**CAMERA MODULE INTERFACING WITH ARDUINO UNO**

**INTRODUCTION:**

A camera module is like a miniaturized camera system, often found in smartphones, laptops, and security cameras. It's essentially a self-contained unit with all the parts needed to capture images and videos. Here's a breakdown of the key components and how they work together:

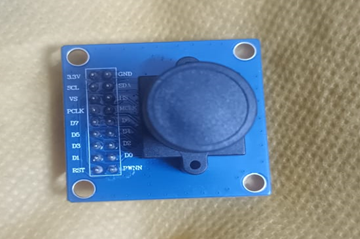
**The Essential Parts:**

* **Lens:** The lens focuses light rays coming from the scene you're capturing. Just like a magnifying glass, it concentrates the light onto a specific area.
* **Image Sensor:** This is the heart of the camera module. It's a light-sensitive chip, like film in traditional cameras. The sensor converts the focused light rays from the lens into electrical signals. There are two main types of image sensors: CMOS (Complementary Metal-Oxide-Semiconductor) and CCD (Charge-Coupled Device). CMOS sensors are more common in consumer electronics due to their lower power consumption.
* **Supporting Circuitry:** This includes various electronic components mounted on a small circuit board. It controls the image sensor, processes the electrical signals, and prepares them for output as a digital image or video.

**The Capture Process:**

* Light from the scene enters the camera module through the lens.
* The lens focuses the light rays onto the image sensor.
* The image sensor converts the light rays into electrical signals.
* The camera module's circuitry processes these electrical signals, adjusting things like color balance and exposure.
* Finally, the processed signals are converted into a digital format (like a JPEG image) and sent to the device it's connected to (like your smartphone).
* Camera modules come in various configurations, with different lens types, sensor resolutions, and functionalities. They are a crucial technology behind the ever-increasing image quality in our everyday devices.

**Camera OV7670**

****

**WORKING PRINCIPLE:**

The working principle of a camera module revolves around capturing light and converting it into a digital image. Here's a closer look at the steps involved:

* **Light Gathering:** Light from the scene you're capturing enters the camera module through the lens.
* **Focusing the Light:** The lens plays a critical role. By adjusting its elements, it focuses the incoming light rays onto a specific area of the image sensor. This ensures a clear and sharp image.
* **Light to Electricity Conversion:** The image sensor, the heart of the module, is a light-sensitive chip. When the focused light hits the sensor, it creates electrical signals. Each tiny light-sensitive spot on the sensor, called a pixel, generates a corresponding electrical signal based on the intensity of the light it receives.
* **Signal Processing:** The raw electrical signals from the sensor are analog and need further processing. The camera module's circuitry performs several tasks here:
  + **Analog-to-Digital Conversion (ADC):** Converts the analog electrical signals from the sensor into a digital format that can be understood by electronic devices.
  + **Signal Processing:** The digital signals might require adjustments for factors like color balance, exposure, and noise reduction. Dedicated processors or integrated circuits within the module handle these tasks.
* **Digital Output:** Finally, the processed digital data, which now represents the captured image, is sent to the device the camera module is connected to. This could be a smartphone, computer, or security system. The format of the data (JPEG, RAW) depends on the camera module's specifications.
* **Additional Points:**
* **Colour Capturing:** Most image sensors capture information only in grayscale. To achieve color images, Bayer filters are often used over the sensor. These filters allow specific colors (red, green, blue) to pass through different areas of the sensor, and the image processing stage reconstructs the final color image.
* **Image Resolution:** The number of pixels on the image sensor determines the image resolution. More pixels generally translate to higher resolution and sharper images.
* **Control Circuitry:** The camera module might also have additional circuitry for functionalities like autofocus, auto exposure, and white balance control. These features are managed electronically based on feedback from sensors and user settings.
* By working together, these components within a camera module capture light, convert it into electrical signals, process them, and deliver a digital representation of the scene as an image.

