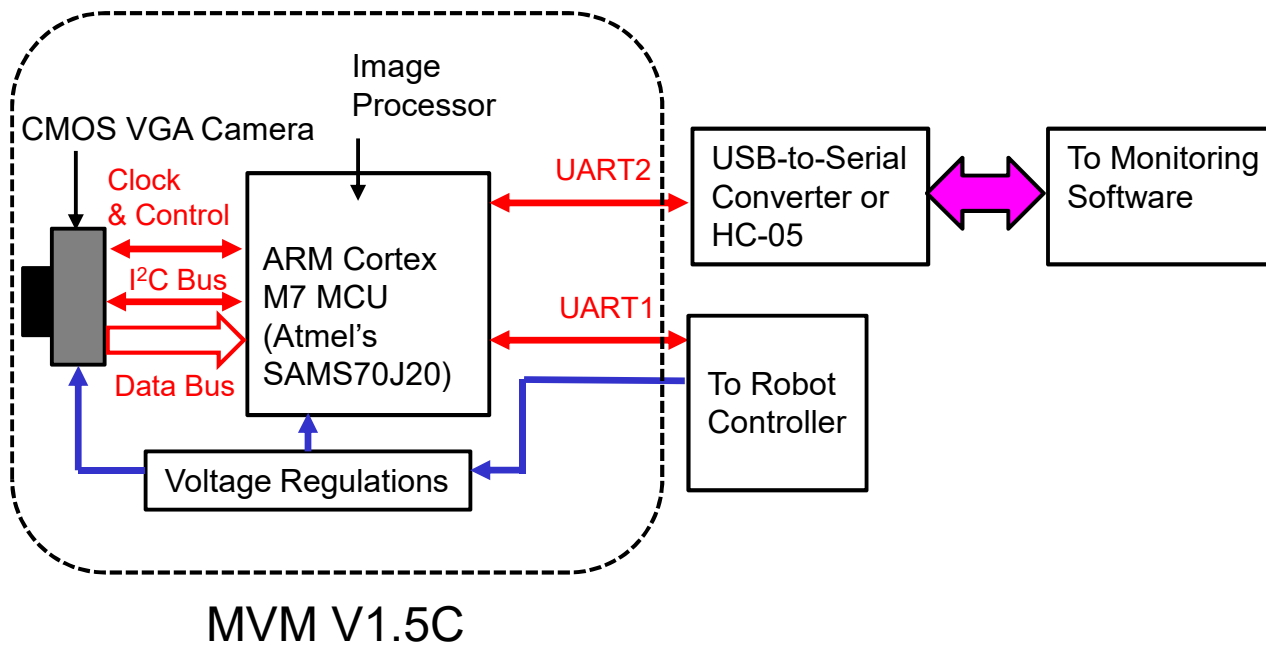
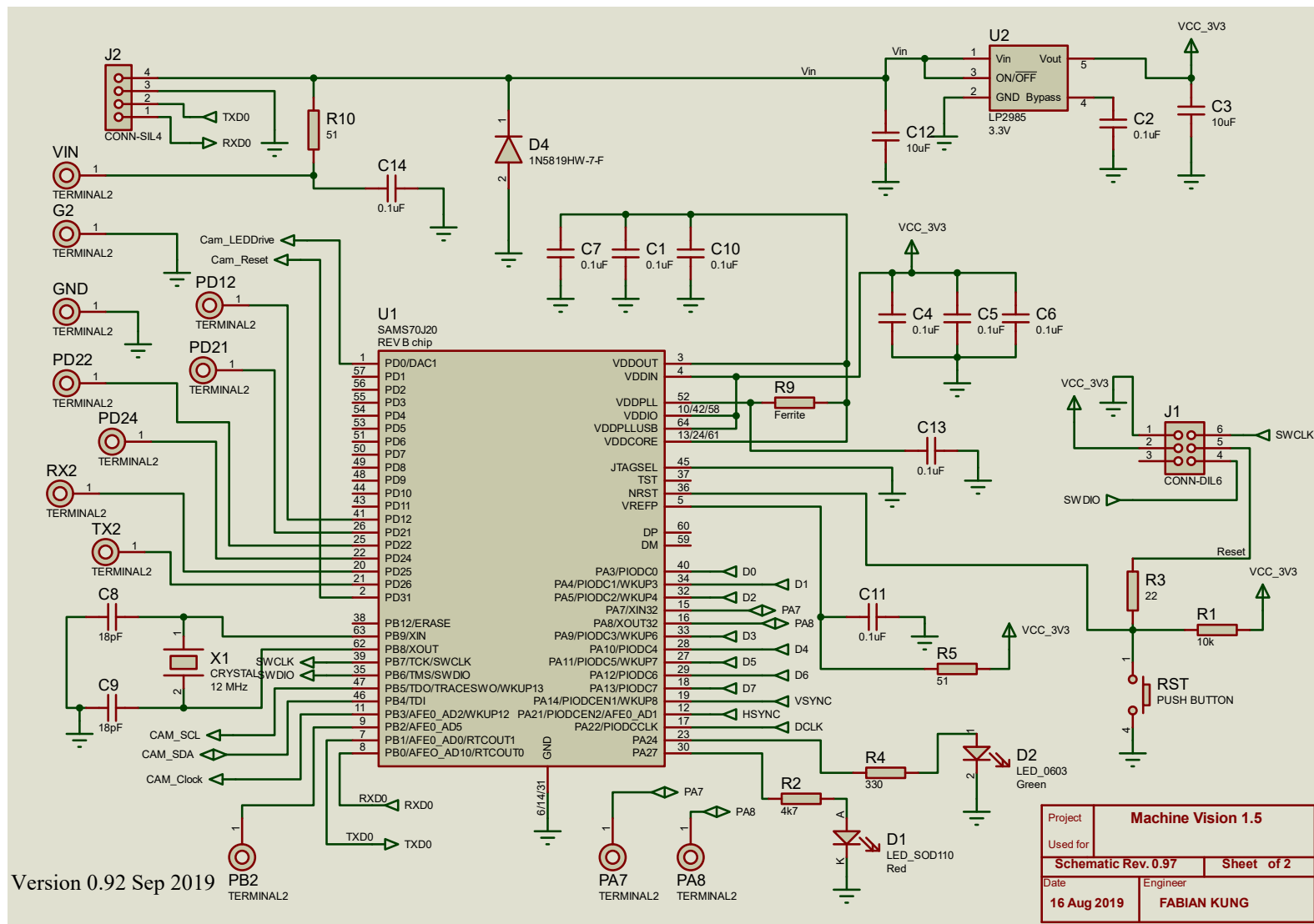

MVM V1.5C Quick Start Guide

Rev 0.92

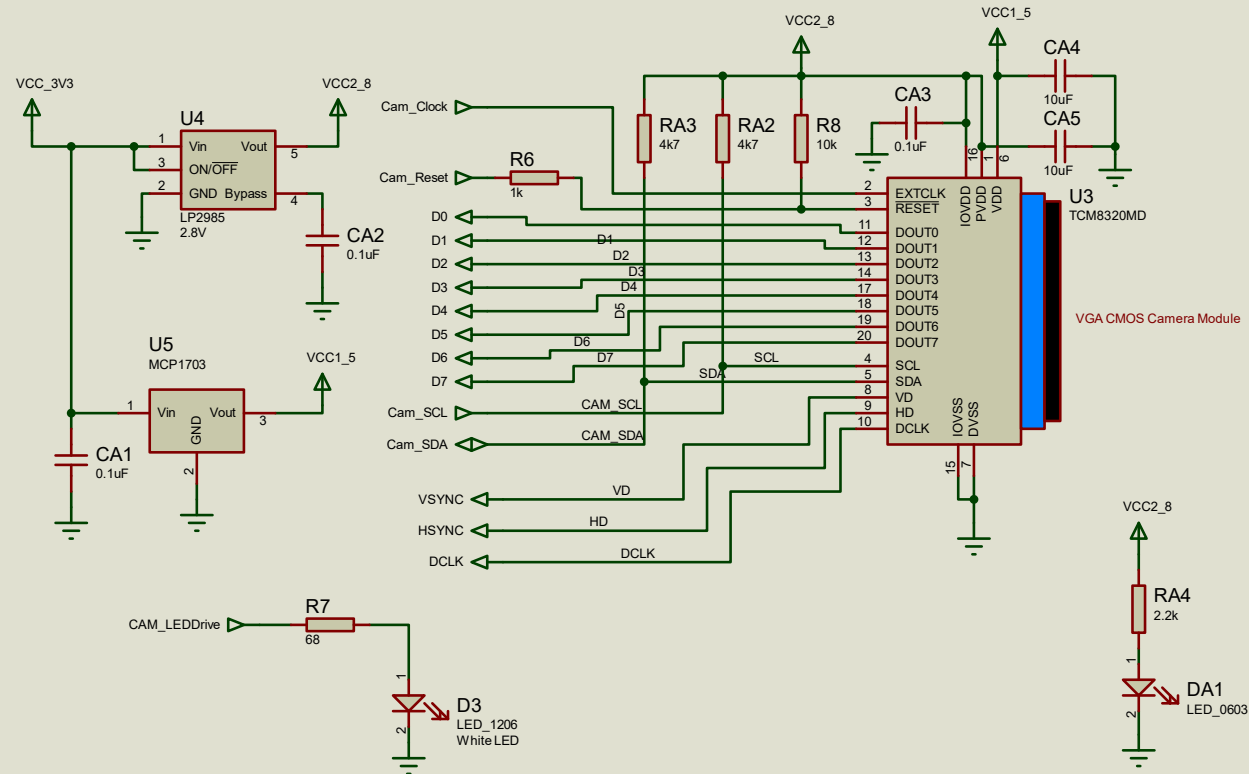
Block Diagram



Schematic 1 – Micro-controller Core



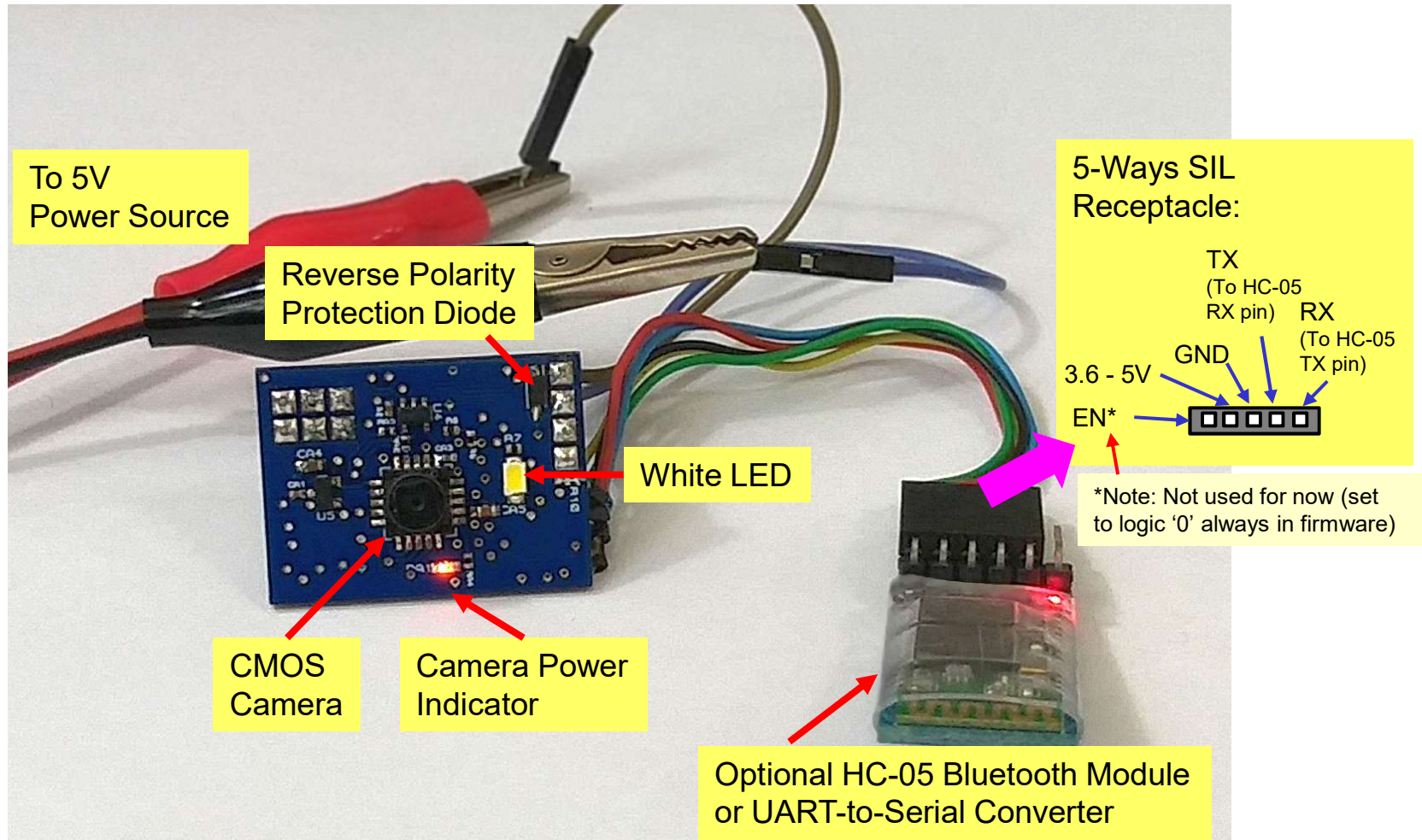
Schematic 2 – Camera Sub-Circuit



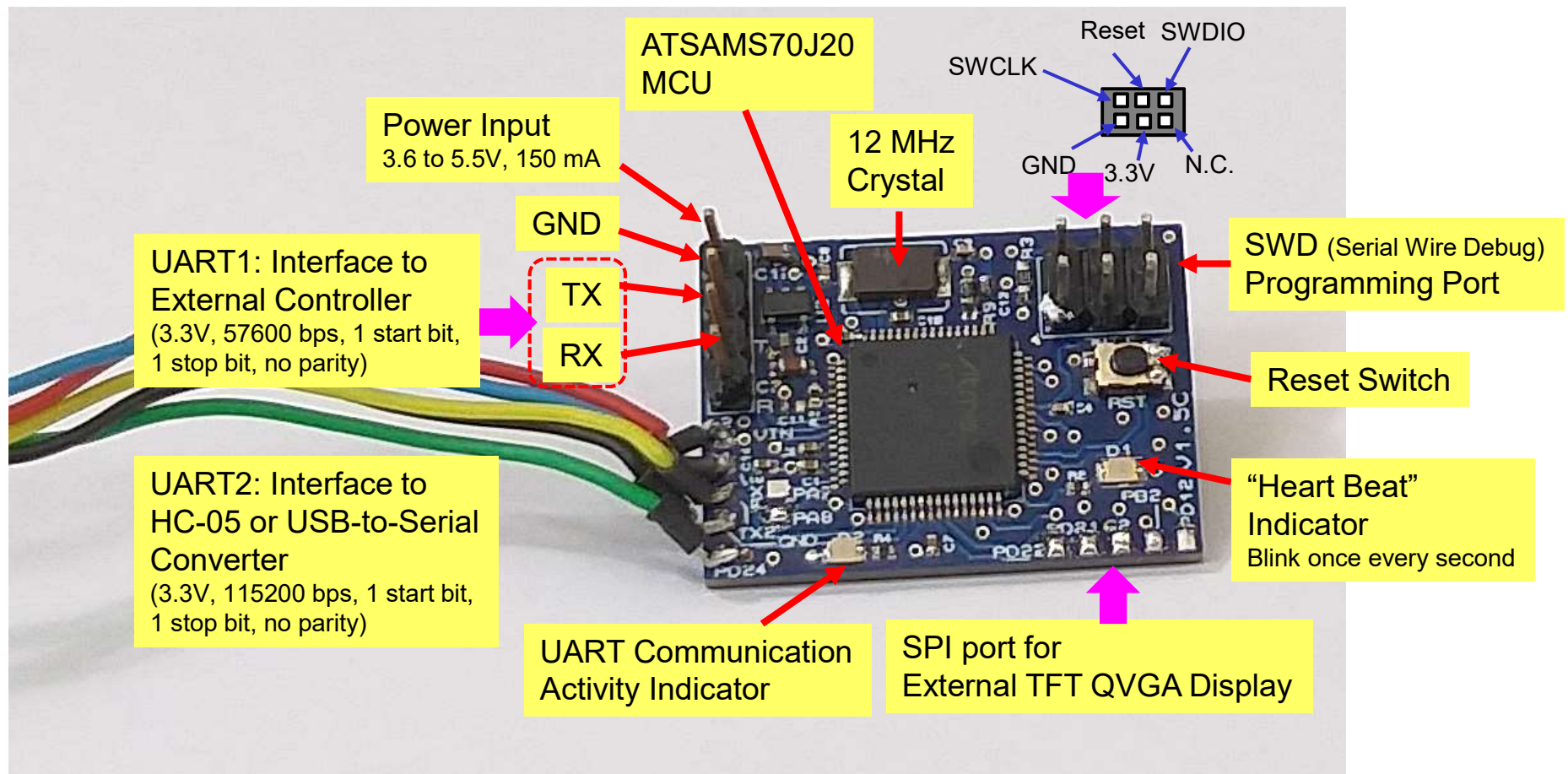
Version 0.92 Sep 2019

Project	Machine Vision 1.5		
Used for			
	Schematic Rev. 0.97		Sheet of 2
Date	16 Aug 2019		Engineer
			FABIAN KUNG

