


7-2015

# Wirelessly Connected Sensor Acquisition System for Remote Nursing Applications

Yuhua Hu  
*Kennesaw State University*

Follow this and additional works at: <http://digitalcommons.kennesaw.edu/etd>

 Part of the [Analytical, Diagnostic and Therapeutic Techniques and Equipment Commons](#),  
[Nursing Commons](#), and the [VLSI and circuits, Embedded and Hardware Systems Commons](#)

---

## Recommended Citation

Hu, Yuhua, "Wirelessly Connected Sensor Acquisition System for Remote Nursing Applications" (2015). *Dissertations, Theses and Capstone Projects*. Paper 681.

This Thesis is brought to you for free and open access by DigitalCommons@Kennesaw State University. It has been accepted for inclusion in Dissertations, Theses and Capstone Projects by an authorized administrator of DigitalCommons@Kennesaw State University. For more information, please contact [digitalcommons@kennesaw.edu](mailto:digitalcommons@kennesaw.edu).

# **WIRELESSLY CONNECTED SENSOR ACQUISITION SYSTEM FOR REMOTE NURSING APPLICATIONS**

**Project Report**

**Prepared By**

**Yuhua Hu**

**In partial fulfillment of the requirements for  
ECET 7704**

**July 23, 2015**

**Kennesaw State University  
Electrical and Computer Engineering Technology Department  
1100 South Marietta Parkway  
Marietta, Georgia 30060**

## TABLE OF CONTENTS

<u>Section</u>		<u>Page</u>
I	INTRODUCTION.....	5
1.1	PURPOSE.....	5
1.2	HARDWARE OVERVIEW .....	5
1.2.1	Arduino.....	5
1.2.2	Raspberry Pi .....	6
1.2.3	Wi-Fi/802.11n Module.....	7
1.2.4	Bluetooth 4.0.....	7
1.2.5	Video Camera Module.....	7
1.2.6	Temperature Sensor & Humidity Sensor.....	8
1.2.7	Servomotor.....	8
1.2.8	Computer.....	8
1.3	SOFTWARE OVERVIEW .....	9
1.3.1	Arduino 1.6.5.....	9
1.3.2	Raspbian.....	9
1.3.3	Application for Video Stream .....	9
II	SYSTEM MODELING AND BASIC DESIGN.....	10
2.1	BIT RATE CALCULATION.....	10
2.2	PROJECT DESIGN.....	13
2.2.1	Arduino-based Part.....	13
2.2.2	Raspberry Pi-based Part.....	14
III	SYSTEM MEASUREMENT.....	15
3.1	POWER CONSUMPTION MEASUREMENT.....	15
3.1.1	Arduino-based System Power Consumption.....	15
3.1.2	Raspberry Pi-based System Power Consumption.....	16
3.2	RANGE MEASUREMENT.....	18
IV	CONCLUSIONS.....	19
V	REFERENCES.....	21
VI	APPENDICES.....	23
A.1	Arduino Mega 2560 Datasheet.....	23
A.2	Raspberry Pi Model B+ Datasheet.....	24
A.3	Camera Module OV5674 Datasheet.....	25
A.4	DHT11 Temperature & Humidity Sensor Library Code.....	26
A.5	Servomotor Library Code.....	28

**LIST OF ILLUSTRATIONS**

<u>Item</u>		<u>Page</u>
Figure 1	Arduino based System .....	14
Figure 2	Raspberry Pi based System .....	14
Figure 3	PCB-LP Antenna .....	18

## LIST OF TABLES

<u>Item</u>		<u>Page</u>
Table 1	Resolution and Highest Frame Rate of the OV5674 Camera.....	11
Table 2	The Color Depth.....	12
Table 3	Arduino Board Power Consumption.....	15
Table 4	Raspberry Pi Board Power Consuming at Compressed Bitrate and Default Bitrate .....	17
Table 5	The Bluetooth and Wi-Fi Range.....	18

# I INTRODUCTION

## 1.1 PURPOSE

Due to increasing labor cost, hospitals are forced to reduce nursing staff for patient monitoring. In some cases, patients can be monitored by nurses or doctors at home, reducing hospital resources. For patients in rural communities, medical services may be scarce and remote patient monitoring is an alternative.

Taking advantage of the equipment used in the “Remote Nursing” project, nurses can watch patients’ status remotely. The equipment, which is installed in the patient’s ward or home, can monitor them and their environment. This project demonstrates this using a microcontroller system that interfaces to a video camera and temperature and humidity sensors in the room. Also a servomotor is installed to control the position of the camera. The microcontroller communicates wirelessly with a PC.

This project is focused on hardware rather than software. It is to test the video camera and sensors being used over the microcontroller through the wireless module. The sensors are controlled by drivers and graphical interfaces that come with the hardware or are freely available online.

Initially, an Arduino microcontroller with Bluetooth wireless communication module was chosen to be the remote sensor platform. However, for the reason of the low bit rate of the Bluetooth module, and low speed of the Arduino’s processor, the performance of the video stream is not good enough for this project. In this case, the project was updated to use the Raspberry pi and the Wi-Fi module for the video stream. The Raspberry pi has a special video chip – the VideoCore IV GPU (Graphics Processor Unit) [1] for the video, so the Raspberry pi is the best choice for the video stream.

Because the microcontroller platform may be used in different situations, including battery-powered, the project included power consumption and the wireless operating range measurements.

Furthermore, some other off-the-shelf medical instruments that can work over USB can also be added in the future in order to make the system work more functionally.

## 1.2 HARDWARE OVERVIEW

The key parts of this project are the microcontrollers and the wireless modules. The microcontrollers are an Arduino which is used to control the sensors and servomotor, and a Raspberry Pi to stream the video. The wireless modules for building communication with microcontrollers and PCs are a Wi-Fi module and a Bluetooth module. Two microcontroller platforms were used since the original chosen sensors were designed to interface to an Arduino but after discovering the poor video performance, a Raspberry Pi and compatible camera were used to improve the video.

