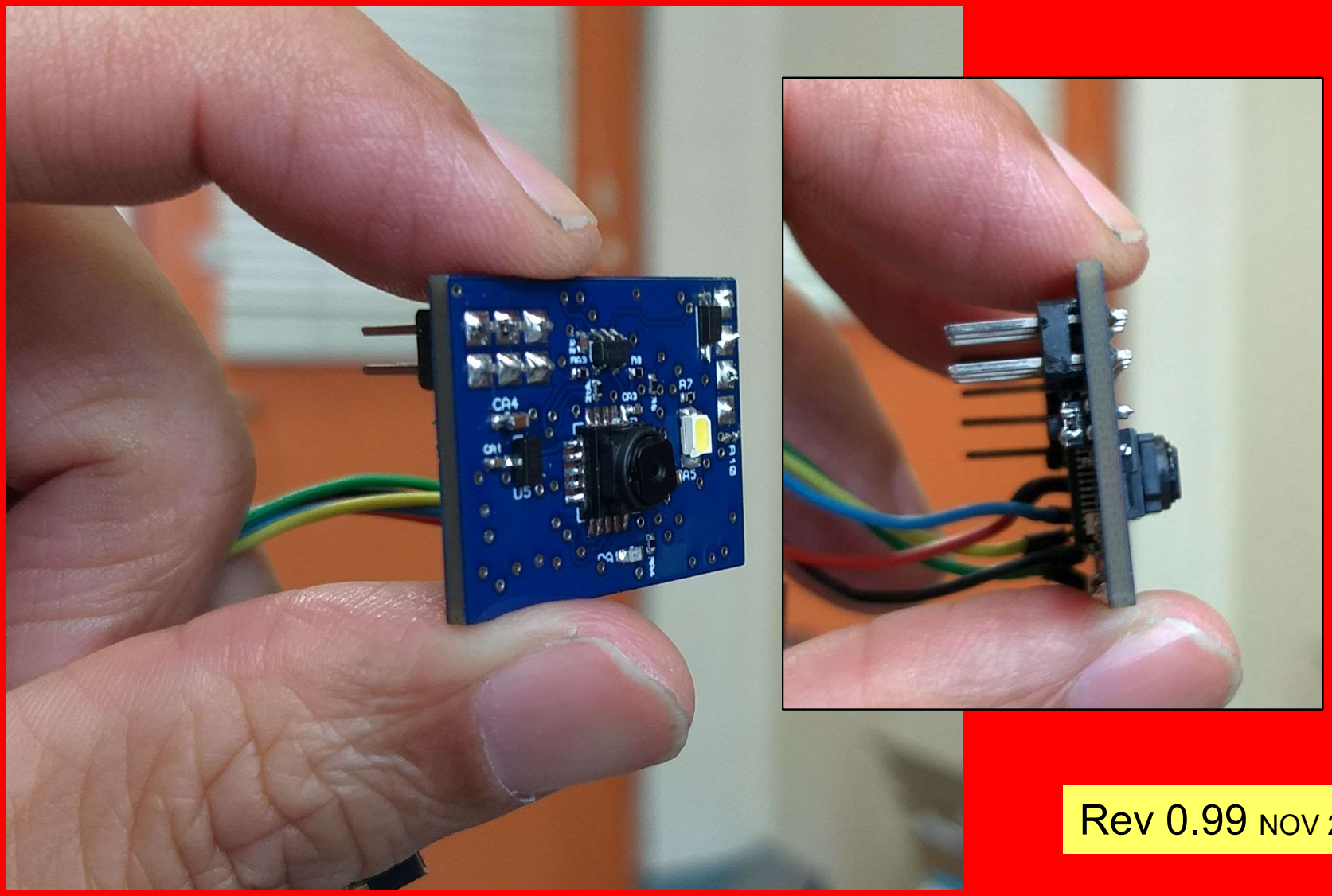
MVM V1.5C Quick Start Guide



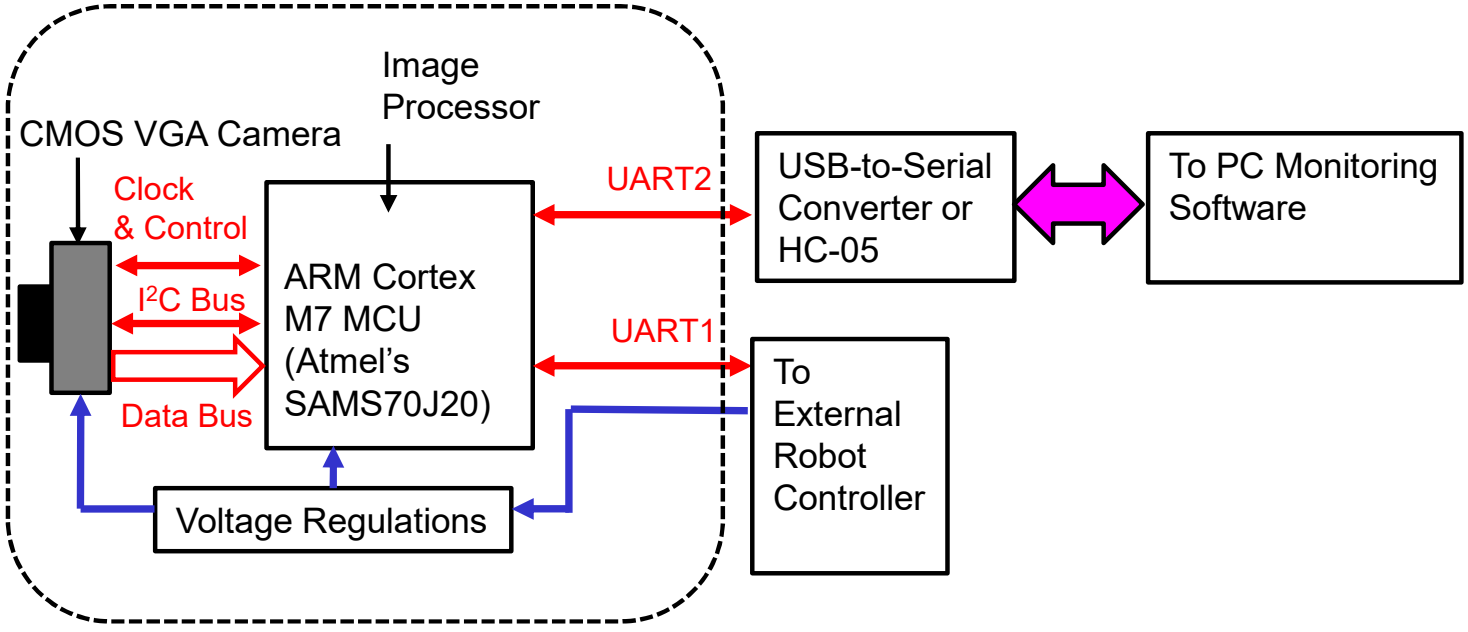
Rev 0.99 NOV 2021

What Is It?

- An open source, easy to use low resolution CMOS camera with on-board real-time image processing.
- Requires 5V, 150 mA power source, and interface through UART port.
- Support 160x120 pixels (QQVGA) and 320x240 pixels (QVGA) color image at 20 frames-per-second.
- Current image processing algorithm:
  - Edge detection via Sobel kernel.
  - Bright spot detection.
  - Obstacle detection using luminance contrast.
  - Color detection.
  - Convolutional Neural-Network.
- Work-in-progress:
  - Line following.
  - Optical flow.

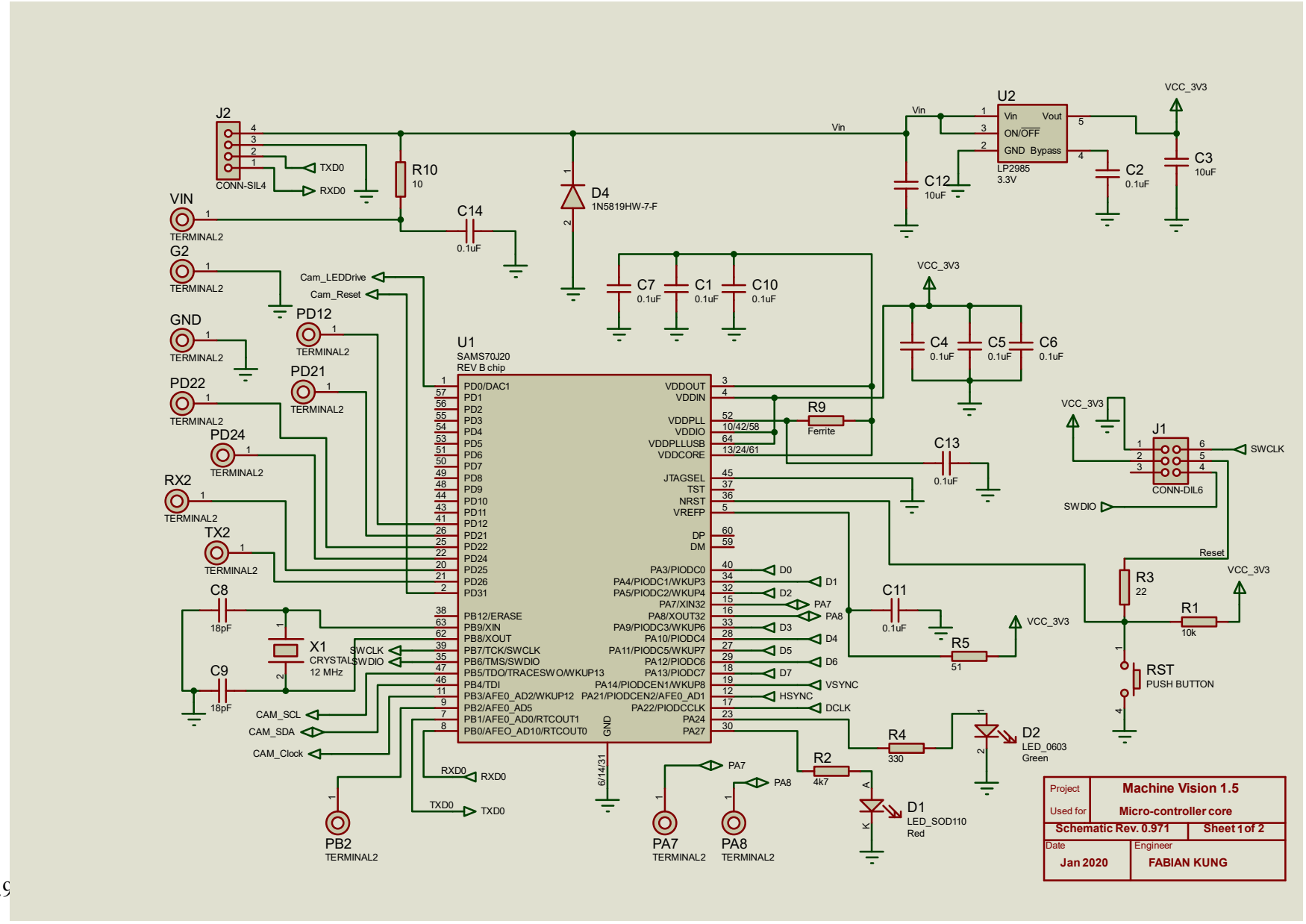
Block Diagram

WARNING: Because the processor runs on 3.3V, UART1 and UART2 works at 3.3V logic level. Applying higher voltage (for example 5V) to the terminals may damage the processor

