

# Intel® Edison Robotics Sample Guide

---

## Introduction

This example uses sensors from the Starter and Robotics kits mounted on a Seeed\* 4WD Hercules Mobile Robotic Platform to illustrate the concept of a smart vehicle. Other rover or robot frames are also suitable for the project as long as all the sensors are available. The robot is controlled via string commands with the idea of using the process as the child of another front-end service (e.g., a webserver) that is running on the Intel® Edison board.

The sensors used are:

- I2C motor shield and DC motors as the driving components.
- IR distance interrupters for obstacle detection and collision avoidance.
- Digital compass for determining orientation and navigation assistance.
- I2C LCD for displaying information and telemetry data.
- Voltage divider for monitoring the battery level.

Source code for the Robotics sample is located in the following repository:

<https://github.com/intel-iot-devkit/iot-devkit-samples/tree/master/kits/robotics>

## Prerequisites

Complete the Getting Started section on the Intel® Developer Zone:

<https://software.intel.com/en-us/iot/library/edison-getting-started>









Even though it is possible to compile and link the source code for this demo directly on the Intel® Edison board using the integrated GNU\* compiler, we recommend using the Eclipse\* IDE as this Robotics sample is already a part of the Eclipse\* project templates. [Eclipse\\* IDE Getting Started Guide](#) includes instructions on how to install Eclipse\*.

Due to the nature of this project and the involved components, you also need a straight-head precision screwdriver to attach the I2C motor shield terminals. A full precision tool set should make any assembly possible regardless of the rover platform used. Insulation tape, long-nose pliers, extra wires, and wire strippers are also useful things to have. Lastly, some zip ties and/or twist ties will help keep wire clutter to a minimum.

## Hardware setup



The project uses the following parts and sensors:

- Intel® Edison kit with an Arduino\* breakout board and a USB-to-microUSB cable
- Grove\* – Base Shield (SLD01099P)
- Grove\* - LCD RGB Backlight (811004001)
- Grove\* - 3-Axis Digital Compass (SEN12753P)
- Grove\* - I2C Motor Driver (ROB72212P)
- Grove\* - Voltage Divider (POW05161P)
- 4x Grove\* - IR Distance Interrupter (SEN09281P)
- 4x Seeed\* 25GA35 DC Motor (RK-370C-3080EP)

			
Intel® Edison Kit	Base Shield	LCD	Compass
			
Motor Driver	Voltage Divider	Distance Interrupter	DC Motor

Additional parts include:

- SeeedStudio\* 4WD Hercules Mobile Robotic Platform Kit (KIT06071P)
- 7.4V (2-cell) LiPo Battery

	
Hercules Mobile Robotic Platform	7.4V LiPo Battery

These additional parts are shown and used throughout this demo, but they are not project-specific and could be replaced by alternatives. The code sample also works for 2WD rovers.

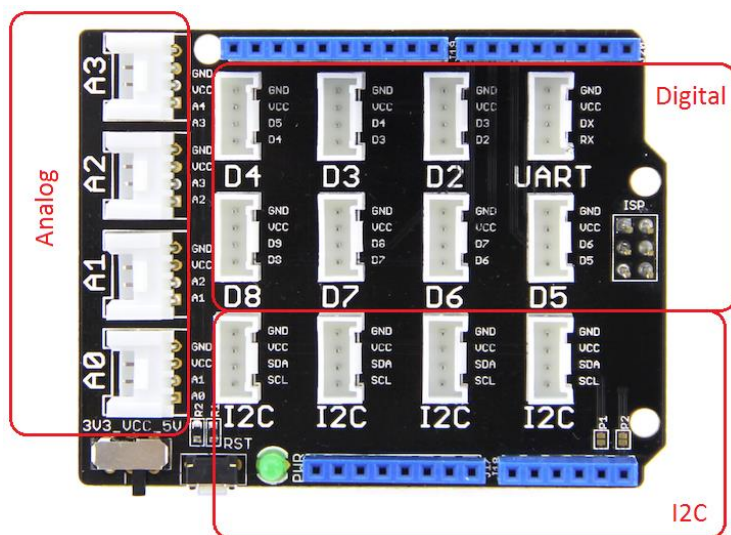
While assembling the parts and testing them, the LiPo battery can be safely replaced with a variable power supply, but make sure your robot is secure and not touching the ground (e.g., leave the wheels off until everything else is in place and tested).