### 1.2.1 Arduino

An Arduino is able to perform sensing and control functions more inexpensively than a desktop computer. It's an open-source physical computing platform based on a simple microcontroller board and a development environment for writing software for the board. [2]

The Arduino version used for this project is the Arduino Mega 2560R3. [2] It is regarded as one of most advanced board in the Arduino family. As shown in Appendix A.1, the Arduino Mega 2560R3 is based on a 32-bit ARM core processor, the Atmel SAM3X8E ARM Cortex-M3 MCU (Microcontroller Unit). It offers 54 digital input/output pins of which 16 can be used as PWM (Pulse Width Modulation outputs), 84 MHz crystal oscillator, two USB connection, a power jacket, 12 analog inputs, 4 UARTS (Universal Asynchronous Receiver/Transmitter) and two DAC (Digital Analog Converter) outputs. These can satisfy the requirement for this project. Last but not least, the power supply for the Arduino DUE is 5V, which is the lowest in the Arduino family.

For the project of “Remote Nursing”, the main purpose of Arduino is data acquisition and servomotor controller. It supplies an open-source platform that designers can program easily by getting support on open-source libraries from the Internet community. The language can be expanded through C language libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR-C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to. For this project, the C language is used for programming.

### 1.2.2 Raspberry Pi

The Raspberry Pi is a low cost, credit card sized, open-source microcontroller. It is developed by the Raspberry foundation in the UK. Due to the high speed processor in the Raspberry, it can do some more complicated tasks. The Raspberry Pi model used for this project is the Raspberry Pi Model B+ [3].

As shown in Appendix A.2, the Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC), which is a powerful ARM1176JZF-S Cortex-A7 based quad-core processor that runs at 900MHz. The board also features a memory capacity of 1Gbyte, a VideoCore IV GPU, and was originally equipped with 512 megabytes of RAM [3]. It has four USB ports that can connect to other modules such as the Wi-Fi module more easily. It also has 40-pin 2.54 mm expansion header including 27 GPIO pins and +3.3V, +5.0V, GND supply lines. The Raspberry does not have embedded storage but provides a MicroSD card slot to expand the storage to 32 GB at maximum.

Because the Raspberry Pi has a higher speed processor than the Arduino, the Raspberry Pi is used to stream the video remotely for this project. Thanks to the built-in VideoCore IV GPU, which is a low-power multimedia processor, and a 15-pin MIPI Camera Serial Interface (CSI-2), the Raspberry Pi can connect a video camera module and stream high

quality video. Because the Raspberry Pi has an embedded Linux-based system – the Raspbian, the Linux commands can be used on Raspberry Pi directly. The commands can be sent through a SSH (Secure Shell) or remote terminal). The driver for the Camera module is V4L2 (Video for Linux). With that driver, generic Linux applications written for cameras can be used [4].

### 1.2.3 Wi-Fi/802.11n Module

The Wi-Fi is a local area wireless computer networking technology that allows electronic devices to network, mainly using 2.5 GHz or 5 GHz radio bands. Compared to other standards for wireless communication such as the Bluetooth, it has a longer range and higher bit rate. Also in theory, the bandwidth of 802.11n can be from 54 Mbps to maximum 600Mbps [5]. The Wi-Fi module for this project is manufactured by Detroit DIY Electronics.

The Wi-Fi module used for this project is a USB Wi-Fi module with a monopole antenna. The frequency band of this Wi-Fi module is 2.5 GHz. The RF output power is from 13 to 17 dBm. It also has a small red LED to indicate the status of the connection. It can be connected to the Raspberry Pi' s USB port directly. In order to make the Wi-Fi module work properly, the driver for the Wi-Fi module was preinstalled in the Raspbian OS.

### 1.2.4 Bluetooth 4.0

Bluetooth is a kind of short-range, low-power communication technology with high-speed data rate and widely used in audio and compressed video applications.

For this project, a HC-06-S module is used as a Slave module. It is a Bluetooth serial interface module using BlueCore4-External chipset from Cambridge Silicon Radio. The S stands for the slave, so this module supplies only the Slave mode operations, which means it can be only used for getting commands from the computer. It helps to establish the communication between the Arduino and computer. The operating frequency range of this module is 2.4GHz to 2.48GHz. The data rate is 2.0Mbps, output power is from -4 to 6 dBm, sensitivity of it is -80dBm at 0.1% BER, and the operating voltage of it is 3.3V [6]. Compared to other Bluetooth versions, it has larger bandwidth and low power cost in its class.

A user can change the default setting of HC-06 module with the AT command via onboard button or “MODE” pin. Once the button pressed or “MODE” pin set as high voltage, two onboard LEDs would be turned on. The RED LED indicates link activity status and the GREEN LED indicates the pairing status.

### 1.2.5 Video Camera Module

The video camera module for this project is the Raspberry Pi 5 MP (MegaPixel) Camera module. This camera module is designed for the Raspberry Pi. This 5 MPcamera module is capable of 1080p video and still images and connects directly to the CSI (Camera



Serial Interface) port of the Raspberry Pi. As shown in Appendix A3, the camera for this module is the OV5674, a color 5-MP image provided by the OmniBSI technology. The maximum resolution can reach 2592\*1944 in theory. For the reason that it has the special CSI port for the Raspberry Pi, it is the best choice for this project [7].

### 1.2.6 Temperature Sensor & Humidity Sensor

The DHT11 Analog Temperature & Humidity Sensor is used with the Arduino for this project. This sensor board has a temperature sensor and a humidity sensor on it. The temperature sensor can measure the temperature range from 0 °C to 50 °C with +/- 2 °C accuracy, which includes the possible temperature range in a wardroom. The temperature sensor outputs a voltage linearly proportional to the Centigrade temperature [8]. The I2C digital humidity sensor is an accurate and intelligent device with an operating range that's optimized from 20% to 80% RH in a typical accuracy of ±2%. Compared to other same types sensor, it is lower cost and qualified to this project. The DHT11 Temperature & Humidity open-source library for the Arduino is shown in the Appendix A4.

### 1.2.7 Servomotor

The Hitec HS-311 servomotor is used with the Arduino for this project. It is low cost but provides sufficient performance and reliability for this project [9]. Its working voltage is ranges from 4.8V to 6.0V. And the motor's speed is 0.19 second/ 60 degree in clockwise direction. The HS-311 has three wires to be connected: the red wire is power, black wire is ground and the yellow wire is connected to the digital signal pin of the Arduino. The angular position of the HS-311 depends on the pulse width of the digital signal to the yellow wire. The servomotor open-source library for the Arduino is shown in the Appendix A5.

### 1.2.8 Computer

The computer for this project is the author's personal computer, which is an Apple Macbook Pro with operating system OS X Yosemite. The computer has an embedded Bluetooth 4.0 module and an embedded Wi-Fi 802.11n module. The project can be connected to any computer with the appropriate Bluetooth and Wi-Fi module.

