D4 Design Project SPECIES Report

Yubo Zhi
yz39g13@soton.ac.uk
Personal Tutor: Professor Alun S Vaughan

Abstract: The objective of this project was to design a secure and mobile communication system in limited time and resources. The design was done by group of 6 students working as a team in 12 days, competing with 10 other teams. The system must be able to transmit human voice securely over short range wireless transmission, but could have any kinds of extensions as well. The prototype our team built utilities 2 Il Mattos, using high speed UART to communicate. The first one was for control and user friendly GUI with touchscreen, the second one was for wireless data communication, audio input and output. The device was able to transmit and receive human voice from microphone to speaker at the opponent end like a walkie talkie, simultaneous and shared sketching, send and receive text messages.

1. Contribution

Favour in programming and familiar with C/C++ programming with the AVR and TFT display, I decided to do the programming part of Il Matto 1 including UART communication protocol between Il Matto 1 and Il Matto 2 to control audio transmission, and transmit varies type of data. The programming part include resistive touchscreen interface and GUI design, which were totally optional from the compulsory audio transmission of the project, but could add great features to our device. Diwen Hu (dh1g14) was helping me at overall GUI design, but not on the code side since he do not know much about C/C++ programming.

At the end of project, what I achieved were:

1) Menu selection GUI with multi-level submenus and scroll ability.
2) Resistive touchscreen working, and able to scroll and activate the menu items.
3) Resistive touchscreen sketch on TFT with full scale colour selection and 9 different brush sizes.
4) Lower case letter and punctuation text input through swiping on virtual keypad.
5) Easy and successful integration with Il Matto 2 through doubly buffered UART protocol with acknowledge.
6) Touch button controls Il Matto 2 audio transmission.
7) Ability to instantly transmit sketch to either side by half duplex.
8) Ability to send text messages and queued at the other side to ensure no messages loss.

2. Specification

The part I were responsible for is the direct interface with user, therefore an easy to use interface would be essential, so TFT display with resistive touchscreen for GUI interface was determined. A user authorisation method would be on the GUI side as well.

The UART protocol used to transfer data packages between Il Matto 1 and Il Matto 2 was using a baud rate of 750kbps, which was quite high. Therefore in order to ensure no data package could got lost, a communication protocol using interrupts and double buffer with acknowledge from opponent side was required.

3. Design & Simulation

3.1 GUI design

The microcontroller used is 8bit AVR core ATmega644P, with Il Matto board. The TFT display is using ili9341 controller with a resolution of 320x240, connected to the Il Matto board through 8bit parallel interface with control pins at PORTA, data port at PORTC.

By looking at datasheet [1] section 9.2.2 Vertical Scroll Mode, a menu selection with scroll ability were decided to be designed as primary GUI entry.

The vertical scroll mode shown by Figure 1 was the base of fast scroll display of menu items.

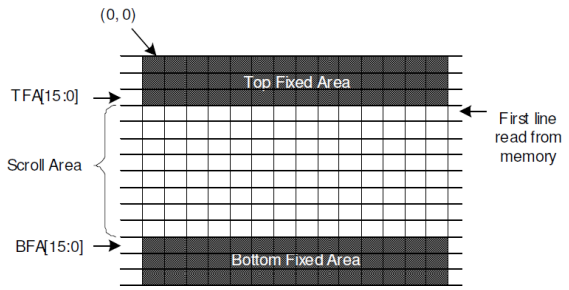


Figure 1: TFT Vertical scroll mode specification (adapted from [1])

Since the scroll mode in built into the TFT controller, by sending a 3 bytes command it was very easy to scroll

the screen area. But there were some limitations, the scroll mode only apply to vertical orientation, and do not translate the coordinate from select area command, it only changes the line read from frame memory when the controller refreshing the TFT. Thus coordinate translation must be implemented from the programming side (implemented in tft.cpp from libtft-cpp [6]).