With a 2-cell 7.4V LiPo battery, the rover powers down on its own before the battery is fully discharged as it drops below the minimum rated voltage (~7V) for the Intel® Edison board. For additional power and running time, a 3-cell 11.1V LiPo battery can be used. In this case, however, the battery level needs to be monitored, and the Intel® Edison board needs to be shut down manually when the battery is low, before the battery is damaged. As a general rule of thumb, the cut-off voltage for a 3-cell 11.1V LiPo battery is 9V under load. The provided code does not include a threshold-based shutdown feature.

**Table 1.** Sensor connections to the Grove\* Base Shield

Sensor	Connection	Notes
Grove* - LCD RGB Backlight	I2C	
Grove* - 3-Axis Digital Compass	I2C	See <b>Known limitations</b>
Grove* - I2C Motor Driver	I2C	Connect right-side motors to M1 and left-side motors to M2
Grove* - Voltage Divider	A0	Input from power supply
Grove* - IR Distance Interrupters	D2	Front Left Sensor
	D4	Front Right Sensor
	D6	Rear Left Sensor
	D8	Rear Right Sensor

Sensor connections to the Grove\* Base Shield are listed in **Table 1**. A snapshot of the Grove\* Base Shield with the different connection types highlighted is given in **Figure 1**. For the I2C connections, it does not matter which slots are used since it is a bus.

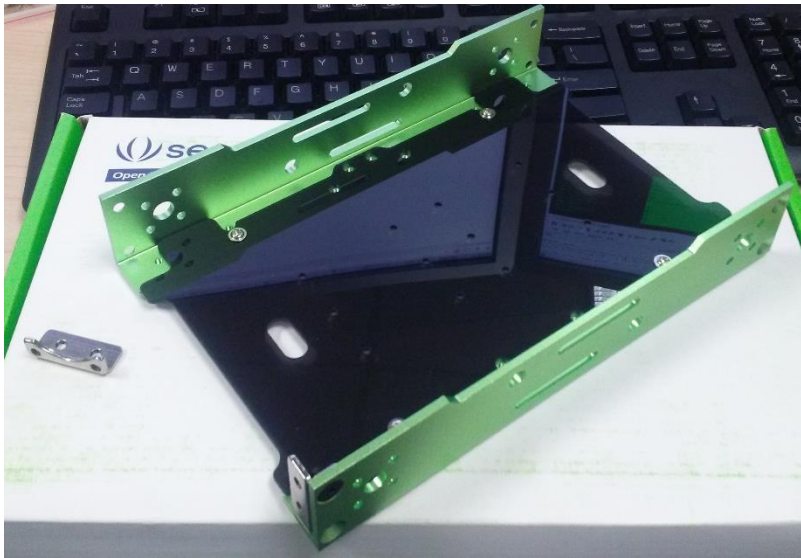


**Figure 1.** Grove\* Base Shield and connection types

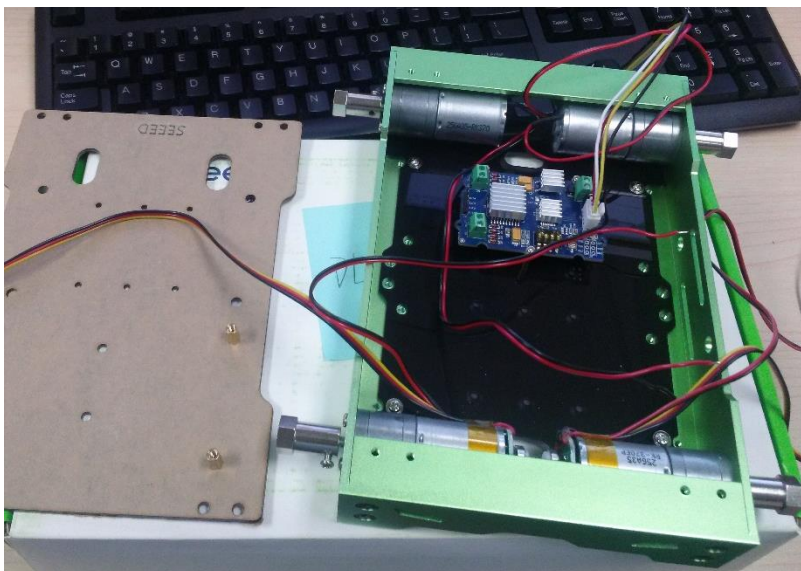
If you're using the SeeedStudio\* Hercules Mobile Robotic Platform, there is a guide available at the following link that explains how it should be assembled:

<http://www.seeedstudio.com/recipe/216-edison-4wd-auto-robotic-platform.html>

Other connections and sensor placement considerations are discussed and illustrated throughout the remainder of the section, starting with some additional figures showing various stages of building the SeeedStudio\* Hercules Mobile Robotic Platform. Use these figures together with the recipe provided by SeeedStudio\* to understand how the sensors from the Robotics kit and the rover frame are used together.

