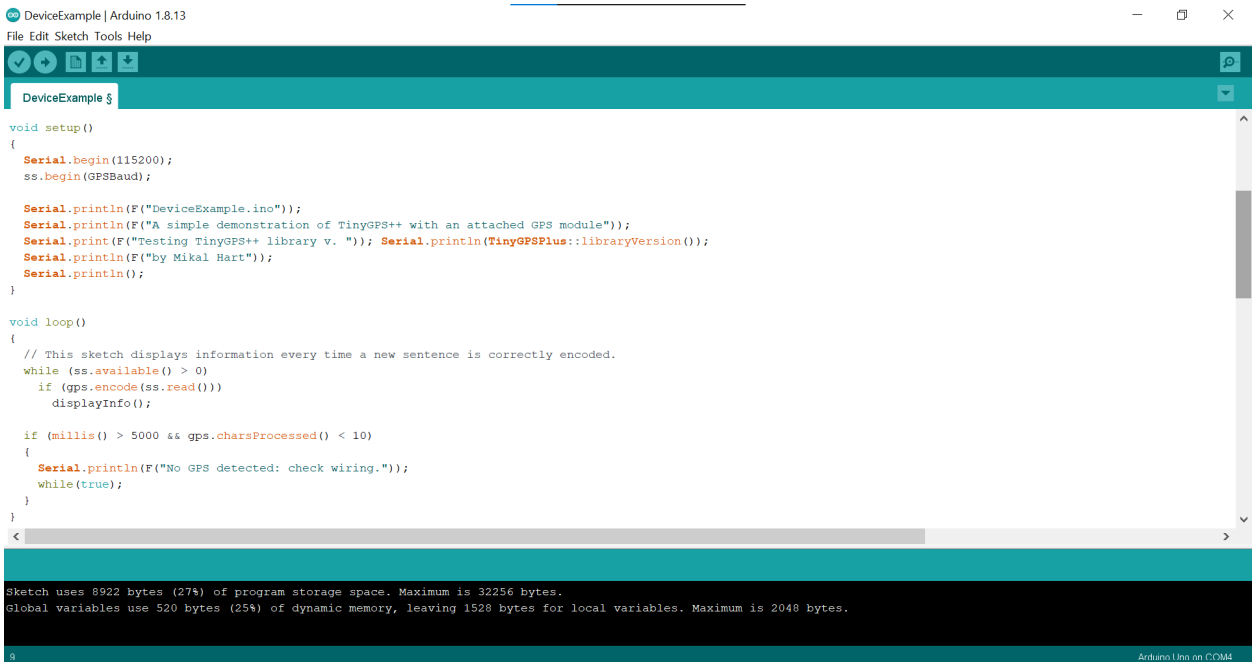
File Edit Sketch Tools Help

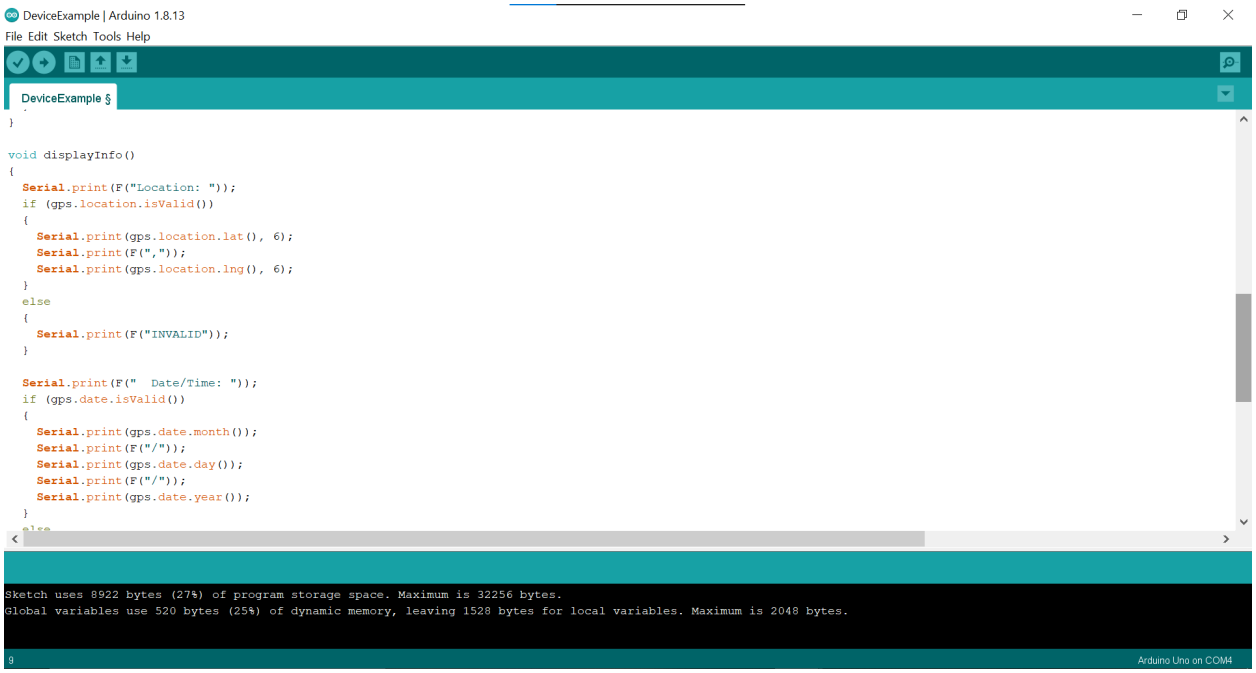
DeviceExample §

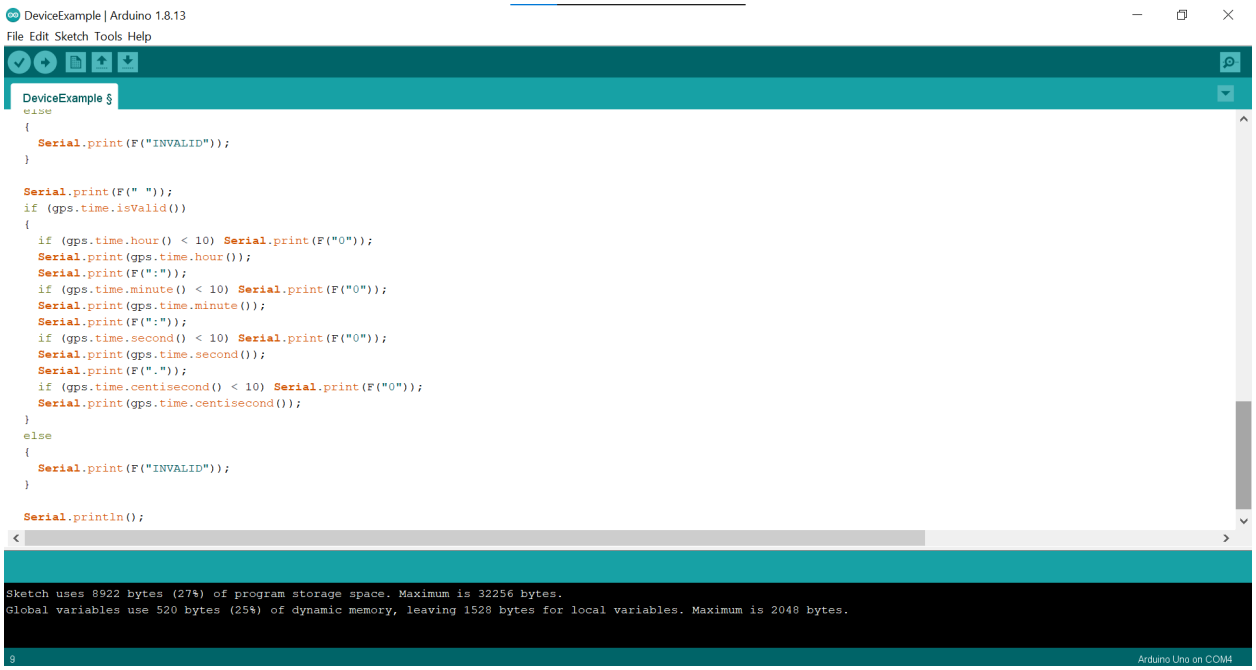
```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
/*
   This sample sketch demonstrates the normal use of a TinyGPS++ (TinyGPSPlu/.s) object.
   It requires the use of SoftwareSerial, and assumes that you have a
   4800-baud serial GPS device hooked up on pins 4(rx) and 3(tx).
*/
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

void setup()
{
  Serial.begin(115200);
  ss.begin(GPSBaud);

  Serial.println(F("DeviceExample.ino"));
  Serial.println(F("A simple demonstration of TinyGPS++ with an attached GPS module"));
  Serial.print(F("Testing TinyGPS++ library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
  Serial.println(F("by Mikal Hart"));
  Serial.println();
```

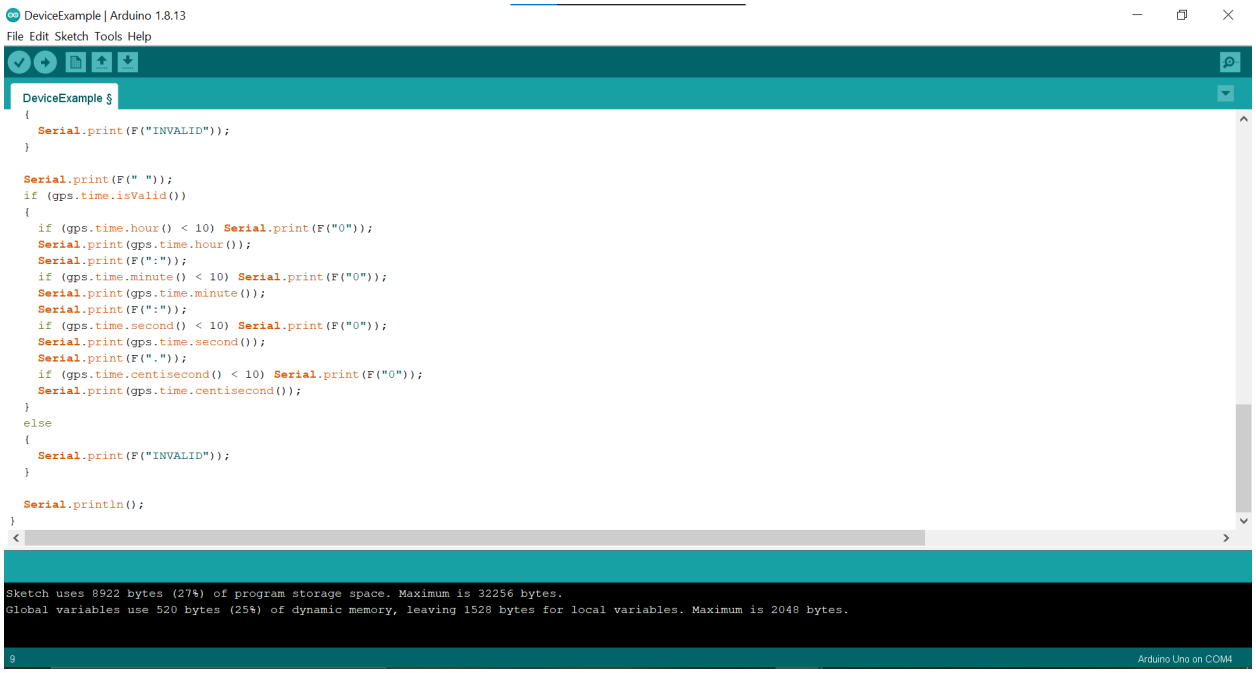
```
Sketch uses 8922 bytes (27%) of program storage space. Maximum is 32256 bytes.
Global variables use 520 bytes (25%) of dynamic memory, leaving 1528 bytes for local variables. Maximum is 2048 bytes.
```

9 Arduino Uno on COM4


DeviceExample | Arduino 1.8.13
File Edit Sketch Tools Help
DeviceExample §
void setup()
{
Serial.begin(115200);
ss.begin(GPSBaud);
Serial.println(F("DeviceExample.ino"));
Serial.println(F("A simple demonstration of TinyGPS++ with an attached GPS module"));
