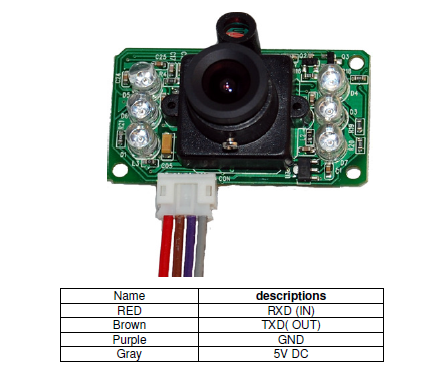
**LS-Y201-Infrared Camera**

**Introduction**

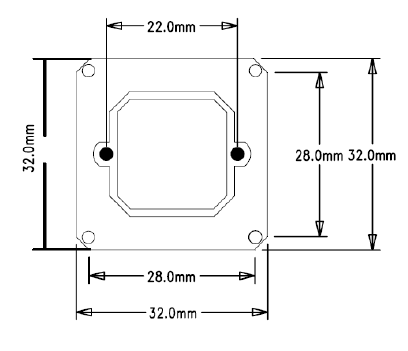


LS-Y201-Infrared camera is a new generation digital camera from LinkSprite. This module provides a Universal Asynchronous Receiver Transmitter (UART) interface. This module captures high resolution images and transmits the images in JPEG format. The JPEG format is encoded in ASCII. The module has the additional feature of capturing pictures in the dark. The module has a built-in sensor to determine the amount of ambient light. This sensor is used to automatically switch on the infra-red LEDs when ambient light is below a certain minimum (dark). When below a certain minimum the module begins capturing infra-red images. The camera module also provides the feature of adjusting the resolution and Baud rate. This is done by sending commands to the camera using the UART interface. This camera has a default capturing resolution of 640x480 and a default baud rate of 38400 Baud.

**Specifications**

|  |  |
| --- | --- |
| **Specifications** |  |
| Resolutions | VGA (640x480)/QVGA (320x240)/ infra-red (160x120) |
| Resolution Configuration | Hexadecimal configuration bytes are sent |
| Power | 3.3 V DC or 5 V AC |
| Size | 32mm X 32mm |
| Current | 80-100mA and 200mA for infra-red shots |
| JPEG file | JPEG starts with FFD8 and ends with FFD9 |
| Communication | 3.3 V level TTL UART communication |
| Baud Rate | 9600/19200/38400(default)/57600/115200 |

**Dimensions**



**Communication Pins**



**Communication**

Communication with the camera module is done by sending instructions to the camera. Below shows a list of instructions and there return values all in hexadecimal.

1. Reset



This instruction resets camera to default configurations.

2. Take picture

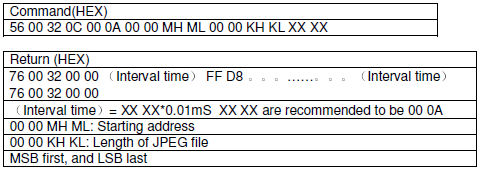


3. Read JPEG file size



XH XL is the file length of JPEG file. MSB is in the front and followed by LSB

4. Read JPEG file



Start reading the JPEG file at 00 00. A chuck size must be chosen of an integer time of 8. The chunk must be read as many times until FF D9 is read. FF D9 is indicates the end of the file.

5. Stop taking pictures

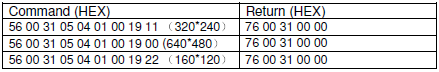


6. Compression Ratio

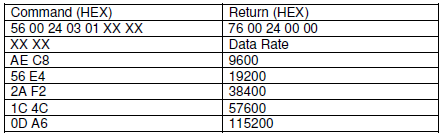


XX: 0x00 to 00xFF

7. Image Size



9. Change Baud Rate



**Note**

When powered on, serial port will output the following message actively:

