# **Weekly Report – W14 Spring 2023**

## **Task & Problem**

1. Confirm the exact wire/cable we are going to use in the experiment for connection of the ESP32 & PC and Arduino board & PC, and generate a purchase list;
2. Examine the sampling rate/frequency of the IMUs (or ESP32) is stabilized at 112 Hz;
3. Optimization about the code.

## **Solution**

1. Final choice for auxiliary cables for experimental use

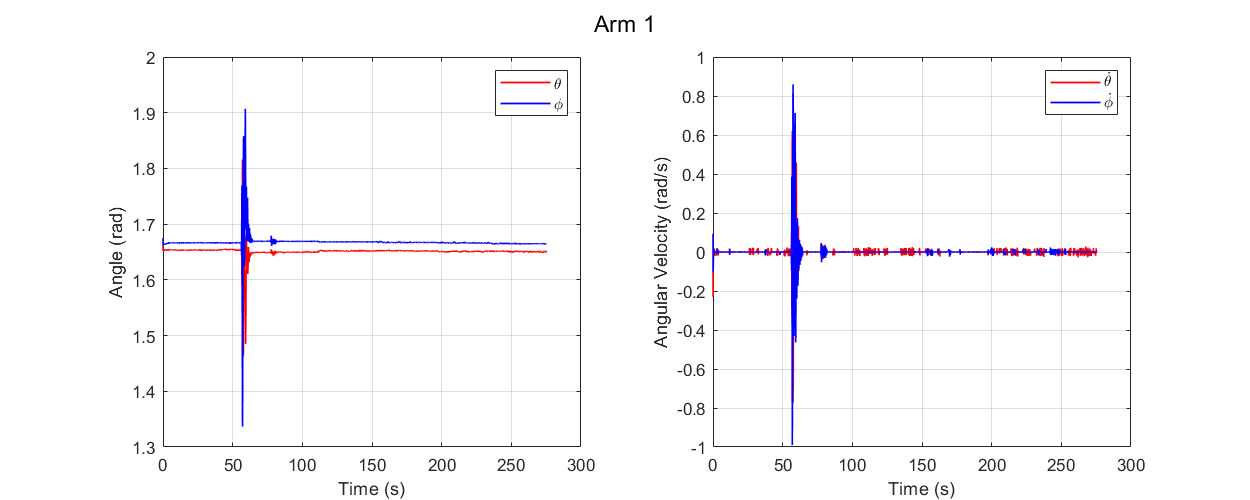
As considered, our final choice will be using 5 25 ft long USB Type A to USB Type A extension cables (3 for ESP32, 1 for Arduino board and the left 1 for backup use). The reason for making this decision is that all the additional cables are in the same type, which will be convenient to change and manage, and also more importantly, since all the sync cables can work properly so far, no one would guarantee that if we directly buy some cables to connect the PC and devices, they can work, so this will be a safe choice.

1. Examine the sampling rate/frequency of IMUs

To test the stability of the sampling rate/frequency of the IMUs, I designed the following groups of experiments to testify this specification from long time experiment durability and the influence of heavy workload from data post processing (it can be the filter, the angular velocity calibration or something else).

