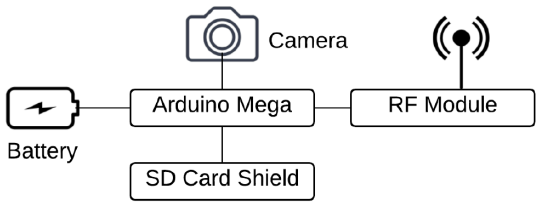
Video Lecture 4: Image Transmission

7COM1030 – Multicast and Multimedia Networking

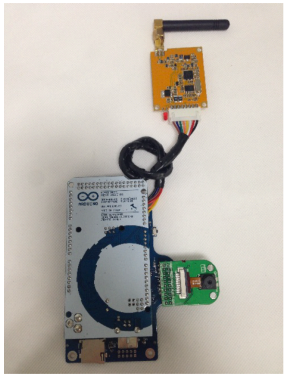
Dr. Xianhui Che (Cherry)
x.che@herts.ac.uk
School of Computer Science
University of Hertfordshire, UK

Wireless Image Sensor Stations

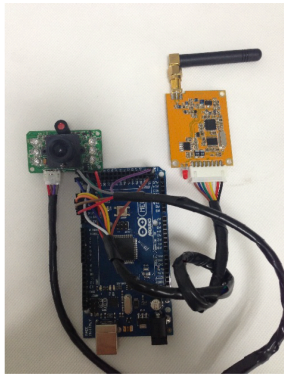
- Typical image sensor stations contain functionalities including capturing images, decomposing the image files into an array of pixel data, transmitting the data though an RF module, and saving the image to SD card.



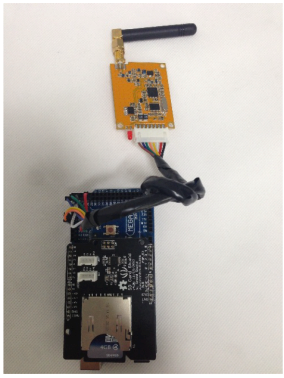
Hardware Example



(a) Transmitter node with ArduCam

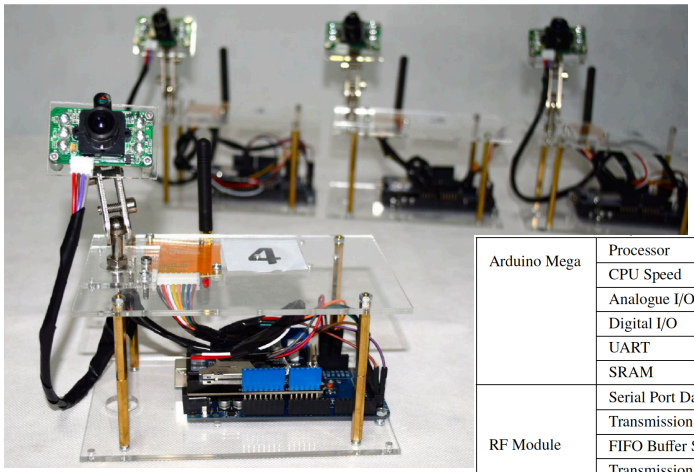


(b) Transmitter node with serial camera



(c) Receiver node with SD storage

Prototype Example



Arduino Mega	Processor	ATmega2560
	CPU Speed	16 MHz
	Analogue I/O	16/0
	Digital I/O	54/15
	UART	4
	SRAM	8 KB
RF Module	Serial Port Data Rate	Up to 57,600 bps
	Transmission Data Rate	Up to 19,200 bps
	FIFO Buffer Size	Up to 256 bytes
	Transmission Power	Up to 20 dBm
	Modulation	GFSK
Image Sensor	Serial Camera	320×240

Typical Code for Transmitting Image

The general approach of image transmission is to capture an image, decompose the image file into an array of pixel data, transmit the data though RF module, and then save the image to SD card.

Algorithm 1: Pseudo code for transmitter node

```
initialization;
take an image;
while not the end of the file do
    while not the end of each line do
        read one byte from camera FIFO;
        add the byte to the image string;
        move onto the next byte;
    save the line of data to SD card;
    add frame footer to image string;
    send the image string to serial port;
    add a delay as guardband period;
```

Item	Data	End of Transmission
Content	-	'\n' (0000 1010)
Length (bytes)	240	1

Item	Data	End of Transmission
Content	-	'\n' (0000 1010)
Length (bytes)	240	1

Item	Data	End of Transmission
Content	-	'\n' (0000 1010)
Length (bytes)	240	1

.
.
.

Typical Code for Receiving Image

Algorithm 2: Pseudo code for receiver node

```
initialization;
while not the end of the file do
    while not the end of the frame do
        read one byte In Byte from serial port;
        if the byte is the frame footer then
            set End of Frame to true;
        else
            add the byte to an image string;
    save the image string to SD card;
```

RGB565 and RGB555 Coding

HI Byte	Red Value					Green Value		
	0	1	0	0	1	1	0	0
LO Byte	Green Value			Blue Value				
	0	0	1	0	1	0	0	0

(a) RGB565 Coding

HI Byte		Red Value				Green Value		
	-	0	1	0	0	1	1	0
LO Byte	Green Value			Blue Value				
	0	0	0	0	1	0	0	0

(b) RGB555 Coding

The most significant bit is unused in RGB555.