Ctrl infr exist

User-defined sensor

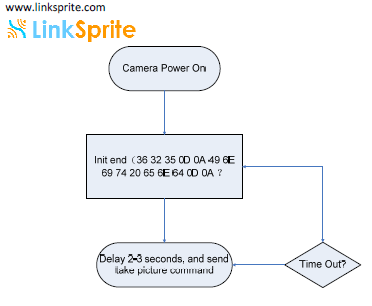
625

Init end

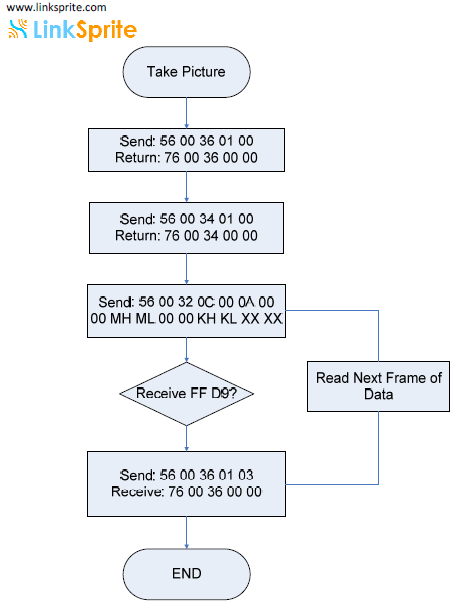
The host only needs to determine when “Init end”（36 32 35 0D 0A49 6E 69 74 20 65 6E 64 0D 0A) is received. After “Init end” is received, the host can send the take picture command after waiting for another 2-3 seconds.