## 1.3 SOFTWARE OVERVIEW

### 1.3.1 Arduino 1.6.5

This open-source Arduino Software (IDE) helps designer write code and upload code to the Arduino board directly. It also has a serial monitor working as an embedded terminal that can send commands to the board. This software can be downloaded freely and run under different operating system such as the Windows, the Mac OS X and the Linux. The computer language for this project is the C language. In order to acquire the data from the sensors and control the servomotor, the open source libraries for sensors and servomotor should be downloaded from open-source website [10].

### 1.3.2 Raspbian OS

Raspbian OS is an embedded operating system for the Raspberry Pi which is based on Debian, a Linux distribution [11]. Compared to the Arduino, the Raspberry Pi has a higher speed processor and a larger RAM. In this case, the Raspberry Pi can be controlled by the Linux commands rather than computer coding language by being installing this embedded operating system. Also some special applications should be installed on the Raspberry Pi directly such as the “VLC Linux version” [12] which is an application for using the video camera.

### 1.3.3 Application for Video Stream

The application installed on computer to stream the video is the VLC media player which is freely available from the VideoLAN organization. It is powerful enough to play the video in many formats such as the AVI, the MPEG, the MKV, and so on. The video format for this project is the H.264 AVC which can be played by the VLC. The VLC also has a function that plays the remote video. In this case, the VLC is a good choice for this project [13]. Through the VLC, the video can be decoded by the VideoCore IV GPU chip, and sent to the Wi-Fi module. Then the Wi-Fi module sends the data to the computer.

## II SYSTEM MODELING AND BASIC DESIGN

### 2.1 BIT RATE CALCULATION

Note in Eq.1 below, in order to stream video, the wireless module's bit rate has to be higher than the required bit rate.

Because of the VideoCore IV GPU chip of the Raspberry Pi, the only limitation of the project is the camera module OV5674.

$$\text{Required Bit Rate} = \text{Total Resolution (Pixels)} \times \text{ColorDepth(bits)} \times \text{Frame Rate} \quad (1)$$

$$\text{Total Resolution (Pixels)} = \text{Width Pixel} \times \text{Height Pixel} \quad (2)$$

According to the Table 1 [7], the video format for this project is H.264 AVC which video graphic array can be QVGA or VGA and the lowest resolution of the camera OV5674 is 320\*240, and the default resolution is 640\*480. The highest frame rate of the 320\*240 resolution can be 120 fps. The frame rate is the number of video frames per second being transmitted. The higher the frame rate, the higher bit rate is. On the contrary, lower frame rate would cause the video jerky and blurry. According to the author's testing, the lowest frame rate that can be used in this project is 10 fps at the resolution is 320\*240, and the default frame rate of the Raspberry Pi is "24".

In order to use the Remote Nurse properly, the project should use the color video, entry from Table 2 [14]. In this project the lowest color depth should be 12-bit color. By using Eq.1 and Eq.2.

The minimum uncompressed bit rate for this project

$$= 320 \times 240 \text{ pixels} \times 12 \text{ bits/pixels} \times 10 \frac{\text{frame}}{\text{s}} \approx 9.21 \text{ Mbit/s}$$

The default uncompressed bit rate for this project

$$= 640 \times 480 \text{ pixels} \times 12 \text{ bits/pixels} \times 24 \frac{\text{frame}}{\text{s}} \approx 88.5 \text{ Mbit/s}$$

In this case, the minimum bit rate of the wireless module has to be larger than 9.21 Mbit/s. Although in theory, the maximum bit rate of the Bluetooth 3.0 or Bluetooth 4.0 can reach 24 Mbit/s. That speed can be a little deceiving though, because the data is actually transmitted over a Wi-Fi (802.11) connection. The embedded Bluetooth module is only used to establish and manage a connection. And the Bluetooth devices are still limited to a maximum of 3 Mbps [16]. In real time testing, the bit rate of the Bluetooth module is not good enough to support the desired video stream. However, the 802.11n standard for the Wi-Fi which is our home use Wi-Fi has the bit rate from 54 Mbit/s to 600 Mbit/s. For this reason, using the Wi-Fi module is better choice for this project. And the project is updated to use the Wi-Fi 802.11n module.

However, in the future, in order to use lower bit rate for video stream, the video can be compressed. The Raspberry Pi creates H.264 compressed video from the camera. The H.264 video compression ratio is dependent on the frame-to-frame changes created by motion. Low motion results in higher compression ratio than high motion. Since video streams are not predictable, so exact calculation of compression ratio is not possible. According to a research paper on the compressed H.264 video from the Adobe company [14], depending on the video's motion rank, the lowest compressed video of H.264's bit rate can as low as

$$= 320 \times 240 \text{ pixels} \times 10 \frac{\text{frame}}{\text{s}} \times (1) \times 0.07 = 53.76 \text{ kbit/s}$$

And for the compressed default video, the bit rate can as low as

$$= 640 \times 480 \text{ pixels} \times 24 \frac{\text{frame}}{\text{s}} \times (1) \times 0.07 = 516 \text{ kbit/s}$$

The factor 0.07 bits/(frame\*pixel) was obtained experimentally. The factor (1) is the motion rank. Motion rank can be 1, 2, or 4, where 1 is used for video with a small amount of motion and 4 is used for video with a large amount of motion. The factor 1 is a good choice for patient monitoring. The number of bits/pixel used in [15] is not specified but is presumably 16 bits/pixel. This means the bit rate may be lower for compressed video.

Table 1. Resolution and Highest Frame Rate of the OV5674 Camera

format	resolution	frame rate
5 Mpixel	2592x1944	15 fps
1080p	1920x1080	30 fps
960p	1280x960	45 fps
720p	1280x720	60 fps
VGA	640x480	90 fps
QVGA	320x240	120 fps

Table 2. The Color Depth

Color Depth	Numbers of Colors	Display
1-bit color	$2^1 = 2$ colors	Monochrome, black and white
2-bit color	$2^2 = 4$ colors	CGA, gray-scale
3-bit color	$2^3 = 8$ colors	Early display, gray-scale
4-bit color	$2^4 = 16$ colors	EGA, between color and gray-scale.
5-bit color	$2^5 = 32$ colors	Original Amiga Chipset, between color and gray-scale.
6-bit color	$2^6 = 64$ colors	Original Amiga Chipset, between color and gray-scale.
8-bit color	$2^8 = 256$ colors	VGA at low resolution, between gray-scale and color.
12-bit color	$2^{12} = 4096$ colors	HAM model, color

## 2.2 PROJECT DESIGN

This project includes two parts: An Arduino based system and a Raspberry Pi based system. These two microcontrollers have their own advantages toward each other. Taking advantages of these two boards helps the project perform better. The Arduino board has a real-time and analog capability while the Raspberry Pi does not have. It helps the Arduino board be more flexible reading analog sensors. Also because the Arduino board does not have embedded operating system, the board has faster reaction time to sensors than the Raspberry Pi. However, the Raspberry Pi has a higher-speed processor, a larger RAM and special port for the Camera, so it performs better with the Camera module.