Serial.print(F("Testing TinyGPS++ library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
Serial.println(F("by Mikal Hart"));
Serial.println();
}
void loop()
{
// This sketch displays information every time a new sentence is correctly encoded.
while (ss.available() > 0)
if (gps.encode(ss.read()))
displayInfo();
if (millis() > 5000 && gps.charsProcessed() < 10)
{
Serial.println(F("No GPS detected: check wiring."));
while(true);
}
}
Sketch uses 8922 bytes (27%) of program storage space. Maximum is 32256 bytes.
Global variables use 520 bytes (25%) of dynamic memory, leaving 1528 bytes for local variables. Maximum is 2048 bytes.
Arduino Uno on COM4


DeviceExample | Arduino 1.8.13
File Edit Sketch Tools Help
DeviceExample §
}
void displayInfo()
{
Serial.print(F("Location: "));
if (gps.location.isValid())
{
Serial.print(gps.location.lat(), 6);
Serial.print(F(","));
Serial.print(gps.location.lng(), 6);
}
else
{
Serial.print(F("INVALID"));
}
Serial.print(F(" Date/Time: "));
if (gps.date.isValid())
{
Serial.print(gps.date.month());
Serial.print(F("/"));
Serial.print(gps.date.day());
Serial.print(F("/"));
Serial.print(gps.date.year());
}
Sketch uses 8922 bytes (27%) of program storage space. Maximum is 32256 bytes.
Global variables use 520 bytes (25%) of dynamic memory, leaving 1528 bytes for local variables. Maximum is 2048 bytes.
Arduino Uno on COM4





The code was found in the TinyGPS++ library. This library can be downloaded from here.

