the LEGO car tracker

In this chapter, you'll build a LEGO Car Tracker that records the coordinates and velocity of a moving vehicle. You could use this tracker to keep a log of your drives around town or long-distance road trips. Or perhaps you'd like to record a favorite bicycling route. You could even place the tracker in a vehicle you'd like to have under surveillance to see where it travels and how long it stops in various locations.

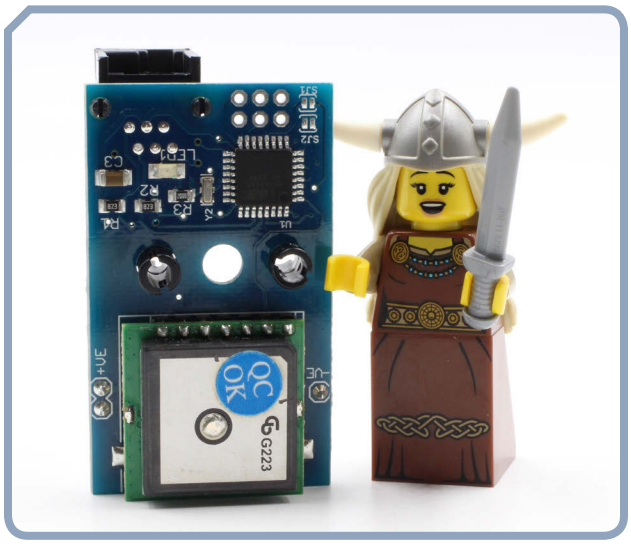


FIGURE 1: The antenna on the dGPS is the square component at the bottom of the circuit board.

The LEGO Car Tracker uses the *Global Positioning System (GPS)* to measure your precise geographical location and time by detecting radio signals from satellites orbiting the earth. Based on these signals, the receiver can determine its distance from each transmitter satellite. If the receiver can calculate its distance from at least four satellites, it can figure out its exact location. Let's use this information to follow a car!

Dexter Industries (*https://www.dexterindustries.com/*) has a GPS sensor, along with a programming block, that you can use with MINDSTORMS EV3 called the *dGPS*. Figure 1 shows the dGPS.

The dGPS sensor plugs into the EV3 Intelligent Brick, allowing you to use its latitude, longitude, altitude, and time data.

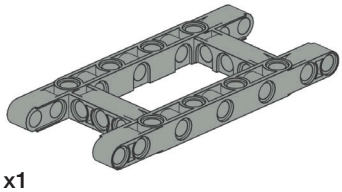
what you'll need

In addition to the dGPS sensor from Dexter Industries, the LEGO parts shown in Figure 2 are used to build the LEGO Car Tracker. You can find all these LEGO parts in the MINDSTORMS EV3 #31313 set.

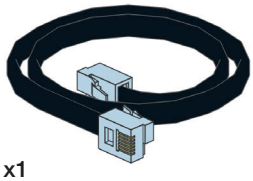
You may also need double-sided tape to attach the LEGO Car Tracker to the car.

FIGURE 2: Parts used to build the LEGO Car Tracker

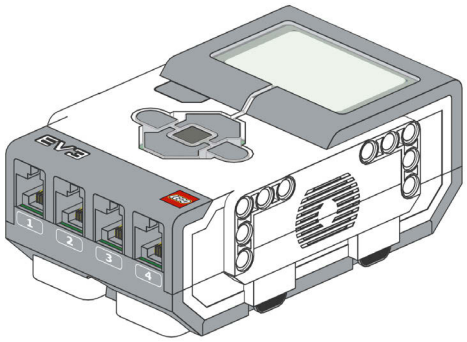
x4
2780
Technic Pin with Friction Ridges