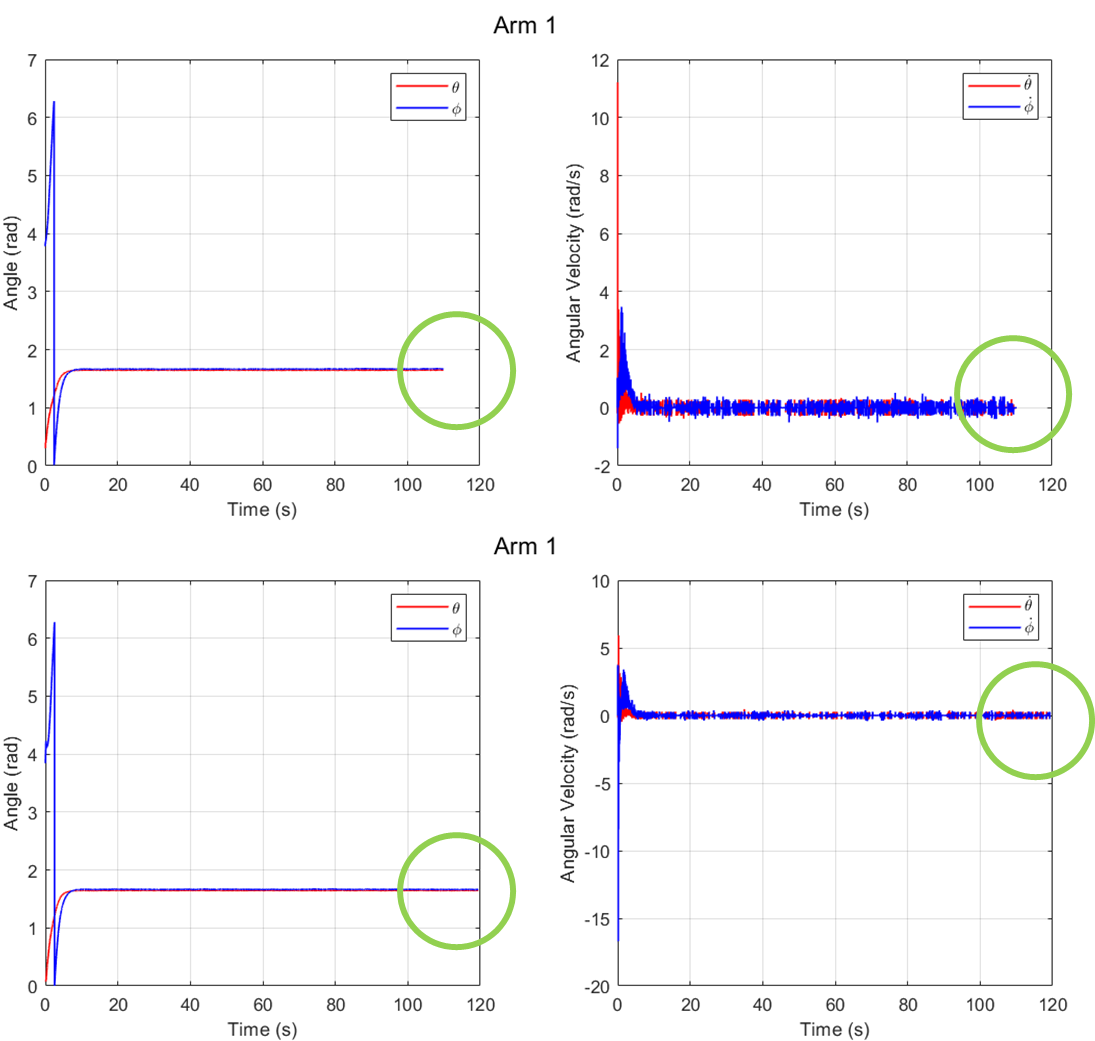
* Keep data collecting for 2 mins and meanwhile let the whole system be stationary (authentic mode);
* Keep data collecting for 2 mins and meanwhile let the whole system be stationary (perfect mode);
* Keep data collecting for 10 mins and meanwhile let the whole system be stationary (perfect mode);
* Keep data collecting for 10 mins and meanwhile keep moving the arm continuously after the 10 s stabilization process until the end of the experiment (authentic mode);
* Keep data collecting for 10 mins and meanwhile keep moving the arm continuously after the 10 s stabilization process until the end of the experiment (perfect mode);

Before I formally did the experiments aforementioned, I did a trial test for 5 mins’ data collection under perfect mode meanwhile keeping the whole system stationary, the only difference for the test condition was that the group of data was collected for the first time the PC (or the MATLAB) started up, which can be seen in the figure below.



