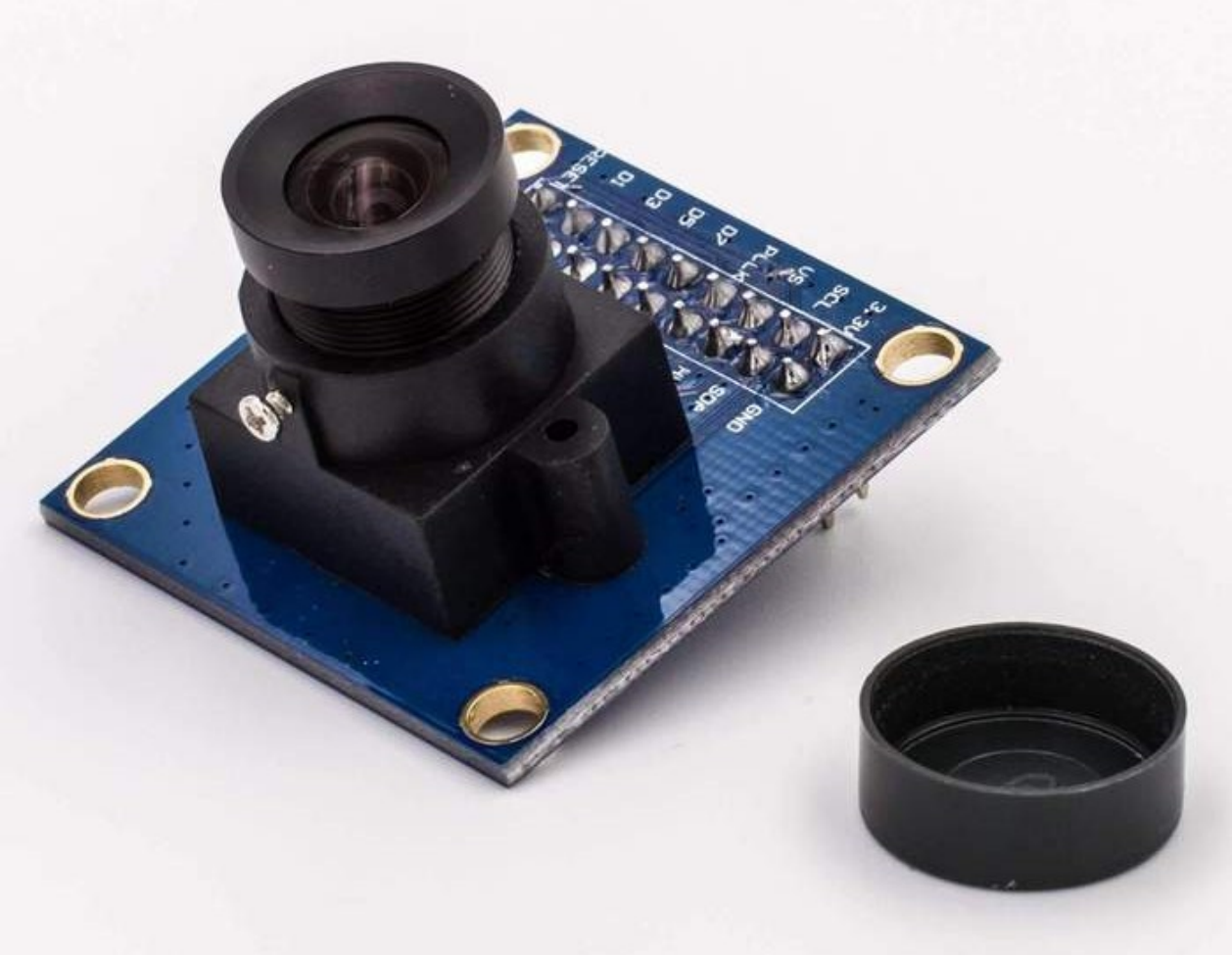
Welcome!

Thank you very much for purchasing our AZ-Delivery OV7670 camera for Arduino. On the following pages, we will introduce you to how to use and setup this handy device.

Have fun!



The camera module OV7670 is available in two versions. One with *First In First Out* memory (*FIFO*) and one without *FIFO*. This description applies to the one we sell, and that is module **WITHOUT FIFO**.

The details, code and specifications explained in this description do not work for the module with FIFO.