Rear View (MVM V1.5C)



Front View (MVM V1.5C)



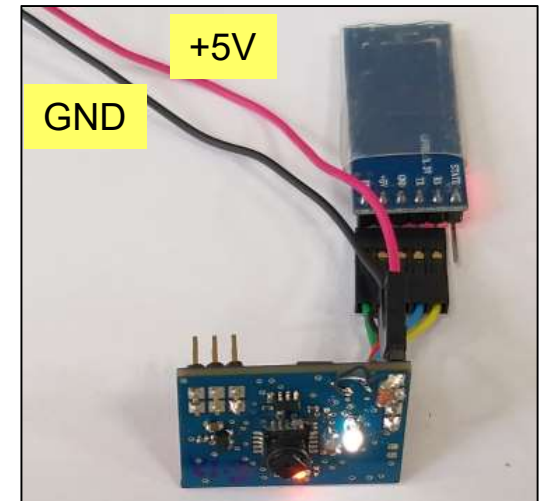
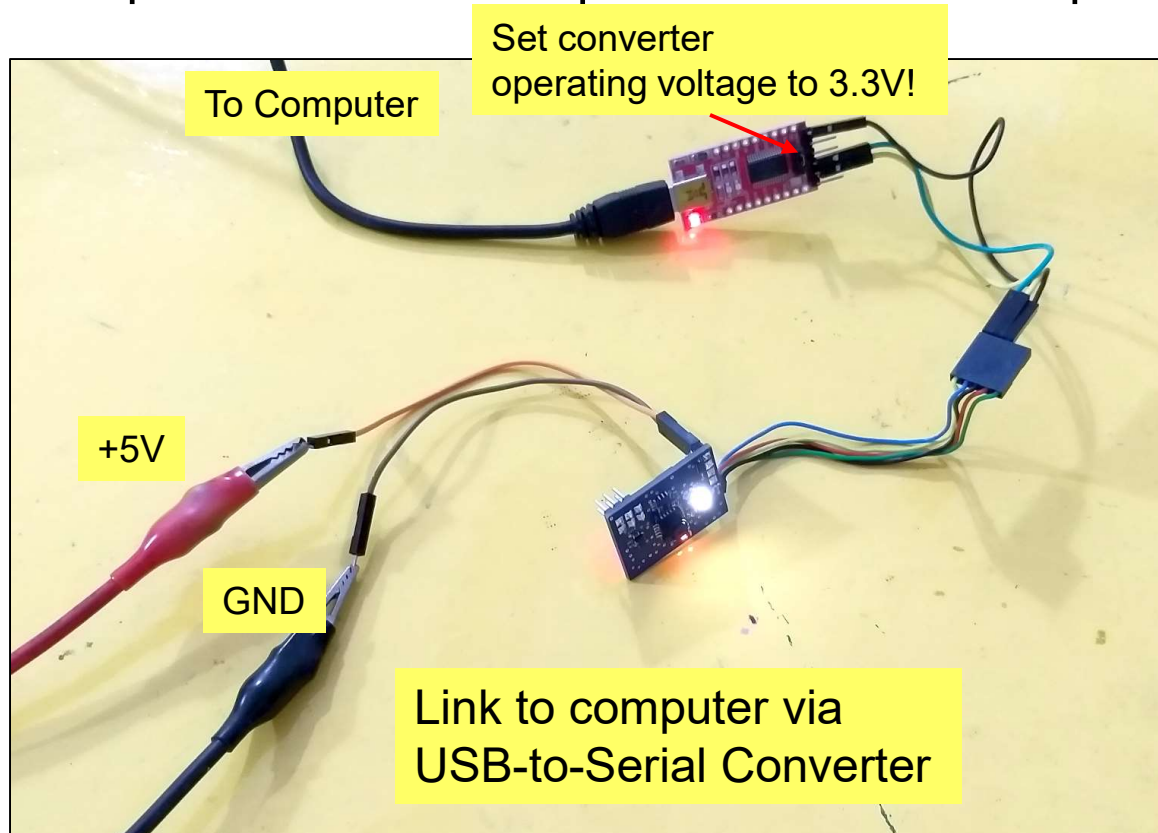
Files

- 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**.

Observe 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.

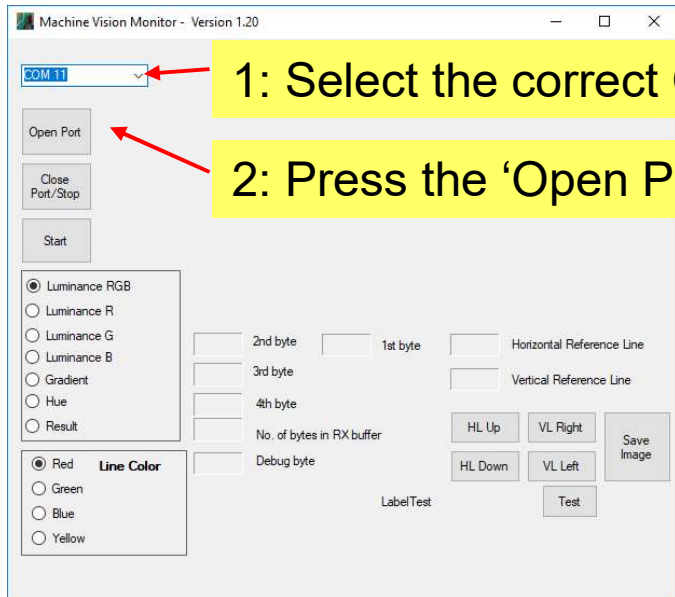


Link to computer via
HC-05 Bluetooth Module

Step 2 – Pair Computer to HC-05

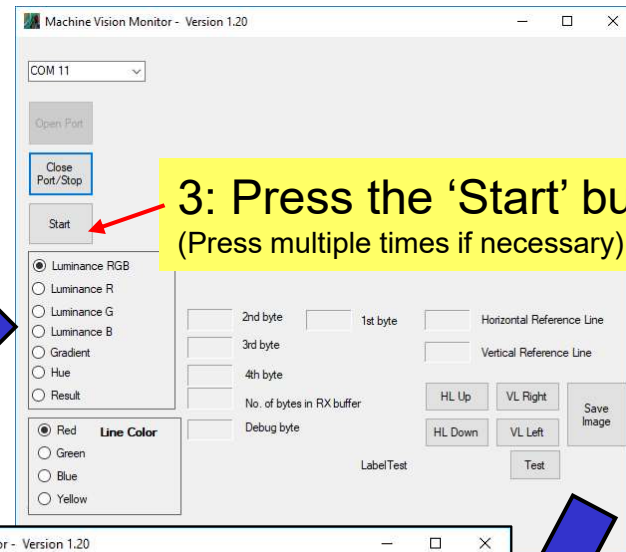
- If need to pair the computer to HC-05.
- Then check virtual COM port number on the computer (for instance by going to the Device Manager).

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

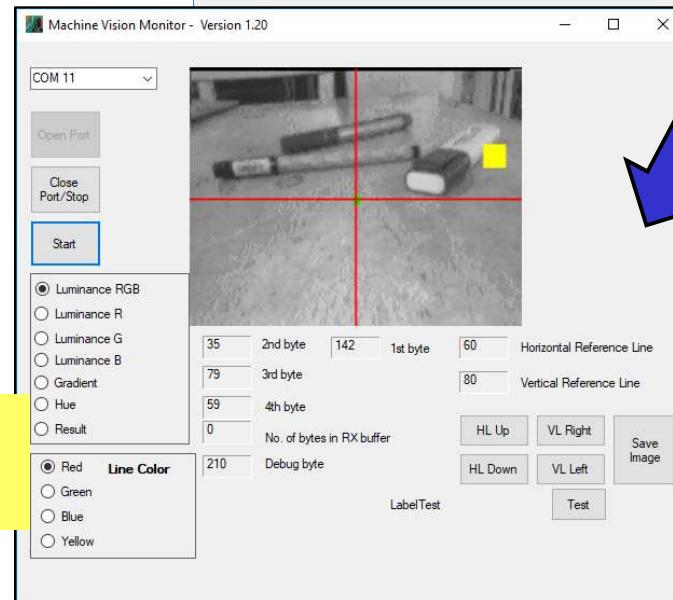
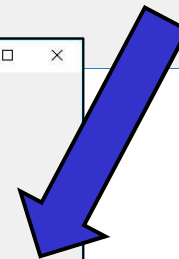


1: Select the correct COM port

2: Press the 'Open Port' button



3: Press the 'Start' button
(Press multiple times if necessary)



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

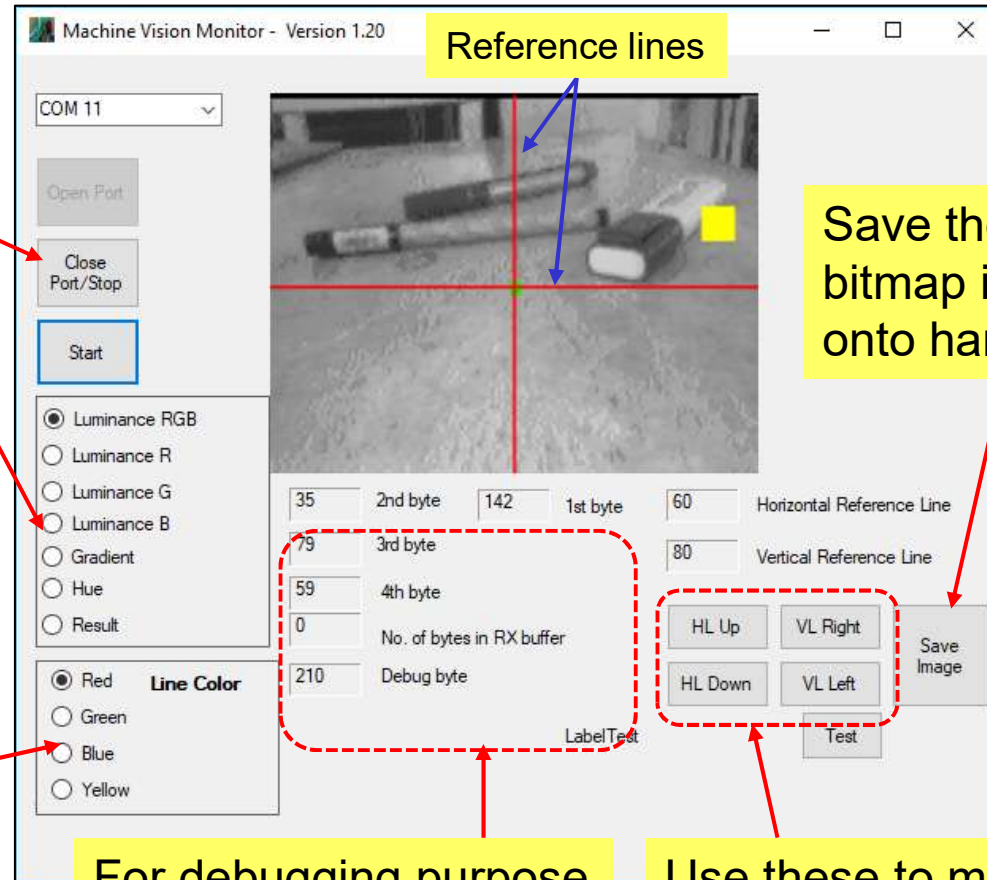
Version 0.92 Sep 2019

Other Information

Always close the COM port before shutting down the software

Select between various display options. All these are processed in the MVM.

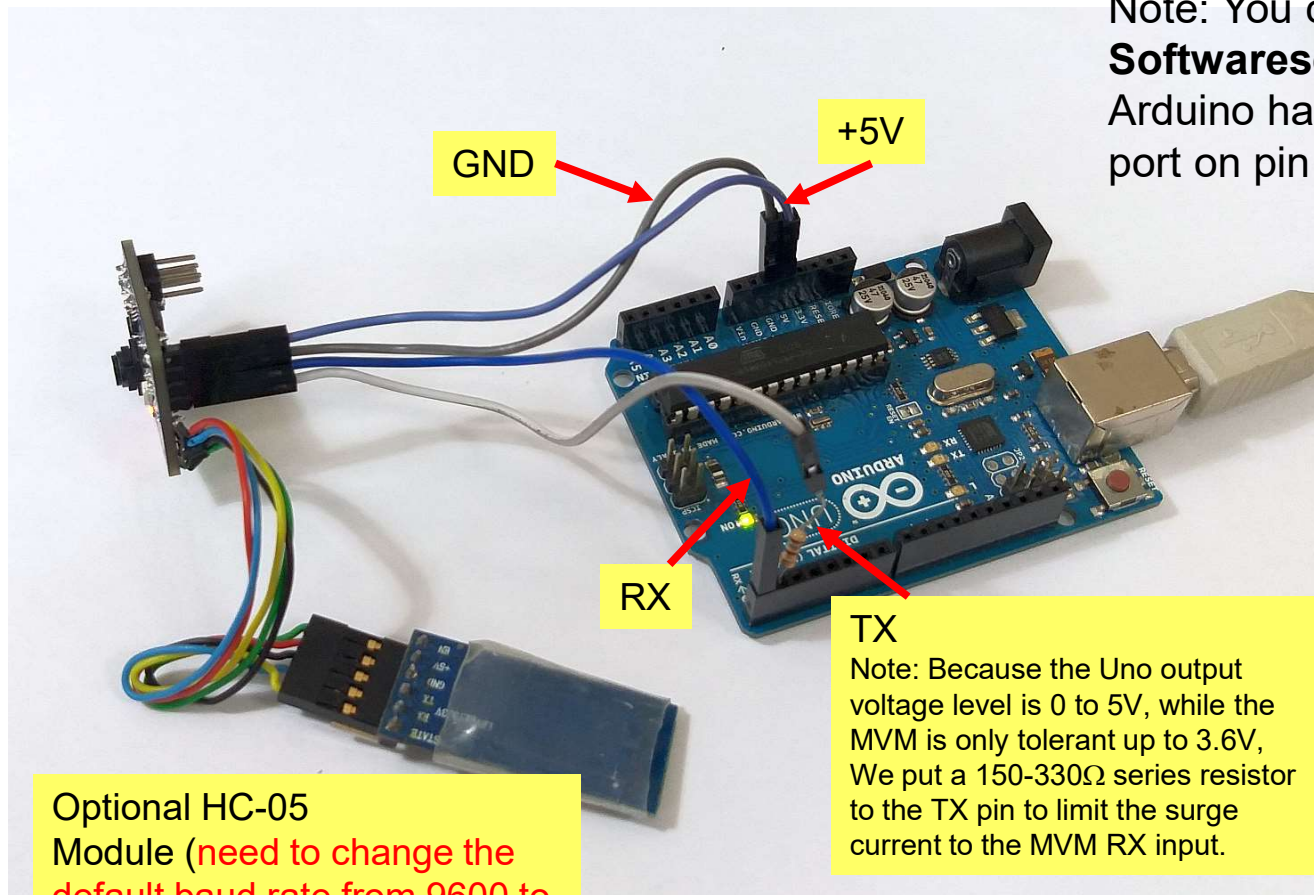
Change reference line color.



