Time Lapse Camera

Design Specification Document

Yuetian Feng

May.26th, 2014

fengyt@uw.edu

HCDE Capstone Project

University of Washington

Sponsor: Andrew Davidson

Table of Contents

1 – Introduction 3
1.1 – Scope 3
1.2 – Target Audience 3
2 – Functional Specifications 4
3 – Product specifications 4
4 – System Requirements 5
4.1 – System Overview 5
4.2 – Hardware requirements 6
4.3 – Software requirements 7
4.4 – Server structure & requirements 8
5 – Hardwire Configuration 8
6 – Software architecture 12
7 –Results 14
8 – User Interface Design (Optional) 15
9 – Conclusion 16
10 – Reference 16
11 – Appendix Program Code 17

1 – Introduction

Time-lapse photography is a technique whereby the frequency at which film frames are captured is much lower than that used to view the sequence (*Wikipedia*) [1]. It is usually used to record slow motion, such as sunset, growth of plants, etc. Using time-lapse photography technique as a documentation method can facilitate documentation and demonstration of a project, the meeting minutes, the recording of class activities or studio work sessions.

1.1 – Scope

This time lapse camera system capture a photo at a specific rate and then save the photos capture either locally to a SD card or wirelessly to a web server space. In this document, I will demonstrate how to design and build this camera system using Arduino UNO R3 [2] board as the microcontroller. Figure 1 shows the final deliverable of the time lapse camera prototype. All the stages in hardware and software design will be described in details.

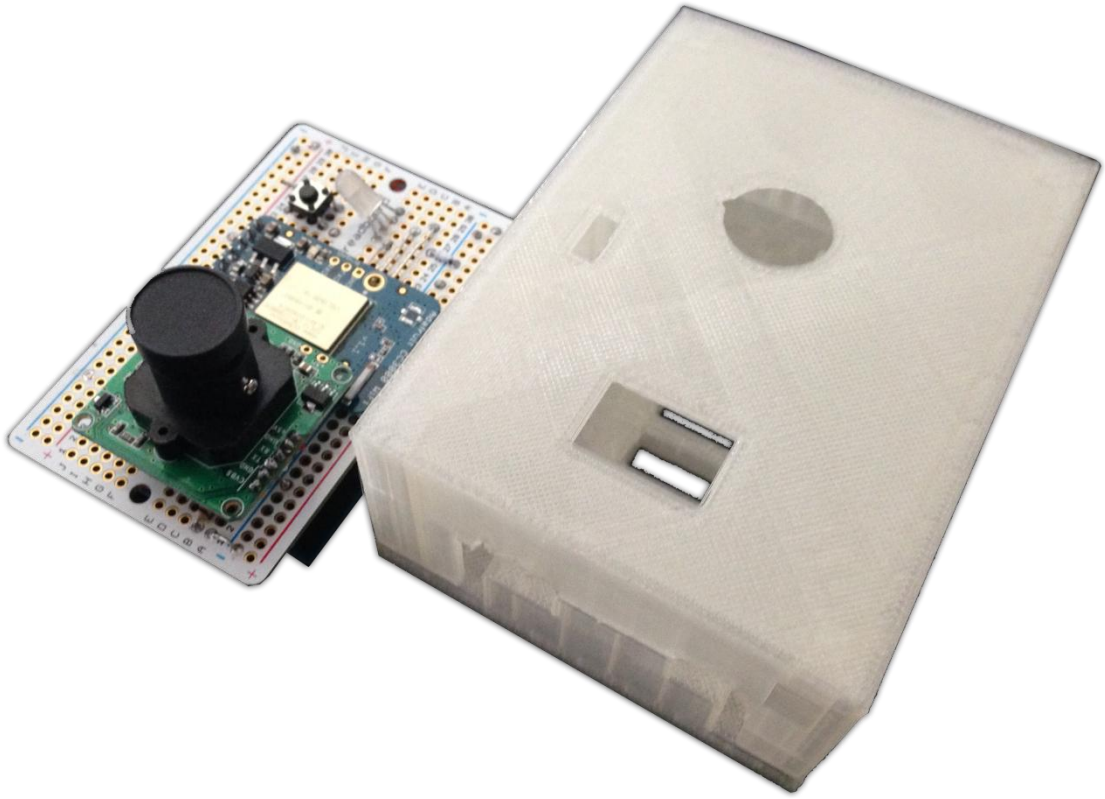

Figure 1 Final deliverable of time lapse camera

1.2 – Target Audience

The deliverables of the project can facilitate documentation and demonstration of a project, the meeting minutes, the recording of class activities or studio work sessions. The target user could be anyone who wants to record a long-time activity with. In addition, the project also serves as a case study for beginners interested in learning physical computing design and prototyping.