You recognize the presence or absence of a FIFO by the following characteristics:

Module without FIFO

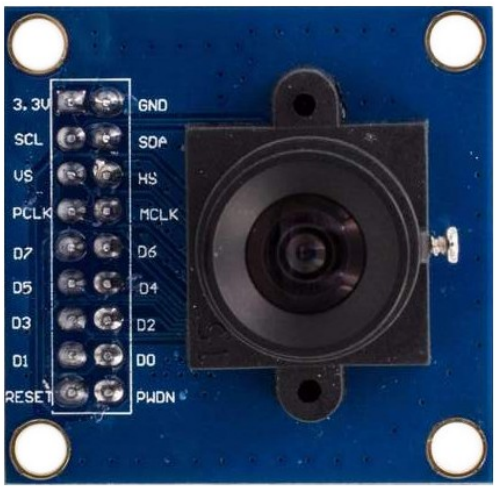
- 18 Pin Connector
- Back side without SMD chip

Module with FIFO

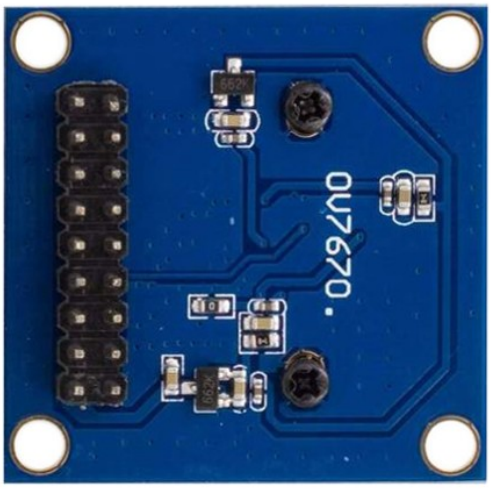
- 20 Pin Connector
- Backside with SMD chip (AVRERL)

Our module (without FIFO) should look like this from the front and back side:

FRONT



BACK (without FiFo)



The camera module has a supply and logic (data) voltage of 3.3V.

Voltages higher than 3.3V can destroy the camera!!!

The camera module comes with a 9x2 male header, the pin functions are shown below (look at the image above for pin assigment):

(Some pins have alternative names, depending on documentation, in parentheses)

PIN	DIRECTION	FUNCTION
3.3V		power supply: **+3.3V**
GND		ground: **0V**
SCL (SIOC)	input	two wire serial interface clock
SDA (SIOD)	input / output	two wire serial interface data
VS (VSYNC)	output	vertical synchronization output
HS (HREF)	output	horizontal synchronisation output
PCLK	output	pixel clock output
MCLK (XCLK)	input	system clock input
D7	output	data bit 7 MSB
D6	output	data bit 6
D5	output	data bit 5
D4	output	data bit 4
D3	output	data bit 3
D2	output	data bit 2
D1	output	data bit 1
D0	output	data bit 0 LSB
RESET	input	reset (active **LOW**, GND)
PWDN	input	power down (active **HIGH**, 3.3V)

