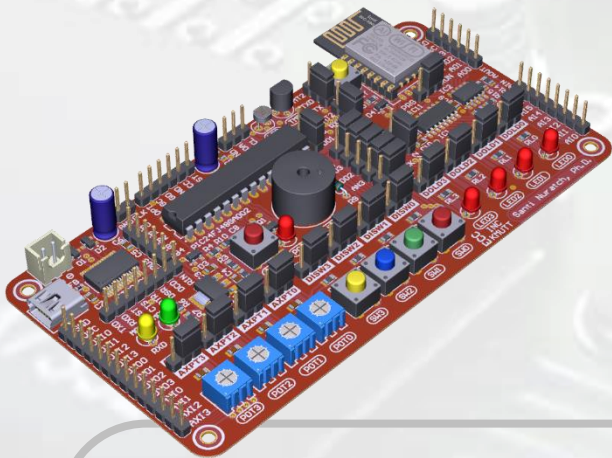


Day #1, Section #2

Embedded C Programming

System Setup &

RTOS+APIs Programming



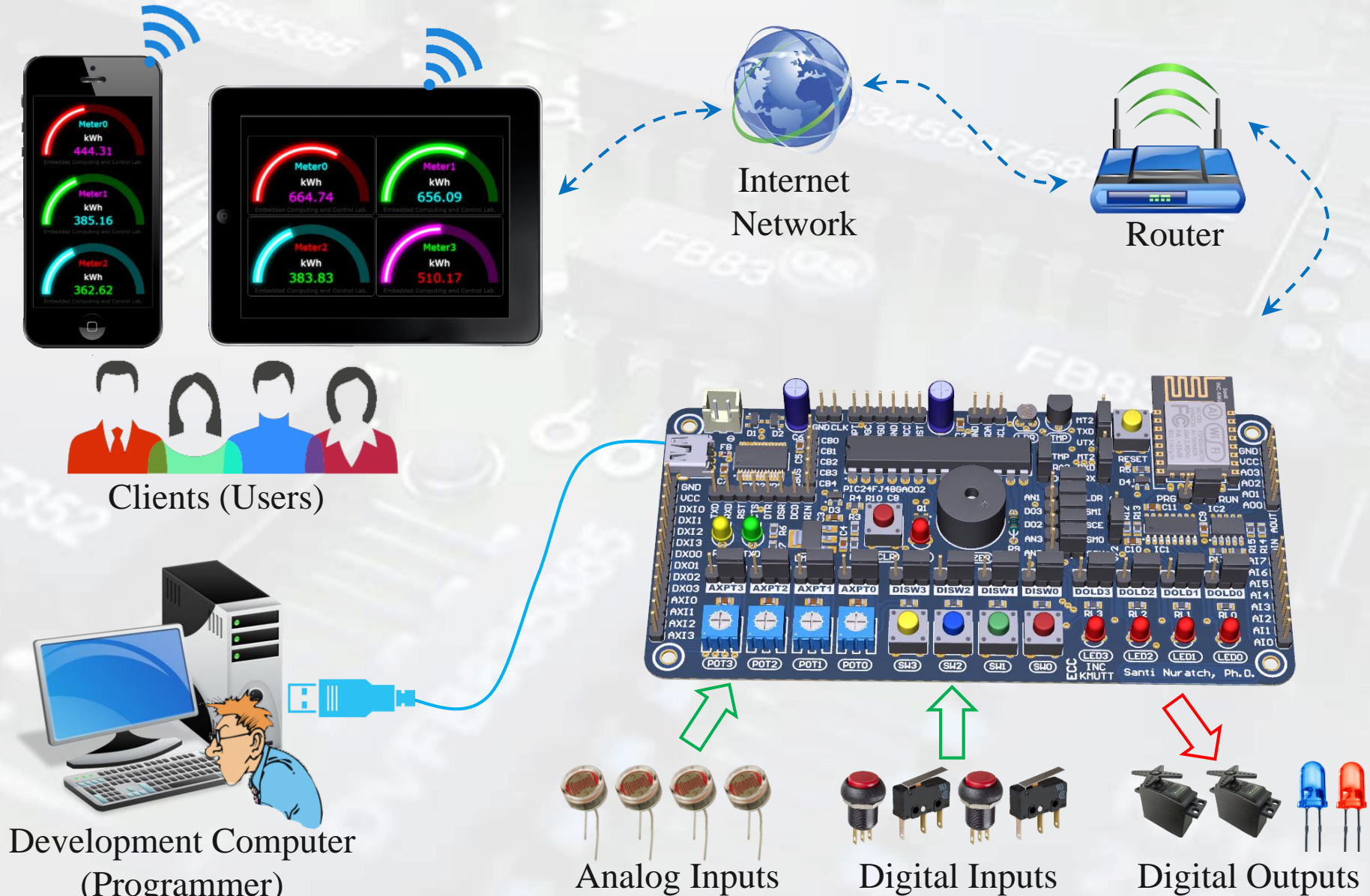
Santi Nuratch., Ph.D.

Embedded Computing and Control Lab. @ INC-KMUTT

santi.inc.kmutt@gmail.com, santi.nur@kmutt.ac.th

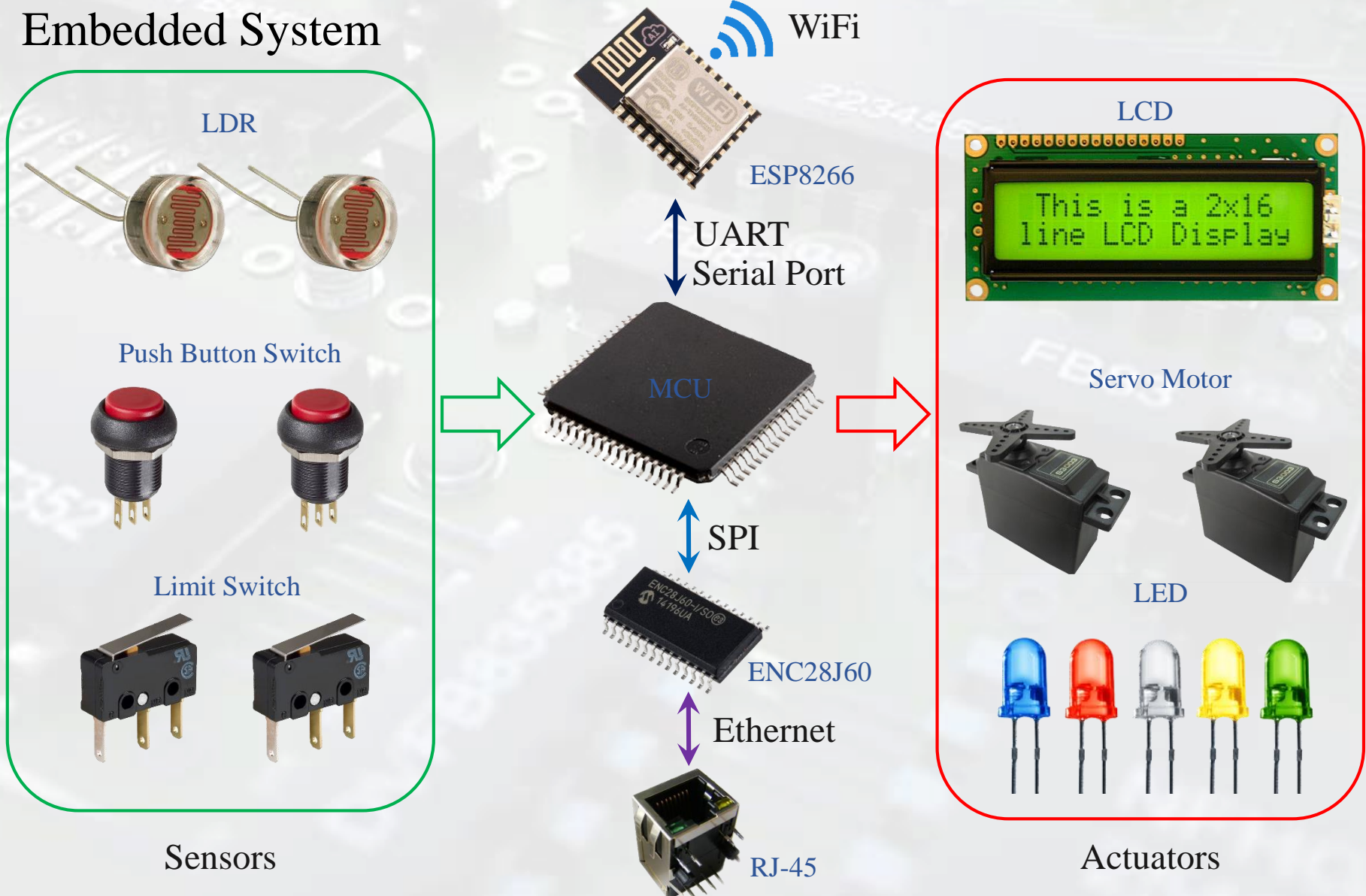
Department of Control System and Instrumentation Engineering,
King Mongkut's University of Technology Thonburi, **KMUTT**

Embedded System and IoT



What is Embedded Systems

Embedded System



Smart Devices vs. Smart Systems

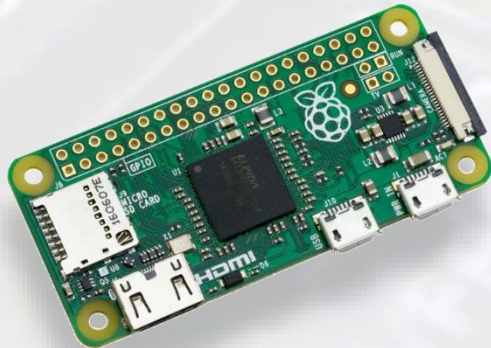
Single-Board Computer and Microcontroller for IoT