The difficulties were, functions for drawing rectangle and characters must be modified in order to mask out displays above the top fixed area and below the bottom fixed area, also continue drawing at the original bottom edge coordinate, which now may become the middle of display, and need to continue drawing from the original top edge coordinate, as illustrated by Figure 2.

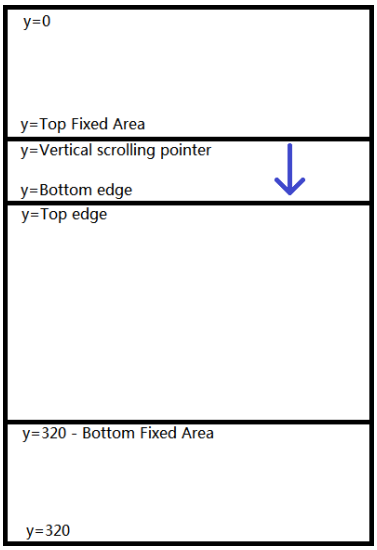


Figure 2: Example vertical scroll mode coordinates

Originally landscape TFT orientation was decided, the menu item selection GUI designed as shown by photo 3.

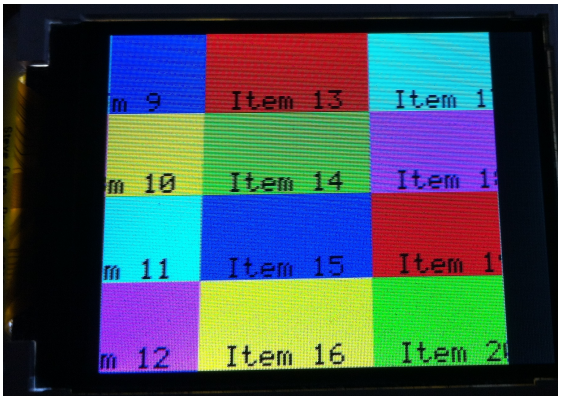


Figure 3: Menu selection GUI at landscape orientation

But by constrains of touchscreen, the orientation later changed to portrait, which actually gave a better looking, also the coding of the menu selection became easier. Figure 4 shows screen capture of the final product root menu at portrait. The final code was packed as liblist and available at [8].

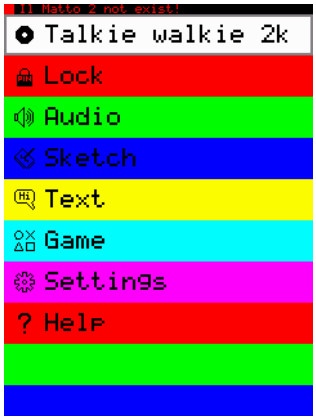


Figure 4: Menu selection GUI at portrait orientation

3.2 Resistive touchscreen design

The touchscreen used in our device is a 4-wire resistive touchscreen, it has 2 thin film with resistive coating, when pressure is applied to the touchscreen, the two layer contacts as demostrated in Figure 5.

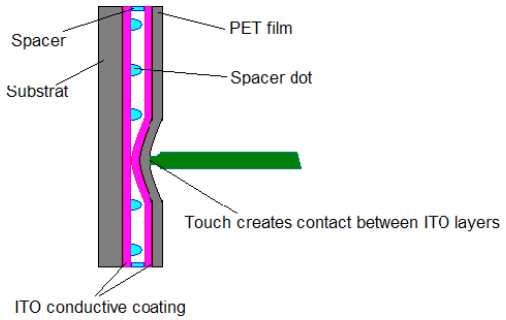


Figure 5: Resistive touchscreens (adapted from [2])

The touchscreen can be modelled as shown by Figure 6, when pressure applied the resistor R_{Touch} would connect the top and bottom resistive layer.