### 2.2.1 Arduino-based Part

As the figure below, the Arduino-based part is to communicate with the computer by the Bluetooth 4.0 module HC-06-S. As we measured before, this Bluetooth module can work well with 10 meters. The signal sent between Bluetooth is digital. A lot of desktops or laptops have built-in Bluetooth chips, so that the “Remote Nursing” Arduino Module can communicate with computers directly.

The Bluetooth module has two modes of operation, one is normal audio link mode, and the other is AT command mode. These two modes can be selected by setting high and low voltage to PIN 11. For this project, the normal audio link mode should be used. Each master/slave can be paired in this mode as default after being powered up. After connection established, data can be transmitted and received via UART interface with each other.

Installed with DHT-11 Temperature and Humidity sensor, the Arduino can acquire temperature and humidity data. After acquiring the analog data, the Arduino’s central processor can transfer the analog signal to digital signal, and then send to the Bluetooth shield. The Bluetooth module can send digital signal to computer. As input the word “M” in the serial monitor of the Arduino 1.6.5, the screen can display the indoor temperature and humidity.

With the help of the HS-311 servomotor, the position of the camera can be controlled through computer remotely. The angular position of the servomotor is decided by the pulse width of the digital control signal. As the digital sending to the servomotor in every 1000  $\mu\text{s}$  pulse width, the angular position is 0 degree; as the digital sending to the servomotor in every 1500  $\mu\text{s}$  pulse width, the angular position is 90 degree; and as the digital sending to the servomotor in every 2000  $\mu\text{s}$  pulse width, the angular position is 180 degree. Similar to the sensor control, the computer sends the digital signal to the Bluetooth module, and the Bluetooth module sends the digital signal to the Arduino. The Arduino can control the angular position of the servomotor by the frequencies of the digital signal.

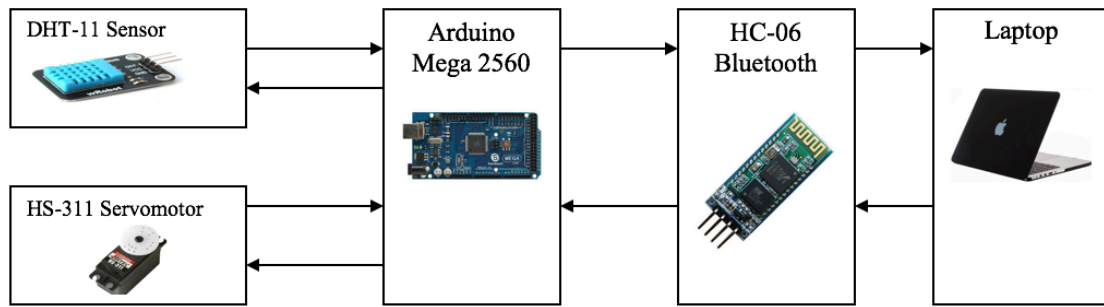


Figure 1. Arduino based System

### 2.2.2 Raspberry Pi-based Part

As the figure below, the wireless communication built between the Raspberry Pi and the computer is based on the Wi-Fi module. In order to build the connection through the Wi-Fi. Either the Raspberry Pi or the computer has to set itself as a virtual server and the other as a client. Normally, the computer can set itself as a hotspot and the Raspberry can connect to this hotspot. Once the communication is set-up, the computer can assign the Raspberry Pi a virtual IP address. Opening a Hyper-terminal software such as the putty.exe, inputting the Raspberry Pi's IP address and passcode, the user can control the Raspberry Pi remotely.

The main purpose of the Raspberry Pi is to stream the video. The OV5674 5MP camera module can capture the video and send the signal to the processor of the Raspberry Pi. And the Raspberry Pi can send the camera signal to the Wi-Fi module. The Wi-Fi module can encode the signal into the digital signal and send the digital to the Computer. By using the application such as the VLC, it can decode the digital signal back to the analog signal and stream it on the computer screen. For this reason, the people can monitor the wards remotely.

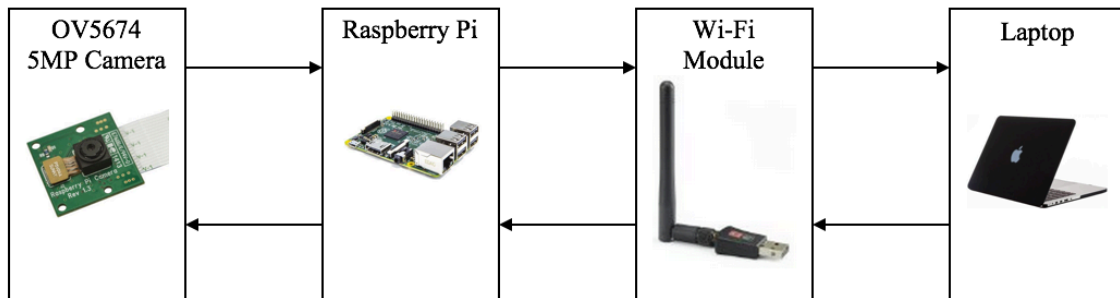


Figure 2. Raspberry Pi based System

### III SYSTEM MEASUREMENT

#### 3.1 POWER CONSUMING MEASUREMENT

##### 3.1.1 Arduino-Based System Power Consumption

Table 3 below is the power consumption of the Arduino board.

The first row of the table shows the Arduino board power consumption without any other sensors which is 276mW. The second row shows the power consumption of the Arduino board with the Bluetooth connecting to the computer which is 427mW. In this case, the Bluetooth module for the Arduino board power consumption is  $427\text{mW} - 276\text{mW} = 151\text{mW}$ .

The third row of table shows the power consumption of the Arduino board with the Bluetooth module and the DHT11 Temperature & Humidity sensor work together. It means the power consumption of the DHT11 Temperature & Humidity sensor is  $518\text{mW} - 427\text{mW} = 91\text{mW}$ .

