Contents

[1 ArduCAM + Arduino Mini D1 1](#_Toc497665887)

[1.1 Installing libraries 1](#_Toc497665888)

[1.1.1 Time 1](#_Toc497665889)

[1.1.2 ArduCAM 1](#_Toc497665890)

[1.1.3 ArduinoJson 1](#_Toc497665891)

[1.2 Installing OV7670 WITH ARDUCAM REV C+ camera shield with SERIAL DATA interface to Wemos D1 mini 1](#_Toc497665892)

[1.2.1 ArduCAM Rev C+ Pin connections: 1](#_Toc497665893)

[1.2.2 Hardware Circuit Connection btwn D1 and Arducam 3](#_Toc497665894)

[1.2.3 SW setup 3](#_Toc497665895)

[1.2.4 Wiring OV7670 w/o FIFO to the Arducam board 3](#_Toc497665896)

[1.2.5 Basic “is it alive” setup 5](#_Toc497665897)

[1.2.6 Wiring OV7670 FIFO 6](#_Toc497665898)

[1.2.7 Tuning 7](#_Toc497665899)

[1.2.8 Executing code 7](#_Toc497665900)

[1.2.9 Alternative solutions: 7](#_Toc497665901)

[1.2.10 Converting RAW to image 7](#_Toc497665902)

[2 Set up Arduino for camera 7](#_Toc497665903)

[2.1 Tech refs: 7](#_Toc497665904)

[2.2 Installing Uno R3 hardware and run “Blink” 7](#_Toc497665905)

[2.3 Installing OV7670 NO FIFO camera 7](#_Toc497665906)

[2.3.1 Wiring OV7670 NO FIFO 7](#_Toc497665907)

[2.3.2 Tuning 7](#_Toc497665908)

[2.3.3 Executing code 7](#_Toc497665909)

[2.3.4 Alternative solutions: 8](#_Toc497665910)

[2.3.5 Converting RAW to image 8](#_Toc497665911)

[2.4 Installing OV7670 WITH FIFO (=AL422) camera 8](#_Toc497665912)

[2.4.1 Basic “is it alive” setup 8](#_Toc497665913)

[2.4.2 Wiring OV7670 FIFO 8](#_Toc497665914)

[2.4.3 Tuning 9](#_Toc497665915)

[2.4.4 Executing code 9](#_Toc497665916)

[2.4.5 Alternative solutions: 9](#_Toc497665917)

[2.4.6 Converting RAW to image 10](#_Toc497665918)

[2.5 Installing wifi shield ESP-01 ESP-8266 10](#_Toc497665919)

[2.5.1 Wiring 10](#_Toc497665920)

[2.5.2 Booting 10](#_Toc497665921)

[2.5.3 Flashing ESP 11](#_Toc497665922)

[3 ESP8266 Arduino board (ESP12-E Arduino = D1) 13](#_Toc497665923)

[3.1 Install drivers and sw 13](#_Toc497665924)

[3.1.1 Install board on Arduino (I have a D1, not a D1R1): 13](#_Toc497665925)

[4 Barcode reader programming 13](#_Toc497665926)

[4.1 Test programs for development 13](#_Toc497665927)

[4.1.1 29/11-16: sending messages to IOT hub from Arduino using Visual Studio: 13](#_Toc497665928)

[4.2 Sending a file over wifi 13](#_Toc497665929)

[5 Visual Studio setup for Arduino 14](#_Toc497665930)

[5.1.1 Running and debugging 14](#_Toc497665931)

[5.1.2 Install the right VS extensions : 14](#_Toc497665932)

[5.1.3 Important note 14](#_Toc497665933)

[5.1.4 In case of errors 14](#_Toc497665934)

[5.1.5 Path variables 14](#_Toc497665935)

[5.1.6 Install curl library: 15](#_Toc497665936)

[6 Java (not used) 16](#_Toc497665937)

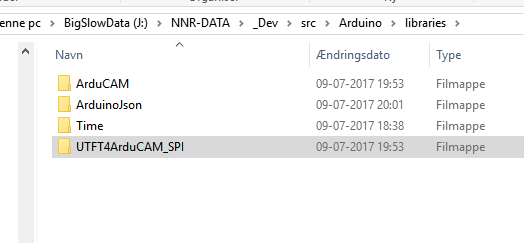
# ArduCAM + Arduino Mini D1

## Installing libraries

You need the following libraries into the Arduino library. Find this from the Arduino IDE => File => Settings.   
The correct folder is <sketchbook location>/libraries.   
On Dell PC: J:\NNR-DATA\\_Dev\src\Arduino\libraries

NOTE: If you want to MODIFY the libraries, they must be installed in a user library folder.  
On Dell-work PC, this is in C:\data\_no\_backup\DevelopmentPriv\Arduino\libraries  
On Dell-work PC, the regular library folder is in C:\Program Files (x86)\Arduino\libraries

When done, it looks like this:



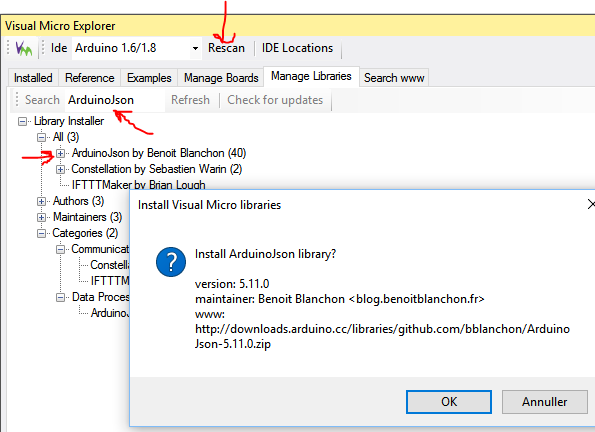
### Time

* Download and install from ZIP using Visual Studio

### ArduCAM

* Download from <https://github.com/ArduCAM/Arduino>
* Copy the two directories ArduCAM and UTFT4ArduCAM\_SPI (leave the rest) to the Arduino library folder (see above for how to find it). Remember to refresh in Visual Stuio (or restart).

### ArduinoJson



## Installing OV7670 WITH ARDUCAM REV C+ camera shield with SERIAL DATA interface to Wemos D1 mini

The D1 mini does not have 8 available data inputs, hence we need to connect it using serial data interface.

Links:

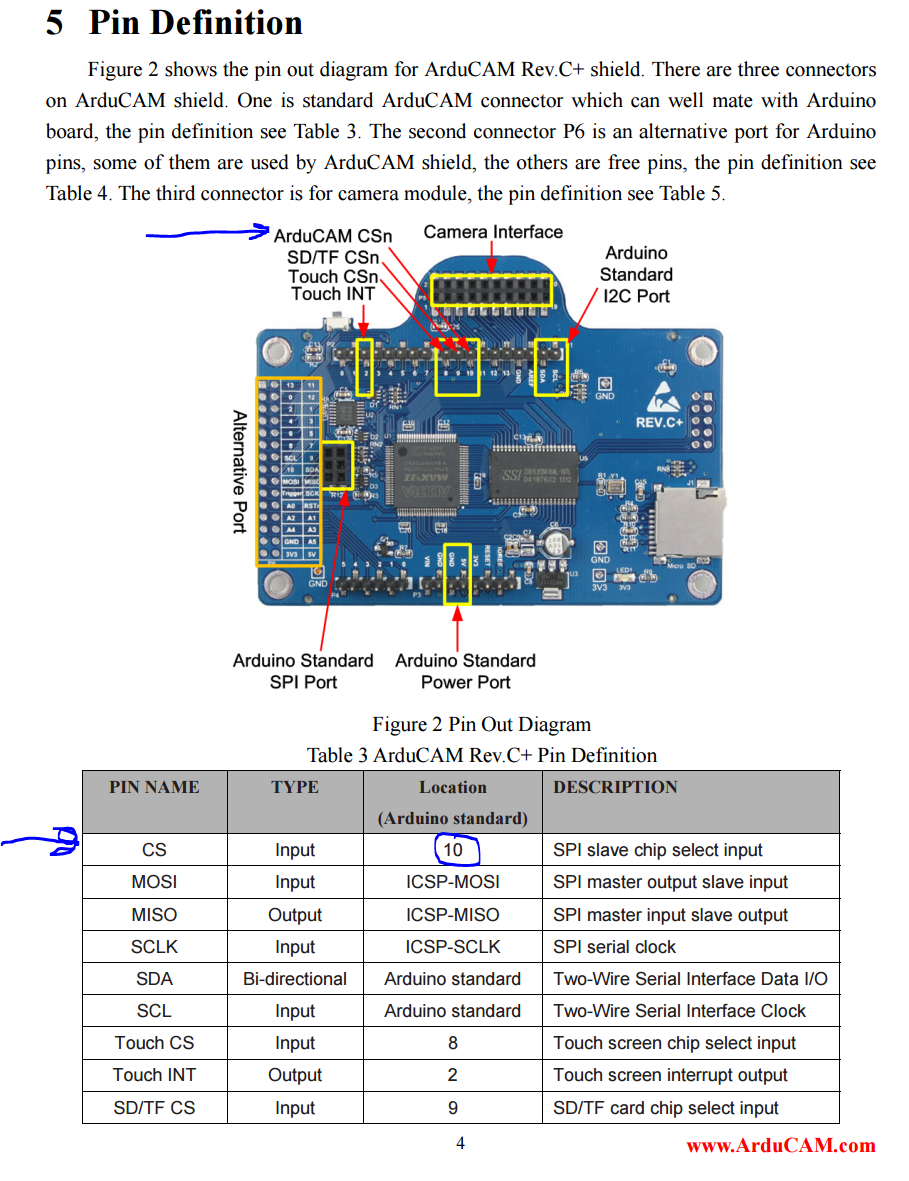
* <http://usemodj.com/2016/08/25/esp8266-arducam-5mp-ov5642-camera-wifi-video-streaming/> (instructions for this build)
* <http://www.arducam.com/downloads/shields/ArduCAM_RevC+_Camera_Shield_DS.pdf> (user manual for Arducam shield)
* <http://www.arducam.com/arducam-rev-c-plus-shield-released/#more-991>
* <http://www.arducam.com/category/introduction/> (I have not used the hardware – only the SW)
* <https://www.wemos.cc/product/d1-mini.html> (Wemos D1 Mini reference)

Hardware note: UC-260 camera module = OV2640 2Mp camera

Follow instructions on <http://usemodj.com/2016/08/25/esp8266-arducam-5mp-ov5642-camera-wifi-video-streaming/>

### ArduCAM Rev C+ Pin connections:

<http://www.arducam.com/downloads/shields/ArduCAM_RevC+_Camera_Shield_DS.pdf>



Figur 1. Arducam REV C+ pin layout

4: VCC

2: SCK

5: MOSI

6: GND

1: RESET

3: MISO

Figur 2. ICSP (=SPI port) seen from front of Arducam = as in picture above

Wemos D1 MINI

|  |  |  |
| --- | --- | --- |
| **Pin** | **Function** | **ESP-8266 Pin** |
| TX | TXD | TXD |
| RX | RXD | RXD |
| A0 | Analog input, max 3.3V input | A0 |
| D0 | IO | GPIO16 |
| D1 | IO, SCL | GPIO5 |
| D2 | IO, SDA | GPIO4 |
| D3 | IO, 10k Pull-up | GPIO0 |
| D4 | IO, 10k Pull-up, BUILTIN\_LED | GPIO2 |
| D5 | IO, SCK | GPIO14 |
| D6 | IO, MISO | GPIO12 |
| D7 | IO, MOSI | GPIO13 |
| D8 | IO, 10k Pull-down, SS | GPIO15 |
| G | Ground | GND |
| 5V | 5V | - |
| 3V3 | 3.3V | 3.3V |
| RST | Reset | RST |

Figur 3. Wemos D1 Mini pin layout

### Hardware Circuit Connection btwn D1 and Arducam

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Pin function | Arducam Rev C+  pin name | Pin position | Wemos D1 mini name | D1 mini pin position | Arduino UNO name | Pin position | Typical name in code |
| CS, SD card | 9 | P1-9 |  |  | 9 | 9 | SD\_CS |
| CS, SPI port | 10 | P1-10 | D0 (GPIO 16) | Upper-6 | 10 | 10 | SPI\_CS |
| SDA | SDA | P1-16 | D2 (GPIO 4) | Lower-5 | SDA |  |  |
| SCL | SCL | P1-17 | D1 (GPIO 5) | Lower-6 | SCA |  |  |
| RESET | - | SPI-1 |  |  | (SPI port, reset) | SPI-1 |  |
| SCK | SCK | SPI-2 | D5 (GPIO 14) | Upper-5 | (SPI port, SCK) | SPI-2 |  |
| MISO | MISO | SPI-3 | D6 (GPIO 12) | Upper-4 | (SPI port,, MISO | SPI-3 |  |
| 5V | 5V | P3-4 | 5V | Lower-1 | (SPI port, 5V) | SPI-4 |  |
| MOSI | MOSI | SPI-5 | D7 (GPIO 13) | Upper-3 | (SPI port,MOSI) | SPI-5 |  |
| GND | GND | SPI-6 | G (GND) | Lower-2 | (SPI-port gnc) | SPI-6 |  |

### SW setup

Modifications from instruction notes (<http://usemodj.com/2016/08/25/esp8266-arducam-5mp-ov5642-camera-wifi-video-streaming/>):

Section “Required libraries” in instruction notes

* Arduino library named *arduCAM* in C:\Users\Nikolaj\AppData\Local\Arduino15\staging\libraries **ELLER** J:\Programmer\Arduino\libraries\

Modify Memorysaver.h:

#define ARDUCAM\_SHIELD\_REVC

//#define ARDUCAM\_SHIELD\_V2

//Step 2: Select one of the camera module, only one at a time

#if (defined(ARDUCAM\_SHIELD\_REVC) || defined(ARDUCAM\_SHIELD\_V2))

#define OV2640\_CAM

Change Wifi AP

const char\* ssid = "TeliaGateway58-98-35-B5-DB-17"; // Put your SSID here

const char\* password = "E5B004E62E"; // Put your PASSWORD here

**OLD STUFF FROM HERE ONWARDS:**

### Wiring OV7670 w/o FIFO to the Arducam board

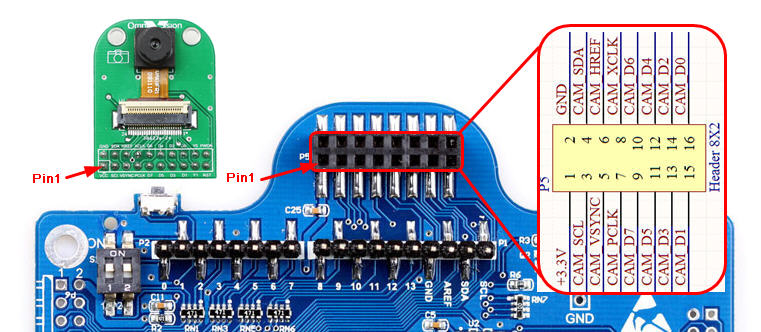


Figure 1. Arducam Rev C+ pinout (http://www.arducam.com/hardware/)

Arducam Rev C+ pinout

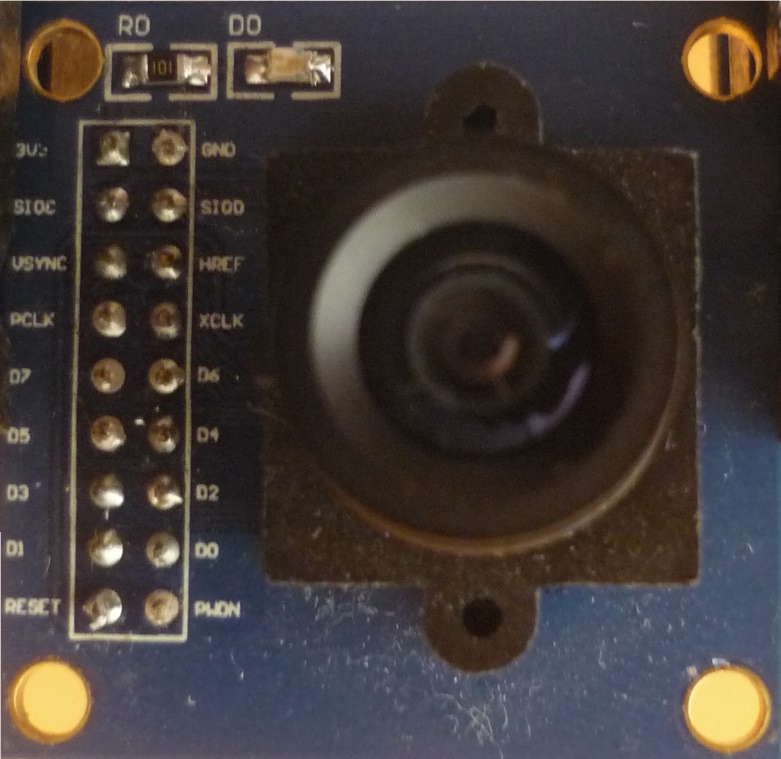


Figure 2. Arducam “7670” cam module pinout (<http://www.arducam.com/camera-modules/0-3mp-ov7670/ov7670_3-2/>) versus standard 7670 cam module. NOTE: one shot from front, the other from back