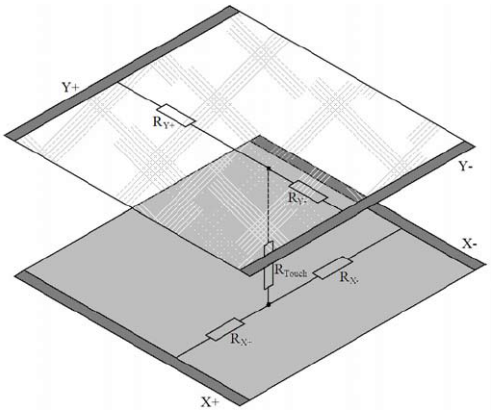


Figure 6: ”Schematic” of a 4-wire touch screen when pressure is applied (adapted from [2])

When detecting touchscreen press, connect Y layer to V_{SS}, attach a pull-up resistor at X+, than read the digital value of X+. If the touchscreen is pressed, X and Y layer connected, thus X+ would be pulled to V_{SS}.

When reading X value, apply V_{CC} and V_{SS} across X+ and X- terminals, put Y- terminal in high-Z mode, than read the voltage on Y+ terminal using ADC. The resistance of Y layer and R_{Touch} could be ignored due to input impedance of ADC was very high ($100M\Omega$ on ATmega644P). By assuming the X layer was linear about resistance and x-coordinate, the X value therefore was the value output from the potential divider formed by R_{X+} and R_{X-}, which was the voltage read from ADC directly.

Y value could be read by using the same method, apply V_{CC} and V_{SS} across Y+ and Y- terminals than read the voltage divided by R_{Y+} and R_{Y-}.

In summary, by use the methods described in Table 1, the microcontroller could interfacing with the touchscreen with 2 ADC channels.

Table 1: Touchscreen interfacing (adapted from [3])

Function	X+	Y+	X-	Y-
Detection	Digital input with pull-up	Open	Open	V_{SS}
Read X	V_{CC}	ADC1	V_{SS}	Open
Read Y	ADC2	V_{CC}	Open	V_{SS}

After read ADC data from touchscreen, it was necessary to map the data to the coordinate system of the display. The method of calibration can be found at [4], it was actually an image move and scale algorithm described using matrices. But bother to implement all the matrix operations, the calibration code was directly copied from an open source touchscreen library[5].

The resistive 4-wire touchscreens ordered were 4.36in, which were even larger than two TFT displays with a size of 2.2in, the extra spaces could be used for other purposes.

Originally landscape orientation was chosen, the extra area of touchscreen would be used for permanent buttons, as illustrated by Figure 7.

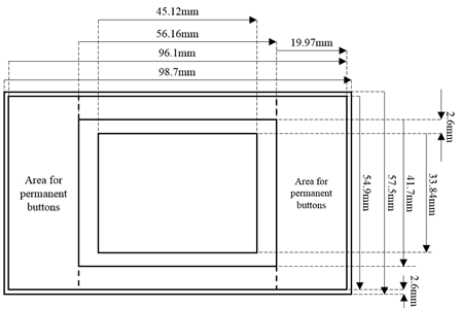


Figure 7: Original landscape orientation layout (produced by ner1g13)

However, the final orientation of the device decided to be portrait, with this arraignment, TFT display was places at the right part with portrait orientation of the touchscreen, while the left part of the touchscreen was used for virtual keypad with 4x4 buttons or colour and brush selection at sketch mode, as demonstrated in Figure 8. Moreover, the device in portrait orientation results in a better look, and also made coding for TFT scrolling mode a lot easier.