x1
64178
5 x 11 Open Center Frame Liftarm



x1
55805
Connector Cable



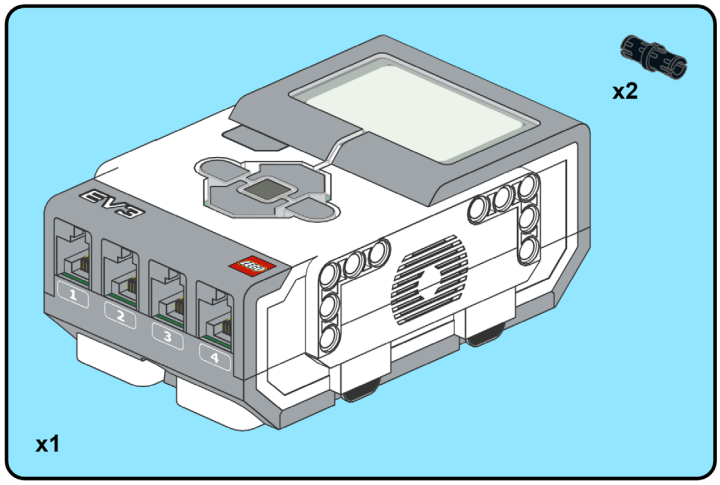
x1
95646
EV3 Intelligent Brick

building the LEGO car tracker

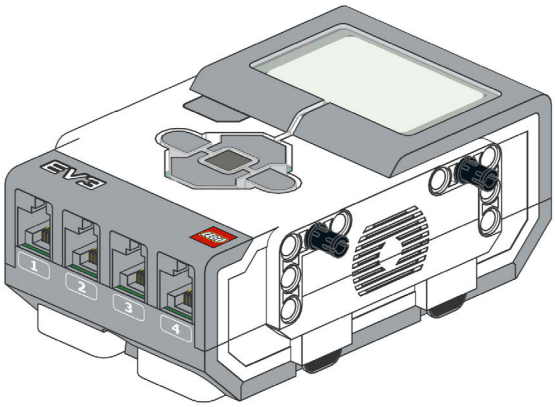
To build the LEGO Car Tracker, you'll mount the dGPS on the EV3 Intelligent Brick, as shown in Figure 3, using an open center frame liftarm. You can do this in several ways, so long as you keep the square GPS receiver pointing toward the sky.

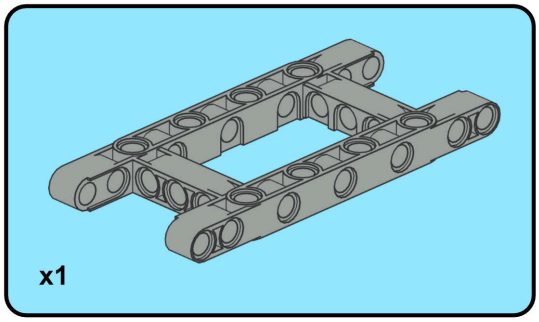
Follow these instructions for one possible build.

FIGURE 3: Attach the dGPS to the EV3 Intelligent Brick with an open center frame liftarm.