The fourth row is the power consumption of the Arduino board with the Bluetooth module and the HS-311 Servomotor work together. In this case, the power consumption of the HS-311 Servomotor is  $935\text{mW} - 427\text{mW} = 508\text{mW}$ .

The last row is the power consumption of the whole system working together. The total consumption is approximately equal to column two + power consumption of the sensor + power consumption of the servomotor.

Table 3. Arduino Board Power Consuming

	Voltage (V)	Current (mA)	Power (mW)
Arduino	5.03	55	276
Arduino+ Bluetooth	5.03	85	427
Arduino+Bluetooth+Sensors	5.03	103	518
Arduino+Bluetooth+Servomotor	5.03	186	935
Arduino+Bluetooth+Sensors+Servomotor	5.03	205	1031



### 3.1.2 Raspberry Pi-Based System Power Consumption

Table 4 below is the power consumption of the Raspberry Pi-based system. The Raspberry Pi camera has numerous settings that can influence bit rate. The designer tested four here, first one is in low bit rate (9.21 Mbit/s, video resolution 320\*240, frame rate 10 fps), the second one is at bit rate 22.1Mbit/s (video resolution 320\*240, frame rate at 24 frames/s), the third one is at bit rate 36.9Mbit/s (video resolution 640\*480, and frame rate is 10 frames/s), and the last one is in default bit rate (88.5 Mbit/s, video resolution 640\*480, the frame rate 24 fps).

The first row shows the power consumption of the Raspberry Pi, Model B+ without any other modules working. The power consumption of the Raspberry Pi is 971mW.

The second row shows the power consumption of the Raspberry Pi with the Wi-Fi module working. But the Wi-Fi module does not send data to the computer. In this case, the power consumption of the Wi-Fi module is  $1369\text{mW} - 971\text{mW} = 398\text{mW}$ .

The third row shows the power consumption of the Raspberry Pi with the Wi-Fi module and the Camera module work together in low bit rate (9.21Mbit/s). It shows the power consumption with low bit rate is  $1484\text{mW} - 1369\text{mW} = 115\text{mW}$ . The video delay at this situation is 3s.

The forth row shows the power consumption of the Raspberry Pi with the Wi-Fi module and the Camera module work together. The resolution of the video is 320\*240, while the frame rate is 24 frames/s, which means the bit rate of the Wi-Fi module can be  $320*240*12*24 = 22.1$  Mbit/s. It shows the power consumption with default bit rate is  $1509\text{mW} - 1369\text{mW} = 140\text{mW}$ . The power consumption difference between this bit rates and lowest bit rate is  $(140\text{mW} - 115\text{mW}) / 115\text{mW} * 100\% = 21.7\%$ . The video delay at this situation is 2.5s.

The fifth row shows the power consumption as the video resolution is 640\*480, while the frame rate is 10 frames/s. It means the bit rate is  $640*480*12*10 = 36.9$  Mbit/s. The power consumption at this bit rate is  $1524\text{mW} - 1369\text{mW} = 155\text{mW}$ . The power consumption difference between this bit rates and lowest bit rate is  $(155\text{mW} - 115\text{mW}) / 115\text{mW} * 100\% = 34.8\%$ . The video delay at this situation is 3.5s.

The last row shows the power consumption at default bit rate (The resolution is 640\*480, and frame rate is 24 frame/s). At this bit rate the power consumption is  $1574\text{mW} - 1369\text{mW} = 205\text{mW}$ . The power consumption difference between lowest bit rates and default bit rate is  $(205\text{mW} - 115\text{mW}) / 115\text{mW} * 100\% = 78.2\%$ . The video delay at this situation is 2s.

Comparing row above, it turns out that as the bit rate changes, the power consumption changes also. It shows that the higher bit rate results in much higher power consumption.

Table 4. Raspberry Pi Board Power Consuming at different Bitrate

	Resolution	Frame Rate	Voltage (V)	Current (mA)	Power (mW)	Delay(s)
Raspberry Pi			4.98	195	971	
Raspberry Pi+ Wi-Fi			4.98	275	1369	
Raspberry Pi+ Wi-Fi + Camera Module	320*240	10	4.98	298	1484	3
Raspberry Pi+ Wi-Fi + Camera Module	320*240	24	4.98	303	1509	2.5
Raspberry Pi+ Wi-Fi + Camera Module	640*480	10	4.98	306	1524	3.5
Raspberry Pi+ Wi-Fi + Camera Module	640*480	24	4.98	316	1574	2

### 3.2 RANGE MEASUREMENT

In order to do a better range measurement, two antennas were used for the Wi-Fi module measurement. One is the normal home use monopole antenna, and the other one is the 900-2600 MHz PCB Log Periodic antenna as shown in figure [17]. This antenna can help the range extension due to its greater directivity compared to the monopole. The measurement was taken twice: one is in a big empty open-space parking lot and the other is measured through the wall.

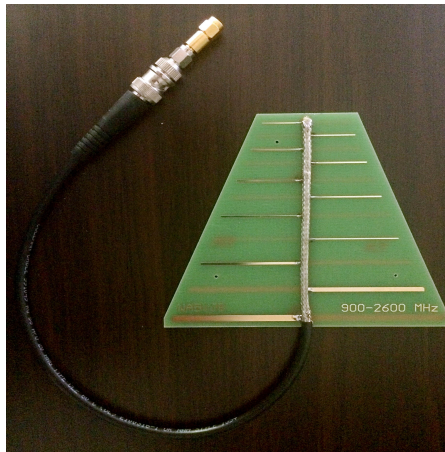


Figure 3. PCB-LP Antenna

Table 5. The Bluetooth and Wi-Fi Range

		Measurement 1 (m)	Measurement 2 (m)	Measurement 3 (m)	Average Value (m)
Open- Space	Wi-Fi+ Monopole Antenna	22.5	24	23	23.16
	Wi-Fi+ PCB-LP Antenna	27	28	27.5	27.5
	Bluetooth	7.9	8.2	8.0	8.03
Through- Wall	Wi-Fi+ Monopole Antenna	19	20	19.5	19.5
	Wi-Fi+ PCB-LP Antenna	23	24	23.5	23.5
	Bluetooth	5.6	6.5	6.0	6.03

## IV CONCLUSIONS

*“Engineering is for others’ better lives”* [18]. As an engineering student, we should have mission to use our knowledge to help others. In this case, I come out the idea of the “Remote Nursing”. This project is aiming to help the hospitals and the families to take care of patients better.