2 – Functional Specifications

The functions of this camera are the followings:

1. Time Lapse photo capture: capture a photo every 30 seconds or 1 minute.

2. Automatically upload photos captured to a web server space

3. Automatically save photos captured to a SD card

3 – Product specifications

The time interval and image size parameters are set in the software program, which can be changed according to requirements.

Time Interval	30 seconds, 1 minute
Module size	32mm x 32mm
Image sensor	CMOS 1/4 inch
CMOS Pixels	0.3M
Pixel size	5.6um*5.6um
Output format	Standard JPEG/M-JPEG
White balance	Automatic
Exposure	Automatic
Gain	Automatic
Shutter	Electronic rolling shutter
SNR	45DB
Dynamic Range	60DB
Max analog gain	16DB
Scan mode	Progressive scan
Viewing angle	60 degrees
Monitoring distance	10 meters, maximum 15meters (adjustable)
Image size	VGA (640*480), QVGA (320*240), QQVGA (160*120)

Baud rate	Default 38400
Current draw	75mA
Operating voltage	DC +5V
Communication	3.3V TTL

Table 1 Product specifications

4 – System Requirements

The system requirements for Time Lapse Camera system are presented in this section.

4.1 – System Overview

The Time Lapse Camera captures photos at specific time intervals and saves photos either locally to a SD card or wirelessly to an online web server. The system could be divided into four modules: Microcontroller, Photo Capture, Photo Saving and User Interaction/Feedback. Microcontroller drives the Photo Capture module to take photos and read them by binary stream. Microcontroller sends the binary stream data to the Photo Saving module. Photo Saving module sets up connection with SD card or web server, and then send the binary data received to the saving space without any modification. The User Interaction/Feedback module provide feedback and system status to users, it also provide a button for users to switch between different time lapse capture rates.

Due to that Arduino board has limited pins which couldn't support both Wi-Fi breakout board and SD card breakout board at the same time, in this document, I will introduce Wi-Fi photo uploading in detail. The SD card photo saving are similar but easier than Wi-Fi board.

4.2 – Hardware requirements

Figure 2 Arduino UNO R3 board

The project is based on Arduino platform, so an Arduino board is necessary for the project. Arduino is an open-source electronics prototyping platform providing both hardware and software. There are a number of boards available currently, such as Arduino Uno, Arduino Leonardo, Arduino Mega, etc. In this project, I used Arduino Uno R3. But you can choose the model you like with slightly changing the pins.

In addition to Arduino board, you'll need some other components. Figure 3 shows the components required in this project. All components could be purchased at *Adafruit.com*. The VC0706 TTL Serial Camera is used to perform the task of photo capture module. Adafruit CC3000 Wi-Fi breakout board is used to send photos to a web server. It has voltage regulators onboard, as well as an onboard antenna. The Micro SD card breakout board saves photo to the SD card. The RGB LED contains three LEDs-a red, a blue and a green. It could technically mix any colors, but actually only a couple colors could be easily differentiated from each other. The RGB LED signals users the current status of the camera system. The button is used to switch between two photo capture rates: 30s/photo or 1min/photo. Finally, you need some jumper wires and resistors to make the connections between the different parts.

