

Project Final Report

Bluetooth Camera Sensor

04/21/2011
Yichao Yu

Project Abstract

There are many ways to control a robot. I'm thinking why we cannot just use a common portable device as a remote controller so that, at least, we don't need to take another controller in our hands. The cell phone with Bluetooth module would be an option. It is very common. Bluetooth works at a fairly enough quick speed. What's more, I want the robot to have vision so that it could get to some places where we human being couldn't access to and we can control it where it will go. I need a camera sensor. So, as my final project, I decided to build the "eyes" and control parts of the robot.

The camera is responsible for generating raw image data and transmitting it to a cell phone by Bluetooth. The cell phone sends control commands to robot's microcontroller. The ZNEO controls all the communications between two devices.

Status

When the project was due, I almost finished all the things I planned to do. The app on the cell phone worked perfectly. It could display the images on the screen and could send commands to "robot". I also reserved some interface for the future concerns, e.g. more functional options. However, the camera sensor once worked, and is broken down now. Although I do not know what is wrong with the sensor, I'm quite sure that the sensor I chose is really vulnerable and is always malfunctioned. And it's large and expensive. The other thing I planned, but failed to accomplish is to continuously retrieve image data and display the images. Now the image is displayed after a command is sent to the microcontroller. I did know what made this happen. But it was still hard to find proper solution due to the bad camera sensor.

Details will be further described in the documents.

Specification

Design overview

The purpose of this project is to create a camera module interface over Bluetooth channel. ZNEO microcontroller is used to retrieve data from the camera module; and control the timing and the data flows between the camera and cell phone over the Bluetooth channel. The microcontroller is also responsible for accepting camera configuration commands and then setting the camera registers to the appropriate values using the UART bus. An mobile app on the cell phone provide GUI for displaying of the captured pictures as well as robot control commands.

The several components and modules will be used in this project:

- Zilog ZNEO Z16F series Z16F2811FI development kit
- LinkSprite JPEG Color Camera sensor LS-Y201
- Roving Networks Class 1 Bluetooth® Module RN-41
- Blackberry torch 9800
- Sparkfun 2.4GHz Duck Antenna RP-SMA

Hardware modules

Z16F2811FI development kit

The Z16F2811FI development kit provides some features for the project:

A microcontroller module; 2 system clock sources; 2 UART serial ports with 16-bit baud rate generator; A-K GPIOs; 5V and 3.3V power supplies.

The system clock frequency is 18.432 MHz or 5.5 MHz. So the baud rate of UART serial ports range from 1152000bps (ex-clock source) to 83.9bps (in-clock source).

The development board also contains an RS-232 connector circuit for UART0, which is used for printf function of debugging. And timers, an interrupt controller in the microcontroller helps to do sampling and debouncing for switches, which are the onboard control interfaces to help test the program at the first stage.

JPEG Color Camera sensor LS-Y201

LS-Y201 is a serial port camera module. It can take high resolution color JPEG pictures. The output of this camera module is UART, so it can be easily integrated into a system design.

The resolutions of the camera module are 160*120 (default), 320*240 and 640*480. The default baud rate is 38400 bps, and it is up to 1152000 bps. The power supply is 5V or 3.3V. The UART is RS232 level. If want to connected to MCU, add a level shifter or remove MAX232IC.

Pin descriptions:

Name	descriptions
+5V	Power
GND	Ground
TXD (OUT)	RS232 level connected to MCU or PC RXD
RXD (IN)	RS232 level connected to MCU or PC TXD

Basic control commands:

1. Reset camera:

Command (HEX)	Return (HEX)
56 00 26 00	76 00 26 00

2. Take picture:

Command (HEX)	Return (HEX)
56 00 36 01 00	76 00 36 00 00

3. Read image file size: XH and XL is msb and lsb byte of the file length

Command (HEX)	Return (HEX)
56 00 34 01 00	76 00 34 00 04 00 00 XH XL

4. Read image file content:

Command(HEX)
56 00 32 0C 00 0A 00 00 MH ML 00 00 KH KL XX XX
Return (HEX)
76 00 32 00 00 (Interval time) FF D8 (Interval time) 76 00 32 00 00
(Interval time) = XX XX*0.01mS XX XX are recommended to be 00 0A
00 00 MH ML: Starting address
00 00 KH KL: Length of JPEG file
MSB first, and LSB last