Appendix II

Code for Light and Altitude Subsystem

Altitude_and_Light_Subsystem__Correct_ | Arduino 1.8.13

File Edit Sketch Tools Help

Altitude_and_Light_Subsystem__Correct_

```
#include <SFE_BMP180.h>
#include <Wire.h>


int led1 = 2;
int led2 = 3;
int led3 = 4;
int led4 = 5;
int led5 = 6;
int led6 = 7;
int led7 = 8;
int led8 = 9;


int intvalue;

SFE_BMP180 pressure;

double baseline;

void setup() {

Serial.begin(9600);
  Serial.println("REBOOT");

  // Initialize the sensor (it is important to get calibration values stored on the device).
```

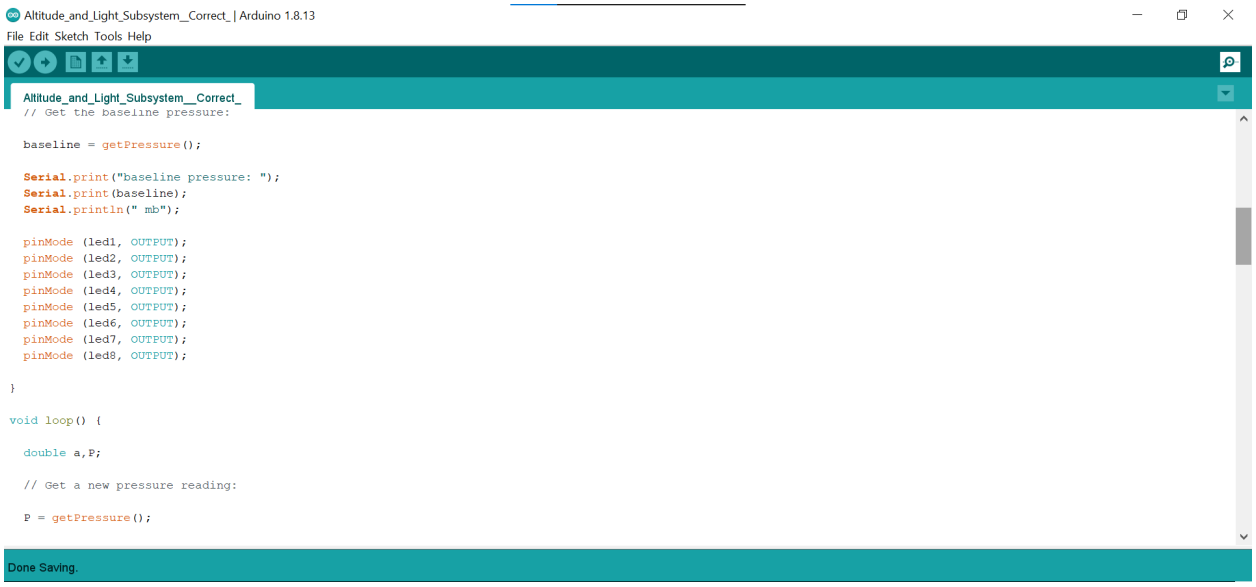
Altitude_and_Light_Subsystem__Correct_ | Arduino 1.8.13

