Enes100ArduinoLibrary

Arduino library for use in the ENES100 course with Vision System v4 and APC220s

Downloading and Installation

To use this library, download the contents from this Github repository by going to the green **Clone or Download** menu and clicking **Download ZIP**. (There is no need to unzip the file.) Open the Arduino IDE and install the library by going to **Sketch > Include Library > Add .ZIP Library** and selecting the downloaded file. You must have Arduino IDE version 1.5.0 or above. The most current version of Arduino IDE can be downloaded from the Arduino website.

If you have an older version of the library on your computer, you *must* delete it before adding a newer version. Failure to do this may cause file conflicts and it is not guaranteed that the library will work properly.

Setup

Hardware

Communication with the Vision System is done using APC220 wireless RF transceivers. APC220s are available in your team electrical toolbox or can be checked out from the tool cabinet through a Teaching Fellow. The APC220 has 7 pins:

Pin	Description	Connect to...
GND	Ground	Common ground
VCC	Voltage supply	+5 V
EN	Enable	+5 V
RXD	Serial receive	Arduino serial transmit
TXD	Serial transmit	Arduino serial receive
AUX	Auxiliary	Not connected
SET	Setting mode	+5 V

Software

To use the library, you have to direct the compiler to include it in your code. Go to **Sketch > Include Library > ENES100**, or add it manually by typing

```
#include "Enes100.h"
```

at the very top of your file.

All the functionality is encapsulated in the `Enes100` class. To use it, you'll first need to create an instance of the class. At the top of your code (outside of the `setup()` and `loop()` functions), add

```
Enes100 enes("BlackBoxTeam", BLACK_BOX, 3, 8, 9);
```

In the code shown above, `enes` is an identifier. You may replace it with any other valid C identifier. The first parameter is the name of your team, which will appear on the Vision System with your serial port. `BLACK_BOX` refers to the mission type of your team. The full list of valid mission names is

- BLACK_BOX
- CHEMICAL
- DEBRIS
- FIRE
- WATER

(As with all code samples in this document, these are case sensitive.) The third argument is the ID of the marker that your team will be using. The third and fourth arguments are your serial receive and serial transmit pins (recall that your RX pin connects to the APC's TX pin and vice versa).

When choosing the pins to use for communication, there are a few things to keep in mind. Firstly, you cannot use pins 0 and 1 for RF communication (or for anything else). The RX and TX labels on those pins refer to the Arduino's serial transmit and receive with the computer. If you block those pins, you will not be able to upload code to your Arduino. Secondly, it is preferable not to use PWM pins for your communication; you'll want to save those for things like controlling your motors.

That's it! You're now ready to communicate!

Usage

Example code for each type of mission is included with the library. To view examples, open Arduino IDE and go to **File > Examples > ENES100**. (You must restart the IDE after adding the library for the examples to show.)

For your OSV to get information about its location and its destination, you will need to communicate with the Vision System using the library. To get the location of the destination, use the `retreieveDestination()` method.

```
enes.retrieveDestination();
```

This method returns a `bool` indicating if the operation succeeded. If the operation succeeded, the coordinates of the destination will be stored in the `Enes100`'s `destination` member. You can access the coordinates using

```
enes.destination.x; // x Coordinate
enes.destination.y; // y Coordinate
```

For the Black Box mission, the destination is unknown. Calling `retrieveDestination()` is not required and doing so will not update the destination coordinates.
To request an update of your OSV's location, call the `updateLocation()` method of your `Enes100` object. This method also returns a value indicating if it succeeded or not.

```
if (enes.updateLocation()) {
    enes.location.x; // x Coordinate
    enes.location.y; // y Coordinate
    enes.location.theta; // Theta
}
```

The x and y coordinates are the distance in meters from the y axis and x axis to the center of your marker. Theta is measured in radians from -π to π, with zero being parallel to the x axis, increasing counter-clockwise.

At points in your mission, you may want to send information to be displayed on the Vision System. Writing information to the Vision System is similar to writing information to the Serial console. The `print()` function will write a message to the console. The `println()` function will write a message to the console followed by a new line. These functions can accept strings, integers, and doubles as arguments.

```
enes.print("Our x coordinate is: ");
enes.println(enes.location.x);
```

Note: Your messages to the Vision System may not include the `#` or `*` characters. They *will not* appear in the message pane and may prevent further communication.
Once your OSV reaches the destination, alert the Vision System that you've done so with the `navigated()` method.

```
enes.navigated();
```

As your OSV completes its objectives, it will need to alert the Vision System. When your OSV completes a base objective that requires transmission, call the `baseObjective()` method with the value that you've calculated. When your OSV completes a bonus objective that requires transmission, call the `bonusObjective()` method with the value that you've calculated.

```
enes.baseObjective(2.34);
enes.bonusObjective(STEEL);
```

For the Water mission, valid water types for the base objective are

- FRESH
- POLLUTED

- SALT

For the Debris mission, valid material types are

- COPPER
- STEEL

Once you've finished your mission, you'll need to alert the Vision System that you're done. Use the `endMission()` method to stop the timer on the Vision System.

```
enes.endMission();
```

The Coordinate Object

The ENES100 library also includes a Coordinate class that holds an x, y, and theta. Students may use Coordinates in their code as desired. The `destination` and `location` members of `Enes100` are Coordinate objects. The x, y, and theta are accessed in the same way as in `destination` and `location`, i.e. `coordinate.x`, `coordinate.y`, and `coordinate.theta`. For convenience, we have included 3 constructors for the Coordinate object.

```
Coordinate coordinate1(); // Represents the point (0,0,0)
Coordinate coordinate2(1.2, 0.7); // Represents the point (1.2,0.7,0)
Coordinate coordinate3(1.2, 0.7, 1.1); // Represents the point (1.2,0.7,1.1)
```

For the Black Box mission, the base objective must be transmitted using a Coordinate object.

```
Coordinate blackBox(3.2, 1.6);
baseObjective(blackBox);
```

Function List

The following are functions of the `Enes100` class.

`void baseObjective(value)`
Sends value for a base objective.
`void bonusObjective(value)`
Sends value for a bonus objective.
`void endMission()`
Alerts the Vision System that you've finished your mission. Prevents further code execution on the Arduino.
`void navigated()`
Alerts the Vision System that you've reached your destination. For chemical teams, splits the timer.
`void print(message)`
Sends a message to be displayed on the Vision System. Accepts a String, integer, or double as an argument.

`void println(message)`
Sends a message to be displayed on the Vision System followed by a new line character. Accepts a String, integer, or double as an argument.
`bool retrieveDestination()`
Returns: True on success, false on failure. Retrieves the coordinates of the destination from the Vision System.
`unsigned long updateLocation()`
Returns: Time elapsed to update the location, or 0 on failure. Updates the OSV's location information.

Competition Procedures

During the competition, messages sent using `print()` and `println()` will not be shown on the Vision System console. The console will only print out the values that you send using `baseObjective()` and `bonusObjective()`.