Broadband Radar Software Development

Quentin de Saint-Christophe

September 2019

1 Introduction

The internship has been realised in the university of TUSUR, Tomsk in Russia, it is an entrepreneurial research university specialised in Radio Engineering and Electric Radio Control. The aim of the work is about designing a reliable and portable interface for a radar. The device studied is a 4G Broadband Radar from the Simrad industries owned by the Department of Design and Production of Radioelectronic Equipment, it can be directly used with a chartplotter or on a computer with the software Navico. The purpose of this study is to create a light software than can be embedded to use the radar outside in real conditions. The program operates under any UNIX or Microsoft operating system and is coded in Python for its large choice of libraries and its code reliability. It is divided in three main parts: the computer-device connection, the display of frames on an user interface and the signal processing code.

2 Radar equipment and connection principle

2.1 The Network

The radar is bound to the display unit with different interfaces, all these interfaces have their role and cannot substitute each other. Three routes are mainly used; the first one establishes a link with the display unit — the computer, an other one receives instructions from the computer and the last one will send information about the radiolocations from the field analysis.

2.2 Establishing the connection to the radar

The radar is bound to the display unit with different interfaces, all these interfaces have their role and cannot substitute each other. Three routes are mainly used; the first one establishes a link with the display unit — the computer, an other one receives instructions from the computer and the last one will send information about the radiolocations from the field analysis.

The interface that establishes automatically first the connection to the display unit is held by the 236.6.7.9:6679 address. In our case the multicast address

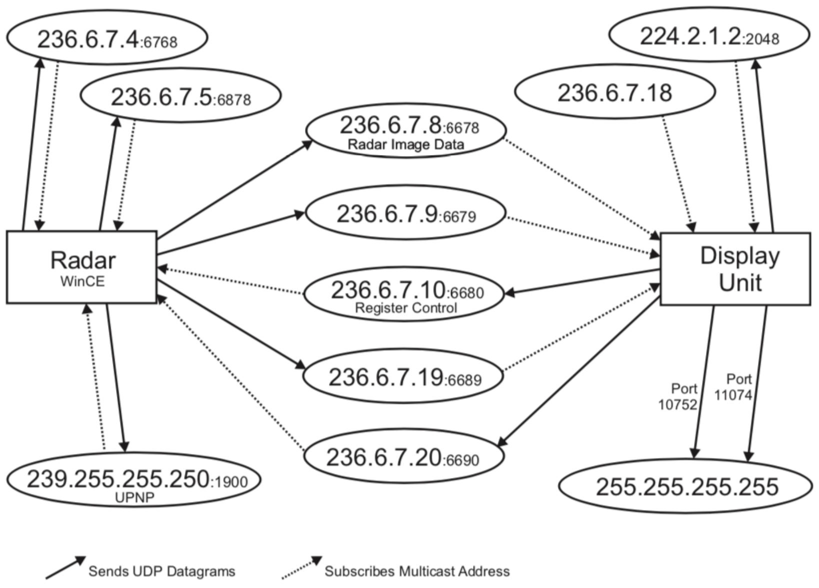

Figure 1: Structure of the interface

is 236.6.7.8:6678, here the computer listen to the radar; and the data-sending address is 236.6.7.10:6680, there the computer sends registers to the device in order to monitor the operating parameters.

The most powerful and efficient way to communicate with the device is to use sockets. The protocol used is IGMP, an IP protocol but with an IGMP header that have to be capped by the routers in order to set multicast groups. The transport layer is as a consequence following the UDP protocol for the data delivering is made on multicast group and for the devices have only to listen on a particular address and interface.

Here, the main advantage of the tandem UDP/IGMP is that the device has nor to relay lost packets nor to setup any connections. It is very useful because it involves no delay — a radar is quite delay-sensitive and needs a constant data flow, and for the router do not have to keep sending and relaying tracks for multiple receivers — unlike in TCP.

A socket is a mean of communication that requires a specific port and IP-address. To bound two devices, both interface has to create a socket to read and write to each other. In our case, the radar is natively deploying sockets that work with UDP protocol.

The creation of the listening socket counts three steps:
• creation of an UDP socket and set the ability of reusing any address on these listening interface.