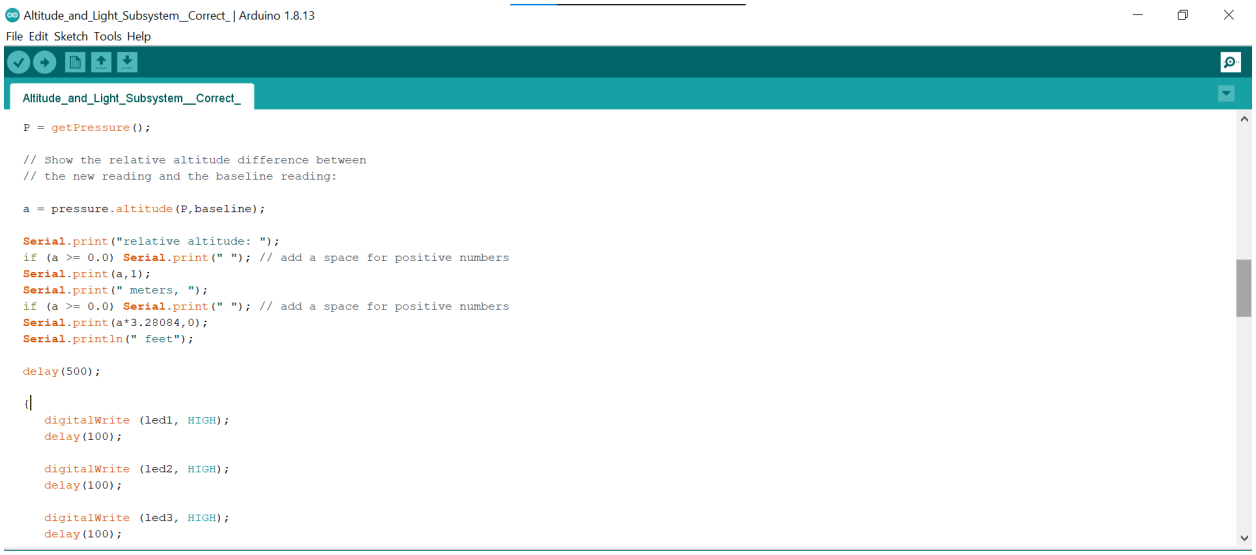
File Edit Sketch Tools Help

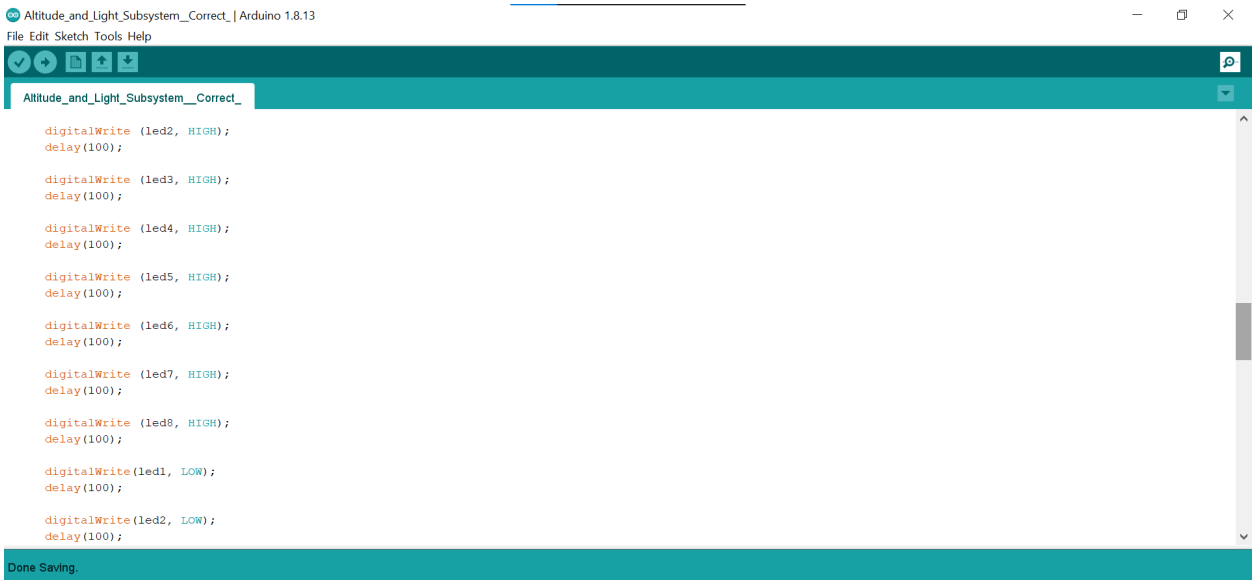
Altitude_and_Light_Subsystem__Correct_

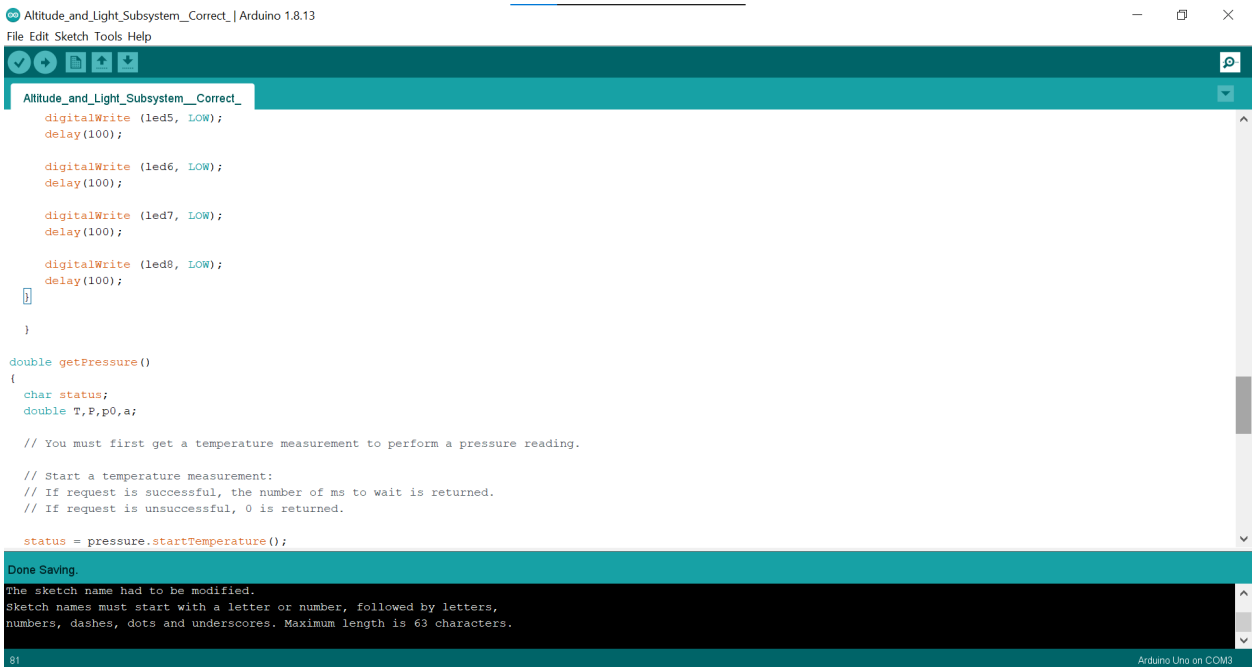
```
void setup() {

Serial.begin(9600);
  Serial.println("REBOOT");

  // Initialize the sensor (it is important to get calibration values stored on the device).

  if (pressure.begin())
    Serial.println("BMP180 init success");
  else
  {
    // Oops, something went wrong, this is usually a connection problem,
    // see the comments at the top of this sketch for the proper connections.

    Serial.println("BMP180 init fail (disconnected?)\n\n");
    while(1); // Pause forever.
  }

  // Get the baseline pressure:

  baseline = getPressure();

  Serial.print("baseline pressure: ");
  Serial.print(baseline);
  Serial.println(" mb");
```