5. Stop taking picture:

Command (HEX)	Return (HEX)
56 00 36 01 03	76 0 36 00 00

Bluetooth® Module RN-41

The RN41 is a small form factor, low power, highly economic Bluetooth radio for OEM's adding wireless capability to their products. The RN41 supports multiple interface protocols, is simple to design in and fully certified, making it a complete embedded Bluetooth solution. The RN41 is the perfect product for engineers wanting to add wireless capability to their product but don't want to spend significant time and money developing Bluetooth specific hardware and software. It also uses UART serial port and the baud rate is 1200bps up to 921Kbps.

The Bluetooth module used in the project is RN-41 on a breakout which is made and retailed by Sparkfun. It supports the SPP (Serial Port Profile) which enable it to emulate a serial connection over the Bluetooth link.

Software designs

ZNEO embedded programming

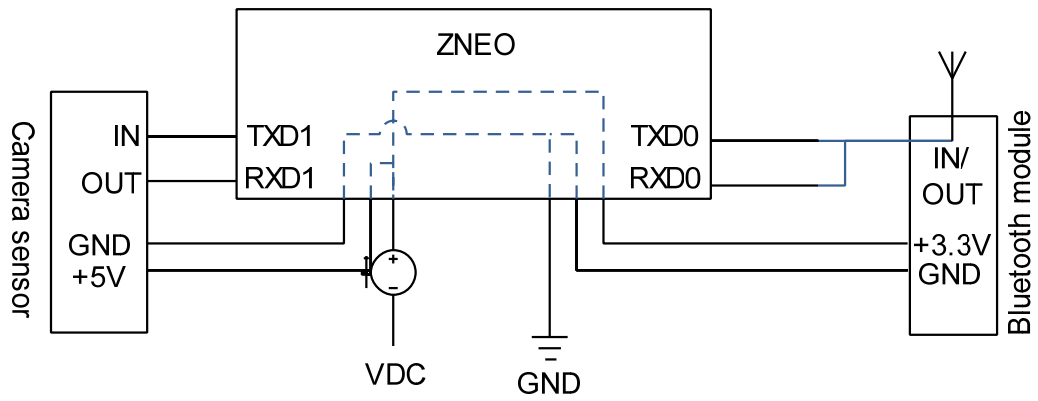
The majority of the work on the ZNEO is how to correctly retrieve and transmit raw image data. Besides these, the program is responsible for initializing all the modules and controlling the Bluetooth module and camera sensor working properly.

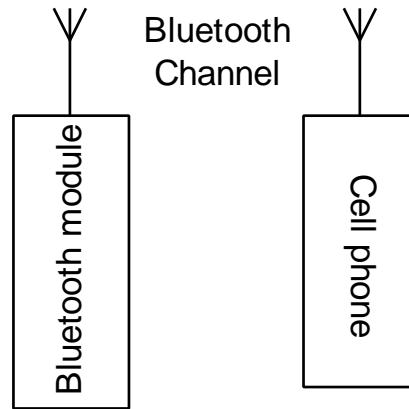
Blackberry OS6.0 client-side app programming

The cell phone end app pairs with and makes the connection with Bluetooth module. And it provides the GUI for user to see images taken by camera sensor and to send robot control commands to ZNEO.