The main purpose of this project is to test video and sensors used remotely over the wireless module. It focuses on the integration of sensors on the microcontrollers and manually poll remote sensor data from the host using application software from the sensor developers. Automatic link multiplexing on the microcontroller was also omitted as a requirement or part of the statement of work. One of the key aspects of the project is to test which wireless module is appropriate for this type of application. So some testing of video quality vs range is included. Real-time video are demanding applications for a wireless sensor system so the project is focusing on testing the ability of which wireless module to support this.

As the world enters the wireless age, the knowledge on RF electronics and microcontroller is becoming more and more important. For this project, I used the knowledge on RF electronics, microcontroller, and computer C language that I learnt from both undergraduate and graduate studying.

During this project, the biggest issue for me is to figure out which wireless module can be more suitable to this project. At the beginning, I tried to use the Bluetooth module to stream the video remotely. However, because of Bluetooth’s low bit-rate, the performance is not sufficient for this project. Because of this, the Wi-Fi module is updated for video streaming. For the reason of the higher bit rate of the Wi-Fi module, the high quality video remotely.

Picking the better module for the project is a very important thing that I learnt from this project. Taking advantages of different modules such as the Arduino vs the Raspberry Pi, the Bluetooth module vs the Wi-Fi module can help the project perform better. And the users can get more benefits from the technology.

In order to get low power consumption, it is best to choose one processor. As Video performance is a limiting factor, the Raspberry Pi is better choice. However, because the Arduino board does not have an operating system, it has quicker reaction to sensor. On the other hand, the power consumption can be a concern in some undeveloped regions. In these regions, people can turn off the video stream on the Raspberry Pi and use partial functions on the Arduino by batteries power supplying, when they meet power-cut. This is the reason why two boards were chosen for this project.

In the next step for this project, the designer will focus on compressing the video and figure out the how to decrease the bit rate of the video delivery to the lowest. In this case, the project can be used on only one Arduino board, and people can use the whole functions with lowest power consumption.

On the other hand, the designer is intending to develop the software to integrate the sensor communication in the link, so that the system can act as a wireless medical platform. In this case, many other off-the-shelf medical devices which work over USB such as the thermometer, blood pressure monitor and heart bit sensor can be added to this system [19].

## V REFERENCES

- [1] VideoCore Architecture Reference Guide. (2013, September) Data Sheet. [Online] Available: <https://www.broadcom.com/docs/support/videocore/VideoCoreIV-AG100-R.pdf>
- [2] Arduino Mega 2560 [Online]. Available: <http://arduino.cc/en/Main/Arduinomega>
- [3] IBEX. Raspberry Pi Model B Hardware General Specification. [Online]. Available: <http://www.raspberry-projects.com/pi/pi-hardware/raspberry-pi-model-b/hardware-general-specifications>
- [4] The Raspberry Pi Camera Module. [Online]. Available: <http://www.ics.com/blog/raspberry-pi-camera-module>
- [5] IEEE 802.11n-2009—Amendment 5: Enhancements for Higher Throughput". IEEE-SA. 29 October 2009. doi:10.1109/IEEESTD.2009.5307322
- [6] Using the HC-06 Bluetooth Module. [Online]. Available: <http://mcuoneclipse.com/2013/06/19/using-the-hc-06-bluetooth-module/>
- [7] OV5674. Data Sheet. Data Sheet. [Online]. Available: [http://www.seeedstudio.com/wiki/images/3/3c/Ov5647\\_full.pdf](http://www.seeedstudio.com/wiki/images/3/3c/Ov5647_full.pdf)
- [8] DHT11 Humidity & Temperature Sensor. Data Sheet. [Online]. Available: <http://www.micropik.com/PDF/dht11.pdf>
- [9] HS 311 Standard. (2015). [Online]. Available: [https://www.servocity.com/html/hs-311\\_standard.html#.VaHLvVPF-Uc](https://www.servocity.com/html/hs-311_standard.html#.VaHLvVPF-Uc)
- [10] Getting with Arduino. (2015). [Online]. Available: <https://www.arduino.cc/en/Guide/HomePage>
- [11] About Raspbian. (2015). [Online]. Available: <https://www.raspbian.org/RaspbianAbout>
- [12] Streaming Video Using VLC Player. [Online]. Available: <http://www.raspberry-projects.com/pi/pi-hardware/raspberry-pi-camera/streaming-video-using-vlc-player>
- [13] Streaming features list: VLC Stream output. [Online]. Available: <http://www.videolan.org/streaming-features.html>
- [14] G.J. Sullivan; J.-R. Ohm; W.-J. Han; T. Wiegand (2012-05-25). "Overview of the High Efficiency Video Coding (HEVC) Standard" (PDF). Available: [http://iphome.hhi.de/wiegand/assets/pdfs/2012\\_12\\_IEEE-HEVC-Overview.pdf](http://iphome.hhi.de/wiegand/assets/pdfs/2012_12_IEEE-HEVC-Overview.pdf)
- [15] Amerasinghe. K. "H.264 For the Rest of Us". Adobe. [Online]. Available: [http://www.images.adobe.com/content/dam/Adobe/en/devnet/video/articles/h264\\_primer/h264\\_primer.pdf](http://www.images.adobe.com/content/dam/Adobe/en/devnet/video/articles/h264_primer/h264_primer.pdf)
- [16] Bluetooth Basic. [Online]. Available: <https://learn.sparkfun.com/tutorials/bluetooth-basics/common-versions>
- [17] 900-2600MHz PCB Log Periodic WA5VJB. Data Sheet. [Online]. Available: <http://www.wa5vjb.com/pcb-pdfs/LogPerio900.pdf>

- [18] Abron. L “Engineering Better Lives for Others” (2015). Washington University in St.Louis. [Online]. Available:  
<https://magazine.wustl.edu/2011/october/Pages/EngineeringBetterLives.aspx>.
- [19] Medical USB Platform. (2015). Texas Instrument. [Online]. Available:  
[http://www.ti.com/ww/en/embedded/medical\\_usb\\_stack/index.shtml?DCMP=msp-f66xx&HQS=f66xx-medicalusb-pr-sw](http://www.ti.com/ww/en/embedded/medical_usb_stack/index.shtml?DCMP=msp-f66xx&HQS=f66xx-medicalusb-pr-sw)

