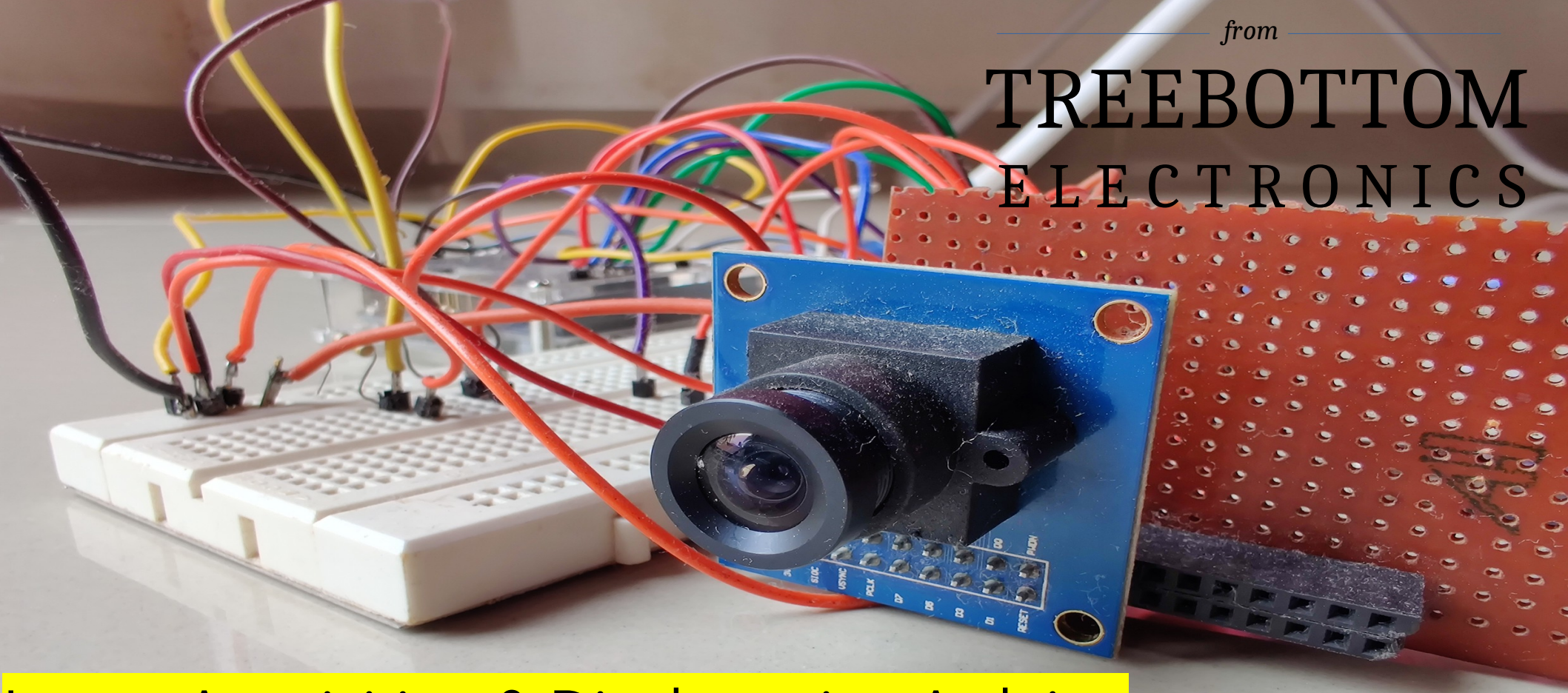


from

TREEBOTTOM

ELECTRONICS

Image Acquisition & Display using Arduino

A Simple Camera Design

Objectives

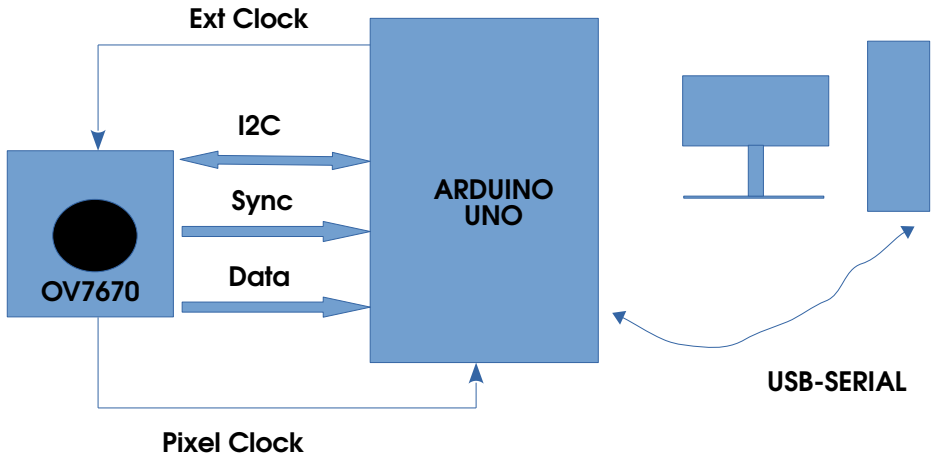
- To capture monochrome image in QQVGA, QVGA, VGA resolutions in YUV422 format from CMOS image sensor OV7670 using Arduino Uno
- To capture colour image and colour bar in QQVGA resolution in YUV422 and RGB565 format from CMOS image sensor OV7670 using Arduino Uno
- To transmit the captured image to PC through Serial to USB interface
- Record the image in PC as bitmap/JPEG
- Implement a MJPEG streamer in PC