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 hardware serial port on pin 0 and 1.

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 ₂ b ₁ b ₀ Byte 3 = 0b00000b ₂ b ₁ b ₀ Byte 4 = 0b00000b ₂ b ₁ b 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

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 1.8.9
File Edit Sketch Tools Help

Example1 $

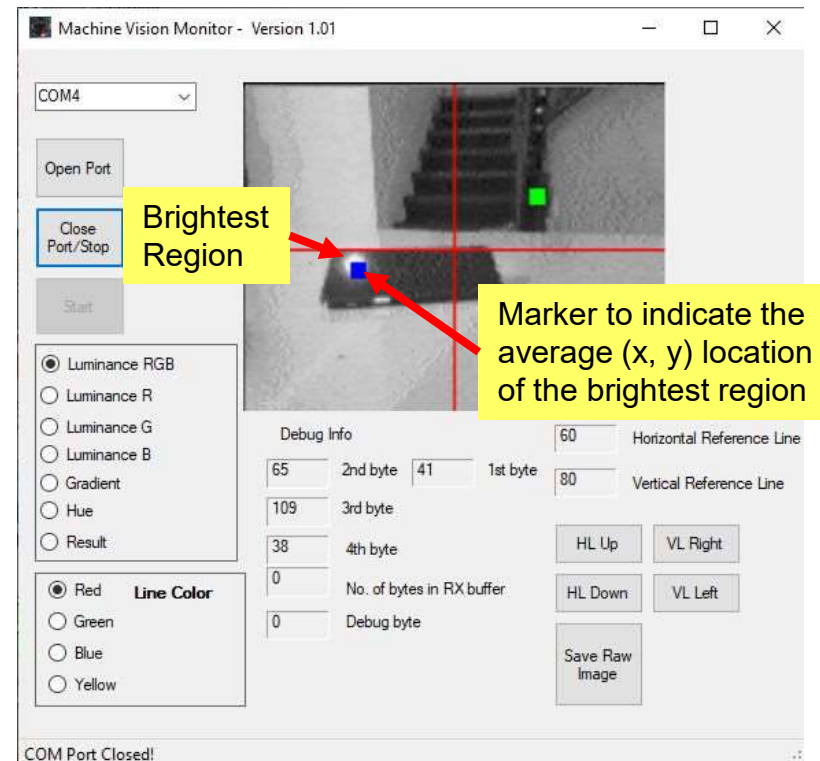
void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
  delay(500); // A 500 ms delay for the module to initialize properly.
  Serial.write(0x10); // Run IPA (image processing algorithm) 1 on interval 1.
  Serial.write(0x10); // Run IPA 1 on interval 2.
}

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

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

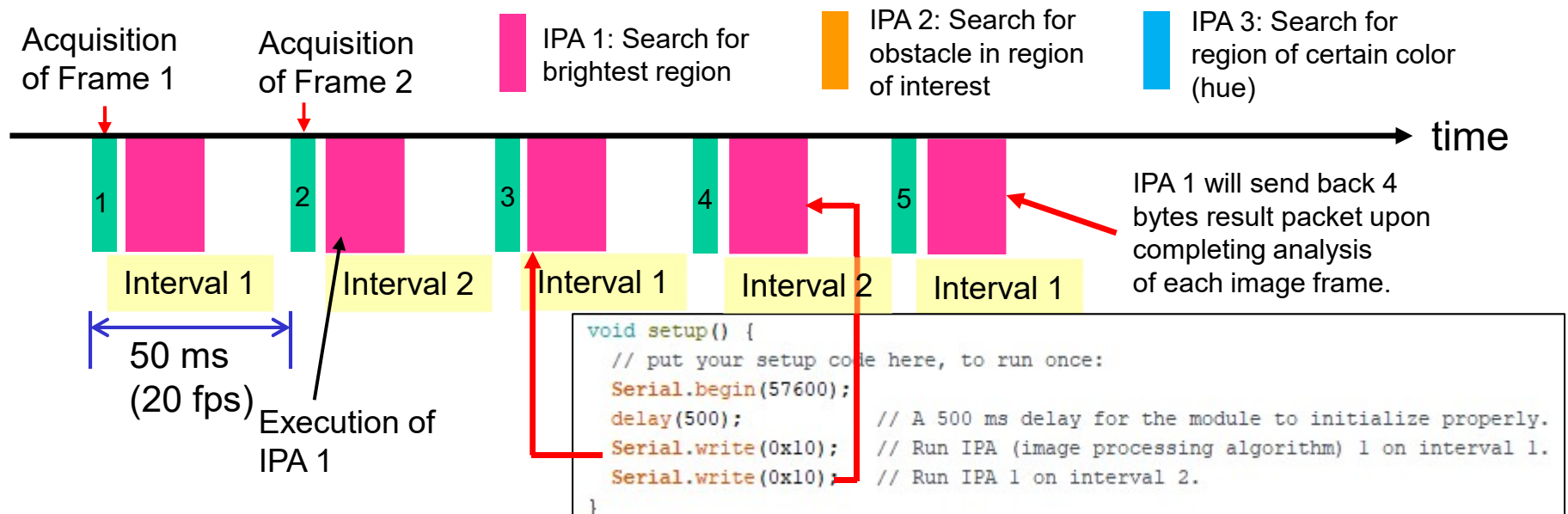
  if (Serial.available() > 3) // Make sure at least 4 bytes in the received buffer
  {
    nID = Serial.read();
    if (nID == 1) // Make sure the data is from IPA 1.
    {
      nLuminance = Serial.read();
      nX = Serial.read();
      nY = Serial.read();

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



Example 1 - More on 'Interval'

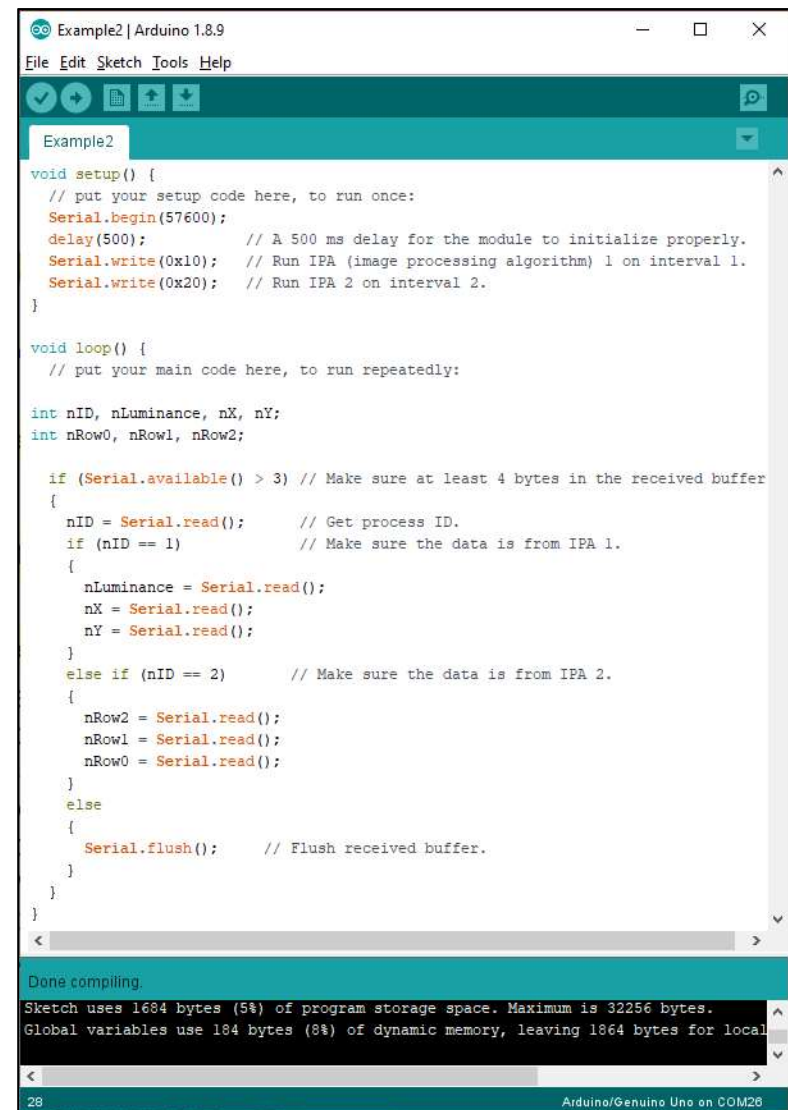
- The firmware of MVM V1.5C assigns odd image frames to *Interval 1* and even image frames to *Interval 2*.
- An image processing algorithm (IPA) can be attached to each interval as shown below and executed after acquisition of a new image frame.



- 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.



```
Example2 | Arduino 1.8.9
File Edit Sketch Tools Help

Example2

void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
  delay(500); // A 500 ms delay for the module to initialize properly.
  Serial.write(0x10); // Run IPA (image processing algorithm) 1 on interval 1.
  Serial.write(0x20); // Run IPA 2 on interval 2.
}

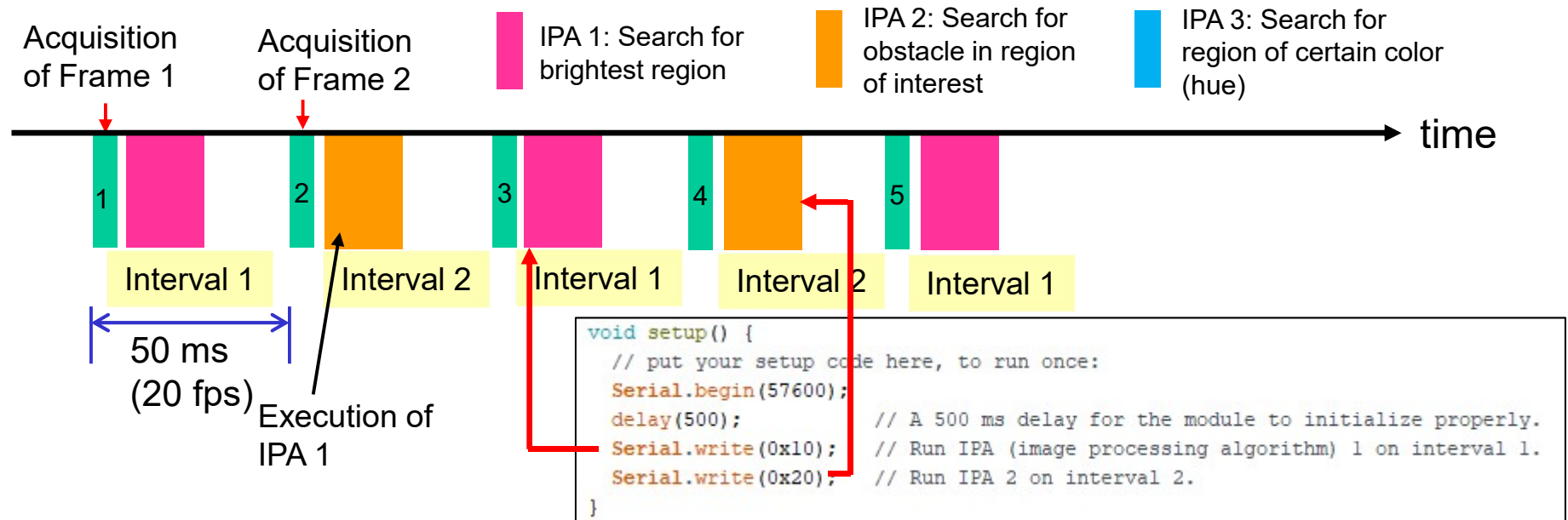
void loop() {
  // put your main code here, to run repeatedly:

  int nID, nLuminance, nX, nY;
  int nRow0, nRow1, nRow2;

  if (Serial.available() > 3) // Make sure at least 4 bytes in the received buffer
  {
    nID = Serial.read(); // Get process ID.
    if (nID == 1) // Make sure the data is from IPA 1.
    {
      nLuminance = Serial.read();
      nX = Serial.read();
      nY = Serial.read();
    }
    else if (nID == 2) // Make sure the data is from IPA 2.
    {
      nRow2 = Serial.read();
      nRow1 = Serial.read();
      nRow0 = Serial.read();
    }
    else
    {
      Serial.flush(); // Flush received buffer.
    }
  }
}

Done compiling
Sketch uses 1684 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 184 bytes (8%) of dynamic memory, leaving 1864 bytes for local
28 Arduino/Genuino Uno on COM28
```

