# Arduino Based Optical Sensor Software

This package contains code to be uploaded to the Arduino microcontroller (“Arduino Code” directory) and also Matlab functions to be used to receive communications from the Arduino (“Matlab Code” directory). Which code files to use depends on whether the Arduino should be set up to read from one or two optical sensors; see the appropriate sections below.

Most of the Matlab code is documented; documentation can be accessed via the help function.

## Prerequisites

### Arduino Software

Download the Arduino Integrated Development Environment (IDE) from:

<https://www.arduino.cc/en/Main/Software>

On Windows systems it is recommended to use the Installer if at all possible. Downloading the ZIP file version requires manual setup of device drivers.

With the Arduino plugged in[[1]](#endnote-1) to a USB port, start up the IDE and make sure that the correct board name/version and COM port appears under the Tools menu.

* The COM port can be seen in the Windows Device Manager.
* Some boards like the Due require a support package to be downloaded via the IDE. Click the menu item Tools 🡪 Board 🡪 Boards Manager, select the appropriate package, and click on the Install button that appears.

### Matlab Software

The following instructions are written with users of the Virtual Reality MATLAB Engine ([ViRMEn](http://virmen.princeton.edu/)) package in mind, i.e. how to translate the optical sensor(s) readout into motion in the virtual world. However, all the code in this package can also be run without ViRMEn, and/or integrated into other Matlab-based experiment control software. To do this, the following should be observed:

1. The code should have an initialization section that is used to set up communications with the Arduino device before the start of any behavior related iterations. In ViRMEn this is called the initializationCodeFun() function of the experiment code, and all related instructions can be ported to your own initialization section.
2. The code should have a termination section that is used to clean up communications-related objects after the end of all behavior related iterations. In ViRMEn this is called the terminationCodeFun() function of the experiment code, and all related instructions can be ported to your own termination section.
   * Failure to perform this clean-up will cause Matlab to report an error in future attempts to access the COM port associated with the Arduino device, e.g. in your next attempt to run your behavioral software. In case this happens you can force clean-up of ***all*** open COM ports by executing fclose(instrfindall) in the Matlab console.
3. ViRMEn uses a struct named “vr” to keep track of quantities used/modified throughout the behavior iterations. Outside of ViRMEn, you can provide your own struct like the following (illustrative) example:

vr = struct();

% ... add any other desired fields here

vr = initializeArduino\_...(vr, ...);

% ... run behavior iterations

delete(vr.mr); % clean up

The initializeArduino\_\* functions (explained in the single/two sensor sections below) adds a field vr.mr that is a MouseReader\_\* object; the latter is used to communicate with the Arduino.

1. ViRMEn allows users to specify functions in the movements directory that can be used to translate the raw sensor readout to animal velocities in the virtual world. You can use the provided moveArduino\*.m files as examples for how to code your own movement rule. To use them as-is, you should provide the following fields in the vr struct (see above):

|  |  |  |
| --- | --- | --- |
| vr | Data type | Description |
| .scaleX | scalar | Scale factor to translate raw sensor readout in x to a displacement |
| .scaleY | scalar | Scale factor to translate raw sensor readout in y to a displacement |
| .scaleA | scalar | Scale factor to translate raw sensor readout to angular change |
| .position | Length 4 array | Current coordinates [x,y,z,viewAngle] in virtual world |
| .dt | scalar | Time interval over which the sensor readout was collected. This is used to compute a velocity given the fact that the optical flow sensor measures displacement. |

The moveArduino\* functions return a length 4 array of velocities [vx,vy,vz,vA] computed by applying particular translation rules to the raw sensor readout recorded by the vr.mr object. Explanations of the rules are available in the function documentation.

## Single Sensor Readout

The following files are required to run in this mode:

|  |  |  |
| --- | --- | --- |
|  | Arduino Code | Matlab Code |
|  | ADNS\_aout\_wUSB\_1sensor\\* | OneSensor\\* |
|  |  | movements\\* |
|  |  | RigParameters.m |
|  |  | MovementSensor.m |

Installation:

1. Load ADNS\_aout\_wUSB\_1sensor.ino in the Arduino IDE and upload it to the board.
2. Add the code in the Matlab Code\movements directory to the movements directory of your ViRMEn installation. If already running, the ViRMEn GUI needs to be restarted to pick up these changes.
3. Add the rest of the Matlab code to somewhere in your Matlab path, including subfolders.
4. Edit RigParameters.m to specify the COM port for the Arduino board and the appropriate calibration constants for your setup (see the Sensor Calibration section).
5. Edit your ViRMEn experiment code:
   * In initializationCodeFun(), add the line

vr = initializeArduino\_1sensor(vr, 1, 1/2.5);

Remember to substitute the desired x/y and view angle scale factors as explained in help initializeArduino\_1sensor.

* + In terminationCodeFun(), add the line

delete(vr.mr);

1. Change the ViRMEn world movement function to moveArduino.
   * It is also possible to use the moveArduinoLinearVelocity or moveArduinoLiteral functions, provided that you change the first line of those functions to the following in order to read from the first (and only) sensor:

[dY,dX] = vr.mr.get\_xy\_change();

## Two Sensor Readout

The following files are required to run in this mode:

|  |  |  |
| --- | --- | --- |
|  | Arduino Code | Matlab Code |
|  | ADNS\_aout\_wUSB\_2sensors\\* | TwoSensors\\* |
|  |  | movements\\* |
|  |  | RigParameters.m |
|  |  | MovementSensor.m |

Installation:

1. Load ADNS\_aout\_wUSB\_2sensor.ino in the Arduino IDE and upload it to the board.
2. Add the code in the Matlab Code\movements directory to the movements directory of your ViRMEn installation. If already running, the ViRMEn GUI needs to be restarted to pick up these changes.
3. Add the rest of the Matlab code to somewhere in your Matlab path, including subfolders.
4. Edit RigParameters.m to specify the COM port for the Arduino board and the appropriate calibration constants for your setup (see the Sensor Calibration section).
5. Edit your ViRMEn experiment code:
   * In initializationCodeFun(), add the line

vr = initializeArduino\_2sensors(vr,1,1,MovementSensor.BottomVelocity);

Remember to substitute the desired x/y and view angle scale factors as explained in help initializeArduino\_2sensors.

* + In terminationCodeFun(), add the line

delete(vr.mr);

1. Change the ViRMEn world movement function to moveArduinoLinearVelocity or moveArduinoLiteral.
   * Note that both these movement functions read from the second sensor. If you want to use the first sensor, change the first line in the function to

[dY,dX] = vr.mr.get\_xy\_change();

* + Similarly if you want to use the moveArduino code with the second sensor, change the first line in it to:

[dY,dX] = vr.mr.get\_xy\_change();

1. For boards with programming vs. data ports, the programming port should be used. [↑](#endnote-ref-1)