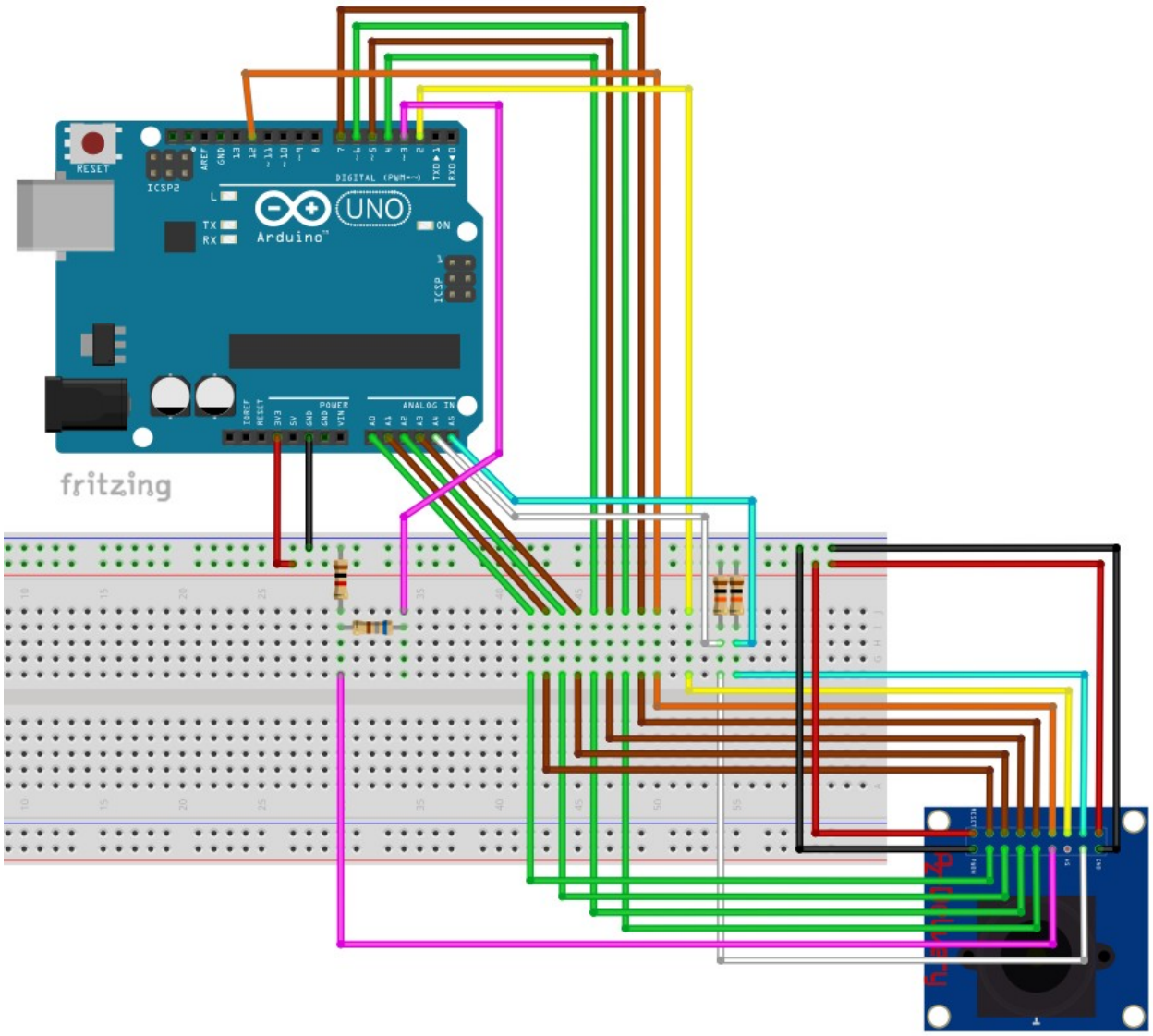
The camera supports the following resolutions:

- VGA (640x480) (standard mode)
- QVGA (320x240)
- CIF (352x240)
- QCIF (176x144)
- Freely definable resolution up to maximum VGA

The data transmission takes place byte by byte in parallel with one of the following possible data transmission formats:

- YCbCr422 (standard)
- RGB565
- RGB555
- RGB444

Connecting the camera with Uno



Camera pin	>	Uno pin	
3.3	>	3.3V	**Red wire**
GND	>	GND	**Black wire**
SIOC	>	A5 (SCL)	**Cyan wire**
SIOD	>	A4 (SDA)	**White wire**
VSYNC	>	Digital pin 2	**Yellow wire**
HREF	>	**Not connected to anything**	
PCLK	>	Digital pin 12	**Orange wire**
XCLK	(via voltage divider)	Digital pin 3	**Pink wire**
Data 7	>	Digital pin 7	**Brown wire**
Data 6	>	Digital pin 6	**Green wire**
Data 5	>	Digital pin 5	**Brown wire**
Data 4	>	Digital pin 4	**Green wire**
Data 3	>	A3	**Brown wire**
Data 2	>	A2	**Green wire**
Data 1	>	A1	**Brown wire**
Data 0	>	A0	**Green wire**
RESET	>	3.3V	**Red wire**
PWDN	>	GND	**Black wire**

Both SDA and SCL lines have **pull up resistor 10kΩ**.

DON'T CONNECT XCLK PIN DIRECTLY TO THE UNO DIGITAL PIN!!!