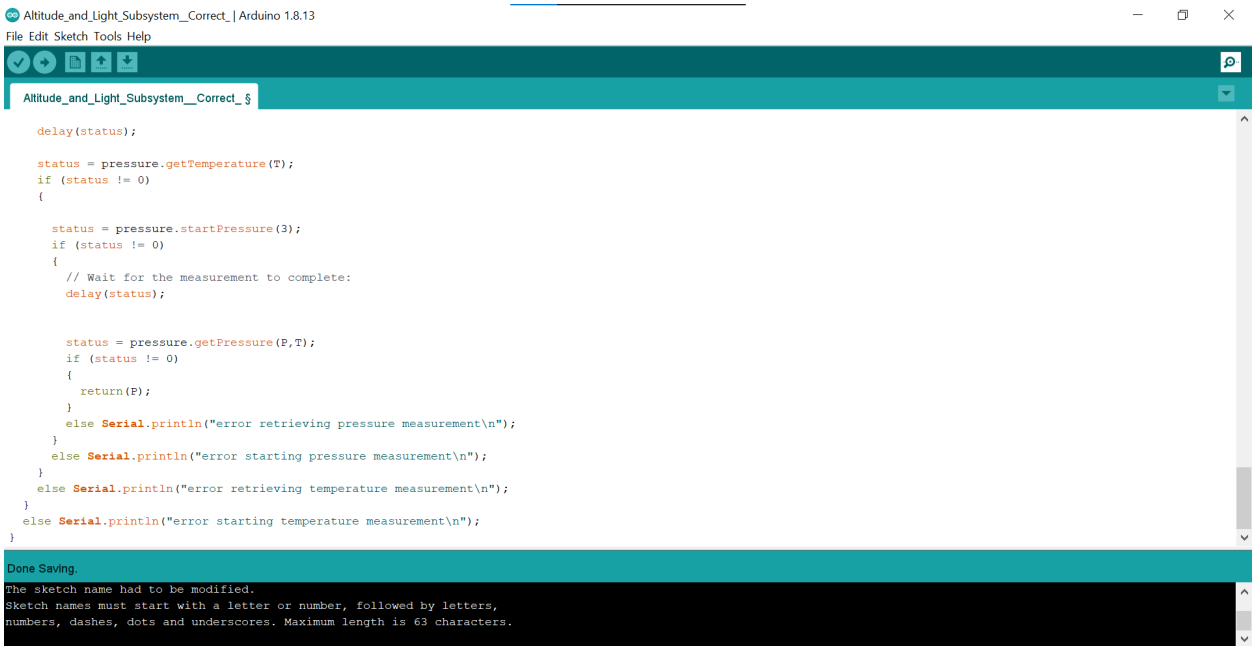
Done Saving.


Altitude_and_Light_Subsystem__Correct_ | Arduino 1.8.13
File Edit Sketch Tools Help
Altitude_and_Light_Subsystem__Correct_
// Get the baseline pressure:
baseline = getPressure();
Serial.print("baseline pressure: ");
Serial.print(baseline);
Serial.println(" mb");
pinMode (led1, OUTPUT);
pinMode (led2, OUTPUT);
pinMode (led3, OUTPUT);
pinMode (led4, OUTPUT);
pinMode (led5, OUTPUT);
pinMode (led6, OUTPUT);
pinMode (led7, OUTPUT);
pinMode (led8, OUTPUT);
}
void loop() {
double a,P;
// Get a new pressure reading:
P = getPressure();
Done Saving.


Altitude_and_Light_Subsystem__Correct_ | Arduino 1.8.13
File Edit Sketch Tools Help
Altitude_and_Light_Subsystem__Correct_
P = getPressure();
// Show the relative altitude difference between
// the new reading and the baseline reading:
a = pressure.altitude(P,baseline);
Serial.print("relative altitude: ");
if (a >= 0.0) Serial.print(" "); // add a space for positive numbers
Serial.print(a,1);
Serial.print(" meters, ");
if (a >= 0.0) Serial.print(" "); // add a space for positive numbers
Serial.print(a*3.28084,0);
Serial.println(" feet");
delay(500);
{
digitalWrite (led1, HIGH);
delay(100);
digitalWrite (led2, HIGH);
delay(100);
digitalWrite (led3, HIGH);
delay(100);


Altitude_and_Light_Subsystem__Correct_ | Arduino 1.8.13
File Edit Sketch Tools Help
Altitude_and_Light_Subsystem__Correct_
digitalWrite (led2, HIGH);
delay(100);
digitalWrite (led3, HIGH);
delay(100);
digitalWrite (led4, HIGH);
delay(100);
digitalWrite (led5, HIGH);
delay(100);
digitalWrite (led6, HIGH);
delay(100);
digitalWrite (led7, HIGH);
delay(100);
digitalWrite (led8, HIGH);
delay(100);
digitalWrite(led1, LOW);
delay(100);
digitalWrite(led2, LOW);
delay(100);
Done Saving.


Altitude_and_Light_Subsystem__Correct_ | Arduino 1.8.13
File Edit Sketch Tools Help
Altitude_and_Light_Subsystem__Correct_
digitalWrite (led5, LOW);
delay(100);
digitalWrite (led6, LOW);
delay(100);
digitalWrite (led7, LOW);
delay(100);
digitalWrite (led8, LOW);
delay(100);
}
}
double getPressure()
{
char status;
double T,P,p0,a;
// You must first get a temperature measurement to perform a pressure reading.
// Start a temperature measurement:
// If request is successful, the number of ms to wait is returned.
// If request is unsuccessful, 0 is returned.
status = pressure.startTemperature();
Done Saving.
The sketch name had to be modified.
Sketch names must start with a letter or number, followed by letters,
numbers, dashes, dots and underscores. Maximum length is 63 characters.
81
Arduino Uno on COM3



There is no source for this code because it was adjusted based on the codes for the altitude and light subsystem.

Code for Voice and Location Subsystem

Location_and_Voice_Subsystem | Arduino 1.8.13

File Edit Sketch Tools Help

Location_and_Voice_Subsystem

```
#include "Talkie.h"
#include "Vocab_US_Large.h"
#include "Vocab_Special.h"
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

Talkie voice;

static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;


TinyGPSPlus gps;

SoftwareSerial ss(RXPin, TXPin);

void setup() {
  Serial.begin(115200);
  ss.begin(GPSBaud);

  Serial.println(F("DeviceExample.ino"));
  Serial.println(F("A simple demonstration of TinyGPS++ with an attached GPS module"));
  Serial.print(F("Testing TinyGPS++ library v. ")); Serial.println(TinyGPSPlus::libraryVersion());
  Serial.println(F("by Mikal Hart"));
  Serial.println();
```

Location_and_Voice_Subsystem | Arduino 1.8.13

File Edit Sketch Tools Help

Location_and_Voice_Subsystem

```
}

void loop() {
   {
    voice.say(spPAUSE2);
    voice.say(sp2_DANGER);
    voice.say(sp2_DANGER);
    voice.say(sp2_MOVE);
    voice.say(sp2_OPERATOR);
    voice.say(sp2_IS);
    voice.say(sp2_ON);
    voice.say(sp2_ALERT);
   }

{
  while (ss.available() > 0)
    if (gps.encode(ss.read()))
      displayInfo();

  if (millis() > 5000 && gps.charsProcessed() < 10)
  {
    Serial.println(F("No GPS detected: check wiring."));
    while(true);
  }
}
```