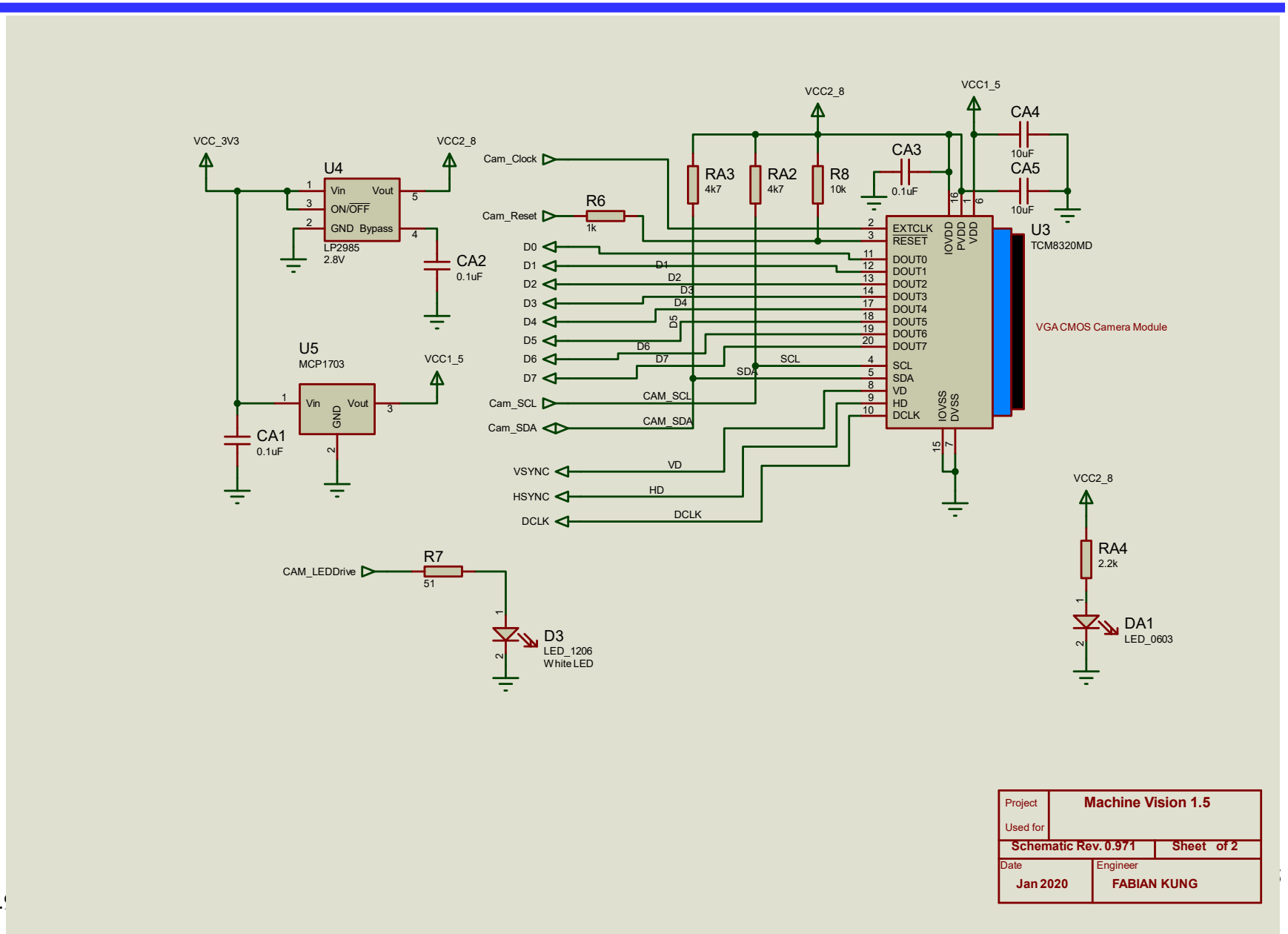


MVM V1.5C

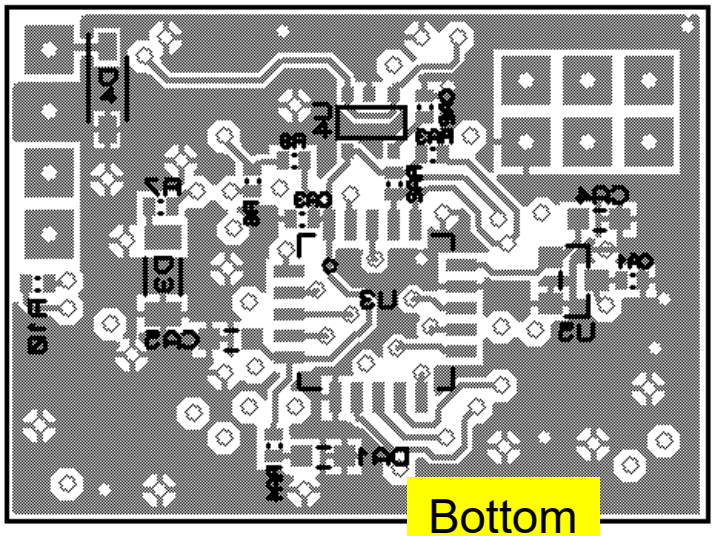
Schematic 1 – Micro-controller Core



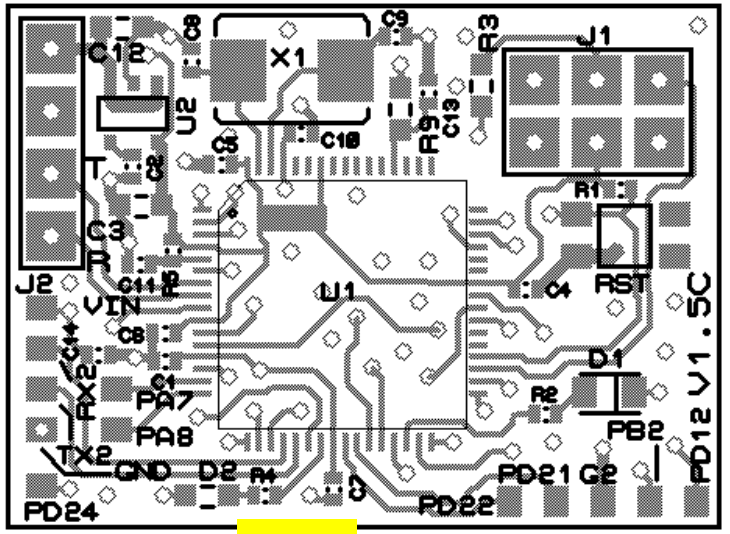
Schematic 2 – Camera Sub-Circuit



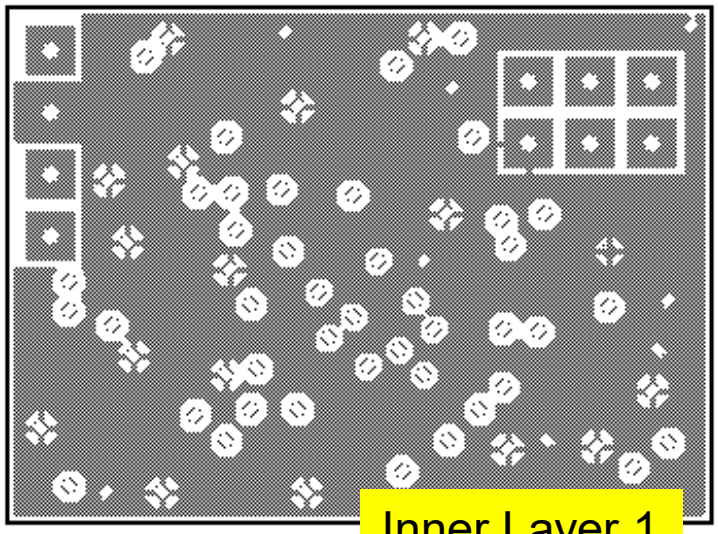
PCB Layers



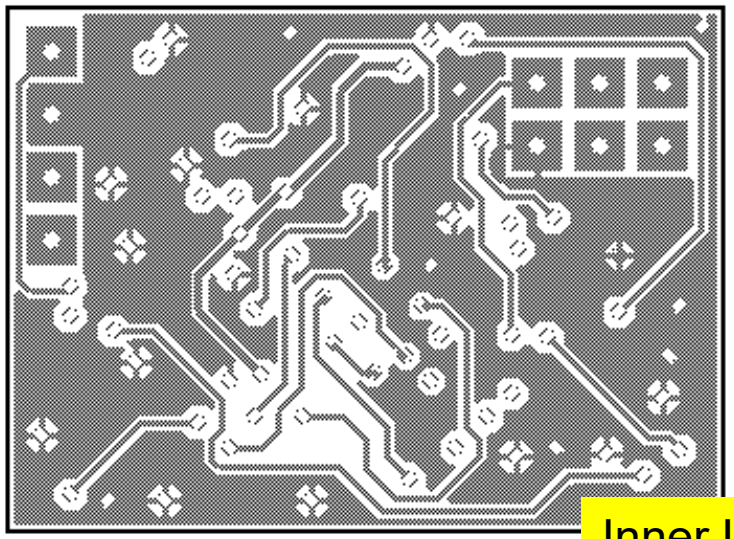
Bottom



Top

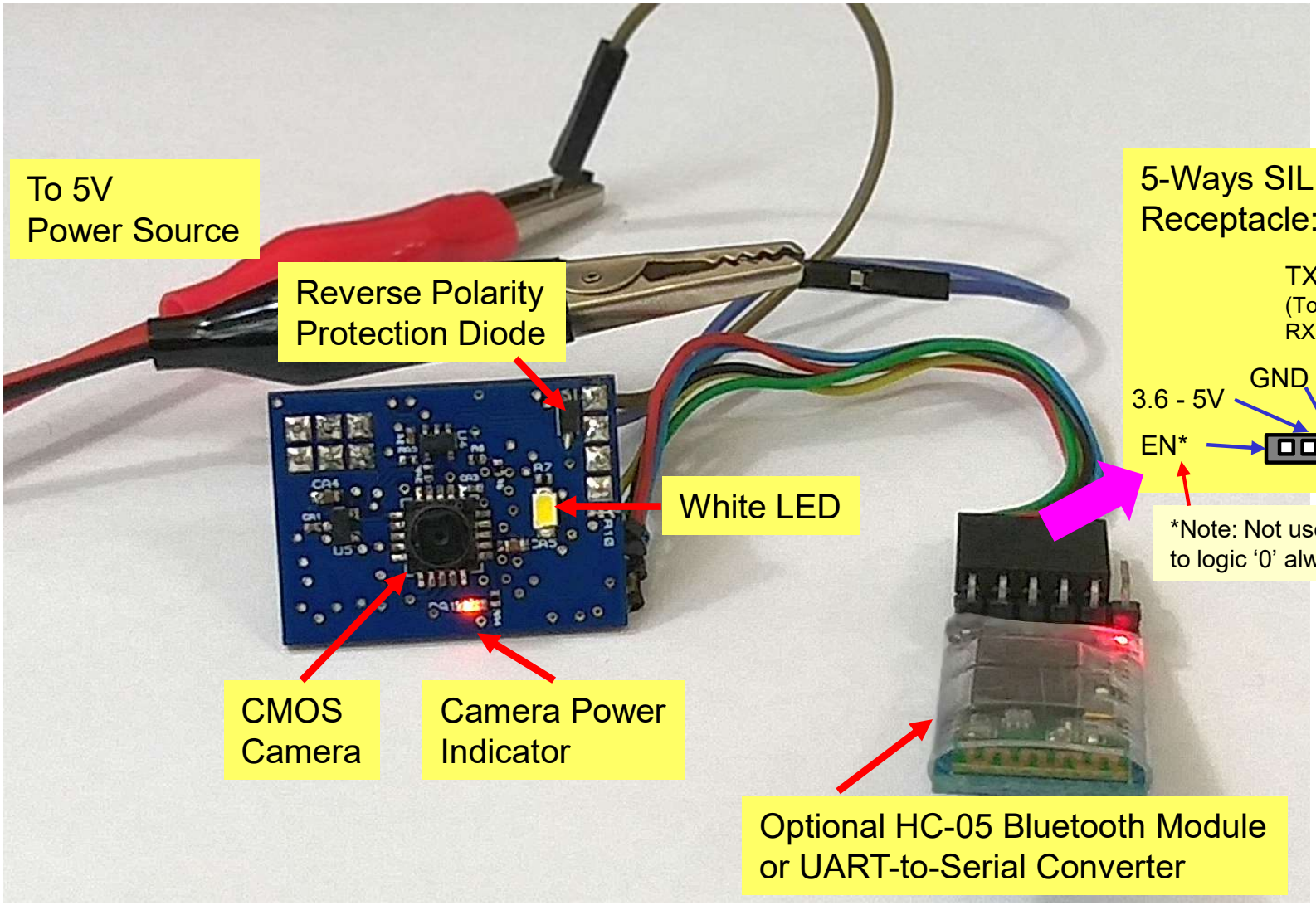


Inner Layer 1

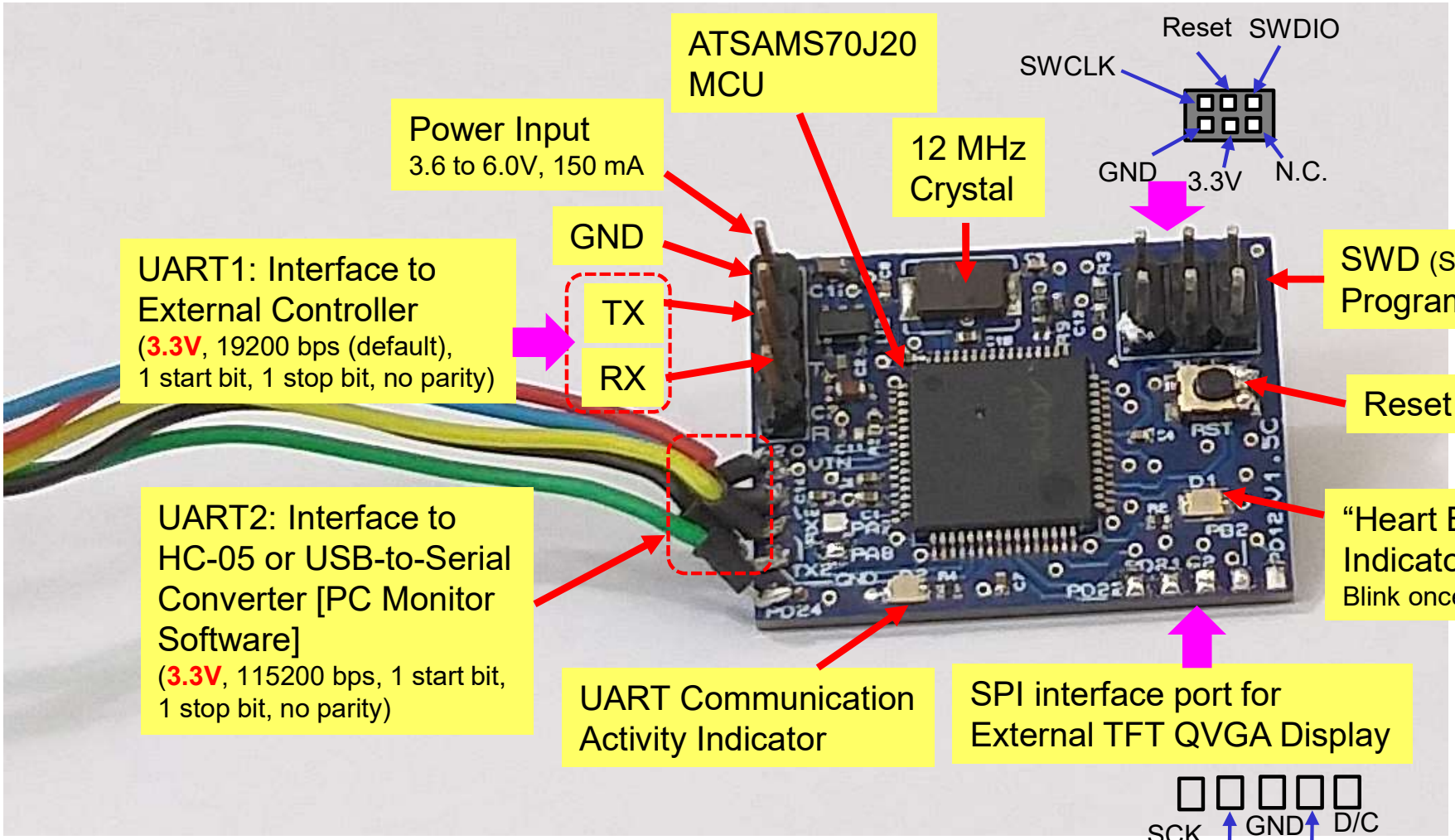


Inner Layer 2

Rear View (MVM V1.5C)



Front View (MVM V1.5C)



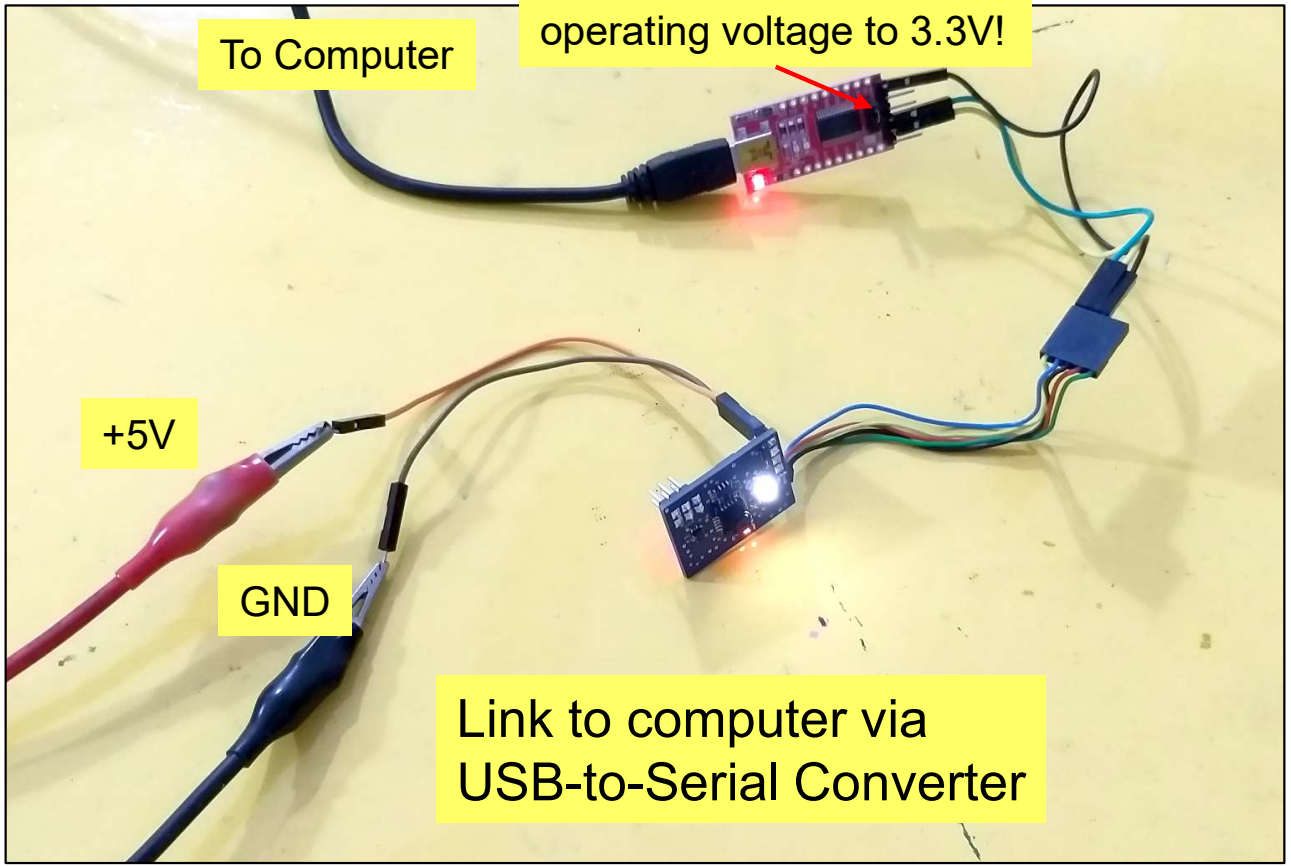
Files

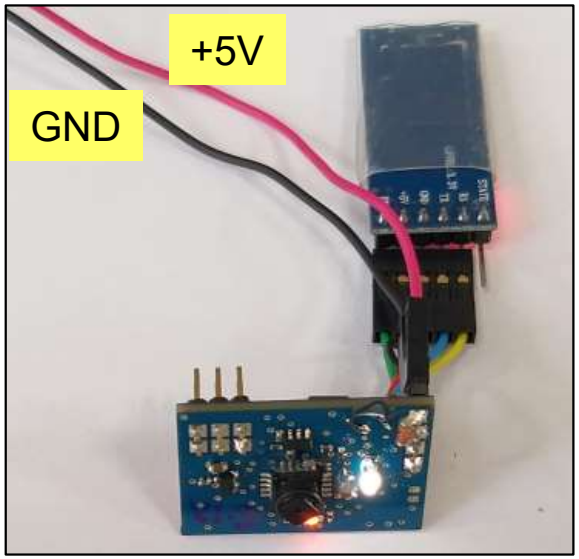
- All relevant files can be obtained from https://github.com/fabiankung/MVM_V1_5C
- Firmware is build using **Atmel Studio 7**.
- PC software is build using **Visual Studio Community 2017** or later.

Observing the Camera Image via Machine Vision Monitor Software

Step 1 – Power Up the MVM

- Here we assume the MVM is connected to HC-05 Bluetooth wireless module or a USB-to-Serial Converter, as shown in the various implementation examples below. Power up the module.



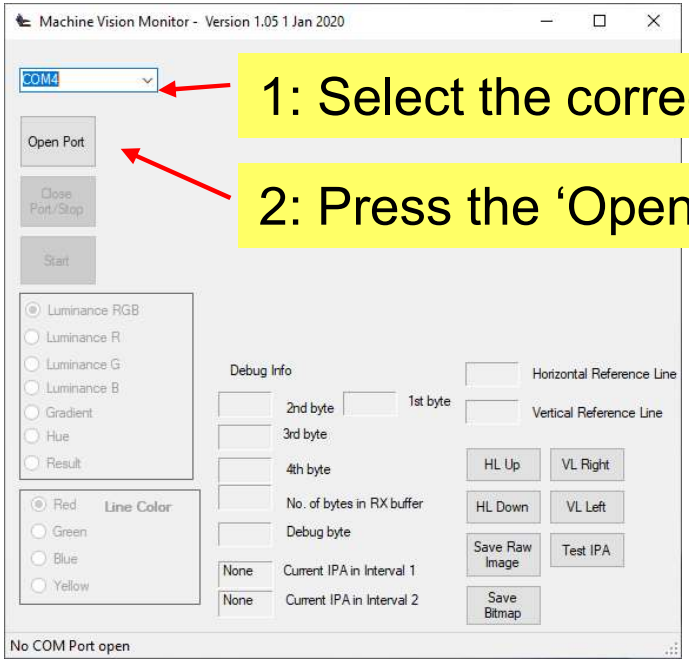


Link to computer via HC-05 Bluetooth Module

Step 2 – Pair Computer to HC-05

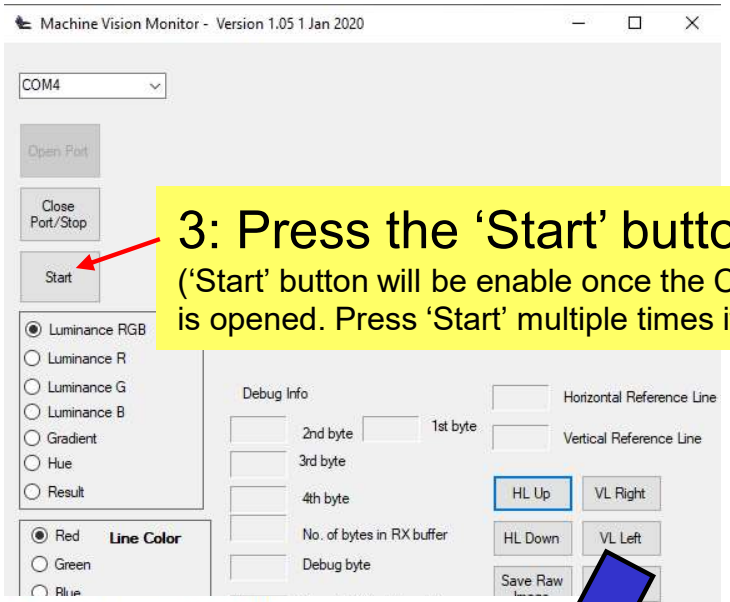
- If need to, pair the computer to HC-05 first.
- Then check virtual COM port number on the computer (for instance by going to the Device Manager in Windows).

Step 3 – Run the Machine Vision PC Monitor Software (MV_Monitor.exe)

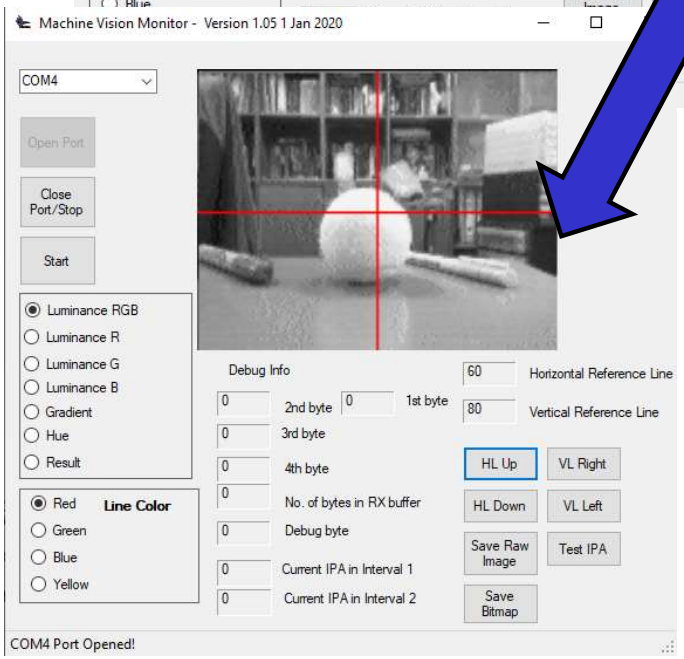


1: Select the correct COM port

2: Press the 'Open Port' button



3: Press the 'Start' button
('Start' button will be enable once the COM port is opened. Press 'Start' multiple times if necessary)



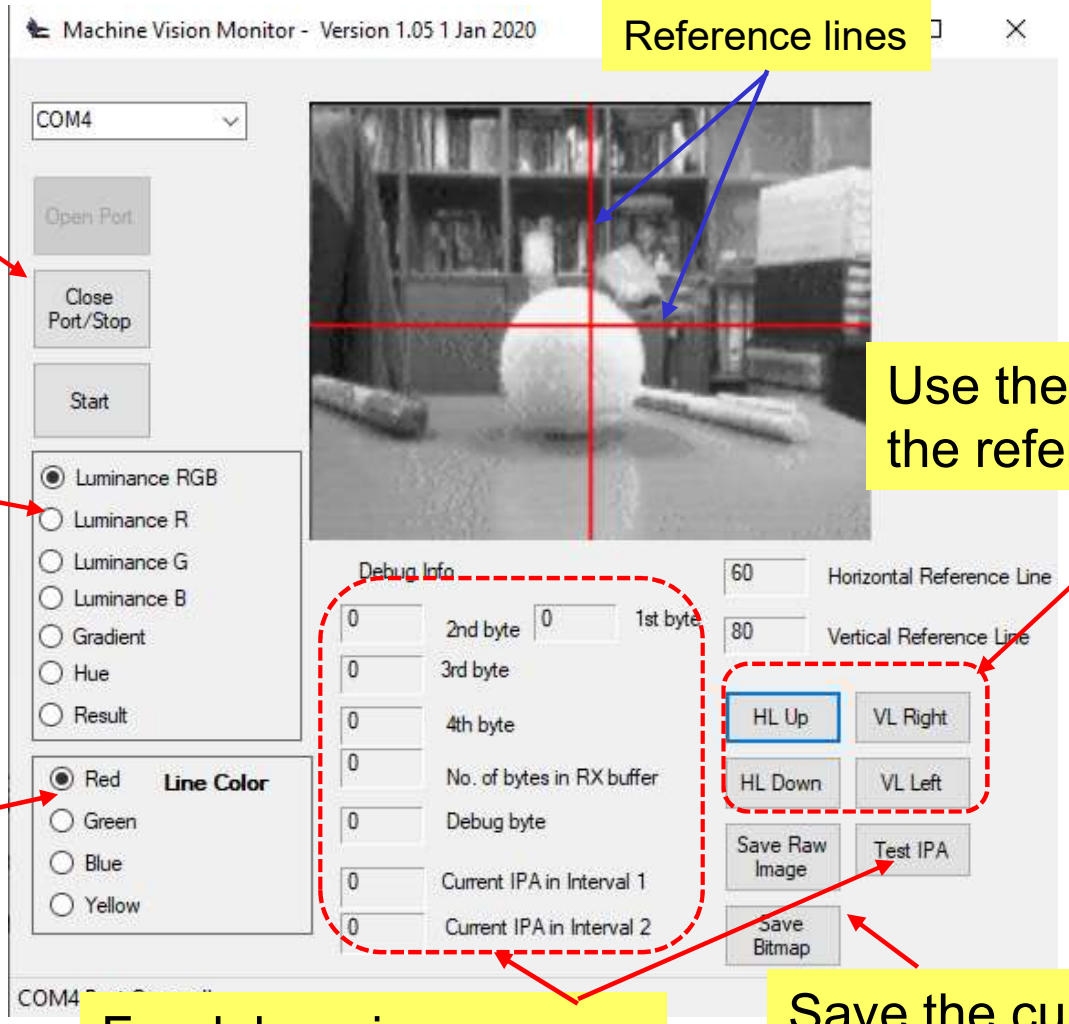
4: Now the MVM will transmit the image data line-by-line to the PC monitor software

Other Information [1 of 2]

Always close the COM port before shutting down the software

Select between various display options. All these are processed in the MVM. (See next slide)

Change reference line color.

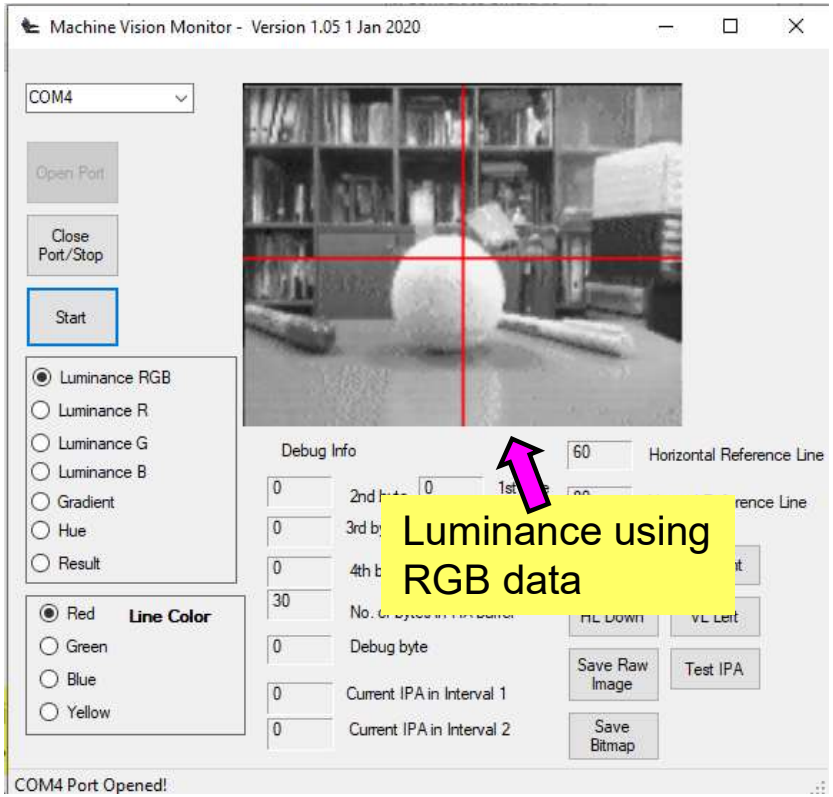


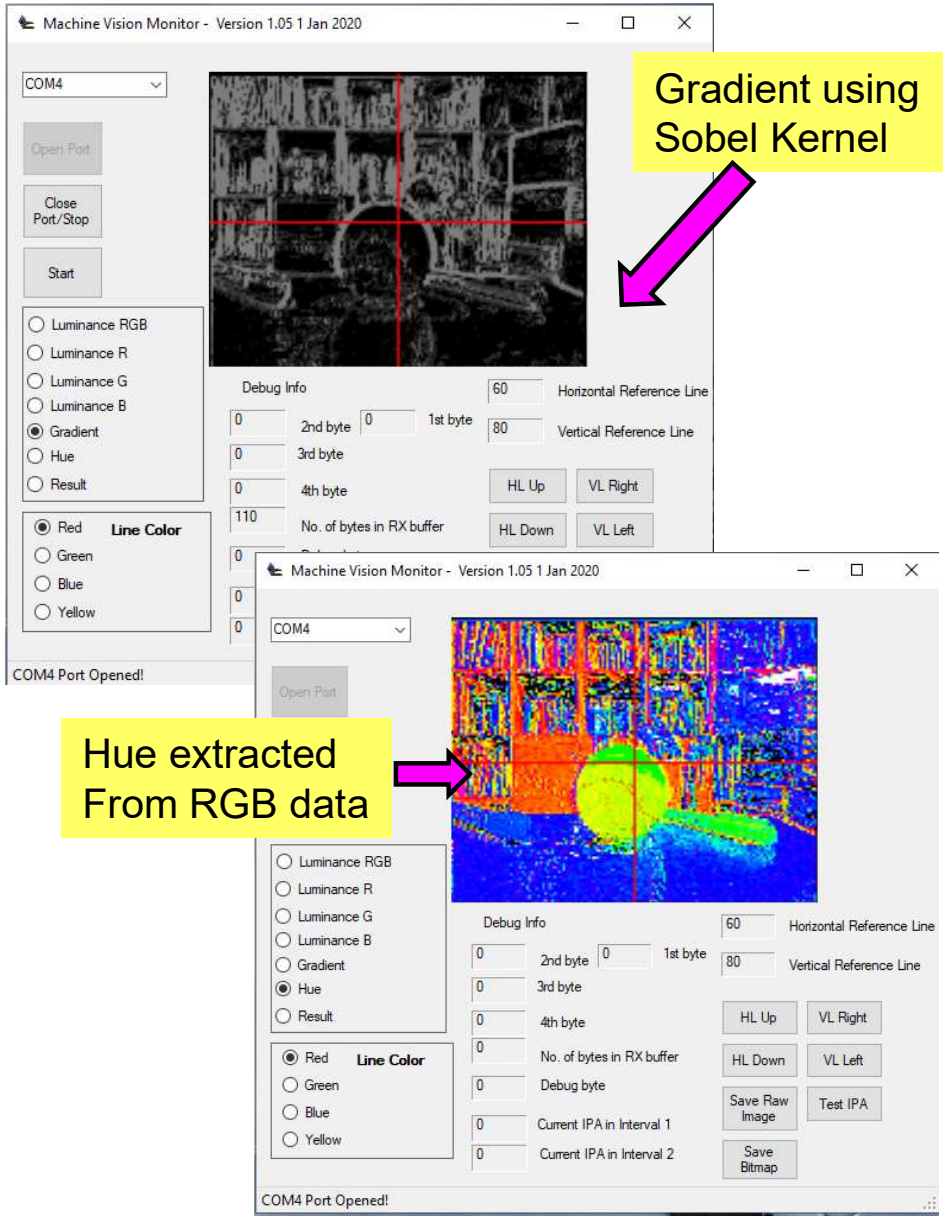
Use these to move the reference lines.