Pins match just fine:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pin nbr | Pin name, uc-246 | OV7670 pin | OV7670 FIFO pin | Arducam Rev C+ camera interface pins |
| 1 | 3V3 | 3V3 | 3V3 | +3.3V |
| 3 | SCL | SIOC | SIOC | SCL |
| 5 | VS | VSYNC | VSYNC | VSYNC |
| 7 | PCK | PCLK | D7 | PCLK |
| 9 | D7 | D7 | D5 | D7 |
| 11 | D5 | D5 | D3 | D5 |
| 13 | D3 | D3 | D1 | D3 |
| 15 | D1 | D1 | RST | D1 |
|  |  |  | STR |  |
|  |  |  | WR |  |
|  |  |  | WRST |  |
|  |  |  |  |  |
| 2 | GND | GND | GND | GND |
| 4 | SDA | SIOD | SIOD | SDA |
| 6 | HS | HREF | HREF | HREF |
| 8 | XCK | XCLK | D6 | XCLK |
| 10 | D6 | D6 | D4 | D6 |
| 12 | D4 | D4 | D2 | D4 |
| 14 | D2 | D2 | D0 | D2 |
| 16 | D0 | D0 | PWDN | D0 |
|  |  |  | RCK |  |
|  |  |  | OE |  |
|  |  |  | RRST |  |

### Basic “is it alive” setup

Note: The Arduino WeMOS ESP-12 D1 board uses 3.3V, i.e. pull-ups are not necessary between D1 board and Arducam (which handles both 3.3 and 5V).

|  |  |  |
| --- | --- | --- |
| Wemos ESP8266 D1 | Arducam |  |
|  | VCC  SCK  MOSI  GND  RESET  MISO  (SPI port seen from front of Arducam = as in picture above) | (for reference only) |
| On-board pin name | On-board pin name (on SPI port) |  |
| 3.3V | VCC |  |
| D12/MISO | MISO |  |
| D11/MOSI | MOSI |  |
| D13/SCK | SCK |  |
| GND | GND |  |
| GND or 3.3V ?? | RESET |  |
|  |  |  |
|  |  |  |

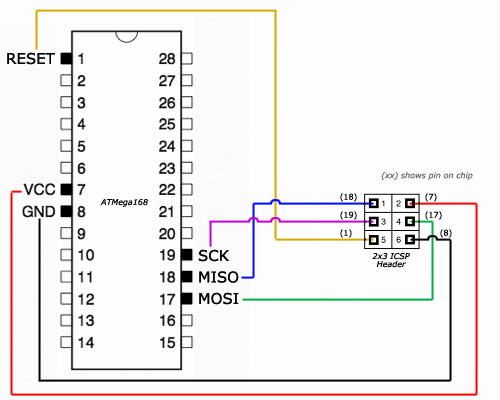


Figure 3. Schematic pin config for SPI-port (male pins located on Arduino). Must be mirrored to reflect Arducam.

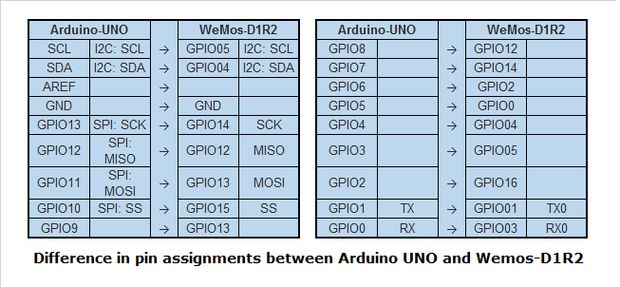
Sketch supporting this test: OV7670FIFO\_simple\_communication.ino

### Wiring OV7670 FIFO

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ESP8266 D1 |  |  |  |  |  | OV7670 FIFO |  |  |
| Name on board | Pin number (pin1=RX) | PIN pins\_arduino.h & common.h | PIN Symbol inherited from pins\_arduino.h & common.h in ov7670fifo.c | PIN Symbol value in ov7670fifo.c | PIN Nbr  new sw | Name on board | Purpose | In or out |
| 3.3V |  |  |  |  |  | 3.3V |  |  |
| GND |  |  |  |  |  | GND |  |  |
| RX<-D0 | 1 | 3 |  |  |  |  |  |  |
| TX->D1 | 2 |  |  |  |  |  |  |  |
| D2 | 3 |  | DO0 = D2 | 16 |  |  |  |  |
| D15/SCL/D3 | 4 | 5 | DO7 = D3 | 5 | 17 | SIOC | SCCB i/f ctrl clock | IN |
| D14/SDA/D4 | 5 | 4 | DO4 = D4 | 4 | 18 | SIOD | SCCB i/f serial data | IN/OUT |
| D13/SCK/D5 | 6 | 14 | DO3 = D13 DO5 = D5 | 14 (/ 23) | 27 | RRST (blue / old:black) | Read pointer reset terminal |  |
| D12/MISO/D6 | 7 | 12 | DO2 = D12 DO6 = D6 | 12 (/ 25) | 28 | WRST (pink / old:white) | Write pointer reset terminal |  |
| D11/MOSI/D7 | 8 | 13 | DO1 = D11 | 13 |  | n/c |  |  |
| D8 | 9 |  |  |  |  | RCK | FIFO memory read clock control |  |
| TX1/D9 | 10 |  |  |  |  | WR | FIFO write control (1=write, 0=no write) | IN |
| D10/SS | 11 | 15 | SS | 15 |  |  |  |  |
| D11/MOSI | 12 |  | MOSI\_PIN | 13 |  |  |  |  |
| D12/MISO | 13 |  | SPI\_SCK\_PIN | 14 |  |  |  |  |
| D13/SCK | 14 |  |  |  |  |  |  |  |
| GND | 15 |  |  |  |  | OE (green / old:green) | FIFO off control | IN |
| (blank) | 16 |  |  |  |  |  |  |  |
| D14/SCM | 17 |  |  |  |  |  |  |  |
| D15/SCL | 18 |  |  |  |  |  |  |  |
| SS |  | 15 |  |  |  |  |  |  |
|  |  |  |  |  |  | HREF | Line sync | OUT |
|  |  |  |  | 22 |  | VSYNC | Frame sync | OUT |
| GND |  |  |  |  |  | PWDN | Power down | IN |
|  |  |  |  | 35 |  | DO0 |  |  |
| 3.3V |  |  |  | 26 |  | RST | Reset | IN |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| D6 | 7 |  | DO2/DO6 | 12 |  |  |  |  |
| D7 | 8 |  | DO1 | 13 |  |  |  |  |
| D8 |  |  |  |  |  |  |  |  |

The pins on OV7670 are explained here: <http://www.electrodragon.com/w/OV7670_Module>

Relations to Arduino UNO pins: <http://www.instructables.com/id/Programming-the-WeMos-Using-Arduino-SoftwareIDE/?ALLSTEPS>





### Tuning

Blurry picture: Turn the lens manually until it’s sharp.

### Executing code

The COM port is used for communication between PC side and Arduino as follows:

Use this setup:



#### PC side:

Use ReadSerialPortWin project:

1. Start the program
2. Select the COM port
3. WAIT pressing start until the Arduino program has been loaded

#### Arduino side

Download the program via the COM port before step 3 above.

Do NOT use Serial.begin, Serial.println etc. Everything written to Serial is (of course) sent to the PC program

DO NOT use the port monitor in Arduino IDE / Visual Studio. This will lock the COM port, and then the PC app cannot read it.