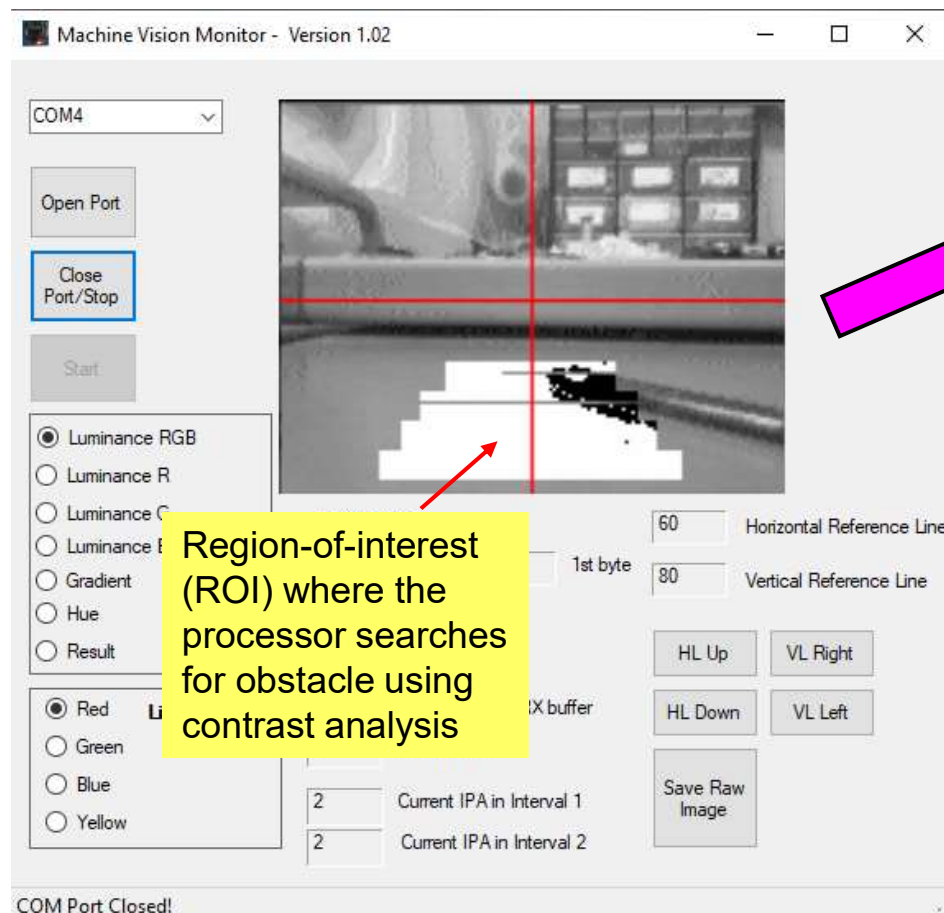
Example 2 – The Assignment of IPAs to Intervals



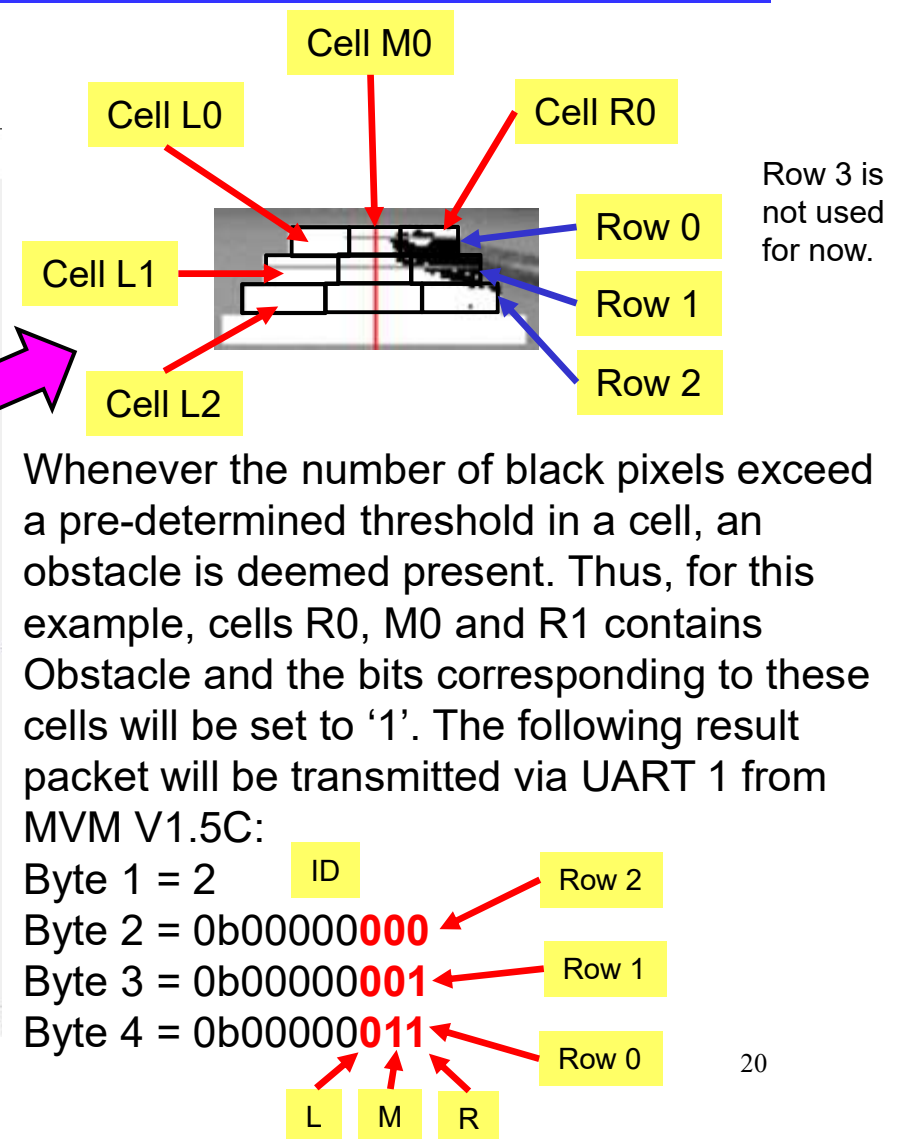
- Each IPA only executes every 100 ms, thus the response time now slows down to 100 ms, however the up side is we get to run two different algorithms simultaneously.

Interpreting the Results of IPA 2

- When IPA 2 is activated:



Version 0.92 Sep 2019



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 sample firmware from https://github.com/fabiankung/MVM_V1_5C

The screenshot shows the GitHub repository 'MVM_V1_5C' with the following files and commit messages:

File	Commit Message	Time
MVM_Arduino_Example/Example1_SoftSerial	Added example codes for Arduino	4 days ago
MVM_Miscellaneous/Scilab	Add files via upload	4 days ago
MVM_Original_Hex_File_R0.53	Delete MVM.hex	4 days ago
MVM_PC_Monitor_Software	Add files via upload	4 days ago
MVM_Sample_Firmware_R0.9	Minor update on PD24 pin	3 days ago
LICENSE	Create LICENSE	4 days ago
MVM_V1_5C_Quick_Start.pdf	Update the quick start guide	4 days ago
README.md	Create README.md	4 days ago

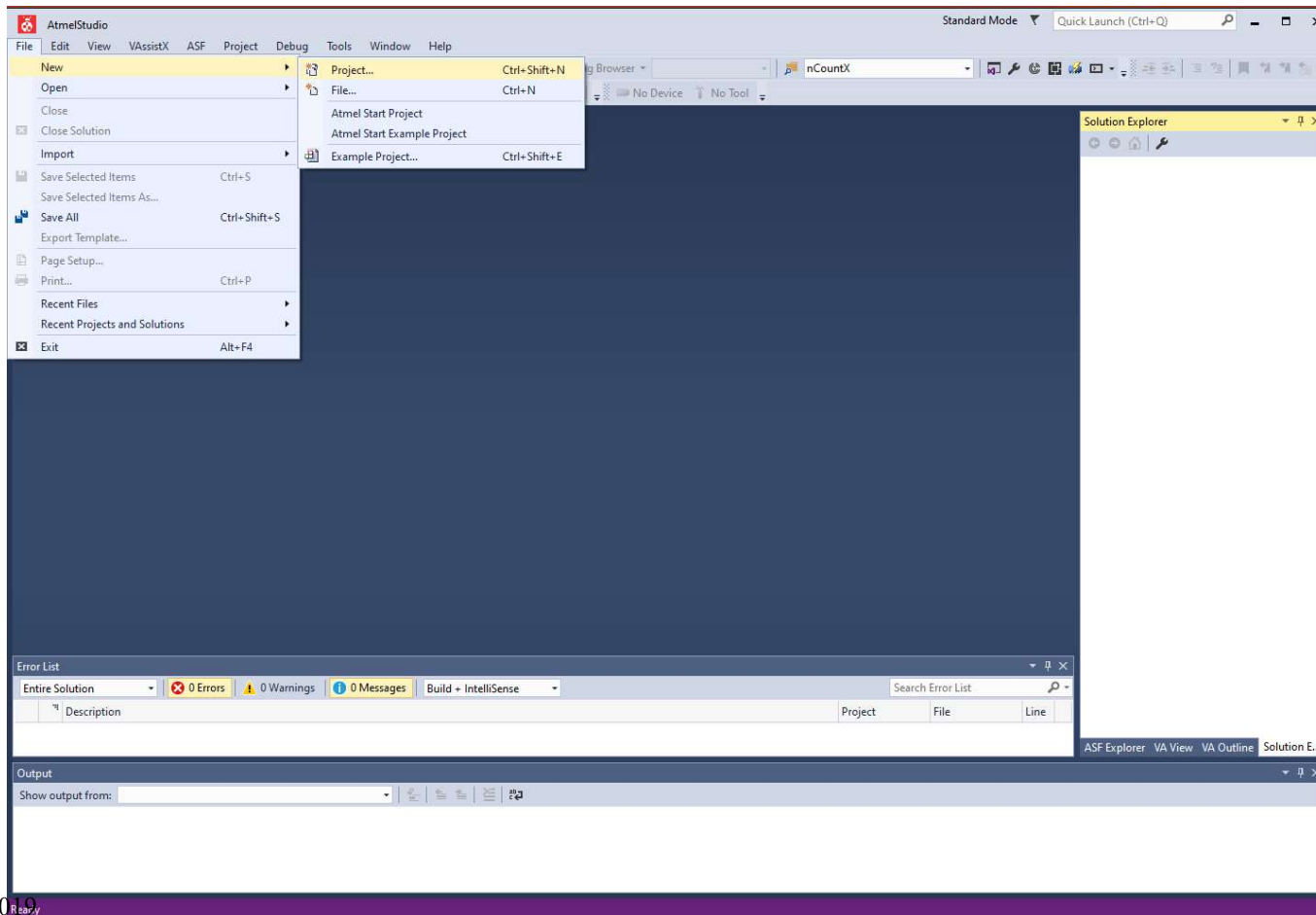
Annotations:

- A pink arrow points from the URL to the repository name.
- A yellow box labeled 'Original firmware in hex format' points to 'MVM_Original_Hex_File_R0.53'.
- A yellow box labeled 'PC Application and Source codes for MV Monitor software' points to 'MVM_PC_Monitor_Software'.
- A yellow box labeled 'Sample firmware in C for MVM V1.5C' points to 'MVM_Sample_Firmware_R0.9'.

- “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.

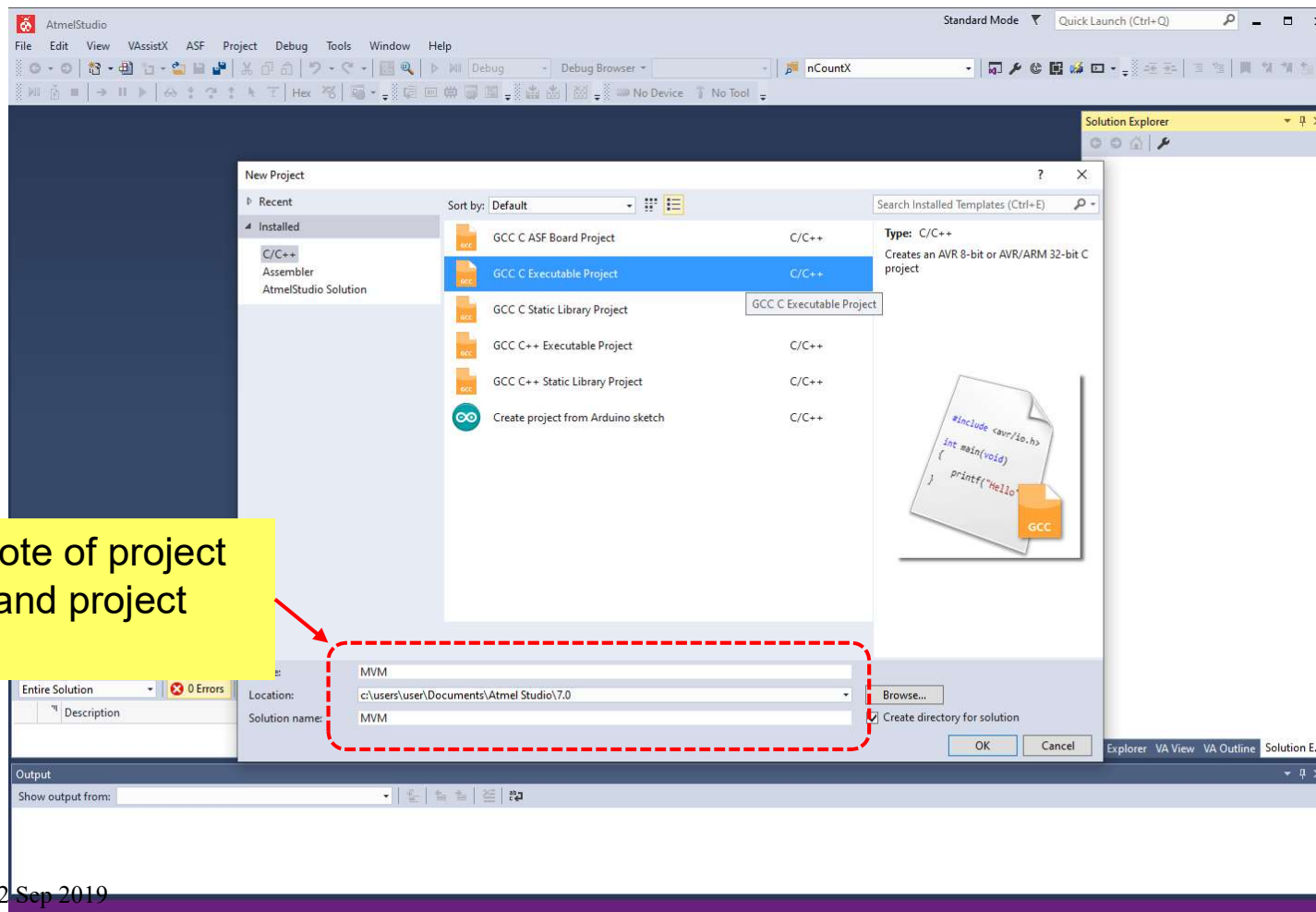
Setting Up An Atmel Studio Project 1

- Start a new project in Atmel Studio 7.



