# Remote Image Sensing Platform Based on Arduino

Zhuge Yan
School of Electrical and Electronic Engineering
University of Nottingham, China
zhuge.yan@nottingham.edu.cn

Xianhui Che
School of Electrical and Electronic Engineering
University of Nottingham, China
cherry.che@nottingham.edu.cn

John Walker
School of Electrical and Electronic Engineering
University of Nottingham, UK
john.walker@nottingham.ac.uk

***Abstract*—Wireless multimedia sensor network drastically stretches the horizon of traditional monitoring and surveillance systems, of which most existing research have utlized Zigbee or WiFi as the communication technology. Both technologies suffer from the relatively short transmission range. The objective of this paper is to assess the feasbility and explore the potentials of transmitting image information using RF transceivers under a carrier frequency of 433MHz. Arduino platform will be used for its low cost and simplicity feature. Details of the hardware properties and the equipment setup will be elabroated. The stablity of the system will be firstly tested by transmitting simple texts, then the control protocol is written to decompose the image files at the software level for the wireless transmission. Comparison and evaluatation will be made for different system settings including image format and serial interface, so that the optimum configuration can be concluded for the future wireless multimedia sensor network experiment.**

***Index Terms*—sensor network, RF, arduino**

## I. Introduction

Wireless sensor network (WSN) is a hot research field which involves a high degree of multidisciplinary, and is the current concern of the international community. With the development of sensor nodes, micro-electro-mechanism systems, modern networks and wireless communication technology, WSNs vastly extend the ability of human beings to obtain information of the physical world, transmits it using wireless networks. Wireless multimedia sensor network (WMSN) is a new type of sensor networks, the nodes are equipped with a microphone, cameras and other sensors, can obtain the Multimedia information. It provides the most direct, effective, and authentic information in the next generation network. Recent survey in this field includes [1], [2], [3].

The industrial scientific medical (ISM) Bands are licence-free RF bands established by the International Telecommunications Union, and can be used for industrial, scientific and medical purposes. Most existing technologies that have taken advantage of the ISM bands for the implementation of WMSN are using either Zigbee or WiFi technology, both of which have the advantages of high bandwidth. However, the line of sight transmission range for Zigbee is 100 meters [4] and the one for WiFi is 140 meters [5], both of which can limit the scope of the network deployment. Research in [6] has demonstrated a simulation of a network with 400 nodes deployed, but the dimension of the area only covers 400m*400m.

The aim of this research is to assess the feasiblity and the potentials of building a WMSN using a low-cost Arduino platform and the license-free 433 MHz RF modules. The band 433.050-434.790 MHz with a center frequency of 433.920 MHz will be used for the text and image transmission tests. Currently this frequency band is mostly utilized for point-to-point wireless communications with low data flow or traditional WSN such as a remote meter reading system. The RF modules used in this research have been tested for 2000 meters line-of-sight effective transmission range. The potentials of this frequency band in the WMSN research will be explored in this paper.

The stability of the RF communication will be firstly tested by transmitting simple texts, then the control protocol is written to decompose the image files at the software level for the wireless transmission. The image sensors will capture both bitmap (BMP) and joint photographic experts group (JPEG) encoded images, both of which will be compared and evaluated. The cameras can either be connected to the Arduino board with serial interface or serial peripheral interface (SPI) which will be fully investigated so as to find the optimum settings for the platform.

Arduino is a flexible and user-friendly electronic prototype platform with open source scheme includes the sharing of knowledge on both hardware and software level [7]. The Arduino product has gone through dramatic growth over the past few years for its simplicity, cost-effective, and low power consumption features. Currently many researchers in the WSN field have utilized Arduino as the fundamental platform, which will be explained in the next section.

Section II briefly describes the existing work in the area of WMSN and using Arduino in WSN. Section III explains in details the system implementation including both hardware and software settings. Section IV presents the initial signal testing outcome of text transmission on the designated system. Section V demonstrates the image transmission results and compares the data with different settings. Section VI overviews the future work, and section VII concludes the paper.