### Alternative solutions:

NOTE: There may be need for an external oscillator. However, I suspect that this is only needed if we need video. If that’s the case, have a look at this video: <https://www.youtube.com/watch?v=gp0FxbEmRSw>

This link seems pretty solid, but also use an oscillator: <http://www.arducam.com/camera-modules/0-3mp-ov7670/>

### Converting RAW to image

# Set up Arduino for camera

## Tech refs:

Pin layout: <https://github.com/Bouni/Arduino-Pinout>

## Installing Uno R3 hardware and run “Blink”

<https://learn.adafruit.com/lesson-0-getting-started/breadboard?view=all>

* Download exe file and install. The Arduino IDE will find the board itself.
* Set correct COM port (On [\\Z](file:///\\Z) it’s COM4)

<https://learn.adafruit.com/adafruit-arduino-lesson-1-blink/other-things-to-do?view=all>

* (just to verify that there is hole through.

## Installing OV7670 NO FIFO camera

This link states: *“if you just want to send data from the camera to your PC using the Arduino you don't need external RAM. Here is the code that will accomplish that”:* <http://forum.arduino.cc/index.php/topic,159557.0.html>

<http://forum.arduino.cc/index.php?topic=159557.780>   
Post #774 has some interesting pin connections to a wifi shield (not the one I have, though).  
Post #754 seems interesting

We will in any case need to convert from 5V to 3.3V, and this is shown in this link.  
However, the link has some notes stating that it’s the wrong camera type. In contrast to this, the picture shows the right model, and other notes states that it works great. So give it a try.   
<http://www.elecfreaks.com/projects/how-to-use-ov7670-camera-module-with-arduino/>

### Wiring OV7670 NO FIFO

<https://github.com/ComputerNerd/ov7670-no-ram-arduino-uno>



### Tuning

Blurry picture: Turn the lens manually until it’s sharp.

### Executing code

The COM port is used for communication between PC side and Arduino as follows:

#### PC side:

Use ReadSerialPortWin project:

1. Start the program
2. Select the COM port
3. WAIT pressing start until the Arduino program has been loaded

#### Arduino side

Download the program via the COM port before step 3 above.

Do NOT use Serial.begin, Serial.println etc. Everything written to Serial is (of course) sent to the PC program

DO NOT use the port monitor in Arduino IDE / Visual Studio. This will lock the COM port, and then the PC app cannot read it.

### Alternative solutions:

NOTE: There may be need for an external oscillator. However, I suspect that this is only needed if we need video. If that’s the case, have a look at this video: <https://www.youtube.com/watch?v=gp0FxbEmRSw>

This link seems pretty solid, but also use an oscillator: <http://www.arducam.com/camera-modules/0-3mp-ov7670/>

### Converting RAW to image

HERTIL

## Installing OV7670 WITH FIFO (=AL422) camera

See also section on OV7670 with FIFO

Links:

* <https://github.com/ArduCAM/Arduino>
* <http://www.arducam.com/category/introduction/> (I have not used the hardware – only the SW)

<http://forum.arduino.cc/index.php?topic=159557.780>   
Post #774 has some interesting pin connections to a wifi shield (not the one I have, though).  
Post #754 seems interesting

We will in any case need to convert from 5V to 3.3V, and this is shown in this link.  
However, the link has some notes stating that it’s the wrong camera type. In contrast to this, the picture shows the right model, and other notes states that it works great. So give it a try.   
<http://www.elecfreaks.com/projects/how-to-use-ov7670-camera-module-with-arduino/>

### Basic “is it alive” setup

With this setup, it’s possible to communicate to the OV7670 (i.e. setting the registers etc.). It is NOT possible to retrieve picture data (no data lines are connected).

Note: The Arduino WeMOS ESP-12 D1 board uses 3.3V, i.e. pull-ups are not necessary between D1 board and OV7670. This is NOT the case for an Arduino UNO board (here, pull-ups must be used).

|  |  |  |
| --- | --- | --- |
| ESP8266 D1 | OV7670 FIFO |  |
| On-board pin name | On-board pin name |  |
| 3.3V | 3.3V |  |
| GND | GND |  |
| D15/SCL/D3 | SIOC |  |
| D14/SDA/D4 | SIOD |  |
| GND | PWDN |  |
| 3.3V | RST |  |

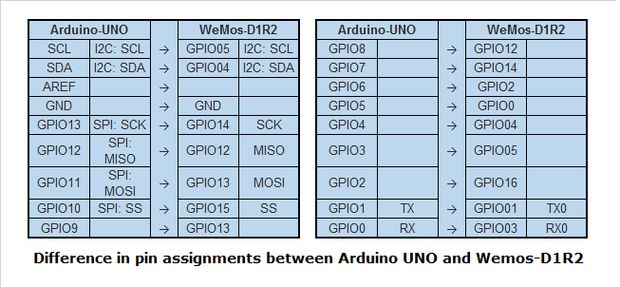
Sketch supporting this test: OV7670FIFO\_simple\_communication.ino

### Wiring OV7670 FIFO

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ESP8266 D1 |  |  |  |  |  | OV7670 FIFO |  |  |
| Name on board | Pin number (pin1=RX) | PIN pins\_arduino.h & common.h | PIN Symbol inherited from pins\_arduino.h & common.h in ov7670fifo.c | PIN Symbol value in ov7670fifo.c | PIN Nbr  new sw | Name on board | Purpose | In or out |
| 3.3V |  |  |  |  |  | 3.3V |  |  |
| GND |  |  |  |  |  | GND |  |  |
| RX<-D0 | 1 | 3 |  |  |  |  |  |  |
| TX->D1 | 2 |  |  |  |  |  |  |  |
| D2 | 3 |  | DO0 = D2 | 16 |  |  |  |  |
| D15/SCL/D3 | 4 | 5 | DO7 = D3 | 5 | 17 | SIOC | SCCB i/f ctrl clock | IN |
| D14/SDA/D4 | 5 | 4 | DO4 = D4 | 4 | 18 | SIOD | SCCB i/f serial data | IN/OUT |
| D13/SCK/D5 | 6 | 14 | DO3 = D13 DO5 = D5 | 14 (/ 23) | 27 | RRST (blue / old:black) | Read pointer reset terminal |  |
| D12/MISO/D6 | 7 | 12 | DO2 = D12 DO6 = D6 | 12 (/ 25) | 28 | WRST (pink / old:white) | Write pointer reset terminal |  |
| D11/MOSI/D7 | 8 | 13 | DO1 = D11 | 13 |  | n/c |  |  |
| D8 | 9 |  |  |  |  | RCK | FIFO memory read clock control |  |
| TX1/D9 | 10 |  |  |  |  | WR | FIFO write control (1=write, 0=no write) | IN |
| D10/SS | 11 | 15 | SS | 15 |  |  |  |  |
| D11/MOSI | 12 |  | MOSI\_PIN | 13 |  |  |  |  |
| D12/MISO | 13 |  | SPI\_SCK\_PIN | 14 |  |  |  |  |
| D13/SCK | 14 |  |  |  |  |  |  |  |
| GND | 15 |  |  |  |  | OE (green / old:green) | FIFO off control | IN |
| (blank) | 16 |  |  |  |  |  |  |  |
| D14/SCM | 17 |  |  |  |  |  |  |  |
| D15/SCL | 18 |  |  |  |  |  |  |  |
| SS |  | 15 |  |  |  |  |  |  |
|  |  |  |  |  |  | HREF | Line sync | OUT |
|  |  |  |  | 22 |  | VSYNC | Frame sync | OUT |
| GND |  |  |  |  |  | PWDN | Power down | IN |
|  |  |  |  | 35 |  | DO0 |  |  |
| 3.3V |  |  |  | 26 |  | RST | Reset | IN |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| D6 | 7 |  | DO2/DO6 | 12 |  |  |  |  |
| D7 | 8 |  | DO1 | 13 |  |  |  |  |
| D8 |  |  |  |  |  |  |  |  |

The pins on OV7670 are explained here: <http://www.electrodragon.com/w/OV7670_Module>

Relations to Arduino UNO pins: <http://www.instructables.com/id/Programming-the-WeMos-Using-Arduino-SoftwareIDE/?ALLSTEPS>





### Tuning

Blurry picture: Turn the lens manually until it’s sharp.