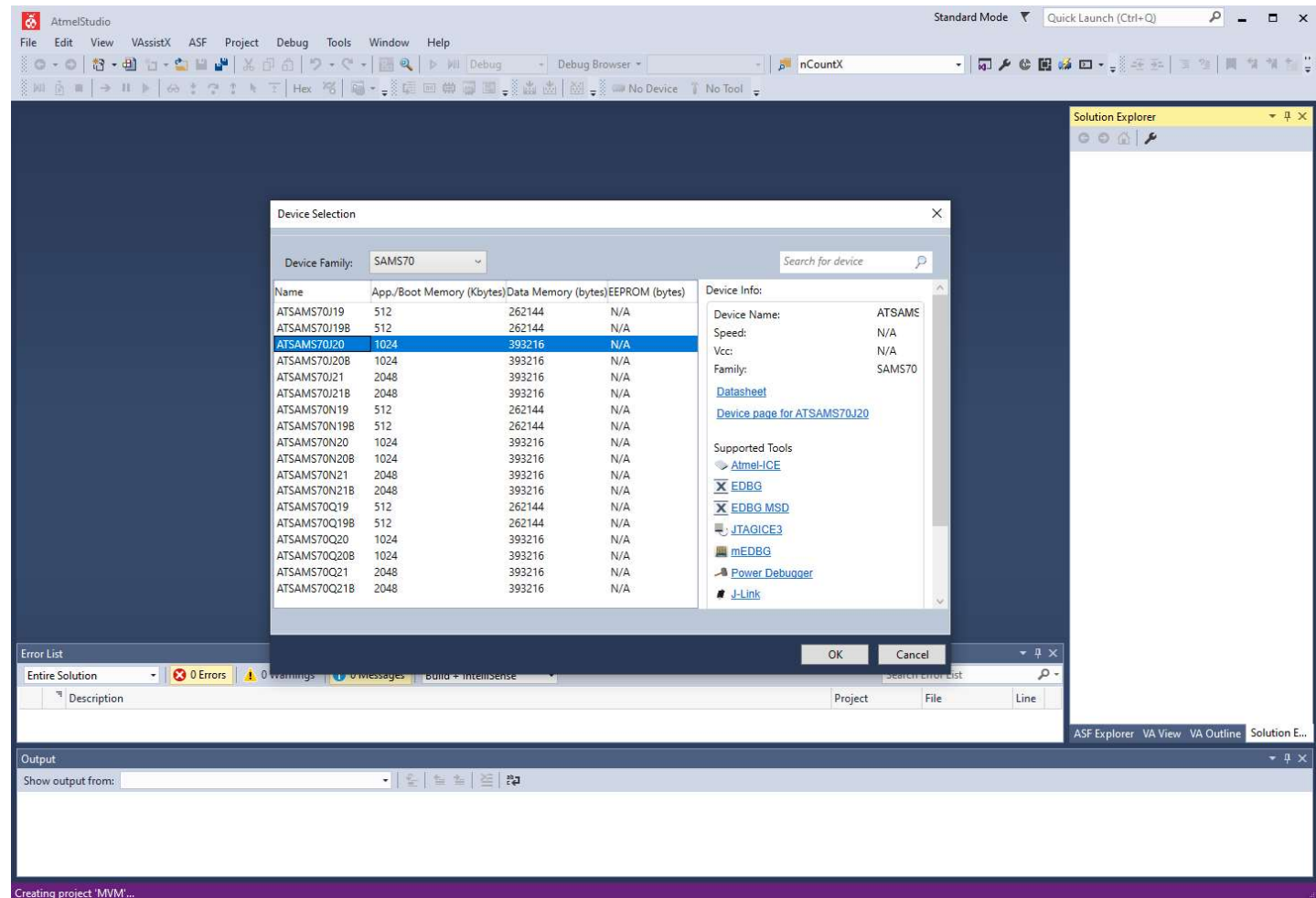
Setting Up An Atmel Studio Project 2

- Create a GCC C executable project.



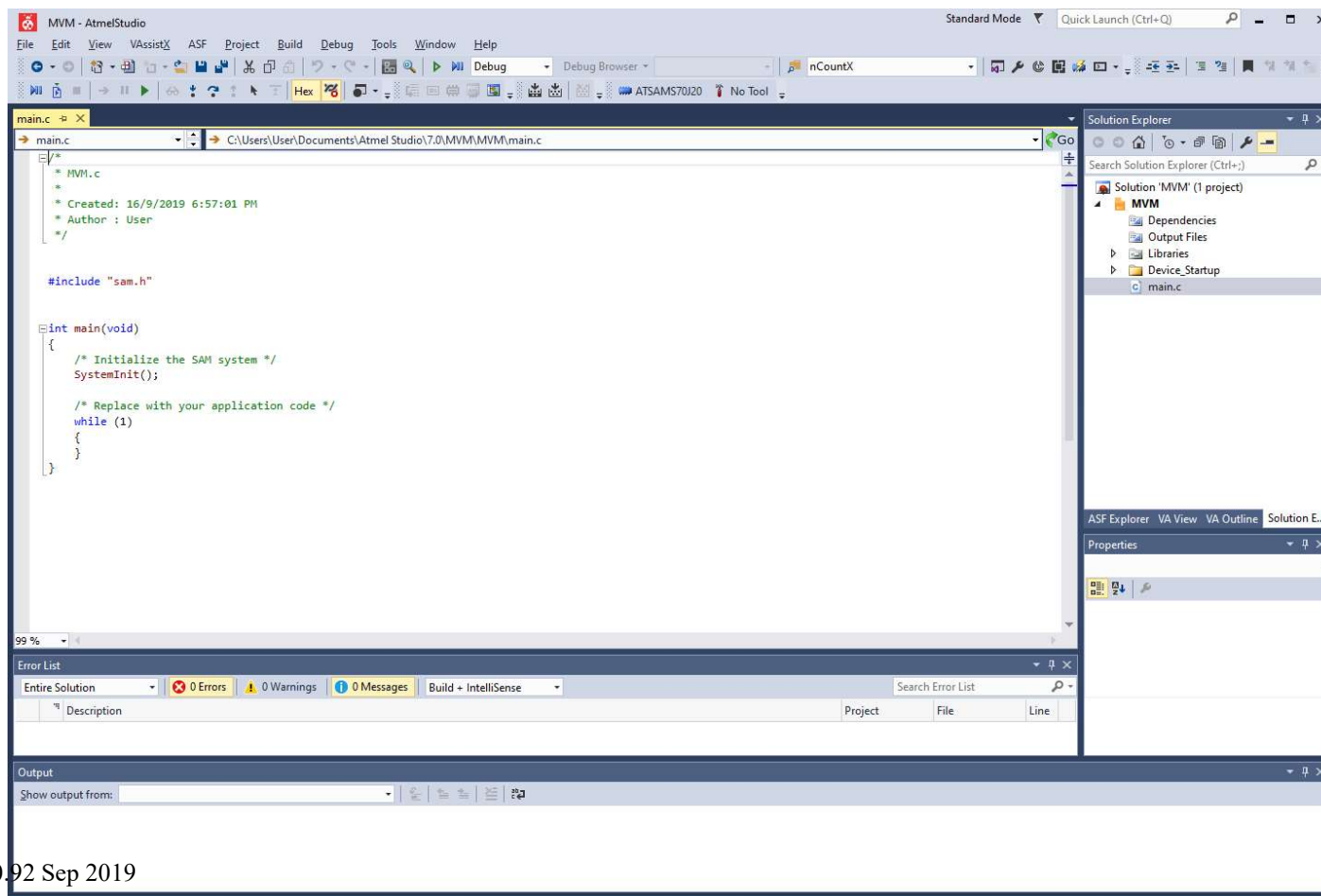
Setting Up An Atmel Studio Project 3

- Select the correct device.