For debugging purpose (see source codes)

Save the current display in Bitmap format on hard disk

Other Information [2 of 2]





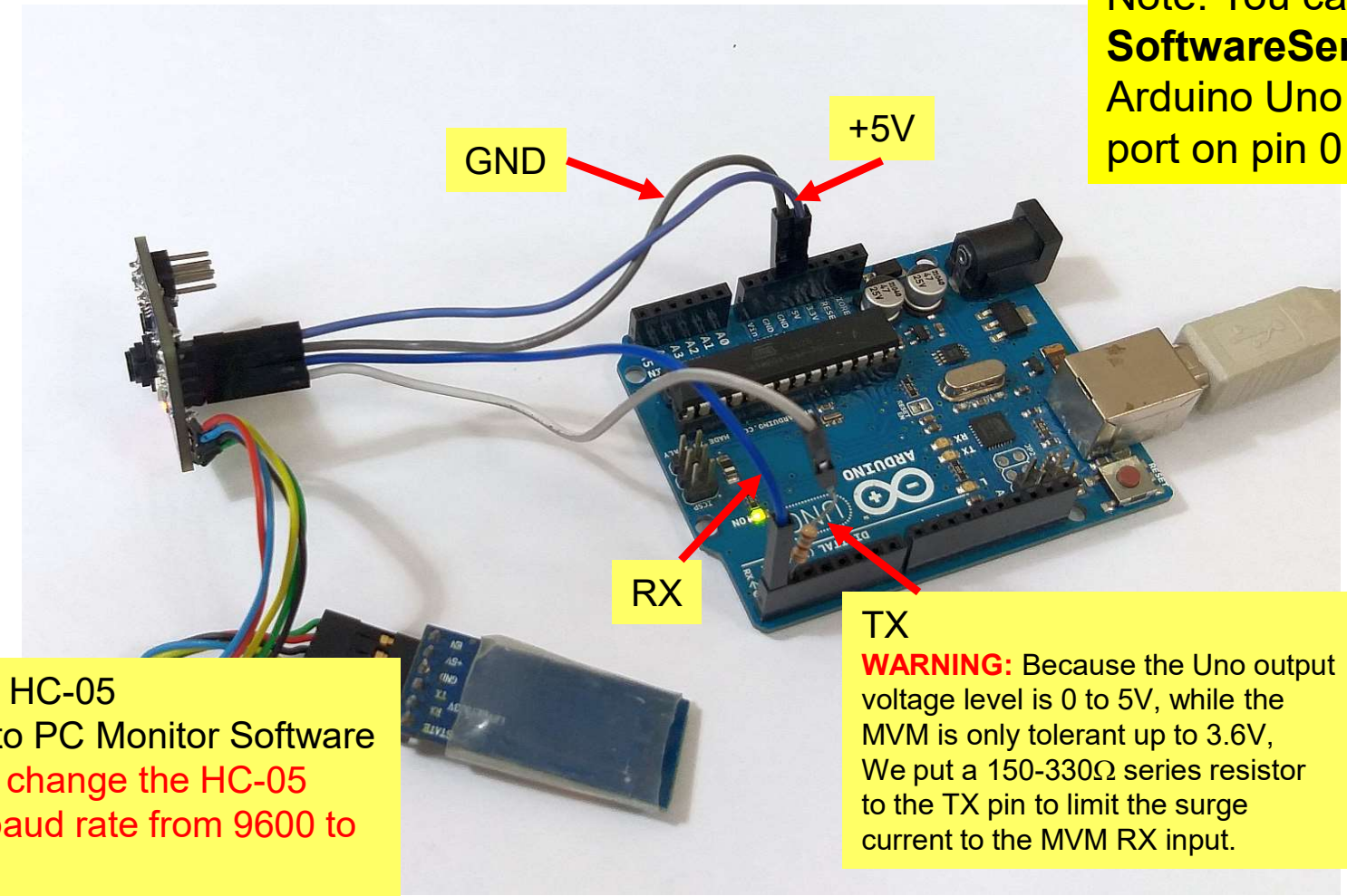
Version 0.982 Feb 2021

Connection to External Controller for Robotic Projects

Connection to External Controllers

- Here we use an Arduino Uno to demonstrate the connection.

Note: You can also use **SoftwareSerial** instead of Arduino Uno hardware serial port on pin 0 and 1.



Optional HC-05 Module to PC Monitor Software (need to change the HC-05 default baud rate from 9600 to 115200)

WARNING: Because the Uno output voltage level is 0 to 5V, while the MVM is only tolerant up to 3.6V, We put a 150-330Ω series resistor to the TX pin to limit the surge current to the MVM RX input.

UART1 Communication Protocol

Image Processing Algorithm (IPA)	To Activate	MVM Output
Search for brightest spot in a scene. Image resolution = 160x120	Send hex values to MVM: 0x10 to search for brightest spot	4 bytes: Byte 1 = 1 (Algorithm ID) Byte 2 = Maximum luminance value (1 to 127). Byte 3 = x coordinate of region Byte 4 = y coordinate of region
Obstacle detection on lower half of the image. Image resolution = 160x120	Send hex value to MVM: 0x20	4 bytes: Byte 1 = 2 Byte 2 = $0b00000b_2b_1b_0$ Byte 3 = $0b00000b_2b_1b_0$ Byte 4 = $0b00000b_2b_1b_0$
Color object detection. Image resolution = 160x120	Send hex values to MVM: 0x30 for yellow-green object 0x31 for red object 0x32 for green object 0x33 for blue object	4 bytes: Byte 1 = 3 Byte 2 = Number of pixels matched Byte 3 = x coordinate of region Byte 4 = y coordinate of region

255 will be send for (x,y) if region with matching color is not found

Example 1 – Activate Search for Brightest Spot Algorithm

- Assume the MVM is connected to an Arduino Uno. The left panel shows a simple Arduino Sketch to activate the image processing algorithm to search for brightest spot on both **Interval 1** and **2**, giving effective response time of 50 ms.

Note:
See Appendix for Another version of this code using SoftwareSerial

Example1 | Arduino 1.8.13

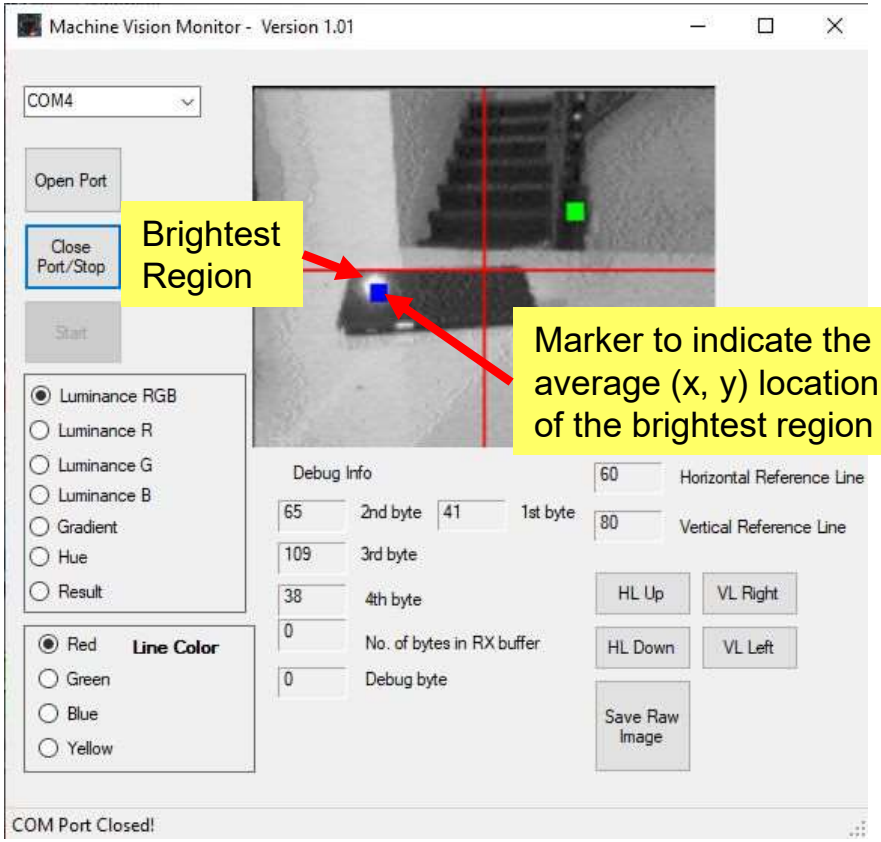
File Edit Sketch Tools Help

Example1

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(19200);
  delay(1000);          // 1000 ms delay for the module to initialize properly.
  Serial.write(0x10);   // Run IPA (image processing algorithm) 1 on interval 1.
  Serial.write(0x10);   // Run IPA 1 on interval 2.
}

void loop() {
  // put your main code here, to run repeatedly:

int nID;
int nLuminance;
int nX;
int nY;

  if (Serial.available() > 3) // Make sure at least 4 bytes in the received buffer.
  {
    nID = Serial.read();
    if (nID == 1)             // Make sure the data is from IPA 1.
    {
      nLuminance = Serial.read();
      nX = Serial.read();
      nY = Serial.read();