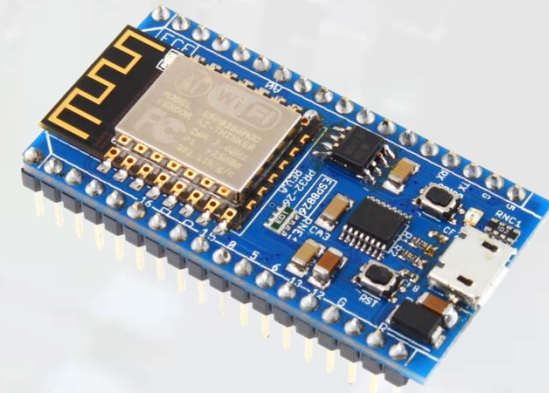
Raspberry Pi



ESP32



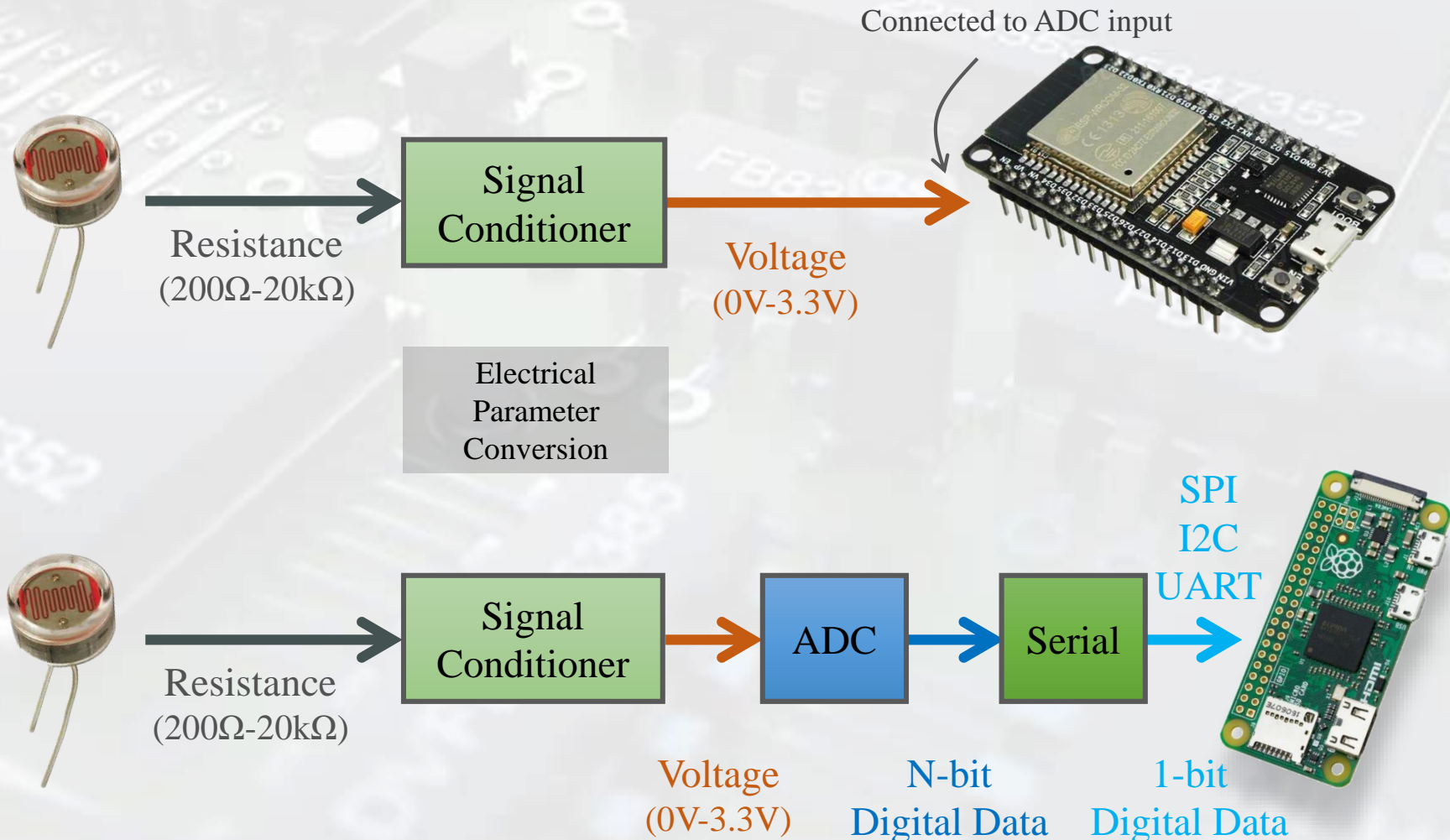
Raspberry Pi-Zero



ESP8266

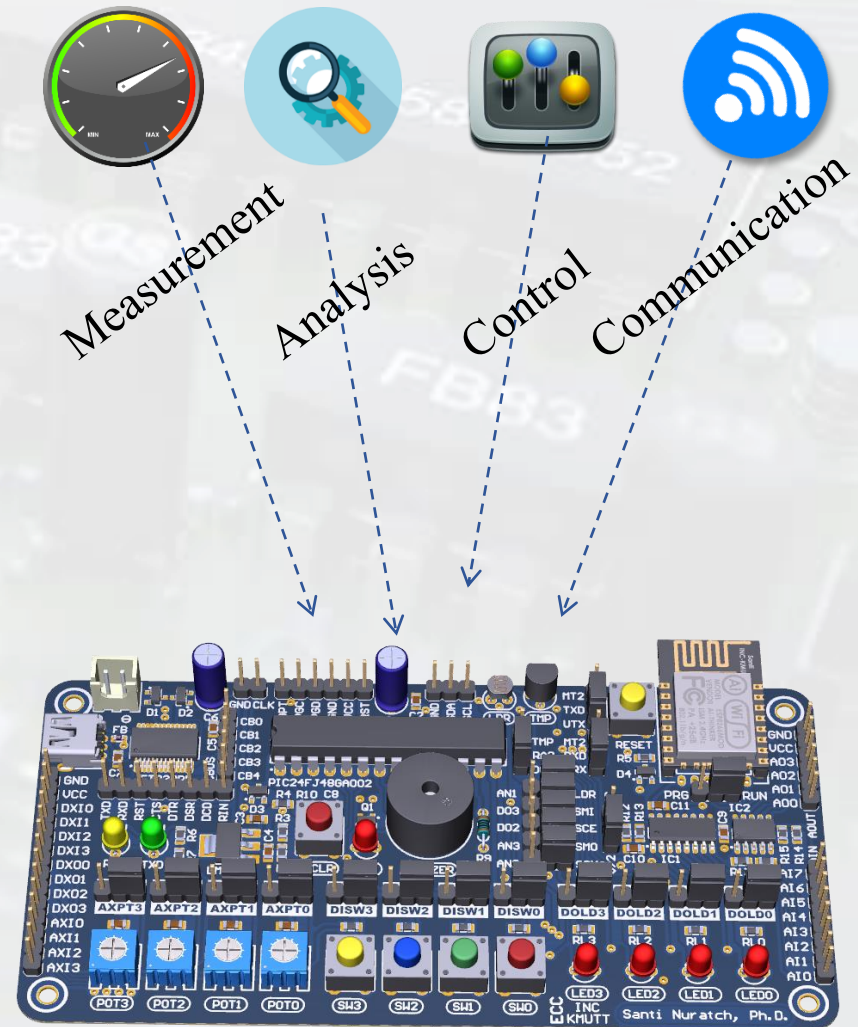
Smart Devices vs. Smart Systems

Sensor and Signal Conditioner/Converter



IoT Devices Key Features

Smart Embedded Devices



Smart Embedded Device

Smart Devices vs. Smart Systems

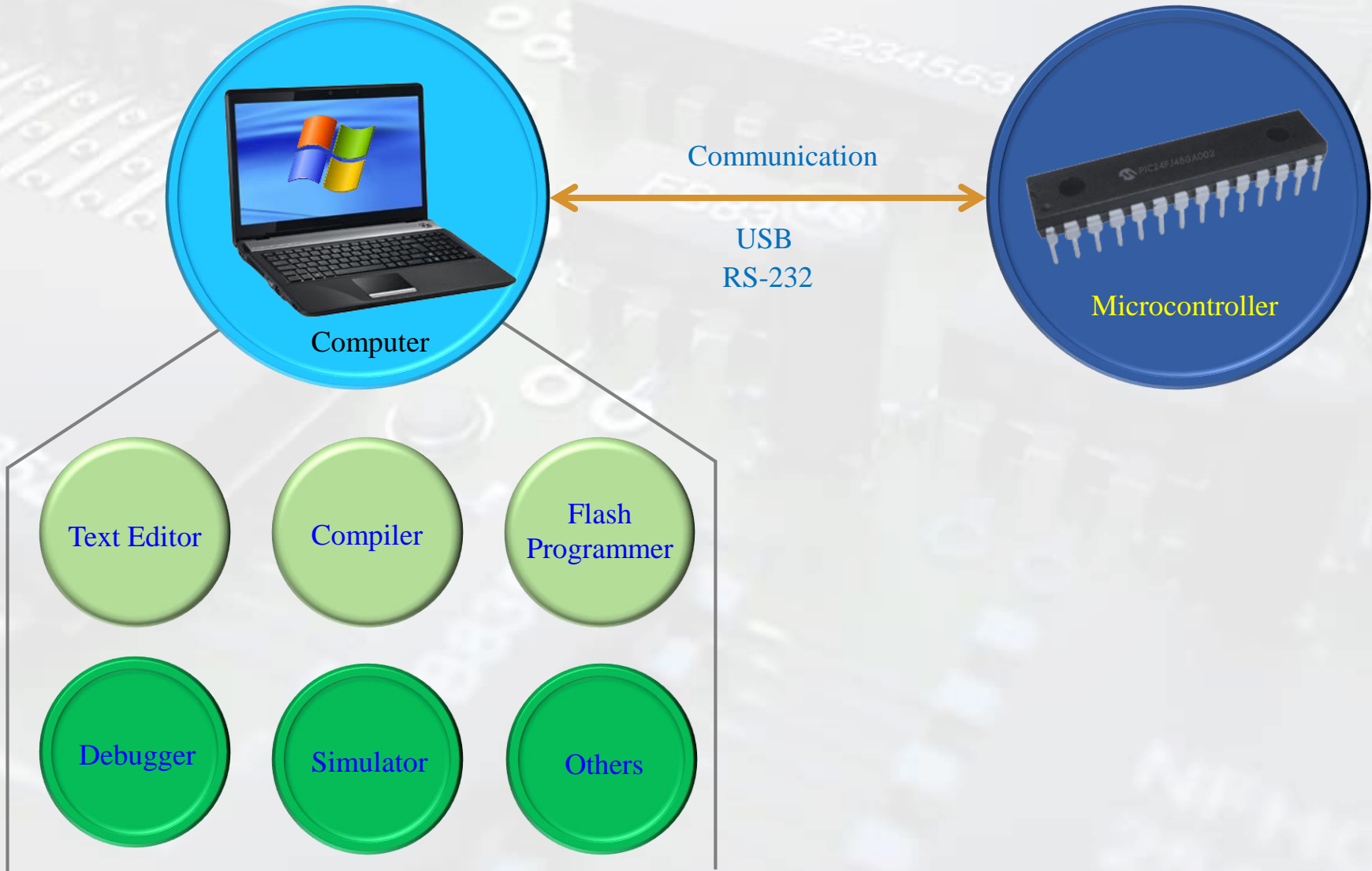
Smart Device (Electronics, Embedded Systems and Computer Technologies)



Smart System is composed of many devices

Software Development

Software Development



Required Software Tools



<https://www.google.com/chrome/>



<https://code.visualstudio.com/>



<http://www.microchip.com/mplab/compilers>



ecc-pic24-cli (given in class)