Implementation & Construction

Hardware interfacing

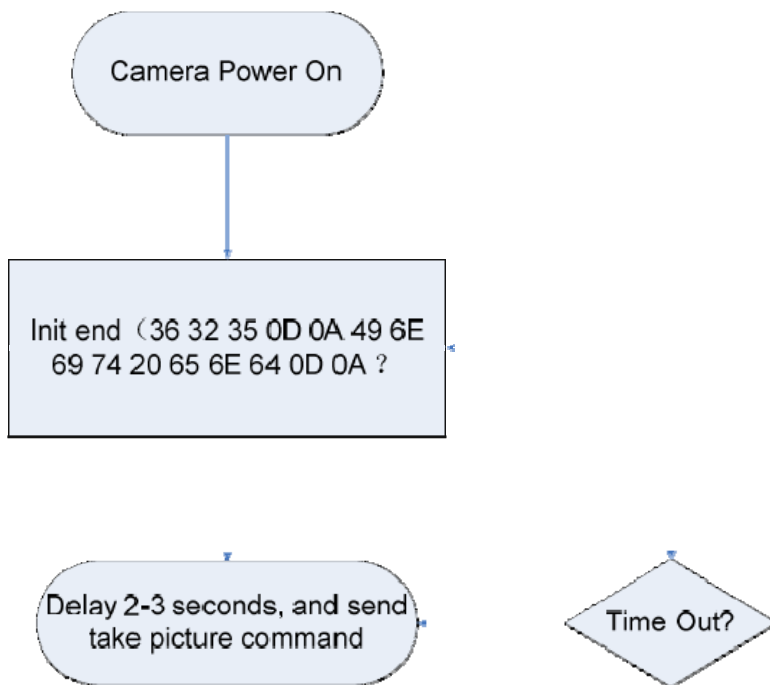




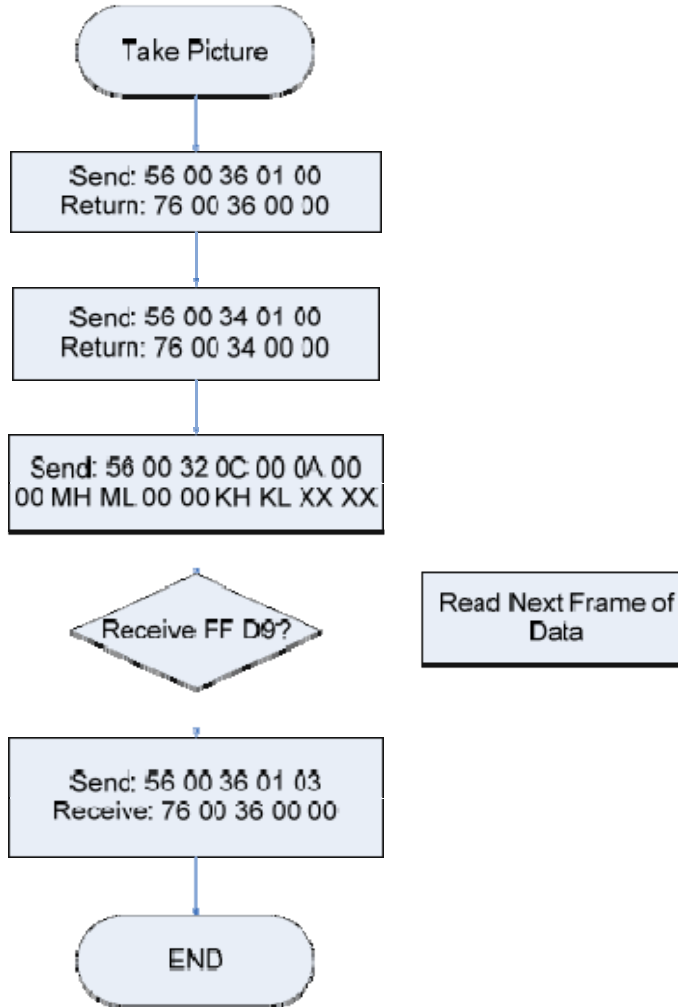
Software flow chart

Camera sensor flow chart:

Initialization:



Capture JPEG picture:



The app on the cell phone runs the almost same as the flow chart above. The only difference is just to keep receiving raw data and putting into an image buffer until meet FF D9.