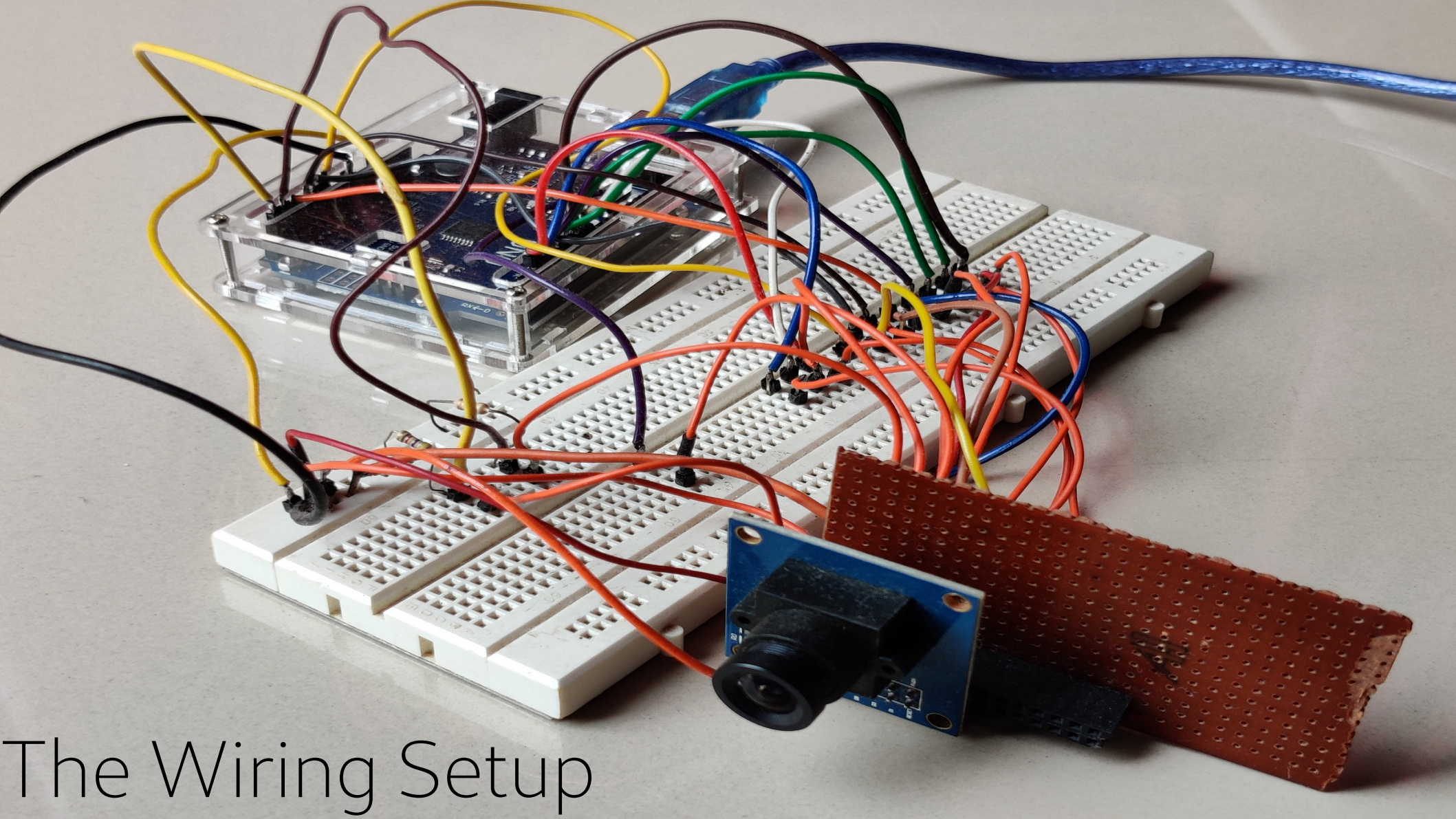
The Setup

Components used:

- Arduino UNO
- OV7670 module without FIFO

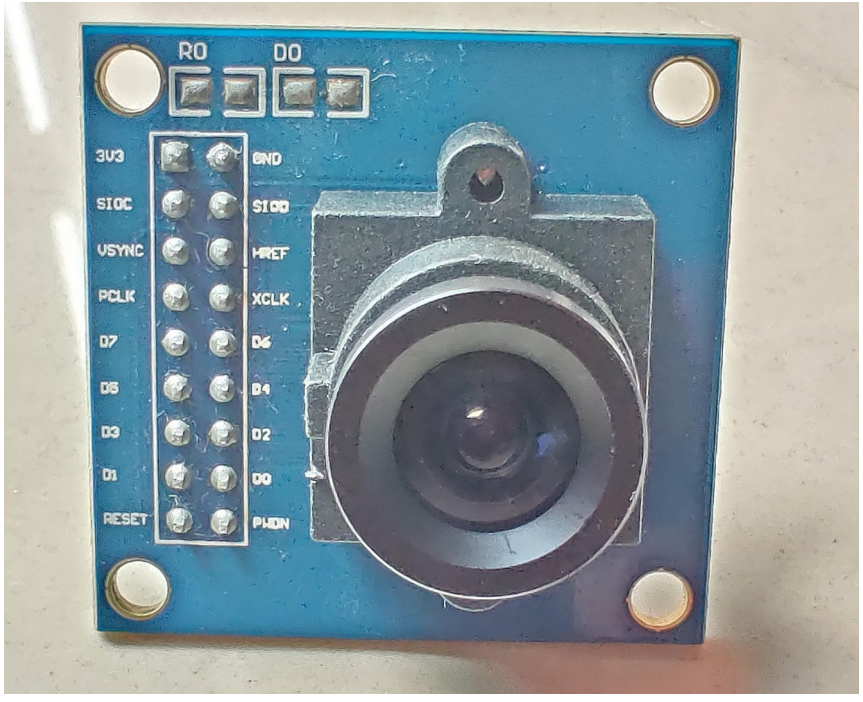
Block Diagram

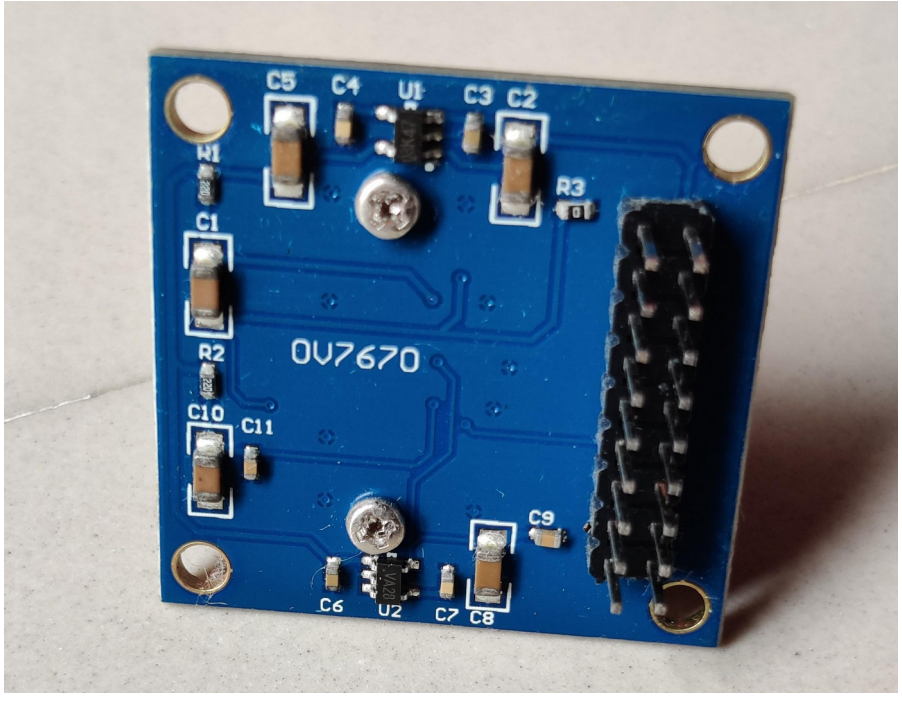




The Wiring Setup

OV7670 Module





Pin Description

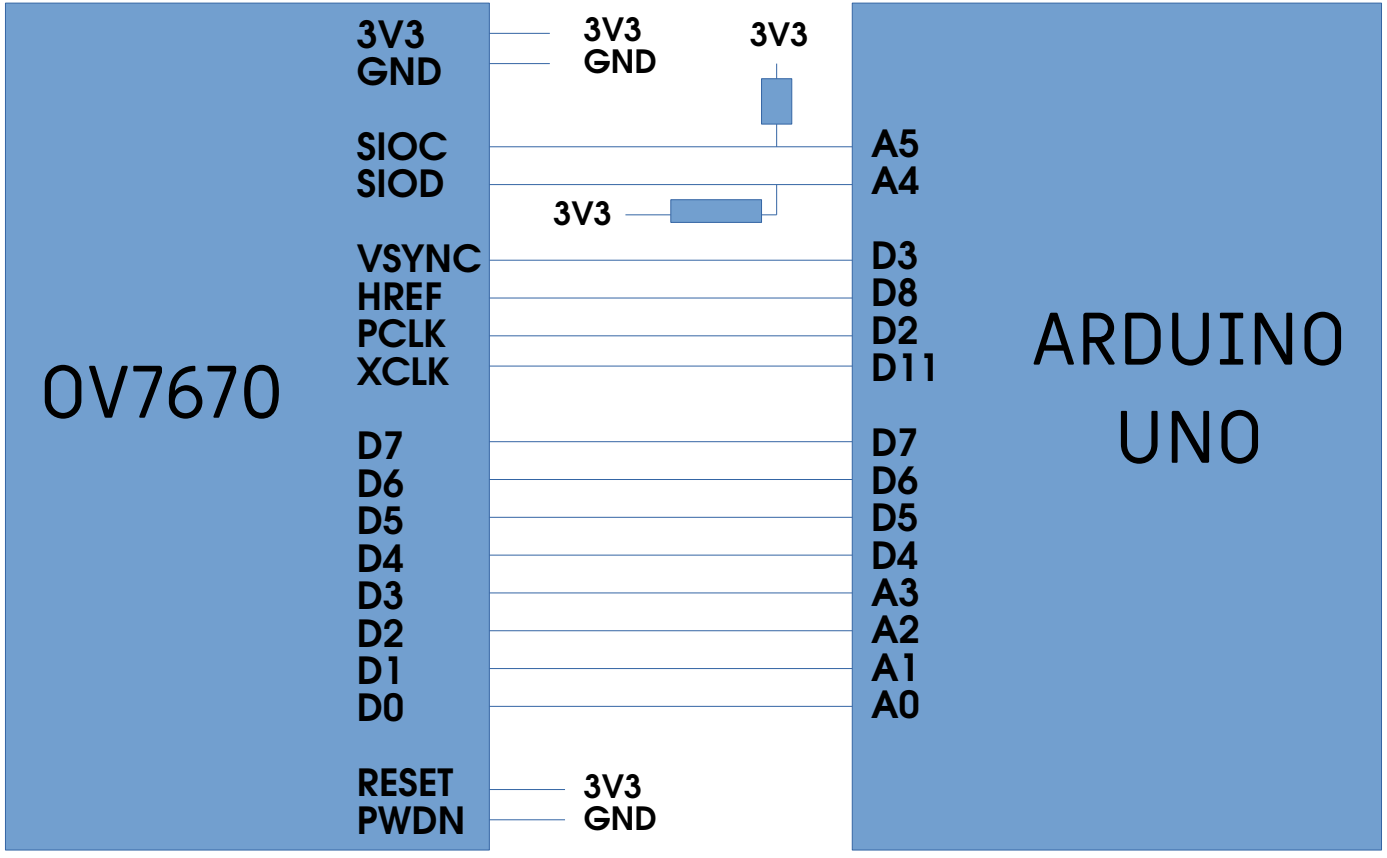
OV760 Pin	Description
3V3	VCC
GND	Ground
RESET	Reset
PWDN	Power Enable/Disable
SIOC	I2C Clock Signal
SIOD	I2C Data Signal
VSYNC	Vysnc
HREF	Hsync
PCLK	Pixel Clock
XCLK	External Clock
D0-D7	Parallel Data Output