**Flow Charts**

**Power on**



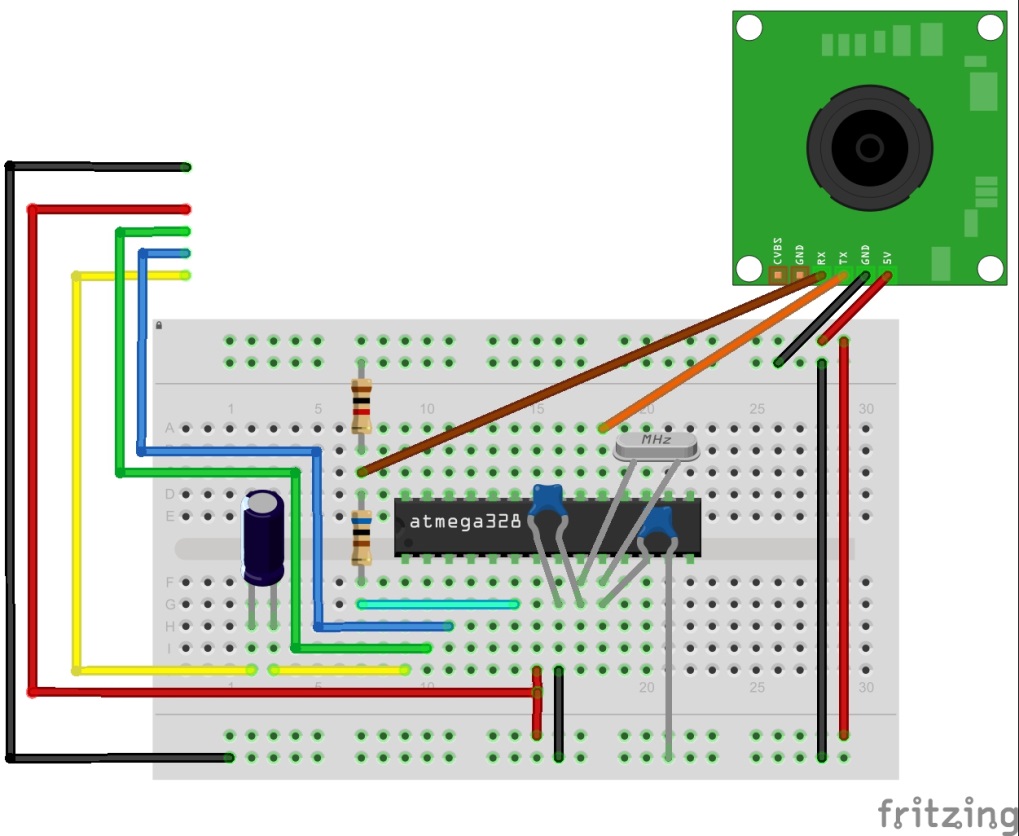
**Capture Image**



**Test Procedure**

**Schematics**

Below is the schematic for operating the LS-Y201 camera. The schematic was built using fritzing PCB software. There are two resistors in the circuit, one being 1000 ohms and the other being 600 ohms. The reason for the two resistors is to create a voltage divider circuit. The camera has 3.3 V TTL level UART interface. This means that the RX pin can only receive a maximum of 3.3 V. The digital output pins of the Arduino output 5±0.5 Volts. It is important the voltage divider circuit is present to prevent damaging the camera. The RX pin receives around 3 V with the voltage divider present.



**Components**

* Bread board
* Jumper cables
* Two 10 kΩ resistors
* Arduino

**Power up**

Check whether the camera module is functional. The following code below is code used to capture the message that is sent to from the camera module to the ATmega328-PU.

**Code**

#include <SoftwareSerial.h>

//Here a virtual UART interface is defined using normal digital //input/output pins.

//Not all pins on the Mega and Mega 2560 support change interrupts, so only //the following can be used for RX: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, //A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15 //(69).

//The rx pins will be 10 and the tx pin will be tx.

//SoftwareSerial(rx of arduino, tx of arduino)

SoftwareSerial mySerial **=** SoftwareSerial**(**13**,** 4**);**

void setup**()**

**{**

//This initialises a serial port that will connect the arduino to a computer via USB at Baud rate of 19200

Serial**.**begin**(**19200**);**

//This initialises a serial port that will connect the arduino to the camera via the rx and tx pins at Baud rate of 38400

mySerial**.**begin**(**38400**);**

**}**

void loop**()**

**{**

//Checks is TTL data is coming from the camera to the arduino

**if(**mySerial**.**available**()** **>** 0 **){**

//Read string values coming in and store in x.

String x **=** mySerial**.**readString**();**

//Send x on Serial line to computer to be displayed

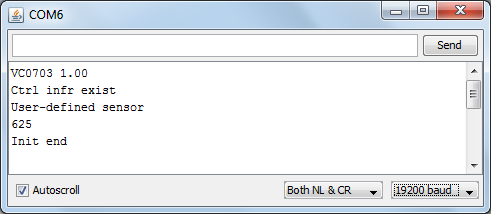
Serial**.**println**(**x**);**

**}**

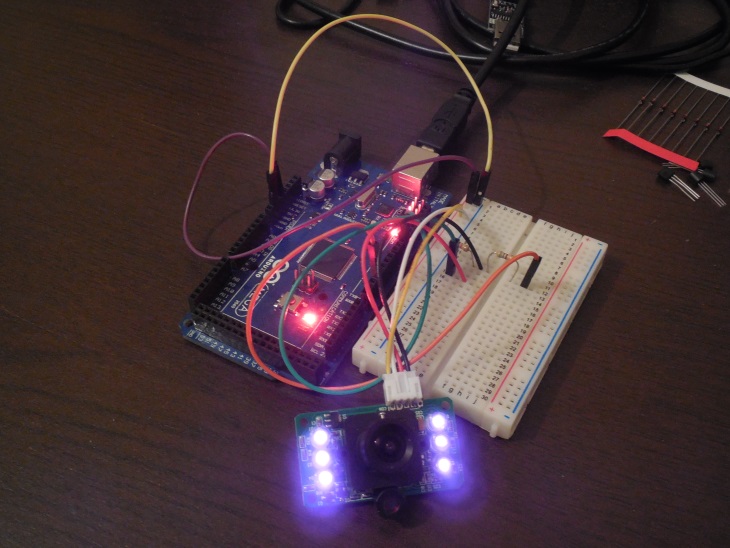
**}**

**Serial Monitor**

Run the code shown above. Open Serial monitor from the Arduino compiler. Set the Baud rate to 19200 Baud. The serial monitor will output what is shown below. The message displays the driver installed, shows that the infra-red and camera sensor exist and shows initialisation values have been initialised.