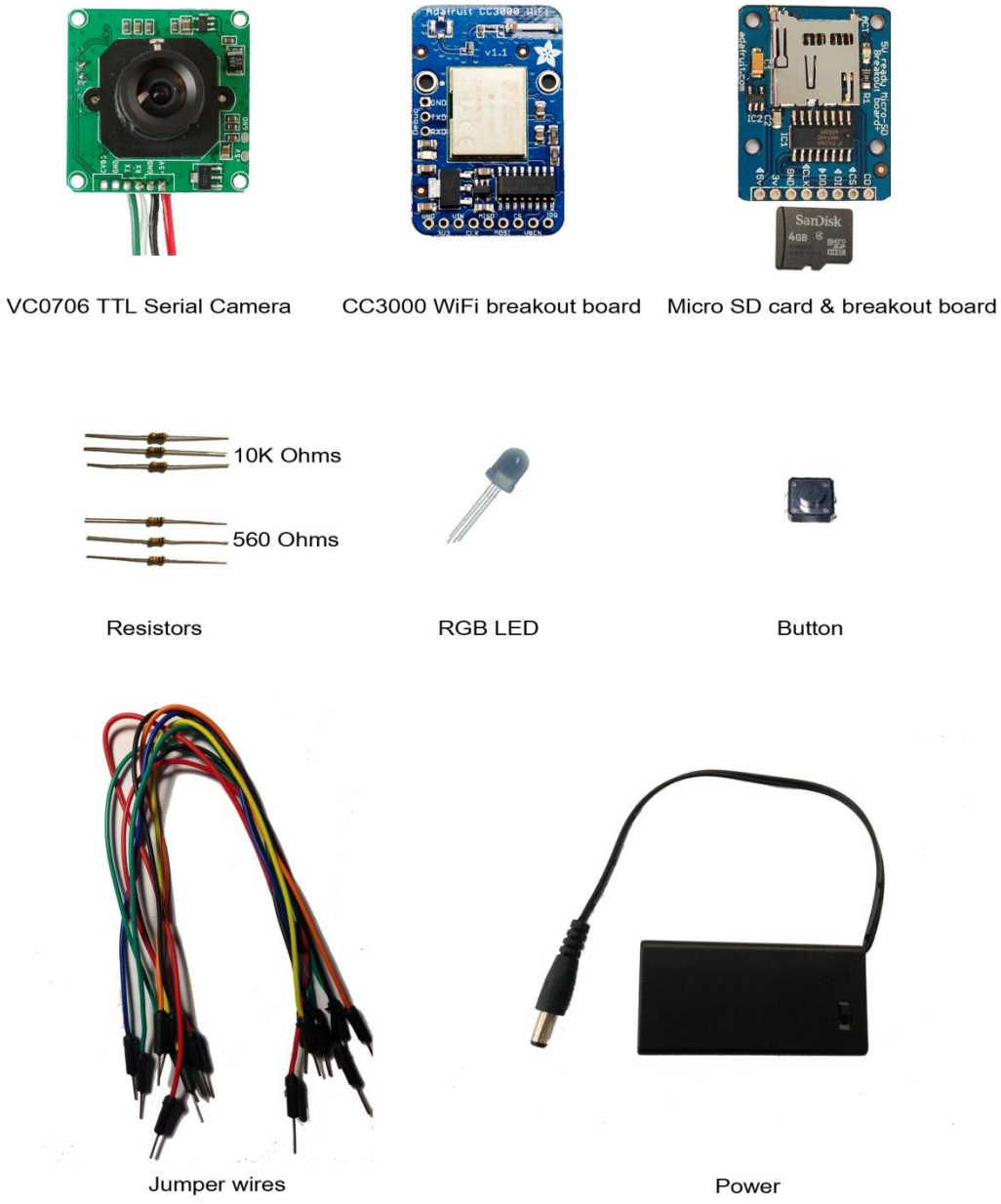

Figure 3 Hardware components required for this project

4.3 – Software requirements

This project uses Arduino IDE, which could be found at *Arduino.cc*. Adafruit's CC3000 library and VC0706 library are required for this project. To install a library, download the required folder, and put it into your /Arduino/libraries/ folder. To use this IDE, you simple input code into a new sketch, compile it and upload it to the Arduino board. The sketch tells Arduino—the microcontroller to drive other components to perform tasks. The program code of this project is available in Appendix.