Module Features

- Highest resolution possible is VGA resolution (640x480).
- The image resizer module inside the sensor allows to scale the images into resolutions like CIF(352x240), QVGA(320x240), QQVGA(160x120) and others. Manual adjustments to resolutions other than this is also possible.
- The maximum image transfer rate in VGA resolution is 30 frames per second.
- The sensor also performs pre-processing of images like automatic gain control (AGC), automatic exposure control(AEC), automatic white balance control (AWB) and many others.
- Various raw image encoding options like YUV422,RGB565,Bayer pattern are also available.
- The image data is transmitted through 8 bit parallel interface. The sync signals and pixel clock determine the validity of data transmitted.
- The sensor registers are accessed via I2C protocol to enable or disable a feature.
- The module needs an external clock for functioning.

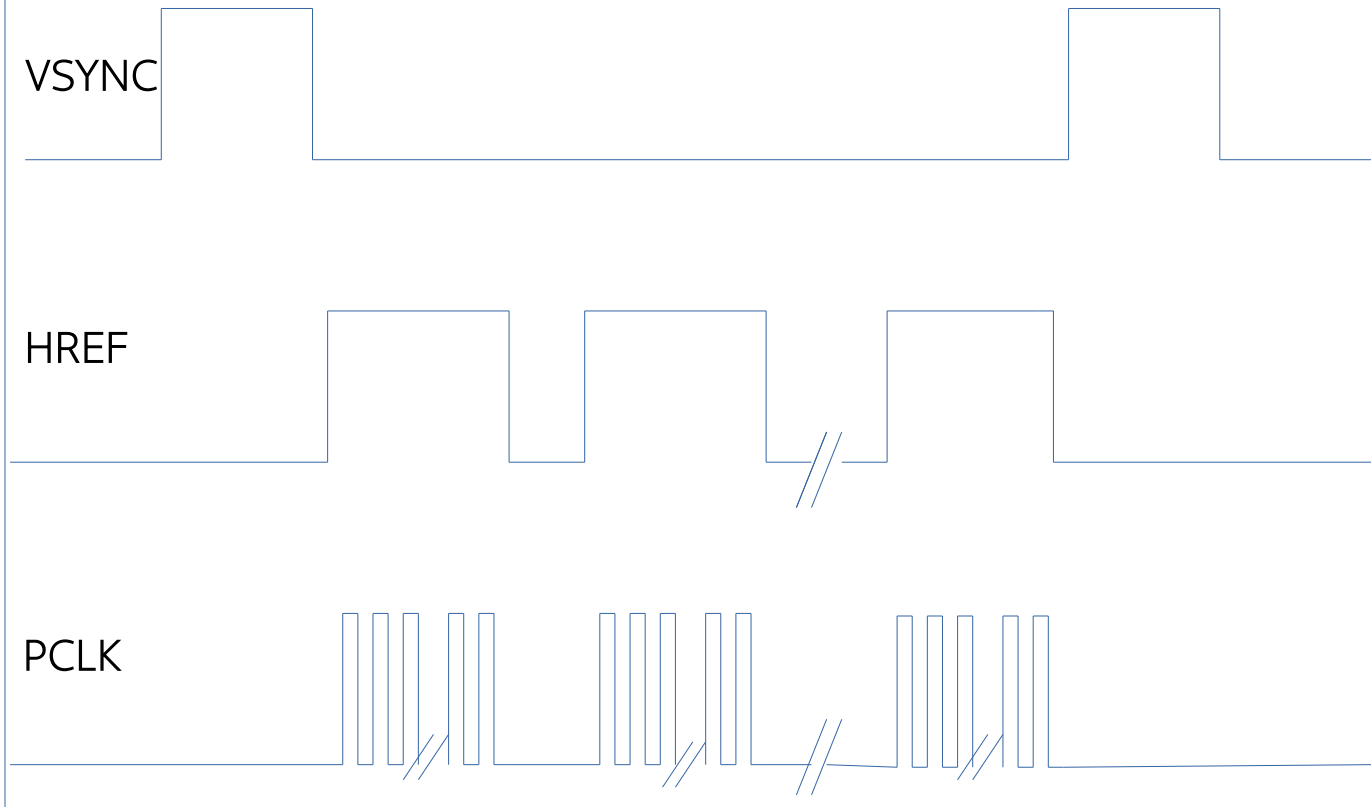
The Schematic

The working voltage of OV7670 sensor is 3.3V. The microcontroller's working voltage is 5V. So there should be a voltage translator or level converter in between for safe and correct working. BSS138 MOSFET level shifter module could be used for voltage translation. Here for demonstration we are not using level translators.