### Executing code

The COM port is used for communication between PC side and Arduino as follows:

Use this setup:



#### PC side:

Use ReadSerialPortWin project:

1. Start the program
2. Select the COM port
3. WAIT pressing start until the Arduino program has been loaded

#### Arduino side

Download the program via the COM port before step 3 above.

Do NOT use Serial.begin, Serial.println etc. Everything written to Serial is (of course) sent to the PC program

DO NOT use the port monitor in Arduino IDE / Visual Studio. This will lock the COM port, and then the PC app cannot read it.

### Alternative solutions:

NOTE: There may be need for an external oscillator. However, I suspect that this is only needed if we need video. If that’s the case, have a look at this video: <https://www.youtube.com/watch?v=gp0FxbEmRSw>

This link seems pretty solid, but also use an oscillator: <http://www.arducam.com/camera-modules/0-3mp-ov7670/>

### Converting RAW to image

HERTIL

## Installing wifi shield ESP-01 ESP-8266

This one works: <http://www.esp8266.com/wiki/doku.php?id=getting-started-with-the-esp8266>   
Tricks:

* GPI0 control ESP’s boot mode. Low => flash mode, high => normal boot.
* Use external 3.3V power supply.
* Use 3.3/5.0 V bi-logic converter between Arduino and ESP.
* Out of the box BAUD rate is 115200.
* Arduino’s SoftSerial is max 57600 (on receive. It can transmit with 115200).
* Default set up as AP with SSID is AI-THINKER\_0956F8 (last 6 is MAC addr).

Key Arduino sketches:

* Setup\_wifi: has all the setup needed
* TEST\_WIFI is the most basic script. It only sets up communication. Default baud is 115200.

Baud rate. Get it right in all 3 places (ESP, Arduino, and terminal window):

* Initialize all 3 with the current BAUD (probably 115200)
* Run AT command AT+IPR=57600
* Change BAUD in the Arduino script and rerun
* Change Baud in the terminal window

Restart:

* After power cycle, the ESP will reset to 115200 BAUD. To re-establish do exactly:
* Turn off both Arduino and ESP
* Wait a while
* Turn on ESP
* Set BAUD correct in Arduino Sketch and monitor (to 115200)
* Reset ESP again
* Make sure there’s some kind of communication.
* Write new BAUD (57600) to ESP
* Set BAUD to 57600 in Arduino Sketch and monitor (to 115200)

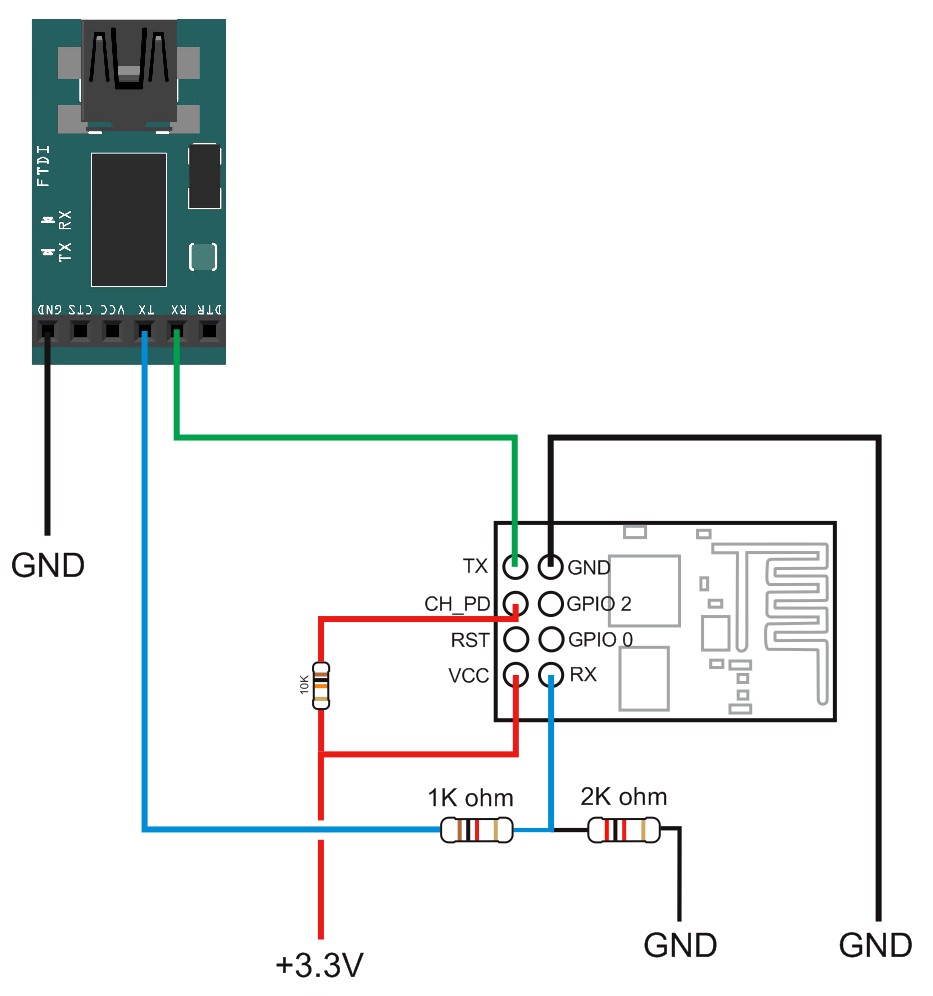
How-to: <http://shin-ajaran.blogspot.dk/2014/12/noobs-guide-to-esp8266-with-arduino.html>

NOTE: SSID is AI-THINKER\_0956F8 (last 6 is MAC).

<http://www.instructables.com/id/noobs-guide-to-ESP8266-with-Arduino-Mega-2560-or-U/>   
 (or: <http://dalpix.com/blog/connecting-your-arduino-wifi-esp-8266-module> )

Ref doc: <http://www.pighixxx.com/test/wp-content/uploads/2014/12/ESP8266Ref.pdf>

### Wiring



### Booting

#### Note that boot BAUD rate is 74880 !!!!! non-marked version

Boot read-out:

* ets Jan 8 2013,rst cause:2, boot mode:(3,0)
* load 0x40100000, len 1856, room 16
* tail 0
* chksum 0x63
* load 0x3ffe8000, len 776, room 8
* tail 0
* chksum 0x02
* load 0x3ffe8310, len 552, room 8
* tail 0
* chksum 0x79
* csum 0x79
* 2nd boot version : 1.5
* SPI Speed : 40MHz
* SPI Mode : DIO
* SPI Flash Size & Map: 8Mbit(512KB+512KB)
* jump to run user1 @ 1000
* n-á
* SD+P×\*Õîj­(
* FROM TOOL:
* flash vendor:
* E0h : N/A
* flash devID:
* 4014h
* QUAD;8Mbit
* crystal:
* 26 Mhz

### Flashing ESP

#### Flash using Python tool (esptool.py) on Windows 10 (x64)

* Install version 2.7 Python: <https://docs.python.org/3/using/windows.html>
* Install GIT
* git clone https://github.com/themadinventor/esptool.git
* Install PySerial from normal command prompt (<http://pythonhosted.org/pyserial/pyserial.html#installation> ):
* I:\DEV\Arduino\ESP flasher tool\pyserial-3.1.1>py setup.py install

#### Flashing:

<https://learn.adafruit.com/building-and-running-micropython-on-the-esp8266/flash-firmware>

* Copy bin file (e.g. “v2.0 AT Firmware(ESP).bin”) to esptool folder.
* cd I:\DEV\Arduino\GIT\esptool
* py esptool.py -p COM7 --baud 460800 write\_flash --flash\_size=8m 0 "v2.0 AT Firmware(ESP).bin"
* py esptool.py -p COM7 --baud 9600 write\_flash -fm dio -fs 8m 0 blank\_1MB.bin