**Fig. W14-1** The data collection for the first time after the PC (or MATLAB) starts up for 5 mins under perfect mode

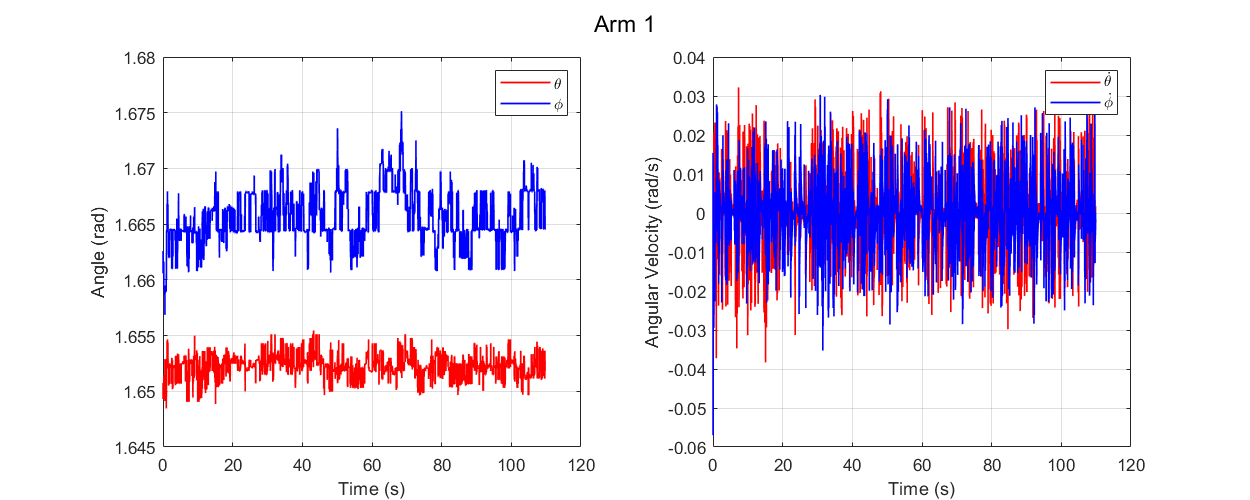
We can see that even after the stabilization process, there still emerges large angle and angular velocity fluctuations at around 50 s (which is actually 60 s in the real time, because for the perfect mode, the first 10 s’ data will be cut out). And we can also find that even though we have excluded the stabilization data points, it seemed that another 15 s’ data was missing, the reason might be owing to it will take some time for MATLAB to set up connections with the devices via serial ports for the first time it starts up, there is lot of servo lag; to confirm if my judgement was correct, I rebooted my PC and changed the test time to 2 mins and collected data when the whole system was stationary for twice under authentic mode (the first and second time after MATLAB started up), and the results are shown below.



**Fig. W14-2** The data collection for the first and second time after MATLAB started up for 2 mins under authentic mode

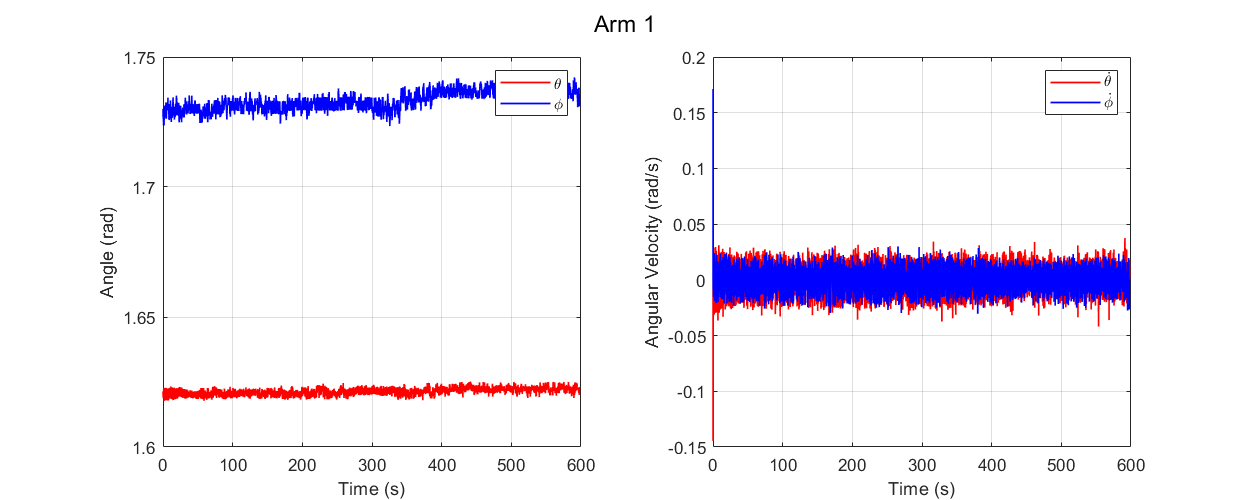
As shown in the part highlighted by the green circles, it seemed that MATLAB will spend about 10 s to finish the setting up for the communications between PC and other electronics. And based on the previous figure, the “first shift” data is not so reliable, I will include this into the experimental schedule.