      //
      // Place user routines here
      //
    }
    else
    {
      Serial.flush();     // Flush received buffer.
    }
  }
}
```

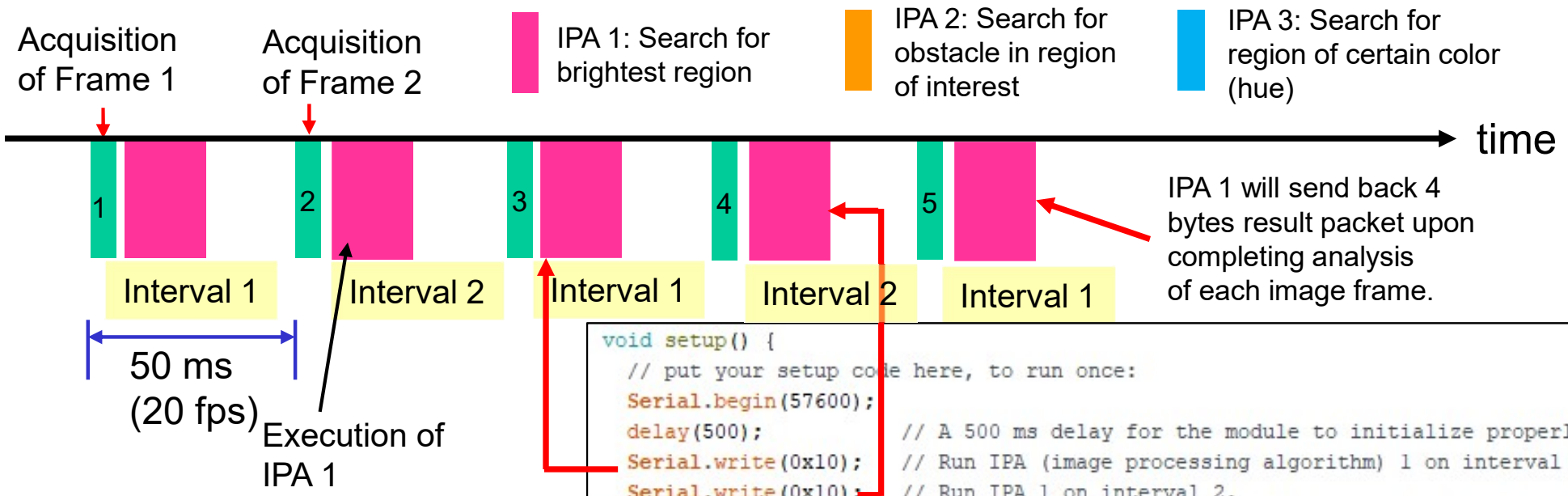
Done Saving.

1 Arduino Uno on COM9



Example 1 - More on 'Interval'

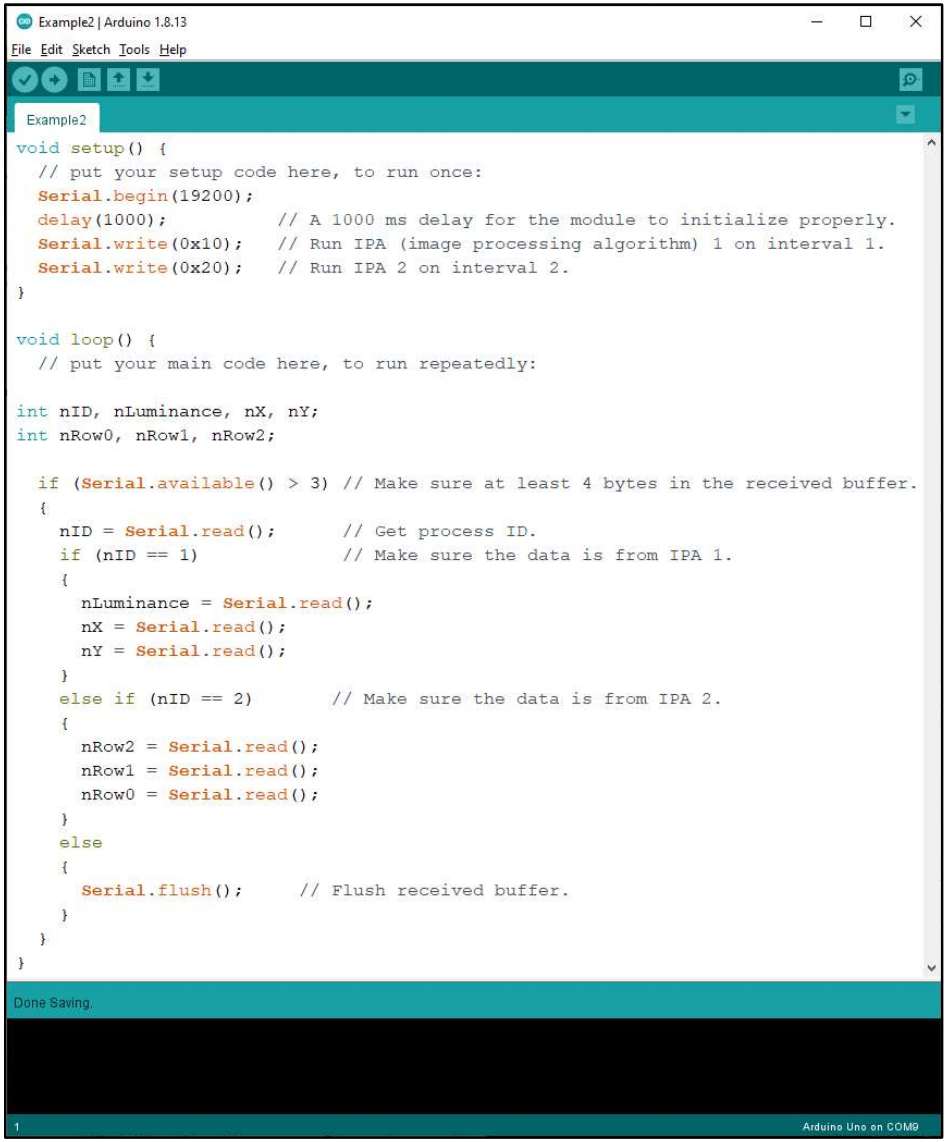
- The firmware of MVM V1.5C assigns odd image frames to *Interval 1* and even image frames to *Interval 2*.
- An image processing algorithm (IPA) can be attached to each interval as shown below and executed after acquisition of a new image frame.



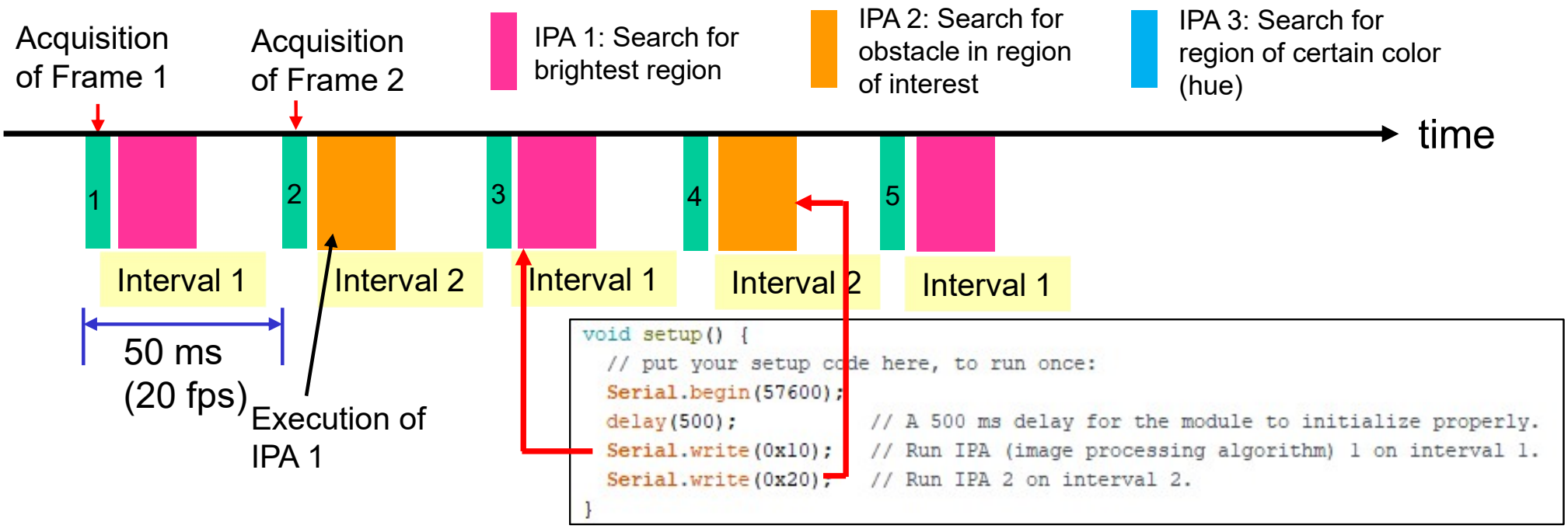
- The C code snippet attaches IPA 1 to both Interval 1 and Interval 2 of the execution flow, thus in this setting IPA 1 runs every 50 ms and any changes in scene is detected within 50 ms.

Example 2 - Activate Both Search for Brightest Region (IPA 1) and Obstacle (IPA 2) Algorithms

- In this example we attach IPA 1 to Interval 1 and IPA 2 to Interval 2.
- Thus a robot using the MVM V1.5C can be programmed to move towards a bright light source while at the same time avoid any obstacle on the floor.



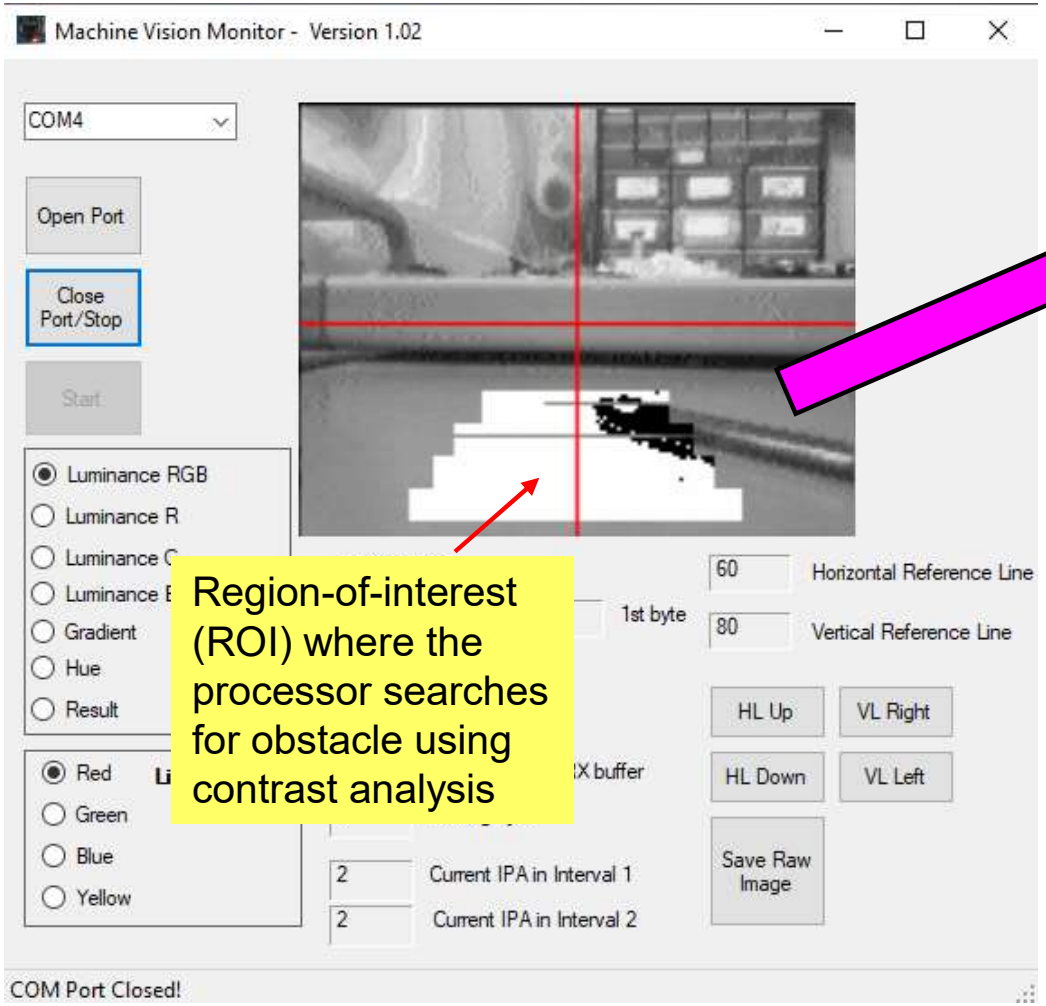
Example 2 – The Assignment of IPAs to Intervals



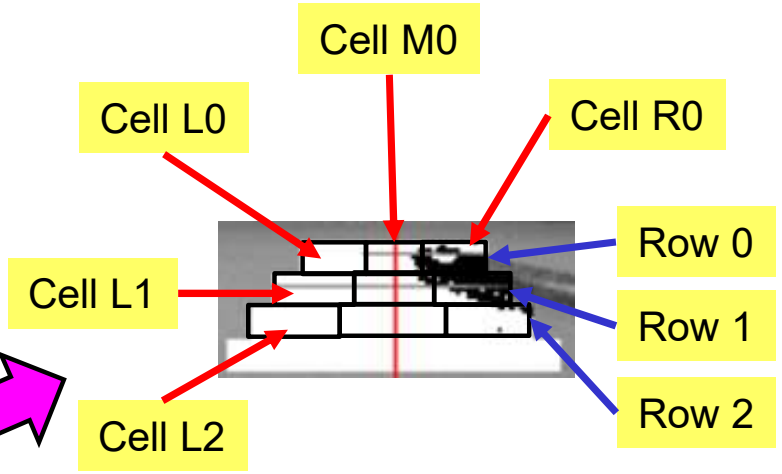
- Each IPA only executes every 100 ms, thus the response time now slows down to 100 ms, however the up side is we get to run two different algorithms simultaneously.

Interpreting the Results of IPA 2

- When IPA 2 is activated:



Region-of-interest (ROI) where the processor searches for obstacle using contrast analysis



Row 3 is not used for now.

Whenever the number of black pixels exceed a pre-determined threshold in a cell, an obstacle is deemed present. Thus, for this example, cells R0, M0 and R1 contains obstacle and the bits corresponding to these cells will be set to '1'. The following result packet will be transmitted via UART 1 from MVM V1.5C:

Byte 1 = 2 (ID)
Byte 2 = 0b00000**000** (Row 2)
Byte 3 = 0b00000**001** (Row 1)
Byte 4 = 0b00000**011** (Row 0)

L M R

Connection to TFT LCD Display

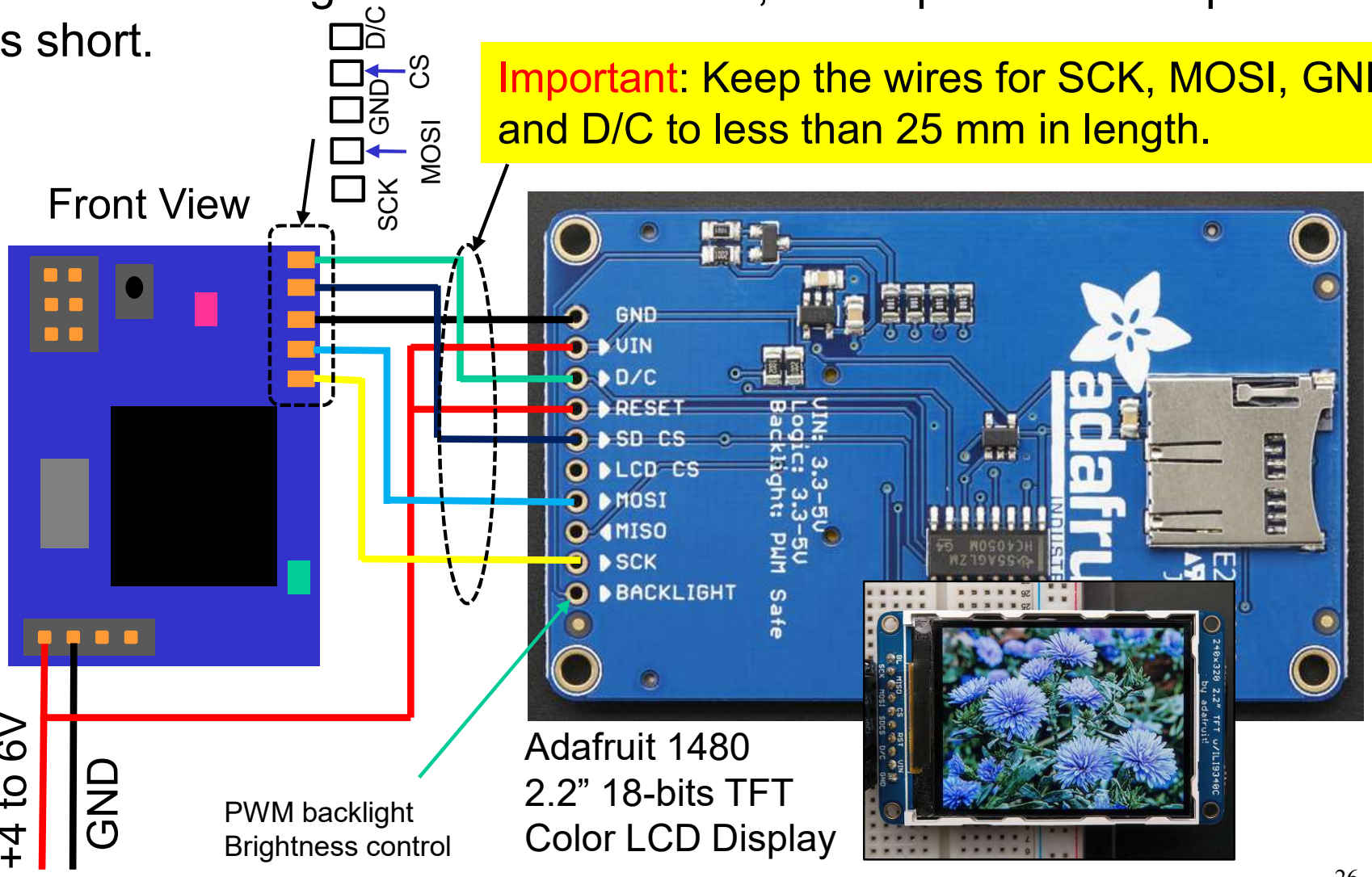
Support for TFT LCD Display

- Presently only ILI9341 TFT LCD controller in 4-wires serial communication mode is supported. The ILI9341 TFT LCD controller support a resolution of 320x240 pixels (i.e. QVGA).
- So LCD display using this controller can be connected to the MVM V1.5C.
- Examples of compatible TFT Displays are products from Adafruit such as product ID 2478, 1770, 1743 and 1480.
- Due to limitation on the micro-controller bandwidth, the firmware can only support either streaming of image to PC (via serial port) or streaming of image to LCD, not both. As default the firmware pre-programed into the micro-controller is the former. Thus if it is desire to interface the MVM V1.5C streaming output to TFT LCD display, another version of the firmware needs to be compiled and loaded into the micro-controller. See next section “Compiling and building your own firmware for MVM V1.5C”.

Connection Diagram 1

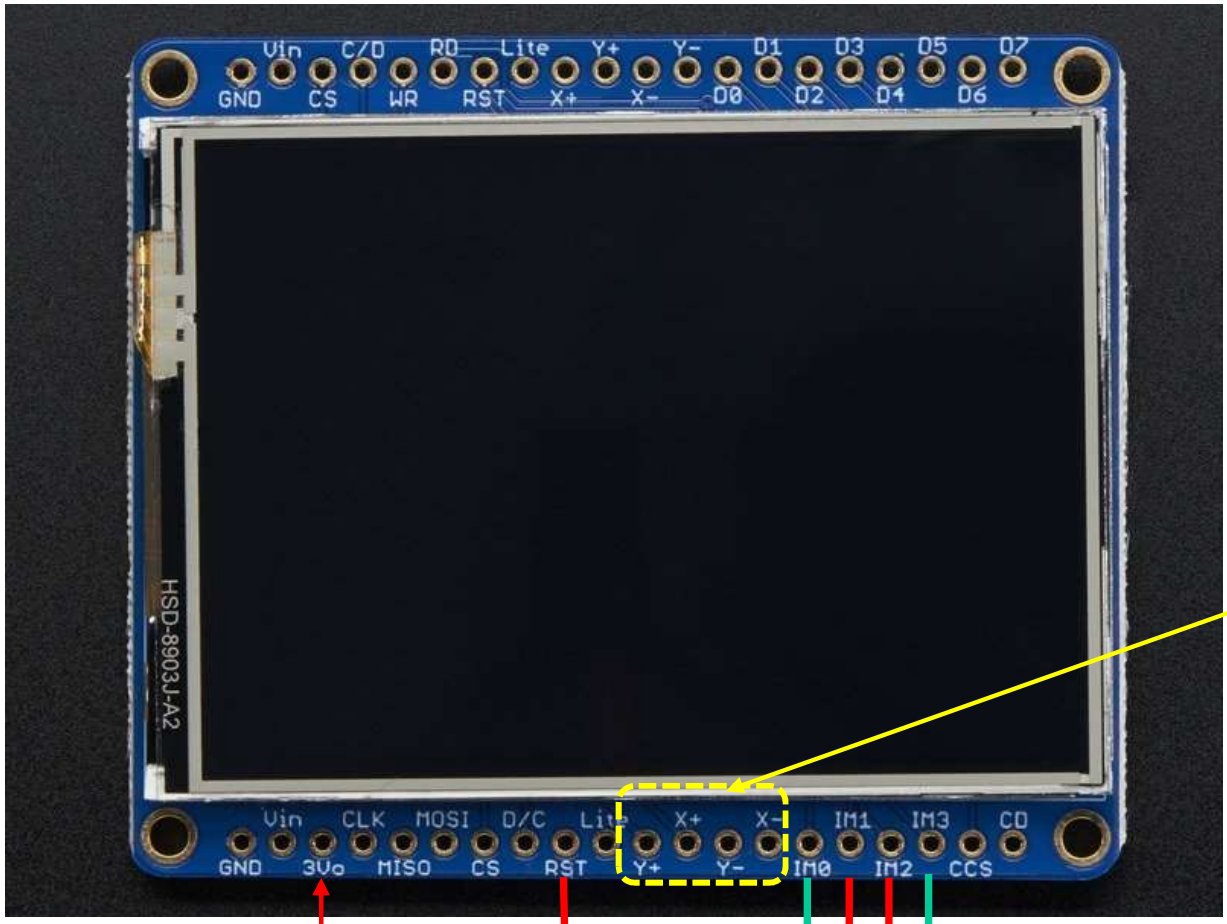
