Handbook for Matlab GUI
Ranging between two devices

PABLO DEL HOYO RODRÍGUEZ
Technische Universität Hamburg-Harburg
October 30, 2016

Abstract

This document presents the detailed information about the program used to range between two Pozyx devices. The program consists of a Graphical User Interface (GUI) for Matlab which easily ranges and set a real distance between two Pozyx devices. The GUI provides the functionalities to restrict the time or the number of samples. Also, it plots the error distribution and its main data.

Keywords: Matlab, GUI, Arduino, Pozyx, Ranging

Contents

1 Prerequisites

As the program will provide a communication between the Arduino board and the computer, two parts will be considered in order to make the installation the Matlab side and the Arduino side:

Figure 1: Both sides of the program

For the Matlab site it is required:

- Matlab R2016a (older versions may be valid)
- Arduino Support from Matlab: package to communicate with an Arduino. The installation steps are available at reference [1] and is usually installed by double clicking the package download file.

For the Arduino side is required:

- Arduino Uno x1
- USB-B Cable x1
- Pozyx Shield for Arduino
- The Arduino IDE and Pozyx Arduino Library: the installation steps are provided at reference [2]
- Pozyx Anchor x1 (or another Pozyx Shield)
- USB-C Cable x1 (or another USB-B Cable and Arduino UNO)

2 Script Files

The ranging.zip file is provided with the following files:

File	Description
HandbookRange.pdf	This file, description and a tutorial of the program
read.m	Main Matlab file. To be executed
read.fig	Matlab Figure for the GUI
ready_to_range_arduino.ino	Scketch to run onto the Arduino, do the ranging See section 3 for more details
(subfolder) functions	Some Matlab functions including these below
changeVisibility.m	Hide buttons to prevent some errors
checkIfFinished.m	Handles a flag to stop the ranging function
initArduino.m	Initialize serial communication Matlab-Arduino
updatePorts.m	Update the communication ports available (USB)

Table 1: List of files.

3 Arduino Setup

Before running the main Matlab file (*read.m*), some restrictions about the scenario have to be taken into consideration:

- First and most important, the identifier of the destination device must be specified in the sketch *ready_to_range_arduino.ino* as well as the UWB channel (1,2,3,4,5 or 7). This can be done by editing the following lines:

```
// Edit with your scenario:
uint16_t destination_id = 0x6670;
int uwb_channel = 1;
```

- There is a minimum separation distance that allows the ranging. Apparently, distances lower than 20 mm provides always a zero estimator.
- As there is some processing time in order to present all the information, **there is a delay between the current data being read and that one being represented**. So the ranging results are not strictly represented in real time. However, all the data is considered when the execution of the code is finished.

4 Overview

Once all the previous sections have been explained, the procedure to run the interface is described. The file localize.m has to be opened (by double-clicking or with the Matlab interface). After setting the path to the one corresponding to this project, the file can be executed (F5). This will open the interface, whose appearance is shown in the screenshot below:

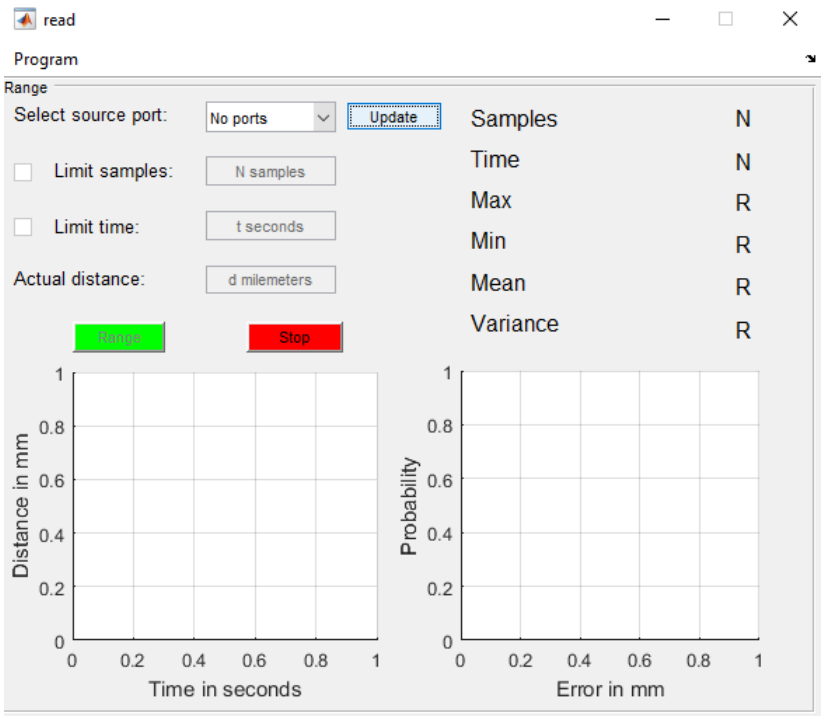

Figure 2: The interface of the program running on Windows 10.

The next sections will indicate the functionalities of the program in detail

5 Control

This section explain all the possibilities for the problem setup:

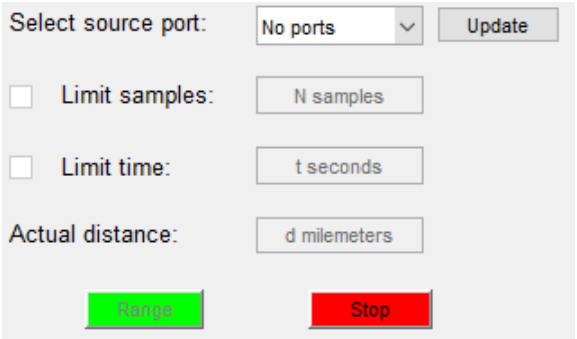

Figure 3: The control panel.