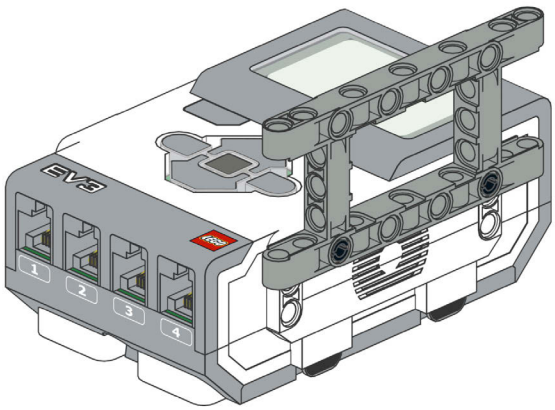


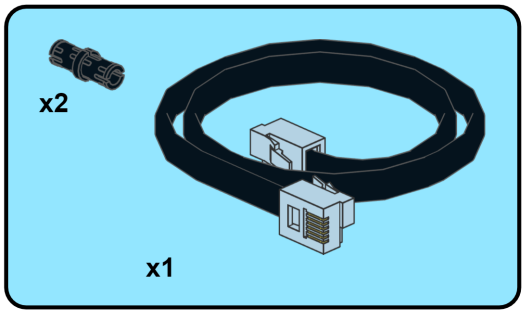
1



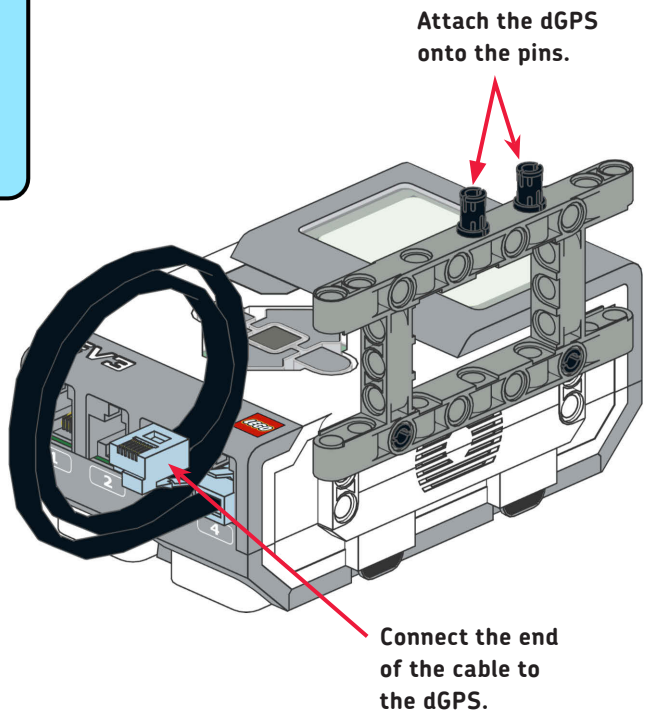


2





3



writing the software

You'll program the tracker to continuously sample four measurements from the dGPS sensor: the current time, latitude, longitude, and velocity. The program will log these four measurements into a file called *navigation log* and show them on the EV3 Intelligent Brick's front panel display every 5 seconds.

Before you program the dGPS, you need to download the programming block made by Dexter Industries from *https://github.com/DexterInd/EV3_Dexter_Industries_Sensors/*. Download the file called *Dexter.ev3b* to your desktop computer. Then import the block into the EV3 programming environment by choosing **Tools ▸ Block Import** from the toolbar at the top of the screen. Several new programming blocks should appear in the Sensor tab. These blocks match the sensors built by Dexter Industries, including the dGPS sensor, so that you can use the dGPS with all the other programming blocks in the MINDSTORMS EV3 programming environment.

Figure 4 shows the program to run the LEGO Car Tracker.