**Specifications of this sensor:**

* Resolution: VGA (640×480) or CIF (352×288)
* Frame rate: 30 fps (VGA) or 60 fps (CIF)
* Image sensor: 1/6-inch VGA CMOS image sensor
* Signal-to-Noise Ratio (SNR): 46 dB
* Sensitivity: 1.3 V/(Lux-sec)
* Dynamic range: 52 dB
* Operating voltage: 2.5V to 3.0V
* Power consumption: 60 mW
* Interface: SCCB (Serial Camera Control Bus) or I2C

### **Connections for Arduino Uno/Nano:**

OV7670 connections:

VSYNC - PIN2  
XCLCK - PIN3 (must be level shifted from 5V -> 3.3V)  
PCLCK - PIN12  
SIOD - A4 (I2C data) - 10K resistor to 3.3V  
SIOC - A5 (I2C clock) - 10K resistor to 3.3V  
D0..D3 - A0..A3 (pixel data bits 0..3)  
D4..D7 - PIN4..PIN7 (pixel data bits 4..7)  
3.3V - 3.3V  
RESET - 3.3V  
GND - GND  
PWDN - GND

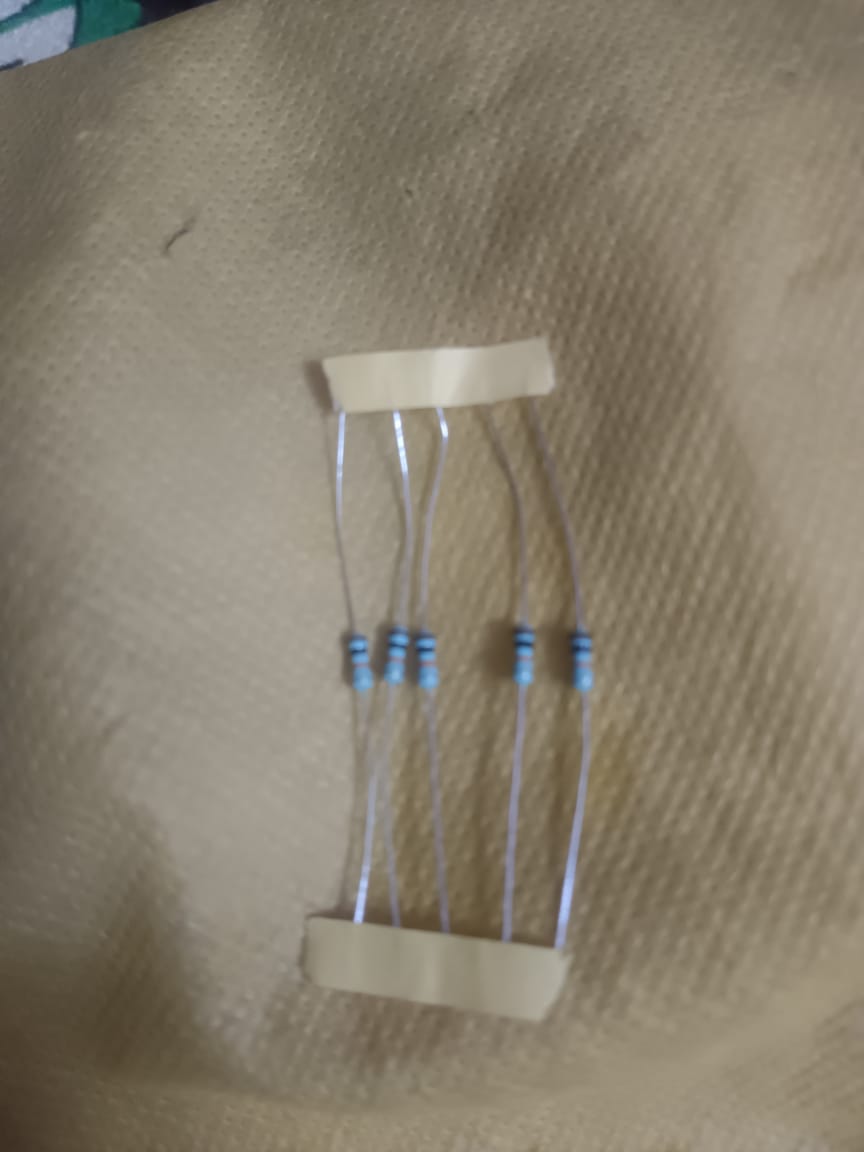
1.8" TFT connections:

DC - PIN8 (5V -> 3.3V)  
CS - PIN9 (5V -> 3.3V)  
RESET - PIN10 (5V -> 3.3V)  
SPI data - PIN11 (5V -> 3.3V)  
SPI clock - PIN13 (5V -> 3.3V)  
VCC - 5V/3.3V (depending on jumper position on the TFT board)  
BL - 3.3V  
GND - GND

## **Required Components Are Given Below:**

## 

## 



## **Installing "ArduImageCapture" Plug-In:**

## Download:

[ArduImageCapture.1.1.zip](https://circuitjournal.com/download?file=ArduImageCapture.1.1.zip)

GitHub page for the source code:

<https://github.com/indrekluuk/ArduImageCapture>.