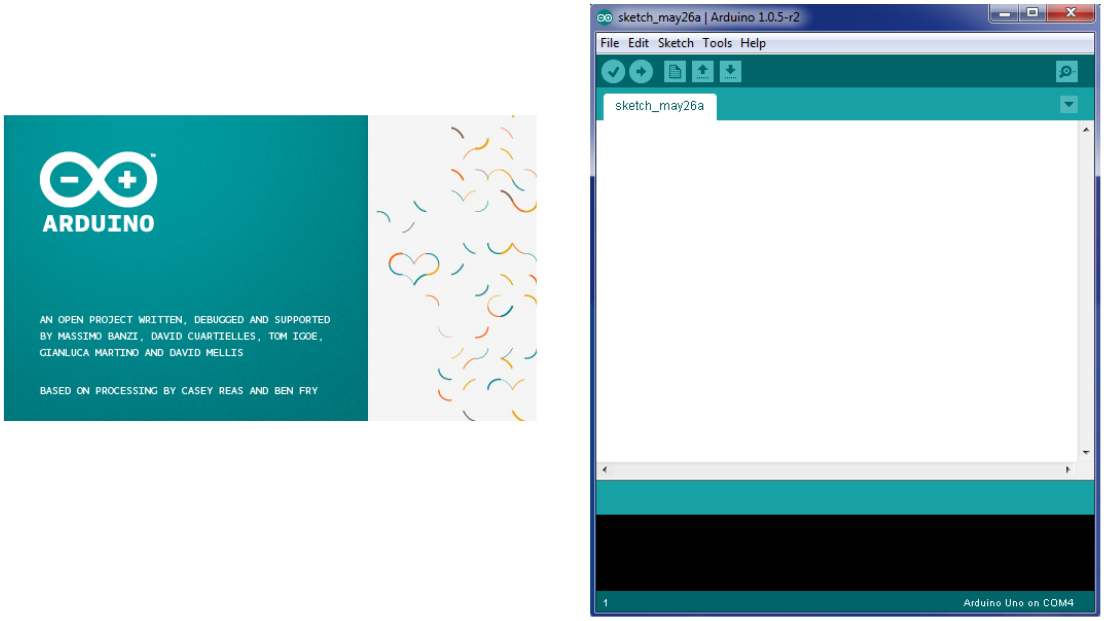

Figure 4 Arduino IDE

Since Arduino has a small memory, it couldn't cache a 12 kilobytes photo (The average photo size of photos captured by VC0706 TTL Serial Camera). Arduino only supports read photos with binary stream, and sends the binary data to Wi-Fi breakout board or Micro-SD card breakout board. The small memory issue complicates the photo saving task since a complete photo file couldn't generate first in Arduino. If you choose to save photos to the Micro-SD card, you'll need to open a new image file on the SD card, and then write the binary data of pixels into the file. If you choose to upload photos to a web server wirelessly, you'll need to configure the server first due to that the internet communication protocols have specific standards to be met. The only binary stream data transfer supported by Arduino excludes most protocols supporting file uploading.

4.4 – Server structure & requirements

If you choose to upload photos captured to a web server via Wi-Fi, one feasible solution is to upload photos via HTTP request. The CC3000 breakout board would set up a TCP connection with your web server and then send binary data stream with HTTP request. To send data with binary stream, a web server with Apache configuration to support "multipart/form-data" form file uploading is required for this project. You need to get a web server, a first-level domain name with independent IP address is suggested, and configure the Apache application.

5 – Hardwire Configuration

The hardware configuration is straightforward for this project. Adafruit provide detailed introduction of the VC0706 camera and CC3000 Wi-Fi breakout board configuration.

VC0706 TTL Serial Camera: There are six pins on the board; four of them are used in this project. Connect VCC to the 3.3V pin of the CC3000 board, and GND to the Arduino ground. The TX pin and RX pin could connect to any two digital pins of Arduino. In the sample code in Appendix, I connect the TX pin to the Arduino pin 7 using a voltage divider with the two 10K ohms resistors, following the picture below. Then, connect RX directly to the Arduino pin number 4.

CC3000 Wi-Fi breakout board: There are 8 pins on board. IRQ pin must connect to an interrupt pin of Arduino, so I connect the IRQ pin to Arduino pin 3. CC3000 uses Serial Peripheral Interface (SPI) to communicate with Arduino, so the MOSI, MISO, and CLK pins go to pins 11, 12, and 13 of Arduino, respectively. The power pins: Vin goes to the Arduino 5V, and GND to GND. The 3.3V pin has already connected to VC0706 camera. The remaining two pins could be chosen randomly. I connected VBAT to pin 5, and CS to pin 6.

Pushbutton: Pushbuttons or switches connect two points in a circuit when you press them. A 10k Ohms resistor is used. The button must connect to a interrupt pin of Arduino, so I connected one point with Arduino pin 2. If the button is pressed, there will be current in the circuit when two points are connected.

RGB LED: A RGB LED contains three LEDs pins: a red, a blue and a green. Connect these pins to Arduino pin 10, 9, 8, respectively with three 560 Ohms resistors.

Figure 5 and 6 show breadboard view and schematic view of the hardware connections.