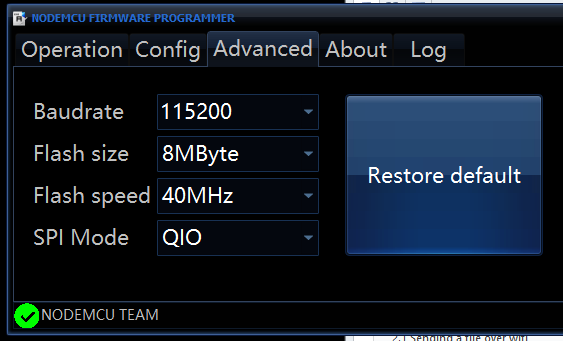
(Note: I had to correct the esptool.py. A lot of “print” without brackets. Seems not to work in this Python version, i.e. print ‘xx’ => print(‘xx’)

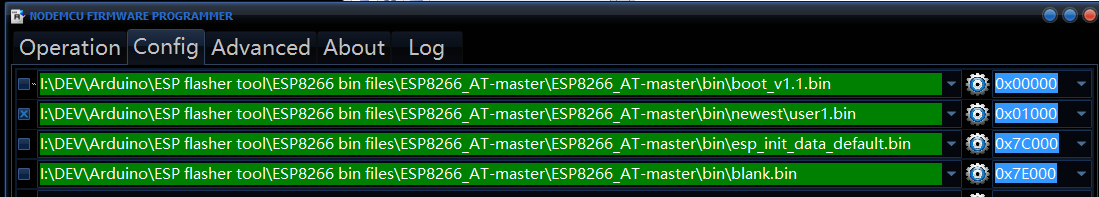
When you're ready to flash the firmware, first put the ESP8266 into its programming mode.  With the Huzzah ESP breakout do this by holding the GPIO0 button and pressing the reset button, then releasing reset and finally releasing GPIO0.  With a bare ESP8266 breakout you'll want to [follow instructions like these](http://hackaday.com/2015/03/18/how-to-directly-program-an-inexpensive-esp8266-wifi-module/) to manually pull the GPIO0 pin low and reset the board.

#### Other flash tools

* Download flash tool: <http://www.electrodragon.com/w/Category:ESP8266_Firmware_and_SDK#Default_Firmware_.28ESP-12F.2C_and_the_series.29> (password is “electrodragon”)
* Run ESP\_DOWNLOAD\_TOOL\_V2.4.exe
* Download BIN from <https://github.com/espressif/ESP8266_AT>

#### Black version:

* ets Jan 8 2013,rst cause:2, boot mode:(3,7)
* load 0x40100000, len 612, room 16
* tail 4
* chksum 0x12
* load 0x3ffe8000, len 788, room 4
* tail 0
* chksum 0x50
* load 0x3ffe8314, len 264, room 8
* tail 0
* chksum 0x4a
* csum 0x4a
* 2nd boot version : 1.1
* SPI Speed : 40MHz
* SPI Mode : QIO
* SPI Flash Size : 4Mbit
* jump to run user1
* user code done
* 



NOTE: Unplug ESP completely and re-insert immediately before flashing. Power off + power on is not enough.

#### Black cross version

* ets Jan 8 2013,rst cause:1, boot mode:(3,7)
* load 0x40100000, len 612, room 16
* tail 4
* chksum 0x12
* load 0x3ffe8000, len 788, room 4
* tail 0
* chksum 0x50
* load 0x3ffe8314, len 264, room 8
* tail 0
* chksum 0x4a
* csum 0x4a
* 2nd boot version : 1.1
* SPI Speed : 40MHz
* SPI Mode : DIO
* SPI Flash Size : 16Mbit
* jump to run user1
* Fatal exception (28):
* epc1=0x40001800, epc2=0x00000000, epc3=0x00000000, excvaddr=0x14a1894c, depc=0x00000000



Best result so far:

* Connect ESP and FTDI tool as shown here: <http://www.martyncurrey.com/arduino-esp8266/>
* Start NODEMCU flash tool “ESP8266Flasher.exe”
* Set 5/3.3 switch to 3.3
* Connect FTDI via USB
* Note in the flash tool, which COM port shows up. Maybe COM7

To enter flash mode on ESP:

* Set GPIO to **GND** (and keep it there)
* “GPIO0- > GND ( Only for when loading the firmware because this pin is what decides if to boot in firmware load mode or regular mode)”
* Set RST to GND
* Set RST floating (or high)
* Now chip is in flash mode

Press Flash

To exit flash mode and transmit normally

* Set GPIO to **3.3V** (and keep it there)
* Set RST to GND
* Set RST floating (or high)
* Now chip is in flash mode

I obtained the flasher software from:

<https://github.com/nodemcu/nodemcu-flasher>

and the latest firmware from:

<https://github.com/nodemcu/nodemcu-firmware>

# ESP8266 Arduino board (ESP12-E Arduino = D1)

## Install drivers and sw

### Install board on Arduino (I have a D1, not a D1R1):

1. Install the “CH340 USB to UART driver” (ch341ser.exe) found on <https://www.wemos.cc/downloads>
2. Install by following this: <https://www.wemos.cc/tutorial/get-started-arduino.html> , i.e.:

* Cd J:\Programmer\Arduino\hardware (install folder of Arduino)
* **git clone** [https:**//**github.com**/**esp8266**/**Arduino.git esp8266](https://github.com/esp8266/Arduino.git%20esp8266)
* **cd** esp8266**/**tools
* python get.py

1. Restart Arduino IDE
2. Compile a sketch and upload (this takes quite a while, easily 30 minutes. Have patience)
3. Change the default sketchbook location to the place where you want your source code:
   1. File => Setting => “I:\DEV\Arduino”
4. Install the Visual Micro plug-in (“Visual.Micro.Arduino.Studio.vsix”) on Visual Studio to integrate the Arduino IDE into VS. (I don’t know if it works to install Visual Micro before installing the ESP-libraries. I had some problems, so I had to de-install and start from scratch).
5. Install the Arduino ESP8266 board: <https://github.com/esp8266/Arduino#installing-with-boards-manager>:

* Start Arduino and open Preferences window.
* Enter http://arduino.esp8266.com/stable/package\_esp8266com\_index.json into Additional Board Manager URLs field. You can add multiple URLs, separating them with commas.
* Open Boards Manager from Tools > Board menu and install esp8266 by ESP8266 Community platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).
* Restart Arduino IDE
* Rebuild the simple sketch
* (after this, the IDE woulnd’t start at all. I then delete the hardware/esp8266 folder completely (well – moved it). And then it worked (!?)

1. Attach board to USB and recompile and upload.

Other useful links:

An alternative installation method plus some info about the board: <http://www.instructables.com/id/Programming-the-WeMos-Using-Arduino-SoftwareIDE/?ALLSTEPS>.

# Barcode reader programming

## Test programs for development

### 29/11-16: sending messages to IOT hub from Arduino using Visual Studio:

/DEV/Arduino/src/CommandcenterVS

Gives warning “WARNING: Missing unmatched library header: time.h”. This is OK.

1/12-16: Retrieving a file via http (on windows):

* DEV/GIT/simplecurl

## Sending a file over wifi

Check:

<https://github.com/itead/ITEADLIB_Arduino_WeeESP8266/tree/master/examples>

<https://github.com/itead/ITEADLIB_Arduino_WeeESP8266>

<http://stackoverflow.com/questions/12251551/send-arduino-data-via-a-wi-fi-shield-to-a-specific-ip-address-on-the-lan>

Additional shopping:

* CD4050 IC (<https://www.elextra.dk/main.aspx?page=article&artno=H15876> , 12,00 DKK)

# Visual Studio setup for Arduino

Use the free plugin from <http://www.visualmicro.com> . It includes a (non free) debugger.

NOTE : Do not use the Windows **Store** version (i.e. the Windows App) of the Arduino IDE.   
Instead use the IDE for download for Windows. The Arduino IDE should be installed first.  
I also installed the additional WeMo boards first on the Arduino IDE.

Install following <http://www.visualmicro.com/page/User-Guide.aspx?doc=Getting-started.html>   
File : Visual.Micro.Arduino.Studio.vsix

### Running and debugging

#### The serial output setup



Restart by either pressing reset hardware button or the green « Connect » button on the IDE serial output window..