Extract the zip file and copy the "ArduImageCapture" folder into your Arduino "tools" folder next to the Arduino "libraries" folder. If the "tools" folder doesn't exist, then you can create it yourself.

Now run/restart your Arduino IDE, and you should see "ArduImageCapture" under the "Tools" menu.

## **Arduino Code:**

* **Download the code of my LiveOV7670 project:**  
  <https://github.com/indrekluuk/LiveOV7670>
* Click on the green "Code" button and then "Download ZIP."
* **Copy the two libraries "LiveOV7670Library" and "Adafruit\_GFX\_Library" from "src/lib" to your Arduino "libraries" folder.**
* On Windows:
* C:\Users\<username>\Documents\Arduino\libraries
* On Mac:
* /Users/<username>/Documents/Arduino/libraries
* "LiveOV7670Library" contains the code that communicates with the OV7670 camera module.
* "Adafruit\_GFX\_Library" is a dependency of the "Adafruit-ST7735" screen library. We aren't using any screens in this tutorial, but the dependency is still required for the LiveOV7670 sketch to compile.
* **Open "src/LiveOV7670/LiveOV7670.ino" with Arduino IDE.**
* This project takes advantage of some of the C++11 features. C++11 is enabled by default since Arduino IDE version 1.6.6. If you have an older version for some reason, you should either upgrade or enable C++11 in the Arduino IDE configuration file.
* **Switch to "EXAMPLE 3" in "setup.h".**
* Select "setup.h" tab and change the definition to:

#define EXAMPLE 3

"EXAMPLE 3" will activate the "ExampleUart.cpp". This is the code that sends images over the USB cable.

The default "EXAMPLE 1" tries to send the image to a tiny TFT screen.

**5. Upload the code to your Arduino.**

We can make the first test before connecting any wires.

***Checkpoint 1***

Open "ArduImageCapture." It should start listening to the serial port automatically. If not, select the correct COM port and click "Listen."

You should see a red image. This means that you have the correct code in your Arduino, but it couldn't detect the camera moduThe default baud rate for the LiveOV7670 project is 1Mbit/sec (1000000) at 320x240 resolution. It works well if you have an Arduino clone with the CH340 serial chip and a Windows PC.

If you have a genuine Arduino with an FTDI serial chip or use a Mac, then you may see a scrambled line ends:

For some reason, a genuine Arduino with an FTDI serial chip will not work with a 1MBit/sec baud rate. So, you need to reduce the speed to 115200 bit/sec.

Similar problem with Mac. For some reason, 1Mbit/sec doesn't work on Mac.

In those cases, select the "ExampleUart.cpp" tab and change the definition to:

#define UART\_MODE 4

This lowers the resolution to 160x120 with a 115200 baud rate. (Unfortunately, 115200 is too slow to transfer higher resolution)

After uploading the code again, choose "115200" from the serial speed selection.

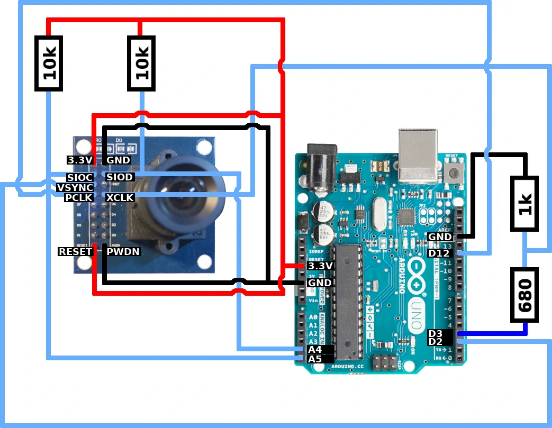
Now you should see this image:

**The wire structure of this sensor:**

## **First Part of the Camera Connections – Power It Up**

We will do the camera wiring in two phases. In this chapter, we will make all the necessary connections to get the camera running and configured by the Arduino.