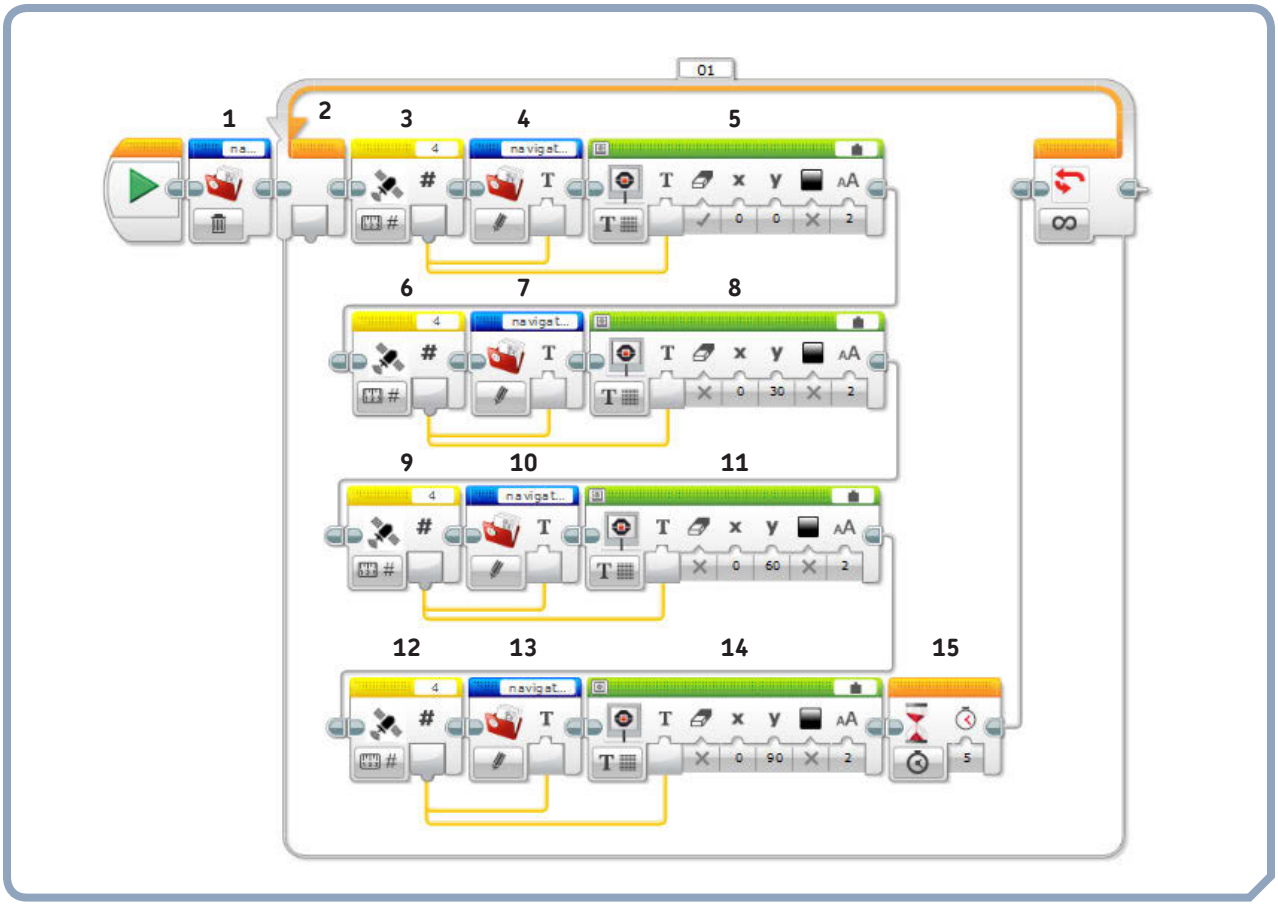


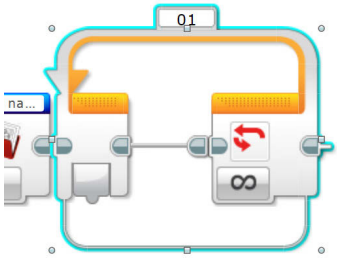
FIGURE 4: The software for the LEGO Car Tracker samples data inside a continuous loop.

To create the program, assemble the following blocks:

1. **File Access block** This clears the memory on the EV3 Intelligent Brick of any file named *navigation log* so that you'll start with a fresh data log each time you run the program.

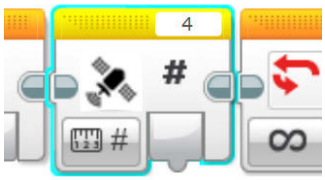
Select a File Access block from the Advanced tab and attach it after the program Start. Set the **File Access** menu to **Delete** and set **File Name** to *navigation log*.

2. **Loop block** This takes sensor readings infinitely.



Attach the Loop block after the File Access block.

3. **dGPS block** This block takes a time measurement from the dGPS sensor.

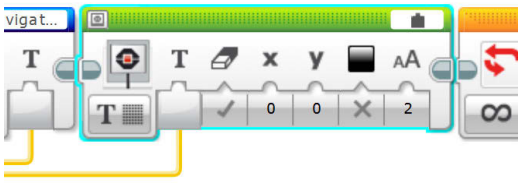


Select the dGPS block from the Sensor tab at the bottom of the screen and place it within the loop. Set **dGPS** to **Read UTC Time** and set **Port** to **4**.

4. **File Access block** This writes the time value taken from the dGPS to the file called *navigation log*.

Place the File Access block after the dGPS block inside the loop and make a wire connection between the two. This will pass data from the dGPS sensor to the file. On the File Access block, set **File Access** to **Write** and **File Name** to *navigation log*.

5. **Display block** This shows the measured time on the EV3 Intelligent Brick display.



Still within the loop, place the Display block after the File Access block and run a wire connection from the dGPS sensor to the Display block's input. Set **Display** to **Text–Pixels**. Then set **Text** to **Wired**, **Clear Screen** to **True**, **x** to **0**, **y** to **0**, **Color** to **False**, and **Font** to **2**. These display settings indicate that the number to display will be wired from another block and that it will display beginning at the horizontal x-position 0 and vertical y-position 0. Since this is the first measurement from a series of measurements from the dGPS, we use **Clear Screen** to clear out any previous measurements from the data display. Run a wire between the **Text** input of the Display block and the output of the dGPS block.

In order to retrieve, log, and display the other measurements, you'll follow the same basic process you used to log and display the current time. In Figure 4, each sequence of blocks forms a separate row, but you can string the blocks together in a straight line if you prefer.

6. **dGPS block** This takes a latitude measurement from the dGPS sensor. Set **dGPS** to **Read Latitude**, and **Port** to **4**.

7. **File Access block** This writes the latitude measurement to the *navigation log* file. Create a wire connection between this block and the dGPS block. Set **File Access** to **Write** and **File Name** to *navigation log*.