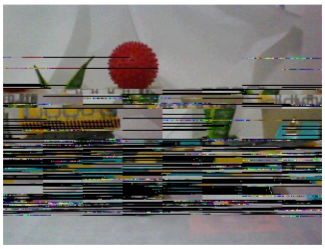
The two-byte color values of each pixel is not ASCII coded.

BMP images are saved in RGB555 format.

Since many cameras are RGB565 based, it is necessary to convert the two formats.

Avoiding Distortions and Noise

(a) Original bitmap image



(b) Bitmap transmission attempt 1

(c) Bitmap transmission attempt 2

(a) Received Bitmap Image

Distortion

- End-of-transmission and end-of-line should be clearly indicated to reflect the end of a frame, therefore reduces the probability of collision.

Item	Data	End of Transmission
Content	-	'\n' (0000 1010)
Length (bytes)	240	1

TABLE II
FRAME FOR IMAGE TRANSMISSION (FIRST ATTEMPT)

Item	Data	End of Transmission
Content	-	'EOT''\n' (0000 0100 0000 1010)
Length (bytes)	240	2

TABLE III
FRAME FOR IMAGE TRANSMISSION (SECOND ATTEMPT)

Noise

- The noise points and black lines mostly starts from black-colored areas.
  - Each black pixel is 2 byte long and presented in binary code 0000 0000 0000 0000.
  - Experimental tests have shown that the instruction "Serial.print()" does not transmit an array completely with a binary 0000 0000 byte in it, instead it will only transmit the bytes before the NULL byte.
- Two solutions:
  - Some embedded system development platform (e.g. Arduino) differentiates "Serial.write()" command and "Serial.print()". Do try both to see if the problem is solved.
  - Design a loop for the print command: (for(int i = 0; i <package length; i++) { Serial.print(package[i]);} ;).

Physical Environment

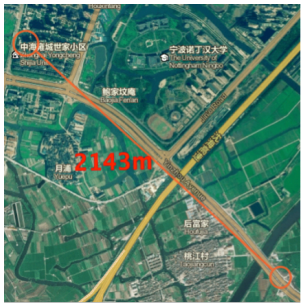


Image format: JPEG
Image size: 50 – 70 KB
Pixels: 320*640
Testing type: line of sight
Temperature: 17 degree
Air pollution index: 290
Transmission latency: 5 – 7 s

- In reality, distortion or noise may occur (mostly unpredictable) as the wireless/mobile signals are susceptible to the limited physical conditions.
- The QoS should be well-defined to quantify the acceptable and the best level of system performance.

BMP vs. JPEG

- **BMP:** A typical QVGA (320*240) image captured by a CMOS camera will possess 2*320*240 = 153,600 bytes.
  - These bytes are divided into multiple frames which will be transmitted in order. In practice, a loop can be used to transmit these frames.
- **JPEG:** The resolution of the taken JPEG image is QVGA 320*240 which is the same as the bitmap files. However, unlike bitmap files, JPEG encoded images of a same resolution may not necessary have the same file size.
  - A fixed length loop cannot be used to send JPEG files.
  - A standard JPEG file always starts with FF D8 in hex and ends with hex FF D9. A control protocol can be programmed at the transmitter end to continuously read the JPEG file until hex FF D9 is detected which means end of the file.

FF D8 XX XX... XX XX XX XX (240 bytes)
XX XX XX XX... XX XX XX XX (240 bytes)
....
....
XX XX XX XX... XX XX XX XX (240 bytes)
XX XX... XX XX FF D9 (100 bytes)

Example Calculation of Data Rate

- The colour values of the pixels has the storage space of 2*320*240 = 153600 bytes. It was divided into 153,600/240 = 640 frames. Each frame is 240 bytes plus overhead information for transmission.
- The required time to send 242 bytes (one data frame plus overhead) at baud rate 19200 b/s is: 242*8/19200 = 0.101s = 101ms
- Therefore total transmitting time is 101*640= 64.6s, and the average data rate is: 153600/64.6 = 2377.7 bytes/second, approximately 2.4 KB/s or 19 Kbps.

Arduino Mega	Processor	ATmega2560
	CPU Speed	16 MHz
	Analogue I/O	16/0
	Digital I/O	54/15
	UART	4
	SRAM	8 KB
RF Module	Serial Port Data Rate	Up to 57,600 bps
	Transmission Data Rate	Up to 19,200 bps
	FIFO Buffer Size	Up to 256 bytes
	Transmission Power	Up to 20 dBm
	Modulation	GFSK
Image Sensor	Serial Camera	320×240

Questions?

- **Email:** x.che@herts.ac.uk
- **Office:** LB218
- **Tel:** 01707 286206