• binding the socket as a server socket on the multicast address and port to beckon any entering connection.
• declare this multicast address as a new member of the group in order to reckon this one in the routing IP-interface.

The creation of the sending socket is trivial for the connection has already been automatically done by the radar and for the listening socket of our display unit is sending acknowledgments to the device.

2.3 Sending datagrams under the shape of registers to control the radar activity

All the registers values are set in the hexadecimal base

To communicate with the radar, the display unit sends registers with the values of the range, the filters, the focus, et cætera... The first number of the register is its number, the second is always C1 — it works like a checksum — and then commences the parameters' setting:

00 — turn on/off	1 byte	on/off
01 — turn on/off	1 byte	on/off
03 — zoom level	4 bytes	sixteen different ranges in hectometers
06 — filters and preprocessing	9 bytes	five different filters
08 — interference rejection	1 byte	off/low/medium/high
0A — target boost	1 byte	low: 24rpm/medium: 36rpm/high: 48rpm
0E — local interference filter	1 byte	off/low/medium/high
0F — scan speed	1 byte	off/low/high
A0 —operation maintain	1 byte	—

Here are the tables to convert the parameters in bytes, they are always precede by the register number and the checksum.

• registers 00 and 01:

turn on	1
turn off	0

• register 03

0.5	F4—1—0—0	20	20—4E—0—0
0.75	EE—2—0—0	30	30—75—0—0
1	EE—3—0—0	40	40—9C—0—0
2.5	C4—9—0—0	60	60—EA—0—0
5	88—13—0—0	80	80—38—1—0
7.5	4C—1D—0—0	120	C0—D4—1—0
10	10—27—0—0	160	00—71—2—0
15	98—3A—0—0	240	80—A9—3—0

Distances are in hectometers.

• register 06

automatic gain	0—0—0—0—1—0—0—0—A1
manual gain	0—0—0—0—0—0—0—0—X
rain clutter filter	4—0—0—0—0—0—0—0—Y
sea clutter harbour automatic	2—0—0—0—1—0—0—0—D3
sea clutter manual	2—0—0—0—0—0—0—0—Z

The values X, Y and Z are gain variables that go from 1 to 50.
There is no automatic rain clutter, as a consequence, the default value of Y has to be set to 4D.

- register 08

off	0	medium	2
low	1	high	3

- register 0A

off	0
low	1
high	2

- register 0E

off	0	medium	2
low	1	high	3

- register 0F

low	0
medium	1
high	2

- register A0

operation maintain	2

The operation maintain value has to be set each ten seconds to keep the radar working.

2.4 Receiving datagrams and displaying the field analysis

The radar is sending datagrams containing 2048 scanlines, each of 512 pixels.

The first scanline is horizontal and represent the first radius oriented with zero radian, the second one is oriented in the anti trigonometric direction of $\frac{2\pi}{2048}$ and this way on and on until all scanlines fill the complete circle.

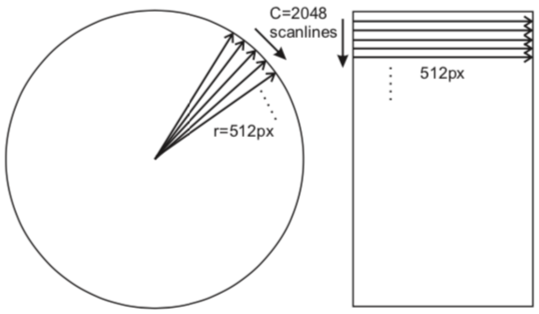

Figure 2: Scanlines spread beside the radius

3 Architecture of the main processus

3.1 Communication and displaying on the CPU unit

The code is mainly separated in five sheets. The core sheet — *MainLoop.py* — is calling functions in the four other sheets, namely *RadarRender.py*, *RadarListener.py*, *RadarDecoder.py* and *SecurityZone.py*. The computing speed of the unit is greater than the radar frame rendering, the communication, and the decoding speed, this is the reason why the core sheet calls in the same process the three first sheet's functions.

The core sheet starts by initialising the variables and calling the class objects of the other sheets, then the main loop commences and lasts while the radar is still listening. The loop is split in four main parts : the first one consists in gathering the data given by the radar and stocking them, subsequently is built a frame pack that will be used to monitor the security zone and processed by a forked son. Later comes the display of the general user interface — GUI — and and lastly the handling of the events — mouse and keyboard interaction.