Capturing the Image

- The sensor needs an external clock for proper working.The maximum clock that could be provided by the Atmega328 is 8Mhz which is pulsed by PWM output.
- After that initialize the sensor according to requirements like resolution,colour format etc.
- PCLK toggles only when HREF is high.So first check Vsync .Wait till it is high.Wait till it is low again.Read the pixels each time PCLK toggles. After the data for one pixel has been gathered, transmit it via serial port.Repeat this for all lines.



The Arduino Code – I2C Routines

- Wire Library is used to access the I2C registers in the sensor.
- Include the header file Wire.h inside the ino file.
- Call Wire.begin(); inside setup() to initialize the Wire Library.
- The I2C read and write address of OV7670 is 0x42 and 0x43 respectively as per datasheet.
- The address has to be right shifted by 1 and then used in the I2c routines.

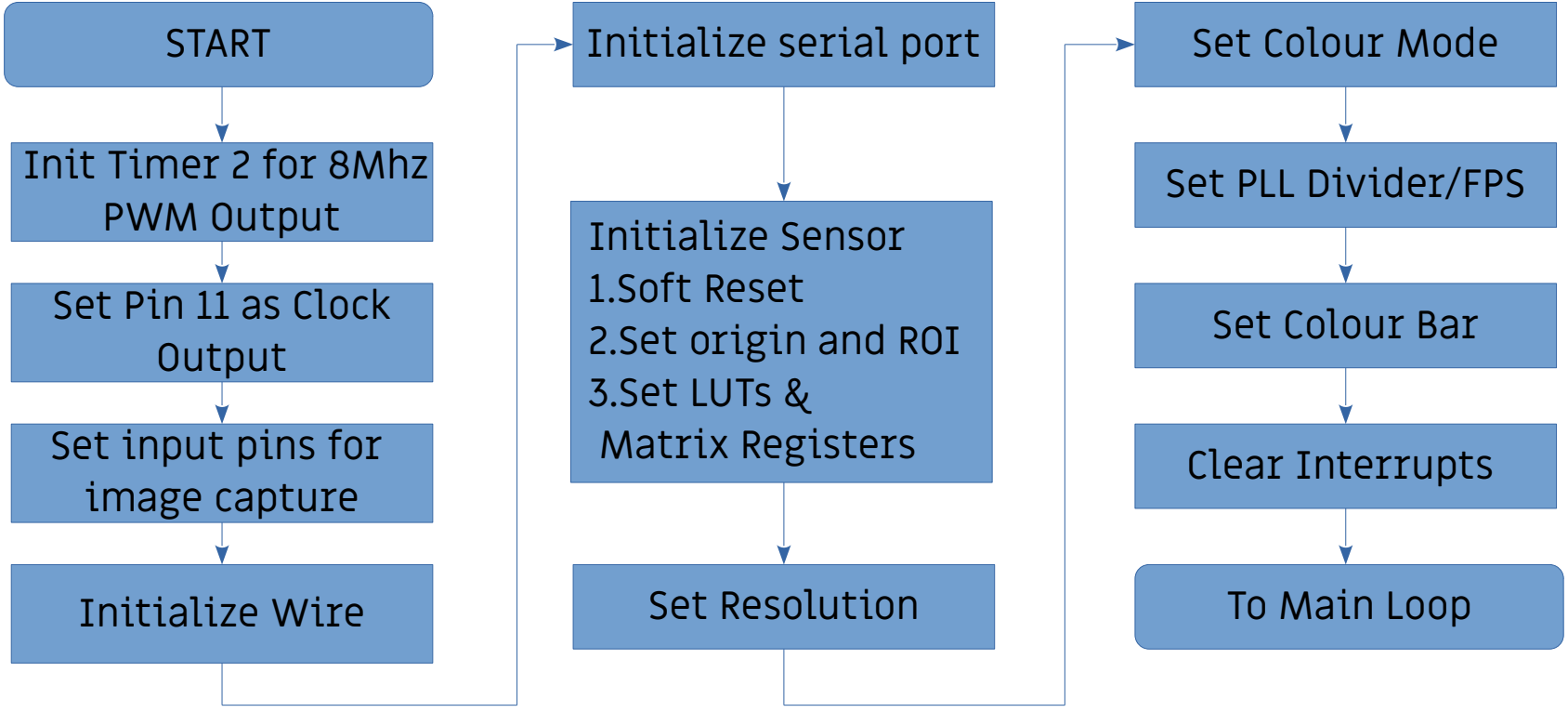
```
void wrReg(unsigned char RegisterID,unsigned char Data)

{

    Wire.beginTransmission(0x21);

    Wire.write(RegisterID);

    Wire.write(Data);

    Wire.endTransmission();

    _delay_ms(5);

}
```