Given this situation, all the rest tests were performed after the “first shift”, and their results can be seen below.



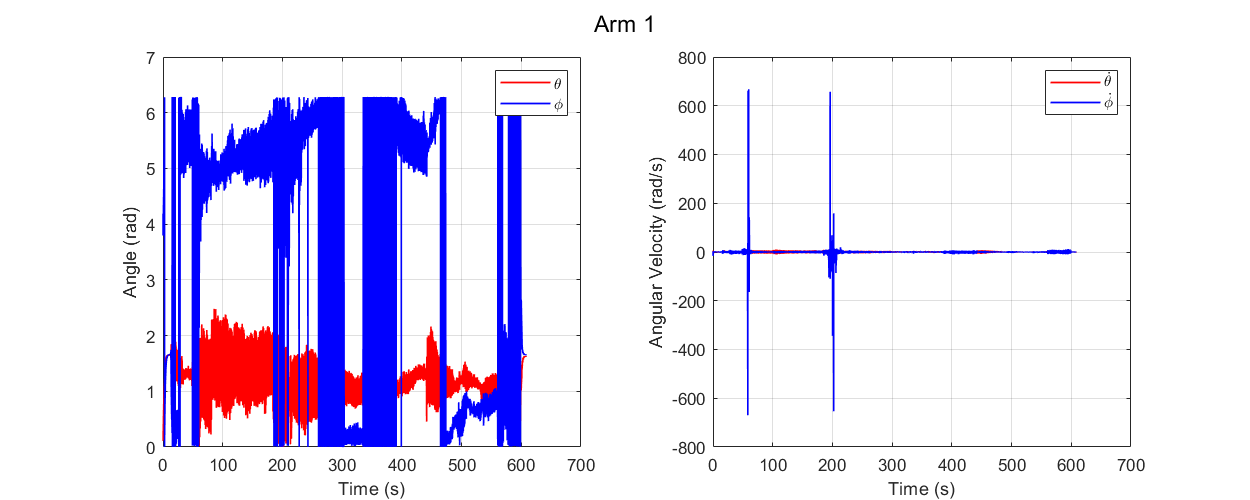
**Fig. W14-3** The data collection for 2 mins when the whole system was stationary under perfect mode

From this figure, no matter for the angle or the angular velocity, they performed quite stable within a reasonable range, and due to the specification of the perfect mode, the first 10 s’ data was cut off, so the total experiment time duration was rightly about 120 s (2 mins).

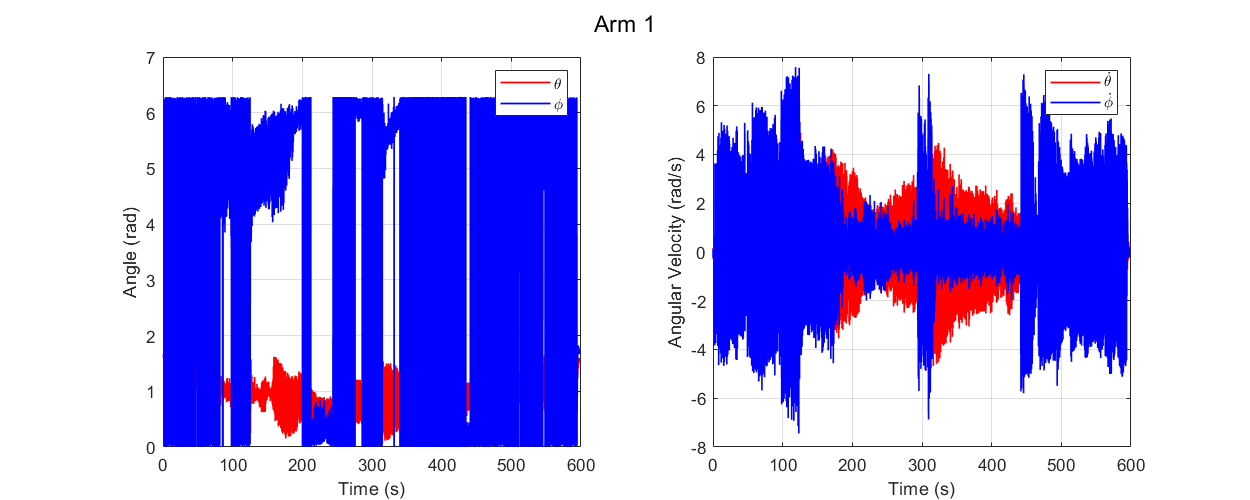


**Fig. W14-4** The data collection for 10 mins when the whole system was stationary under perfect mode

And from this figure, we can see that long time experiment did not quite affect the real time data collection and sampling rate even under perfect mode, it seems that the efficiency of my code is acceptable, but it still needs to be verified under tests with continuous motions.