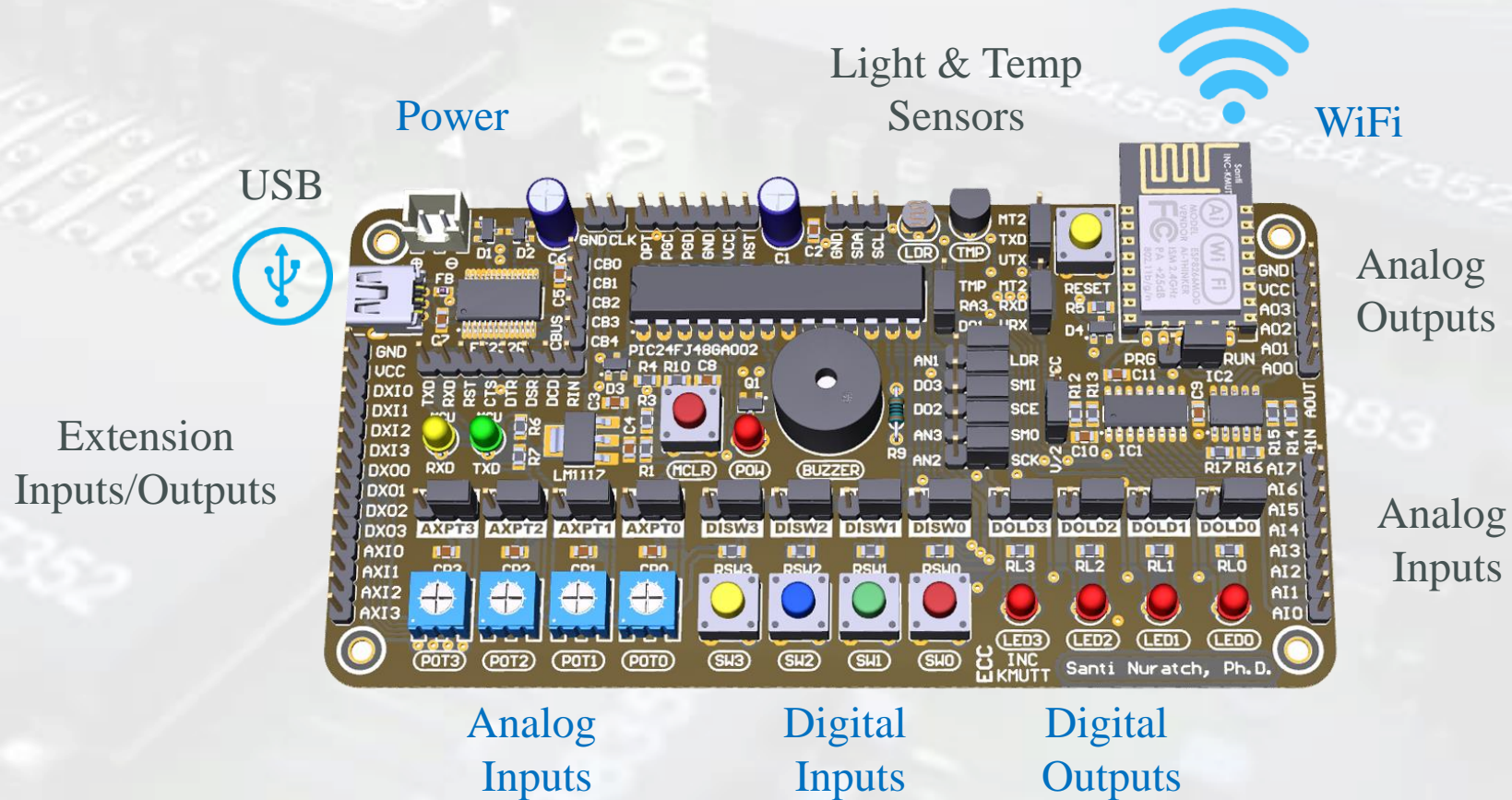


<https://nodejs.org/en/download/>



<https://www.mongodb.com/download-center>

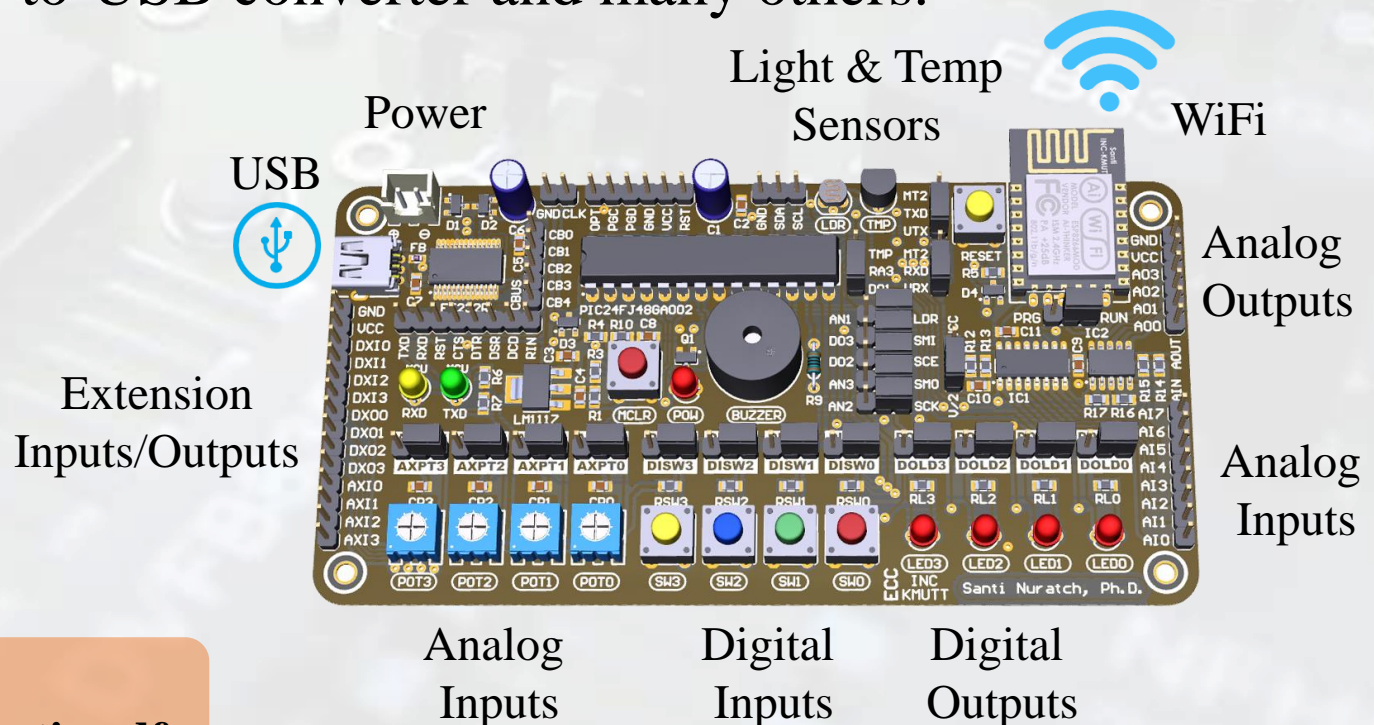
Required Experimental Board



The ECC-PIC24 MCU Board

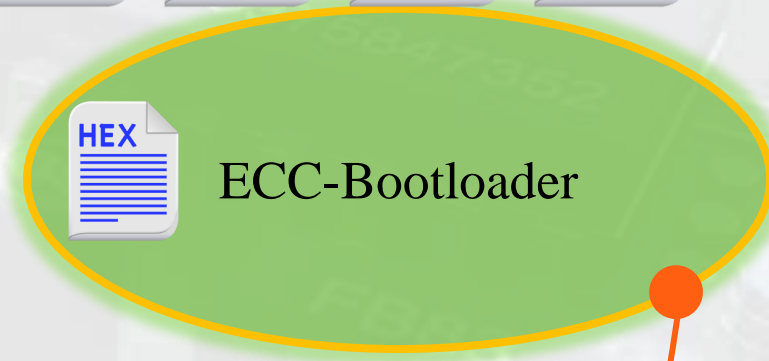
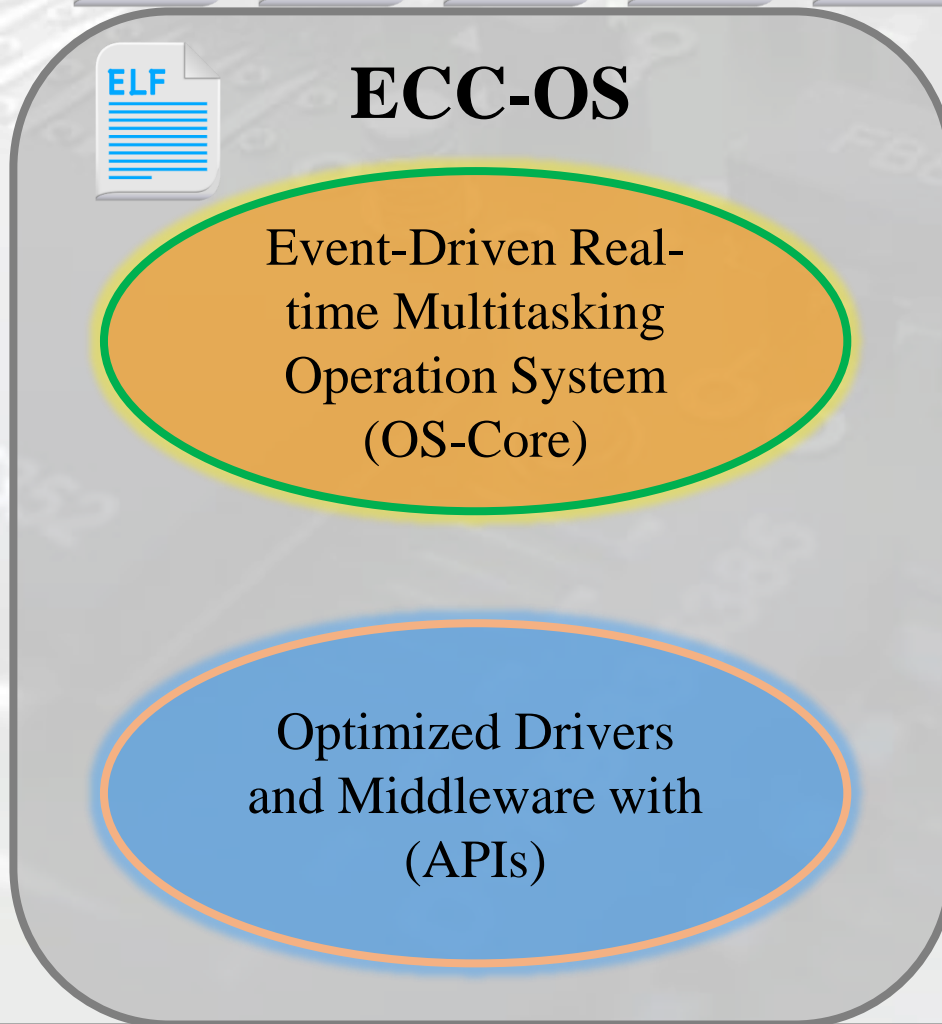


The ECC-PIC24 experimental board is design for embedded learners. The board contains many onboard input/output devices/modules, e.g.; light-emitting diodes (LEDs), push button switches (PBS), light-dependent resistor (LDR), Buzzer, UART-to-USB converter and many others.



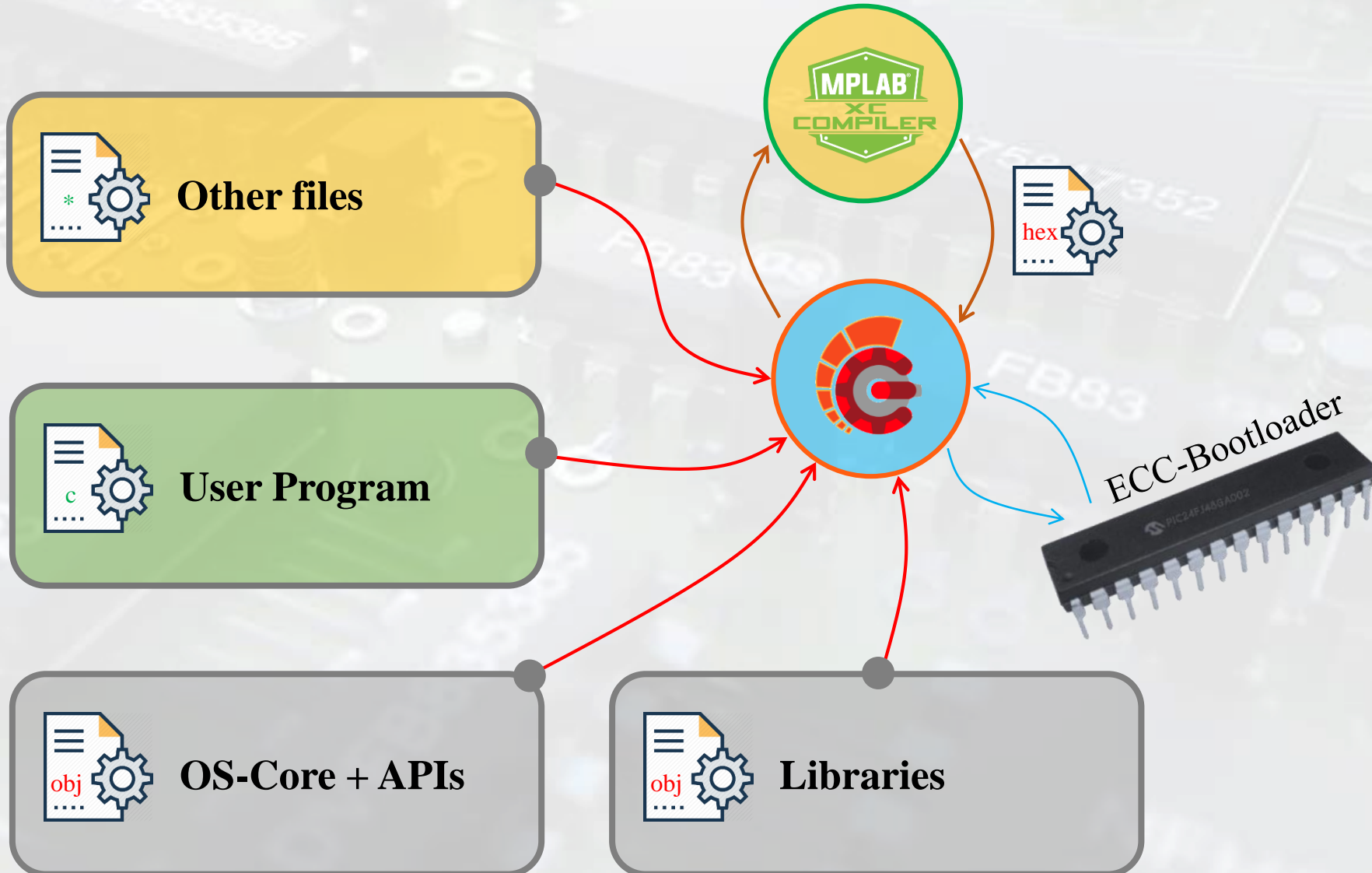
More details:
[EccPic24_Schematic.pdf](#)

MCU + RTOS + Drivers



PIC24FJ48GA002

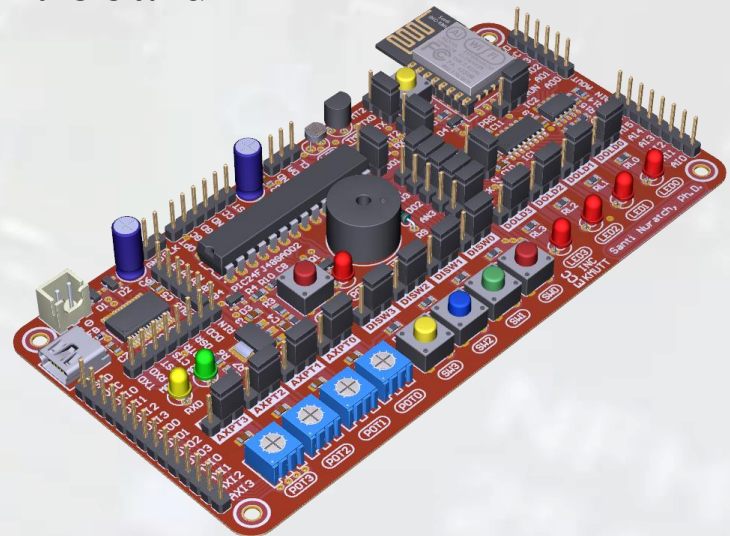
MCU + RTOS + Drivers



The ECC-OS (The Real-time Multitasking Operating System)



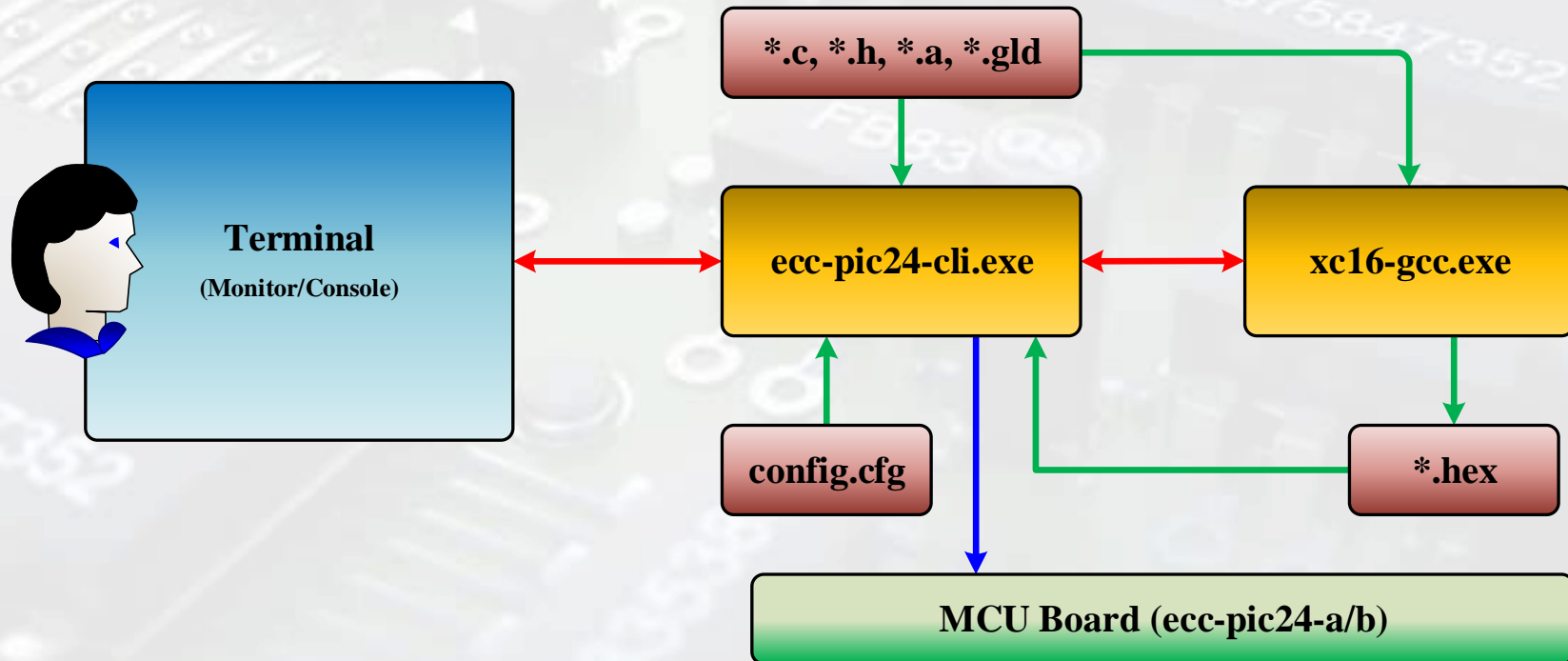
The ECC-OS is a small footprint operating system for small microcontrollers. It is designed and developed using event-driven and callback-function techniques. It fully supports many real-time multitasking applications. Internally, the ECC-OS provides many utility functions, e.g.; serial communication, basic function of input/output ports (Digital and Analog) and time management. Also, it has built-in functions to working with the ECC-PIC24 experiment board



What is the **ecc-pic24-cli**?



The **ecc-pic24-cli** is a command-line interface application used for linking to **xc16-gcc** compiler and microcontroller board



Mainly, the **ecc-pic24-cli** receives user's command(s), reads a configuration file (config.cfg) and prepares special commands for the **xc16-gcc** (compiler). It also reads and processes HEX file (*.hex) before sending to flash memory of target microcontroller.