## VI APPENDICES

### A.1 Arduino Mega 2560 Datasheet

# Technical Specification

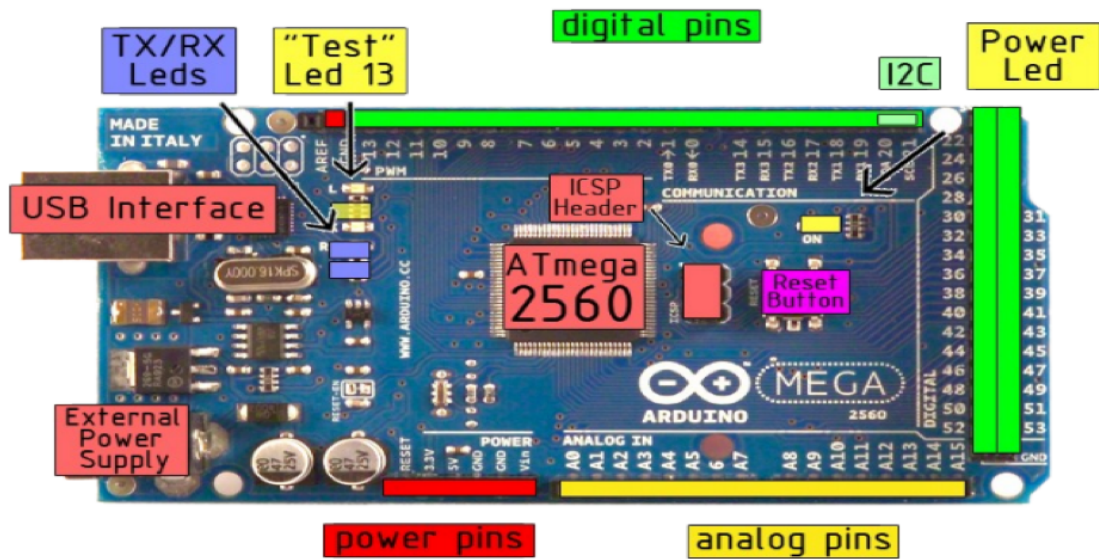


EAGLE files: [arduino-mega2560-reference-design.zip](#) Schematic: [arduino-mega2560-schematic.pdf](#)

## Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

## the board



*radiospares*

*RADIONICS*





## A.2 Raspberry Pi Model B+ Datasheet



### Raspberry Pi 2, Model B

**Product Name** Raspberry Pi 2, Model B

**Product Description** The Raspberry Pi 2 delivers 6 times the processing capacity of previous models. This second generation Raspberry Pi has an upgraded Broadcom BCM2836 processor, which is a powerful ARM Cortex-A7 based quad-core processor that runs at 900MHz. The board also features an increase in memory capacity to 1Gbyte.

#### Specifications

<b>Chip</b>	Broadcom BCM2836 SoC
<b>Core architecture</b>	Quad-core ARM Cortex-A7
<b>CPU</b>	900 MHz
<b>GPU</b>	Dual Core VideoCore IV® Multimedia Co-Processor Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
<b>Memory</b>	1GB LPDDR2
<b>Operating System</b>	Boots from Micro SD card, running a version of the Linux operating system
<b>Dimensions</b>	85 x 56 x 17mm
<b>Power</b>	Micro USB socket 5V, 2A

#### Connectors:

<b>Ethernet</b>	10/100 BaseT Ethernet socket
<b>Video Output</b>	HDMI (rev 1.3 & 1.4)
<b>Audio Output</b>	3.5mm jack, HDMI
<b>USB</b>	4 x USB 2.0 Connector
<b>GPIO Connector</b>	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
<b>Camera Connector</b>	15-pin MIPI Camera Serial Interface (CSI-2)
<b>JTAG</b>	Not populated
<b>Display Connector</b>	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
<b>Memory Card Slot</b>	Micro SDIO

### applications

- cellular phones
- toys
- PC multimedia
- digital still cameras

### ordering information

- **OV05647-G04A** (color, chip probing, 200  $\mu\text{m}$  backgrinding, reconstructed wafer)

### features

- 1.4  $\mu\text{m}$  x 1.4  $\mu\text{m}$  pixel with OmniBSI technology for high performance (high sensitivity, low crosstalk, low noise)
- optical size of 1/4"
- automatic image control functions: automatic exposure control (AEC), automatic white balance (AWB), automatic band filter (ABF), automatic 50/60 Hz luminance detection, and automatic black level calibration (ABLC)
- programmable controls for frame rate, AEC/AGC 16-zone size/position/weight control, mirror and flip, cropping, windowing, and panning
- image quality controls: lens correction, defective pixel canceling
- support for output formats: 8-/10-bit raw RGB data
- support for video or snapshot operations
- support for LED and flash strobe mode
- support for internal and external frame synchronization for frame exposure mode
- support for horizontal and vertical sub-sampling
- standard serial SCCB interface
- digital video port (DVP) parallel output interface
- MIPI interface (two lanes)
- 32 bytes of embedded one-time programmable (OTP) memory
- on-chip phase lock loop (PLL)
- embedded 1.5V regulator for core power
- programmable I/O drive capability, I/O tri-state configurability
- support for black sun cancellation

### key specifications

- **active array size:** 2592 x 1944
- **power supply:**
  - core: 1.5V  $\pm$  5% (with embedded 1.5V regulator)
  - analog: 2.6 ~ 3.0V (2.8V typical)
  - I/O: 1.7V ~ 3.0V
- **power requirements:**
  - active: TBD
  - standby: TBD
- **temperature range:**
  - operating: -30°C to 70°C (see [table 8-2](#))
  - stable image: 0°C to 50°C (see [table 8-2](#))
- **output formats:** 8-/10-bit RGB RAW output
- **lens size:** 1/4"
- **lens chief ray angle:** 24° (see [figure 10-2](#))
- **input clock frequency:** 6~27 MHz
- **S/N ratio:** TBD
- **dynamic range:** TBD
- **maximum image transfer rate:**
  - QSXGA (2592 x 1944): 15 fps
  - 1080p: 30 fps
  - 960p: 45 fps
  - 720p: 60 fps
  - VGA (640 x 480): 90 fps
  - QVGA (320 x 240): 120 fps
- **sensitivity:** TBD
- **shutter:** rolling shutter / global shutter
- **maximum exposure interval:** 1968 x  $t_{\text{ROW}}$
- **pixel size:** 1.4  $\mu\text{m}$  x 1.4  $\mu\text{m}$
- **well capacity:** TBD
- **dark current:** TBD
- **fixed pattern noise (FPN):** TBD
- **image area:** 3673.6  $\mu\text{m}$  x 2738.4  $\mu\text{m}$
- **die dimensions:** 5520  $\mu\text{m}$  x 4700  $\mu\text{m}$