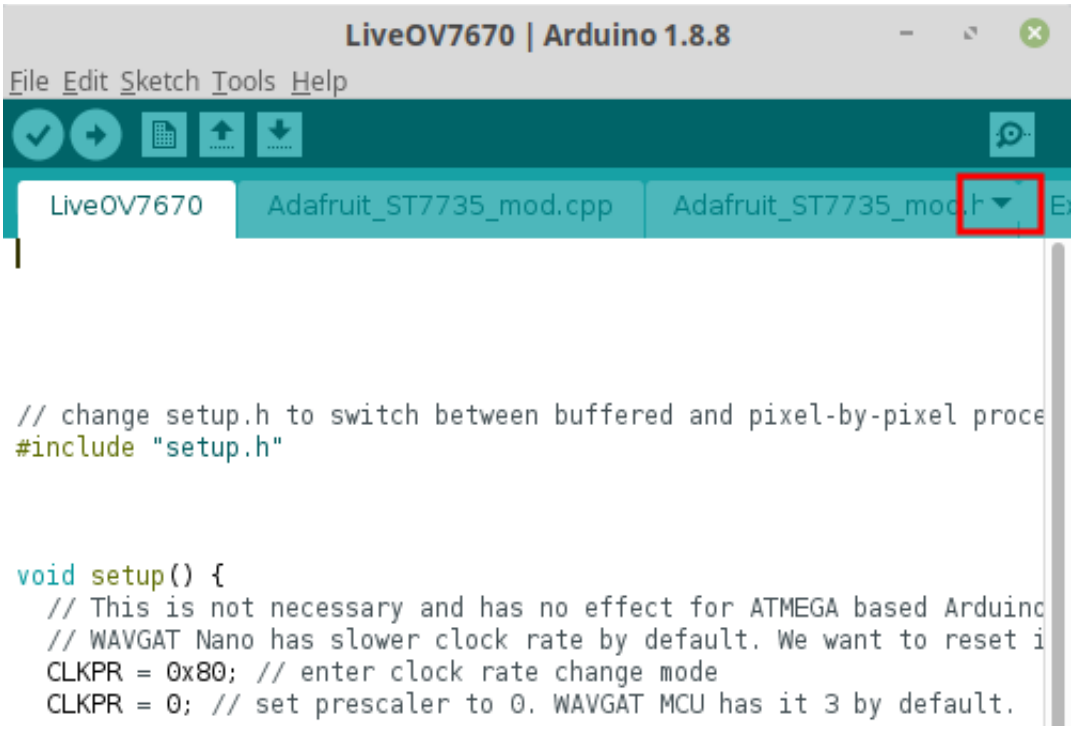
XCLK pin is input pin of camera and it works on 3.3V logic, so we use **voltage divider** to lower UNOs 5V logic to 3.3V. We use one **1kΩ resistor** (one end to the GND and second to XCLK), and one **680Ω resistor** (one end to D3 of Uno and second end to XCLK).

Sketch

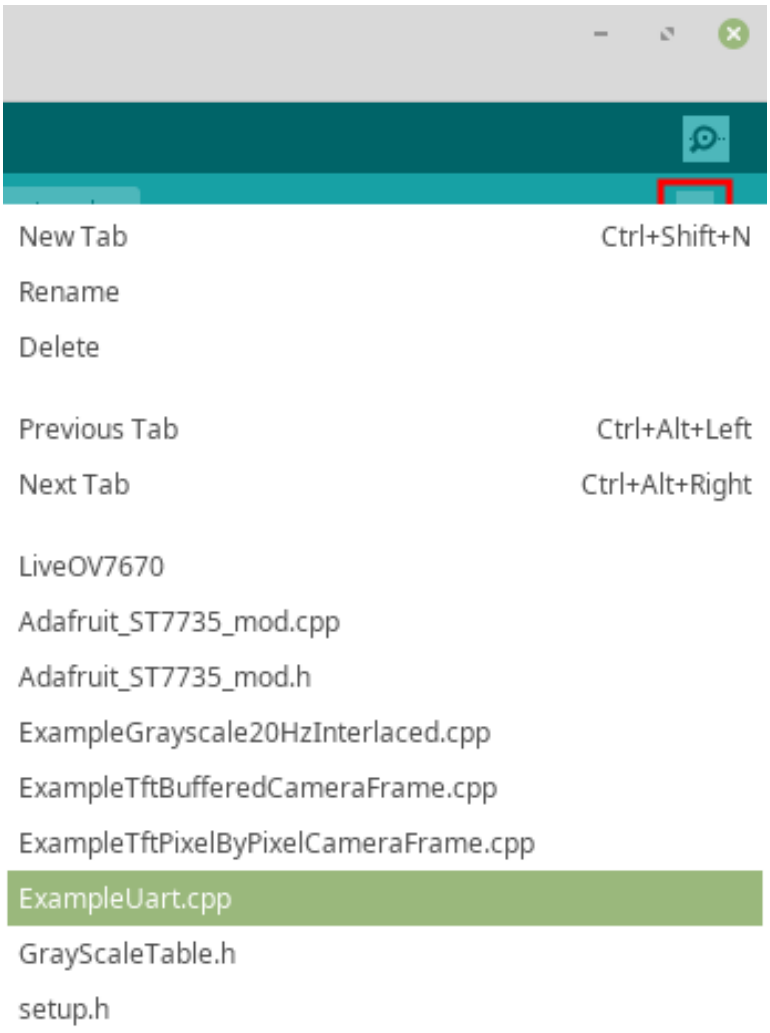
This is example sketch made by **indrekluuk**, and we will show you how to use it. Go to https://github.com/indrekluuk/LiveOV7670 and download it to your local machine. Extract it, and open this sketch:

src/LiveOV7670/LiveOV7670.ino

in your Arduino IDE. Go to *File* > *Open* and search to your extracted folder and open it. Several tabs (sketches) in one Arduino IDE window will be opened. Navigate to the *ExampleUard.cpp* by pressing CTRL + TAB or by clicking on switch button (in red rectangle on image below)



New drop-down menus will open, and select *ExampleUart.cpp*



From line 22 to line 28 there are several comments, explaining modes of operation for this sketch, followed by one #define line in which we specify the mode of operation. For purpose of this manual we will use mode 3, as seen below.

```
// select resolution and communication speed:
// 1 - 320x240 with 2M baud (may be unreliable!)
// 2 - 320x240 with 1M baud
// 3 - 160x120 with 1M baud
// 4 - 160x120 with 115200 baud
// 5 - 320x240 grayscale with 1M baud
// 6 - 160x120 grayscale with 1M baud
#define UART_MODE 3
```

Then navigate to *Setup.h.* In *Setup.h* we need to set example which we will use. From lines 11 to 30 there are comments that explain what each example does, followed by one #define line in which we specify the example. For purpose of this manual we will use Example 3, as seen below.

```
/*
 *  EXAMPLE = 1
 *  Use LiveOV7670Library class that reads line into buffer and
 *  sends data to TFT over SPI during blank lines.
 *
 *  EXAMPLE = 2
 *  Use LiveOV7670Library class that processes data pixel by pixel
 *  sends data to TFT during each pixel reading loop.
 *  VGA can not be used with line buffer since there is no
 *  time (no blank lines) to do something useful with a buffered line
 *
 *  EXAMPLE = 3
 *  Reads data from LiveOV7670Library and send it over UART to computer
 *  Java application for receiving picture from UART
 *  https://github.com/indrekluuk/ArduinoImageCapture
 *
 *  EXAMPLE = 4
 *  Gray scale image @20Hz. Interlaced image.
 *
 */
#define EXAMPLE 3
```

After these changes, connect your Arduino Uno to computer via usb cable, and upload this sketch to Arduino board.

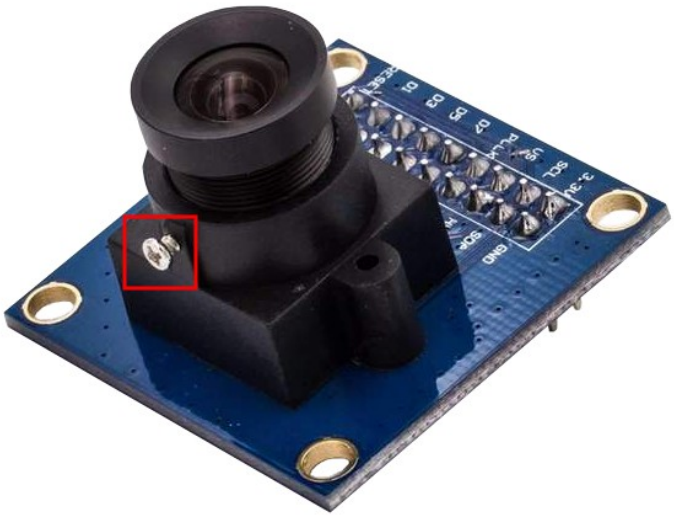
Reading images on computer

For this you will need ***IntelliJ Idea***. To download it go to:
https://www.jetbrains.com/idea/download