## II. Existing Work

Various work have been carried out in the WMSN field, although most of which have utilized either Zigbee or WiFi technology. Low-performance platforms such as TelosB mote uses the TI MSP430 family processor, and high-performance

platforms like IMote2 which uses an 32-bit Intel/Marvell PXA271 processor clocked up to 416 MHz [8]. [9] introduced a high-end Wireless Image Sensor Network testbed with an ARM7 32-bit core up to 48 MHz, which reaches a good equilibrium between cost and performance. Some testbeds are even robot-based to achieve the mobility of sensor nodes, such as Explorebots [10] and Emulab [11]. However, both of them are of relatively high cost. There are also software tools developed for WSN testing, such as MoteLab [12], while its compatibility to sensor nodes is limited.

Arduino is also widely used as testbed platforms in the development of WSNs [13], [14]. We chose Arduino because of its low-cost, high usability and popularity. The attempts made to build wireless networks mostly use XBee modules which is based on the 802.15.4/ZigBee standard [15], [16], [17], [18]. Traditional WSNs transmit data of bytes, such as temperature [15] and logistics [17]. Nowadays, the demand for multimedia is becoming greater. Other than data in digits people are now looking for images, sound and even videos. [6] used XBee to build Wireless Image Sensor Networks based on 2-hop neighborhood knowledge. A standard XBee module has a indoor/urban range of 30m, an outdoor range of 90m and a RF data rate of 250 kbps, while an 433 MHz RF transceiver module has a much longer indoor/urban range up to hundred meters and an outdoor range up to kilometers. Moreover, although 433 MHz RF modules have a slower data rate (usually under 50 kbps), it uses a carrier frequency band not as crowded as XBee (2.4 GHz, which is used on most wireless communication technologies (WiFi, Bluetooth, WirelessUSB, etc.) uses the 2.4 GHz band, there is a high probability of signal interference) does.

## III. System Implementation

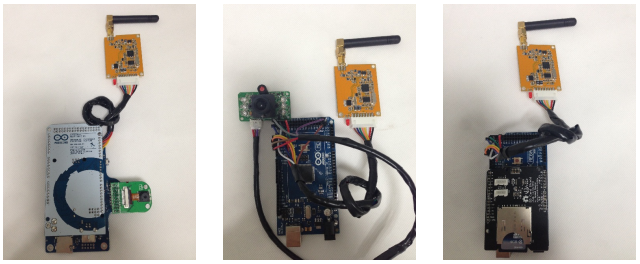
### A. Hardware

There are various models available in the Arduino family. Arduino Mega has been chosen for this platform because of its built-in specifications. Compared with other Arduino models, the larger number of I/O interfaces and universal asynchronous trceiver/transmitter (UART) ports on the Mega model makes the system implementation relatively future-proof and easy to scale, as more features can be easily added to the application. The technical specifications of the system are described in Table I.

One of the limitations of Arduino family products is the shortage of extended high performance capabilities, and when networking or image processing is to be enabled on Arduino, there is often a need for a shield to interface a separate functional hardware module. ArduCam was born for the purpose of imaging capturing, which most Omnivision (OV) and Aptina camera modules [19]. It transmits the captured images to Arduino board through SPI. OV2640 camera module is used as an image sensor in this system thanks to its high popularity and availability, as shown in Figure 1(a). At the receiving end an SD card shield serves as data storage, as shown in Figure 1(c). A SD card storage can also be installed

TABLE I
System Hardware Specifications

| | | |
|---|---|---|
| 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 kpbs |
| | Transmission Data Rate | Up to 19,200 kpbs |
| | FIFO Buffer Size | Up to 256 bytes |
| | Transmission Power | Up to 20 dBm |
| | Modulation | GFSK |
| | Transmission Range | 2000 meters (tested) |
| Image Sensor | ArduCam | 320×240 |
| | Serial Camera | 320×240 |



(a) Transmitter node with ArduCam (b) Transmitter node with serial camera (c) Receiver node with SD storage

Fig. 1. System hardware

in the transmitter node to retrieve the original image which the received image can be compared with.

Another image sensor under test is a serial port camera module as shown in Figure 1(b), It directly takes JPEG pictures and transmits them to Arduino through serial communication port. An SD card shield needs to be installed on the transmitter node to save the taken images temporarily before the wireless transmission. The receiver node is the same as is shown in Figure 1(c).

A highly integrated RF module is employed in the system, as shown in Figure 1, which works on half-duplex mode and connects to the Arduino board via serial interface. It uses GFSK modulation to provide multiple channel choice and it is equipped with a first-in-first-out (FIFO) buffer. The serial port rate, transmission power, radio frequency (RF) rate and other parameters can be configured via software. The specifications of the RF module are listed in Table I.