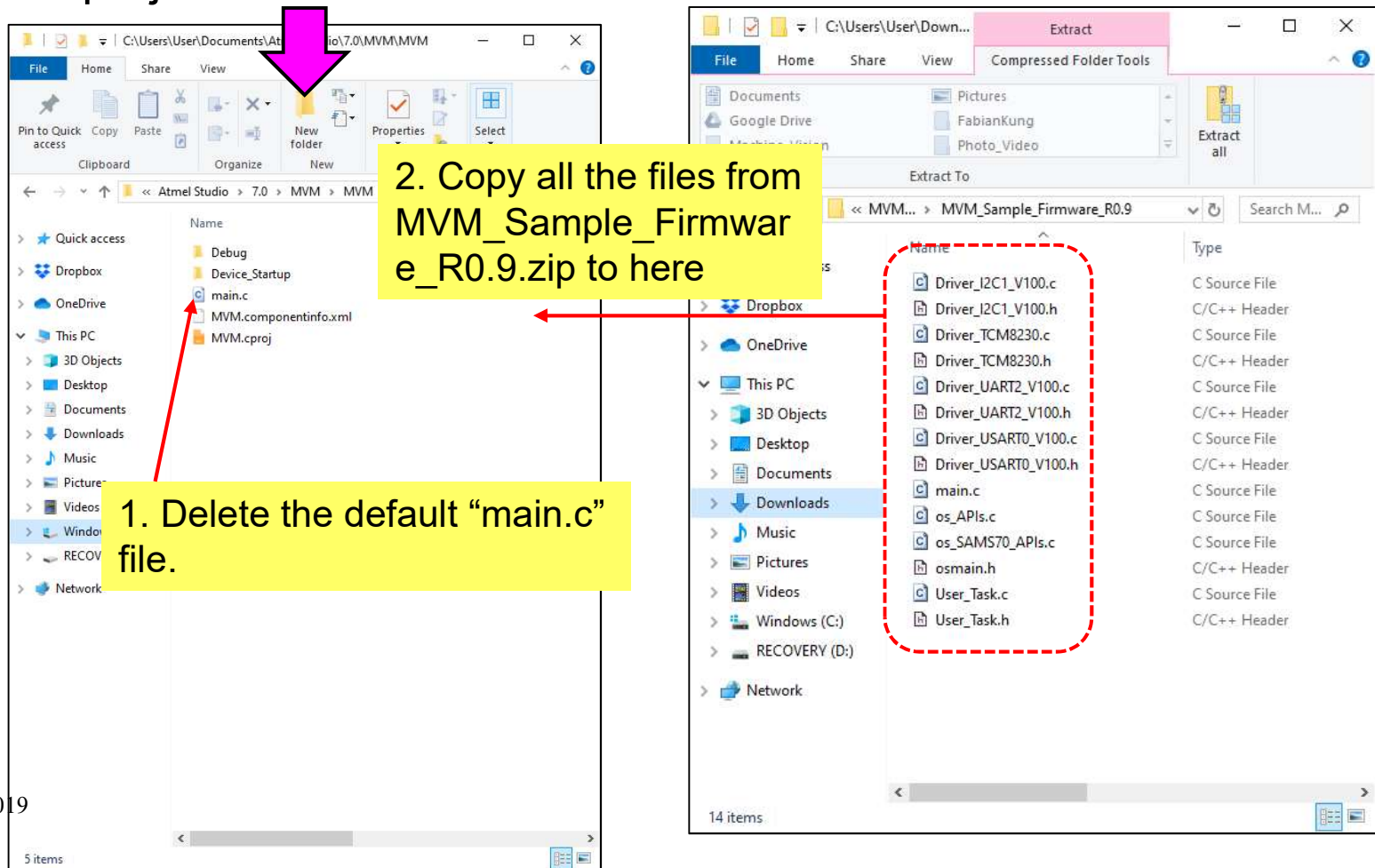
Setting Up An Atmel Studio Project 4

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



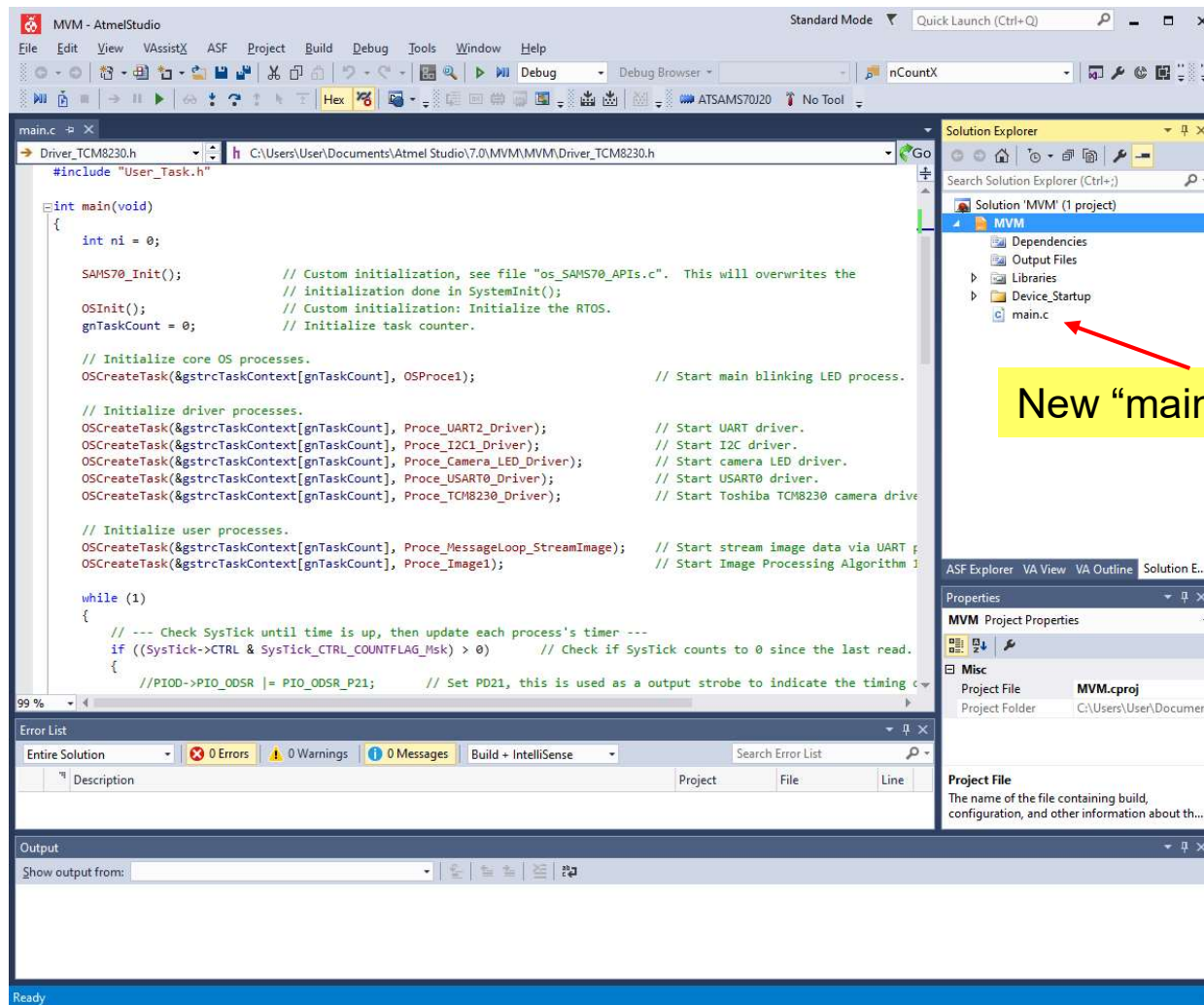
Setting Up An Atmel Studio Project 5

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



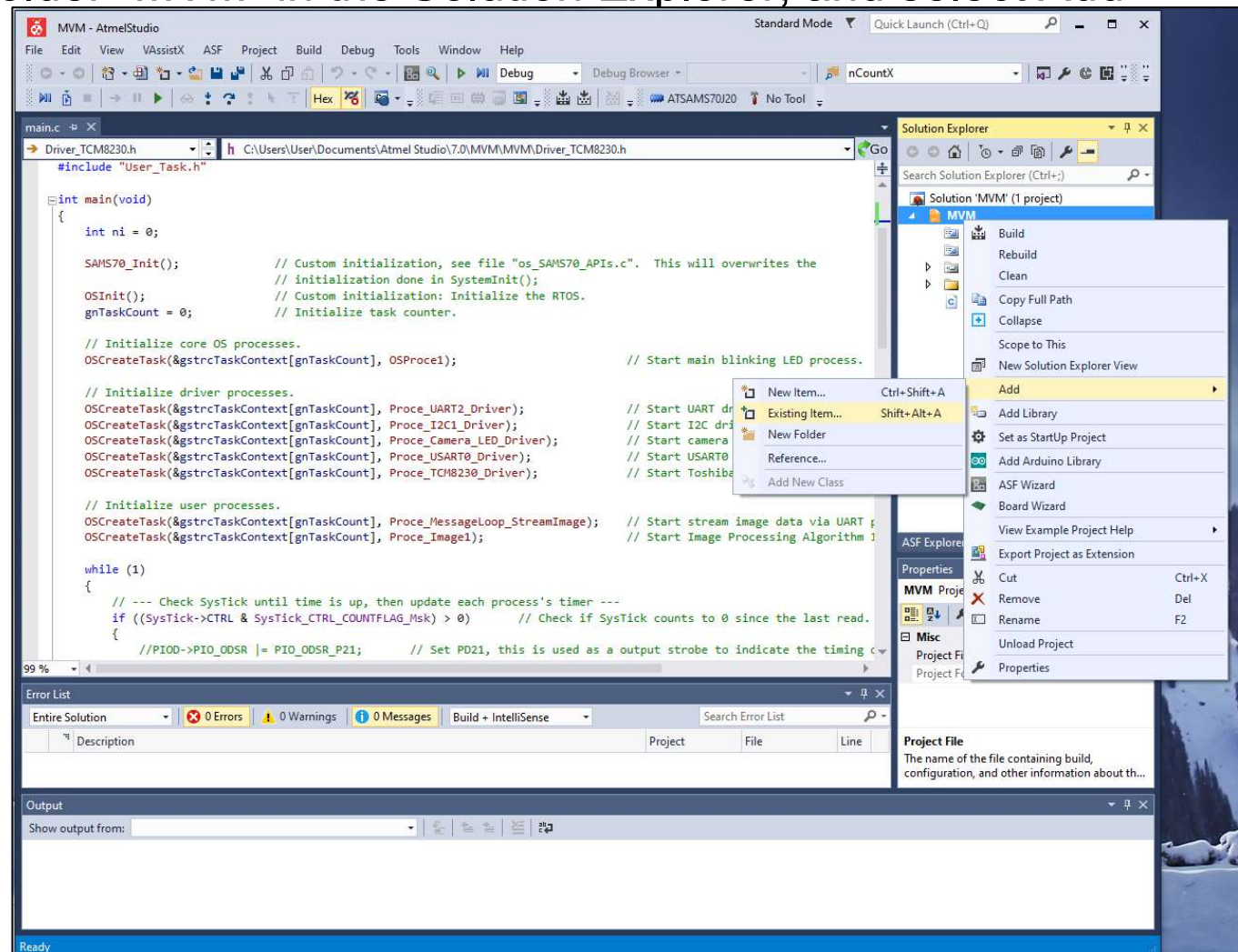
Setting Up An Atmel Studio Project 6

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



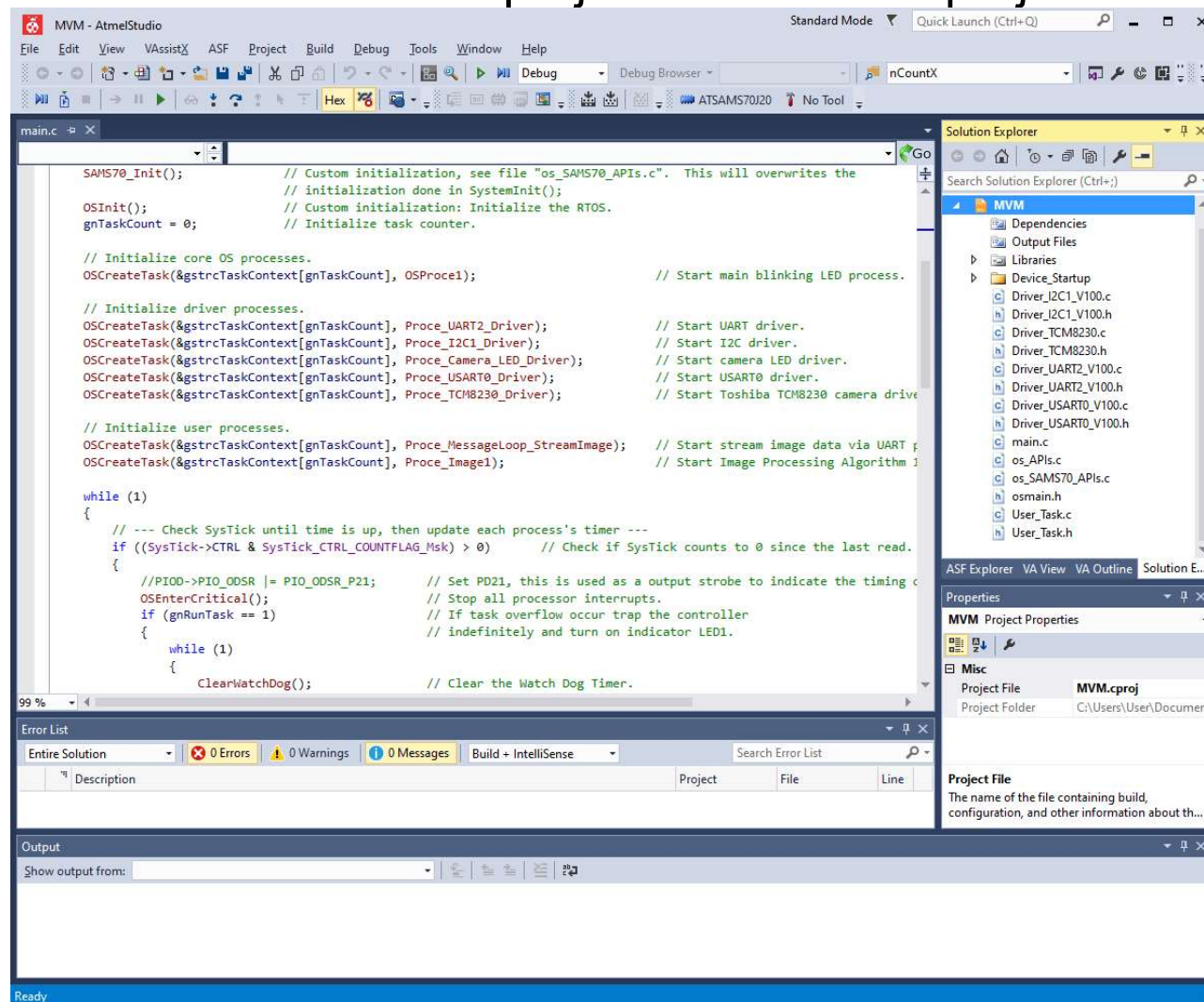
Setting Up An Atmel Studio Project 7

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



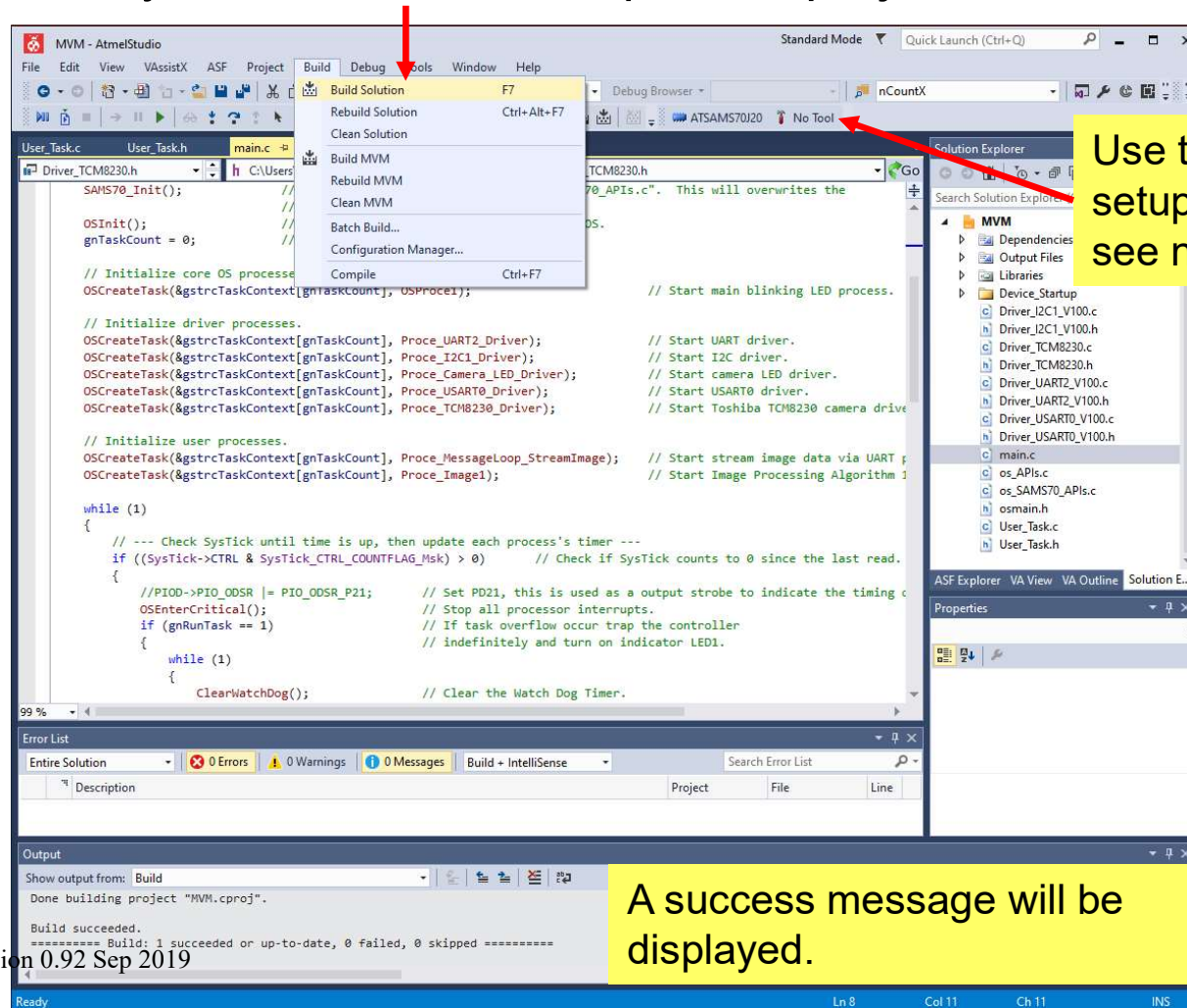
Setting Up An Atmel Studio Project 8

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



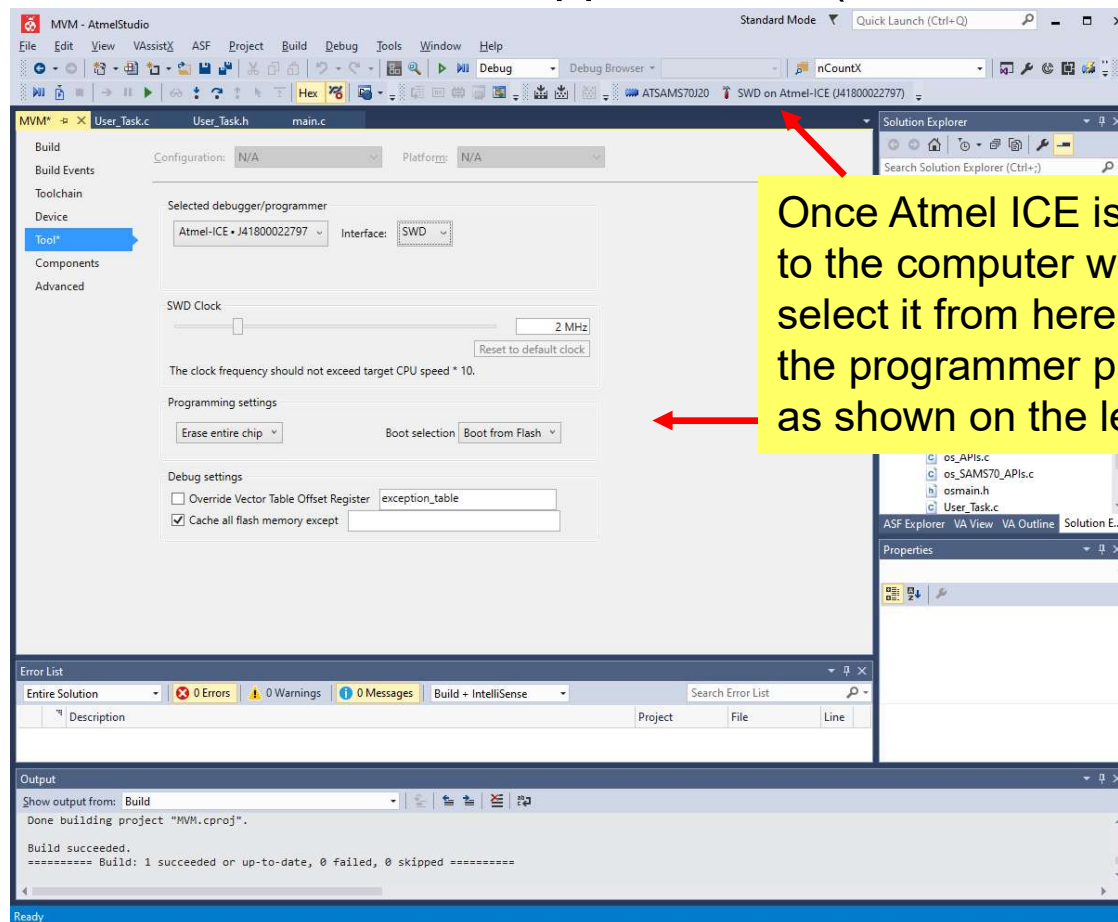
Setting Up An Atmel Studio Project 9

- Now you can build or compile the project.



Setting Up the Programming Tool – Atmel ICE

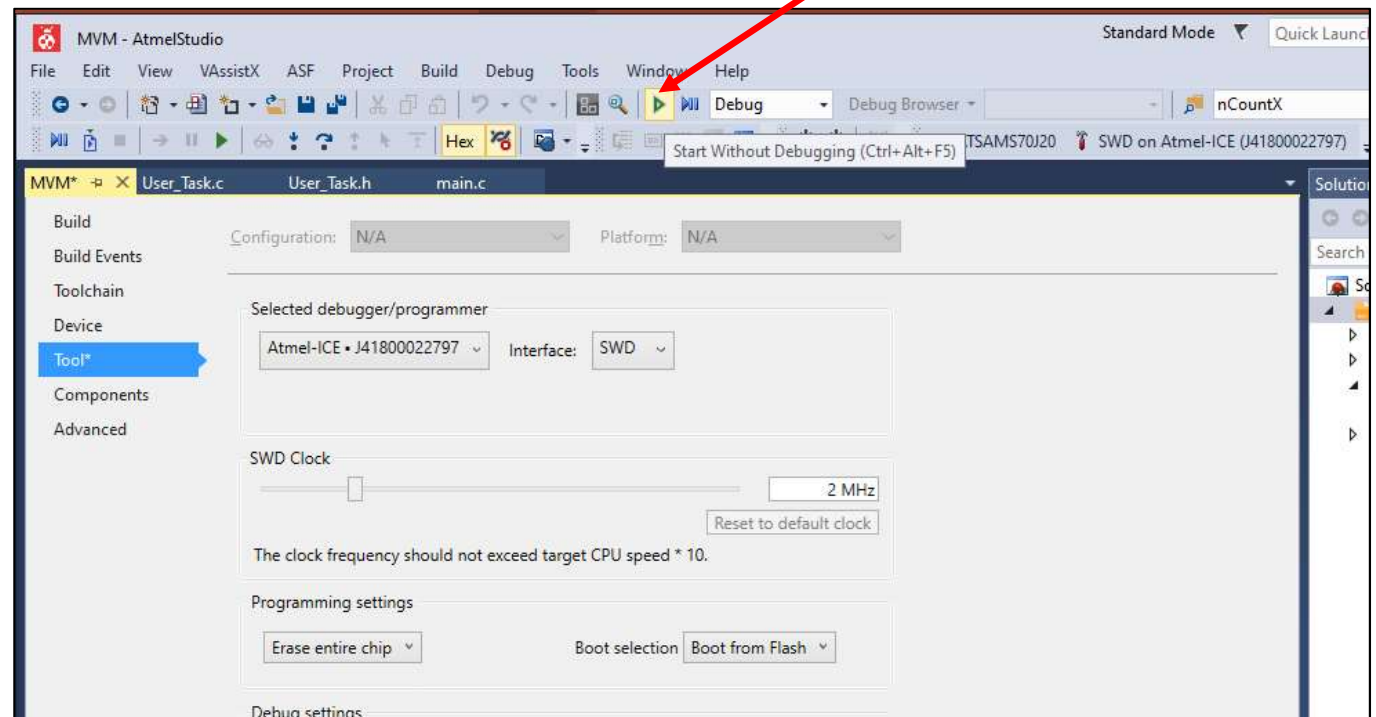
- Now you can load the firmware into the micro-controller with a suitable programmer. Here we are using Atmel ICE, but any programmer compatible with Atmel Studio 7 and support SWD (serial wire debug) mode is fine.