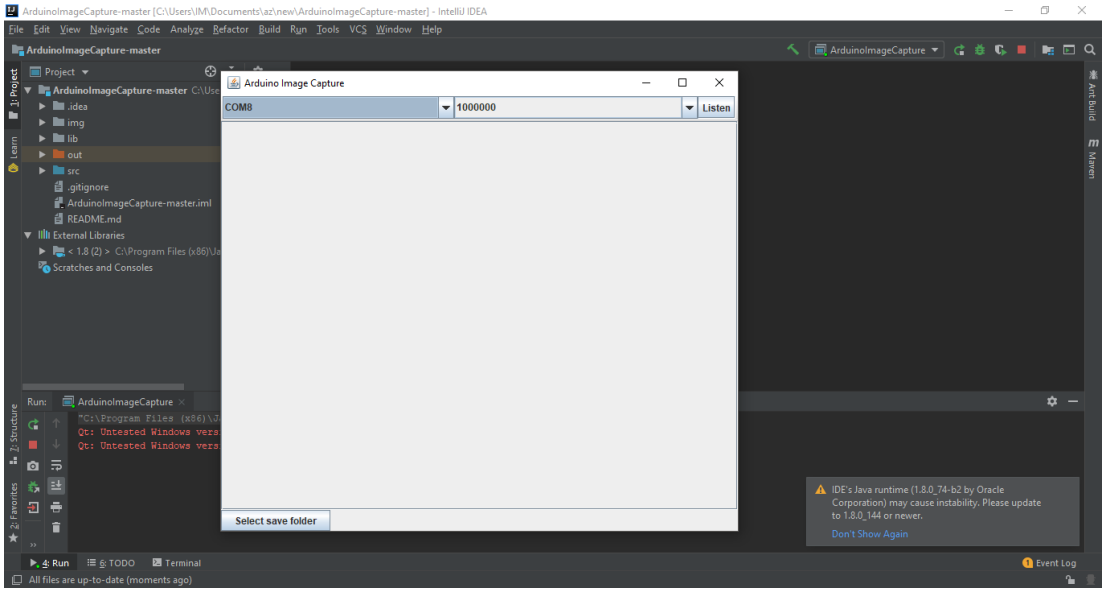
After installation, we need to import project, an app which will read data sent by Arduino Uno via usb cable (serial line). But first we need to download it. Also, this projects was created by **indrekluuk** so to download it, go to:
https://github.com/indrekluuk/ArduinoImageCapture

Follow instructions on the link above on how to import, build and run the app.

There is one screw on the camera (red rectangle on image below), which is used as lock, when camera focus is adjusted. When it is unscrewed we can change camera focus, and when we setup camera focus, then we scruw the screw and lock the camera.

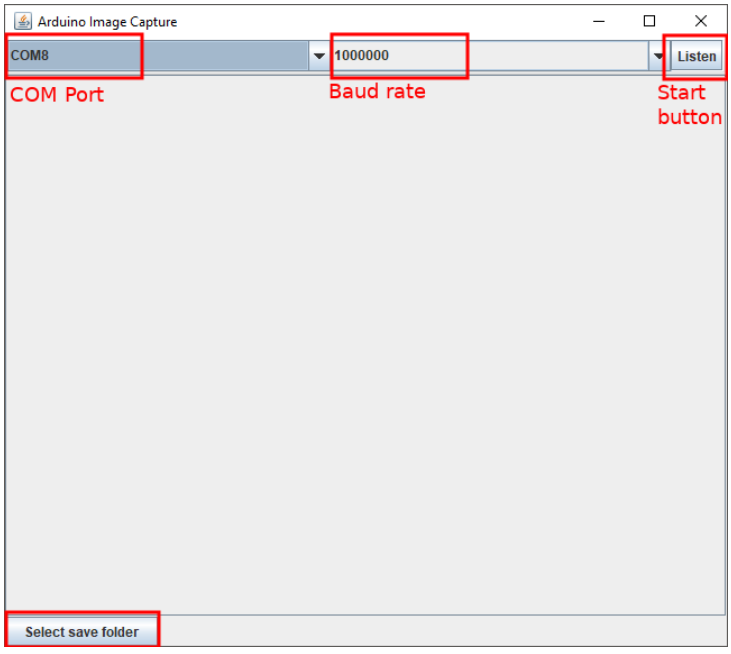


When you start an app in IntelliJ Idea, this is what you can seen. You have to connect your Arduino Uno board to PC before starting this app.