The Arduino Code – Transmit Code

- The UART of Atmega328 is used to transmit the captured image data.
- The baud rate used is 2M
- A header data is transmitted at the start of every frame
- "*RDY*" is the header or marker, followed by the image sequence number. This is followed by pixel data.

```
//Header Transmit Code
StringPgm(PSTR("*RDY*"));
UDR0 = imgcnt&0xFF;
while (!(UCSR0A & (1 << UDRE0)));

//UART Init Code
UBRR0H = 0;
UBRR0L = 0;//2M baud rate
UCSR0A |= 2;//double speed aysnc
UCSR0B = (1 << RXEN0) | (1 << TXEN0);//Enable receiver and transmitter
UCSR0C = 6;//async 1 stop bit 8bit char no parity bits
```

The Arduino Code – Initialization Code



The Arduino Code – Initialization Code

```
//Setup the 8mhz PWM clock.This will
be on pin 11
DDRB |= (1 << 3);//pin 11
ASSR &= ~(_BV(EXCLK) | _BV(AS2));
TCCR2A = (1 << COM2A0) | (1 <<
WGM21) | (1 << WGM20);
TCCR2B = (1 << WGM22) | (1 << CS20);
OCR2A = 0;//(F_CPU)/(2*(X+1))


//Image Capture Pins
DDRC &= ~15;//low d0-d3 camera
DDRD &= ~252;//d7-d4 and interrupt pins
```

```
void setup()
{
    arduinoUnoInit();
    camInit();
    SetVGAMonochromeCapture
   (COLOURBAR_DISABLE);
    Wire.end();
     cli();//disable interrupts
    _delay_ms(100);
}
```

The Arduino Code – Image Capture

VGA - Monochrome

```
SetVGAMonochromeCapture();
CaptureImg(640, 480);
```

QVGA - Monochrome

```
SetQVGAMonochromeCapture();
captureImg(320, 240);
```

QQVGA - Monochrome

```
SetQQVGAMonochromeCapture();
captureImg(160, 120);
```

```
while (!(PIND & 8));//wait for high
while ((PIND & 8));//wait for low
y = hg;
while (y--)
{
      x = wg;
      while (x--)
      {
        while ((PIND & 4));//wait for low

        //for qvga , qqvga monochrome and qqvga Colour
        //UDR0 = (PINC & 15) | (PIND & 240);
        //while (!(UCSR0A & (1 << UDRE0)));

        //for vga monochrome
        //UDR0 = (PINC & 15) | (PIND & 240);

        while (!(PIND & 4));//wait for high
        while ((PIND & 4));//wait for low
        //for qqvga colour
        //UDR0 = (PINC & 15) | (PIND & 240);
        //while (!(UCSR0A & (1 << UDRE0)));
        while (!(PIND & 4));//wait for high
      }
}
```

VGA

QVGA

QQVGA

The Arduino Code – Image Capture

Colour Bar QQVGA yuv

```
SetQQVGAColourImageYUVCapture(COLOUR BAR_ENABLE)
CaptureImg(160, 120);
```

Colour Bar rgb565

```
SetQQVGAColourImageRGBCapture(COLOUR BAR_ENABLE)
CaptureImg(160, 120);
```