**Take a picture**



Below is code that used to take a single picture from the camera. The code contains the commands which are sent to the camera to take pictures. The code will output ASCII hexadecimal values onto the serial monitor. The hexadecimal values can be copied and pasted into a text file. Using a programming language such as python or java the string values in the txt file can be converted to ASCII binary and saved in a .jpg file to be displayed as an image.

**Code**

/\* Linksprite \*/

#include <SoftwareSerial.h> // A arduino

byte incomingbyte**;**

//Here a virtual UART interface is defined using normal digital output pins.

//The rx pins will be 10 and the tx pin will be tx.

SoftwareSerial mySerial **=** SoftwareSerial**(**10**,** 3**);** //SoftwareSerial(rx of arduino, tx of arduino)

void setup**()**

**{**

//This initialises a serial port that will connect the arduino to a computer via USB at Baud rate of 19200

Serial**.**begin**(**19200**);**

//This initialises a serial port that will connect the arduino to the camera via the rx and tx pins at Baud rate of 38400

mySerial**.**begin**(**38400**);**

**}**

void loop**()**

**{**

**if(**mySerial**.**available**()** **>** 0 **){** //Checks is TTL data is coming from the camera to the arduino

String x **=** mySerial**.**readString**();** //Read string values coming in and store in x.

Serial**.**println**(**x**);** //send x on Serial line to computer to be displayed

**}**

**}**/\* Linksprite \*/

#include <SoftwareSerial.h>

//Here a virtual UART interface is defined using normal digital //input/output pins.

//Not all pins on the Mega and Mega 2560 support change interrupts, so only

//the following can be used for RX: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53,

//A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15

//(69).

//The rx pins will be 10 and the tx pin will be tx.

//SoftwareSerial(rx of arduino, tx of arduino)

SoftwareSerial mySerial **=** SoftwareSerial**(**10**,** 3**);** //SoftwareSerial(rx of arduino, tx of arduino)

int a**=**0x0000**,**j**=**0**,**k**=**0**,**count**=**0**;** //Read Starting address

uint8\_t MH**,**ML**;**

//Flag is used to check whether the data from from the image is finished coming in

boolean EndFlag**=**0**;**

//Byte used to store 2 hex values that come in. 2 hex values = 1 byte.

byte incomingbyte**;**

void SendResetCmd**();** //Reset camera module.

void SendTakePhotoCmd**();** //Command used to take a photo.

void SendReadDataCmd**();** //Command used to request JPEG data be sent.

void StopTakePhotoCmd**();** //Command to stop taking a picture.

void setup**()**

**{**

//This initialises a serial port that will connect the arduino to a

//computer via USB at Baud rate of 19200.

Serial**.**begin**(**19200**);**

//This initialises a serial port that will connect the arduino to the //camera via the rx and tx pins at Baud rate of 38400.

mySerial**.**begin**(**38400**);**

**}**

void loop**()**

**{**

SendResetCmd**();**

//After reset, wait 2-3 second to send take picture command

delay**(**4000**);**

SendTakePhotoCmd**();**

//Check if data is returned from calling SendTakePhotoCmd().

**while(**mySerial**.**available**()>**0**)**

**{**

//Read data coming from mySerial port and save into incomingbyte.

incomingbyte**=**mySerial**.**read**();**

**}**

byte a**[**32**];**

**while(!**EndFlag**)**

**{**

j**=**0**;**

k**=**0**;**

count**=**0**;**

SendReadDataCmd**();** //Request JPEG data be sent

delay**(**400**);**

**while(**mySerial**.**available**()>**0**)**

**{**

incomingbyte**=**mySerial**.**read**();**

k**++;**

**if((**k**>**5**)&&(**j**<**32**)&&(!**EndFlag**))**

**{**

a**[**j**]=**incomingbyte**;** //Check if the picture is over

**if((**a**[**j**-**1**]==**0xFF**)&&(**a**[**j**]==**0xD9**))**

**{**

EndFlag**=**1**;**

**}**

j**++;**

count**++;**

**}**

**}**

**for(**j**=**0**;**j**<**count**;**j**++)**

**{** **if(**a**[**j**]<**0x10**)**

Serial**.**print**(**"0"**);**

//Send jpeg picture over the serial port.