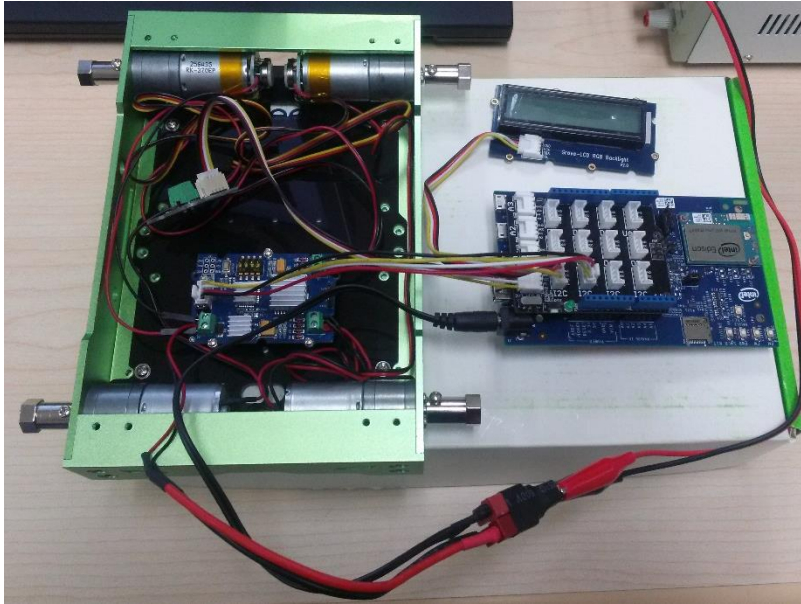


**Figure 2.** Lower rover frame assembled



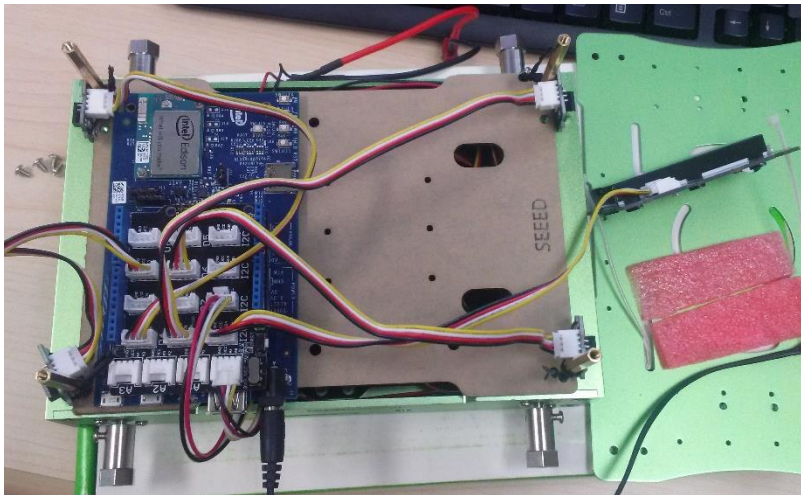
**Figure 3.** DC Motors and I2C Motor Shield added to the frame





**Figure 4.** Motors connected, Intel® Edison, LCD, and voltage divider in place for testing

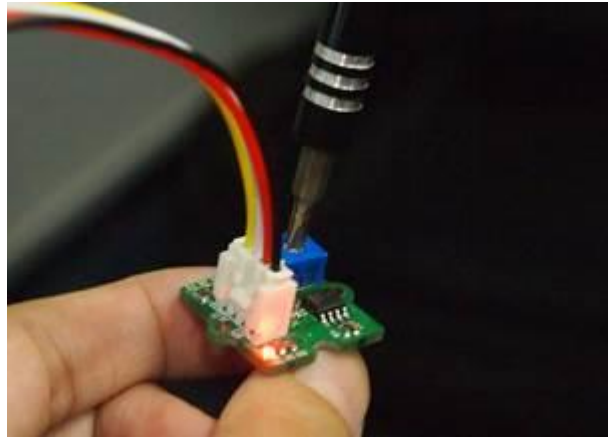
Note that the Grove\* 3-Axis Digital Compass is not shown due to a **Known limitation**. Also, keep in mind that in **Figure 4**, the front of the rover is facing down. To ensure compatibility with the provided code, the right-side DC motors need to be connected to Channel 1 (M1) on the motor driver, and the left-side motors to Channel 2 (M2). The power terminal block on the I2C motor driver can connect both the power source and the voltage divider input - used to monitor the battery level - to the circuit.