Milestone chart:

Construction point	Z/B	Content
Hardware wiring 1	Z	Connect the camera sensor with the UART0 port on ZNEO. Run the “reset” command by polled transmits and retrieve return value by interrupt receive. This is to verify whether the camera sensor works and figure out how to communicate with the camera.
The first basic app 1 on blackberry	B	Power on the Bluetooth module, run the app on cell phone, which is to pair with Bluetooth module and to test whether the cell phone can successfully establish the connection between two devices. The message “DTR: true” means successful.
Read datasheet and configure camera	Z	Figure out set the baud rate and the image size. Increasing the baud rate can speed up the image

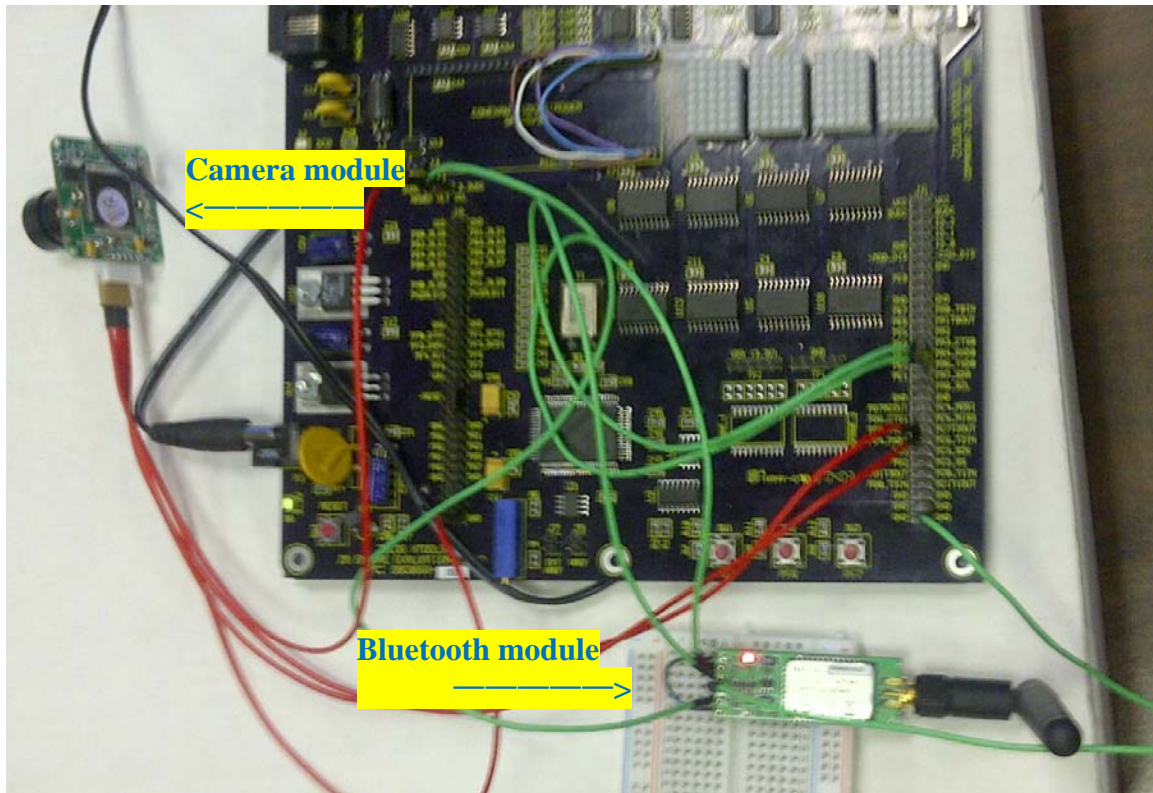
Project Final Report

sensor		transmission and changing the file size to an appropriate value is intended to make the image display in a real-time way. But, up to then, there was no way to ensure the configuring be successful because the return value tells nothing about correctness, which I found out later.
Take pictures and display raw data on terminal	Z	After configuring the camera, the next stage was to look at the raw data, how it was organized. By sending “read file length” command, the program could get the file length of the image taken just now. The “read file content” command returned part of the image file according to the MH and ML parameter (start address) in the command. That meant that the program needed to retrieve the image data recursively by changing the start address within the file. The page size, KH and KL, decided how many bytes each time the program retrieved.
Hardware wiring 2 and basic app 2	Z	Connect the Bluetooth module with the UART1 on ZNEO and configure the ZNEO from terminal display to Bluetooth mode. Modify the cell phone app to receive the byte from the Bluetooth module. Do some simple transmit/receive test.
ZNEO program and Cell phone app co-debugging	Z/B	This period took much time. It was trying to find a way correctly transmit all image data and none junk byte to Blackberry app. At the beginning of this period, images could not display on cell phone, but the data byte. But the problem was there were 0x0eff bytes transmitted and displayed. So it was hard to track the data byte on cell phone. At the end of this stage, an image could be shown on the screen.
Continuously image transmit debugging	Z	The goal was every time when a button had been pressed, an image could be shown on the screen. The update rate was around 2 seconds per image, which was not good enough. So I tried two methods, each of which worked: polled receive data from camera and change the baud rate of the camera.
Adding control command to cell phone app.	B	Add some control command for robot to cell phone app, which was also the reserved interface for future’s works.