- As the SPI serial signals runs at 75 MHz, it is important to keep the wires short.

Important: Keep the wires for SCK, MOSI, GND, CS and D/C to less than 25 mm in length.



Connection Diagram 2

https://learn.adafruit.com/adafruit-2-4-color-tft-touchscreen-breakout/pinouts



Adafruit 2478
2.4” 18-bits TFT
Color LCD Display
With resistive touchscreen

Resistor touchpad output
(analog)

3.3V output from
on-board voltage regulator

3.3V

GND
3.3V
3.3V
GND

4-line SPI interface with D/C
(Note: I think can also set IM0 = 0V, and the rest 3.3V as per the instruction on the bottom side of the PCB)

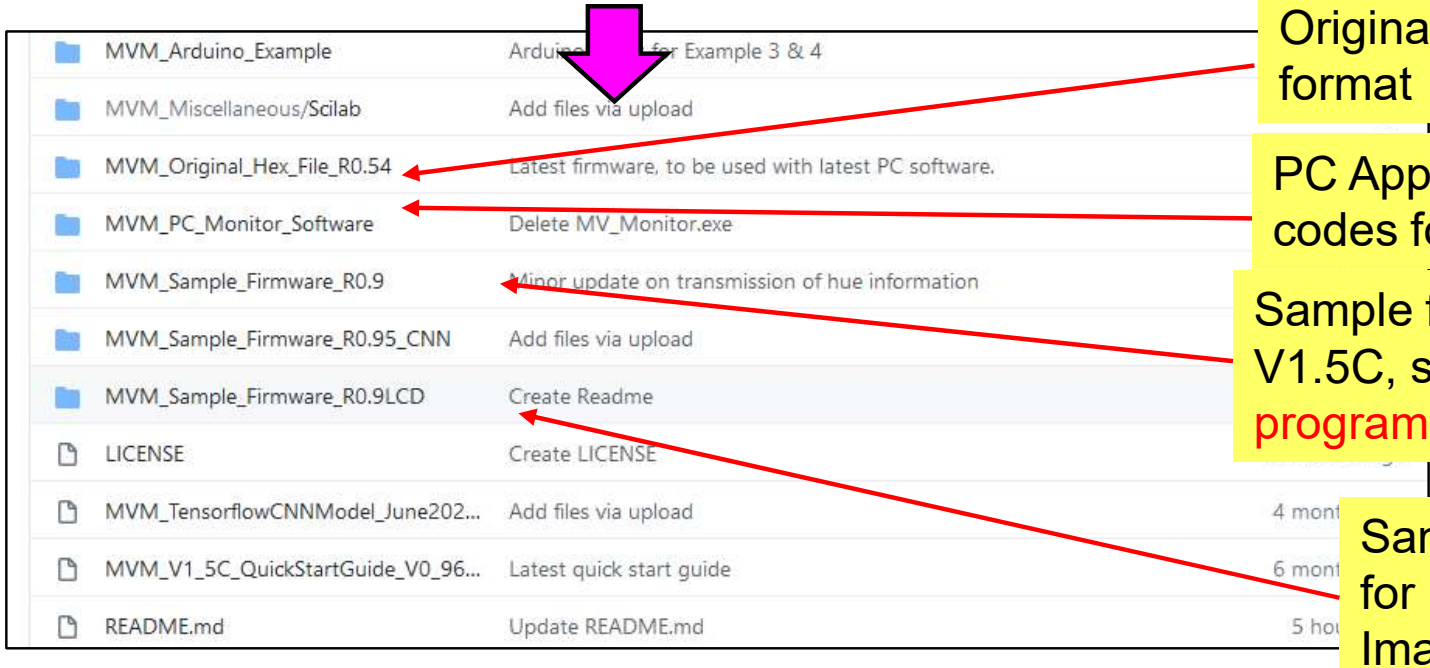
Compiling and Building Your Own Firmware for MVM V1.5C

Introduction 1

- The source codes for the sample firmware is a simplified version of the application pre-loaded into the MVM V1.5C micro-controller.
- The codes for IPA 1 is provided with the sample firmware and if the micro-controller is programmed with the sample firmware hex output, the micro-controller will run IPA 1 continuously at 20 fps upon power up.

Introduction 2

- Clone the MVM_V1_5C folder from https://github.com/fabiankung/MVM_V1_5C



Original firmware in hex format

PC Application and Source codes for MV Monitor software

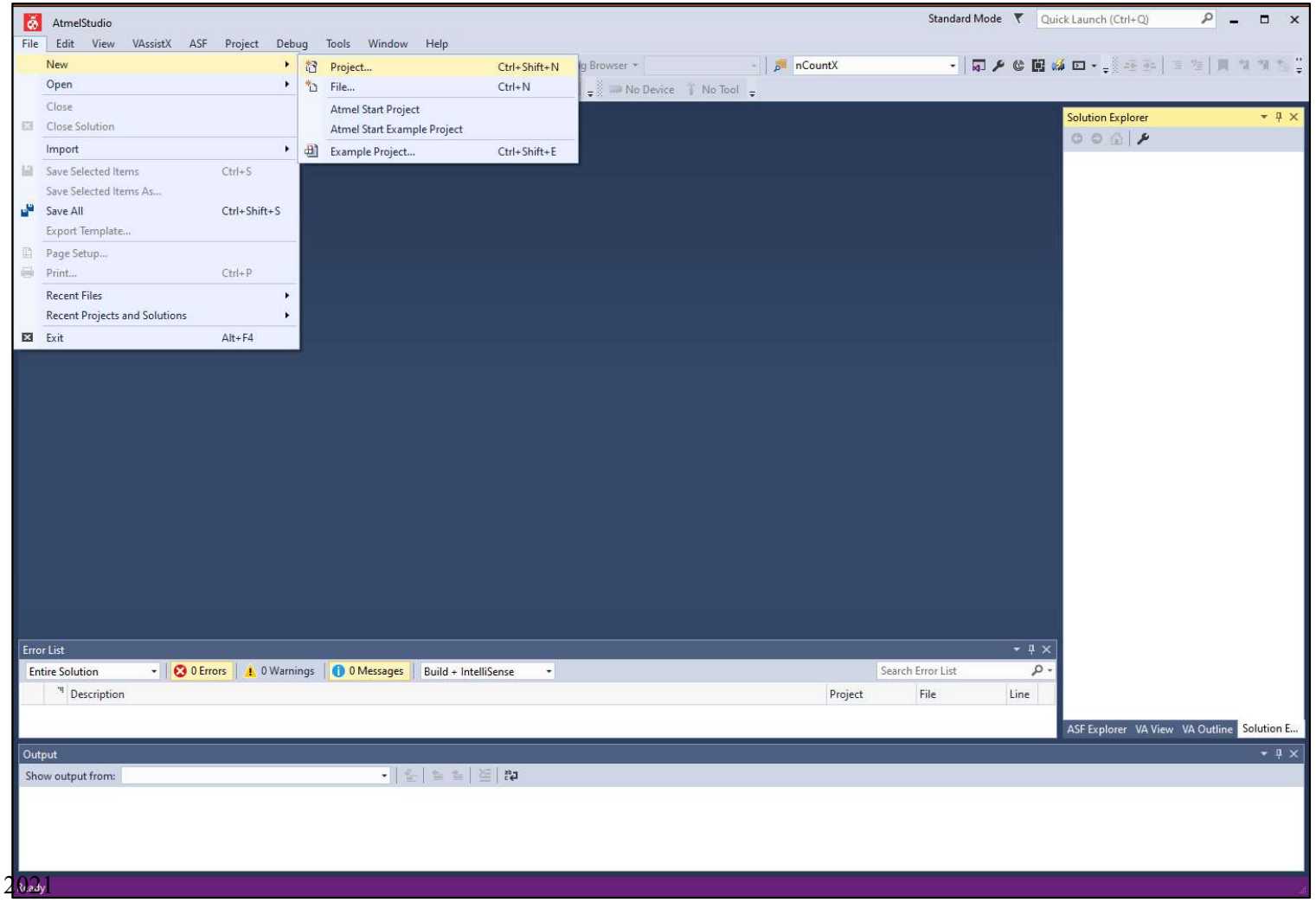
Sample firmware in C for MVM V1.5C, stream image to PC, pre-programmed into unit

Sample firmware in C for MVM V1.5C (stream Image to TFT LCD)

- "MVM_Sample_Firmware" contains all the drivers files and IPA 1 routines. You can use this to build your own custom applications.
- "MVM_PC_Monitor_Software" contains the Visual Studio template to build up the Machine Vision Monitor software in Visual Basic .NET and also the executable (be sure to allow exception from anti-virus software to download the *.exe file)

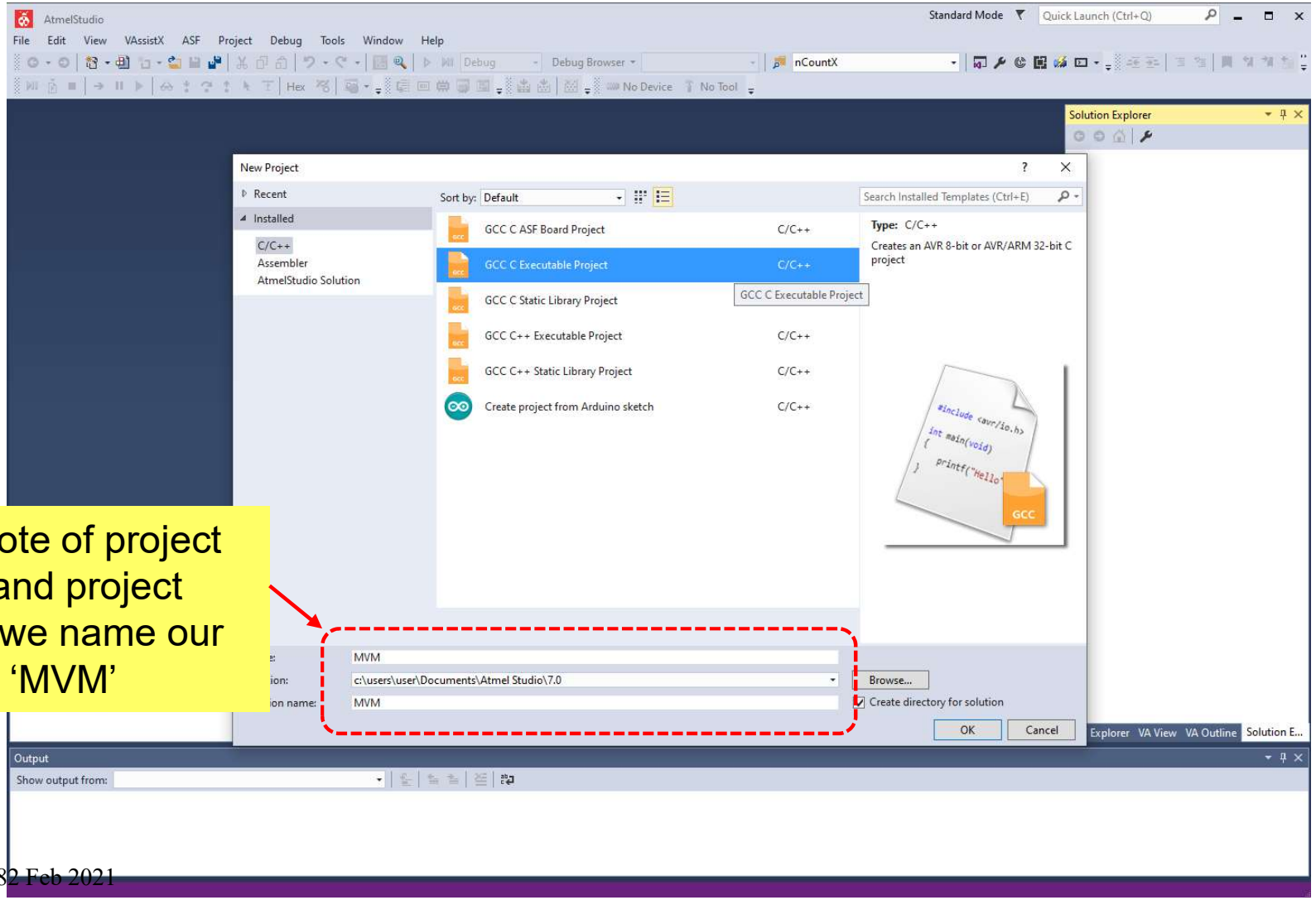
Setting Up An Atmel Studio 7 Project [1 of 10]

- Start a new project in Atmel Studio 7.



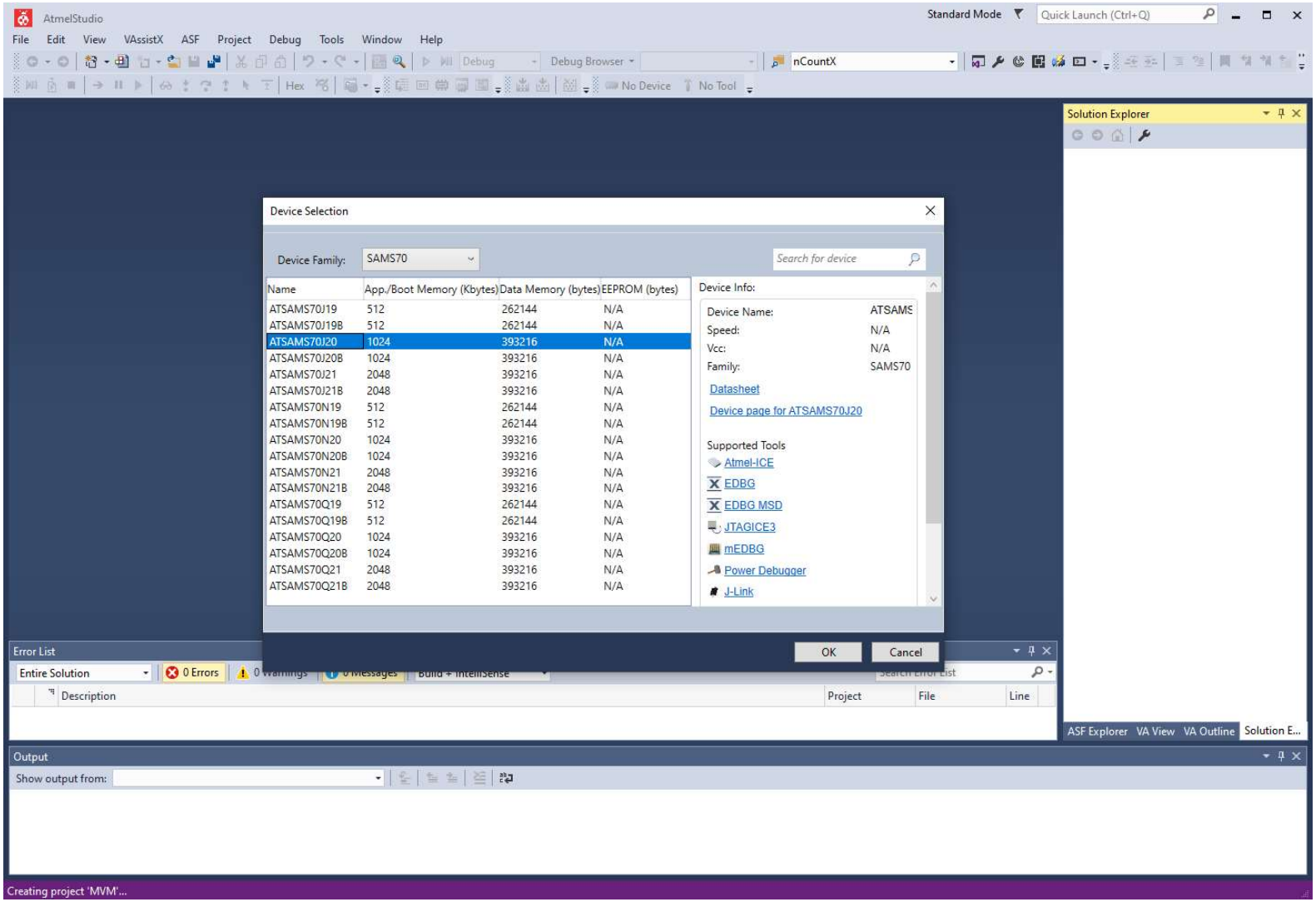
Setting Up An Atmel Studio 7 Project [2 of 10]

- Create a GCC C executable project.



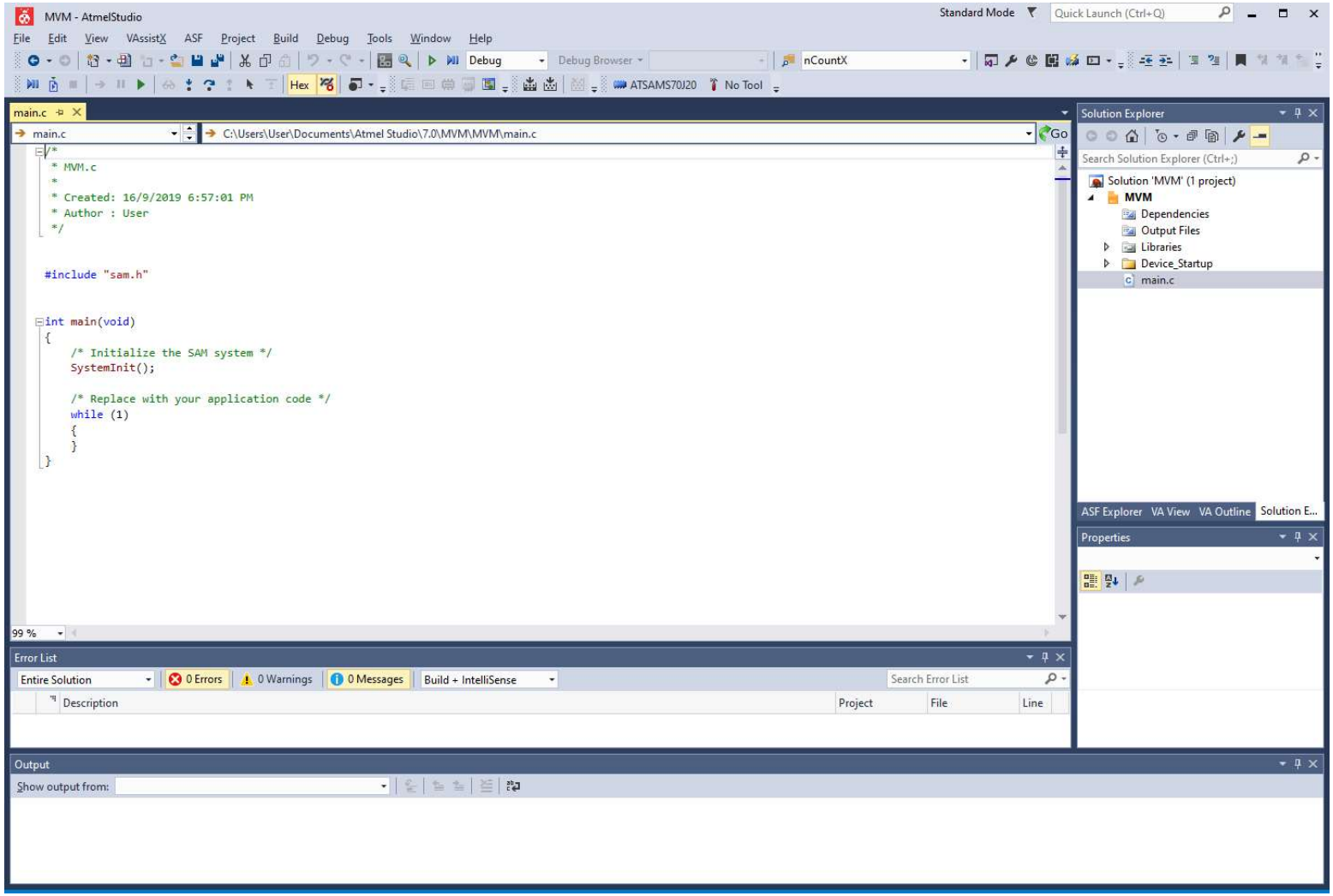
Setting Up An Atmel Studio 7 Project [3 of 10]

- Select the correct device.



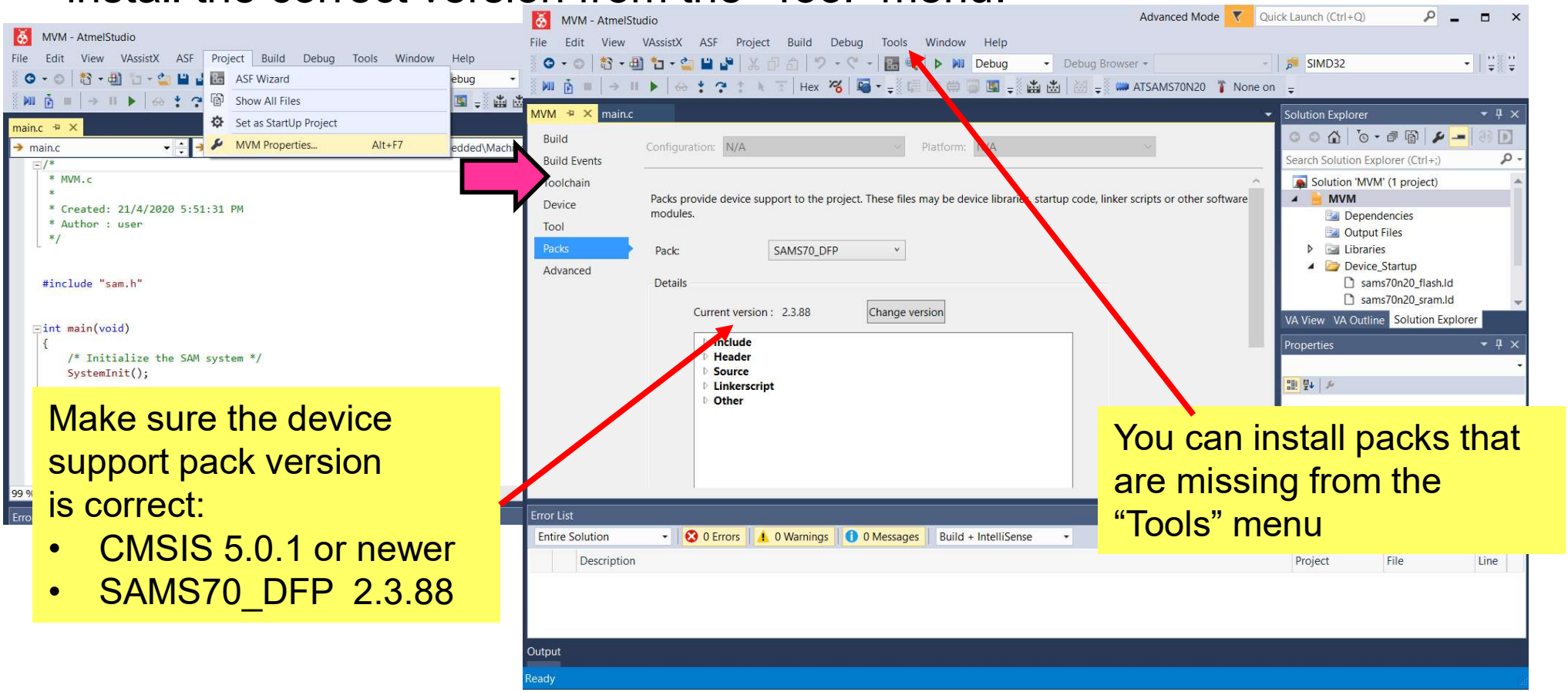
Setting Up An Atmel Studio 7 Project [4 of 10]

- A project with a default “main.c” file will be created.



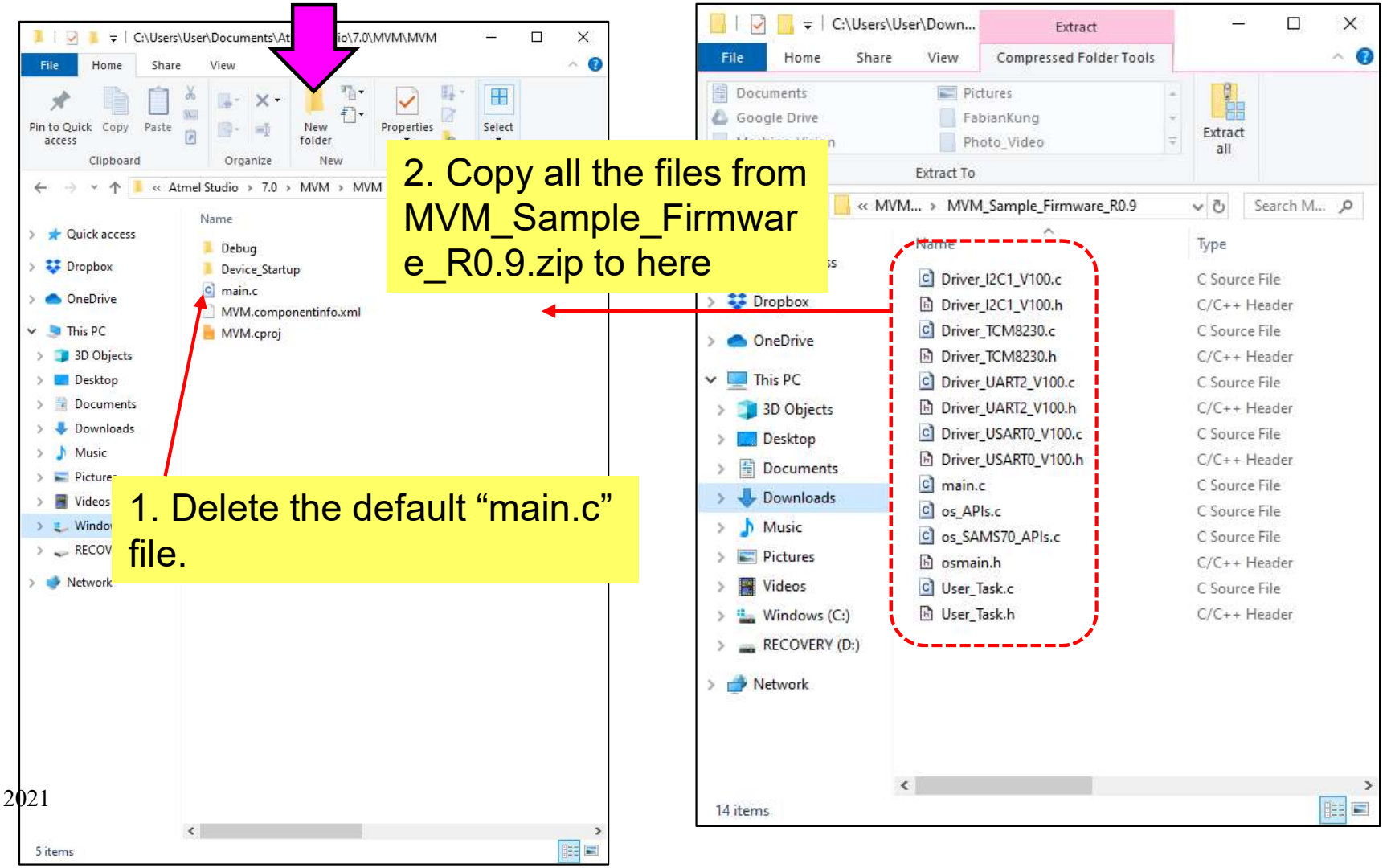
Setting Up An Atmel Studio 7 Project [5 of 10]

- Now view the properties of the project under the “Project” menu, make sure the CMSIS and device support package version is correct. Else you can install the correct version from the “Tool” menu.



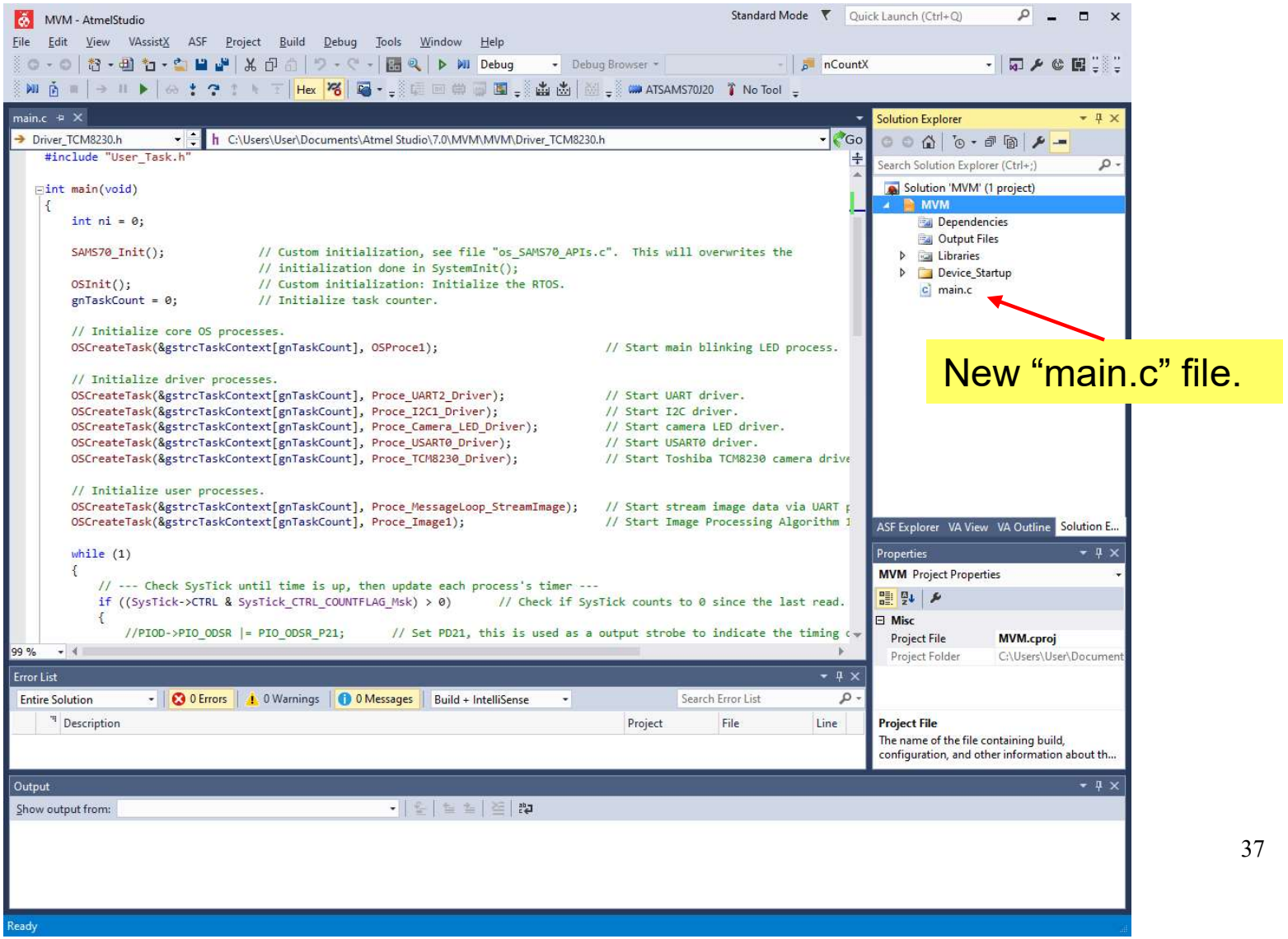
Setting Up An Atmel Studio 7 Project [6 of 10]

- Now close Atmel Studio 7.
- Go to the project folder.



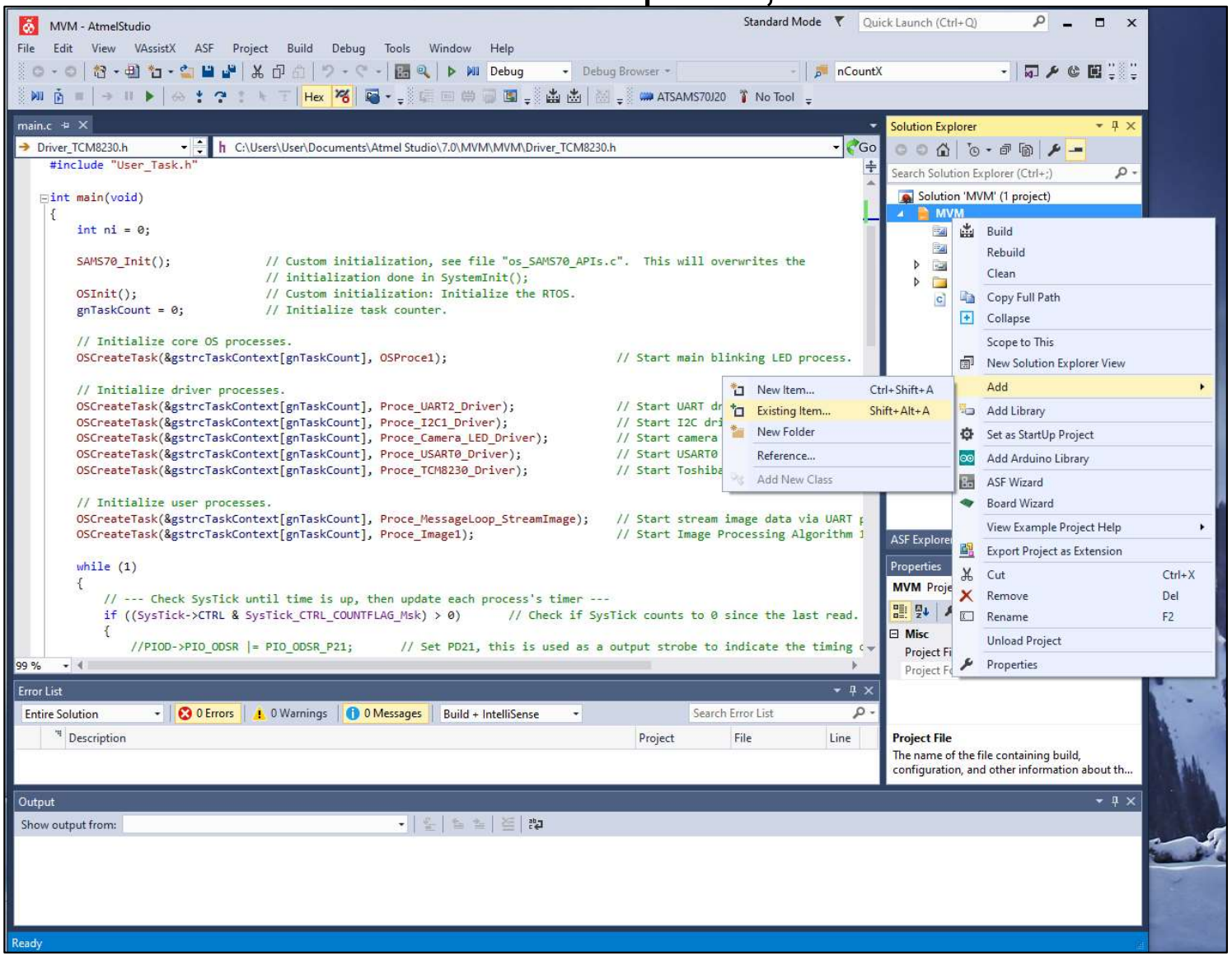
Setting Up An Atmel Studio 7 Project [7 of 10]

- Now reopen Atmel Studio 7. The new “main.c” file will be reflected in the window.



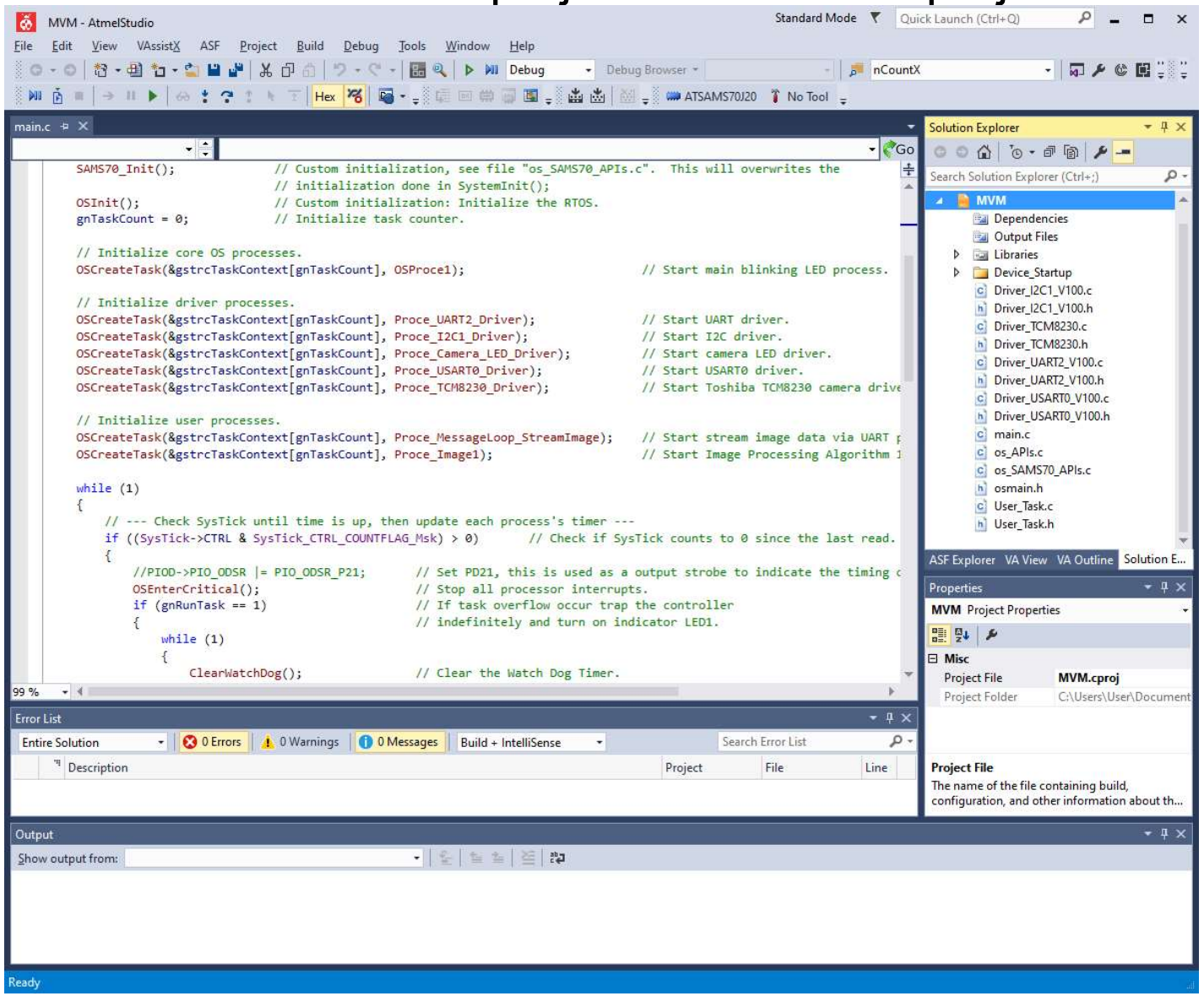
Setting Up An Atmel Studio 7 Project [8 of 10]