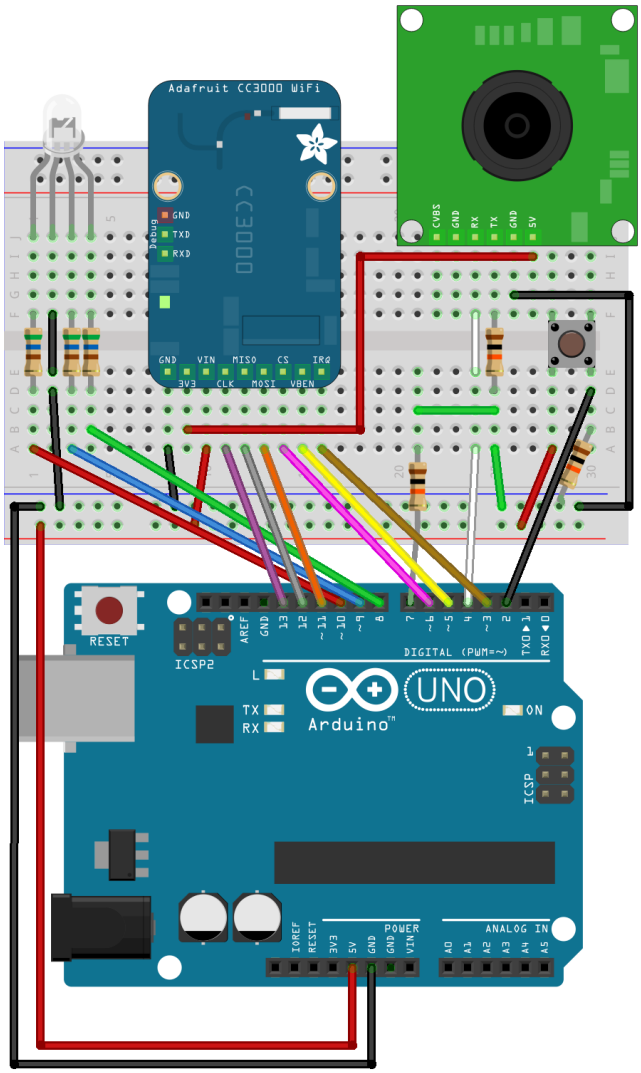

Figure 5 Breadboard view of the camera prototype

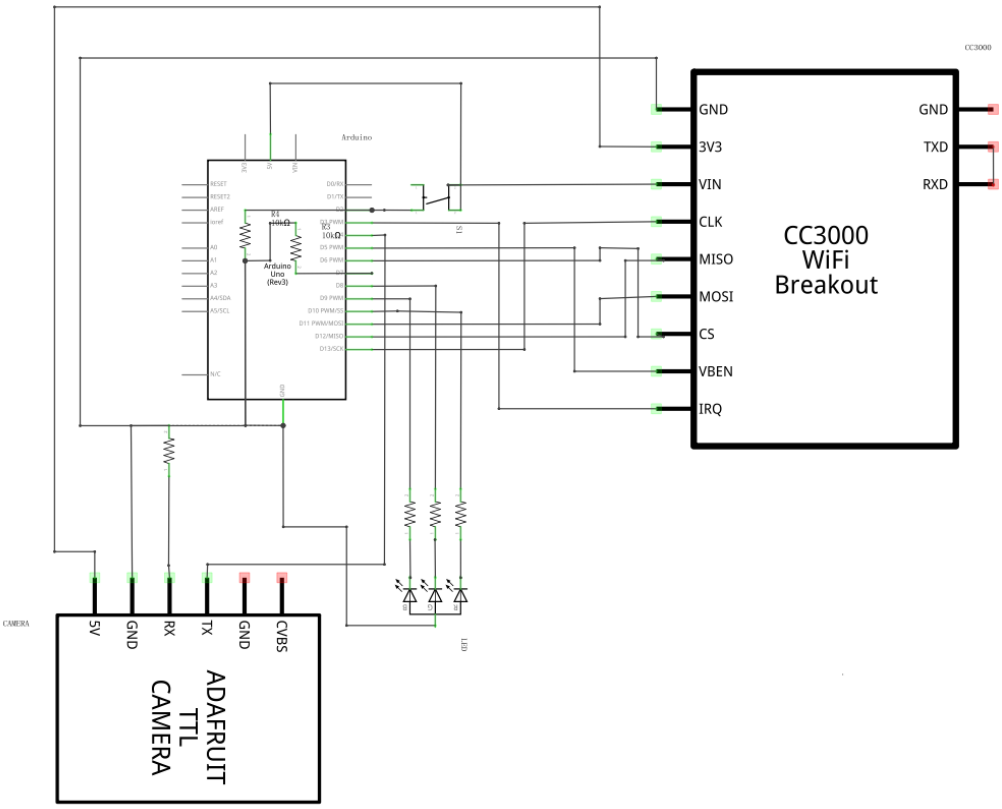

Figure 6 Schematic view of the camera prototype

Figure 7 shows the actual camera prototype.