Setup the ecc-pic24-cli



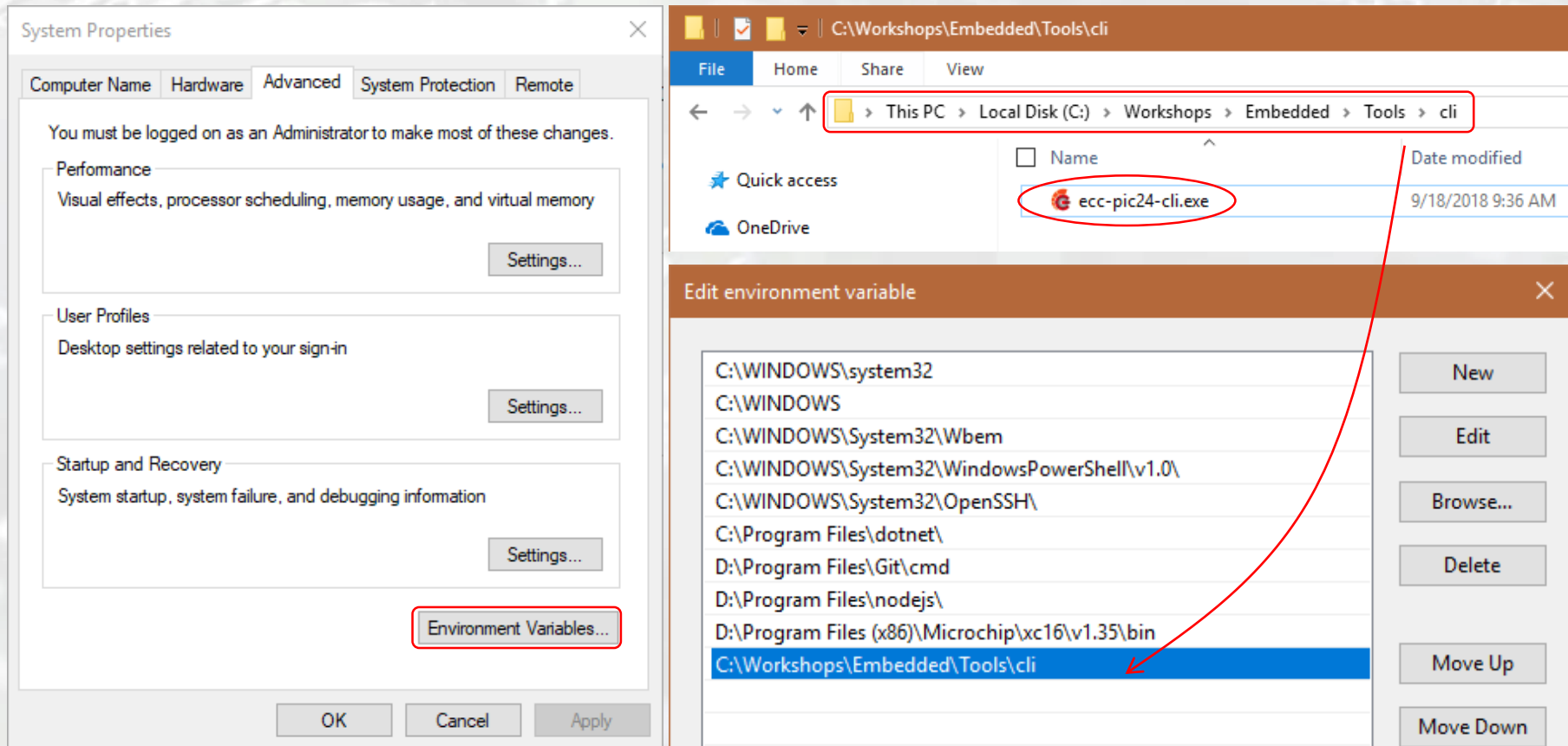
ecc-pic24-cli
Setting up



ccc-pic24-cli Setup

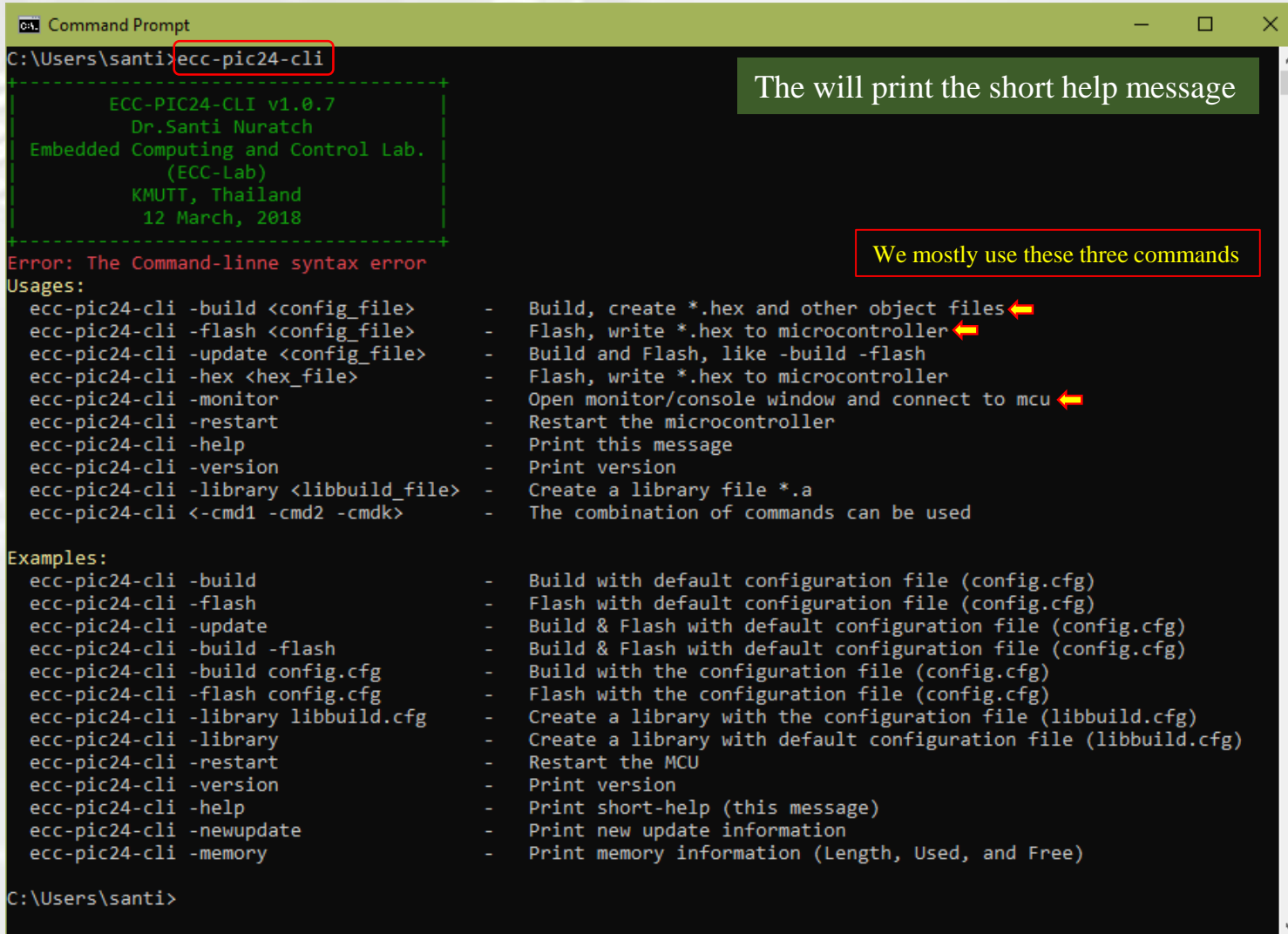
01 : Copy the **Workshops** folder and paste in **C:/** (or another e.g. D:/)

02 : Add the **C:\Workshops\Embedded\Tools\cli** into System **Environment Variables**



The image displays two Windows interface elements side-by-side. On the left is the 'System Properties' dialog box, with the 'Advanced' tab selected. The 'Environment Variables...' button at the bottom is highlighted with a red rectangle. On the right is a File Explorer window showing the path 'C:\Workshops\Embedded\Tools\cli'. The file 'ccc-pic24-cli.exe' is circled in red. Below the File Explorer is the 'Edit environment variable' dialog box, which lists various system paths. The path 'C:\Workshops\Embedded\Tools\cli' is highlighted in blue, and a red arrow points from the circled file in the File Explorer to this highlighted path.

03 : Run a Terminal/Console and give it the command **ecc-pic24-cli**



```
Command Prompt
C:\Users\santi>ecc-pic24-cli

ECC-PIC24-CLI v1.0.7
Dr.Santi Nuratch
Embedded Computing and Control Lab.
(ECC-Lab)
KMUTT, Thailand
12 March, 2018

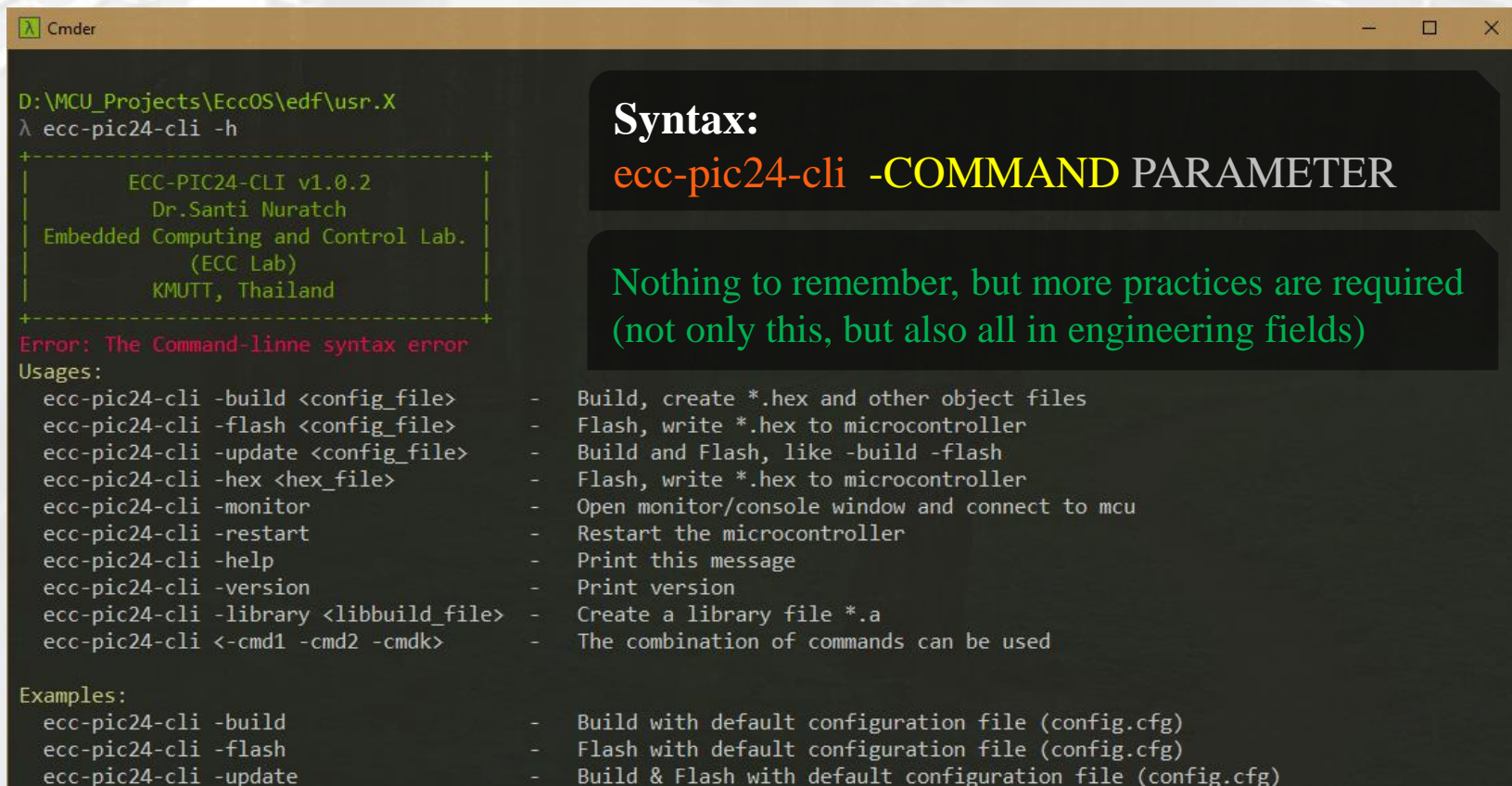
Error: The Command-line syntax error
Usages:
ecc-pic24-cli -build <config_file>      - Build, create *.hex and other object files
ecc-pic24-cli -flash <config_file>      - Flash, write *.hex to microcontroller
ecc-pic24-cli -update <config_file>     - Build and Flash, like -build -flash
ecc-pic24-cli -hex <hex_file>          - Flash, write *.hex to microcontroller
ecc-pic24-cli -monitor                 - Open monitor/console window and connect to mcu
ecc-pic24-cli -restart                  - Restart the microcontroller
ecc-pic24-cli -help                     - Print this message
ecc-pic24-cli -version                  - Print version
ecc-pic24-cli -library <libbuild_file> - Create a library file *.a
ecc-pic24-cli <-cmd1 -cmd2 -cmdk>      - The combination of commands can be used

Examples:
ecc-pic24-cli -build                    - Build with default configuration file (config.cfg)
ecc-pic24-cli -flash                    - Flash with default configuration file (config.cfg)
ecc-pic24-cli -update                   - Build & Flash with default configuration file (config.cfg)
ecc-pic24-cli -build -flash              - Build & Flash with default configuration file (config.cfg)
ecc-pic24-cli -build config.cfg          - Build with the configuration file (config.cfg)
ecc-pic24-cli -flash config.cfg          - Flash with the configuration file (config.cfg)
ecc-pic24-cli -library libbuild.cfg      - Create a library with the configuration file (libbuild.cfg)
ecc-pic24-cli -library                  - Create a library with default configuration file (libbuild.cfg)
ecc-pic24-cli -restart                   - Restart the MCU
ecc-pic24-cli -version                   - Print version
ecc-pic24-cli -help                     - Print short-help (this message)
ecc-pic24-cli -newupdate                 - Print new update information
ecc-pic24-cli -memory                    - Print memory information (Length, Used, and Free)

C:\Users\santi>
```



To work with the **ecc-pic24-cli**, or others command-line interface applications, a terminal/console is required, e.g.; the Command Prompt (CMD) or other console emulators for Windows. The **cmdr** is recommended. Let's try the first command, **ecc-pic24-cli -help**



```
Cmdr
D:\MCU_Projects\EccOS\edf\usr.X
λ ecc-pic24-cli -h
+-----+
| ECC-PIC24-CLI v1.0.2          |
| Dr.Santi Nuratch             |
| Embedded Computing and Control Lab. |
| (ECC Lab)                   |
| KMUTT, Thailand              |
+-----+
Error: The Command-line syntax error
Usages:
ecc-pic24-cli -build <config_file> - Build, create *.hex and other object files
ecc-pic24-cli -flash <config_file> - Flash, write *.hex to microcontroller
ecc-pic24-cli -update <config_file> - Build and Flash, like -build -flash
ecc-pic24-cli -hex <hex_file> - Flash, write *.hex to microcontroller
ecc-pic24-cli -monitor - Open monitor/console window and connect to mcu
ecc-pic24-cli -restart - Restart the microcontroller
ecc-pic24-cli -help - Print this message
ecc-pic24-cli -version - Print version
ecc-pic24-cli -library <libbuild_file> - Create a library file *.a
ecc-pic24-cli <-cmd1 -cmd2 -cmdk> - The combination of commands can be used

Examples:
ecc-pic24-cli -build - Build with default configuration file (config.cfg)
ecc-pic24-cli -flash - Flash with default configuration file (config.cfg)
ecc-pic24-cli -update - Build & Flash with default configuration file (config.cfg)
```

Syntax:
ecc-pic24-cli -COMMAND PARAMETER

Nothing to remember, but more practices are required
(not only this, but also all in engineering fields)