Although the RF model has a large FIFO buffer size of 256 bytes, the average serial port rate must be lower than 60% of the RF rate. Otherwise, an average serial port rate which is higher than 60% of the RF rate will lead to data overflow. For example, one is using a package length of 100 bytes, at serial port and RF baud rate 9600. Therefore the required time to send 100 bytes through serial port is 100*10/9600 = 0.104s = 104ms. Hence we need time slots which are (104/60%)*(9600/9600) - 104 = 69ms long follow to each

frame. To improve the efficiency of the system, FIFO-reading data and data-saving-to-SD are processed simultaneously during the time slots. The testing result regarding this issue is shown in Section IV.

### *B. Software*

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. Future research will develop an end processing software so that all received images can be received and displayed live on a monitor screen. At the first research stage, the transmitting process is point-to-point and one-way, the transmitter and Receiver node are programmed separately. The pseudo codes for the transmitter node and receiver node are shown in Algorithm 1 and Algorithm 2 respectively.

**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;
```

**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;
```

## IV. Signal Testing and Text Transmission

The RF transceiver has adjustable serial port rate and RF rate. The default serial port baud rate and RF baud rate are both 9600. There are 7 choices for the serial port baud rate: 1200, 2400, 4800, 9600, 19200, 38400 and 57600, and 4 choices for the RF baud rate: 2400, 4800, 9600 and 19200. The default setting is used during the initial signal testing. And when testing text and image transmission, the serial port rate and RF rate were both set to the largest value, i.e. serial port baud rate 57600, RF baud rate 19200.

The average serial port rate must be lower than 60% of the RF rate. To test this issue the serial port and RF baud rate were both set to 9600. A continuous data signal was sent to the RF Transceiver using Arduino Mega which is a series of ASCII characters and printable in the console window as follows:

```
A, dec: 65, hex: 41, oct: 101, bin: 1000001
B, dec: 66, hex: 42, oct: 102, bin: 1000010
C, dec: 67, hex: 43, oct: 103, bin: 1000011
D, dec: 68, hex: 44, oct: 104, bin: 1000100
E, dec: 69, hex: 45, oct: 105, bin: 1000101
F, dec: 70, hex: 46, oct: 106, bin: 1000110
G, dec: 71, hex: 47, oct: 107, bin: 1000111
H, dec: 72, hex: 48, oct: 110, bin: 1001000
.....
```

An oscilloscope was used to monitor the digital output and input of the serial communication port. The result is compared in Figure 2. While the original signal is continuous, the received signal consists of discrete data packages. As the transceivers have the same serial port and RF baud rate of 9600 bps, it can be inferred that data overflow occurred during the transmission which caused the information loss. The readout of the received data also proved that some transmitted data was missing.

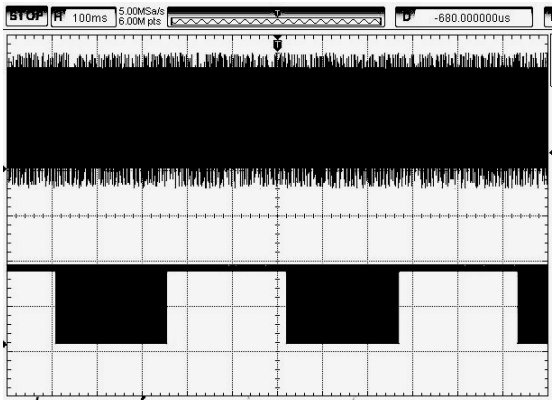
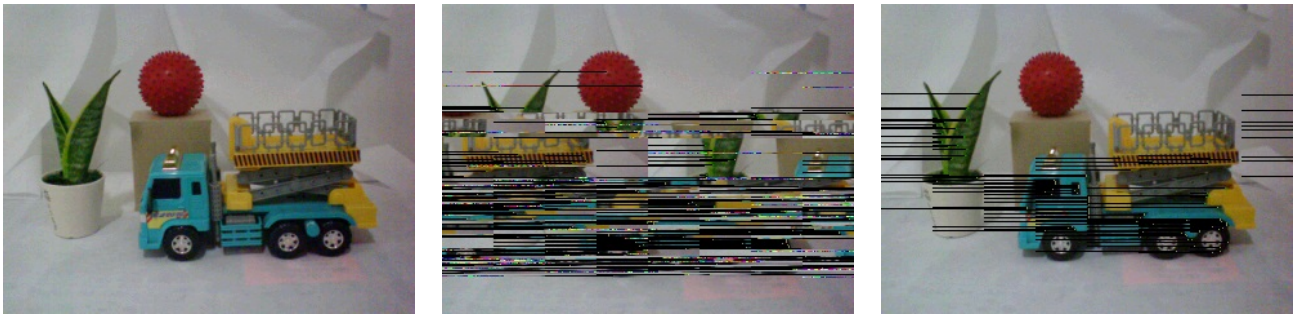