3.2 Forking processes for parallel calculations

To monitor the security zone the *SecurityZone.py* sheet is processing images, which takes a lot of resources and time to the CPU, the program is thus multiprocessed in order to use the maximum of the CPU resources. As far as the

Broadband Radar is operating fairly quick and as it is communicating with a very few delay thanks to the multicast protocol, it would have been underproductive to dwindle on efficiency: as a consequence, the core loop is forking to call the functions of the *SecurityZone.py* sheet in a son process.

4 Image processing and moving objects detection algorithm

As written before, the image processing part is realised by a son process in order to perform asynchronous multiprocessing and have better results. To prevent a zone from being entered the father process gathers a series of eight frames and send them under the shape of a matrix to a new son, created by the *multiprocessing.Process*() function. Then the son process computes the mean squared error and detects any moving object by the mean of a hysteresis thresholding. The principle of the thresholding consists of classifying the values of the matrix — the pixels of the frame — in two categories: the ones which are greater than the upper threshold and the ones that are lower than the lesser threshold. Those values are changed to, respectively, 0 or 255.

For all the remaining values that are between the two thresholds, a comparison is made with the direct eight nearby pixels, and according to the already-classified neighbour's values, they will change their values by matching with the more appropriated one: 0 or 255. This hysteresis process spreads a homogeneous way a trend in a matrix and stains it with convex shapes, like ships or beacons.

Finally, the coordinates of the moving pixels located within the radius of the security zone are returned to the father process by the mean of pipes, created by the *multiprocessing.Pipe*() function. All the sibling-workers are programmed for auto-destruction to free the resources after computation.

5 User Interface

The user interface is made with the *pygame* library in a standing alone *window*() because of her efficient and quick method of displaying pixels — *window.blit*().

5.1 Changing the radar's settings

This interface offers the choice of adjusting the range of the radar by interacting with the up and down arrows or directly clicking on buttons provided on the upper left-hand corner of the window. By pressing the s, q or x keys, the speed of the radar can be modified — 24, 36 or 48rpm, this functionality is on the upper right-hand corner. On the upper right-hand corner, by entering a numerical value lower than the range of the radar, the user can display a circle of a chosen value on the dial. He can also decide to turn on the security monitoring by switching the ON/OFF button and directly change the security zone radius in the box provided.

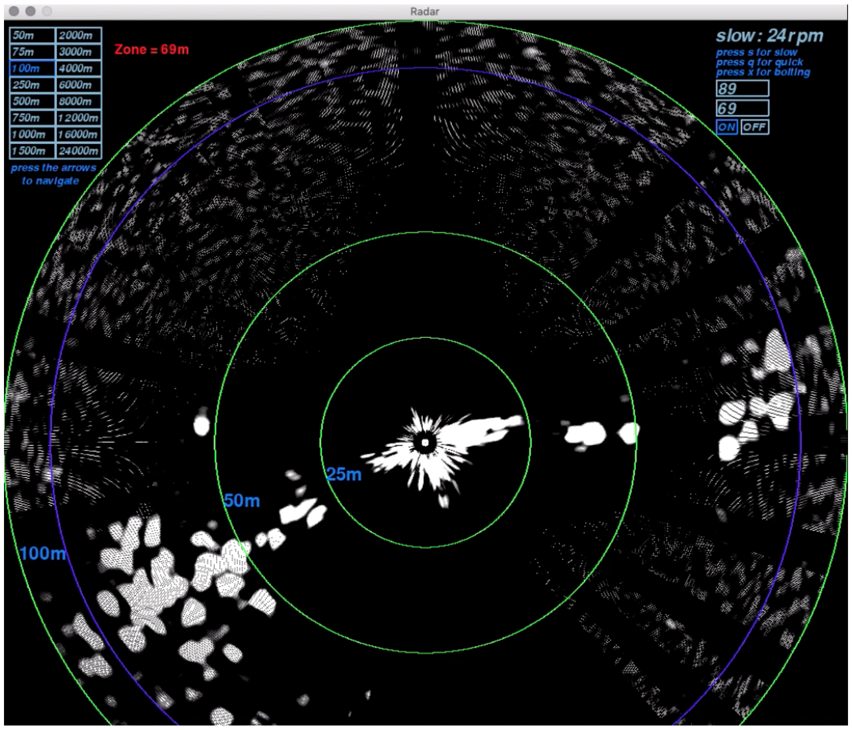

Figure 3: A chosen radius displayed in blue with the moving objected detected in red