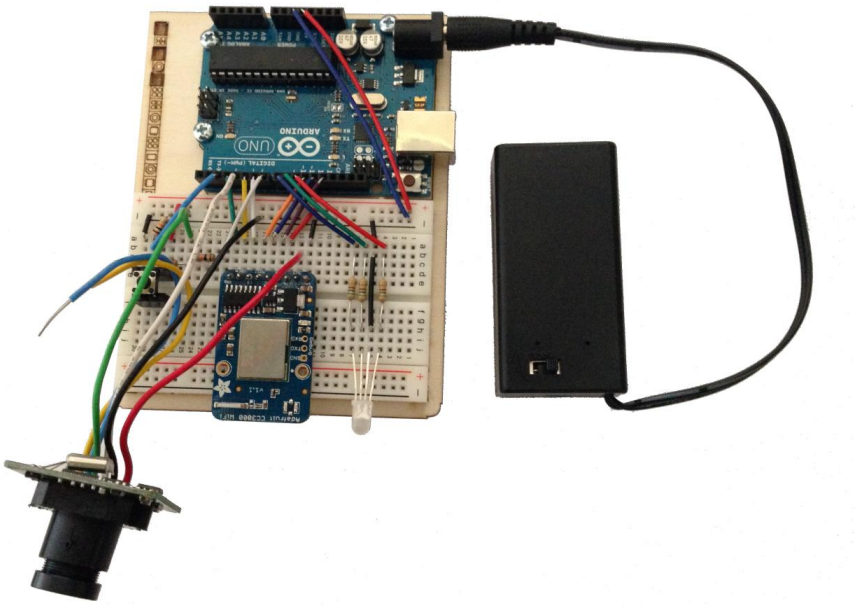

Figure 7 Camera prototype

6 – Software architecture

After the hardware configuration, write the program to drive the components. As I stated earlier, both camera and Wi-Fi library should be imported to the program. Then create objects to use both components on the right pins as configured before.

Photo Capture: Check if the camera could be initialized.

```
if (cam.begin()) {
    Serial.println("Camera found:");
  } else {
    Serial.println("Camera not found !");
    return;
  }
```

You can set the image size here (Three options: 640*480, 320*240 and 160*120):

```
cam.setImageSize(VC0706_640x480);
```

Take a photo by

```
    if (! cam.takePicture()){
      Serial.println("Failed to snap!");
    }
    else {
      Serial.println("Picture taken!");
    }
```

Wi-Fi uploading: To send data via Wi-Fi, you need to connect to a router nearby first and successfully set up a TCP connection.

```
cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY);

Adafruit_CC3000_Client client = cc3000.connectTCP(ip, port);
```

The multipart/form-data content type file uploading has a specific format for binary uploading. After successfully connects to your web server, a HTTP request to set the HTTP header information must be sent first. A boundary string signals the server of the start and end of the uploading. I used “--frontier" as the boundary string.

```
start_request = start_request + "\n" + "--frontier" + "\n" + "Content-Disposition: form-data; name=\"file\"; filename="+filename+"\n" + "Content-Type: image/jpeg" + "\n" + "Content-Transfer-Encoding: binary" + "\n" + "\n";

end_request = end_request + "\n" + "--frontier--" + "\n";
```

After the HTTP request, the length of the HTTP request must be sent to the server.

```
uint16_t jpglen = cam.frameLength();

uint16_t extra_length;

   extra_length = start_request.length() + end_request.length();
   uint16_t len = jpglen + extra_length;
```

Then you can send the binary data of photos. Due to Arduino's memory limits, a buffer must be used to read part of code and send to web server and read another part of code again.

```
byte wCount = 0;
      while (jpglen > 0) {
        uint8_t *buffer;
        uint8_t bytesToRead = min(32, jpglen);
        buffer = cam.readPicture(bytesToRead);
        client.write(buffer, bytesToRead);

        jpglen -= bytesToRead;
      }
      client.print(end_request);
      client.println();
```

Light feedback: The RGB LED mixes the color signals. Call this function to set colors.

```
void setColor(int red, int green, int blue)
{
 #ifdef COMMON_ANODE
   red = 255 - red;
   green = 255 - green;
   blue = 255 - blue;
 #endif
 analogWrite(redPin, red);
 analogWrite(greenPin, green);
 analogWrite(bluePin, blue);
}
```

Time Interval Switch: The button could invoke the interrupt function—changeRate()—to change the time interval between two photos captured.

```
volatile boolean state = false;
attachInterrupt(0, changeRate, FALLING);
void changeRate(){
 state = !state;
 Serial.println(state);
}
```

The complete program code is available in Appendix.

7 –Results

I tested the camera prototype with my personal web server. Figure 8, 9 and 10 show the test results captured.

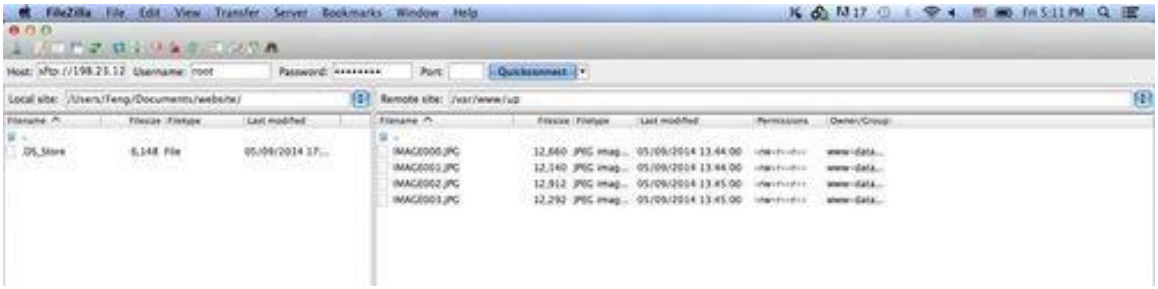

Figure 8 Photos uploaded to web server wirelessly

Figure 9 Photos downloaded from web server

8 – User Interface Design (Optional)

You can also solder the camera prototype together after it successfully performs the tasks. Figure 8 is my soldered result.