Fig. 2. Packet dropout when transmitted data is continuous (top: Original Signal, bottom: Received Signal)

The waveform of the transmitted and received signal in Figure 2 can be regarded as data volume since they have the same baud rate of 9600 bps. Hence the received signal has approximately 50-60% information of the transmitted signal. And because the RF baud rate is also 9600, it can be inferred that the average serial port rate should be lower than 50-60% of the RF rate. Further tests show that the average serial port rate should be be lower than 55% of the RF rate to allow some margin, which is a slower rate than the maximum rate. Since the FIFO buffer size of the RF transceivers is up to 256 bytes, the maximum length of a data package should be 256 bytes. A data package exceeding 256 bytes will also leads to data overflow even when the average serial port rate is lower than 55% of the RF rate.



(a) Original bitmap image (b) Bitmap transmission attempt 1 (c) Bitmap transmission attempt 2

Fig. 3. Original and received bitmap images

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 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

Fig. 4. RGB565 and RGB555 byte format

## V. Image Transmission

### A. Bitmap Transmission

The general approach is to use ArduCam to take a picture, read the original data from its FIFO buffer, carry out an RGB565 to RGB555 conversion, transmit it to the receiver node and than save it as a bitmap (BMP) image. The resolution of the image is QVGA (320*240).

The data read from the camera module is RGB565 coded. Each pixel returns two bytes: the HI-Byte and the LO-Byte. The first 5 bits of the HI-Byte contains the red value, the last 3 bits of the HI-Byte and the first 3 bits of the LO-Byte contains the green value and the last 5 bits of the LO-Bytes contains the blue value. To save the data to a bitmap file, it must be converted to RGB555, which uses 5 bits for each color's value. The most significant bit is left unused. Figure 4 shows the detailed usage of bits of RGB565 and RGB555. The transmitter node does the conversion and then send the data to the receiver node and save it as a bitmap file.

The color values of the pixels has the storage space of 2*320*240 = 153,600 bytes. It was divided into 153,600/240 = 640 frames of 240 bytes (plus overhead information) for transmission. As there are only one transmitter and one receiver and the transmission is purely one-way, , header was not included in the frame to reduce the complexity of the frame in the test. A new line semble ('\n' (binary 0000 1010) was used to indicate the end of each frame. The packages were transmitted using the "Serial.print()" instruction and a for loop which iterates for 640 times. Table II shows the detailed design of the frame.

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

TABLE II
Frame for image transmission (first attempt)

As is stated in Section IV, the average serial port rate needs to be lower than 55% of the RF rate, hence serial port baud rate of 57600 bps and RF rate of 19200 bps were used. Therefore time slots must be added between data packages (each package is a frame in this case) to avoid data overflow when sending images. The required time to send 241 bytes at baud rate 57600 is 241*10/57600 = 0.0418s = 41.8ms. So the time slot required is equal to (41.8/55%)*(57600/19200) - 41.8 = 186.2 ms. Hence the total transmitting time is (186.2 + 41.8)*640 = 144s, and the average data rate is 153,600/144 = 1066.7 bytes/second.

To assess the quality of transmission, the transmitted data was also saved as a bitmap file on the transmitter node. By comparing the difference of the two images, it can be found weather the transmission has reduced the quality of the image. The original image is shown in 3(a), and the first attempt of the received one is shown in Figure 3(b).

There is a severe distortion in the received image. Figure 3(b) shows serious malposition of pixels, which is caused by package dropout or data errors. Based on Table II, the end of each frame is indicated by a end-of-line symbol ('\n'). However, as the two-byte color values of each pixel is not ASCII coded, hence each byte can be any value of 0-255, including the new line symbol '\n' (binary 0000 1010). There is a probability for data error to occur, for example, one of the 153,600 bytes is binary 0000 1010 which for the receiver means end-of-frame, and hence it would end the receiving process while the frame was still incomplete. The frame was then reformed.