Once Atmel ICE is connected to the computer we can select it from here, and set the programmer parameters as shown on the left.

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.

The screenshot shows the Atmel Studio interface with the 'Device Programming' window open. The 'GPNVM Bits' section is highlighted, showing the following settings:

GPNVM Bit	Value
GPNVMBITS.BOOT_MODE	<input checked="" type="checkbox"/>
GPNVMBITS.TCM_CONFIGURATION	0x01

A red arrow points to the 'Program' button in the bottom right corner of the 'Device Programming' window.

The Heartbeat LED of the MVM V1.5C should start blinking once all the GPNVM bits are programmed.

Hit the 'Program' button once the parameters are properly setup.

Version 0.92 Sep 2019

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 scheduler and basic structure of the C codes for ARM Cortex-M see <https://fkeng.blogspot.com/2016/02/atmel-arm-cortex-m4-microcontroller.html>

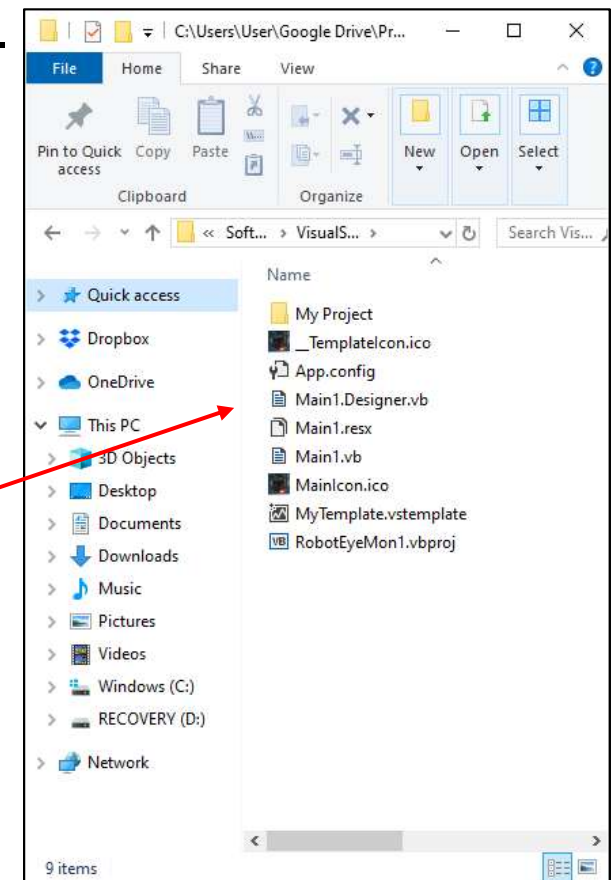
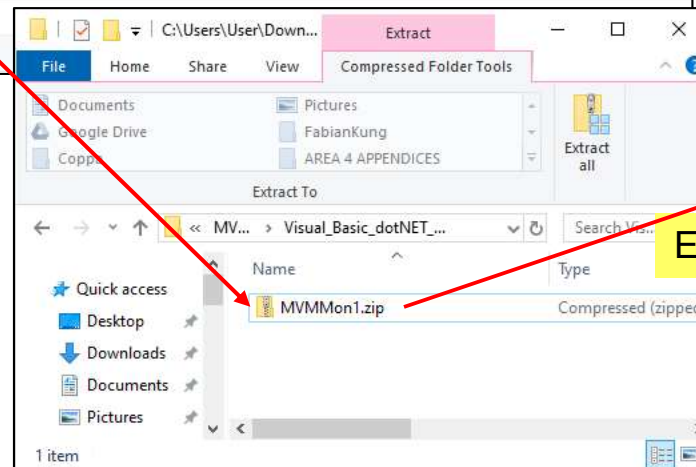
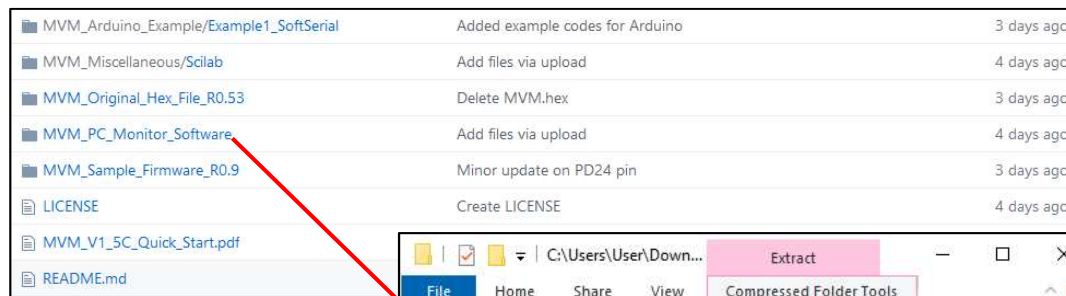
Compiling and Building the Machine Vision Monitor Software

Introduction

- The PC application (*.exe) to observe the image frames captured by the MVM and the corresponding source codes are 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.