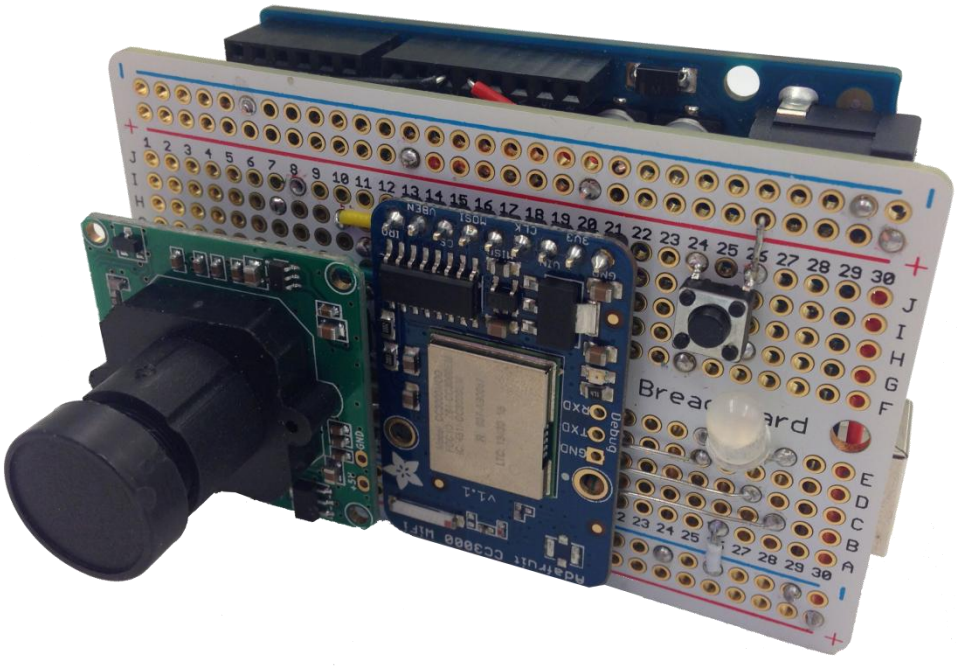

Figure 8 the camera prototype 2

If you prefer a clean user interface, consider making a case for the camera. I used Makerbot 3D printer created a case for the camera prototype (Figure 9).

Figure 9 Final version of the camera prototype

9 – Conclusion

This document demonstrates the design and implementation of a Time Lapse Camera prototype. A project weekly blog is available here if you need further information. After building the prototype, I find that although Arduino is a great platform to get to the interactive physical computing world, it has limited memory and hard drive. It cannot save a 16 kilobytes sketch nor cache a 12 kilobytes photo file, which complicates the project. A more advanced circuit board, such as Raspberry Pi, would be a wiser choice. Another option is to use the Eye-Fi SD card, it could automatically upload photos saved in SD card to an online space since the SD card saving is much more stable than Wi-Fi uploading. Also, Wi-Fi photo uploading generates low quality/error photos more frequently than SD card photo saving. I posted a couple of questions in Arduino forum but couldn't get a positive feedback.

10 – Reference

[1] Time lapse photography, Wikipedia definition. No original sources found.
http://en.wikipedia.org/wiki/Time-lapse_photography

[2] Arduino UNO R3 official introduction: http://arduino.cc/en/Main/arduinoBoardUno