(a) Received Bitmap Image (b) Received JPEG Image taken by ArduCam (c) Received JPEG Image taken by LS-Y201

Fig. 5. Comparison of Received Bitmap Image Taken by ArduCam and JPEG Images Taken by ArduCam and LS-Y201

In order to solve the image distortion problem of the first attempt, an end-of-transmission symbol and a new-line symbol was used to indicate the end of a frame, therefore reduces the probability of collision. Table III shows frame design of the second attempt. The frame length is now 242 bytes. It takes 242*10/57600 = 42 ms to send 242 bytes through serial ports, so the time slots required is (42/55%)*(57600/19200) - 42 = 187.1 ms long. Hence the total transmitting time is (187.1 + 42)*640 = 146.6 s, and the average data rate is 153,600/146.6 = 1047.7 bytes/second, which is approximately equal to 1 KByte/second. Figure 3(c) below shows the received image of the second 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)

The image quality at the receiver end has improved significantly at the second attempt, however there are still some green noisy points followed by black lines on the image as shown in Figure 3(c). The noise points and black lines mostly starts from black-colored areas, and since the organization of the RGB555 pixels is from left to right and from bottom to top, therefore the assumption is made that it was the black color pixels in the image that were not transmitted properly. 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. Therefore if a frame has black pixels or some other pixels that contains a NULL byte in its 2-byte color values, the bytes after the NULL byte will not be transmitted. The bytes not transmitted leads to the noisy points and black lines.

There are two solutions for the noise problem that occurred in the second attempt. One is to use "Serial.write()" command instead of "Serial.print()". According to Arduino's online documentation [7], the print command will send data of many forms to the serial ports as human-readable ASCII text, and the write command will simply write binary data to the serial port as a byte or series of bytes. Hence another solution is to design a loop for the print command: (for(int i = 0; i < package_length; i++) {Serial.print(package[i]);};). The result has shown that the received image is identical to the original transmitted image, as shown in Figure 5(a). The time taken is still 146.7 seconds as the design of frame remains unchanged.

### *B. JPEG Transmission*

The investigation of JPEG image transmission is worthwhile because of its compressed information volume. JPEG transmission test used a similar approach as bitmap transmission. 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 is programmed at the transmitter end to continuously read the JPEG file until hex FF D9 is detected which means end of the file. The file can still be divided into packages of certain length (240 bytes used in this test), although the last package may be shorter than 240 bytes. Below is an example:

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)

The data packages were encapsulated in the format specified in Table III and sent through serial port. The receiver was programmed to stop receiving while hex FF D9 is detected to adapt the uncertainty of the file size. Figure 5(b) shows the received JPEG image taken by ArduCam. The received image is identical to the original image, indicating zero data error during the transmission. Due to the lossy compression algorithm used in JPEG images, areas of monochrome are normally expected [20], however the comparison between Figure 5(b) and Figure 5(a) indicates a satisfactory image quality with normal complexion.

JPEG images taken by ArduCam usually has a file size of 6-8 KBytes, which is significantly smaller than an RGB555 encoded bitmap file in the same resolution. With an average transmitting data rate of 1 KBytes, the total latency for transmitting JPEG images taken by ArduCam is approximately 6-8 seconds.

JPEG image taken by the serial port camera was also tested using the same approach. Figure 5(c) shows the received image taken by this camera. The image compression technique used in this system is slightly different from ArduCam which leads to a more crystal image quality and a larger file size, i.e. normally between 11 to 14 KBytes, which takes approximately 11-14 seconds to transmit point-to-point.

## VI. Applications

I. Akyildiz [1] has pointed out that WMSN can be used for a tremendous diversity of applications including multimedia surveillance sensor networks, activity storage, traffic avoidance and control, family and health care, environmental monitoring with acoustic and video feeds, personal locator services, and industrial process control. The images captures and transmitted in the experiments explained in this paper will be suitable for a wide range of applications mentioned above. The shortest delay discovered in this system is 5-7 seconds with JPEG images captured by ArduCam, which may not fit in time-critical applications, however the system will be effective for the application scenarios that are delay-tolerant in the order of seconds, such as traffic surveillance, environmental monitoring, etc.