### Install the right VS extensions :

<https://msdn.microsoft.com/en-us/magazine/mt694088.aspx>

* Visual C++ for Linux Development
* Visual C++ for IoT Development (NOT INSTALLED. Requires android setup. Is that really necessary?)
* Azure IoT Hub Connected Service
* Java extension (for the com-reader)

### Important note

.ino files must ONLY contain alphanumeric characters. No underscores, no spaces, no nothing. You will get errors like this:

* Compiling 'Multi\_Starter2\_jmd\_ports2-9\_VS' for 'Arduino Uno'  
  core.a(main.cpp.o)\*:In function `main'

<http://www.visualmicro.com/forums/YaBB.pl?num=1416749264>

### In case of errors

#### Class unknown class VisualMicroDebug

There are breakpoints somewhere inthe code. Disable them and recompile. (<http://www.visualmicro.com/forums/YaBB.pl?action=print;num=1416846955> )

#### esptool

For the ESP8266 (« D1 ») board to work in case of a problem with « esptool » or   
 *«exec: "J:\\Programmer\\Arduino\\hardware\\esp8266com\\esp8266/tools/xtensa-lx106-elf/bin/xtensa-lx106-elf-g++": file does not exist  
Error compiling for board WeMos D1(Retired).* » 🡺 close Visual Studio and follow this CAREFULLY:   
<http://www.instructables.com/id/Portable-installation-guide-of-Arduino-IDE-v165-fo/?ALLSTEPS>

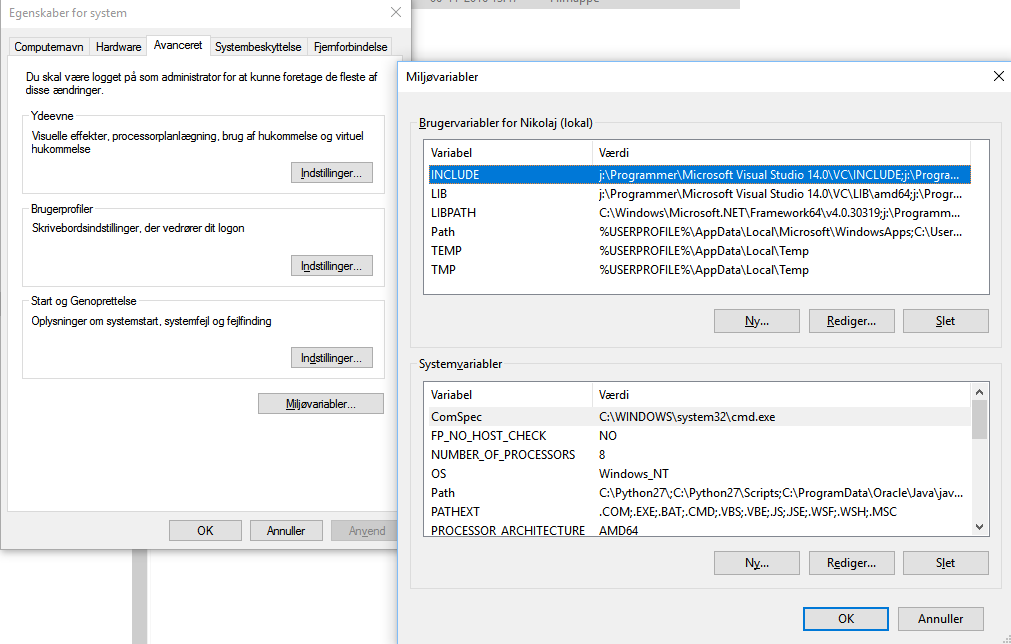
Hereafter, restart Visual Studio.

And then: pip install esptool

### Path variables

In my case the following environment variables did not exist, so I created them. Check that the paths exists.

* set INCLUDE=j:\Programmer\Microsoft Visual Studio 14.0\VC\INCLUDE;j:\Programmer\Microsoft Visual Studio 14.0\VC\ATLMFC\INCLUDE;C:\Program Files (x86)\Windows Kits\10\Include\10.0.10240.0\ucrt;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\include\um;
* set LIB=j:\Programmer\Microsoft Visual Studio 14.0\VC\LIB\amd64;j:\Programmer\Microsoft Visual Studio 14.0\VC\ATLMFC\LIB\amd64;C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6\lib\um\x64;
* set LIBPATH=C:\Windows\Microsoft.NET\Framework64\v4.0.30319;j:\Programmer\Microsoft Visual Studio 14.0\VC\LIB\amd64;j:\Programmer\Microsoft Visual Studio 14.0\VC\ATLMFC\LIB\amd64;C:\Program Files (x86)\Windows Kits\10\UnionMetadata;C:\Program Files (x86)\Windows Kits\10\References;C:\Program Files (x86)\Windows Kits\10\References\Windows.Foundation.UniversalApiContract\1.0.0.0;C:\Program Files (x86)\Windows Kits\10\References\Windows.Foundation.FoundationContract\1.0.0.0;C:\Program Files (x86)\Windows Kits\10\References\indows.Networking.Connectivity.WwanContract\1.0.0.0;
* cmd.exe /k ""C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\vcvarsall.bat"" amd64
* cmd.exe /k ""C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\vcvarsall.bat"" x64
* cmd.exe /k ""C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\vcvarsall.bat"" x86



### Install curl library:

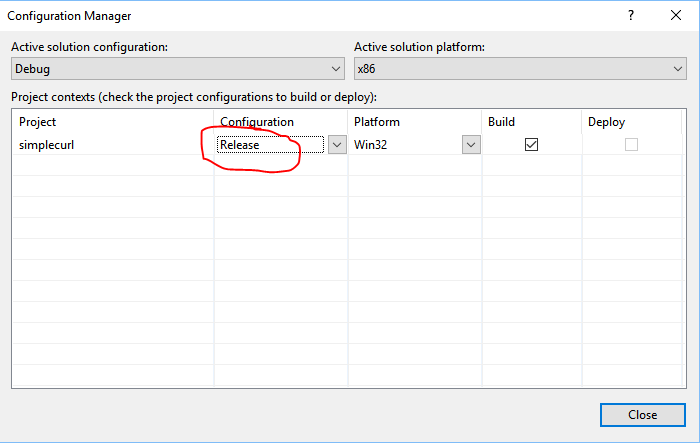
Follow <http://www.cplusplus.com/forum/windows/172335/> CAREFULLY.   
I did as follows:

* Make sure all environment variables are set as above (“Path variables”)
* Install cmake for windows from <https://cmake.org/download/> (not sure if this is actually needed afterall)
* Install Visual Studio **Command prompt** if not already existing: <http://stackoverflow.com/questions/21476588/where-is-developer-command-prompt-for-vs2013> (exchanging 2013 with 2015)

Follow I:\DEV\GIT\GitRepo\curl\winbuild\BUILD.WINDOWS.txt :

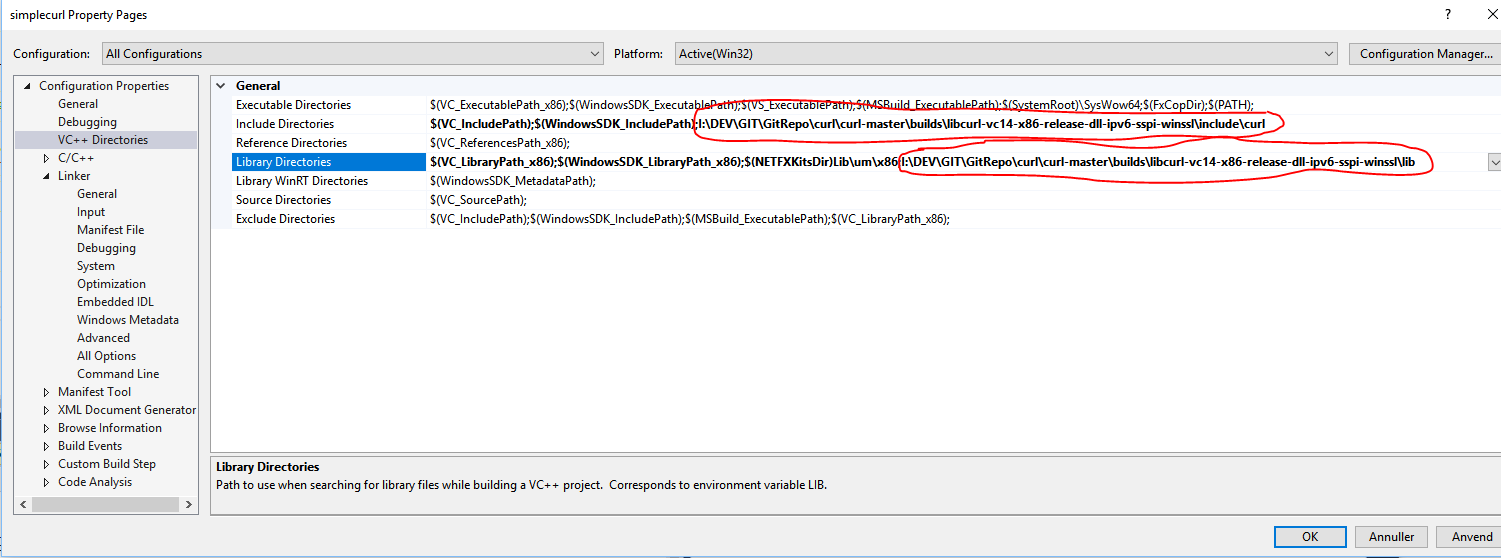
* From VS command prompt (NOT regular command prompt):
* Cd I:\DEV\GIT\GitRepo\curl\winbuild
* nmake /f Makefile.vc mode=dll VC=14
* cd I:\DEV\GIT\GitRepo\curl\src
* cmake -G "Visual Studio 14 2015 Win64"