Setup the xc16-gcc

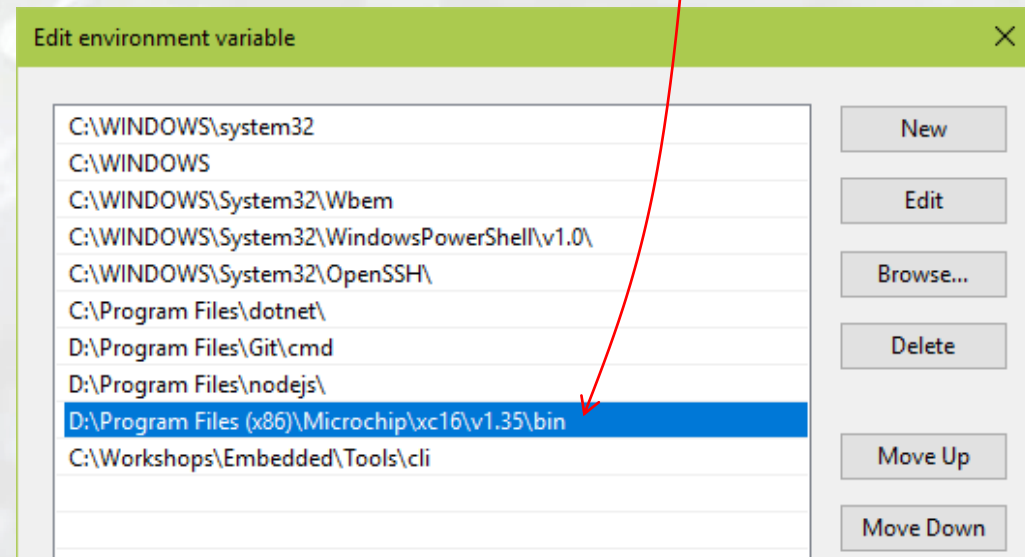
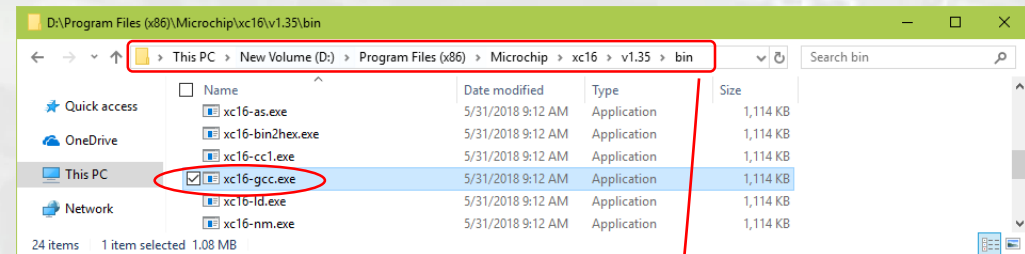
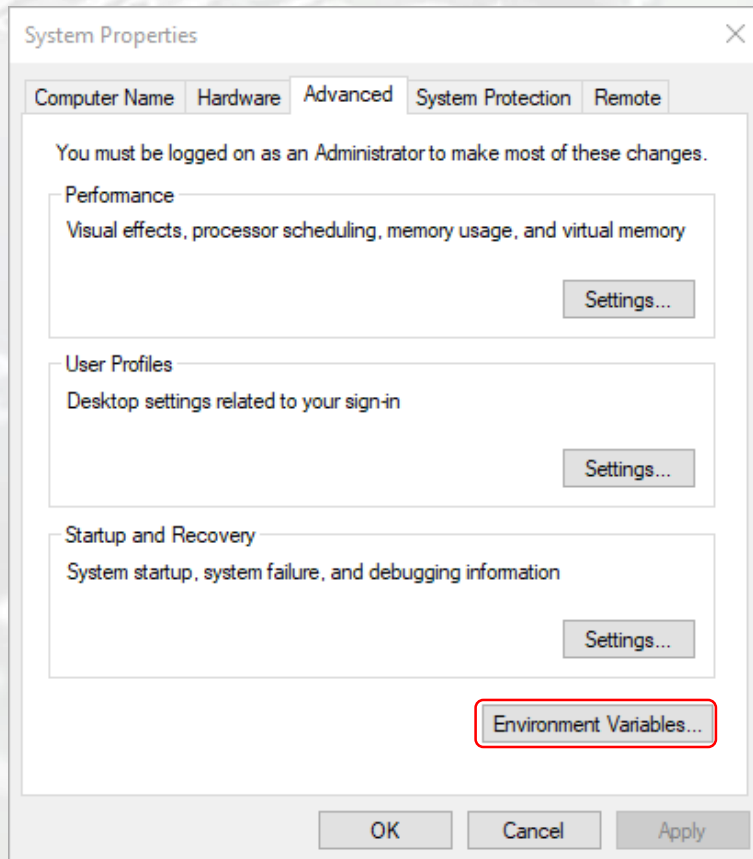


xc16-gcc
Setting up

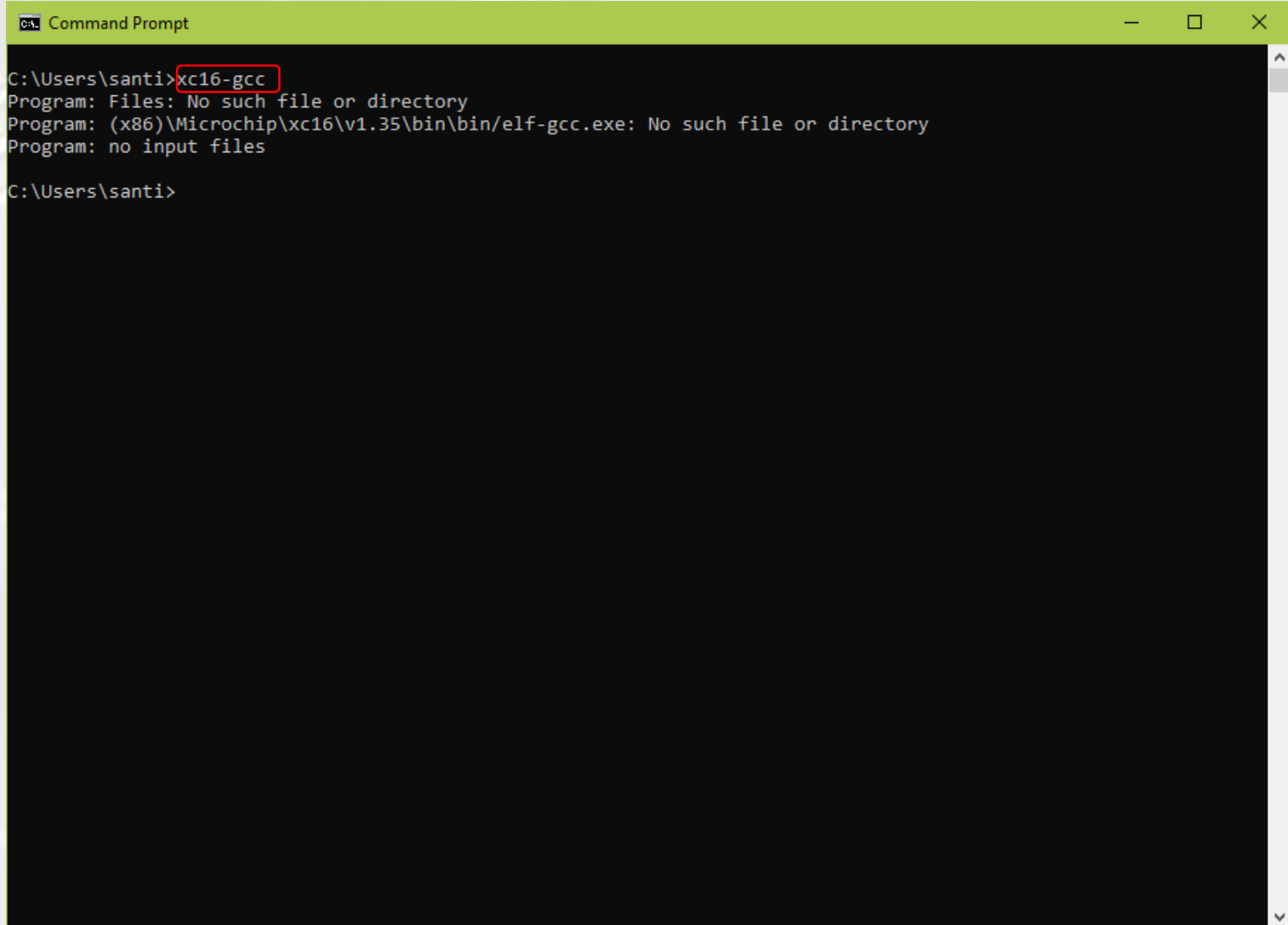
xc16-gcc Setup

01 : Install the MPLAB **XC16 C Compiler v1.35** (<http://www.microchip.com/mplab/compilers>)

02 : Add the **D:\Program Files (x86)\Microchip\xc16\v1.35\bin** into System Environment Variables



03 : Run a Terminal/Console and give it the command **xc16-gcc**

A screenshot of a Windows Command Prompt window titled 'Command Prompt'. The window has a green title bar and standard minimize, maximize, and close buttons. The command prompt shows the user 'santi' at the 'C:\Users\santi' directory. The command 'xc16-gcc' has been entered and is highlighted with a red rectangle. The output shows three error messages: 'Program: Files: No such file or directory', 'Program: (x86)\Microchip\xc16\v1.35\bin\bin\elf-gcc.exe: No such file or directory', and 'Program: no input files'. The prompt then returns to 'C:\Users\santi>'.

```
Command Prompt
C:\Users\santi>xc16-gcc
Program: Files: No such file or directory
Program: (x86)\Microchip\xc16\v1.35\bin\bin\elf-gcc.exe: No such file or directory
Program: no input files
C:\Users\santi>
```

Setup the config.cfg



config.cfg
Setting up



Example of the **ecc-pic24-cli** configuration file



To make the **ecc-pic24-cli** works correctly, a configuration file (config.cfg) is needed to be written carefully. The config.cfg contains all information of a project, e.g.; source files and directories, communication properties, compiler's path and others.

```
SRC_FILE      = ./main.c
# COM_NAME    = COM5
# COM_BAUD    = 115200
MCU_PART      = 24FJ48GA002
INC_DIR       = ../../library/header
LNK_FILE      = ../../library/linker/24FJ48GA002.gld
LIB_FILE      = ../../library/os-lib/lib.X.a
OUT_DIR       = ./output
HEX_FILE      = ./firmware.hex
XC16_DIR      = C:\Program Files (x86)\Microchip\xc16\v1.35\bin
# XC16_DIR    = D:\Program Files (x86)\Microchip\xc16\v1.35\bin
```



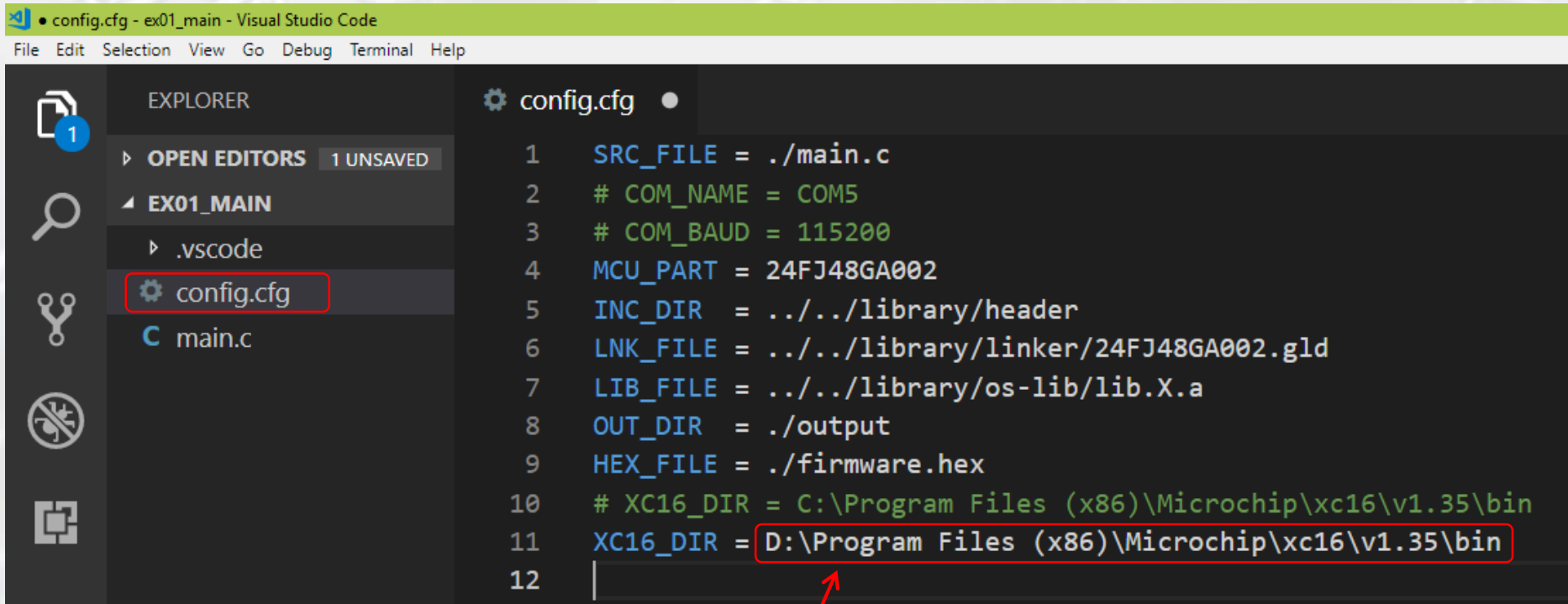
We have to know exactly their meanings

config.cfg Setup

01 : Run Visual Code Studio and Open Folder

C:\Workshops\Embedded\Examples\ex01_main

02 : Double-click the **config.cfg** to open and check all lines carefully

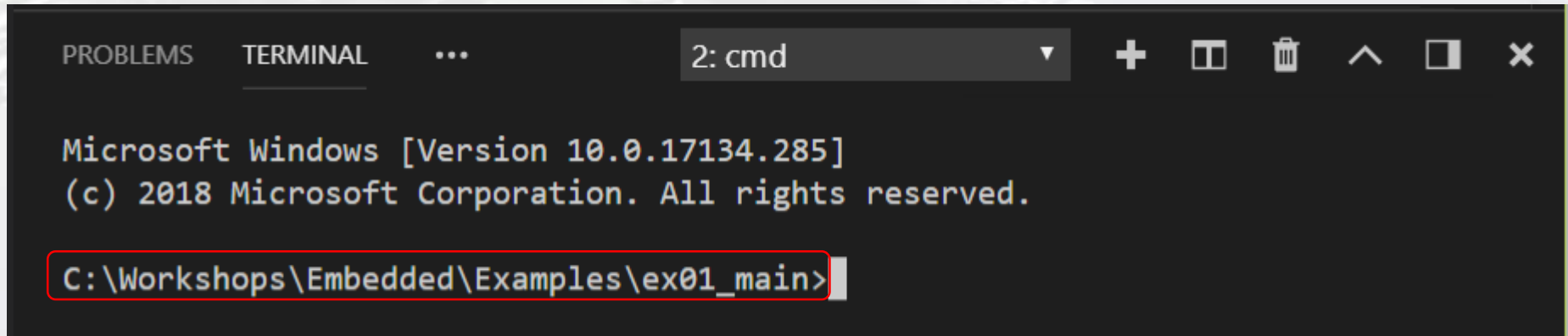


```
1 SRC_FILE = ./main.c
2 # COM_NAME = COM5
3 # COM_BAUD = 115200
4 MCU_PART = 24FJ48GA002
5 INC_DIR = ../../library/header
6 LNK_FILE = ../../library/linker/24FJ48GA002.gld
7 LIB_FILE = ../../library/os-lib/lib.X.a
8 OUT_DIR = ./output
9 HEX_FILE = ./firmware.hex
10 # XC16_DIR = C:\Program Files (x86)\Microchip\xc16\v1.35\bin
11 XC16_DIR = D:\Program Files (x86)\Microchip\xc16\v1.35\bin
12
```