Location_and_Voice_Subsystem | Arduino 1.8.13

File Edit Sketch Tools Help

Location_and_Voice_Subsystem

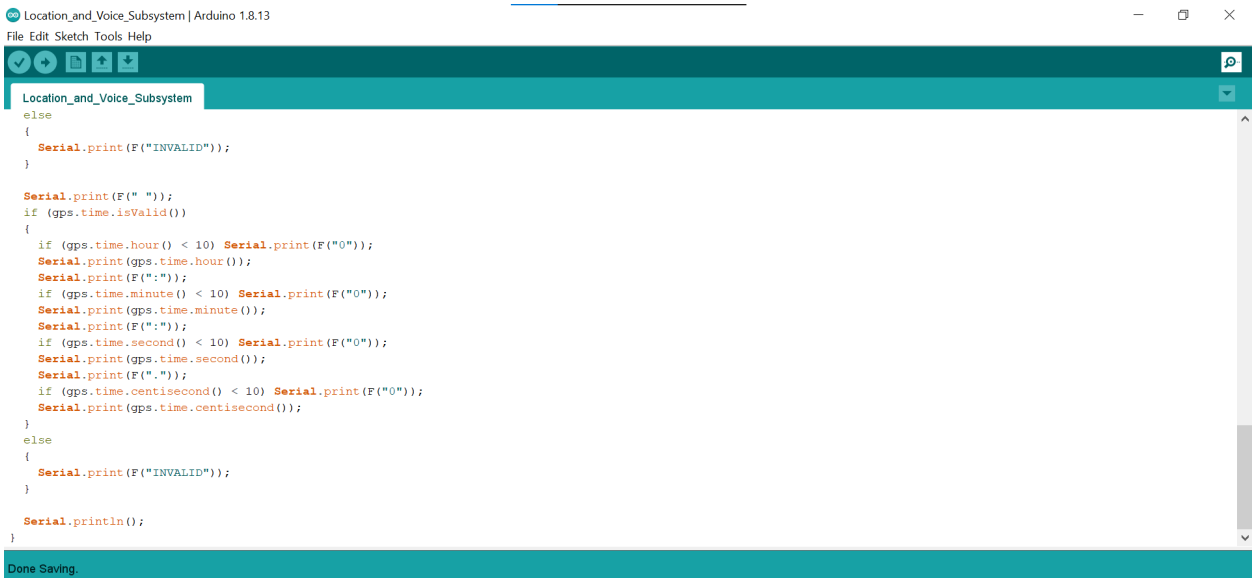
```
void displayInfo()
{
  Serial.print(F("Location: "));
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 6);
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.print(F("  Date/Time: "));
  if (gps.date.isValid())
  {
    Serial.print(gps.date.month());
    Serial.print(F("/"));
    Serial.print(gps.date.day());
    Serial.print(F("/"));
    Serial.print(gps.date.year());
  }
  else
  {
    Serial.print(F("INVALID"));
```

Location_and_Voice_Subsystem | Arduino 1.8.13

File Edit Sketch Tools Help

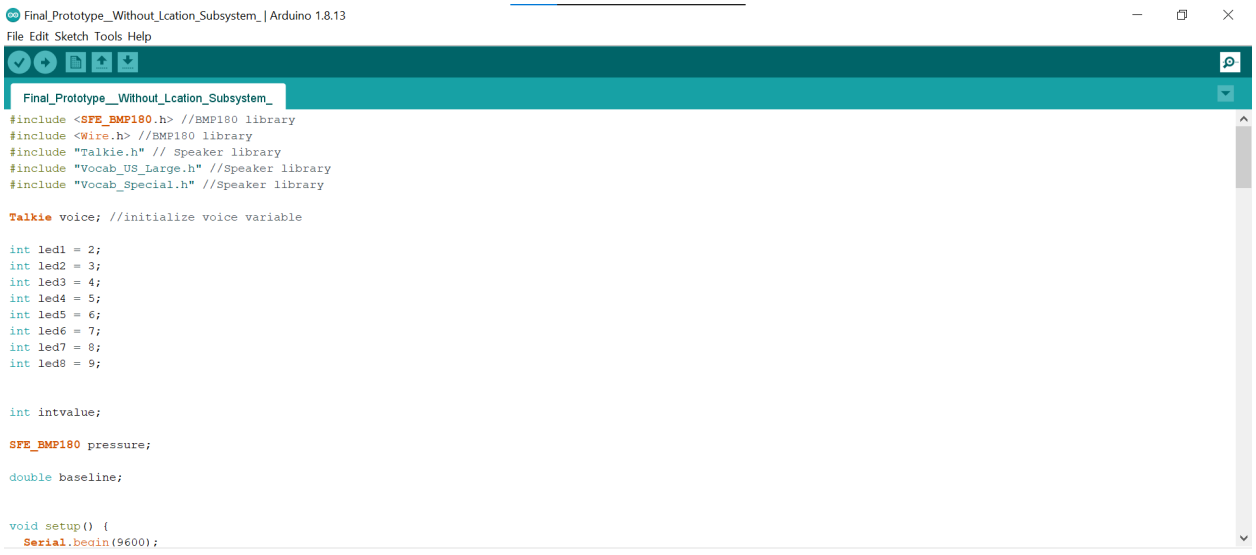
Location_and_Voice_Subsystem

```
    Serial.print(gps.date.year());
  }
  else
  {
    Serial.print(F("INVALID"));
  }

  Serial.print(F(" "));
  if (gps.time.isValid())
  {
    if (gps.time.hour() < 10) Serial.print(F("0"));
    Serial.print(gps.time.hour());
    Serial.print(F(":"));
    if (gps.time.minute() < 10) Serial.print(F("0"));
    Serial.print(gps.time.minute());
    Serial.print(F(":"));
    if (gps.time.second() < 10) Serial.print(F("0"));
    Serial.print(gps.time.second());
    Serial.print(F("."));
    if (gps.time.centisecond() < 10) Serial.print(F("0"));
    Serial.print(gps.time.centisecond());
  }
  else
  {
    Serial.print(F("INVALID"));
  }
```



There is no source for this code because it was adjusted based on the codes for the altitude and light subsystem.