Colour Image QQVGA yuv

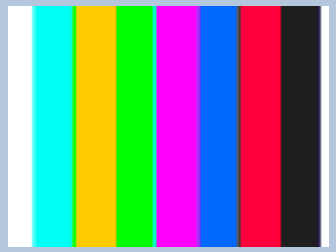
```
SetQQVGAColourImageYUVCapture(COLOUR BAR_DISABLE)
CaptureImg(160, 120);
```

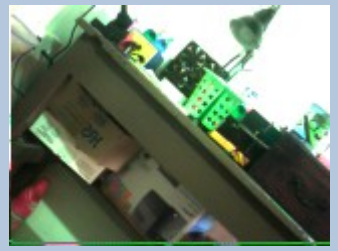
Colour Image QQVGA rgb565

```
SetQQVGAColourImageRGBCapture(COLOUR BAR_DISABLE)
CaptureImg(160, 120);
```

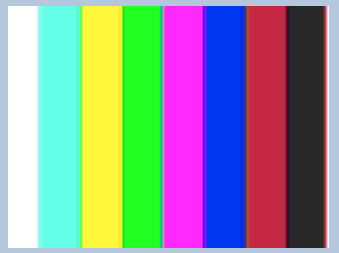
```
while (x--)
{
      while ((PIND & 4));//wait for low

      //for qqvga Colour
      //UDR0 = (PINC & 15) | (PIND & 240);
      //while (!(UCSR0A & (1 << UDRE0)));

      while (!(PIND & 4));//wait for high
      while ((PIND & 4));//wait for low

      //for qqvga colour
      UDR0 = (PINC & 15) | (PIND & 240);
      while (!(UCSR0A & (1 << UDRE0)));

      while (!(PIND & 4));//wait for high
}
```

QQVGA YUV422





QQVGA RGB565



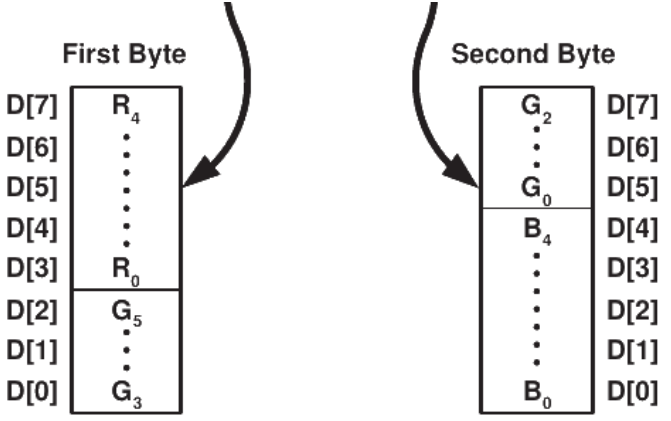
The PC Code

YUV422 toRGB888

- In yuv422 format there will be 4 bytes per 2 pixel.ie Y bytes for each pixel and shared 1 U byte and 1V byte for both pixels.
- Saving the Y byte alone gives a gray scale or monochromatic image.
- To save in bitmap format, Y values have to be converted to RGB888 format.Assign Y to R,G and B.Then call routines to save in bitmap format.
- To save in colour mode,first parse the Y,U,V bytes of each pixel correctly,then apply the standard equation to convert each pixel from YUV to RGB888 format.Then call routines to save in bitmap format.

RGB565 to RGB888

- There will be 2 bytes per pixel in rgb565 format.The bits are allocated as in figure.
- First parse the bytes and arrange them in correct order.
- Then apply the standard equation to convert each pixel from RGB565 to RGB888 format.Then call routines to save in bitmap format.



The MJPEG Streamer

JPEG Compression
Convert bitmap files to jpeg files using ImageMagick tool in linux. Install it according to the Linux Distribution.The command for usage is as follows

“convert -quality 90 file.bmp file.jpeg”.This will be a system call in the PC program.

MJPEG Streamer
Implement a simple jpeg streamer utility employing socket functions and HTTP snippets in C in Linux.The MJPEG streamer allows to view the images transmitted from Arduino on a browser.
The usage is as follows “http://ip of the computer:8080”

The PC codes and Arduino Codes are attached for reference.
The Arduino codes have been reused from other sites and modified..
PC Code is a mess. But it works. You are free to modify it and use it to build your own solutions.

THANK You