8. **Display block** This displays the latitude measurement on the EV3 Intelligent Brick. Create a wire connection between this block's input and the dGPS block. Set **Display** to **Text–Pixels** and **Text** to **Wired**. Then set **Clear Screen** to **False**, **x** to **0**, **y** to **30**, **Color** to **False**, and **Font** to **2**. Since this is the second in a series of four measurements from the dGPS, we display it lower on the screen, at a y setting of 30.

9. **dGPS block** This takes a longitude measurement from the dGPS sensor. Set **dGPS** to **Read Longitude** and **Port** to **4**.

10. **File Access block** This saves the longitude measurement in the *navigation log* file. Make a wire connection between this block and the dGPS block. Set **File Access** to **Write** and **File Name** to *navigation log*.

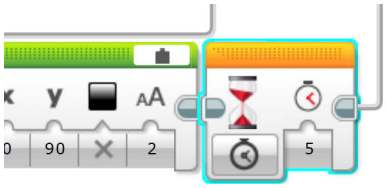
11. **Display block** This displays the longitude measurement. Make a wire connection between this block and the dGPS block. Set **Display** to **Text–Pixels**, **Text** to **Wired**, **Clear Screen** to **False**, **x** to **0**, **y** to **60**, **Color** to **False**, and **Font** to **2**.

12. **dGPS block** This takes a velocity measurement from the dGPS sensor. Set **dGPS** to **Read Velocity** and **Port** to **4**. This measurement will tell you how fast the car is going.

13. **File Access block** This writes the velocity measurement to the *navigation log* file. Make a wire connection between this block and the dGPS block. Set **File Access** to **Write** and **File Name** to *navigation log*.

14. **Display block** This displays the velocity measurement. Make a wire connection between this block and the dGPS block. Set **Display** to **Text–Pixels**, **Text** to **Wired**, **Clear Screen** to **False**, **x** to **0**, **y** to **90**, **Color** to **False**, and **Font** to **2**.

15. **Wait block** This pauses for 5 seconds before repeating the procedure.



Place the Wait block after the last Display block. Set **Wait** to **Time** and **Seconds** to **5**.

using the LEGO car tracker

Once you activate the LEGO Car Tracker, it should record the current time, latitude, longitude, and velocity every 5 seconds. These numbers should update on the EV3 Intelligent Brick's front panel display. The LEGO Car Tracker should also record the measurements in the *navigation log* file stored in the EV3 Intelligent Brick's memory.

Activate your LEGO Car Tracker by running the program in Figure 4 and attaching the device to the dashboard of your car with double-sided tape. If you want to be stealthy, you might try attaching it instead to the area behind the headrest of the backseat or on the floor behind the driver's seat. It's best to position the tracker so it has a view of the sky, such as near a window. Then track your car as you drive around town.

You'll know the dGPS sensor is working properly if the blue LED lights up on the device's circuit board, as shown in Figure 3. This blue light indicates that the signals from several satellites are strong and clear. If you live in large city, you may find that the GPS loses signal around tall buildings.

viewing the navigation log file

When you're done tracking, download the navigation log by plugging the EV3 Intelligent Brick into your computer, opening the MINDSTORMS programming environment, and selecting **Tools ▸ Memory Browser** from the toolbar. Export the navigation log into a convenient place on your computer and then open it with a text editor, such as WordPad on a PC or TextEdit on a Mac. You should see a long list of numbers, as in Figure 5.

The first number in the list is the current time, the second is the latitude, the third is the longitude, and the fourth is the vehicle's velocity. The file then repeats this pattern of measurements.

These numbers might not make much sense unless you know how to interpret them. Let's look at each in turn.

time

The *time measurement* lacks the colons that typically separate hours, minutes, and seconds. To read the time, add these colons yourself. For example, you should interpret the first time measurement in Figure 5 as 17:09:29.