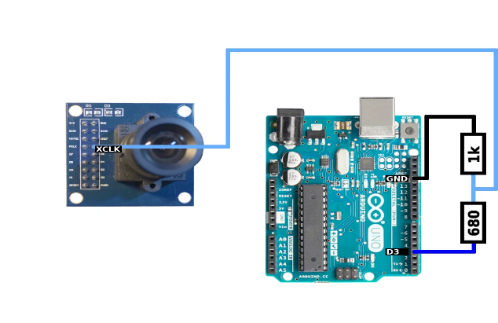
**Schematic:**

****

**Step 1:**

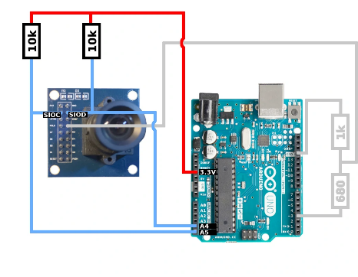
**Make a voltage divider from the Arduino pin 3 to the XCLK pin of the camera.**

Create a voltage divider to get the 5V signal from Arduino to 3.3V for the camera. It is necessary because The OV7670 camera module is not 5V tolerant.XCLK is the input clock that makes the camera run. The maximum frequency that Arduino can put out is 8Mhz. For full speed, the camera module needs 30Mhz, but eight is enough to send a small image to the computer.

****

**Step 2:**

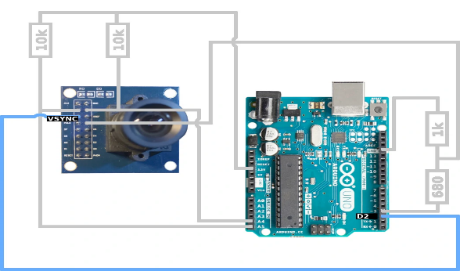
Make the I2C connections. Arduino pin A5 to SIOC and Arduino pin A4 to SIOD. Then add a 10k pull-up resistors to 3.3V to both wires (A5 to 10k to 3.3V, A4 to 10k to 3.3V).I2C is necessary for sending configuration data to the camera (resolution, pixel format, etc.).

****

**Step 3:**

**VSYNC to Arduino pin 2.**

It's a 3.3V signal from the camera to the Arduino. This connection can be made directly without a voltage divider. We need vertical sync to know when a new frame begins. Otherwise, it looks to Arduino like a constant pixel stream with no start or end.

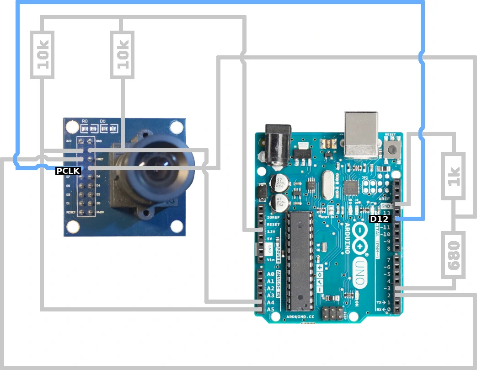
****

**Step 4:**

**PCLK to Arduino pin 12.**

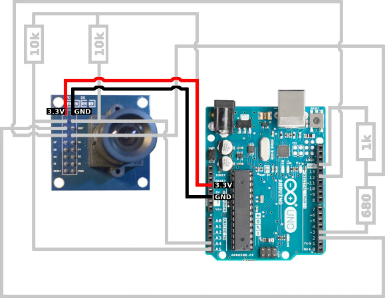
This is also a 3.3V signal from the camera to the Arduino and can be connected directly.

Pixel cock is necessary for knowing the exact time when to read pixel data.

