IMU Installation Instructions

A more in-depth overview of the library installation instructions can be found here:

https://www.phidgets.com/docs/OS_-_Linux#Source%20Install

Please note that Phidget is the company who produces the IMU you are using.

Furthermore, all Linux terminal commands should be run without the $ character.

1. The first step is to install all the required libraries from Phidget. You can use the instructions above for a more in-depth explanation if you have problems with the installation. For now, simply run the following commands to install everything you will need:

   ```
   $wget -qO- http://www.phidgets.com/gpgkey/pubring.gpg | apt-key add - && echo 'deb http://www.phidgets.com/debian bullseye main' > /etc/apt/sources.list.d/phidgets.list
   ```

   ```
   $apt-get update
   ```

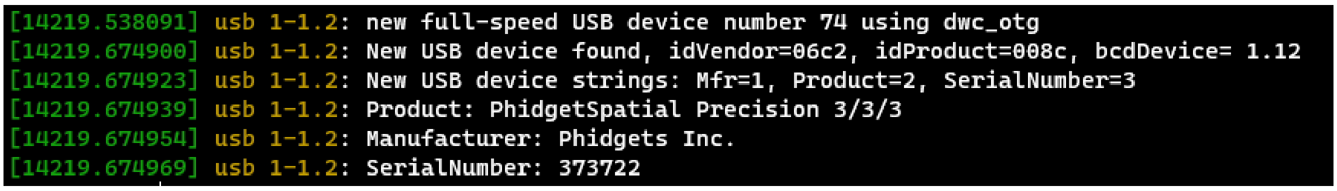
   ```
   $apt-get install libphidget22
   ```

   ```
   $apt-get install libphidget22-dev
   ```

2. You will need to obtain the Serial number of your IMU. This number is necessary for the library to work. If you have already plugged your IMU into the Raspberry Pi, then unplug it. Now plug the IMU into the Raspberry Pi. Wait for a couple of seconds, and then run the following command:

   ```
   $dmesg
   ```

   You will see the terminal print something similar to the following:



```
[14219.538091] usb 1-1.2: new full-speed USB device number 74 using dwc_otg
[14219.674900] usb 1-1.2: New USB device found, idVendor=06c2, idProduct=008c, bcdDevice= 1.12
[14219.674923] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[14219.674939] usb 1-1.2: Product: PhidgetSpatial Precision 3/3/3
[14219.674954] usb 1-1.2: Manufacturer: Phidgets Inc.
[14219.674969] usb 1-1.2: SerialNumber: 373722
```

   You need to look for the line that says Product: PhidgetSpatial Precision 3/3/3. This will ensure that this device is in fact your IMU, and you can then record the serial number, as shown on that bottom line.

3. You will now need to setup the library that we have provided. This contains two files, the c file and the header file. As with the GPS, you should place these two files within some sort of "Libraries" directory, ideally in the same directory as the GPS library files. The library now has to be compiled into an object file so that it can be used.
   Ensure that the terminal is within the same directory as the library files, and use the following command to compile it:

   ```
   $gcc -c -g imu_library.c
   ```

4. To understand how to use the library, we will also share the following demo code and go over how it works.

```
//Written by Tiziano Wehrli
//With library, compile with:
// gccarm -g -o imu_lib_demo.out imu_lib_demo.c -lphidget22 ../Libaries/imu_library.o
#include "eyebot.h"
#include <stdio.h>

void getHeading(double heading, double timeStamp) {
    printf("I have obtained the heading from my function: %lf \n", heading);
}

int main() {
    initializeIMU(373722, &getHeading);
    while(1) {

    }
    closeIMU();
}
```

This is the IMU demo program. It operates on a very similar principle to the GPS demo program, except that it has been simplified even further. The IMU library takes a function, and whenever the IMU has a reading available, the library will call that function. You would write the code to do whatever you need with the IMU data in this function. You can then have a main loop running where you do anything else that needs to be done, and whenever you have IMU data available, the program will jump into the IMU function and process it, effectively working as an interrupt system.

```
void getHeading(double heading, double timeStamp) {
    printf("I have obtained the heading from my function: %lf \n", heading);
}
```

This part is the function to be called whenever IMU data is available. You would replace the print statement with whatever you wish your program to do. It is important to note that you can obtain the current heading reading using the heading parameter, and you can also obtain a timestamp using the timeStamp parameter. You may find this timestamp useful when implementing a control routine.

```
initializeIMU(373722, &getHeading);
```

This is the initialization line for the IMU. The first parameter is the serial number of the IMU, which you would have previously determined. The second parameter is the address of the function to be called whenever a reading is obtained. Please note that to give the address, simply type the name of the function with the & character before it. Do not include the brackets or any parameters. Once this line has been called, the interrupt is “attached” and any subsequent readings from the IMU will trigger the function.

```
closeIMU();
```

This line needs to be called once when you finish, to ensure everything is nicely cleaned up before the program ends. You can also use this line to “detach” the interrupt, in case you need to temporarily stop the interrupt function from being called. You could then “reattach” the interrupt by using the initialisation function mentioned above.

5. The last step is to compile the program so that it can be run. It would be a good idea to compile the test program without modifying it first, just to see that everything is working. The compile instructions have been written at the top of the demo program, and are as follows:

```
gccarm -g -o imu_lib_demo.out imu_lib_demo.c -lphidget22 ../Libaries/imu_library.o
```

The first term is the compiler command, the second and third term are compiler flags, the fourth term is the name of the compiled program (you can change this as you see fit), the fifth term is the name of the program you are compiling and the sixth term is important, as it is required to link the library we have created with the Phidget library.

Take close note of the seventh term, which is the final one. This is the path to the imu_library object file you created previously. This path is the relative path from your current directory to wherever you have placed imu_library object file. It is important that you correctly determine this path, or the compile will fail.

Furthermore, if you rename the demo file, or create a new one, you will need to change the fifth term (which reads “imu_lib_demo.c”) to reflect the new name of the code you are attempting to compile.

You can then run the program using:

$./imu_lib_demo.out

You should see the heading of the sensor being printed on the screen.

Documentation

The documentation of the functions we provide in our library are listed as follows. They can also be found within the imu_library.c file.

To initialize the IMU:

```
/*
    Function to initalize the IMU.

    @PARAM IMU_serial_number The serial number of the IMU being used.
    @PARAM inputFunction The function to be called whenever IMU data is available. It should take two
parameters, the first one is the heading, and the second is the timestamp.
    @return Exit status, 0 if successful.
*/
int initializeIMU(int IMU_serial_number, void (*inputFunction)(double, double)) {
```

To close the IMU:

```
/*
    Function to close the IMU down neatly.

    @return Exit status, 0 if successful.
*/
extern int closeIMU() {
```