**Figure 5.** IR Distance Interrupters attached to the top platform struts

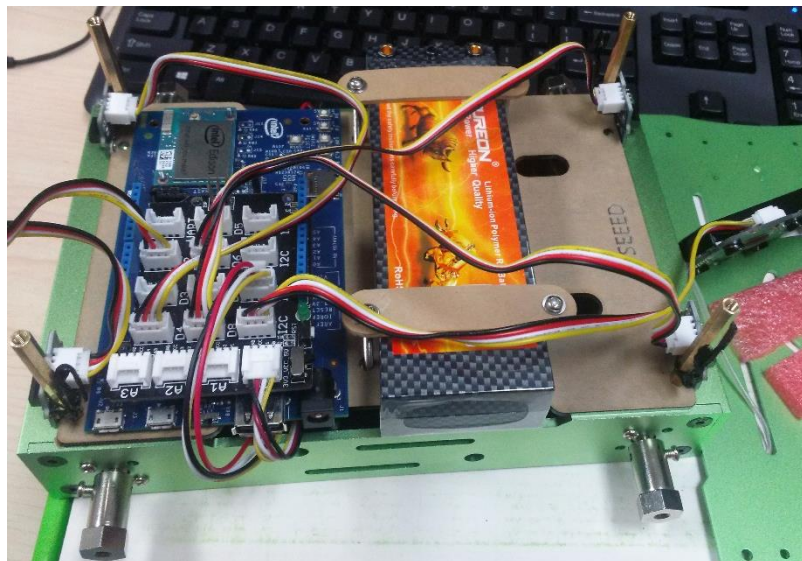
The infrared distance interrupters are used to stop the robot and prevent collisions with static or dynamic obstacles. Positioning and calibration of these sensors are essential for maximizing their usability in this application. Detection range of the sensors can be adjusted with the R8

potentiometer located on the back side of each sensor (same side as the 4-pin connector). In order to calibrate the detection range, place the sensor at the desired distance from some object and adjust the potentiometer until the LED indicator changes from on to off, as shown in **Figure 6**.



**Figure 6** - Adjusting an IR distance interrupter.

Common practice is to place the distance interrupters on the corners of the robot (see **Figure 5**) and tilt them slightly to the side to cover the wide open wheels. We recommend attaching them to the hexagonal distance holders between the base of the rover and the top plate, because it enables you to rotate them to the desired angle and test their coverage before fastening them into place.



**Figure 7.** Top level complete with the battery installed

Once this stage is complete, simply mount the top platform and the wheels and secure the cables with some zip or twist ties - and the rover is ready.

## Known limitations

Grove\* 3-Axis Digital Compass is not compatible with the Intel® Edison Arduino\* board when the board is in the 5V mode. To work around this issue, we recommend using an external voltage level shifter (also known as a level translator/converter). Some products that can be used are:

- [Adafruit\\* 4-channel I2C-safe Bi-directional Logic Level Converter - BSS138](#)
- [SparkFun\\* Logic Level Converter - Bi-Directional](#)
- [SparkFun\\* Level Translator Breakout - PCA9306](#)

If you're using an Intel® Edison mini breakout board instead, this limitation does not apply. When the compass is not used, it is best to remove or comment the lines of code responsible for starting and joining the corresponding thread in the main function of the sample.

Once the front left wheel is in position, the microUSB ports on the Intel® Edison board are obstructed, so make sure the board has Wi-Fi connectivity beforehand.

The Java\* version of the sample needs to be run directly over a ssh or serial connection for full functionality, because the Eclipse\* IDE doesn't forward console input.

## Running and using the code

The preferred method to run the application is using the Eclipse\* IDE as the project is available as a template with the rest of the Intel® IoT Developer Kit projects. You can download the Eclipse\* IDE here:

<https://software.intel.com/iot/downloads#ide>

By the time you're reading this document, the sample might have been updated with additional features. In this case, update the Eclipse\* sample project with the new code available in the GitHub\* [iot-devkit-samples](#) repository. If new sensors were added, make sure you also update the linker flags in the project settings either manually or by using the IoT Sensor Support view (can be found under IoT DevKit menu). All the information required to become familiar with the Intel® Developer Kit version of the Eclipse\* IDE can be found in the [Eclipse\\* IDE Getting Started Guide](#).