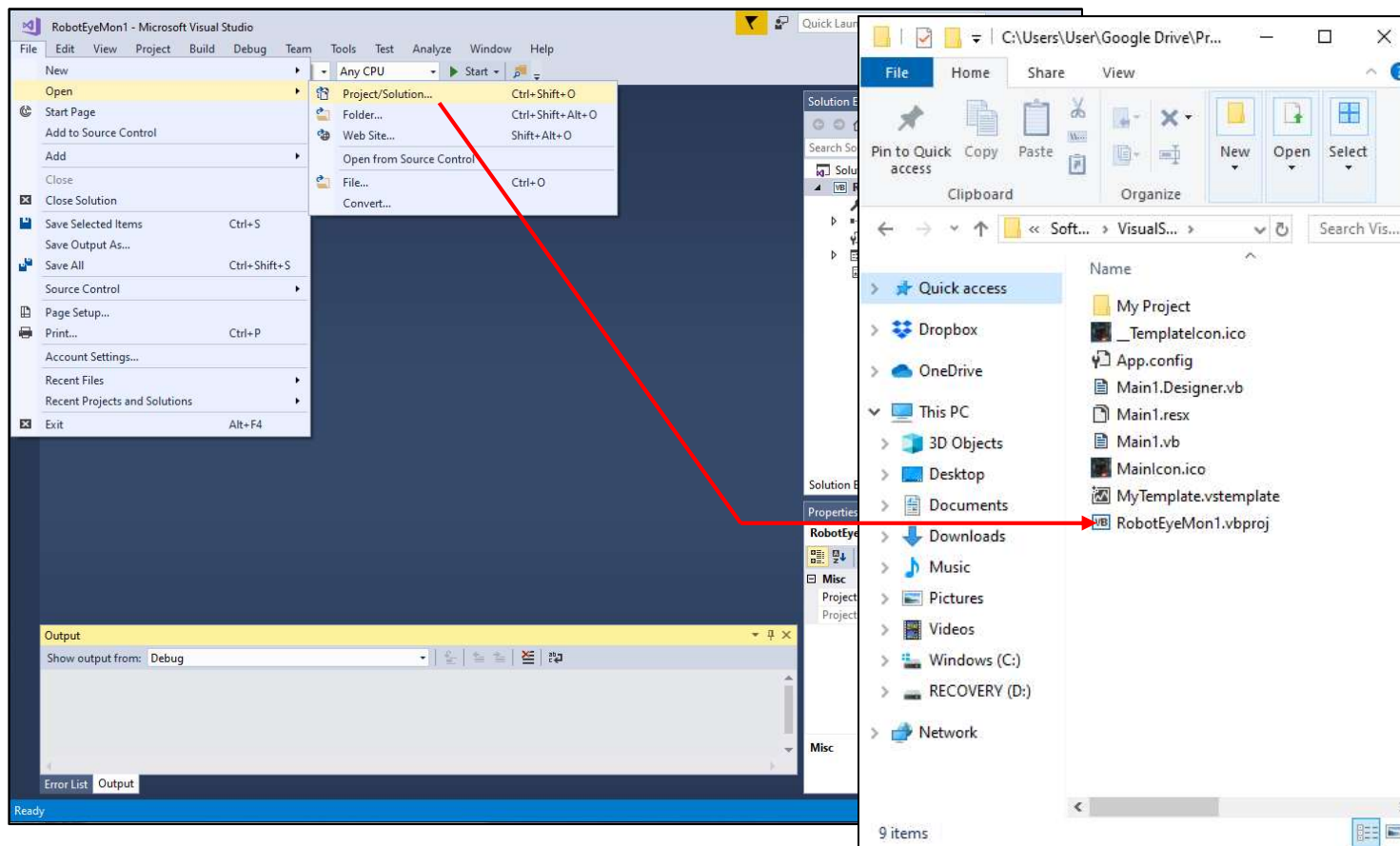
Setting Up Visual Studio Project 1

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



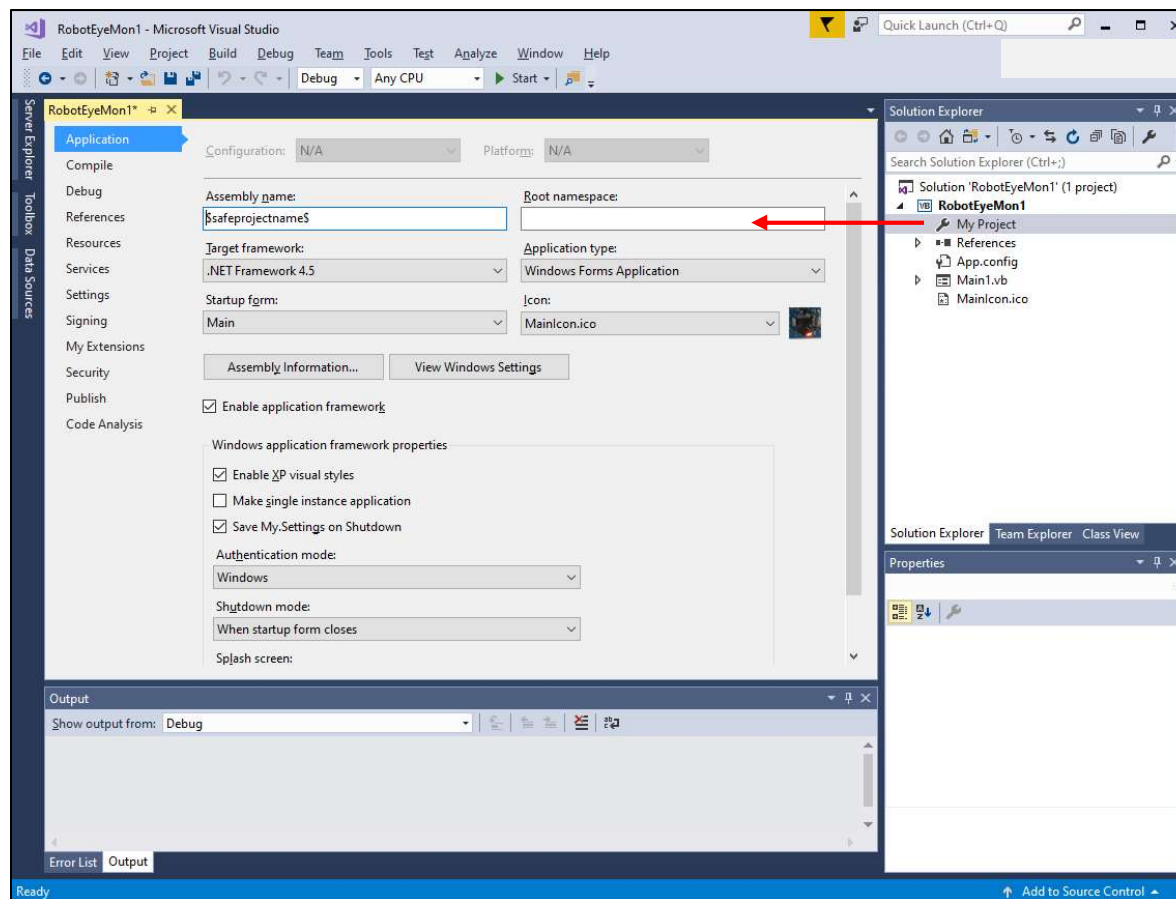
Setting Up Visual Studio Project 2

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



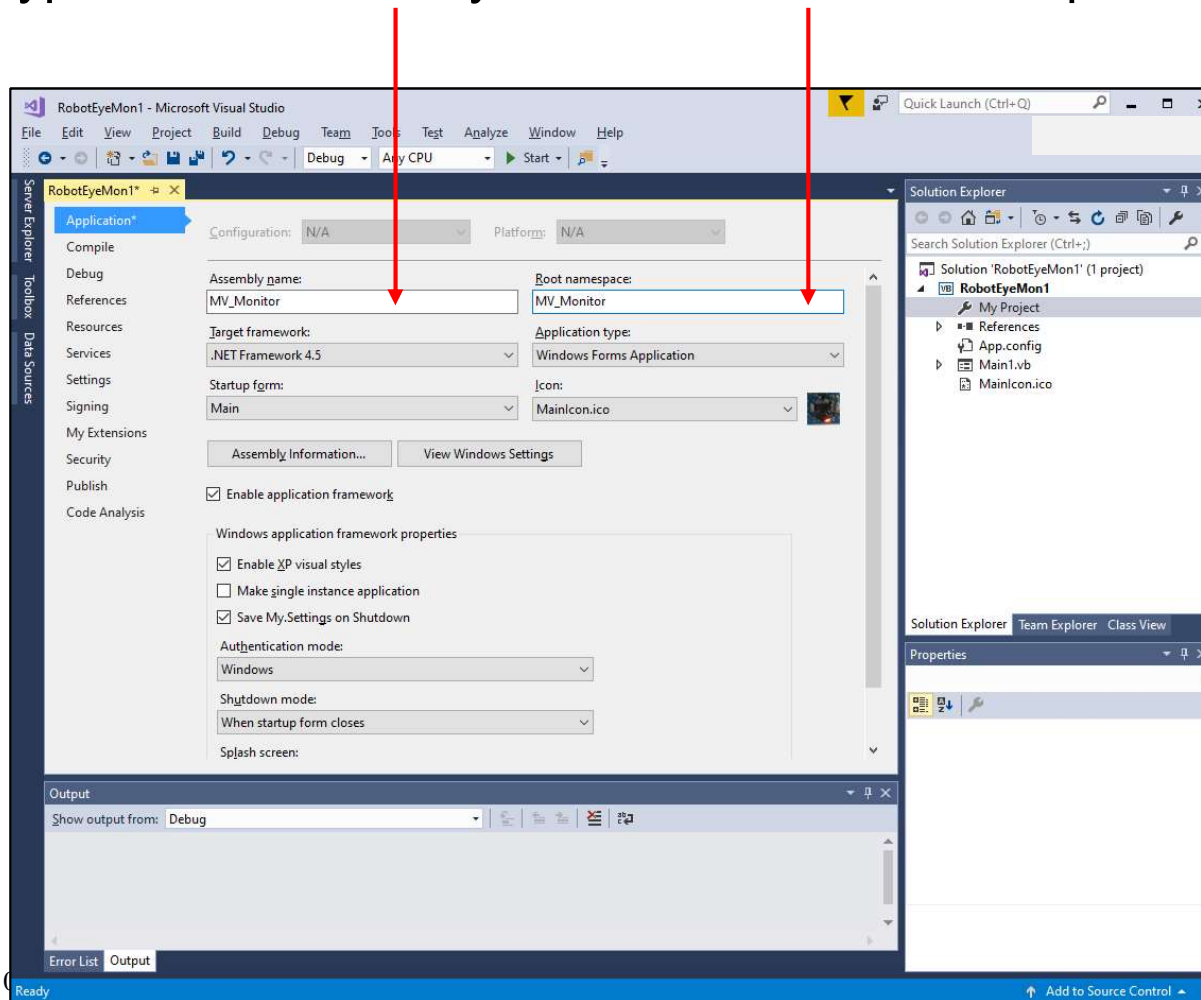
Setting Up Visual Studio Project 3

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



Setting Up Visual Studio Project 4

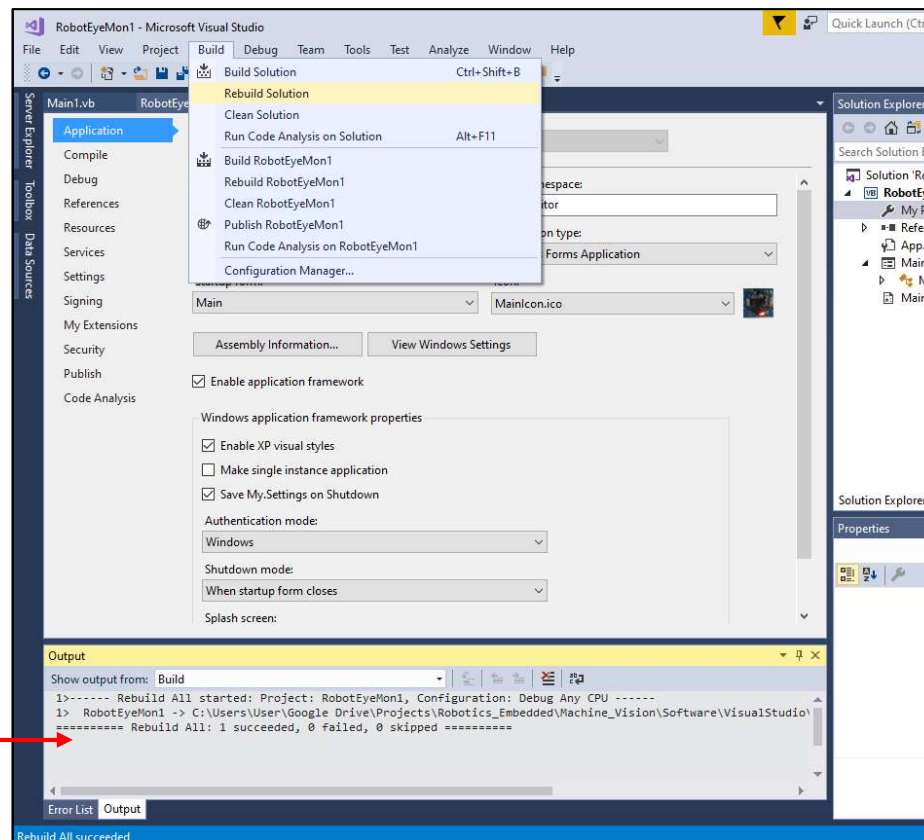
- Type in the Assembly Name and Root Namespace as shown.



Setting Up Visual Studio Project 5

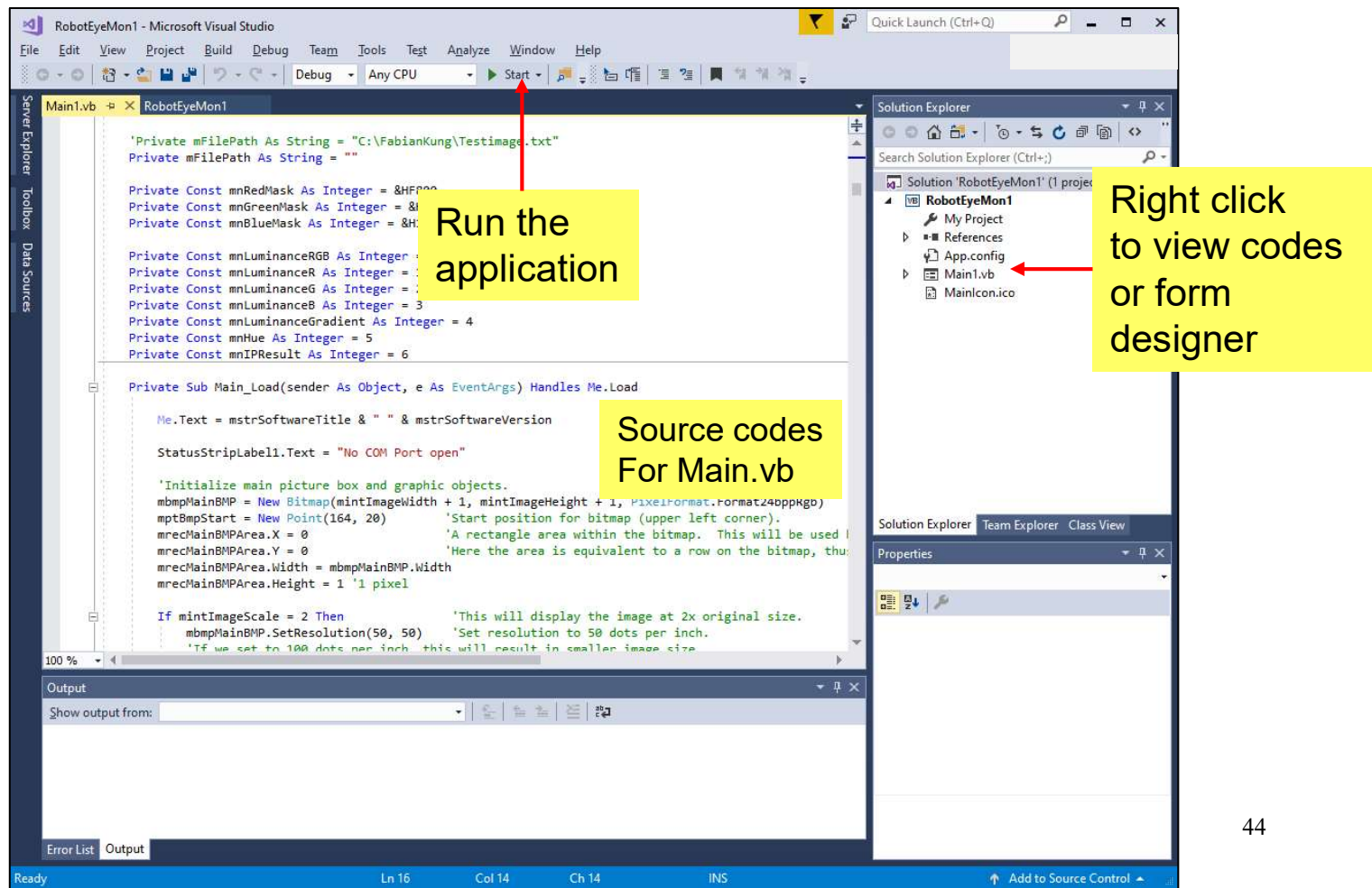
- Now rebuild the project as shown and you should get a success message in the Output window.

Success
Message



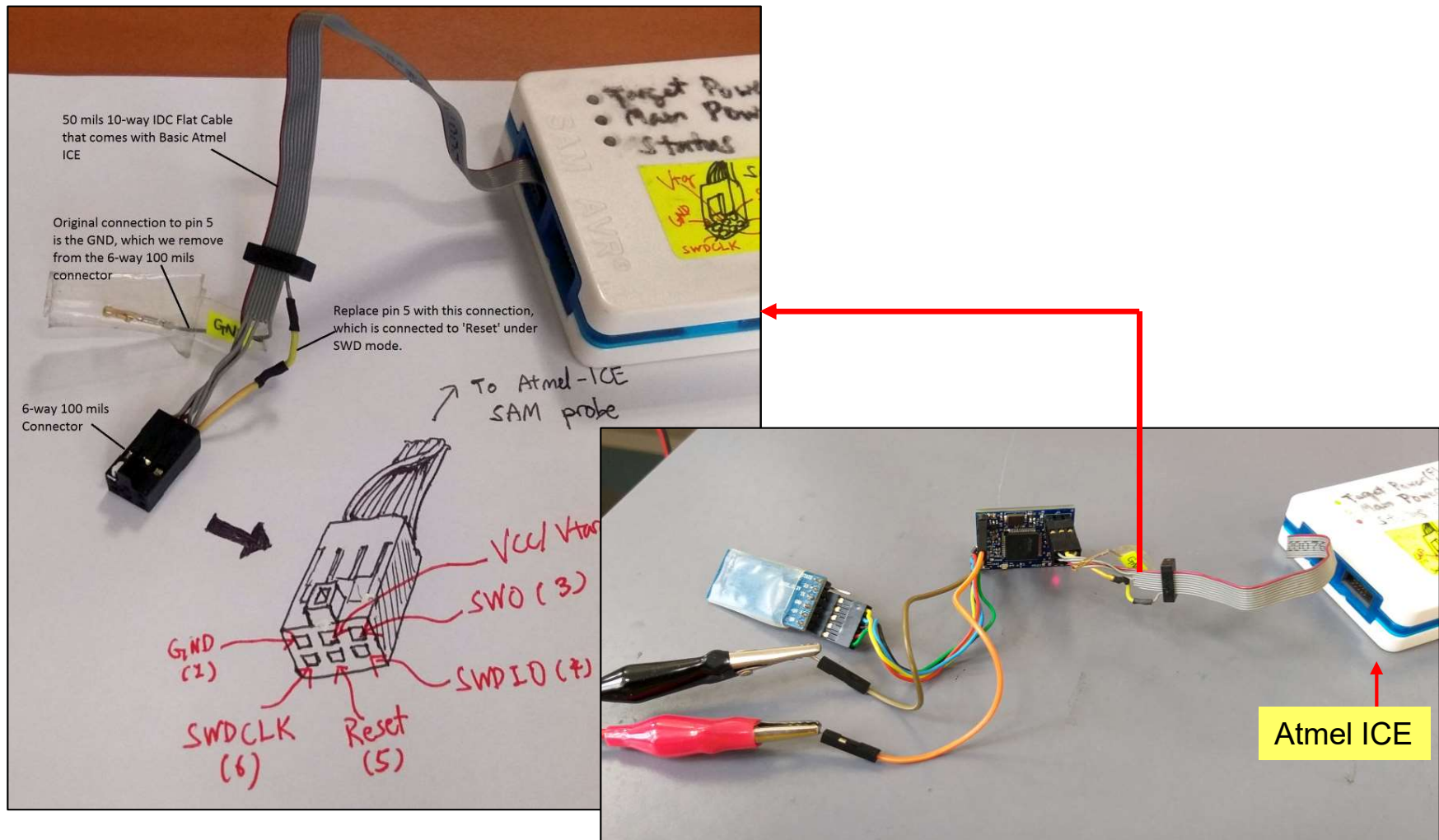
Setting Up Visual Studio Project 6

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



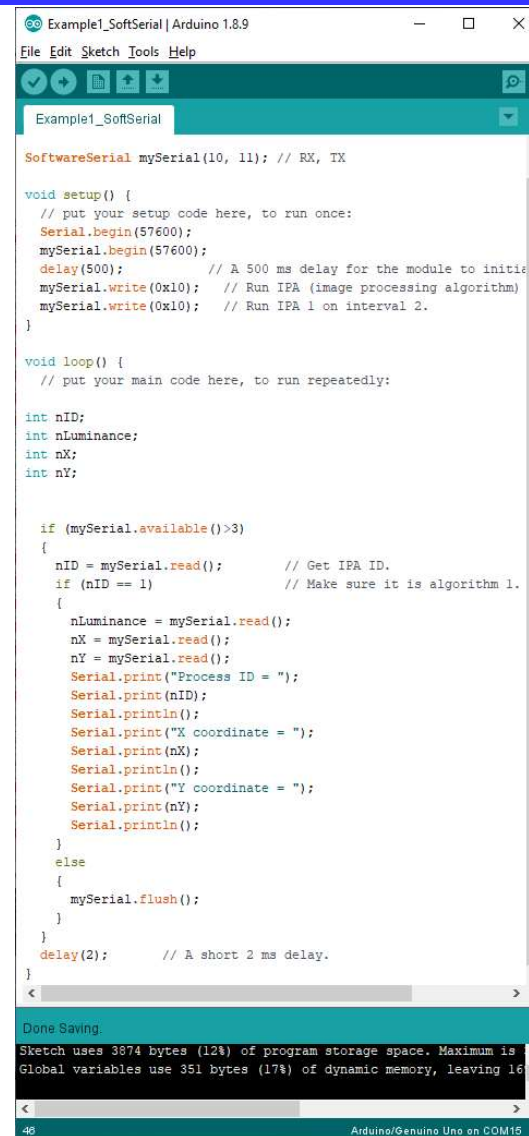
APPENDIX

Appendix 1 – Connecting Atmel ICE to MVM V1.5C



Appendix 2 – Example 1 Using SoftwareSerial

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.



```
Example1_SoftSerial | Arduino 1.8.9
File Edit Sketch Tools Help

SoftwareSerial mySerial(10, 11); // RX, TX

void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
  mySerial.begin(57600);
  delay(500); // A 500 ms delay for the module to initiate
  mySerial.write(0x10); // Run IPA (image processing algorithm)
  mySerial.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 (mySerial.available() > 3)
  {
    nID = mySerial.read(); // Get IPA ID.
    if (nID == 1) // Make sure it is algorithm 1.
    {
      nLuminance = mySerial.read();
      nX = mySerial.read();
      nY = mySerial.read();
      Serial.print("Process ID = ");
      Serial.print(nID);
      Serial.println();
      Serial.print("X coordinate = ");
      Serial.print(nX);
      Serial.println();
      Serial.print("Y coordinate = ");
      Serial.print(nY);
      Serial.println();
    }
    else
    {
      mySerial.flush();
    }
  }
  delay(2); // A short 2 ms delay.
}

Done Saving.
Sketch uses 3874 bytes (12%) of program storage space. Maximum is 32768 bytes.
Global variables use 351 bytes (17%) of dynamic memory, leaving 1639 bytes free.

46 Arduino/Genuino Uno on COM15
```