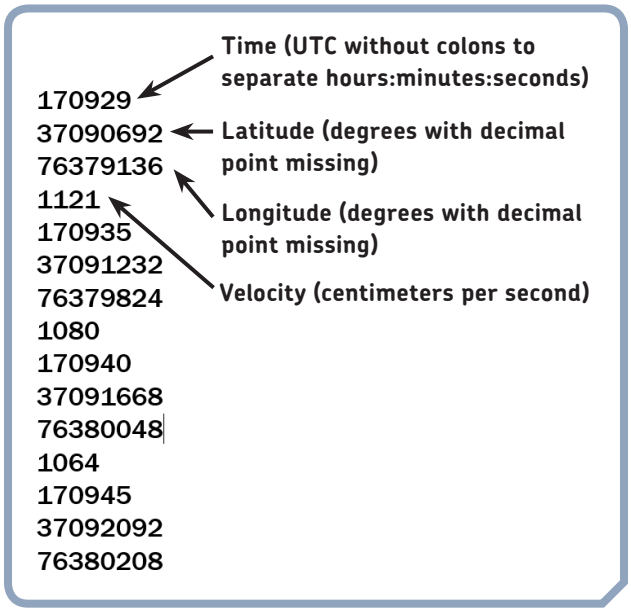


FIGURE 5: The navigation log records a long list of numbers that represent the time, latitude, longitude, and velocity.

The file stores the time as *Coordinated Universal Time (UTC)*, a time measurement based on the prime meridian that passes through Greenwich, England. To convert UTC to your local time, add or subtract the number of time zones away you are from Greenwich, adjusting for daylight saving time. For example, the east coast of the United States is 4 hours behind UTC during daylight saving time (March–November) and 5 hours behind when daylight saving time is not in effect. So someone on the East Coast could subtract 4 hours in daylight saving time (or 5 hours otherwise) from UTC to get the local time. You can also calculate your local time adjustment automatically by using an online time-conversion program such as Timebie (*http://timebie.com/*).

latitude and longitude

The next two numbers in the navigation log, the *latitude* and *longitude*, lack decimal points. Place the decimal points after the second digit of each measurement. The latitude and longitude also lack positive or negative signs. To interpret the values correctly, you just have to know which Earth hemisphere you live in: latitude is positive if you're north of the equator and negative if you're south of the equator; longitude is positive if you're east of Greenwich, England. For example, the United States is north of the equator and west of Greenwich, so if you live there, you can interpret the latitude in Figure 5 as +37.090692 degrees and the longitude as –76.379136. You can enter the coordinates you measure into Google Maps (*https:/www.google.com/maps/*) to visualize the location on a map.

velocity

The fourth number in the sequence is the vehicle's *velocity* in centimeters per second. To convert this number to meters per second (a more appropriate unit for the velocity of a moving vehicle), divide this number by 100. For example, the first velocity value in Figure 5 is 11.21 meters per second.

plotting the vehicle's journey

A good way to visualize the data is to plot it in a graph by importing the navigation log measurements into a spreadsheet or graphing program. You have many such programs to choose from, such as Microsoft Excel on a PC or Numbers on a Mac. Options for open source, free programs include LibreOffice Calc and SciDAVis. This section shows you how to use Excel to process data from the navigation log and then feed it to Google Maps so you can see the path of the car overlaid on a map.

reducing data from the navigation log

To plot the journey on a map, all you need are the latitude and longitude coordinates. You can use Excel to parse out the latitude and longitude data from the navigation log in the following steps:

1. Open Excel and select **Blank workbook**.
2. Select **Data ▸ From Text/CSV**. A pop-up menu will appear, from which you should select **Text Files ▸ All Files** and then find the navigation log that you stored on your computer. Click **Import**.
3. A pop-up showing the navigation log should appear, as in Figure 6. You want only the second and third numbers in each set of four measurements. To parse out the desired numbers, click **Transform Data**.
4. Select **Remove Rows ▸ Remove Alternate Rows**. From the pop-up menu that appears, set **First row to remove** to **1**, **Number of rows to remove** to **1**, and **Number of rows to keep** to **3**. This action strips out the time measurements, leaving latitude as the first number in each repeating series. Then select **Remove Rows ▸ Remove Alternate Rows** again. From the pop-up menu, set **First row to remove** to **2**, **Number of rows to remove** to **2**, and **Number of rows to keep** to **1**. Click **Close & Load**. A *Sheet 2* should now be created out of raw latitude measurements. These measurements are "raw" because they are not yet formatted with a decimal point. Click **A** to highlight the column and type **raw latitude** into the Formula Bar to label the column.
5. To begin the capture of the column of longitude data, repeat step 2.
6. Repeat step 3.
7. In a step similar to step 4, parse out the set of longitude measurements by selecting **Remove Rows ▸ Remove Alternate Rows** and setting **First row to remove** to **1**, **Number of rows to remove** to **2**, and **Number of rows to keep** to **2**. Then select **Remove Row ▸ Remove Alternate Rows** again and set **First row to remove** to **2**, **Number of rows to remove** to **1**, and **Number or rows to keep** to **1**. Click **Close & Load**. This should create a *Sheet 3* of raw longitude values. Label the column by clicking **A** and entering **raw longitude** into the Formula bar. Select the entire column again by clicking **A** and then **Copy** the column.