5.2 User direct measurements

The mouse always displays the range in meters from the center; and to compute distances between two objects, the user can simply drag the mouse on the dial and watch it upon the screen. For better comfort, the security zone radius is always displayed and the user can, by holding the space bar, zoom on the values of the dial and bold its range rings. The static objects are displayed on the dial by levels of gray and moving detected objects are displayed in red.

5.3 User recommendations about the security zone

To reconnoitre moving objects, the algorithm must base its calculations on a series of eight frames. If the range of the radar is changed while the security zone is controlled, the detection of moving objects could be deteriorated. As a con-

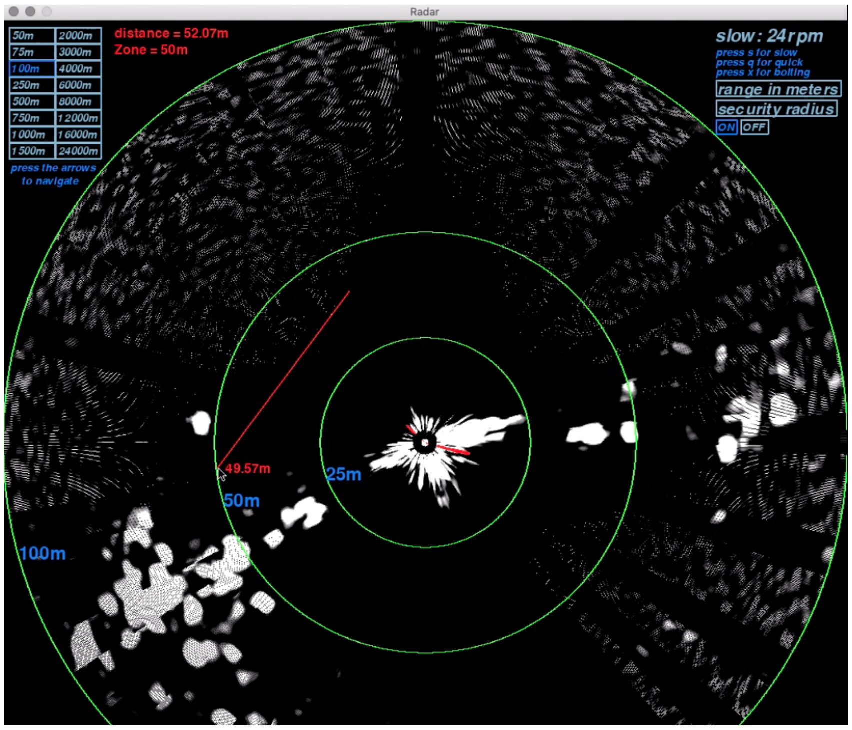

Figure 4: Dragged distance displaying with the moving shapes detected in red

sequence, the user is beckoned to take care of his adjustments while monitoring a security zone.

6 Real Condition Tests on Tom River

The interface has firstly been tested in real conditions upon the water in the Tom river. The program was not finished, it was to check the reliability and the efficiency of the frame rendering.

The cruise has been done on a motor boat in order to take advantage on more manoeuvrability. The radar was set on the top of the ship, its basic position. (Figure 5)

On the Tom river, many motor or sailing boat are cruising. There are also a lot of rusty barges and gravel trawlers. (Figure 5)

The radar operated and performed especially on quickness, it was connected to a computer: here are some views from inside the boat. (Figure 7)