When using Eclipse\*, the option to compile and run the code on the target device - in this case, the Intel® Edison board - is readily available. This can be used for testing purposes, but keep in mind that the binary is saved in a temporary folder and removed after the device reboots. Thus, we recommend manually copying the binary to a location on the Intel® Edison board where it can be accessed later so that a different process automatically running on the board on a reboot can load this control application as needed.

The application is multithreaded, with separate threads for separate sensors. The main thread is responsible for receiving and processing commands and sending them to the motor driver. A mutex is in place to avoid contention on writing to the LCD from the compass and the voltage divider threads, while the fourth and final thread handles the infrared distance interrupters. Any of the threads, except for the main one, can be disabled by editing the code if the corresponding sensor functionality is not desired.

**Table 2.** Available commands for rover control

Command	Notes
<i>fwd [0-255]</i>	Move rover forward at set speed. Speed value is mandatory.
<i>rev [0-255]</i>	Move rover in reverse at set speed. Speed value is mandatory.
<i>left [0-255]</i>	Turn left in place at set speed. Speed value is mandatory.
<i>right [0-255]</i>	Turn right in place at set speed. Speed value is mandatory.
<i>stop</i>	Stop movement. Equivalent to any of the previous commands with the speed value of 0.

**Table 2** lists the commands available as input to control the rover and its movements. A basic filter is in place that does not accept commands other than the ones listed above, in the exact format specified. That is, after every directional command, an integer between 0 and 255 needs to be added, signifying the speed. The stop command can be used without the numerical argument, and is also executed by default when an invalid input is received or when any of the infrared distance interrupters is triggered.

A global *batteryThreshold* variable is defined and used to indicate a low battery state by comparison with values read from the voltage divider. The value is hardcoded and set by default to 7.2f. The number represents voltage and can be changed to any desired value before compiling the code. Please refer to the **Hardware** section for more information about minimum operating voltages and LiPo battery considerations.

To exit the control application cleanly, send SIGINT (Ctrl+C on Linux\* OS) followed by EOF (Ctrl+D on Linux\* OS). Since the main loop is waiting for input, it will not process the interrupt signal until the input stream is closed; thus, these two separate steps are required. This should be rather trivial to achieve when the control application is running as a child process of another process. While it can be used as a standalone program, the intended use for the provided code is to be launched by a parent server process hosted on the Intel® Edison board. The server would then be responsible for both controlling the rover and handling communication between the board and a remote device.

## Possible extensions for this sample

The most straightforward extension to a 4WD rover would be to allow for remote control and operation of the vehicle through a nicely crafted user interface. Using an Intel® Edison board, this becomes very intuitive as the integrated Wi-Fi adapter is well suited for such an application.



A built-in u.fl connector also makes it possible for external antennas to be connected. When coupled with a powerful access point or base station, this allows for bidirectional communication over extended ranges, enabling you to send commands to the rover and receive feedback from the onboard sensors.

Live video streaming in compressed formats is also possible given the power of the Intel® Atom™ processor, and some example projects, most notably [edi-cam](#), can show how to set up a video camera on the Intel® Edison board with minimal effort.

Note that the *edi-cam* project does not provide audio streaming and uses a Node.js\* server and WebSockets to stream video. The encoder is FFmpeg, and decoding is done on the client side with jsmpeg using the MPEG-1 format for video. UVC drivers are already bundled with the latest Intel® Edison Yocto\* images, enabling the supported video cameras as soon as they are plugged into the board.

Additionally, the GStreamer\* and ALSA drivers are available in the [Intel® Edison](#) repositories. When installed, they should enable compiling video and audio streaming applications using the newer H.264 and MPEG-4 codecs.

Another interesting idea involves the addition of a GPS sensor for tracking the location of the rover and potential automated navigation when combined with a compass. An example of how to use a GPS sensor is given in our [Transportation & Security Sample](#).

For those using the Intel® Edison mini breakout board for this project, a 5V low-dropout voltage regulator helps power the 5V sensors since the mini breakout design does not provide a steady 5V supply by default.

Follow the [iot-devkit-samples](#) repository for examples and code files showcasing some of these extensions as they are made available.

## Other considerations and disclaimer

Unlike most of the other samples provided within the [Intel® IoT Developer Kit](#) repository, this particular sample uses an additional power supply and/or a lithium battery, adding the risk of electric shock or fire resulting from battery damage. The H-Bridge chip on the motor driver gets hot under continued use, especially at full power and under load. Always double-check the polarities of the connections as inversions are the most common reason for hardware damage. Also, be aware of the moving parts of the rover when controlling it.

Intel Corporation should not be held responsible in the event of misuse, malfunction, damaged hardware/property, or personal injury arising from the assembly and use of this sample project or the included code.