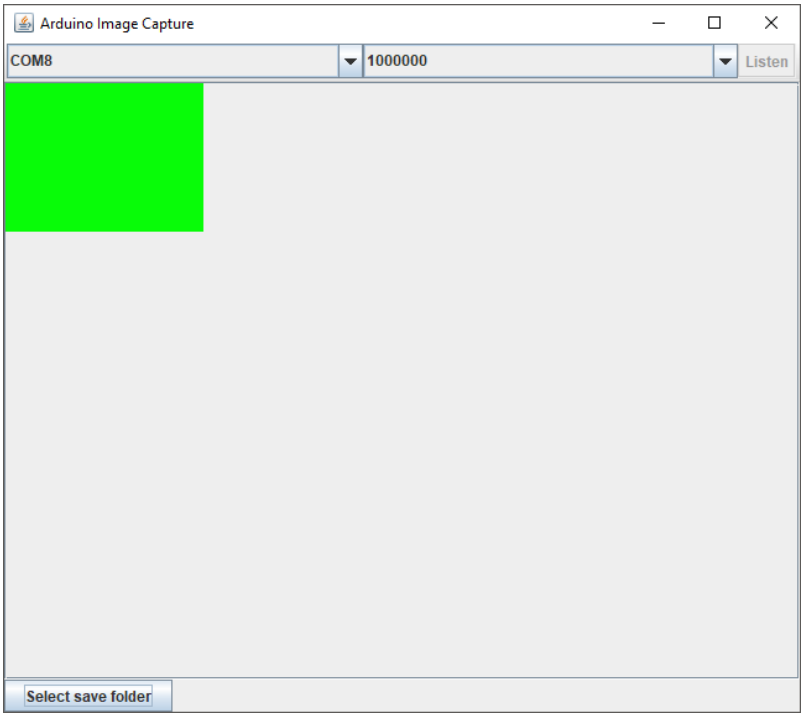


As you can see, in the Arduino Image Capture window, top left, an app detected our Arduino Uno on com port: COM8.

1000000 number next to COM8 is 1MB baud rate that we are using. You can change with an arrow, on the right for other modes of operation.

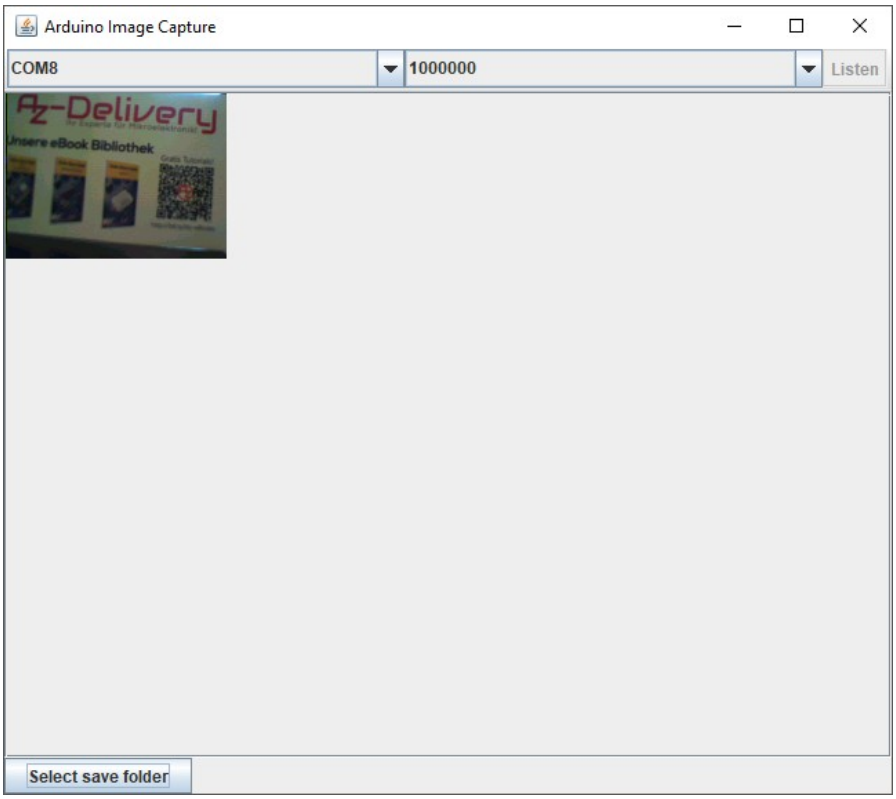


To start taking images click on ***Listen*** button (Start button). If everything worked fine, green rectangle should appear.