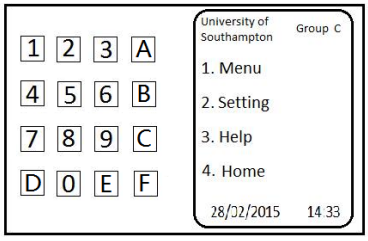


Figure 8: Portrait touchscreen layout (produced by dh1g14)

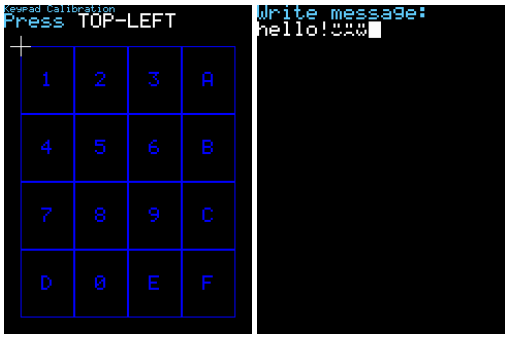
The calibration data were stored in the EEPROM built inside ATmega644P once calibration completed. Thus calibration process would not needed every time start up. If calibration failed, another mechanism also implemented. Recalibration will occurred when user power up the device while pressing the touchscreen, so that a re-flash of EEPROM was no longer needed.

3.3 Sketch design

The sketch mode draw rectangles on TFT centered at touched point. The numbers 1-9 on the virtual keypad were used to select brush sizes, which were actually the size of rectangles drawn. Full scale brush colour selection was done by swipe at the virtual keypad region as shown by Figure 10. A second indicator located at top left indicates current brush size and colour. Figure 9 demostrated the sketch mode. Source code for sketch mode were at software listing 10 and 11 with group report.

Figure 9: Screen capture of sketch mode

Figure 10: keypad drawings (produced by ner1g13)



(a) Keypad calibration (b) Message writing

Figure 12: Text input GUI

In shared sketching mode, the sketch data was buffered and arranged into variable length packages then send to the other end wirelessly through Il Matto 2. The package structure will be described later in UART protocol section, the package data for shared sketching mode was arranged as following:

1) Type (1 byte), which is PKG_TYPE_SKETCH.
2) Size (1 byte), the brush size.
3) Colour (2 bytes), colour used throughout this package.
4) Coordinates (4 bytes array), unsigned 2 bytes x-coordinate followed by unsigned 2 bytes y-coordinate.

3.4 Pin authorisation design

The authorisation method designed was a typical 4 digits PIN passcode, input from the virtual keypad. A new passcode will be asked at first time EEPROM initialisation, and stored inside EEPROM, so would not lost after power down. The PIN can be later reseted at settings menu, but the original passcode will be required, a menu item also added to lock the device. No timeout or retry limits implemented currently. Data package were also handled while PIN locking, but notifications were not shown until device unlocked. Figure 11 demonstrated the locked state.

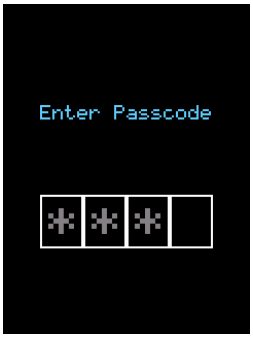


Figure 11: PIN locked state screen capture

3.5 Text input design

The left extra area of touchscreen was used as virtual keypad, by attaching a piece of paper with a keypad printed on it, and a simple calibration procedure shown in Figure 12a.

For full alphabetic and punctuation input, swipe input was implemented, so that every key on keypad could have 5 characters assigned, press and swipe at four directions. The 'F' key was implemented as backspace. There were still unassigned space left after assigned all punctuations on a standard keyboard, so 3 emoji were added to TFT library and assigned to 'A', 'B' and 'C' keys. The detailed assignments could be seen from Figure 10. Figure 12b shows the message writing GUI.

The text message package was designed as follows:

1) Type (1 byte), which is PKG_TYPE_TEXT.
2) Index (1 byte), for acknowledge and existing check.
3) String (char *), the actual string data.

3.6 Audio controller design

The GUI for control audio transmission and receiving is simply a big press-to-talk button, as shown in Figure 13.