- Right click the folder “MVM” in the Solution Explorer, and select Add Existing Item…



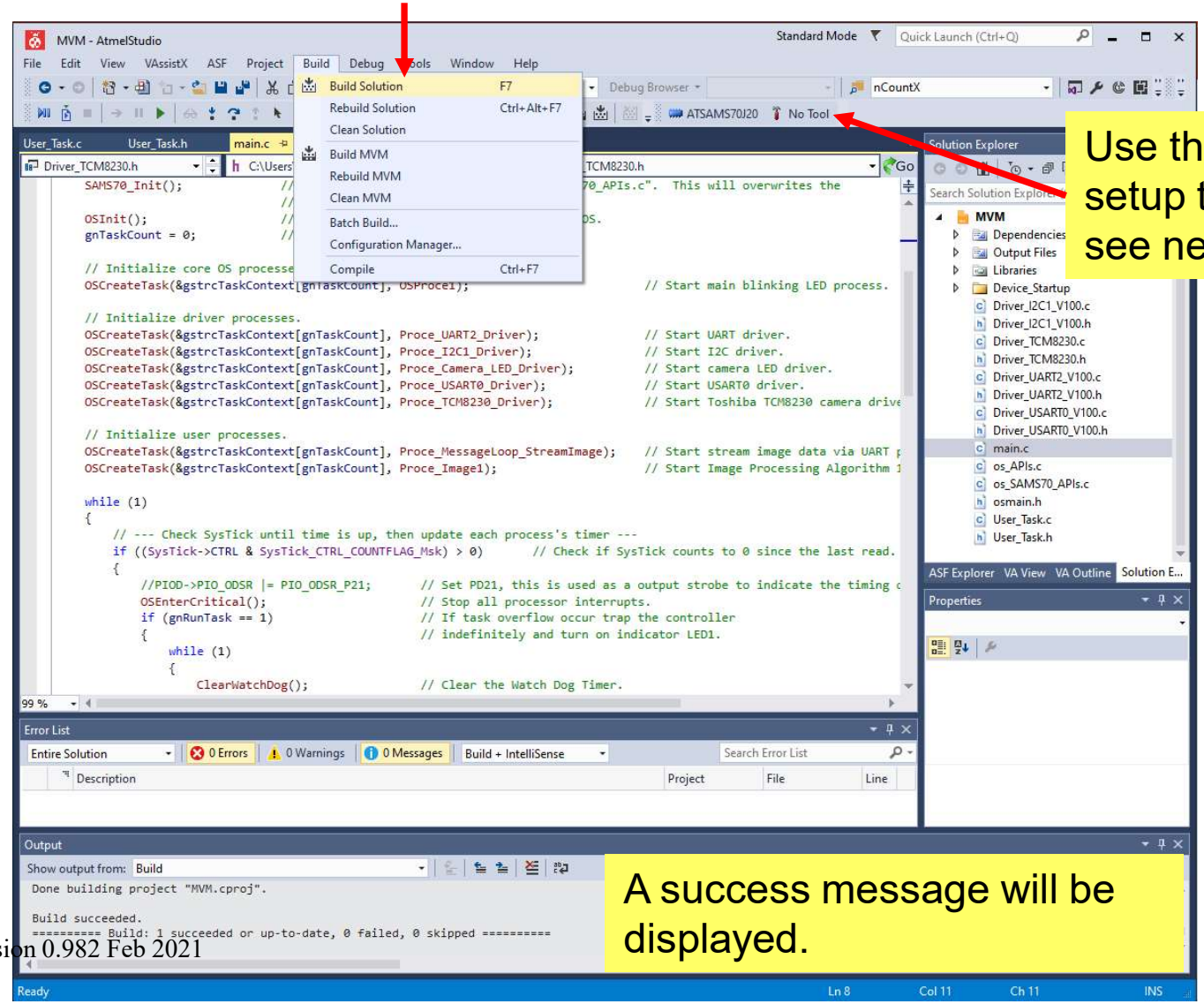
Setting Up An Atmel Studio 7 Project [9 of 10]

- Add all the *.c and *.h files in the project folder to the project.



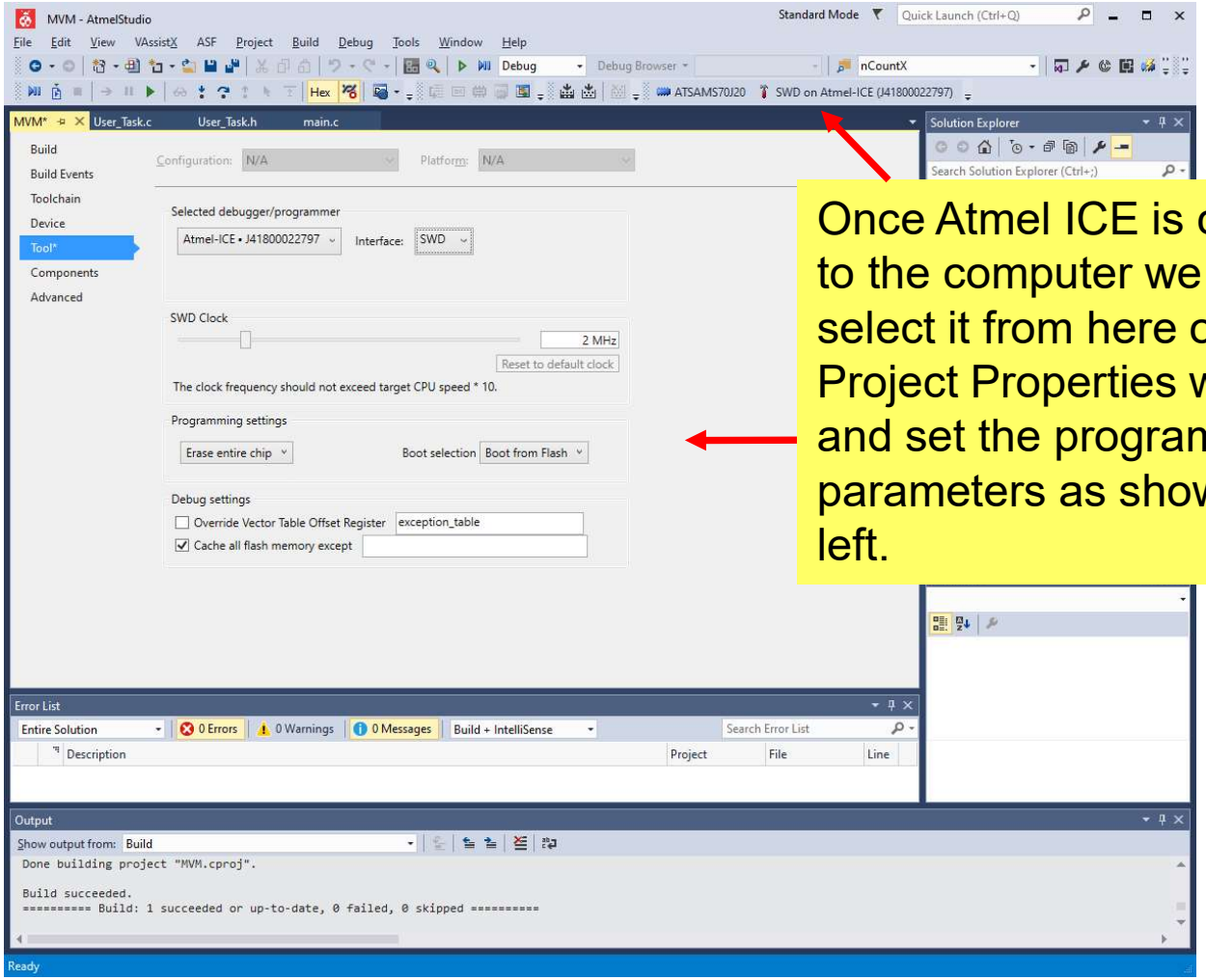
Setting Up An Atmel Studio 7 Project [10 of 10]

- Now you can build or compile the project.



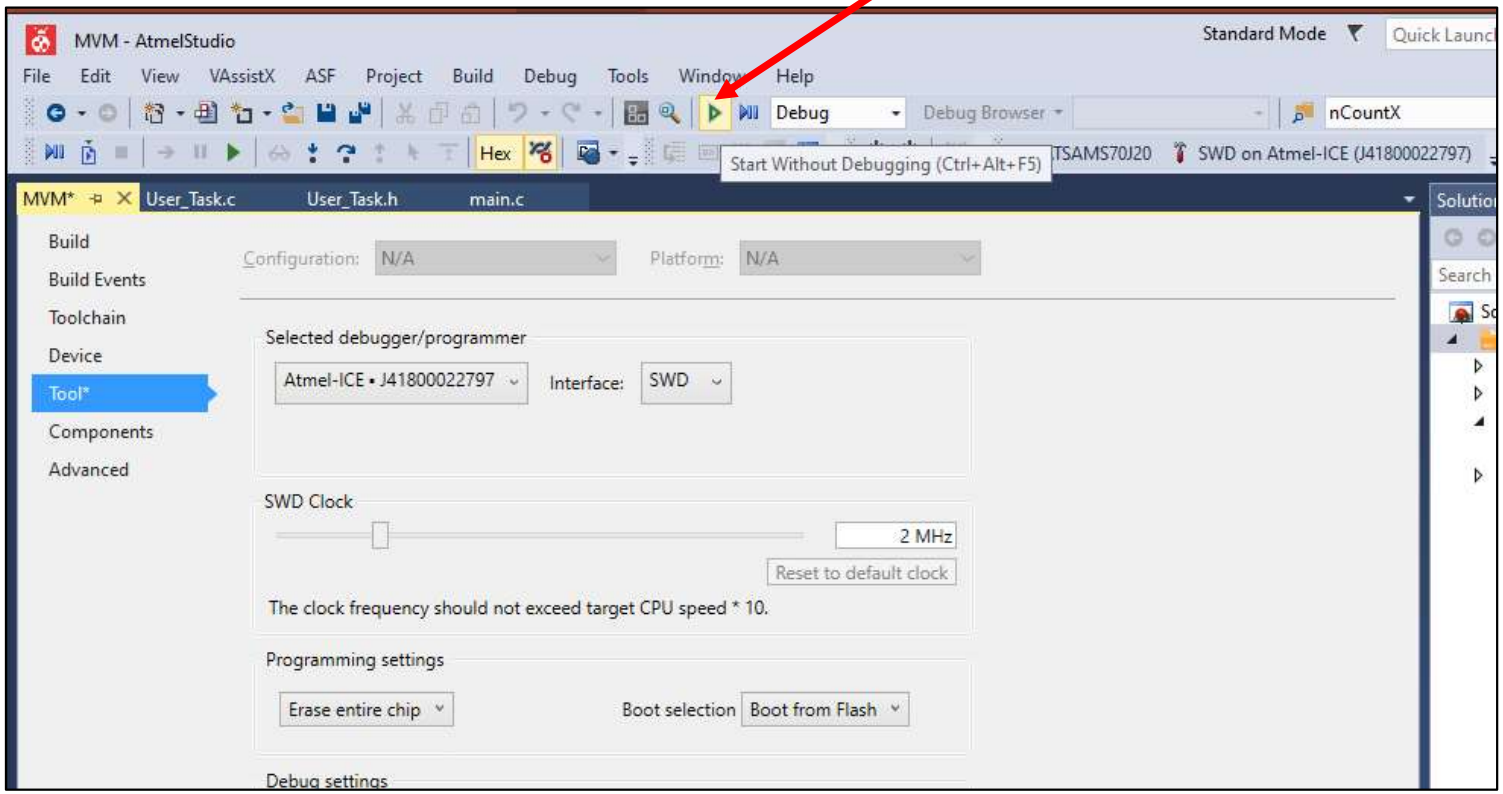
Setting Up the Programming Tool – Atmel ICE

- Now you can load the firmware into the micro-controller with a suitable programmer. Here we are using Atmel ICE, but any programmer compatible with Atmel Studio 7 and support SWD (serial wire debug) mode is fine.



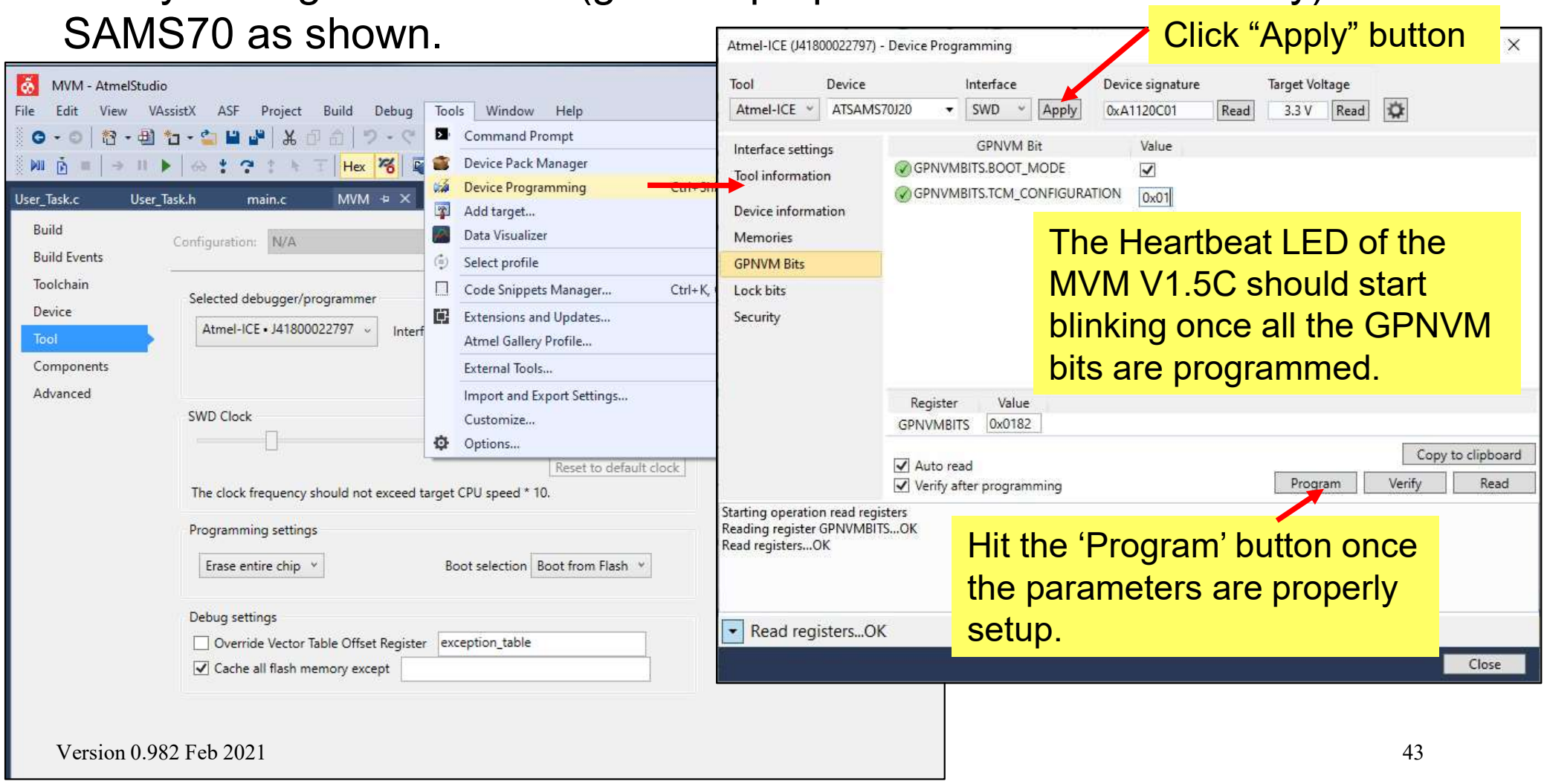
Flashing the Micro-Controller 1

- Connect the MVM to Atmel ICE. Power up the MVM and click this button to program the flash memory.
- See **Appendix** on the pin assignment on the 2x3 ways receptacle that comes with Atmel ICE.



Flashing the Micro-Controller 2

- Finally you need to setup the TCM (tightly coupled memory) size of Cortex M7 by setting the GPNVM (general purpose non-volatile memory) bits of SAMS70 as shown.



Coding Your Own Routines

- The source files “User_Task.c” and “User_Task.h” contains the routines and declarations for **image processing task 1** that search for the brightest region in an image.
- Use this as the basis to add on your own routines. Do remember to use the state machine approach to code your tasks, and keep the total execution time for all tasks within 1 system ticks!
- For more information on the round-robin scheduler and basic structure of the C codes for Atmel/Microchip ARM Cortex-M see https://fkeng.blogspot.com/2016/02/atmel-arm-cortex-m4-microcontroller.html

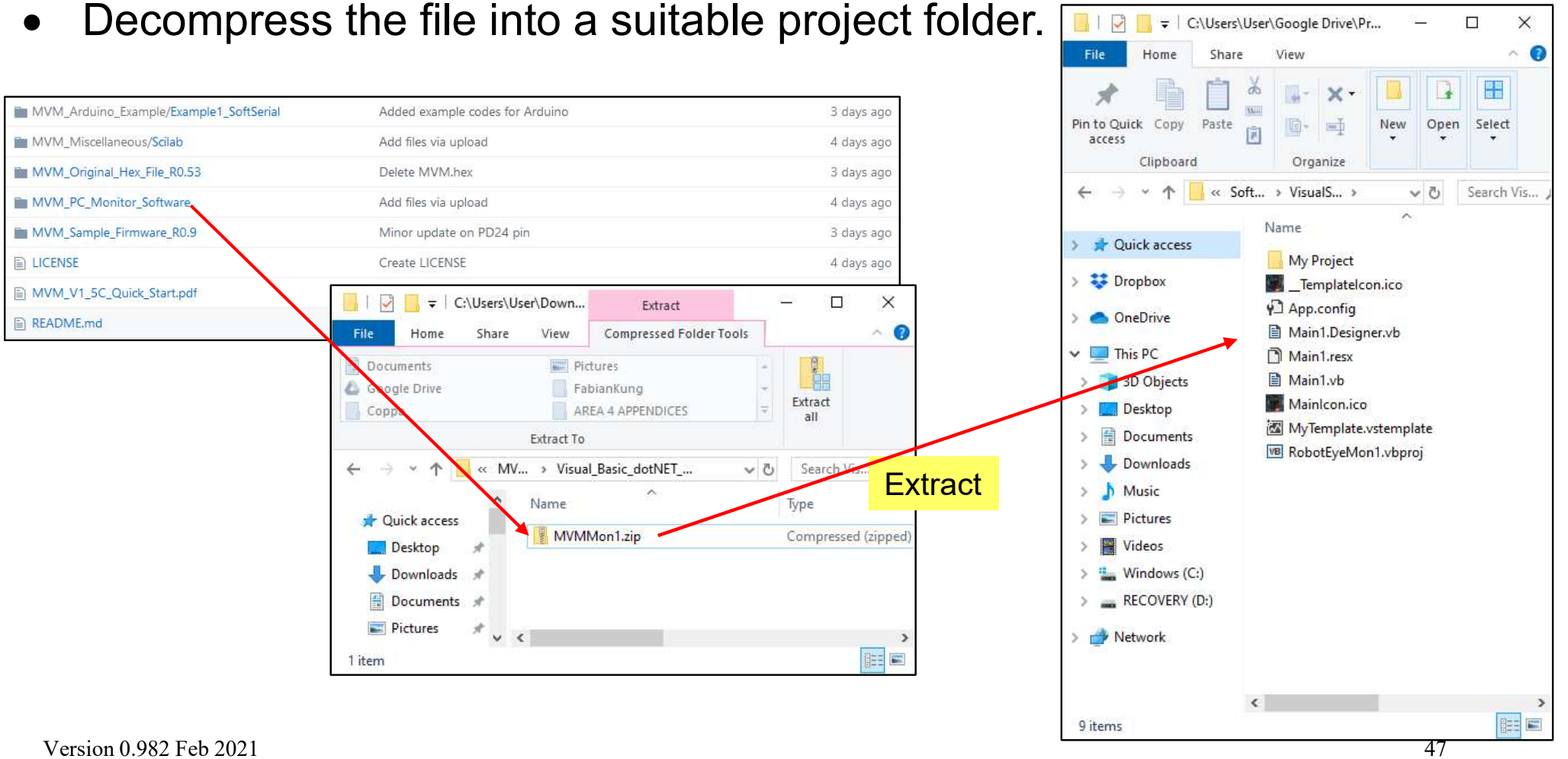
Compiling and Building the Machine Vision Monitor Software

Introduction

- The PC application (*.exe) to observe the image frames captured by the MVM and the corresponding source codes are also provided.
- If needed, you can rebuild the application using Visual Studio Community version and customize the software features.
- The following slides show how to setup the Visual Studio project from the source codes provided.

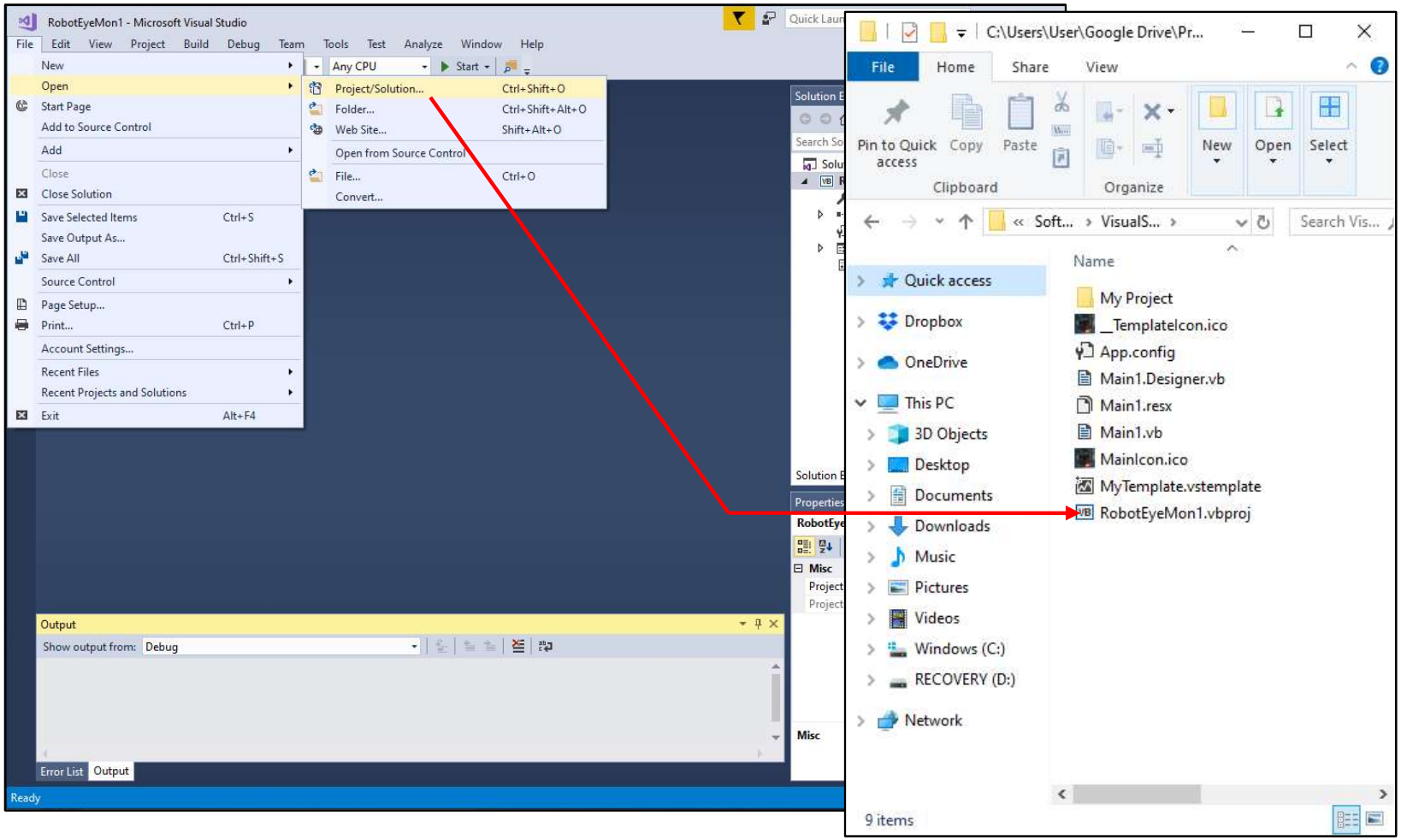
Setting Up Visual Studio Project [1 of 6]

- In the folder “MVM_PC_Monitor_Software” look for the file MVMMon1.zip.
- Decompress the file into a suitable project folder.



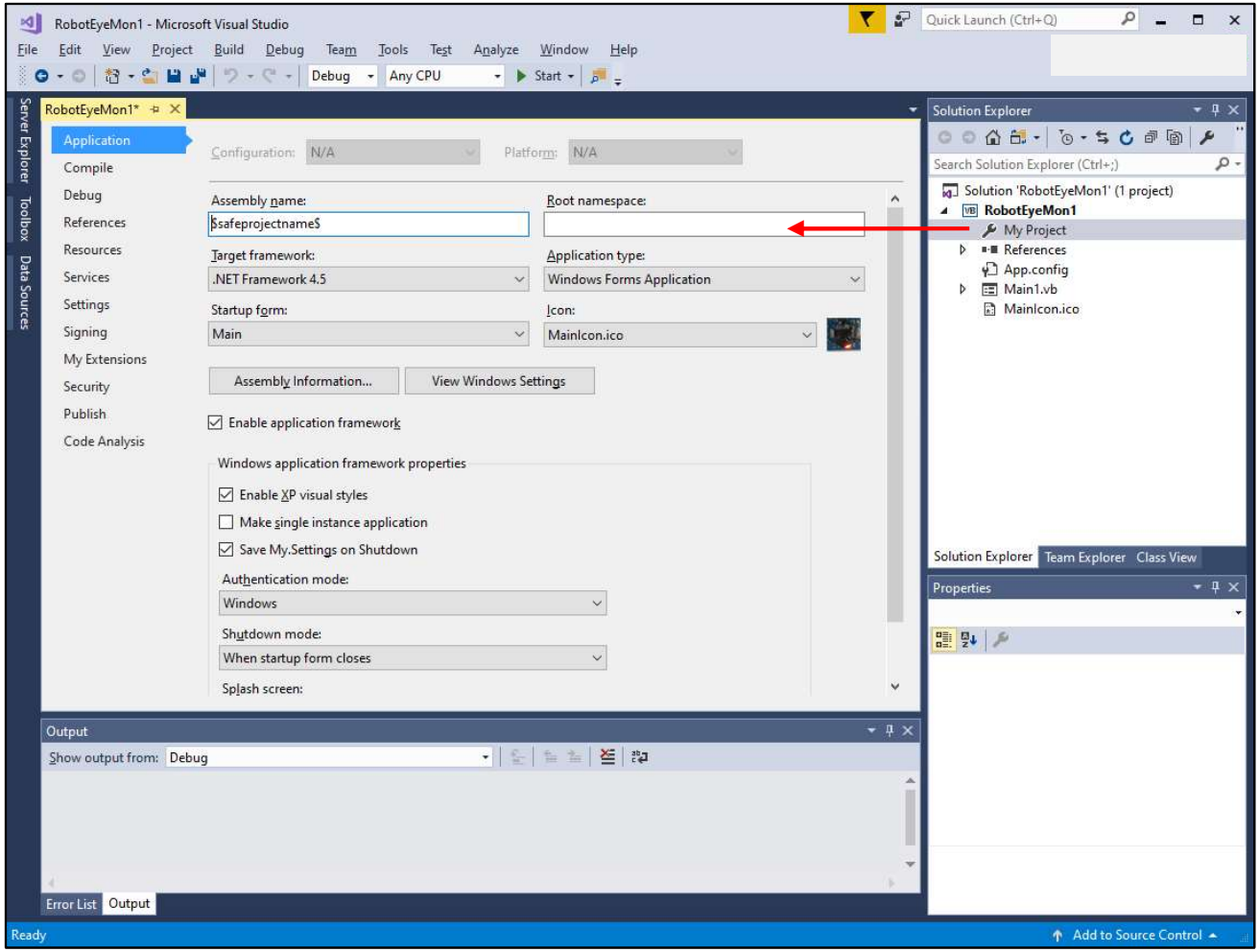
Setting Up Visual Studio Project [2 of 6]

- Open Visual Studio, and open the VB project (*.vbproj) as shown.



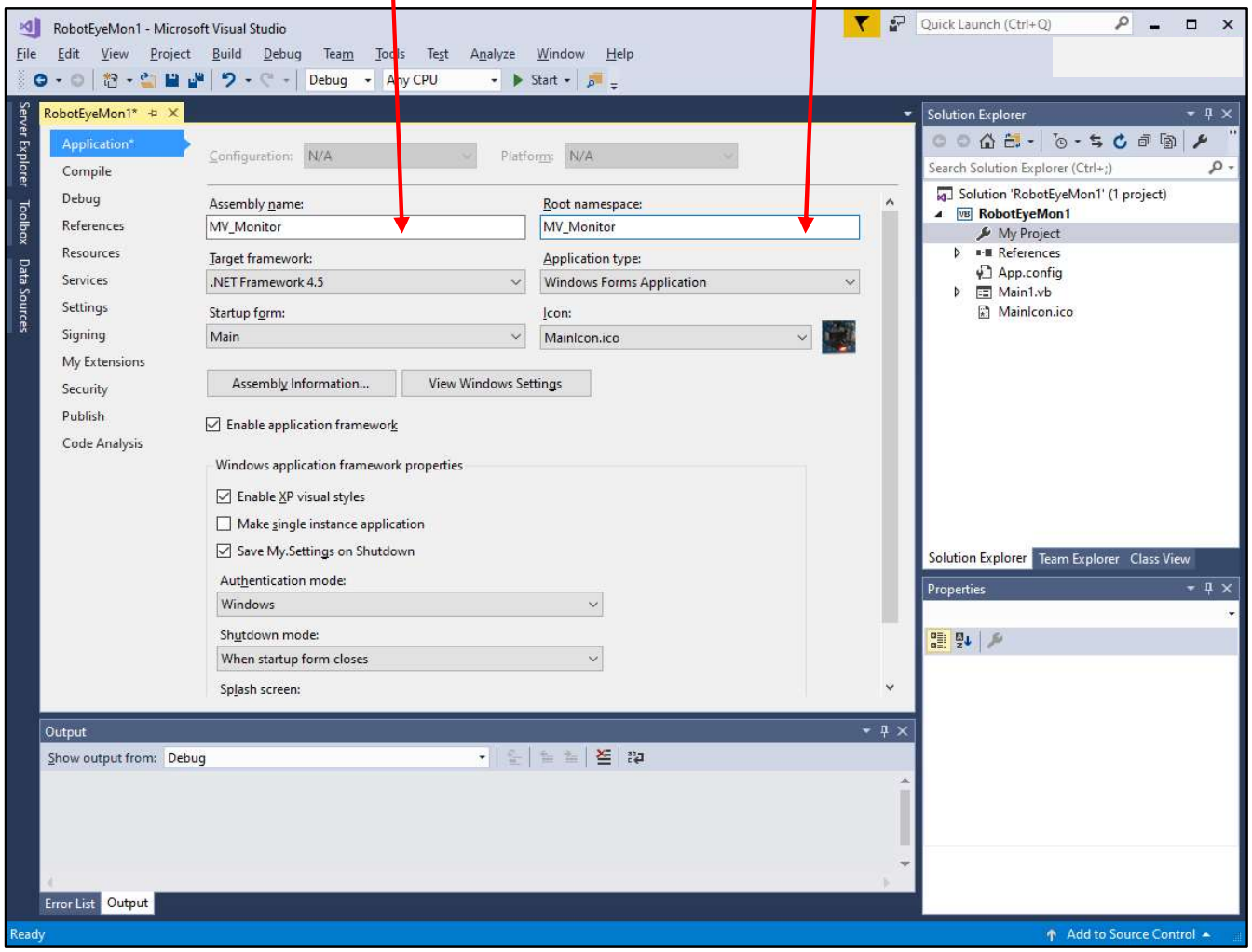
Setting Up Visual Studio Project [3 of 6]

- Double-click the MyProject icon to bring up the project setting.



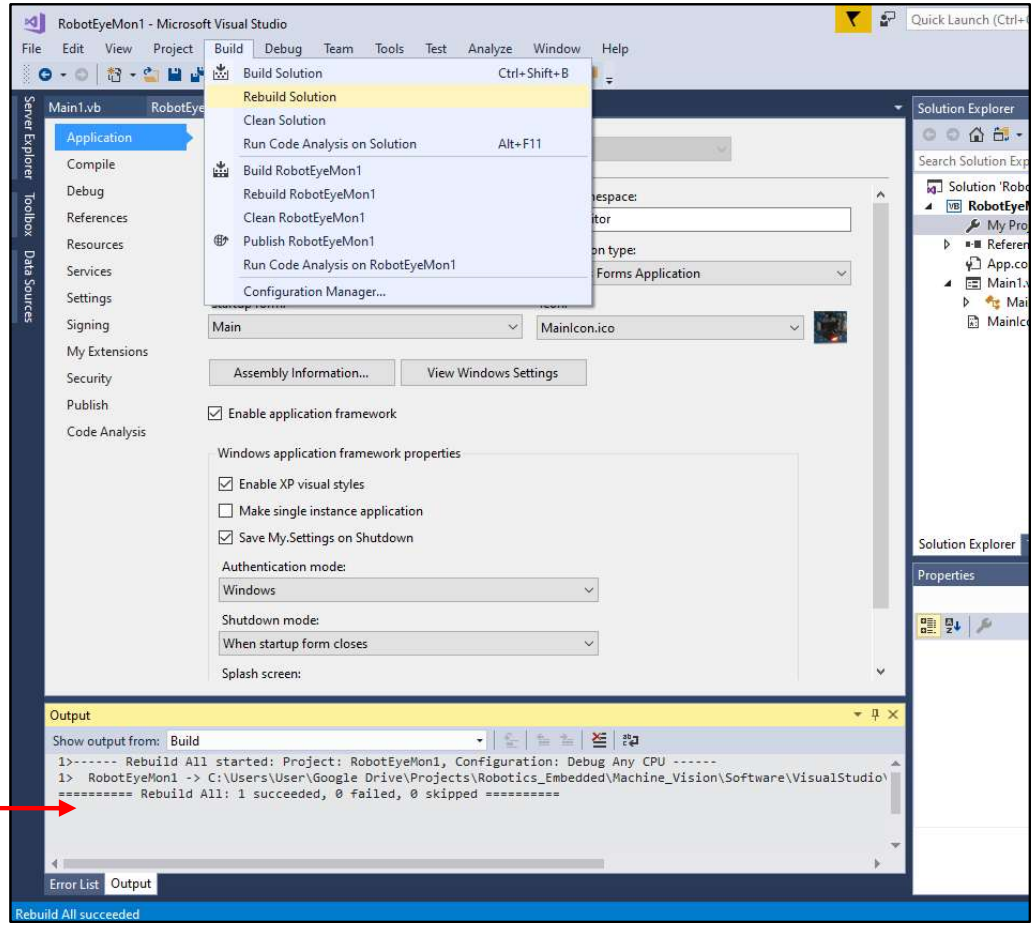
Setting Up Visual Studio Project [4 of 6]

- Type in the Assembly Name and Root Namespace as shown.



Setting Up Visual Studio Project [5 of 6]

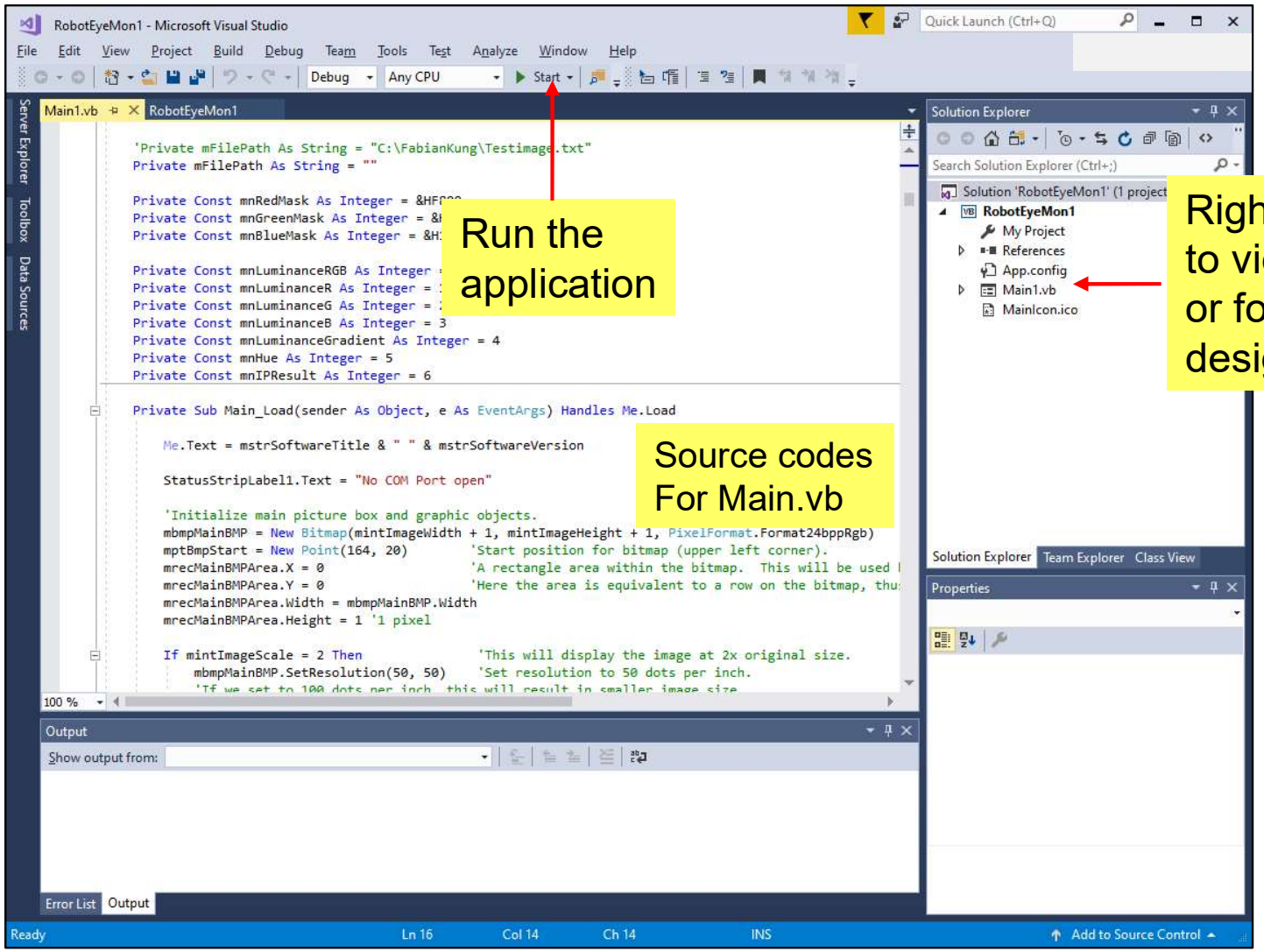
- Now rebuild the project as shown and you should get a success message in the Output window.