- The list of ports pop-up: indicates the currently connected COM ports. In case that nothing is connected, the execution of the code is not permitted.

Please make sure that the selected COM port is the one corresponding to the SOURCE Arduino running the *ready_to_range_arduino.ino sketch*.

- The update button: as its name indicates, updates the available ports and enable the buttons if at least one is connected:

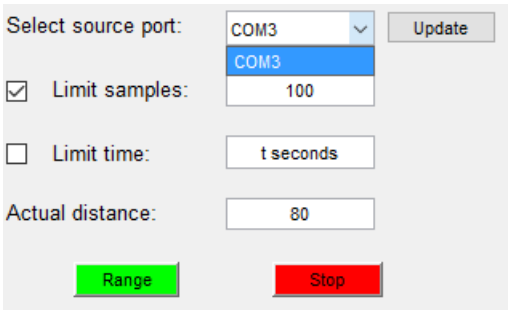

Figure 4: Setting the port after updating the list.

- n samples text box: if it is checked, this sets the number of samples that will be used for the ranging. **Please make sure that only positive integers are considered**. Other values may arise unconsidered errors. Also notice that larger values produces longer calculation times and larger delays between the current ranging and the one being represented.
- t time text box: if it is checked, this sets the number of samples that will be used for the ranging. **Please make sure that only positive numbers are considered**. Other values may arise unconsidered errors. Also notice that larger values produces longer calculation times and larger delays between the current ranging and the one being represented.
- actual distance text box: this will set the real distance between the devices. See section 6 for more details. This is the parameter $d(t)$.
- the range button: it sends to the tag a command to range to de destination device, shwoing the information as soon as it is processed. If is limited by the number of samples or by the time (or both), the current progress will be displayed. If not, it is required to press the stop button. Once the ranging finishes, a matrix containing n samples with the number of sample, the time in ms and the distance in mm is printed.
- the stop button: interrupts the execution of the code, for example in the case a very large number of samples was set and the process is wanted to be closed.

6 Results

The left plot shows the evolution of the distance over time, defined as $\hat{d}(t)$. Thus, the green graph represents the distance between the tag a the destination calculated by the Pozyx:

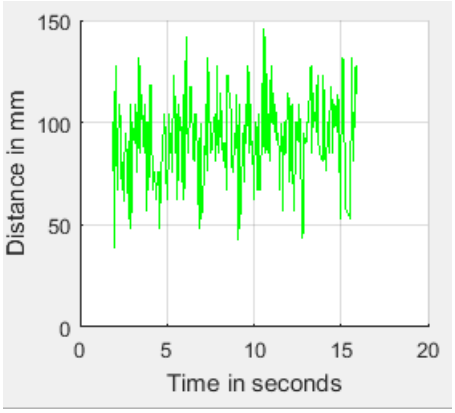

Figure 5: Distance $d(t)$ over time.

The right plot represents the histogram of the error distance defined as $e(t) = \hat{d}(t) - d(t)$. So the blue bars represent the number of appearances for each error:

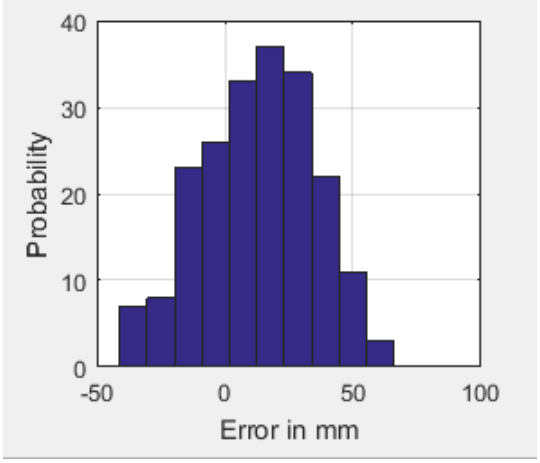

Figure 6: Error $e(t)$ over time. Seems to be Gaussian.

Finally the rigth data displays the most relevant information of the error distribution, including the mean and the variance:

Samples	100
Time	8.904
Max	146
Min	29
Mean	78.35
Variance	350.634

Figure 7: Data of error $e(t)$ over time. Seems to be Gaussian.

7 Future Work

Once the program has been presented, it has been observed that many functionalities can be implemented with the interface.

It would be interesting to provide a tool to manually set the destination device id. However, it was difficult to do it as the transmitted data is hexadecimal throughout a serial bus, interpreting the Arduino sketch the wrong id.

Also, some synchronization issues have been observed, as the data is read faster in some computers. If there is no data to be read, the code waits a time until it is again available, defining a buffer. However, it has been observed some unsuitability with these waiting times. Improve this buffer to work correctly in every scenario and computer is an important issue to be solved.

References

[1] Arduino support from matlab. `https://de.mathworks.com/hardware-support/arduino-matlab.html`. Accessed: 7/10/2016.

[2] Arduino getting started. `https://www.pozyx.io/Documentation/Tutorials/gettingStarted`. Accessed: 7/10/2016.