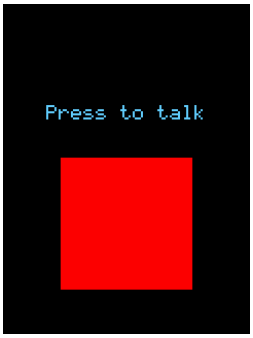
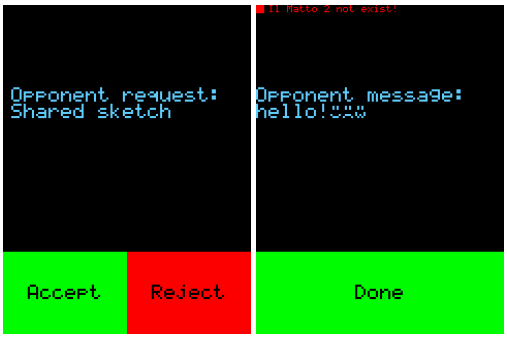


Figure 13: Audio controller GUI

After pressed or released the button, Il Matto 1 send a command to Il Matto 2 to enable or disable audio transmit.

3.7 Notification system design

Notification system was designed to handle incoming messages and requests. It store the messages and requests by first-in-first-out, than displayed when the user is free, e.g. at menu mode. Example incoming request and text message GUI shown in Figure 14. Source code listed at 6 and 7 attached with group report.



(a) Sketch request (b) Text message

Figure 14: Notification system GUI

3.8 UART communication protocol design

UART0 on the ATmega644P microcontroller was used for communication between Il Matto 1 and Il Matto 2. Interrupts and acknowledge used for lossless data transmit and receive. Double buffer was implemented, so that the communication could be full duplex, and microcontroller could be dealing with one package while the UART interrupts were transmitting another one to improve time slot efficiency. Source code listed at 14 and 15 attached with group report.

Data was packaged with a maximum size of 64 bytes, detailed as follows:

1) Command (1 byte), the required operation. Commands without COM_DATA macro does not have data, length byte would not transmitted as well.
2) Length (1 byte).
3) Data (Maximum 64 bytes).

The commands were defined in 'communcation.h' header file, listed at 33 attached with group report.

To help with coding, flow chats were drawn, attached at Appendix A and B.

4. Testing & Results

Since the entire GUI is coded by one person only, the integration of modules for GUI was done at design stage, working together without any problem.

4.1 Menu selection GUI

The menu selection GUI involves a complex coordinate translate and drawing algorithm, was the most buggy part, but all solved. It was tested by some different ways, scroll quickly tests for large area refresh, scroll slowly tests for small area refresh at the edge and changing top and bottom fixed area tests coordinate transform. All of them worked as expected.

4.2 Touchscreen

Originally landscape orientation was chosen, after implemented some code for read and calibrate touchscreen [7], the touchscreen was tested by plotting points onto TFT, however the points plotted were very noisy as shown by photo 15.

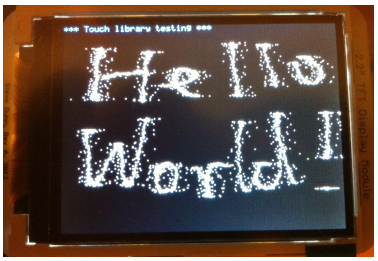


Figure 15: Touchscreen noisy when placed on top of TFT

The touchscreen noise reduced dramatically while holding up, therefore the final orientation of the device decided to be portrait, using the box for Il Matto to support it underneath the edge, which would gave some space between the touchscreen and TFT. Later changed to a 3D printed case designed by dh1g14.

For reduce touchscreen noises further, a software ADC averager was implemented with glitch filter [7], result demonstrated by Figure 16.

Figure 16: Touchscreen sketch after software averager

Near the end of prototyping, one of the touchscreen was damaged, produces glitches and panning in x direction, made sketch not understandable, causing difficulty in keypad input and pressing menu items. The other touchscreen has the same problem as well, but not as severe as the former one. This was probably because of inappropriate assembly and frequent removing of the touchscreen without enough care. The two thin resistive layer on top of the touchscreen split by a small space, twist of the touchscreen during removing would easily cause the top layer to distort, damage the entire touchscreen.