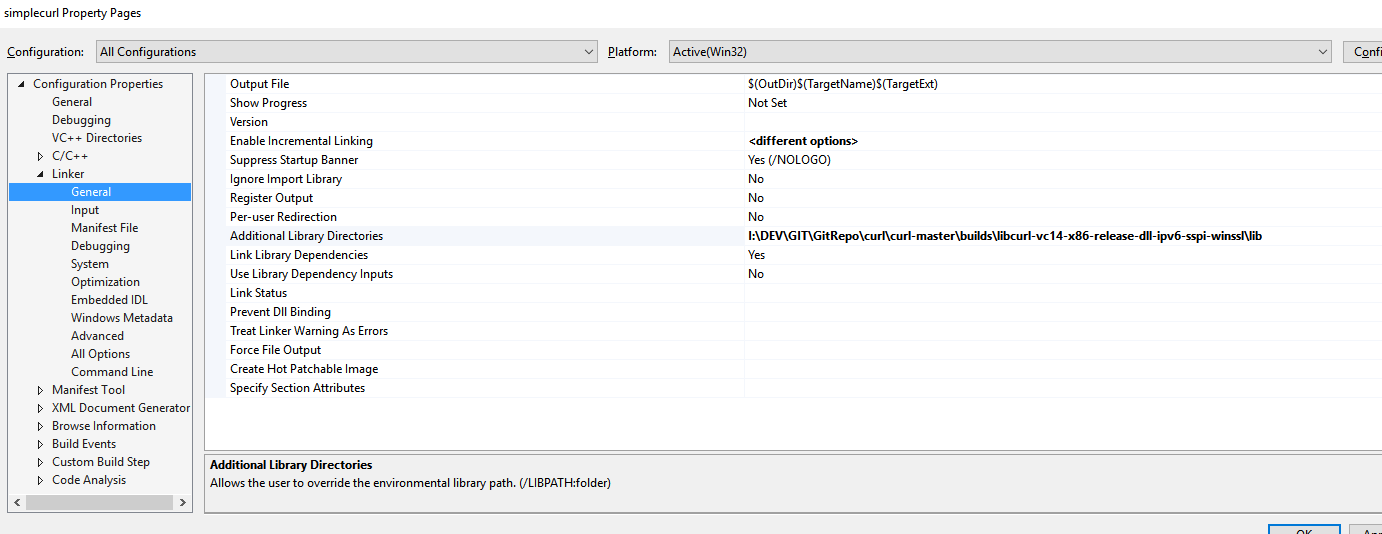
Set project to release mode (not debug) in Build => Confuragtion manager



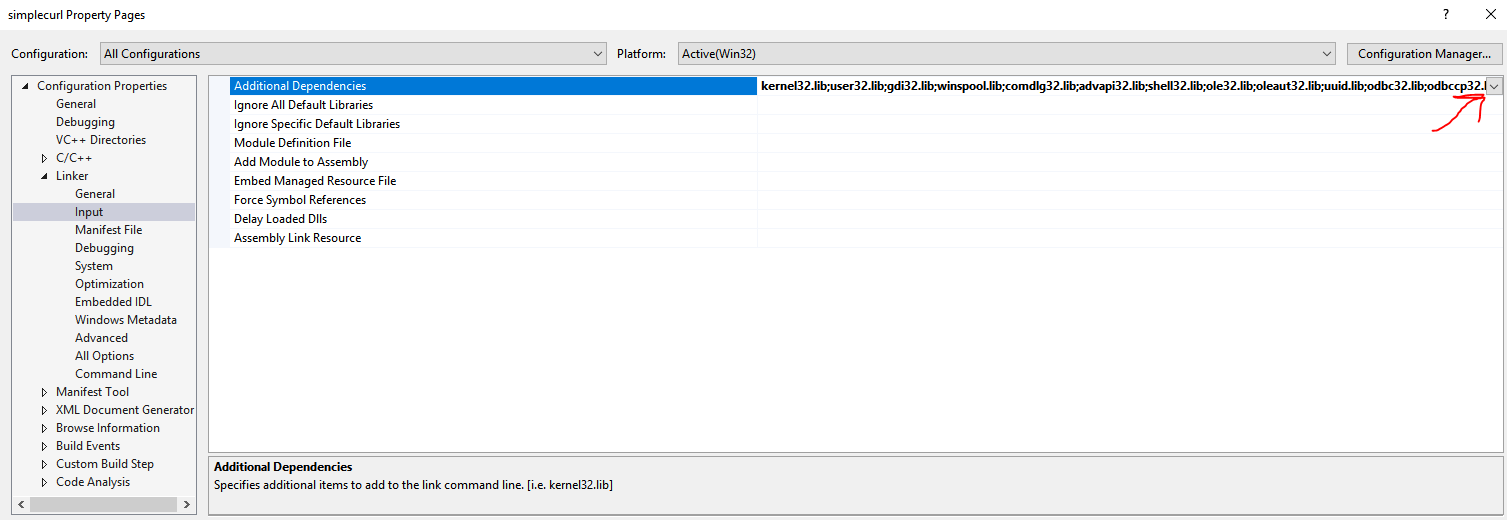
Add the paths to eachVS project usding curl:

* VC++ Directories => Include directories:
* I:\DEV\GIT\GitRepo\curl\curl-master\builds\libcurl-vc14-x86-release-dll-ipv6-sspi-winssl\include\curl
* VC++ Direcotries => Library directories:
* I:\DEV\GIT\GitRepo\curl\curl-master\builds\libcurl-vc14-x86-release-dll-ipv6-sspi-winssl\lib
* Linker => Input => additional dependencies:
* libcurl.lib
* Copy the file I:\DEV\GIT\GitRepo\curl\curl-master\builds\libcurl-vc14-x86-release-dll-ipv6-sspi-winssl\bin\libcurl.dll to the project source code folder (I:\DEV\GIT\simplecurl\simplecurl).





Add libcurl.lib at the end of this line:



Copy the file I:\DEV\GIT\GitRepo\curl\curl-master\builds\libcurl-vc14-x86-release-dll-ipv6-sspi-winssl\bin\libcurl.dll to the project source code folder (I:\DEV\GIT\simplecurl\simplecurl).

The VS project simplecurl gets a file from an external site, downloads and stores it locally.

# Java (not used)

NOT used because the ReadSerial program didn’t work. Cause: the latest JDK (v8) does not support the javax.comm library.

* Remove all Java and Java Dev Kit versions and updates. Start from a clean sheet.
* Install latest Java SDK 32-bit version (there’s no working javax.comm for 64-bit) <http://stackoverflow.com/questions/3959743/javax-comm-api-on-64-bit-windows>
* Install Java Visual Studio plugin
* Install javax.comm API’s
* Note: Not supported on any windows platform anymore
* install
* : <https://circuitnegma.wordpress.com/2007/02/07/how-to-install-the-java-communications-api-in-a-windows-environment/>
* Copy 'win32com.dll' to JDK\_HOME\jre\bin. (C:\Program Files (x86)\Java\jdk1.8.0\_121\jre\bin)
* Copy 'javax.comm.properties'to to JDK\_HOME\jre\lib.
* Copy 'comm.jar'to to JDK\_HOME\jre\lib