****

**Step 5:**

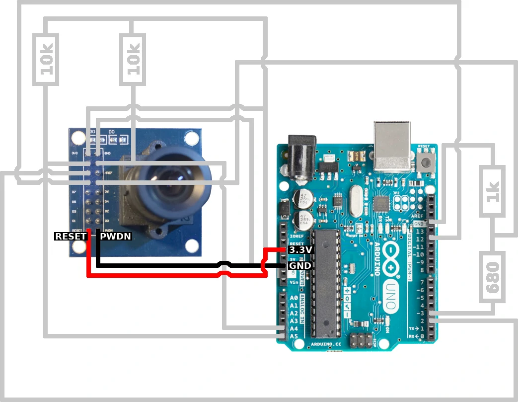
Connect power to the camera. From Arduino 3.3V pin to the camera's 3.3V input and from Arduino GND pin to the camera's GND.

****

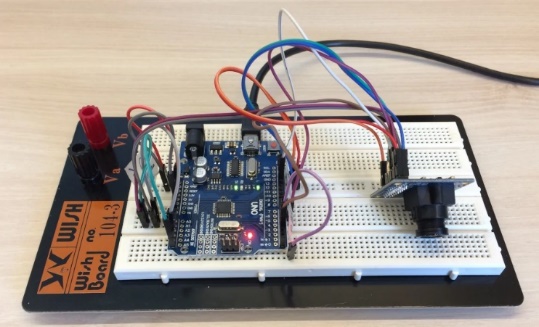
**Step 6:**

**Connect the camera's RESET pin to 3.3V and PWDN to GND.**

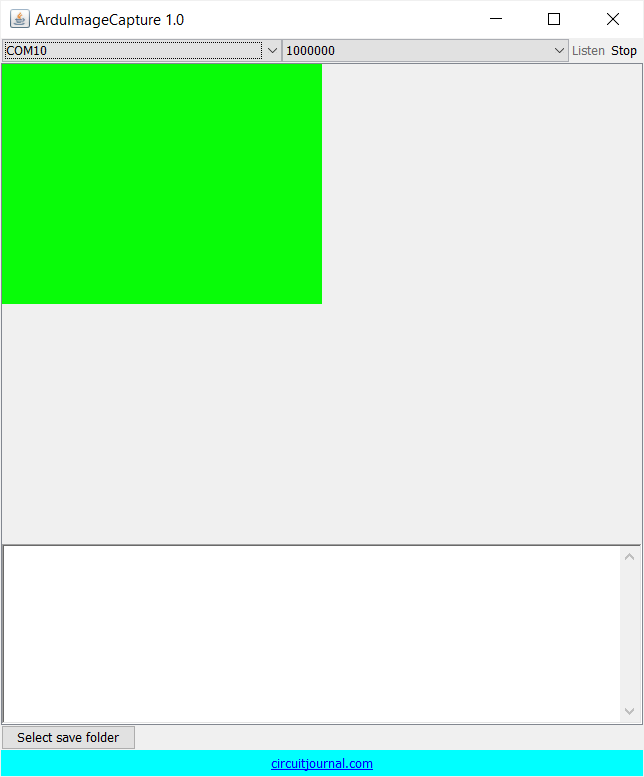
The Reset pin could be used to reset the camera module and power down to turn it off. But since we don't have any left-over pins, we let it run all the time.

****

#### Checkpoint 2

Let's validate that the Camera is Running. This is the second test before we get to the actual images from the camera.

Your wiring should look something like this:

When you start your Arduino again and open "ArduImageCapture," it should flash a green image. This means that the LiveOV7670 library was able to detect and configure the camera successfully.

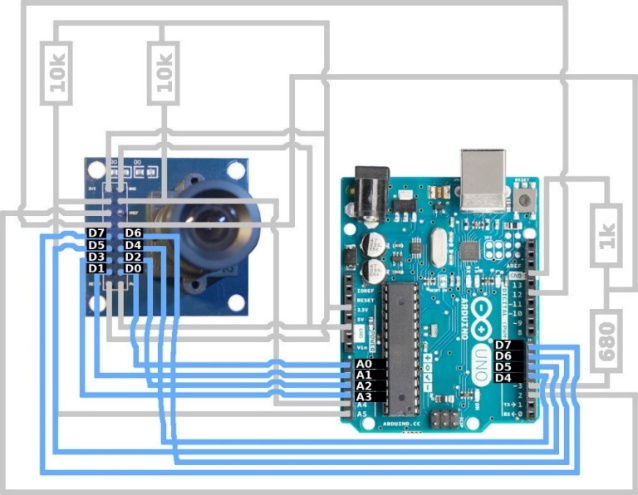
You can't see any images yet since the pixel data pins are not connected.