4.3 Communication

The UART communication protocol designed was the same for transmitter and receiver, thus could be easily tested by using two Il Matto 1 connected together, after implemented some basic commands like ping. A submenu with diagnosis function also implemented, mainly for testing UART communication, as shown in Figure 17. One Il Matto could be configured to pooling for data, another one send different data packages from shared sketch mode etc. to test data packaging functionality, or try to send lots of data to test for data loss.

Figure 17: Diagnosis menu

During test, some critical bugs found. The initial version wrote without utilise flow chart could not do full duplex, therefore the entire UART module almost rewrote. Another bug found was both end might waiting for acknowledge at the same time, completely blocked UART communication, was corrected in current version.

Current version of UART communication module worked without any problem. Double buffer, wait for acknowledge before send the next package all worked as expected. Also tested by connect two Il Matto 1, than using shared sketch mode, sketch at both side very quick at the same time, which would produce lots of data packages at both end quickly. None of either side ever stopped responding or lost a portion of sketch, indicates the UART module worked as expected.

4.4 Integration

The interface designed for UART module made integration with Il Matto 2 fairly easy. After implemented some essential commands like ping, send and receive data packages on Il Matto 2 within hours, both end communicated through wireless successfully. At short distance, sketch transmitted to either side instantly, but might lost when trying to sketch on both side at the same time, because the wireless module could not do full duplex transmission. At greater distance, sketch data start to loss. That was approximately 5 meters in the computer lab, improved a lot outside the computer lab. That could also be improved after implemented data acknowledge at Il Matto 2, which was not implemented because of time constrains.

The audio controller press-to-talk button worked as expected after implemented audio start and audio stop commands on Il Matto 2.

Text messages could be sent to either side without data loss or corruption, mainly because the data packages will be automatically resent if acknowledge not received within a period of time implemented on Il Matto 1.

Because of the acknowledge functionality of UART communication, if Il Matto 2 stopped responding, Il Matto 1 could also stuck. Generally the indicator on top left of Il Matto 1 TFT would turn to red indicates Il Matto 2 stopped responding. In this case, reset the whole system would bring them back to working state.

5. Team Working

The ideas of the device was collected firstly, so that everyone could choose what they were good at. That become more detailed after we got the block diagram of the device. Since we were 6 students, every 2 were assigned to a particular task. I were assigned to Il Matto 1 GUI design and UART communication with Diwen (dh1g14), while Alaa and Nathan assigned to analog stuff including microphone amplifier and loudspeaker amplifier, Joseph and Fiona were going to get wireless transmission module working.

Facebook group created at our first meeting, so in case anyone has any ideas or questions we can discuss it online. For example, when I found out landscape orientation was not working, I got some ideas from Facebook instantly.

Source code were shared and managed through SourceKettle. But only I and Nathan were using it, because of the complicated things to learn at first time.

The team were managed 'flatly', everyone were able to help others. Targets were requested by everyone from where they were good at to be most productive. Team leader was mainly care about progress, time and meeting arrangement.

Risks were planned as in Risk Management table. The only unexpected event would be the damage of touchscreen, but we could not do anything about it actually, we did not have enough budget to buy another touchscreen, the damaged touchscreen was actually still useable but just not stable.

6. Reflection

We planned SD card, data acknowledge through wireless communication and some other ideas, but had not got time to implement them. Text message implemented on the last day, without intensive test and debug.