This line links the **ecc-pic24-cli** to the **xc16-gcc**

config.cfg Setup

03 : In the Visual Code Studio, click the **Terminal** | **New Terminal** on the menu bar, and check the working directory (the directory can be changed by **cd**

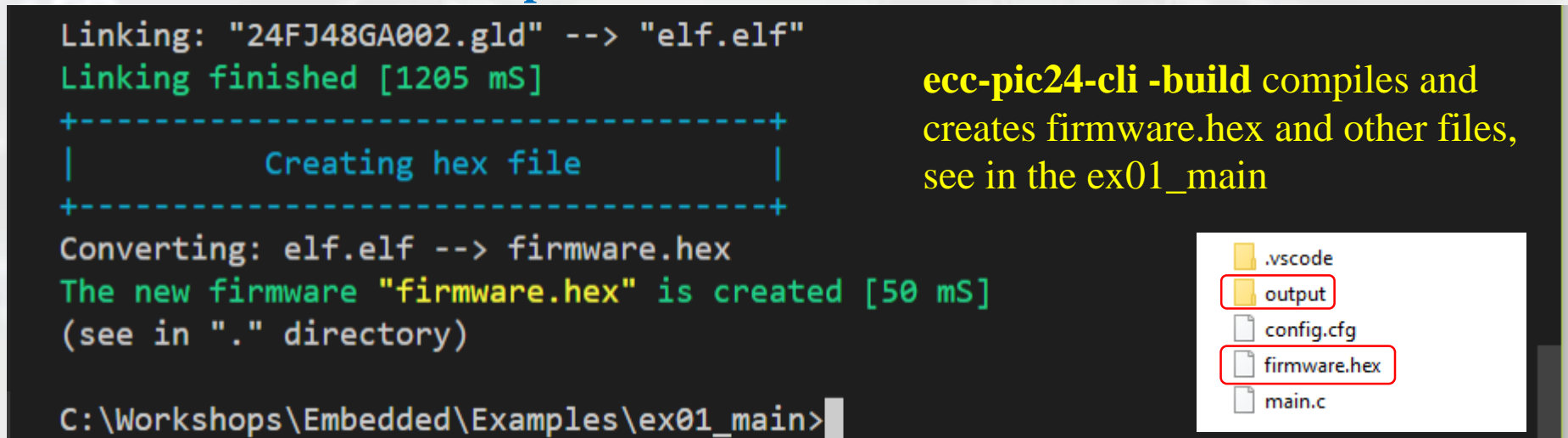


```
PROBLEMS  TERMINAL  ...  2: cmd  +  [icon]  [icon]  [icon]  [icon]  [icon]  [icon]  [icon]

Microsoft Windows [Version 10.0.17134.285]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Workshops\Embedded\Examples\ex01_main>
```

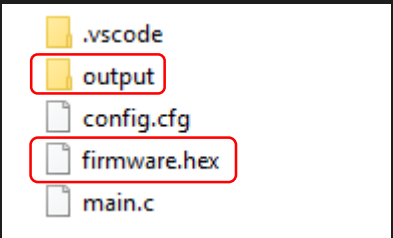
04 : Enter a command **ecc-pic24-cli -build**



```
Linking: "24FJ48GA002.gld" --> "elf.elf"
Linking finished [1205 mS]
+-----+
|           Creating hex file           |
+-----+
Converting: elf.elf --> firmware.hex
The new firmware "firmware.hex" is created [50 mS]
(see in "." directory)

C:\Workshops\Embedded\Examples\ex01_main>
```

ecc-pic24-cli -build compiles and creates firmware.hex and other files, see in the ex01_main





The background image shows a green microcontroller board with a small screen. The screen displays a heart rate monitor interface with a green line graph, a heart icon, and the number '82'. The board also has several blue potentiometers and a battery connected to it. The text 'VS-Code IntelliSense Setting up' is overlaid on the image in a large, bold, yellow and green font.

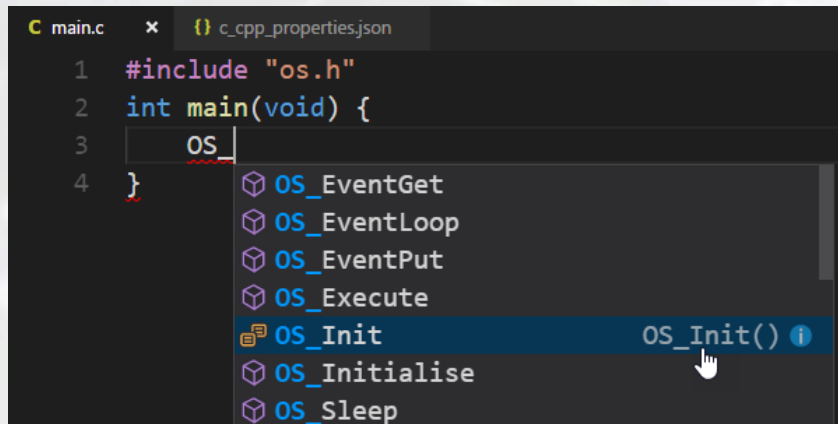
VS-Code IntelliSense Setting up



To make the VS-Code shows **intellisense**, the following steps are required:

- 1) Open the application directory (root or app)
- 2) Open the **c_cpp_properties.json**, and add the lines shown below into the **includePath** and **intelliSenseMode**

```
"D:/MyDirectory/root/library/header/*",  
"C:/Program Files (x86)/Microchip/xc16/v1.35/include",  
"C:/Program Files (x86)/Microchip/xc16/v1.35/support/generic/h",  
"C:/Program Files (x86)/Microchip/xc16/v1.35/support/PIC24F/h",
```



All are depended on your system.

- 3) Restart the VS-Code and check the intellisense

The First C-Program (based-on ECC-OS)



Programming with ECC-OS (and others) requires special knowledge of C-Programming, the Embedded C Programming. In this class, we are not going in details on that topic, but you have to do!. The lines of code below are the main function of the C-Program. Inside the main, just two functions of the OS are required. The **OS_Init()** must be executed before all functions of the ECC-OS are called. The **OS_Start()** must be executed to start the OS.

```
#include "os.h"
int main(void)
{
    // peripherals and variables initialization
    OS_Init();
    // Other initializations
    OS_Start();
}
```

Setup the Microcontroller Board

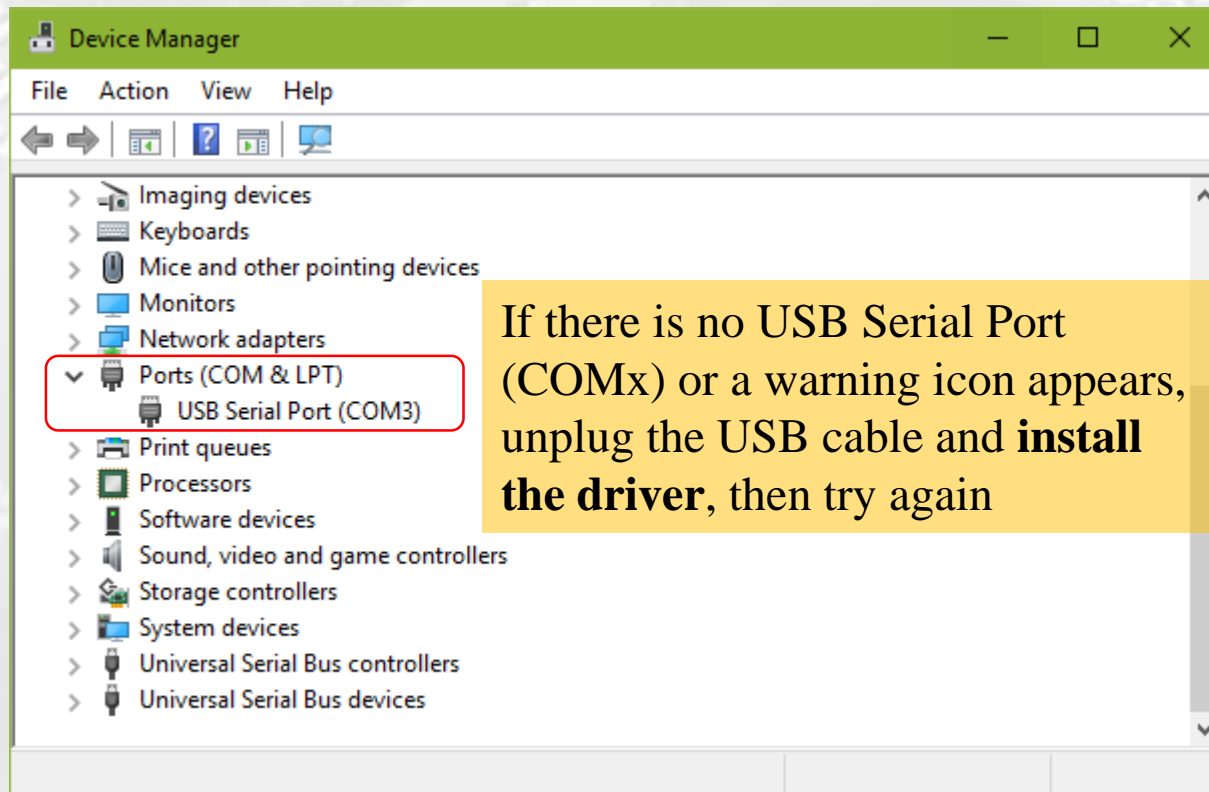


The background image shows a green printed circuit board (PCB) for the ECC-PIC24 MCU. It features a PIC24 microcontroller, a 1602 LCD display showing a heart rate monitor interface with a green waveform, and a 9V battery connected to the power pins. The board also has several blue potentiometers and a push button. The text 'ECC-PIC24 MCU Board' is overlaid in yellow, and 'Setting up' is overlaid in green.

ECC-PIC24 MCU Board Setting up

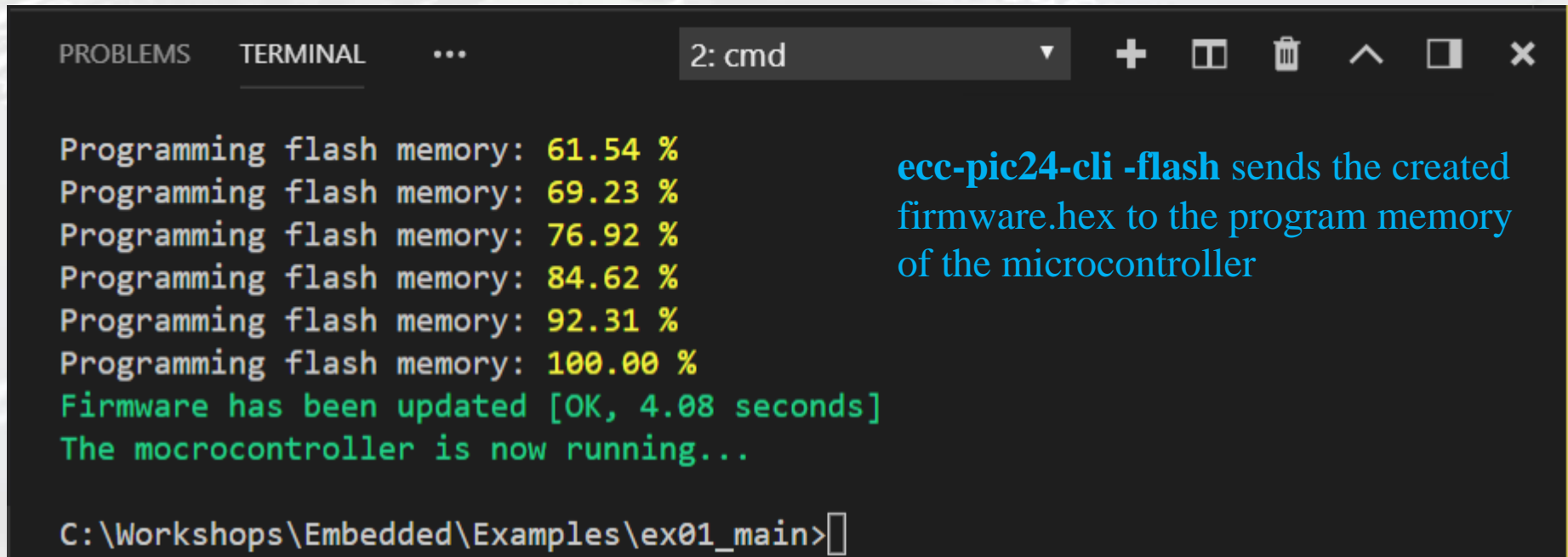
ECC-PIC24 MCU Board Setup

01 : Connect the ECC-PIC24 MCU Board to the computer using the USB cable and check its existing in the Device Manager



The VCP driver (FTDI) can be downloaded from <https://www.ftdichip.com/Drivers/VCP.htm>

02 : Back to the Terminal of the Visual Code Studio again and give it a command **`ecc-pic24-cli -flash`**




```
PROBLEMS  TERMINAL  ...  2: cmd  +  [icon]  [icon]  [icon]  [icon]  [icon]  [icon]

Programming flash memory: 61.54 %
Programming flash memory: 69.23 %
Programming flash memory: 76.92 %
Programming flash memory: 84.62 %
Programming flash memory: 92.31 %
Programming flash memory: 100.00 %
Firmware has been updated [OK, 4.08 seconds]
The microcontroller is now running...

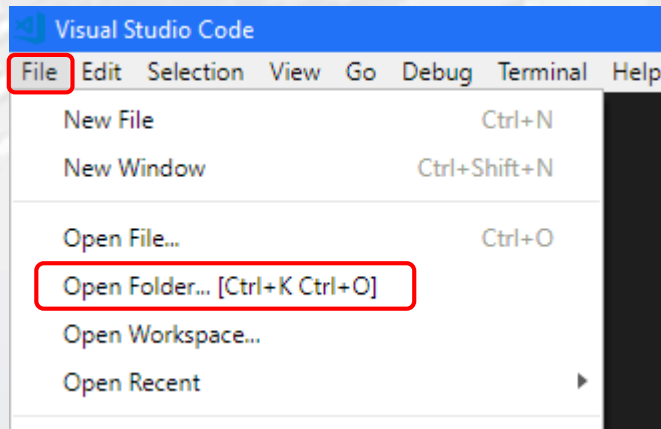
C:\Workshops\Embedded\Examples\ex01_main>
```

`ecc-pic24-cli -flash` sends the created firmware.hex to the program memory of the microcontroller

