CHAPTER 5

RESULT AND DISCUSSION

5.1 Result

From the display in user interface, user can get the image data in both the binary and ASCII code form, the user will be able to use the image data and process it into image by mapping the pixels into its own camera color coding. The data from the camera were compared both in ASCII form and binary form to ensure data that were received are valid and at the same time it is for the user convenience, as the user do not need to convert it to back to binary form when the user want to process the data into image.

To compare the data, an ASCII code table is needed to make sure the result is accurate. For example, the figure below, Figure 5.1 shows the data that was gathered from the camera and displayed it into the user interface.

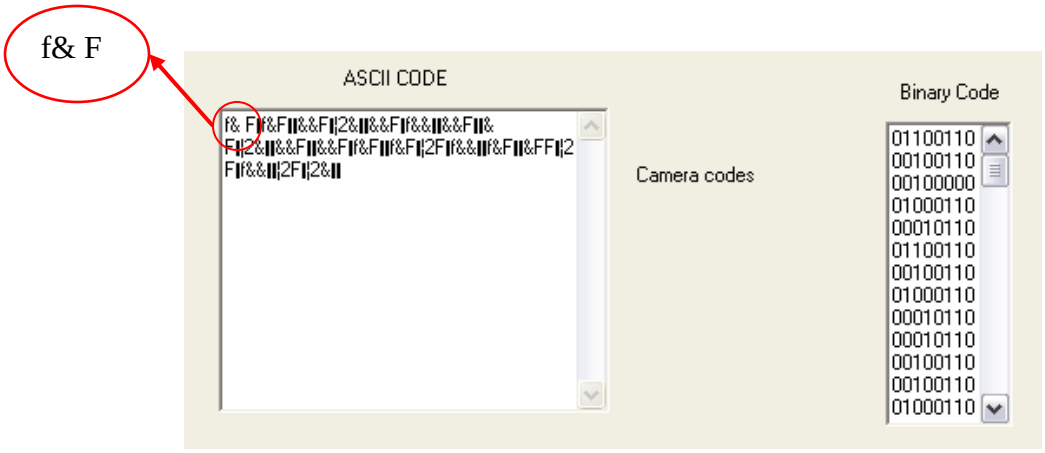

Figure 5.1: ASCII code and Binary Code Comparison

Please refer to appendix E for further reference, from Figure 5.1, it can be seen that the 1st character in the ASCII code is ‘f’ while the binary give out the 8 bit data of ‘01100110’. The binary data value when convert to decimal is 102, and from the ASCII table in appendix, it was shown that the character is ‘f’, this shows that the data from the ASCII code are valid and further validation can be seen from the table 5.1 below, please refer to both Figure 5.1 and ASCII code table in Appendix E,

Table 5.1: Data Comparison between ASCII and Binary

ASCII	Binary	Decimal
f	01100110	102
&	00100110	38
<Space>	01000000	32
F	01000110	70

For image processing, usually, the user need either the binary data or converts it to Hex code, before breaking up the 8 bits data into 4 bits that represent 2 pixels, before mapping the pixels into color code according to the color coding of the camera. Though this can only be applicable using a serial port camera, for further discussion, please refer to Chapter 6, page 48

5.2 Discussion

From this project, it can be seen that the data displayed are in ASCII code and also in binary data. It need further processing as describe just now for the image to be displayed. In order to display the image, image processing technique is needed, and also a serial port camera is needed.

The camera used in this project is a CMOS CCTV type, and the data that were sent to the computer through microcontroller are ever-changing. This happens because the images that the camera captured are ever-changing. In order to let the user secure the image data that the user required, the user can press pause to pause the current data flow and take the data that was required. The auto-save format that was incorporated into the user interface also allows the user to check the previous data in the saved file. As the image data keep flowing in, the data that were display are ever-changing and user might not get the data that he or she required. This can be solved by having the auto-save feature, the previous image data can be taken from the saved file in notepad form. Since the data is from a CCTV type camera, the data that were received are in video data format, this make it very hard to convert it back to image form since the data go through ADC and the operating frequency range for the ADC and the video data are not the same.

The RF module had been tested and found it to be quite reliable and secure. This RF module requires low power consumption and is suitable for short range communication. The maximum range that the RF module can support is around 20 meter. It's range can be further enhance by giving it more power, by changing the supplied voltage from 5 volt to 9 volt. Even though the RF module are quite reliable, still sometime the data that was transfer sometime might had disturbance or noise from other sources. Disturbance or noise can cause unwanted data displayed on the user interface screen and might cause the data to be inaccurate. This can be solved by using remote control encoder and decoder that can prevent noise by encoding the waveform in a special waveform

Sometime problems such as data loss or data transferring delay between the microcontroller and the PC based station might occur; this was assumed to happen due to the traffics between ports and too many operations that have to be executed concurrently. To solve it, the microcontroller sending rate had to be delayed by delaying the sending rate by calling the ‘Delay’ when the data was send (please refer to appendix D, line 34)