References

[1] ILITEK "ILI9341 specification" [*Online*]. Available: https://www.adafruit.com/datasheets/ILI9341.pdf

[2] ATMEL "AVR341: Four and five-wire Touch Screen Controller" [*Online*]. Available: http://www.atmel.com/images/doc8091.pdf

[3] NXP "Interfacing 4-wire and 5-wire resistive touchscreens to the LPC247x" [*Online*]. Available: http://www.nxp.com/documents/application_note/AN10675.pdf

[4] Texas Instruments "Calibration in touch-screen systems" [*Online*]. Available: http://www.ti.com/lit/an/slyt277/slyt277.pdf

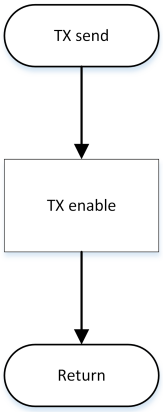
[5] K. Russell, L. Douglas, L. Chris "tslib-1.0" [*Online*]. Available: https://packages.debian.org/source/wheezy/tslib

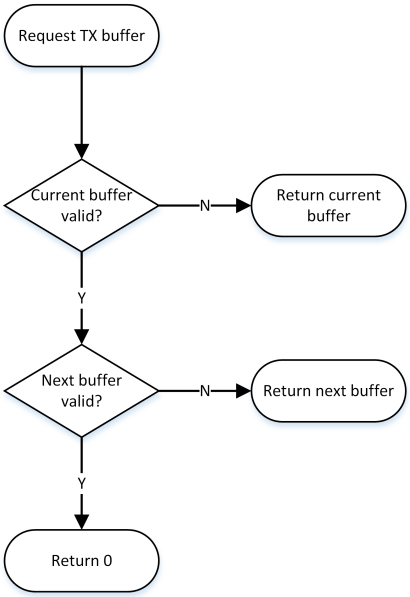
[6] Z. Yubo (yz39g13) "libtft-cpp" [*Online*]. Available: https://github.com/zhiyb/Il-Matto/tree/master/lib/libtft-cpp

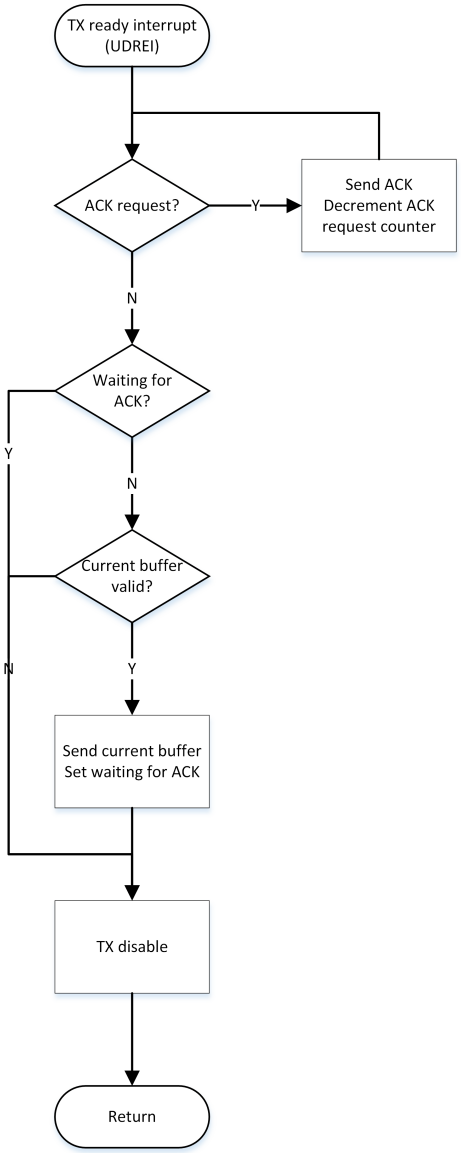
[7] Z. Yubo (yz39g13) "librTouch" [*Online*]. Available: https://github.com/zhiyb/Il-Matto/tree/master/lib/librTouch

[8] Z. Yubo (yz39g13) "liblist" [*Online*]. Available: https://github.com/zhiyb/Il-Matto/tree/master/lib/liblist

Appendix A: UART transmit flow chat







Appendix B: UART receive flow chat