8. Go to *Sheet 2* and click column **B**. Click **Paste**. This action will bring the latitude and longitude measurements into side-by-side columns.

9. Convert the raw latitude and raw longitude measurements by applying arithmetic to each column. The raw latitude number needs a decimal place after the second digit, so click cell **C2**. In the formula panel, enter **=a2/1000000**. To convert the raw longitude measurements, click cell **D2**. In the formula panel, enter **=-b2/1000000** (remember the minus sign). Columns C and D now represent latitude and longitude in correct format. Label the columns **latitude** and **longitude**, as shown in Figure 7.

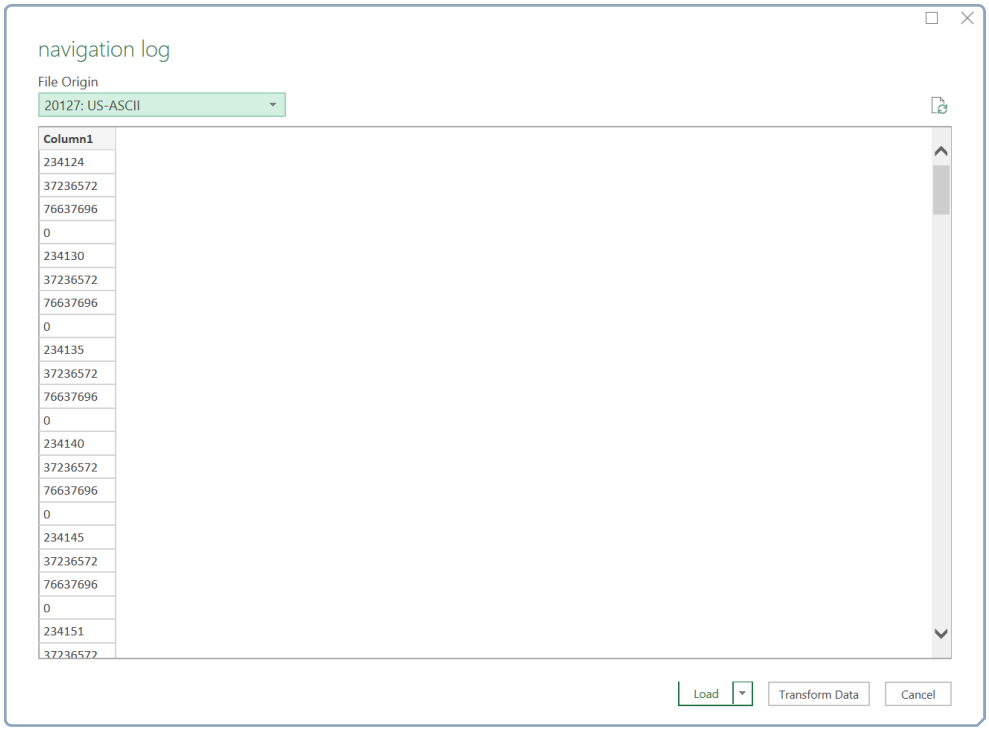


FIGURE 6: As the navigation log is imported into Excel, the repeated sequence of time, latitude, longitude, and velocity displays in a preview.