## A.4 DHT11 Temperature & Humidity Sensor Library Code

```
// FILE: dht22.cpp
// VERSION: 0.1.00
// PURPOSE: DHT22 Temperature & Humidity Sensor library for
// Arduino
// DATASHEET:
// 0.1.0 by Rob Tillaart (01/04/2011)
// inspired by DHT11 library
#include "dht.h"
#define TIMEOUT 10000
int dht::read11(uint8_t pin)
{
    // READ VALUES
    int rv = read(pin);
    if (rv != 0) return rv;

    // CONVERT AND STORE
    humidity = bits[0]; // bit[1] == 0;
    temperature = bits[2]; // bits[3] == 0;

    // TEST CHECKSUM
    uint8_t sum = bits[0] + bits[2]; // bits[1] && bits[3] both 0
    if (bits[4] != sum) return -1;

    return 0;
}
int dht::read22(uint8_t pin)
{
    // READ VALUES
    int rv = read(pin);
    if (rv != 0) return rv;

    // CONVERT AND STORE
    humidity = word(bits[0], bits[1]) * 0.1;

    int sign = 1;
    if (bits[2] & 0x80) // negative temperature
    {
        bits[2] = bits[2] & 0x7F;
        sign = -1;
    }
    temperature = sign * word(bits[2], bits[3]) * 0.1;
    // TEST CHECKSUM
    uint8_t sum = bits[0] + bits[1] + bits[2] + bits[3];
    if (bits[4] != sum) return -1;

    return 0;
}
```

```

int dht::read(uint8_t pin)
{
    // INIT BUFFERVAR TO RECEIVE DATA
    uint8_t cnt = 7;
    uint8_t idx = 0;

    // EMPTY BUFFER
    for (int i=0; i< 5; i++) bits[i] = 0;

    // REQUEST SAMPLE
    pinMode(pin, OUTPUT);
    digitalWrite(pin, LOW);
    delay(20);
    digitalWrite(pin, HIGH);
    delayMicroseconds(40);
    pinMode(pin, INPUT);

    // GET ACKNOWLEDGE or TIMEOUT
    unsigned int loopCnt = TIMEOUT;
    while(digitalRead(pin) == LOW)
        if (loopCnt-- == 0) return -2;

    loopCnt = TIMEOUT;
    while(digitalRead(pin) == HIGH)
        if (loopCnt-- == 0) return -2;

    // READ THE OUTPUT - 40 BITS => 5 BYTES
    for (int i=0; i<40; i++)
    {
        loopCnt = TIMEOUT;
        while(digitalRead(pin) == LOW)
            if (loopCnt-- == 0) return -2;

        unsigned long t = micros();

        loopCnt = TIMEOUT;
        while(digitalRead(pin) == HIGH)
            if (loopCnt-- == 0) return -2;

        if ((micros() - t) > 40) bits[idx] |= (1 << cnt);
        if (cnt == 0) // next byte?
        {
            cnt = 7;
            idx++;
        }
        else cnt--;
    }

    return 0;
}

```

## A.5 Servomotor Library

```
#include <Servo.h>
Servo *Servo::first;
#define NO_ANGLE (0xff)
Servo::Servo() :
pin(0),angle(NO_ANGLE),pulse0(0),min16(34),max16(150),next(0)
{}
void Servo::setMinimumPulse(uint16_t t)
{
    min16 = t/16;
}

void Servo::setMaximumPulse(uint16_t t)
{
    max16 = t/16;
}

uint8_t Servo::attach(int pinArg)
{
    pin = pinArg;
    angle = NO_ANGLE;
    pulse0 = 0;
    next = first;
    first = this;
    digitalWrite(pin,0);
    pinMode(pin,OUTPUT);
    return 1;
}

void Servo::detach()
{
    for ( Servo **p = &first; *p != 0; p = &((*p)->next) ) {
        if ( *p == this ) {
            *p = this->next;
            this->next = 0;
            return;
        }
    }
}

void Servo::write(int angleArg)
{
    if ( angleArg < 0 ) angleArg = 0;
    if ( angleArg > 180 ) angleArg = 180;
    angle = angleArg;
    pulse0 = (min16*16L*clockCyclesPerMicrosecond() + (max16-
min16)*(16L*clockCyclesPerMicrosecond())*angle/180L)/64L;
}
```

```

uint8_t Servo::read()
{
    return angle;
}

uint8_t Servo::attached()
{
    for ( Servo *p = first; p != 0; p = p->next ) {
        if ( p == this) return 1;
    }
    return 0;
}

void Servo::refresh()
{
    uint8_t count = 0, i = 0;
    uint16_t base = 0;
    Servo *p;
    static unsigned long lastRefresh = 0;
    unsigned long m = millis();

    // if we haven't wrapped millis, and 20ms have not passed,
    then don't do anything
    if ( m >= lastRefresh && m < lastRefresh + 20) return;
    lastRefresh = m;

    for ( p = first; p != 0; p = p->next ) if ( p->pulse0)
count++;
    if ( count == 0) return;

    // gather all the servos in an array
    Servo *s[count];
    for ( p = first; p != 0; p = p->next ) if ( p->pulse0) s[i++]
= p;

    // bubblesort the servos by pulse time, ascending order
    for(;;) {
        uint8_t moved = 0;
        for ( i = 1; i < count; i++) {
            if ( s[i]->pulse0 < s[i-1]->pulse0) {
                Servo *t = s[i];
                s[i] = s[i-1];
                s[i-1] = t;
                moved = 1;
            }
        }
        if ( !moved) break;
    }
    for ( i = 0; i < count; i++) digitalWrite( s[i]->pin, 1);
}

```

```

uint8_t start = TCNT0;
uint8_t now = start;
uint8_t last = now;

// Now wait for each pin's time in turn..
for ( i = 0; i < count; i++) {
uint16_t go = start + s[i]->pulse0;

// loop until we reach or pass 'go' time
for (;;) {
    now = TCNT0;
    if ( now < last) base += 256;
    last = now;

    if ( base+now > go) {
digitalWrite( s[i]->pin,0);
break;
    }
}
}
}

```