arduinolibrary

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

Downloading

To use this library, download the contents from this Github repository by going to the green **Clone or Download** menu and clicking **Download ZIP**. Once you have downloaded the ZIP file, unpack it. Move the extracted folder into your Arduino libraries folder, typically located at **Documents > Arduino > libraries**.

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 will cause file conflicts and it is not guaranteed that the library will work properly.

Setup

To use the library, you have to insert it into your code. At the very top of your Arduino file, add

```
#include "enes100.h"
```

This file will include the rest of the files required to use the library and APC for you.

The `RF_Comm` class is how you communicate with the Vision System. To use it, you'll first need to create an instance of the `SoftwareSerial` class. At the top of your code (outside of the `setup()` and `loop()` functions), add

```
SoftwareSerial mySerial(8, 9);
```

Replace 8 and 9 with the pins you plan on using for RX and TX respectively (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. **You must use a PWM pin for your TX pin, although this is not required for your RX pin.** It is preferable not to use PWM pins for your RX pin; you'll want to save those for things like controlling your motors.

Next, you'll need a `Marker` object to store the information about your OSV's location. Instantiate one immediately below your `SoftwareSerial` declaration.

```
Marker marker(3);
```

Replace 3 with your team's marker number.

Then, immediately below your `Marker` declaration, instantiate your `RF_Comm` object.

```
RF_Comm rf(&mySerial, &marker);
```

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

Usage

For your OSV to get information about its location, you will need to communicate with the Vision System using the library. To request an update of your OSV’s location, call the `updateLocation()` method of your `RF_Comm` object.

```
rf.updateLocation();
```

The information sent by the Vision System will be stored in your marker object. To access it, use `marker.x`, `marker.y`, and `marker.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.

At points in your mission, you will need 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.

```
rf.print(“Our x coordinate is: “);
rf.println(marker.x);
```

As your OSV completes its objectives, it will need to alert the Vision System. To distinguish these messages from normal transmissions, use the `transmitData()` function. `transmitData()` accepts two parameters: the objective the OSV is completing and the data associated with that objective. Specific keywords have been defined for use with this function. **You must use these values when appropriate, as the Vision System will use them to print out the objective that you complete.**

```
rf.transmitData(BASE, 2.8);
rf.transmitData(BONUS, FIRE_SITE_C);
rf.transmitData(END_MISSION, NO_DATA);
```

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.

Function List

The following are functions of the `RF_Comm` class.

`int updateLocation()`
Returns: True on success, false on failure.
Updates the Marker object with information transmitted by the Vision System.

`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.

`void transmitData(objective, data)`
Returns: True on success, false on failure. Failure occurs if an invalid objective is provided or if the function has been called too many times for a given objective.
To be used when starting or ending a run, or when a mission objective has been completed. The objective keywords are listed below, with sub-lists of data keywords for each one. **You must use the keywords when using this function.**

1. START_MISSION
    - NO_DATA
2. NAV (Specify your mission)
    - CHEMICAL

        - **This will split the timer on the Vision System**
    - FIRE
    - MATERIAL
    - TERRAIN
    - WATER
3. BASE
    - For fire teams...
        - FIRE_SITE_A
        - FIRE_SITE_B
    - For material teams...
        - STEEL
        - COPPER
    - For terrain teams...
        - The length of the boulder
    - For water teams...
        - The depth of the water
    - For chemical teams...
        - The pH of the pool
4. BONUS
    - For fire teams...
        - FIRE_SITE_C
        - FIRE_SITE_D
    - For material teams...
        - The mass of the material
    - For terrain teams...
        - GREEN
        - BLACK
        - The surface area of the boulder
    - For water teams...
        - FRESH
        - SALT
    - For chemical teams...
        - The pH of the pool

5. END_MISSION

    - NO_DATA

    Specifying END_MISSION will stop the timer and prevent your Arduino from executing any more instructions

Competition Procedures

During the competition, **only the messages you send using transmitData will be scored.** There is a toggle box named "Show debug messages" on your Serial monitor on the Vision System. When this is checked, all messages sent using "transmitData" and "print" will be displayed. When this box is not checked, only transmitData will be allowed to print to the monitor. This is the setting that will be used during the competition, and so any information sent using "print" will not be seen or scored.

Below is a general flow of when you should use the different flags in transmitData to communicate with the Vision System:

1. START_MISSION: **DO NOT CALL.** This is automatically called when your code initializes
2. NAV: When your OSV reaches its destination
    - NOTE: for the Chemical team, this will split the timer on the Vision System
3. BASE: When a base mission that requires transmission with the Vision system is completed

4. BONUS: When a bonus mission that requires transmission with the Vision system is completed
5. END_MISSION: When the OSV has completed all of its base and bonus objectives
    - NOTE: your Arduino will be prevented from executing any instructions once this is called so **do NOT call** transmitData with this flag until your OSV has completed all objectives