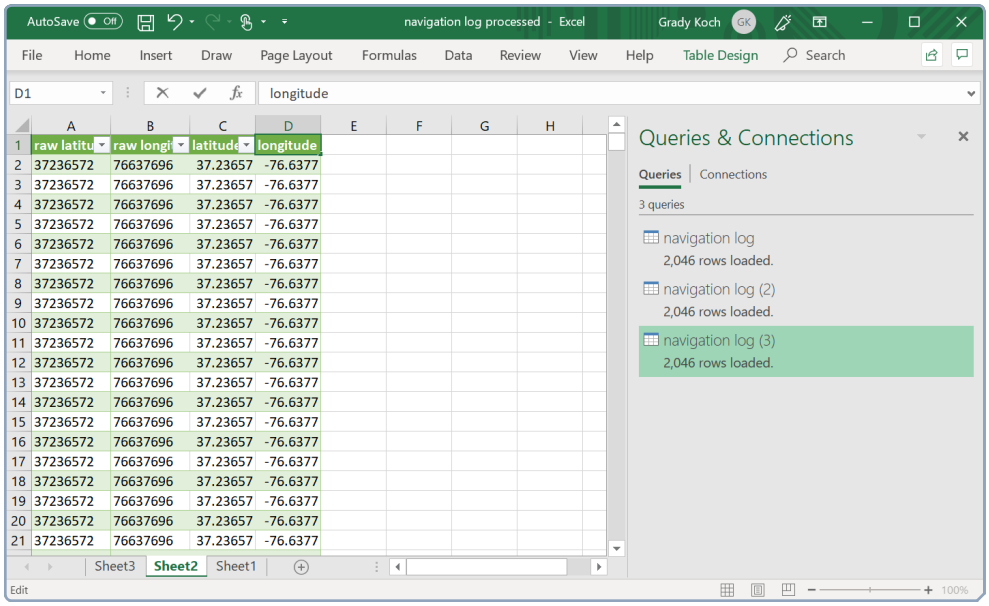


FIGURE 7: Manipulating the raw latitude and longitude coordinates in Excel

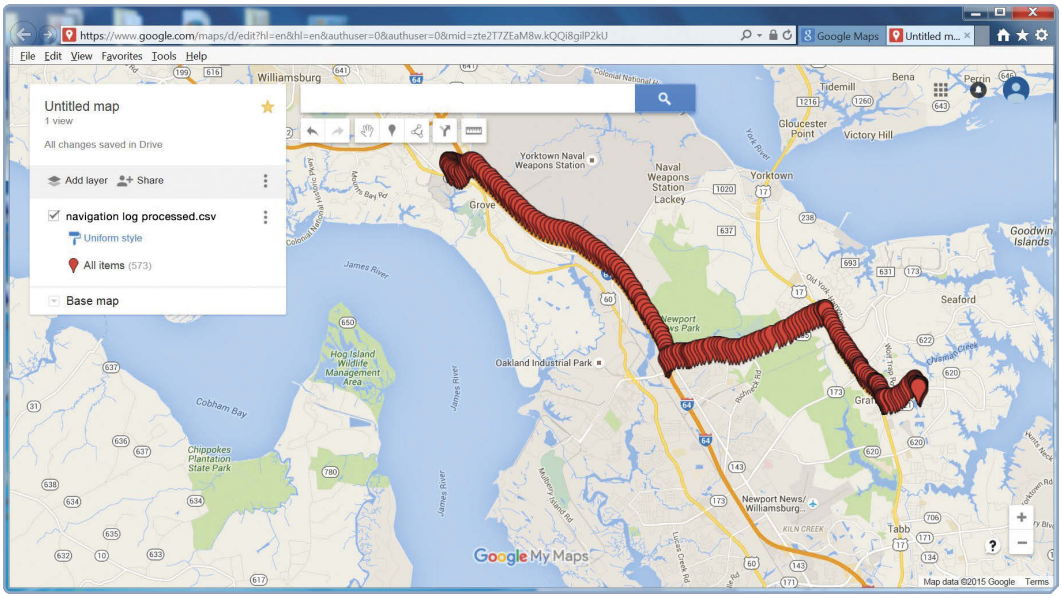


FIGURE 8: *Draw the vehicle's trip on a map by importing coordinates into Google Maps.*

10. Click **C** and **D** to highlight these two columns and then click **Copy**.
11. Go to *Sheet 1* and click **Paste ▸ Values**.
12. Click **Save As ▸ CSV(Comma delimited)(*.csv)** to save the file with a name, such as *navigation log processed*, in a format that other programs can read.

viewing the journey in Google Maps

Now that you've reduced the coordinates into a file with properly formatted sign and decimal places, you can import the coordinates into Google Maps to view the vehicle's journey on a map, as shown in Figure 8.

To make such a map, use a web browser to go to *https:/www.google.com/maps/*. From the Google Maps menu (depicted as three horizontal lines in the upper-left corner of the window), choose **Your places ▸ MAPS**. At the bottom of the MAPS screen, click **CREATE MAP**. Click **Import** and then drag your CSV file (called *navigation log processed* in this example) into the import window and click **Select**.

what you learned

In this chapter, you built a tracking device with a GPS sensor that lets you record and analyze data about a moving vehicle. You collected time, latitude, longitude, and velocity data and then stored and retrieved that data from the EV3 Intelligent Brick. By plotting this data in Google Maps, you visualized a vehicle's movements.