Code for Final Prototype



Final_Prototype__Without_Lcation_Subsystem_ | Arduino 1.8.13

File Edit Sketch Tools Help

Final_Prototype__Without_Lcation_Subsystem_

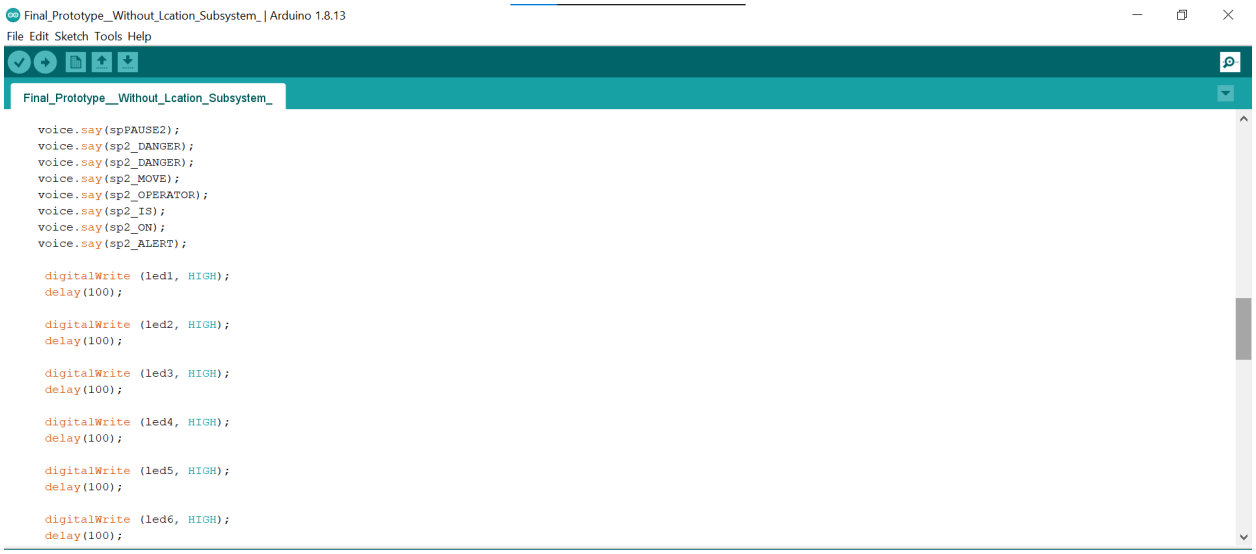
```
void setup() {
  Serial.begin(9600);
  Serial.println("REBOOT");

  if (pressure.begin())
    Serial.println("BMP180 init success");
  else
  {
    Serial.println("BMP180 init fail (disconnected?)\n\n");
    while(1);
  }

  baseline = getPressure();

  Serial.print("baseline pressure: ");
  Serial.print(baseline);
  Serial.println(" mb");

  {

  pinMode (led1, OUTPUT);
  pinMode (led2, OUTPUT);
  pinMode (led3, OUTPUT);
  pinMode (led4, OUTPUT);
  pinMode (led5, OUTPUT);
```

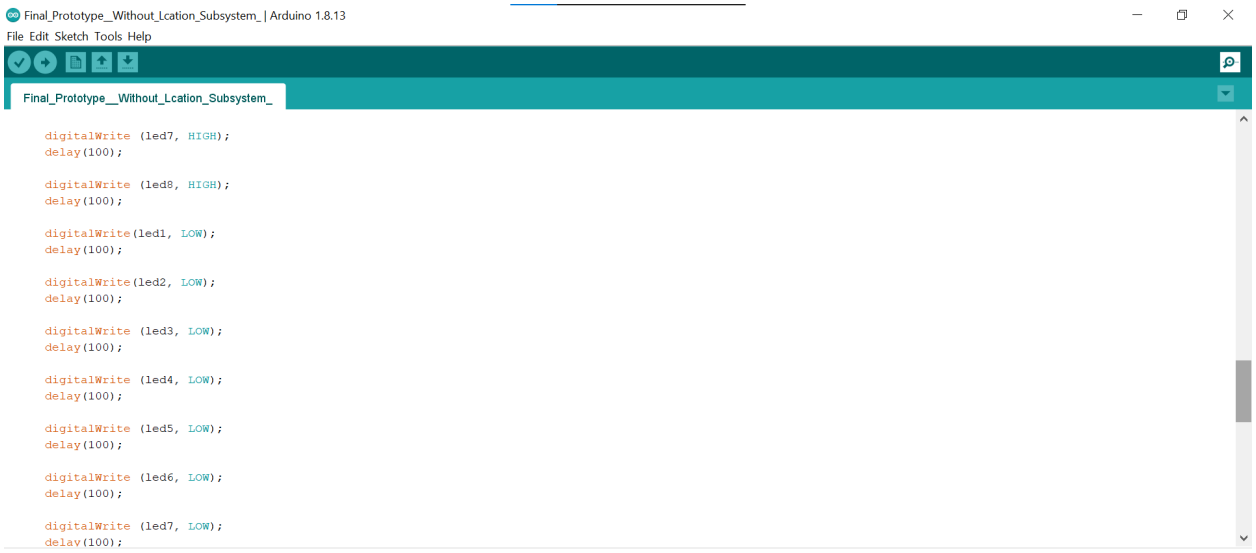
Final_Prototype__Without_Lcation_Subsystem_ | Arduino 1.8.13