One major advantage of the designated system is the long transmission range. Two stations were deployed at the locations with GPS coordinates of (N29.80128°, E121.55468°) and (N19.79132°, E121.56602°) respectively, and the test result demonstrated an effective line-of-sight transmission range of 2143 meters. Compared to existing WMSN work using Zigbee, Xbee or WiFi where 100-200 transmission range is allowed for, the designated system in this research is more suitable for deployment in relatively large dimensions.

## VII. Conclusion and Future Work

This paper investigates the feasibility and potentials of using the ISM 433 MHz band to transmit data captured by image sensors. The system is based on Arduino Mega board, and two cameras are tested with one using SPI-interface and one using serial interface. Upon a successful signal testing with simply texts, algorithms are designed to transmit image files while two issues have been discovered and solved. The quality of captured images are compared for BMP and JPEG format. The effective line-of-sight transmission range has reached over 2000 meters for the designated system. The findings in this paper will form an important foundation for the future WMSN research. The designated system has demonstrated both feasibility and effectiveness for specific sets of applications.

Future work will look into the mobility aspect of the sensor nodes. Multi-hop data relay will be enabled to transmit data to from a node that is outside the range of the sink, and network efficiency will be examined. Algorithms are required for the transmission of different data types such as audio. In terms of video transmission, it will be rather challenging given the limited bandwidth offered by the ISM 433 MHz carrier, although more practical solutions will be fully explored. The aim is to fully utilize the network capability so as to deliver the maximum WMSN performance.

## References


[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks", Computer Networks, vol. 51, pp. 921-960, 2007.

[2] I. F. Akyildiz, T. Melodia, and K. R. Chowdury, "Wireless multimedia sensor networks: a survey," Wireless Communications, IEEE, vol. 14, pp. 32-39, 2007.

[3] A. Sharif, V. Potdar, and E. Chang, "Wireless multimedia sensor network technology: a survey," Proceedings of 7th IEEE International Conference on Industrial Informatics, pp. 606-613, 2009.

[4] P. Baronti, P. Phillai, et al, "Wireless sensor networks: a survey on the state of the art and the 802.15.4 and Zigbee standards", Computer Communications, Elsevier, 2007.

[5] "IEEE802.11 wireless local area networks", The Working Group for WLAN Standards, last updated at 2014.

[6] M. Diop, C. Pham, et al., "2-hop neighborhood information for cover set selection in mission-critical surveillance with wireless image sensor networks", Proceedings of IFIP Wireless Days, pp. 1-7, 2013.

[7] www.arduino.cc

[8] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "Wireless multimedia sensor networks: applications and testbeds," Proceedings of the IEEE, vol. 96, pp. 1588-1605, 2008.

[9] I. Downes, L. B. Rad, and H. Aghajan, "Development of a mote for wireless image sensor networks," Proceedings of COGnitive Systems with Interactive Sensors, 2006.

[10] T. A. Dahlberg, A. Nasipuri, and C. Taylor, "Explorebots: a mobile network experimentation testbed," Proceedings of the 2005 ACM SIGCOMM workshop on Experimental Approaches to Wireless Network Design and Analysis, pp. 76-81, 2005.

[11] D. J. T. S. R. Fish, D. M. Flickinger, and L. S. R. R. J. Lepreau, "Mobile emulab: a robotic wireless and sensor network testbed," Proceedings of IEEE INFOCOM, 2006.

[12] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: a wireless sensor network testbed," in Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, p. 68, 2005.

[13] A. D'Ausilio, "Arduino: a low-cost multipurpose lab equipment", Behavior Research Methods, vol. 44, pp. 305-313, 2012.

[14] M. Schmidt, "Arduino: Pragmatic Bookshelf", 2011.

[15] V. Boonsawat, J. Ekchamanonta, K. Bumrungkhet, and S. Kittipiyakul, "XBee wireless sensor networks for temperature monitoring", Proceedings of the Second Conference on Application Research and Development, 2010.

[16] R. Faludi, "Building wireless sensor networks: with ZigBee, XBee, arduino, and processing", O'Reilly Media, 2010.

[17] L. Ruiz-Garcia, P. Barreiro, and J. Robla, "Performance of ZigBee-based wireless sensor nodes for real-time monitoring of fruit logistics", Journal of Food Engineering, vol. 87, pp. 405-415, 2008.

[18] N. Waddington and R. Taylor, "Arduino & Open Source Design", ItaliaDesign, 2007.

[19] www.arducam.com

[20] Gl Wallace, "The JPEG still picture compression standard", ACM Communication Magazine, April 1991.