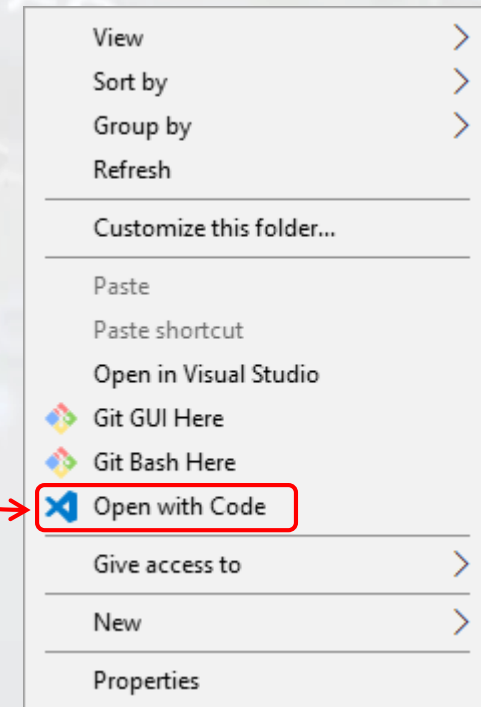


Embedded C-Programming with ECC-OS + APIs

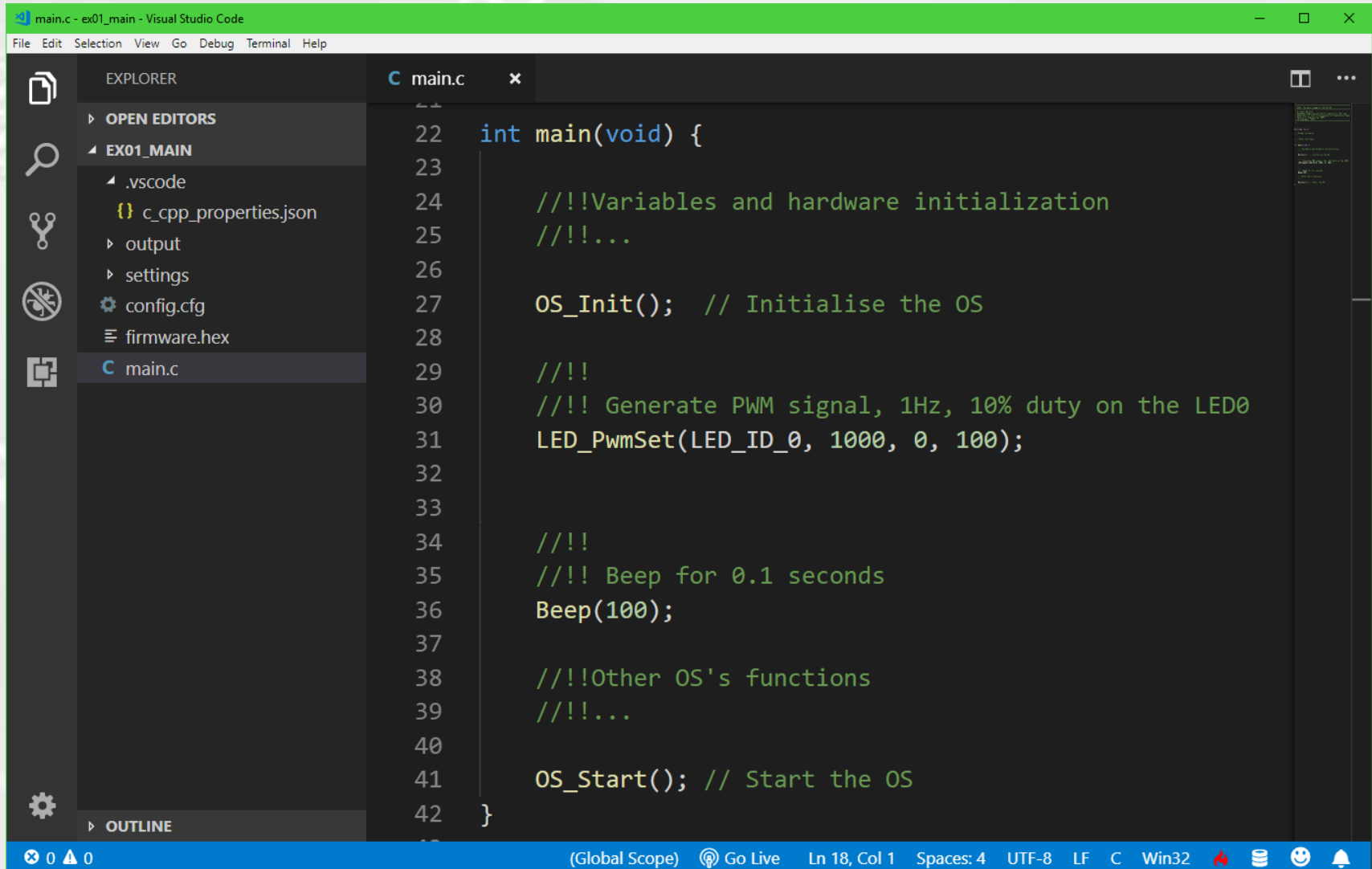
01 : Run the VS-Code and open the first example,
[C:\Workshops\Embedded\Examples\ex01_main](#), File | Open Folder...



If the VS-Code is configured correctly, go to the target directory, right-click and choose Open with Code

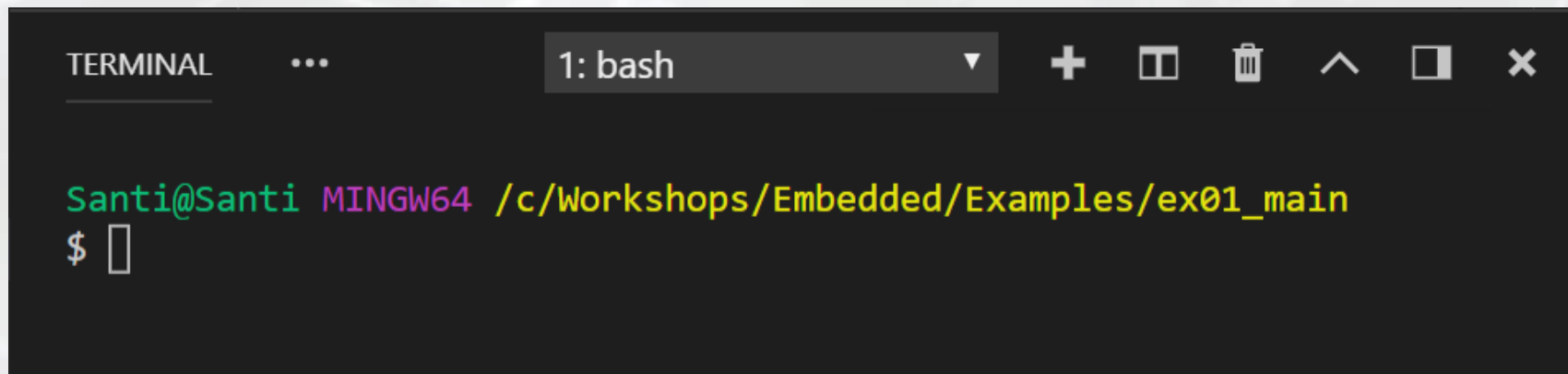
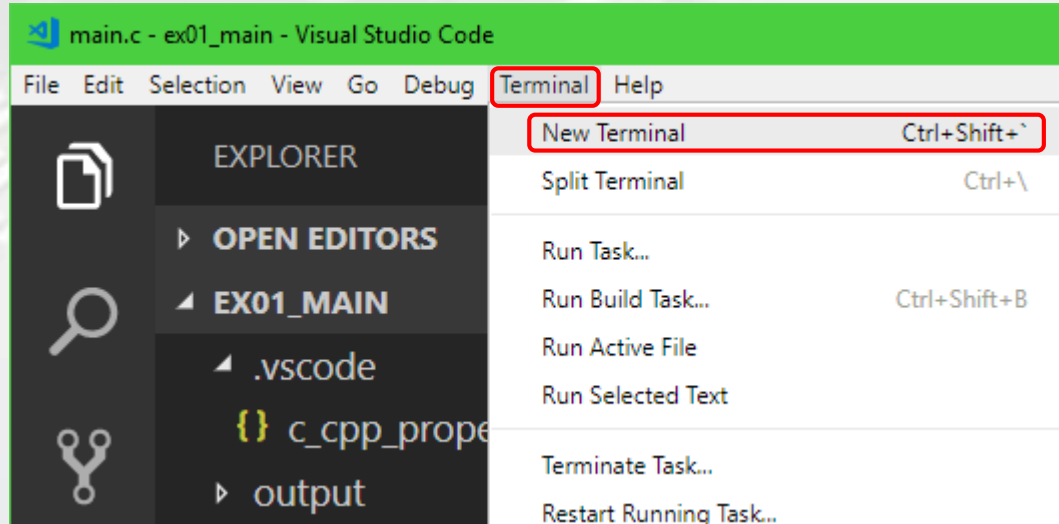


02 : Double-click on the main.c and see closely how it is written



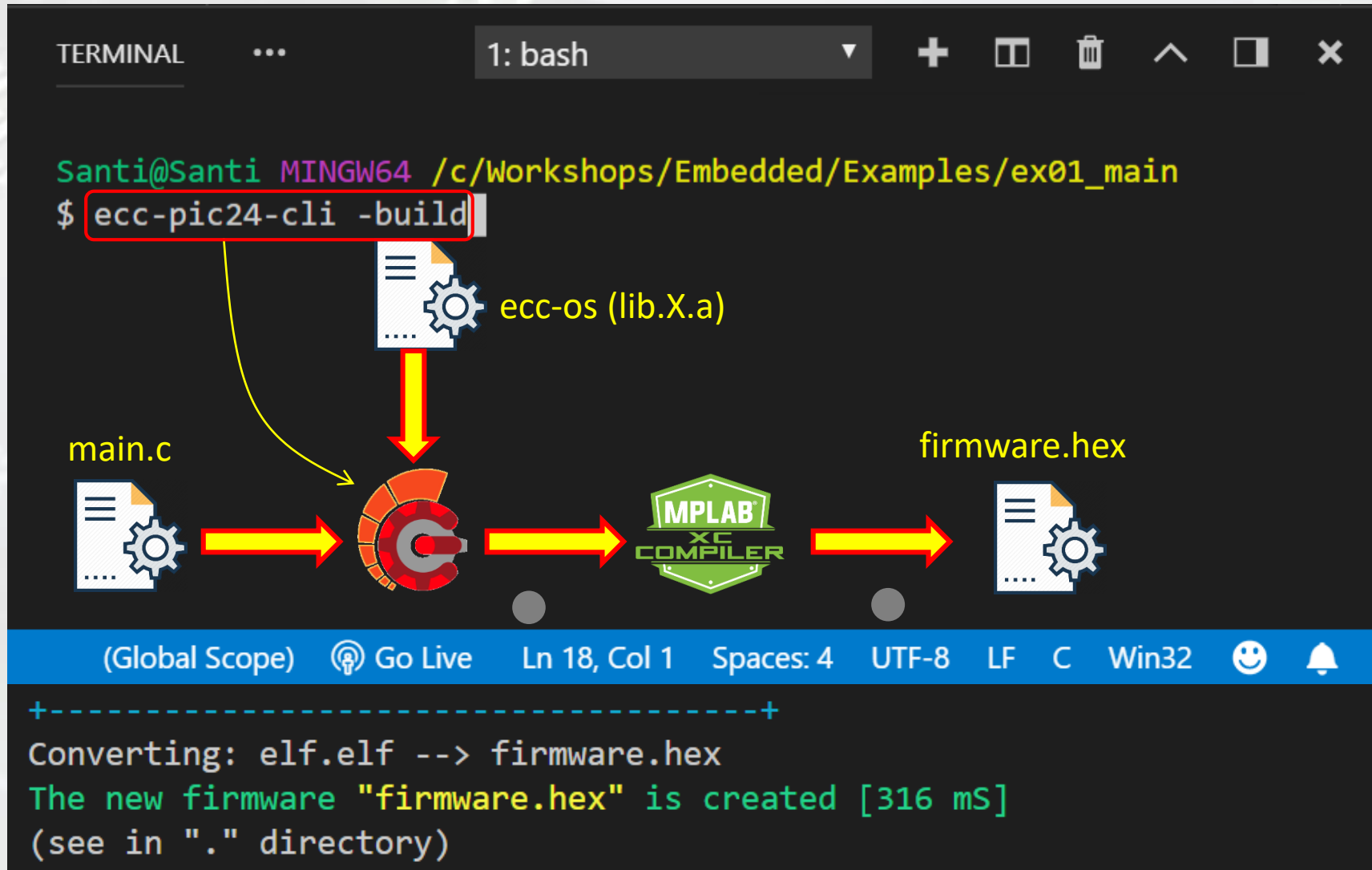
```
22 int main(void) {
23
24     ///!!Variables and hardware initialization
25     ///!!...
26
27     OS_Init(); // Initialise the OS
28
29     ///!!
30     ///!! Generate PWM signal, 1Hz, 10% duty on the LED0
31     LED_PwmSet(LED_ID_0, 1000, 0, 100);
32
33
34     ///!!
35     ///!! Beep for 0.1 seconds
36     Beep(100);
37
38     ///!!Other OS's functions
39     ///!!...
40
41     OS_Start(); // Start the OS
42 }
```