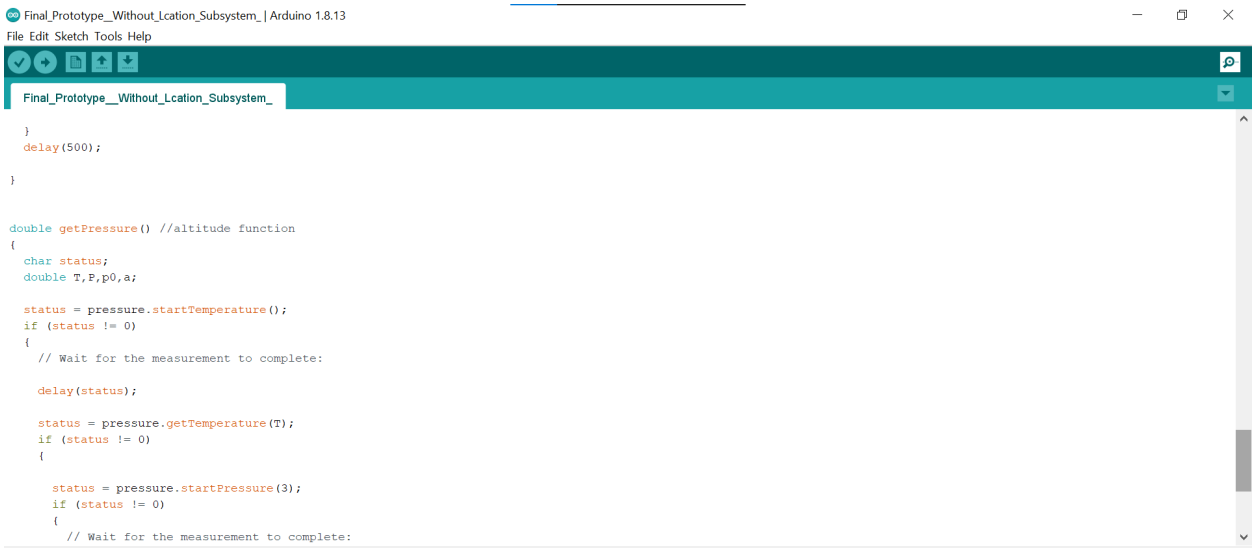
File Edit Sketch Tools Help

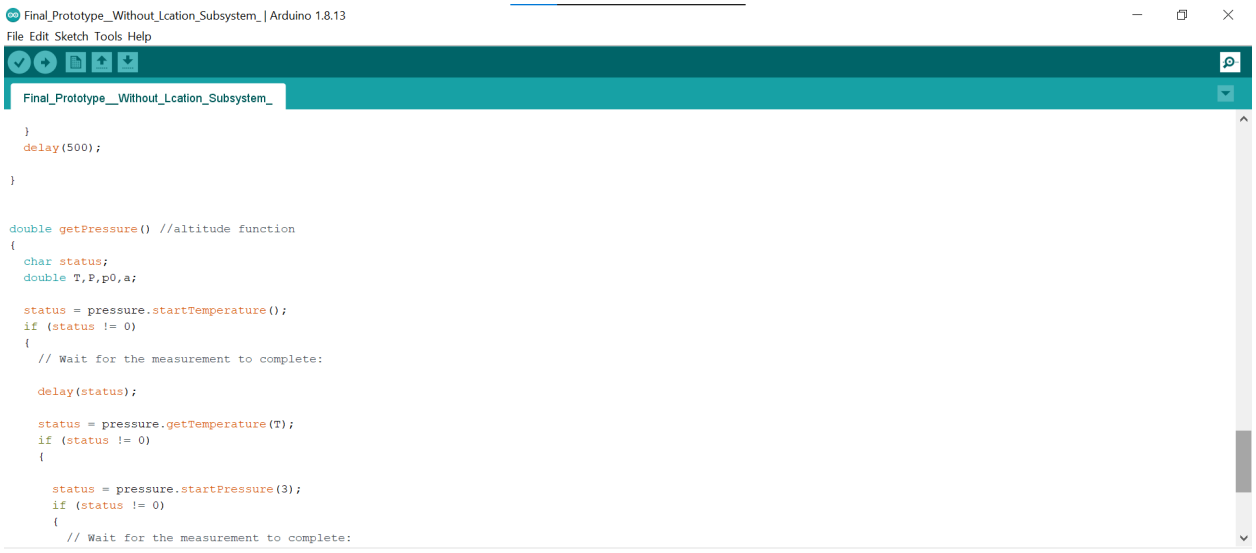
Final_Prototype__Without_Lcation_Subsystem_

```
  pinMode (led4, OUTPUT);
  pinMode (led5, OUTPUT);
  pinMode (led6, OUTPUT);
  pinMode (led7, OUTPUT);
  pinMode (led8, OUTPUT);

  }


}

void loop() {

  double a,P;

  P = getPressure();

  a = pressure.altitude(P,baseline);

  Serial.print("relative altitude: ");
  if (a >= 0.0) Serial.print(" ");
  Serial.print(a,1);
  Serial.print(" meters, \n");
  if (a <= 0.0) //if altitude reading
  {
    Serial.print("The drone has fallen.\n");
```


Final_Prototype__Without_Lcation_Subsystem_ | Arduino 1.8.13
File Edit Sketch Tools Help
Final_Prototype__Without_Lcation_Subsystem_
voice.say(spPAUSE2);
voice.say(sp2_DANGER);
voice.say(sp2_DANGER);
voice.say(sp2_MOVE);
voice.say(sp2_OPERATOR);
voice.say(sp2_IS);
voice.say(sp2_ON);
voice.say(sp2_ALERT);
digitalWrite (led1, HIGH);
delay(100);
digitalWrite (led2, HIGH);
delay(100);
digitalWrite (led3, HIGH);
delay(100);
digitalWrite (led4, HIGH);
delay(100);
digitalWrite (led5, HIGH);
delay(100);
digitalWrite (led6, HIGH);
delay(100);


Final_Prototype__Without_Lcation_Subsystem_ | Arduino 1.8.13
File Edit Sketch Tools Help
Final_Prototype__Without_Lcation_Subsystem_
digitalWrite (led7, HIGH);
delay(100);
digitalWrite (led8, HIGH);
delay(100);
digitalWrite(led1, LOW);
delay(100);
digitalWrite(led2, LOW);
delay(100);
digitalWrite (led3, LOW);
delay(100);
digitalWrite (led4, LOW);
delay(100);
digitalWrite (led5, LOW);
delay(100);
digitalWrite (led6, LOW);
delay(100);
digitalWrite (led7, LOW);
delay(100);


Final_Prototype__Without_Lcation_Subsystem_ | Arduino 1.8.13
File Edit Sketch Tools Help
Final_Prototype__Without_Lcation_Subsystem_
}
delay(500);
}
double getPressure() //altitude function
{
char status;
double T,P,p0,a;
status = pressure.startTemperature();
if (status != 0)
{
// Wait for the measurement to complete:
delay(status);
status = pressure.getTemperature(T);
if (status != 0)
{
status = pressure.startPressure(3);
if (status != 0)
{
// Wait for the measurement to complete:


Final_Prototype__Without_Lcation_Subsystem_ | Arduino 1.8.13
File Edit Sketch Tools Help
Final_Prototype__Without_Lcation_Subsystem_
}
delay(500);
}
double getPressure() //altitude function
{
char status;
double T,P,p0,a;
status = pressure.startTemperature();
if (status != 0)
{
// Wait for the measurement to complete:
delay(status);
status = pressure.getTemperature(T);
if (status != 0)
{
status = pressure.startPressure(3);
if (status != 0)
{
// Wait for the measurement to complete:

There is no source for this code because it was adjusted based on the codes for the altitude and light subsystem.