If you still see the red image, then check the wiring. Make sure that the XCLK wire isn't too long. The square wave of the input clock signal to the camera may become too deformed for it to operate correctly.

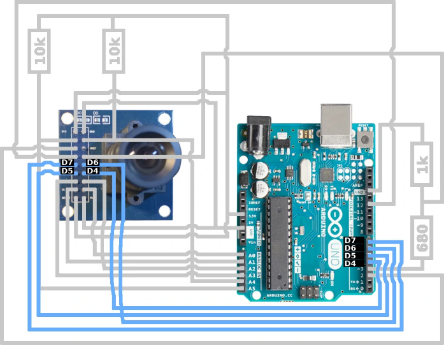
## **Second Part of the Camera Connections – Pixel Data Pins:**

Now we can finish the camera wiring by connecting pixel data inputs. Pixels are streamed from the camera one byte at a time. Each pixel consists of two bytes that are read sequentially.

**Schematic:**



**Step 1:**

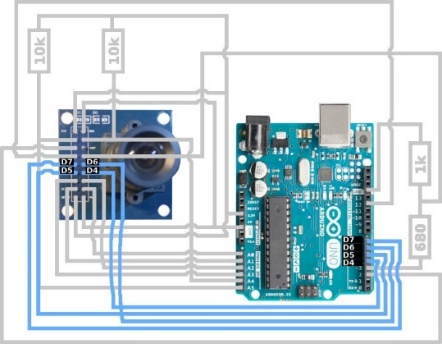
**Connect Camera's D0 to D3 to the Arduino pins A0 to A3.**

These are the lower four bits of a pixel byte.

**Step 2:**

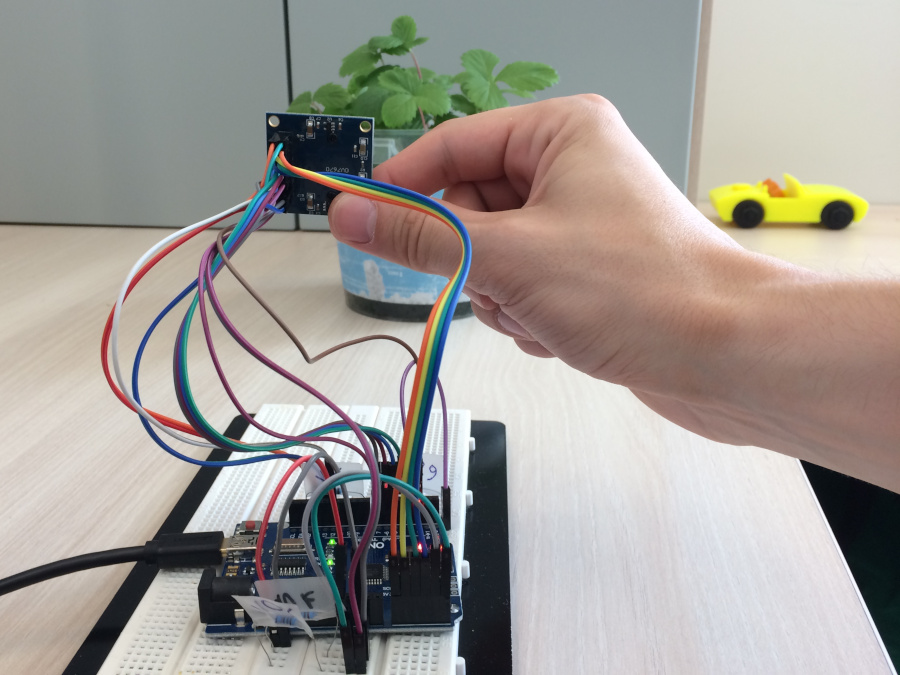
**Connect Camera's D4 to D7 to the Arduino pins 4 to 7.**