System test	Z/B	Test all basic functionalities of the system on both the ZNEO and Blackberry.

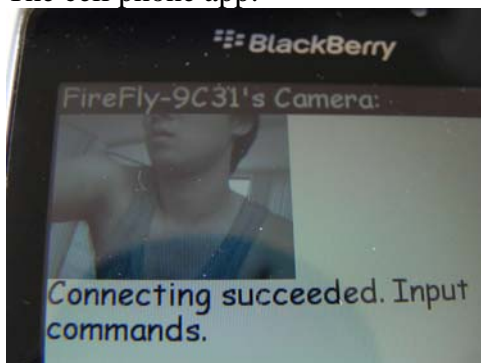
Note: Z/B indicates that whether it’s a ZNEO or Blackberry milestone.

View of the entire system

ZNEO's view



The cell phone app:



Retrospective

The most important decision I made was to use the polled receive method for raw image data retrieval. It worked on both good and evil ways. As I demonstrated in last section, I changed the interrupt receive to polling. That's because, first, the configuration of the camera sensor is a mess. It was hard to know which baud rate and image size the programmer set. The programmer must try several times to know where he is and which baud rate he set. This time he used 0x11 as a parameter for 11520000 bps. Then, next time after reset the camera, he may use 0x22 for that baud rate. And sometimes, he had to reset the camera sensor. Keeping the baud rate as default should be a safe way to do the rest of work. So, in order to improve the performance of the system, I chose the polled

receive method to read raw data from the camera, which worked better than interrupt way under the speed of 38400 bps.

Now, here was another problem. The polled method would return junk bytes from camera, after reading all raw image data, which sometimes would make the program stuck. A solution was to run some while loops to kick the junks out. The latest version of the system, though still will fail to transmit the image data, could recover from the failure and keep going on. The failure looks like “frame skip” when the system is running. Next time, I think I will change the camera module to a smaller, more stable and cheaper one. A RCA camera should be a good choice.

Attachments

ZNEO code files list:

Bluetooth.c Bluetooth.h: Bluetooth transmit and receive control.

Camera.c camera.h: camera configuration commands and data transmission control.

InterruptModule.c InterruptModule.h; timerModule.c timerModule.h: deal with the debug part of program (led array display, buttons debouncing) and UART port.

Uart.c uart.h: the uart initialization.

Main.c: the entry of the program and synchronization of the program so that each part works properly and is easy to communicate with the cell phone.

Header.h: define some global constant values and #define values, and import ZNEO libraries.

The rest header files are the ZNEO’s libraries.

The resources:

Datasheet:

- <http://www.linksprite.com>
- <http://www.rovingnetworks.com>
- ZNEO Z16F Series Product Specification PS0220
- ZNEO Contest Kit Users Manual UM0197

Wiring references:

- <http://wiring.org.co/learning/tutorials/bluetooth>

Blackberry app

- <http://www.blackberry.com/developers/docs/6.0.0api/>
- [Blackberry 6.0 Bluetooth demo](#)