11 – Appendix Program Code

```
/*************************************************
  This is a sketch for a Time Lapse Camera project
  Author: Yuetian Feng
  Functions implemented:
  1. Photo Capture at a specific speed
  2. Upload photos to a web server
  3. Could switch between two photo capture rates: 30s/photo and 1min/photo
  4. Four light colors to signal users of current status:
     Red--Initializing
     Blue--Photo uploading
     Green--Waiting status of 30s interval
     Yellow--Waiting status of 1min interval

  Components:
  1. a VC0706 TTL Serial Camera
  2. an Adafruit CC3000 Wi-Fi breakout board
  3. an Arduino board
  4. a RGB LED
  5. a button
  6. resistors,jumper wires and battery
 *************************************************/

#include <Adafruit_VC0706.h>
#include <SoftwareSerial.h>
#include <Adafruit_CC3000.h>
#include <ccspi.h>
#include <SPI.h>
#include <string.h>
#include "utility/debug.h"
#include<stdlib.h>

//Interrupt invoke by pressing the button
int redPin = 10;
int greenPin = 9;
int bluePin = 8;
```

```
volatile boolean state = false;

int32_t interval1 = 30000;//30s
int32_t interval2 = 60000;//1min

int32_t waitingTime = interval1;


// VC0706 Camera connection Software serial
SoftwareSerial cameraconnection = SoftwareSerial(7, 4);
Adafruit_VC0706 cam = Adafruit_VC0706(&cameraconnection);

// Define CC3000 chip pins
#define ADAFRUIT_CC3000_IRQ   3
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    6


// WiFi network (change with your settings !)

#define WLAN_SSID       "Your network"
#define WLAN_PASS       "Password"
// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA or
WLAN_SEC_WPA2
#define WLAN_SECURITY   WLAN_SEC_WPA2


// Create CC3000 instances
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT,
                                SPI_CLOCK_DIV2);
uint32_t ip = cc3000.IP2U32(198,0,0,1)//input your web server IP address
int port = 80;

char filename[14] = "IMAGE000.JPG";
int i = 0;
Adafruit_CC3000_Client client;
```

```
void setup() {

  Serial.begin(115200);
  setColor(255, 0, 0);//Red--Initializing
  //Set up button interrupt
  attachInterrupt(0, changeRate, FALLING);

  Serial.println("Camera test");

  // Try to locate the camera
  if (cam.begin()) {
    Serial.println("Camera found:");
  } else {
    Serial.println("Camera not found !");
    return;
  }

  // Set picture size, could change to other sizes
  cam.setImageSize(VC0706_640x480);

  // Initialise the module
  Serial.println(F("\nInitializing..."));
  if (!cc3000.begin())
  {
    Serial.println(F("Couldn't begin()! Check your wiring?"));
    while(1);
  }

  // Connect to  WiFi network
  cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY);
  Serial.println(F("Connected!"));

  // Display connection details
  Serial.println(F("Request DHCP"));
  while (!cc3000.checkDHCP())
  {
    delay(100);
  }

  Serial.println("Ready");
  setColor(0,0, 255);  // gGreen--Waiting
  delay(3000);
}
```

```
void loop() {

   if (! cam.takePicture()){
     Serial.println("Failed to snap!");
   }
   else {
     Serial.println("Picture taken!");
     setColor(255,255,0);//purple--snap!
     delay(100);
   }

   //calculate uploading time
   int32_t time = millis();

   // Get the size of the image (frame) taken
   uint16_t jpglen = cam.frameLength();
   filename[5] = '0' + i/100;
   filename[6] = '0' + i%100/10;
   filename[7] = '0' + i%100%10;

   // Prepare HTTP request
   String start_request = "";
   String end_request = "";
   start_request = start_request + "\n" + "--frontier " + "\n" + "Content-Disposition:
form-data; name=\"file\"; filename="+filename+"\n" + "Content-Type: image/jpeg" +
"\n" + "Content-Transfer-Encoding: binary" + "\n" + "\n";
   end_request = end_request + "\n" + "--frontier --" + "\n";

   uint16_t extra_length;
   extra_length = start_request.length() + end_request.length();
   uint16_t len = jpglen + extra_length;

   // Set up TCP connection with web server
   client = cc3000.connectTCP(ip, port);
    if (client.connected()) {
      Serial.println("Start uploading...");
      setColor(0, 255, 0);//blue--Photo Uploading
      client.println(F("POST /camera.php HTTP/1.1"));
      client.println(F("Host: 198.0.0.1:80"));//HTTP port 80
      client.println(F("Content-Type: multipart/form-data; boundary=AaB03x"));
      client.print(F("Content-Length: "));
      client.println(len);
      client.print(start_request);
```

```
      // Send binary data
      byte wCount = 0;
      while (jpglen > 0) {

        uint8_t *buffer;
        uint8_t bytesToRead = min(32, jpglen);
        buffer = cam.readPicture(bytesToRead);
        client.write(buffer, bytesToRead);

        jpglen -= bytesToRead;
      }

      client.print(end_request);
      client.println();

    }
    else{

     Serial.println("Web server connection failed");

    }

    client.close();
    Serial.println("done...");

    time = millis() - time;
    Serial.print(time); Serial.println(" ms elapsed");


    i++;
    cam.resumeVideo();


    if(state){
      setColor(255, 0,165);//orange--Waiting for 1min interval
      waitingTime = interval2;
    }
    else{
      setColor(0, 0, 255);  // green--Waiting for 30s interval
      waitingTime = interval1;
    }

    delay(waitingTime - time);

}
```

```
//Invoke when button pressed
void changeRate(){

  state = !state;
  Serial.println(state);

}

//Set color of RGB LED
void setColor(int red, int green, int blue)
{
  #ifdef COMMON_ANODE
    red = 255 - red;
    green = 255 - green;
    blue = 255 - blue;
  #endif
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```