Success Message

Setting Up Visual Studio Project [6 of 6]

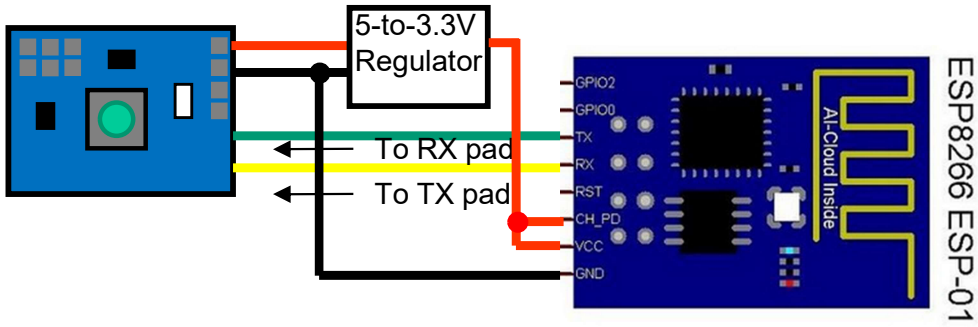
- You can now run the application, view/edit the source code and the main window form.

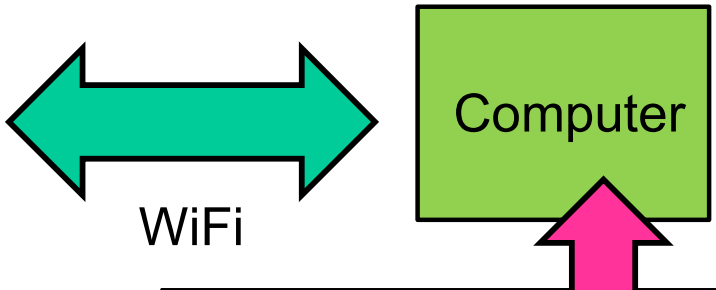


WiFi Connectivity Using ESP8266 [New]

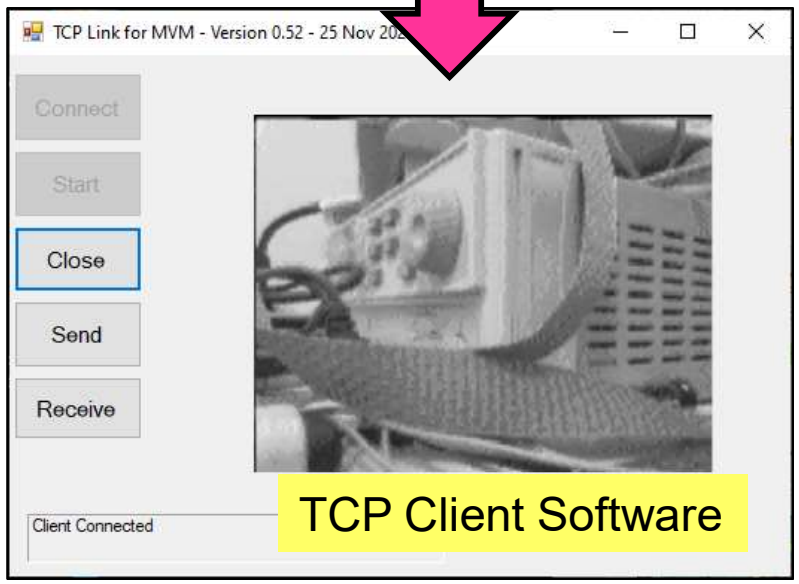
Connecting MVM V1.5C to ESP8266 WiFi Module

- It is now possible to connect MVM V1.5C to ESP8266 module, and stream image to a computer using TCP/IP protocol.



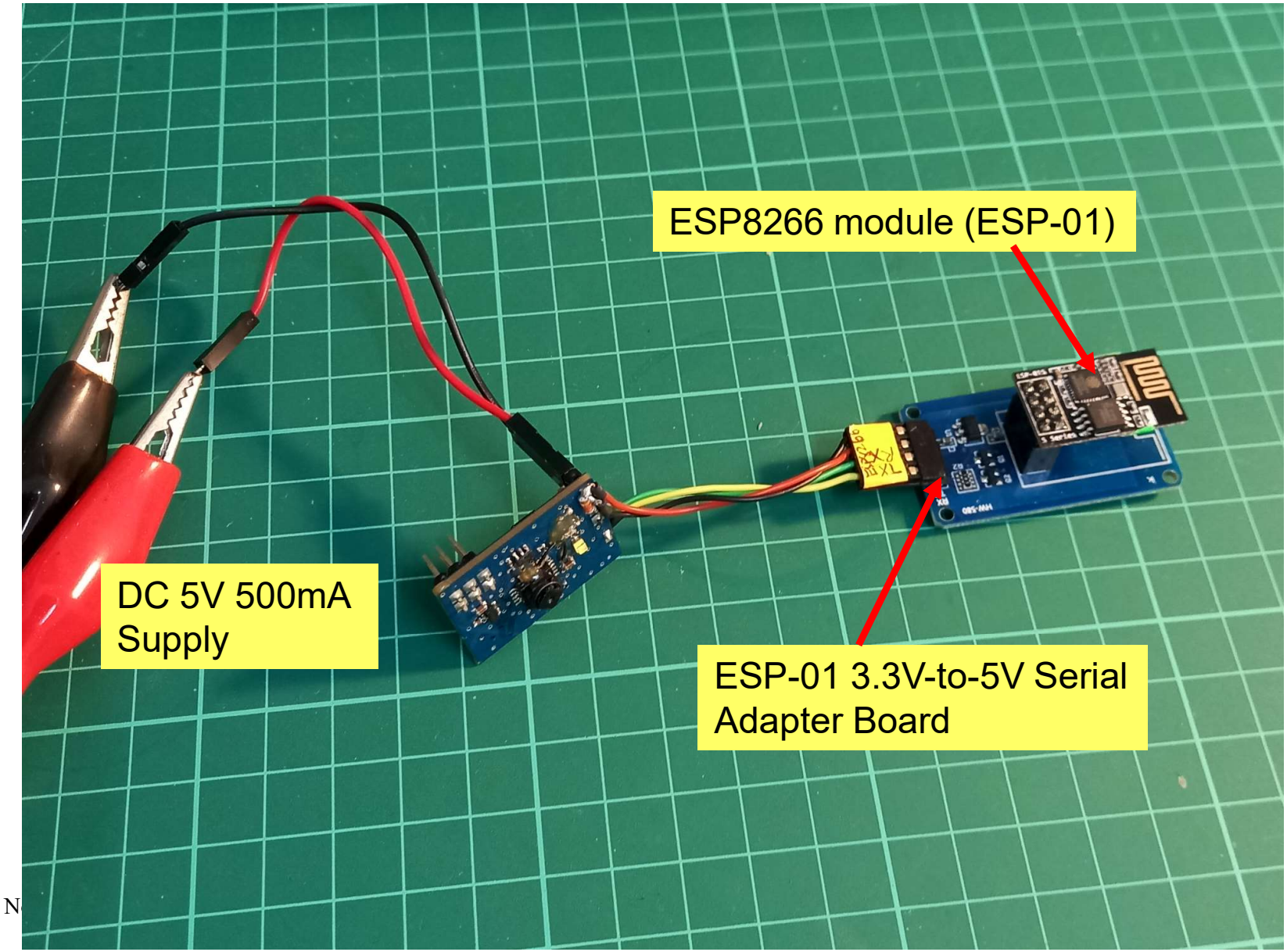


Set ESP8266 as:
1. Soft AP (access point)
2. TCP Server
3. Baud rate = 230400 bps or 345600 bps
(use AT command *AT+UART_DEF = 345600,8,1,0,0* to change the default baud rate to 345600 bps)



See GitHub for source codes and executable

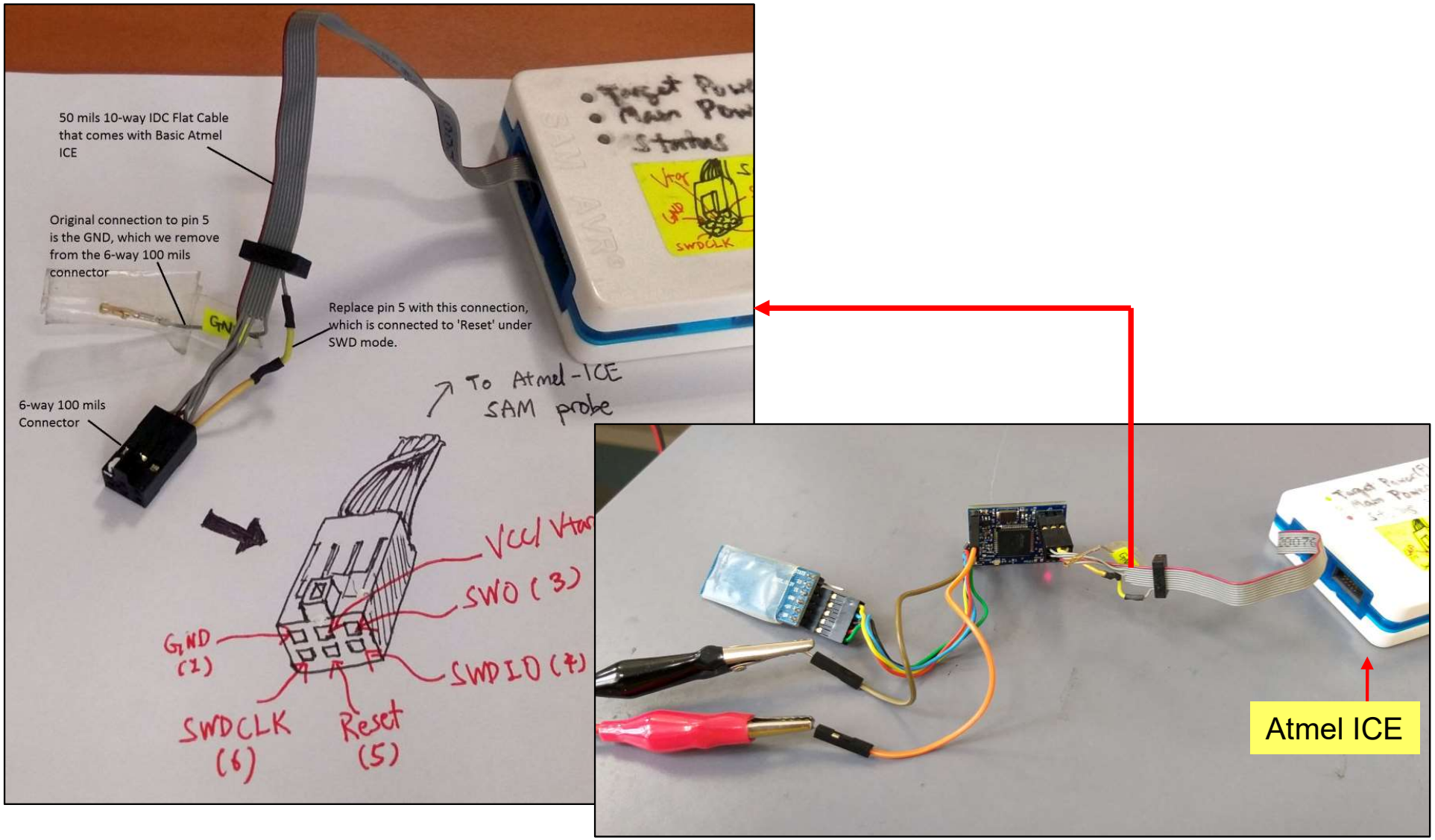
The System Setup



APPENDIX

Programmer Connection, Examples and File Export

Connecting Atmel ICE to MVM V1.5C

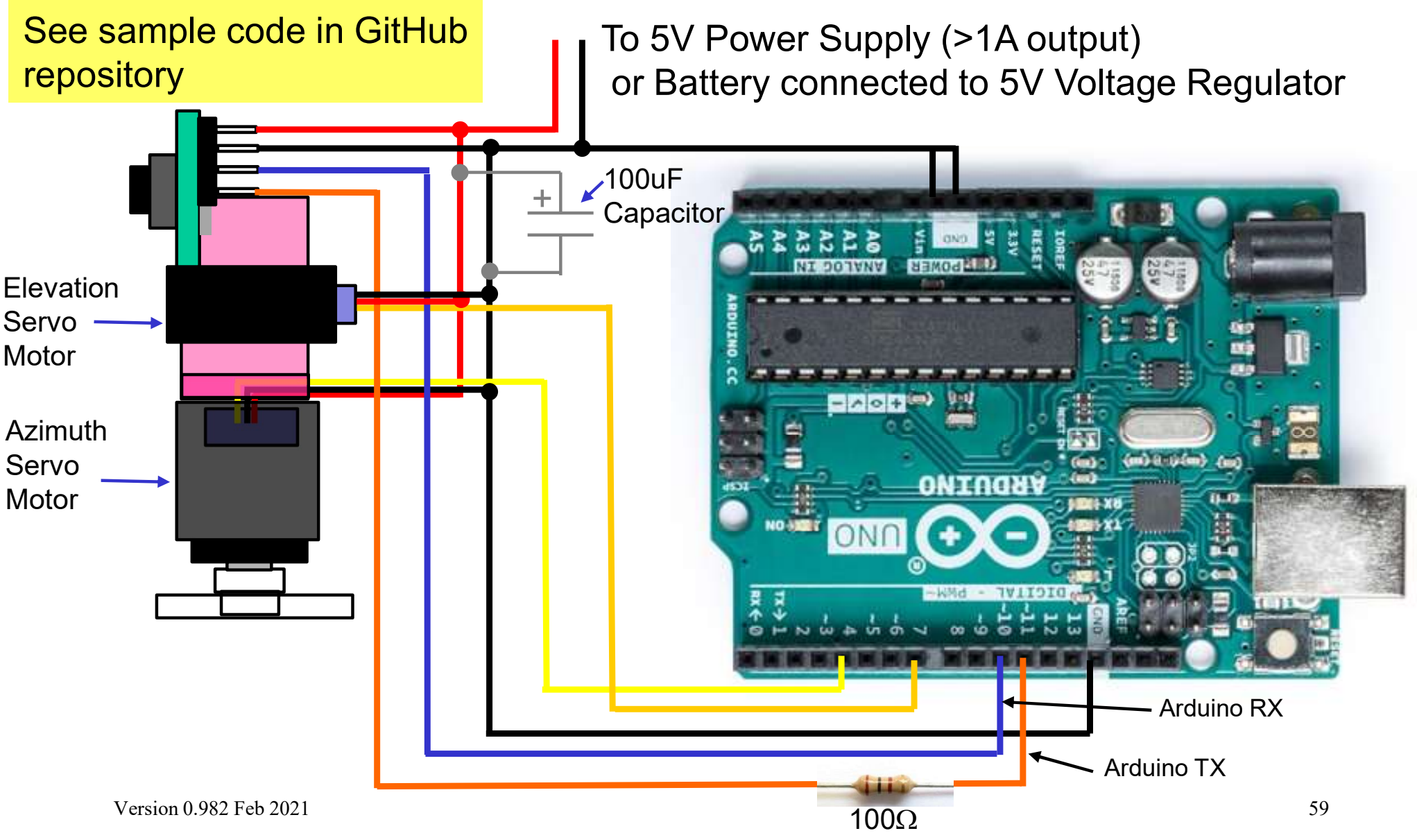


Example 1 - Using MVM with Arduino Uno and SoftwareSerial Library

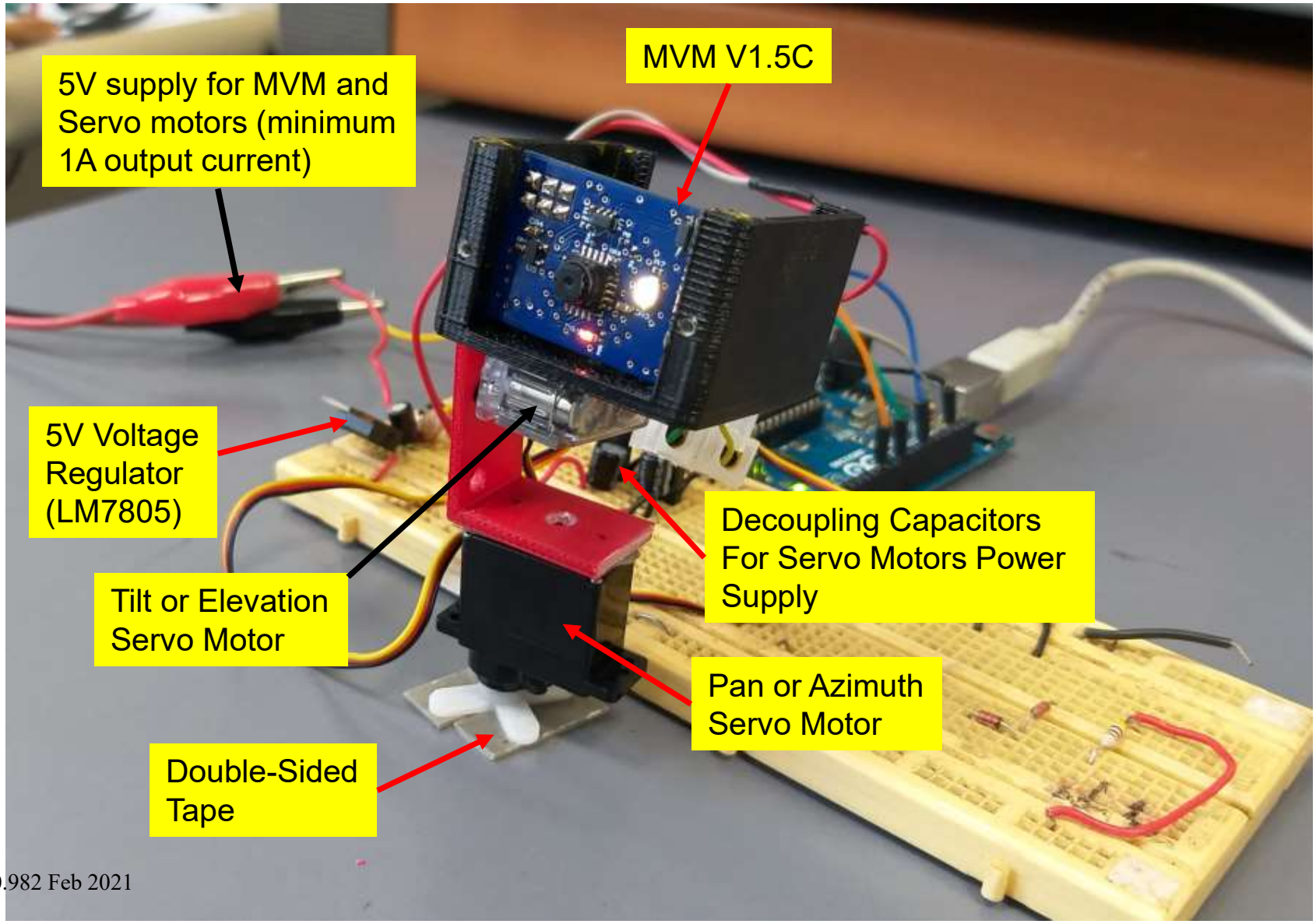
In this code we use SoftwareSerial port to communicate with MVM V1.5C, while the hardware serial is used in conjunction with Serial Terminal for debugging.

Example 3 - Color Object Tracking with Arduino Uno

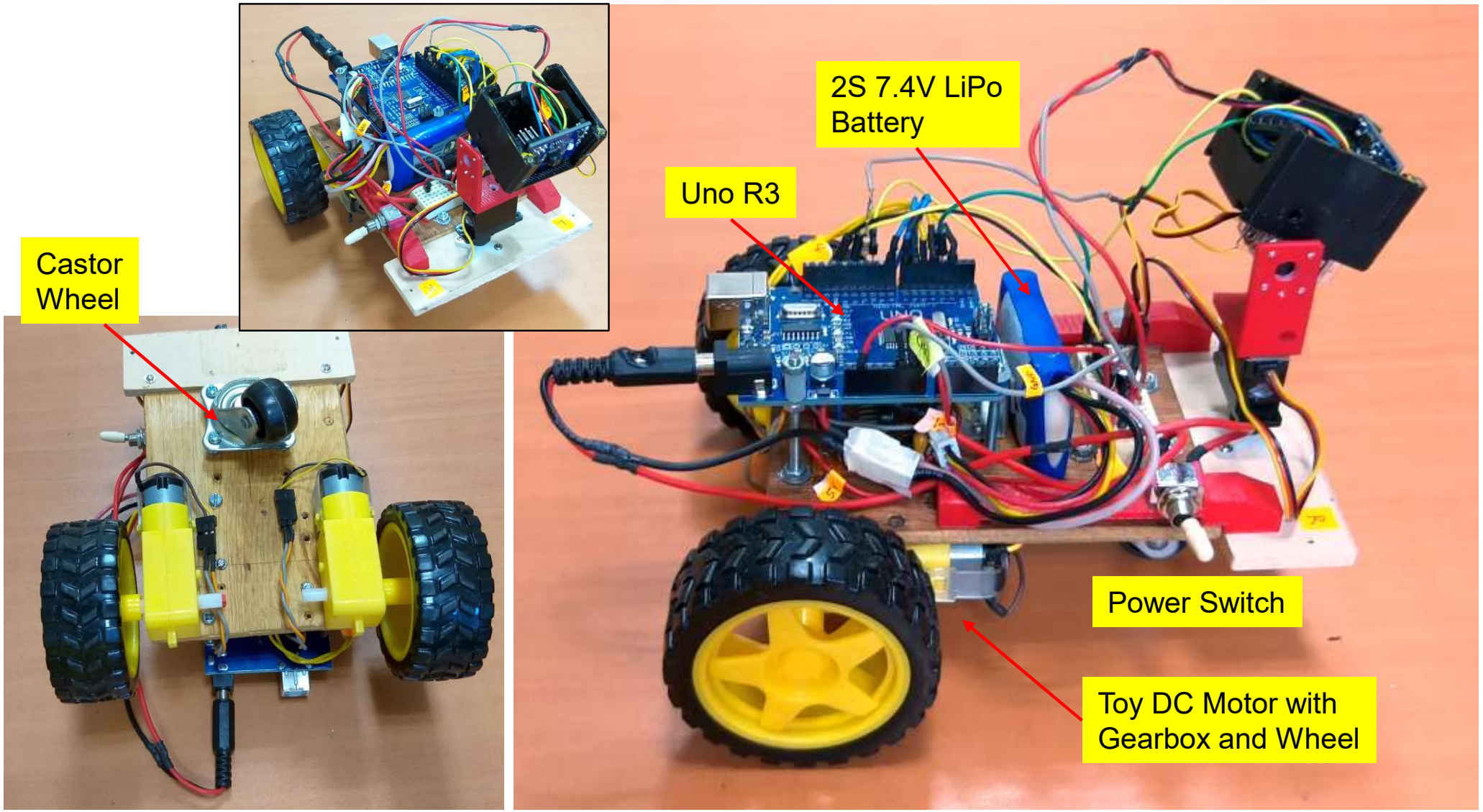
See sample code in GitHub repository



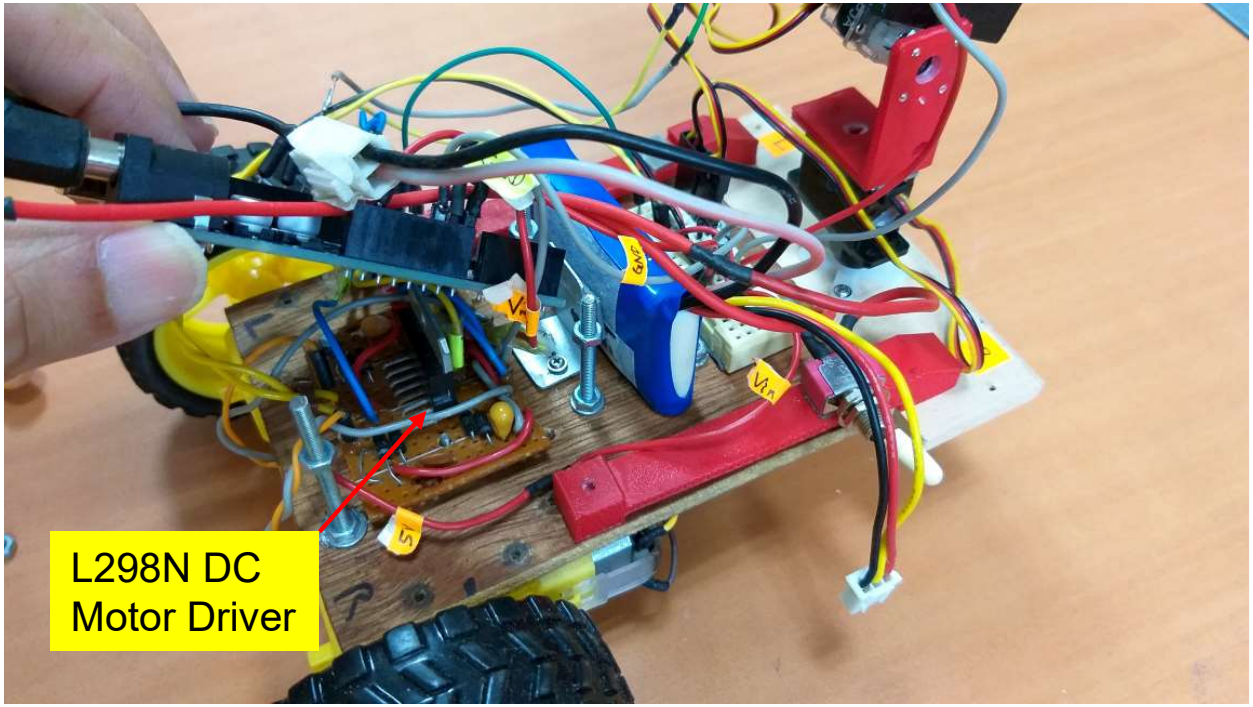
Example 3 - Mechanical Setup



Example 4 - Autonomous Navigation with Arduino Uno Based Robocar



Example 4 - Wiring Information of Robocar



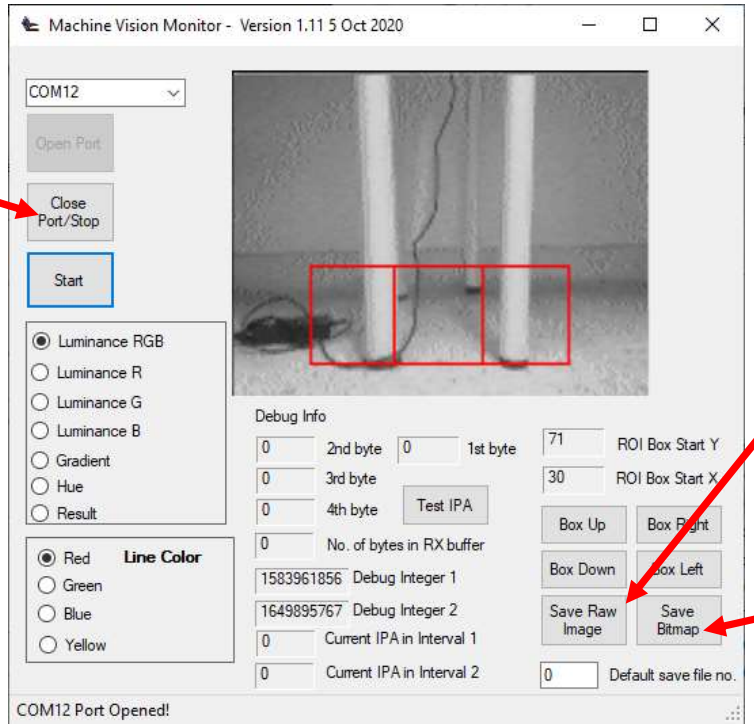
Arduino Uno R3

- Pin 2, 4 – Left DC motor direction control.
- Pin 7, 8 – Right DC motor direction control.
- Pin 5 – Left DC motor speed control.
- Pin 6 – Right DC motor speed control.
- Pin 10 – RX, to MVM TX pin.
- Pin 11 – TX, to MVM RX pin via 100Ω resistor.
- Optional: Pin 3 for azimuth servo motor control and Pin 12 for elevation servo motor control.

Saving the Image Frame onto Computer Hard Disk

- One can save the image displayed in the Machine Vision Monitor software onto hard disk. The image file can be saved as a raw binary file containing 2D array of luminance pixels or in Bitmap format.
- Scilab or MATLAB and python software can be used to read the file and display the image. This is useful when one is developing a new algorithm.
- Steps to do this are shown below.

1. Close the COM port to stop streaming image



2. Save the current display image as raw binary format on hard disk **OR**

3. Save the current display image as bitmap format on hard disk

Retrieving the Saved Raw Image using Scilab or MATLAB software

- The Scilab script to read the saved image file is also provided in the MVM_V1_5C folder. The script listing is shown below.

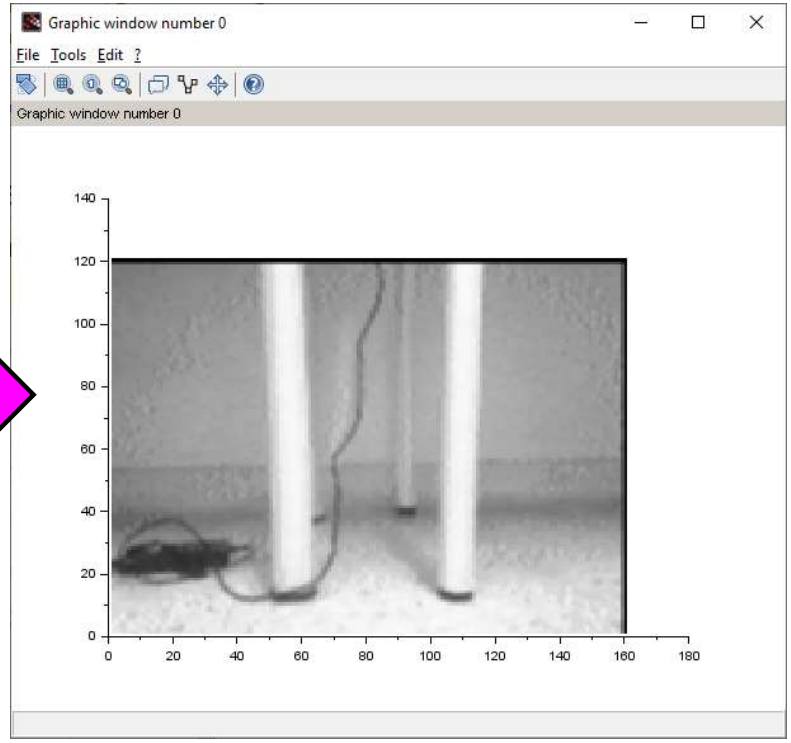
Basic_GrayScale_GrabImageCamera.sce (C:\Users\User\Google

File Edit Format Options Window Execute ?

*Basic_GrayScale_GrabImageCamera.sce

Make sure the path declaration matches your file path in the hard disk