Serial**.**print**(**a**[**j**],**HEX**);**

**}**

**}**

**);**

**while(**1**);** //Created to make camera module only take one picture and

//stop

//Send Reset command

void SendResetCmd**()** **{**

mySerial**.**write**((**byte**)**0x56**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x26**);**

mySerial**.**write**((**byte**)**0x00**);**

**}**

//Send take picture command

void SendTakePhotoCmd**()** **{**

mySerial**.**write**((**byte**)**0x56**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x36**);**

mySerial**.**write**((**byte**)**0x01**);**

mySerial**.**write**((**byte**)**0x00**);**

a **=** 0x0000**;** //reset so that another picture can be taken

**}**

void FrameSize**()** **{**

mySerial**.**write**((**byte**)**0x56**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x34**);**

mySerial**.**write**((**byte**)**0x01**);**

mySerial**.**write**((**byte**)**0x00**);**

**}**

//Read data

void SendReadDataCmd**()** **{**

MH**=**a**/**0x100**;**

ML**=**a**%**0x100**;**

mySerial**.**write**((**byte**)**0x56**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x32**);**

mySerial**.**write**((**byte**)**0x0c**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x0a**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**MH**);**

mySerial**.**write**((**byte**)**ML**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x20**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x0a**);**

a**+=**0x20**;**

**}**

void StopTakePhotoCmd**()** **{**

mySerial**.**write**((**byte**)**0x56**);**

mySerial**.**write**((**byte**)**0x00**);**

mySerial**.**write**((**byte**)**0x36**);**

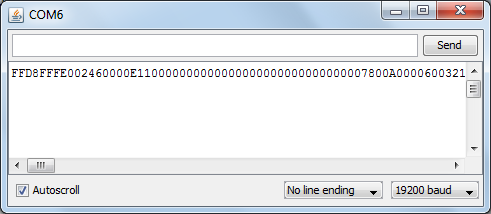
mySerial**.**write**((**byte**)**0x01**);**

mySerial**.**write**((**byte**)**0x03**);**

**}**

**Serial Monitor**

Run code shown above. Open the Serial monitor and the output will be a line of ASCII hexadecimal values. The output is should look like what is shown below.



**Turn Hexadecimal values into a Picture**

Copy (CTRL+A) and paste all the values from the serial monitor into a txt file called image. Use the python code shown below to convert the String Hexadecimal values to ASCII binary values and save into a jpg file. Ensure the python script is saved in the same directory as the txt file called image. Run the python script and an image called image will be produced in the same directory.

**import** binascii

f **=** open **(**"image.txt"**,**"r"**)** #Open file image.txt

nf **=** open**(**"image.jpg"**,**"wb"**)** #Create a file called image and open it.

**while** 1**:**

c **=** f**.**readline**()** #Read line from text file.

d **=** c**.**strip**()** #Remove all spacing and save in d.

**if** **not** c**:**

**break**

#Convert hex string d into ASCII binary and save in image.jpg

nf**.**write**(**binascii**.**a2b\_hex**(**d**))**

# Close the file

f**.**close**()** #Close image.txt

nf**.**close**()** #Close image.jpg

**Result**

Night-time caption

Daytime caption





**Problems encountered**

The manufacturers of the camera provide Arduino code that can operate the camera. The problem was that it was out of date and so the Arduino compiler gave compilation errors. It was found that the SoftwareSerial library had been updated recently. The code given by the manufacturer has the error of defining the virtual interface shown below as:

SoftwareSerial mySerial**(**13**,** 4**);**

This was corrected with:

SoftwareSerial mySerial **=** SoftwareSerial**(**10**,** 3**);**

**References**

<https://www.sparkfun.com/datasheets/Sensors/Imaging/1274419957.pdf>

<http://robotic-controls.com/learn/arduino/linksprite-jpeg-camera>