These are the higher four bits of a pixel byte



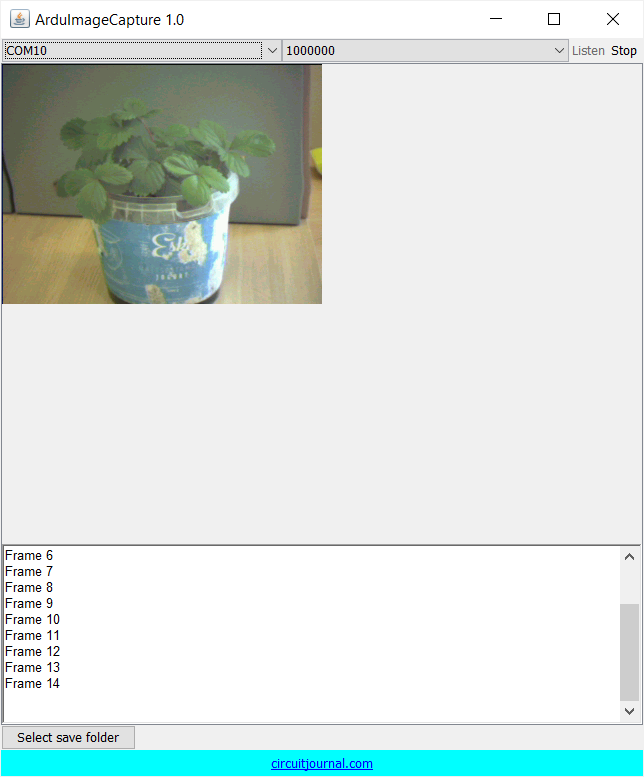
## Done!

Now you can power up your Arduino and open "ArduImageCapture" again.



It starts with a green image like in the previous checkpoint, and then you should see a live video streaming from the camera.

**OUTPUT:**



**APPLICATIONS OF CAMERA MODULES :**

Camera modules are incredibly versatile pieces of technology. Their primary function is to capture images and videos, but they have a wide range of applications beyond that. Here are some of the most common uses of camera modules:

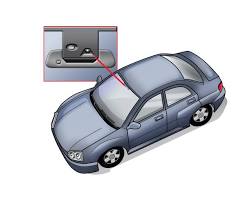
* **Consumer electronics:** Camera modules are essential components in many consumer electronics devices, including smartphones, tablets, laptops, and digital cameras. They allow us to take pictures and videos of our loved ones, document our experiences, and video chat with friends and family.



* **Security and surveillance:** Camera modules are widely used in security and surveillance systems. They can be used to monitor homes, businesses, and public areas for security purposes. They can also be used to track traffic flow and identify criminals.

[](https://www.lowes.com/pd/GE-Interior-Exterior-Simulated-Security-Camera/1000550959)

* **Automotive industry:** Camera modules are becoming increasingly important in the automotive industry. They are used in features such as rearview cameras, lane departure warnings, and blind spot detection. In the future, they may be used for self-driving cars.



* **Industrial automation:** Camera modules are used in industrial automation for a variety of purposes, such as quality control, machine vision, and robotic guidance. They can be used to inspect products for defects, track objects on a conveyor belt, and help robots navigate their environment.

[ w](https://roboticsandautomationnews.com/2016/06/21/robotiq-launches-camera-system-for-industrial-robot-hand/5852/)

* **Medical imaging:** Camera modules are used in a variety of medical imaging devices, such as endoscopes, microscopes, and ultrasound machines. They allow doctors to see inside the body for diagnostic purposes.



* **Education and entertainment:** Camera modules are used in a variety of educational and entertainment applications. For example, they can be used to create augmented reality experiences or to capture educational videos. They are also used in video game consoles and other entertainment devices.

**REFERENCES:**

[**https://youtu.be/vEvQv5-RL-s?si=I8zhzvCT0yLG9q22**](https://youtu.be/vEvQv5-RL-s?si=I8zhzvCT0yLG9q22)**.**

[**https://youtu.be/R94WZH8XAvM?si=3WF\_jvZ5dTBX6g5Z**](https://youtu.be/R94WZH8XAvM?si=3WF_jvZ5dTBX6g5Z)**.**

[**https://youtu.be/6bfY9JXOppI?si=i3KbXDJQEn9iD6Gh**](https://youtu.be/6bfY9JXOppI?si=i3KbXDJQEn9iD6Gh)**.**