```
// Author      : Fabian Kung
// Last modified : 29 Oct 2016
// Purpose     : Basic code to load a 8-bit grayscale image from file

clear;
ImageWidth = 160;              // Set the size of the image. QQVGA.
ImageHeigth = 120;
Hgraf = scf();                 // Get the handle to current graphic window.
path = cd("C:\tmp");           // Path for the image file.
Hfile = mopen("testimage.txt",'rb');   // Open a text file for reading
                               // (don't skip 0x0D, newline character)
M = zeros(ImageHeigth+1,ImageWidth+1);  // Matrix to hold the gray scale image data
Mt = zeros(ImageWidth+1,ImageHeigth+1); // Another matrix also to hold the gray scale image data.
                               // 't' indicate transpose.
for i=1:ImageHeigth-1          // ImageHeigth-1 due to the last line is not
                               // exported from the camera monitor software
    for j=1:ImageWidth
        M(i,j) = mget(1,'c',Hfile); // Read 1 pixel data, convert to double.
    end
    mgeti(1,'c',Hfile);        // Read the newline/carriage return character.
end

for i=1:ImageHeigth-1          // Transpose and flip the image so that
    for j=1:ImageWidth         // it appear at the correct orientation.
        Mt(j,ImageHeigth-i) = M(i,j) // The original format of M[]is:
    end                        //---------------------> Column
end                            //|
                               //|
                               //|
                               //V
                               //Row

mclose(Hfile);                 // Release file handle.

//Hgraf.color_map = graycolormap(127);   // Set current graphic window color map
Hgraf.color_map = graycolormap(255);   //to gray scale, 127 or 255 levels.
row = 1:ImageWidth + 1;
col = 1:ImageHeigth + 1;
grayplot(row,col,Mt);          // Plot the transpose and flip image.
```



Retrieving the Saved Bitmap Image using Python

- The python script to read the saved bitmap file is also provided in the MVM_V1_5C folder. The script listing is shown below. Here the IDE Spyder is used.