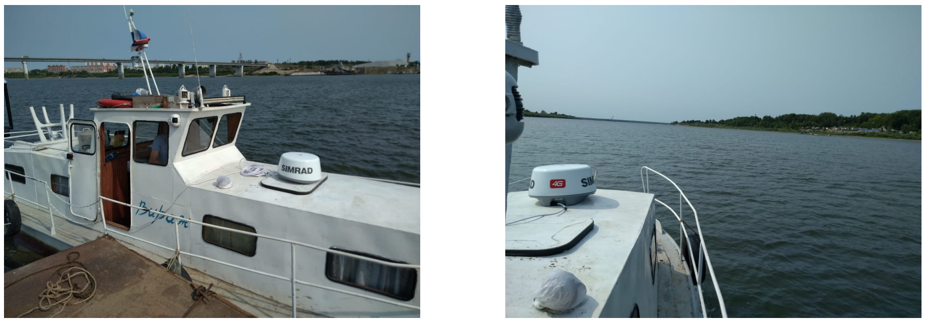

Figure 5: On the left, the motor testing boat. — On the left, the radar's position of listening.

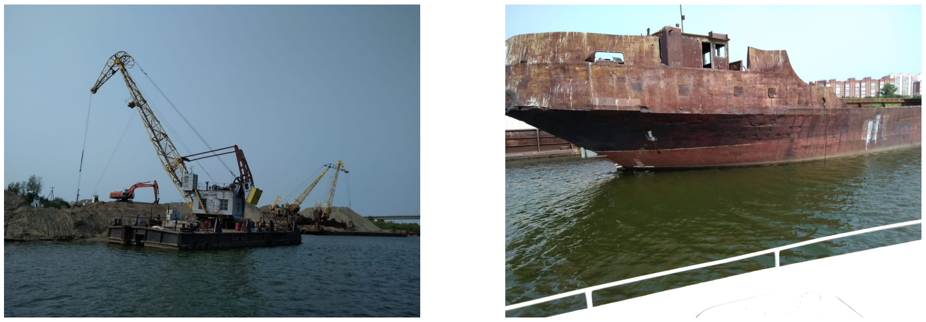

Figure 6: On the left, a gravel trawler working. — On the left, an obsolete rusty barge.

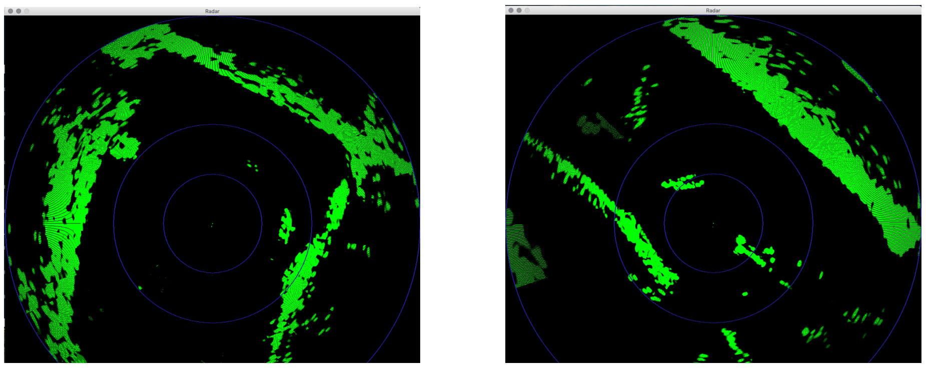

Figure 7: On the left, a bridge is detected above the radar: the shape at right angle of the shores. A gravel trawler can be reconnoitred. — On the left, the river course is flowing along the the two riversides, upon the water, three different barges are detected.

Figure 8: Mr Igor Artemov and I during the experiences.

7 Conclusion

The main challenge of this work was to make compromises with the used libraries. Actually, the most efficient library to make asynchronous multiprocessing does not exist under any Microsoft OS and also, under both Microsoft OS and Mac OS, the general user interface cannot be modified or called by son processes. Working under Linux would have been, a posteriori, a better mean to properly make multiprocessed displaying. With these difficulties, I had to learn a lot of new libraries and functions and it took me a lot of time, time I could have spent testing the radar many times in real conditions, upon water. Despite these details, a reliable and functional software has been developed under Windows 10, it can be used in real conditions for field analysis and for detecting moving objects while the ship remains static.

8 Acknowledgements

I will particularly thank Mr Igor Artemov for his technical and wise guidance and suggestions, Mme Maria Afanasyeva for her daily support and advice, and last but not least, Mr Rivet for giving me the opportunity of doing this internship in Russia with the university of TUSUR.