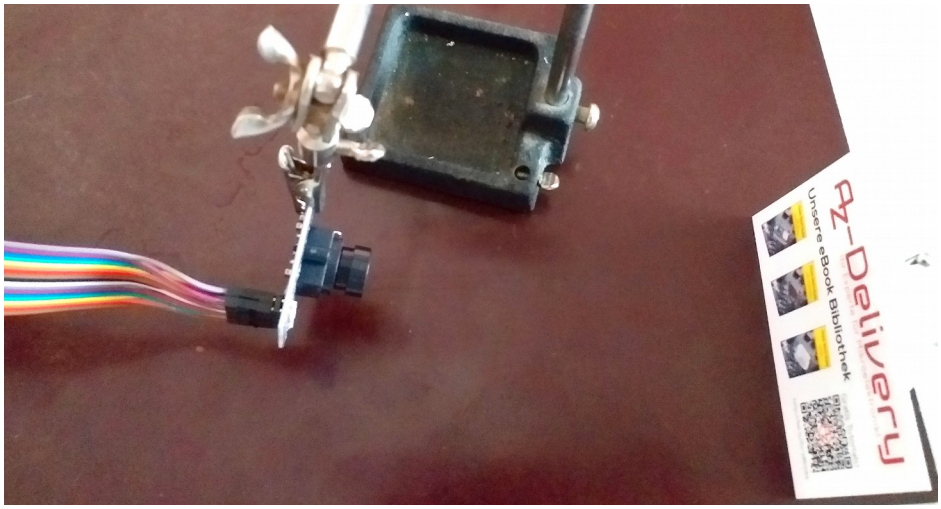


But if red rectangle apeared, that an app and Uno **detected** camera, but sending/receiving data was **not correct**. Check again connections between Uno and camera.

If everything worked fine, after green rectangle an image from camera should appear.

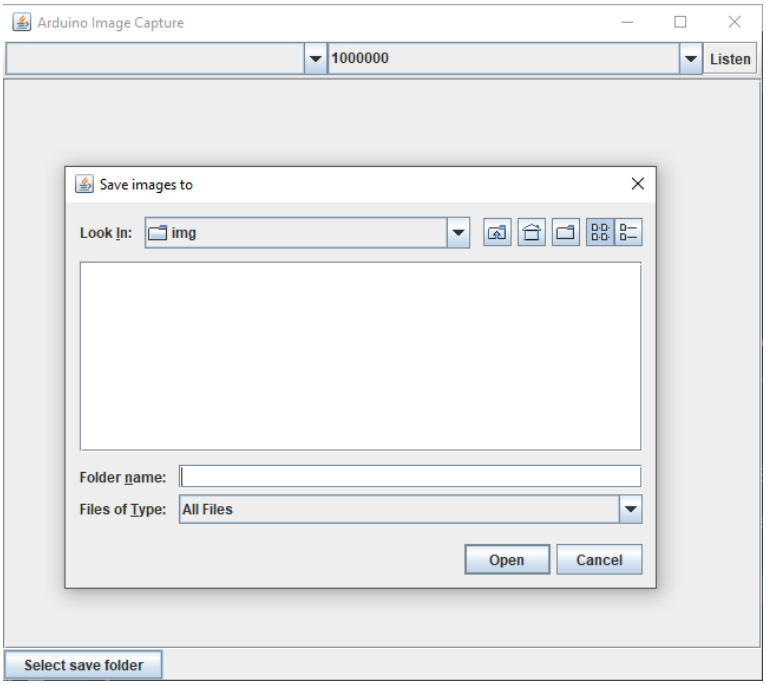


On the camera image there is our visit card, you can see camera setup on the image below.



And the image that camera captured is:

There is one more button in Arduino Image Capture window, on bottom left, ***Select save folder*** button. It is used to select folder where received images will be saved.



You've done it, you can now use and program your module for your projects.

Now it is time to learn and make the Projects on your own. You can do that with the help of many example scripts and other tutorials, which you can find on the internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us