**Fig. W14-5** The data collection for 10 mins when the arm was moving continuously after the first 10 s’ stabilization process under authentic mode



**Fig. W14-6** The data collection for 10 mins when the arm was moving continuously after the first 10 s’ stabilization process under perfect mode

From the two figures above, we can temporarily ignore the noise consisted in angular velocities (it heavily relies on the hardware or the PC), we can clearly see that no matter for the authentic or the perfect data collection mode, long time experiment and complex data post processing did not affect the sampling rate; back to the noise issue, the perfect mode performed much better than the authentic mode, which can also prove that our filter worked properly.

To figure out what the specific sampling rate for all the experiments performed above, I made a table below.

**Table W14-1** Verification of sampling rate under different experimental conditions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Experiment Type | Data Collection Mode | Duration | Data Point No. | Sampling Rate |
| First trial after PC starts – stationary | Perfect | 5 mins | 31962 | 106.54 Hz |
| First trial after PC starts – stationary | Authentic | 2 mins | 12323 | 102.69 Hz |
| Second trial after PC starts – stationary | Authentic | 2 mins | 13380 | 111.5 Hz |
| Stationary | Perfect | 2 mins | 13423 | 111.86 Hz |
| Stationary | Perfect | 10 mins | 68091 | 113.49 Hz |
| Moving | Authentic | 10 mins | 68124 | 113.54 Hz |
| Moving | Perfect | 10 mins | 68105 | 113.51 Hz |

We can see that except for the first trial after PC starts, the sampling frequency performs pretty stable for other cases, which are reliable.

1. Optimization about the code

Given the situation that there is quite a large chance we need to run the code on different organizers, each time we have to assign different serial ports with the same electronic devices, it will slow down the experimental process, and meanwhile increase the human error. So my plan for this week is to let the PC recognize what has been connected automatically and assign specific port to the devices according to our needs.

My solution is first to design a function that can read all the devices connected with PC via UART protocol so that influence of other devices like U disk or our phone can be eliminated. Because in the current stage, we have to connect 3 IMUs with the PC, the concern is that even though the PC can read the name of the devices connected and their port occupied, since the name of IMUs are exactly the same, it’s unlikely for the PC to distinguish which one is base and which one is arm end effector, so we need to request users to connect to the base IMU first, after the serial port connection of base has been set up, the other devices can be connected with random sequence. However, soon I found another issue, even for the designer myself, I found it is quite inconvenient to plug out all the devices before a new experiment and plug them in with a specific order for each time; in the real experiment, it is of great possibility that if we have finished the set up for the first time, we don’t have to make some modifications any more. Regarding this limitation, I have designed another function, to compare if the mac address will match for the current and previous experiment. The previous data including the information of the PC will be stored in the “Results” folder, then the code will search in this folder path and read the names of all the mat files, and sort them in the order of the date they were created, the latest one will be our target. Using the information inside, we can confirm if we are using the same PC as previously, if so, there is no need for us to reset all the settings.

However, there is much more to consider about, for example, we have to think about the case that when the number of devices connected to the PC currently is not identical to that previously, the algorithm for it can be extremely complex, but I think it’s still worth to do that because in the future we might have more sensors. And I will make a flow chart when I complete and verify all my code in the near future to let the users have a better idea about its utilization.

So far, I have completed all the features and functions that I mentioned above, which is a huge project, the coding work is at least three times than previously. After a series tests this afternoon, there are no issues for the first time set up, but it indicated that the devices were not connected even though I didn’t plug a single of them out for the subsequent experiments on the same PC. And some of the indications in the command window of MATLAB performed differently on different PCs, not because the different port numbers the devices will be assigned, I guess it is related to the RAM (several weeks ago I thought it was the limitation of MATLAB version).