03 : On the menu bar, click **Terminal | New Terminal**



Getting Started with Embedded C-Programming

04 : Use the **ecc-pic24-cli -build** to compile the **main.c** to ***.hex** (firmware.hex)



```
TERMINAL 1: bash

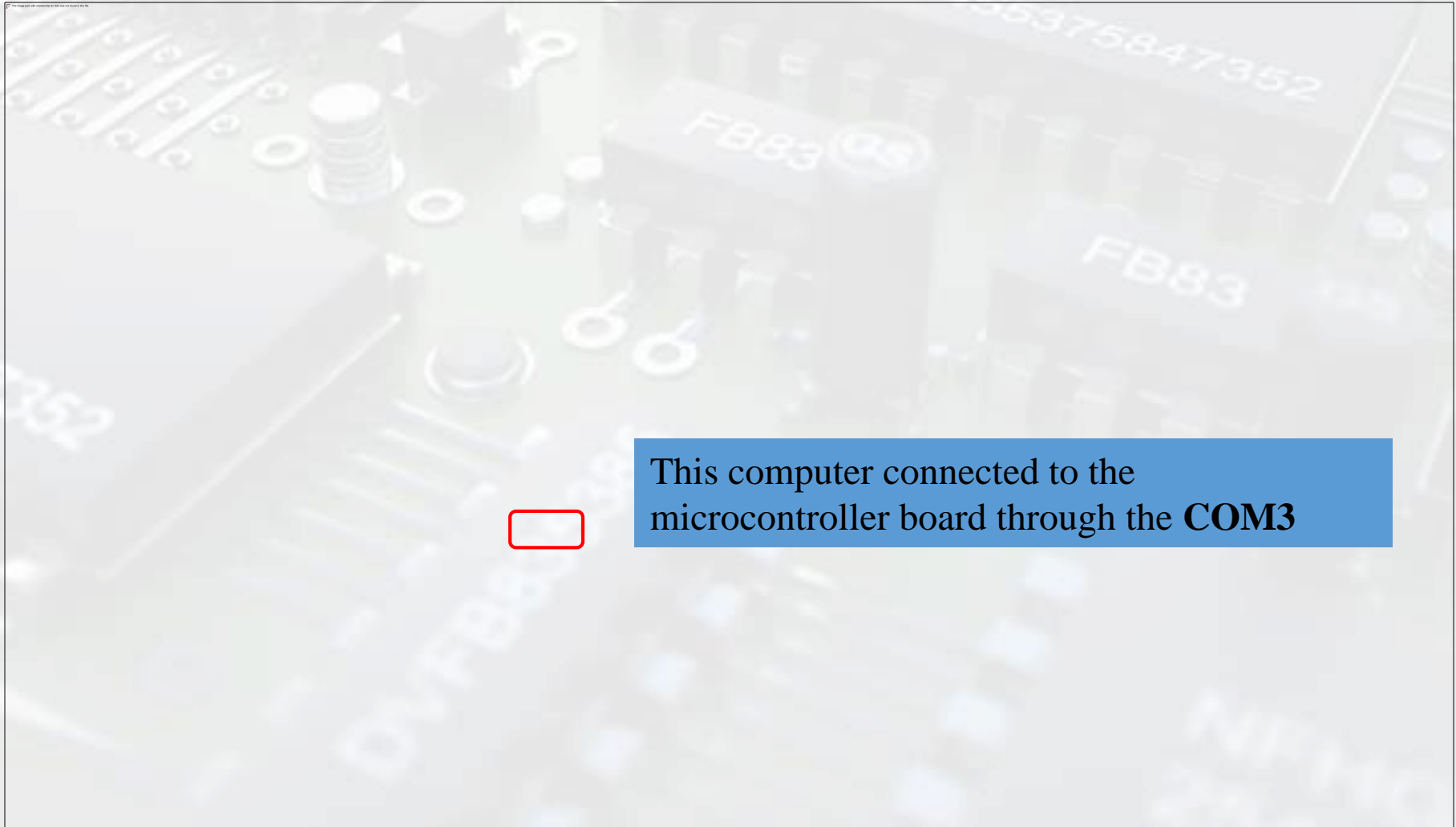
Santi@Santi MINGW64 /c/Workshops/Embedded/Examples/ex01_main
$ ecc-pic24-cli -build
```

The diagram illustrates the compilation process:

- main.c** (represented by a document icon with a gear) is the input source file.
- A yellow arrow points from **main.c** to a large red gear icon, representing the compilation step.
- Another yellow arrow points from the red gear icon to the **MPLAB XC COMPILER** (represented by a green shield icon).
- A yellow arrow points from the **MPLAB XC COMPILER** to **firmware.hex** (represented by a document icon with a gear).
- A yellow arrow also points from the terminal command `ecc-pic24-cli -build` to the red gear icon.
- A document icon with a gear and the text **ecc-os (lib.X.a)** is shown above the red gear icon, indicating a library being used.

```
+-----+
Converting: elf.elf --> firmware.hex
The new firmware "firmware.hex" is created [316 mS]
(see in "." directory)
```

05 : Connect the MCU board to computer using a USB cable, then check if the driver of USB-Serial is installed and ready to exchange data



This computer connected to the microcontroller board through the **COM3**

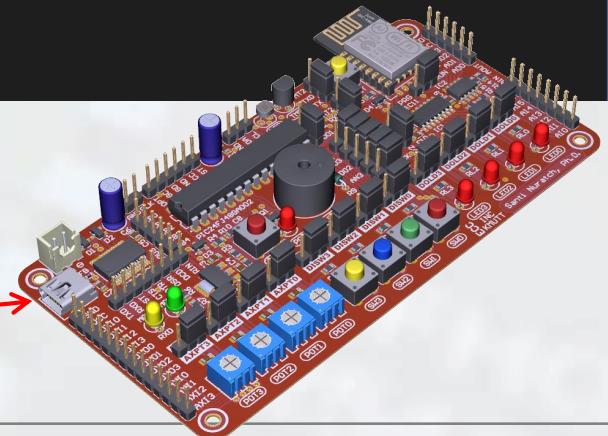
06 : Back to the terminal window of the VS-Code, and give it the command **`ecc-pic24-cli -flash`**

```
Santi@Santi MINGW64 /c/Workshops/Embedded/Examples/ex01_main  
$ ecc-pic24-cli -flash
```

```
+-----+  
|      ECC-PIC24-CLI v1.0.7      |  
|      Dr.Santi Nuratch         |  
| Embedded Computing and Control Lab. |  
|      (ECC-Lab)                |  
+-----+
```

```
Programming flash memory: 84.62 %  
Programming flash memory: 92.31 %  
Programming flash memory: 100.00 %  
Firmware has been updated [OK, 4.26 seconds]  
The microcontroller is now running...
```

firmware.hex



07 : Open the **main.c** and change this line

```
LED_PwmSet(LED_ID_0, 1000, 0, 100);
```

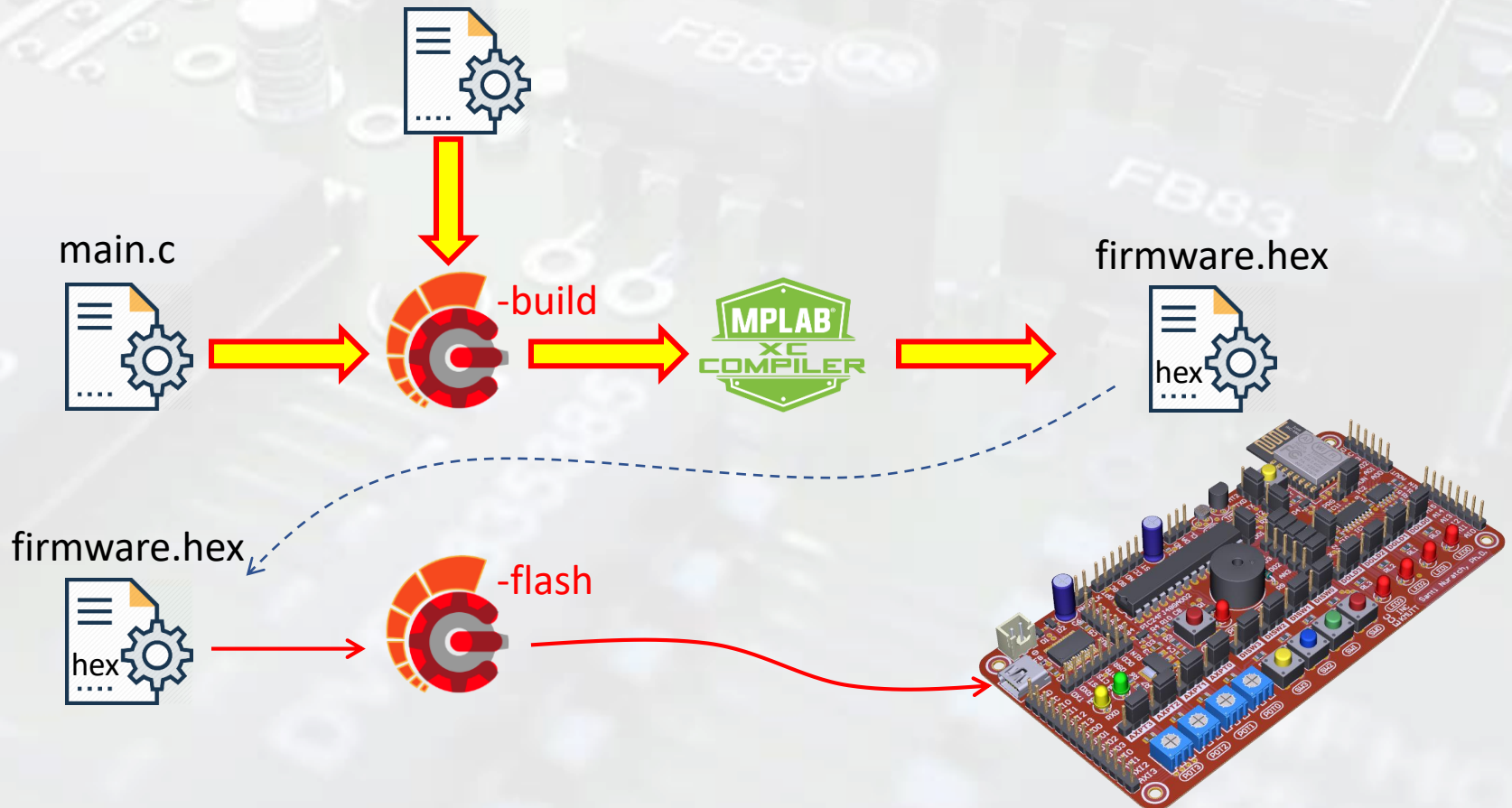
to

```
LED_PwmSet(LED_ID_3, 500, 0, 100);
```


Getting Started with Embedded C-Programming

08 : In the terminal window, enter the command **ecc-pic24-cli -build -flash**

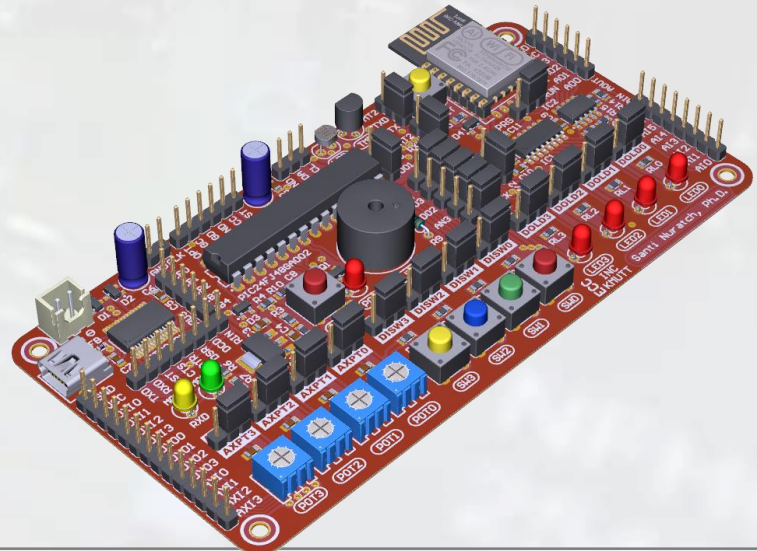
```
Santi@Santi MINGW64 /c/Workshops/Embedded/Examples/ex01_main  
$ ecc-pic24-cli -build -flash
```



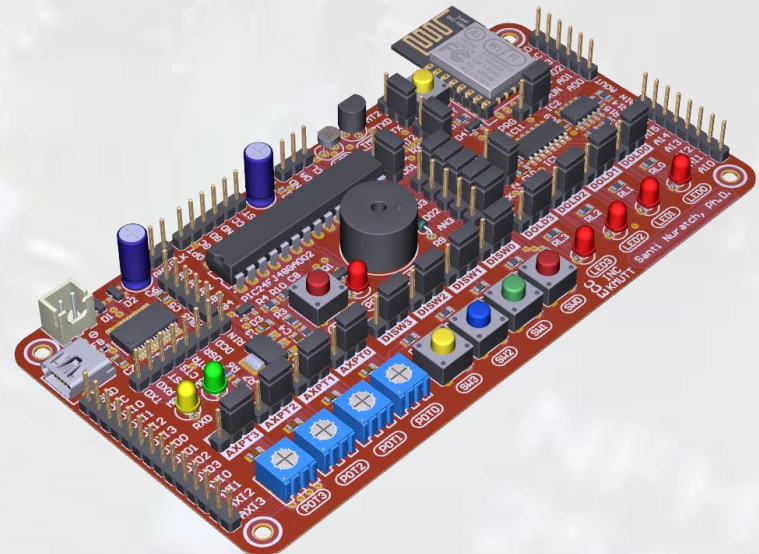
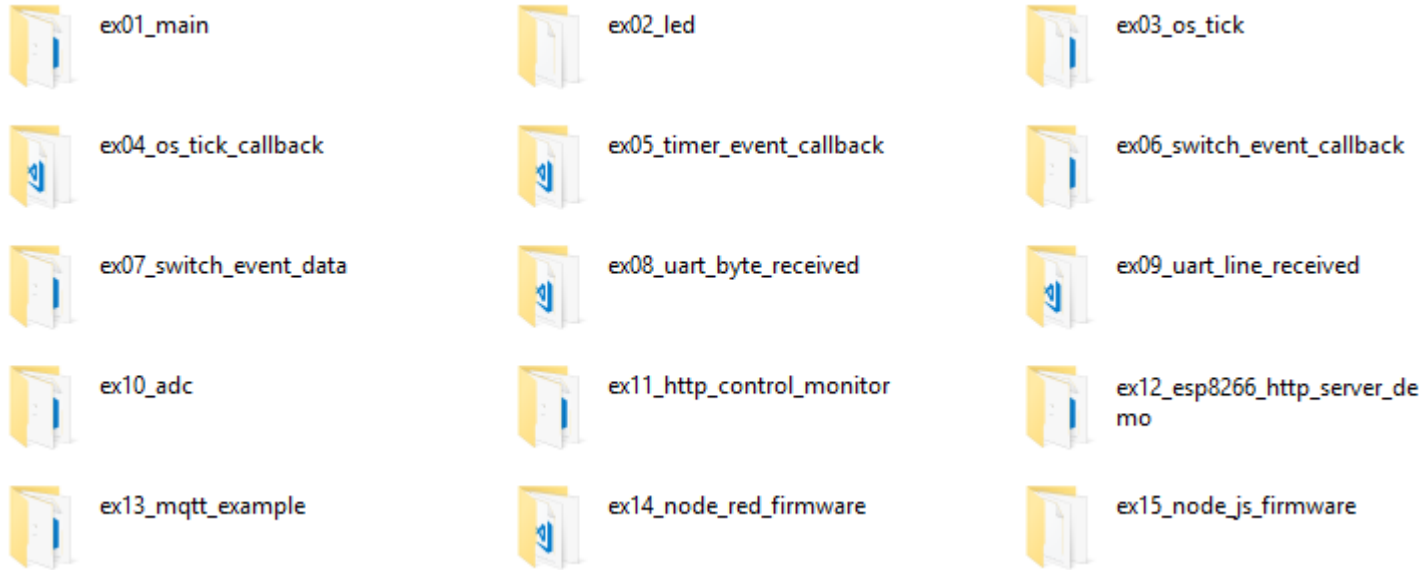
Let's Coding...

the MORE
YOU PRACTICE
THE BETTER
YOU GET

```
543 int main(void)
544 {
545     OS_Init();
546     AT_Init();
547     ESP_Init();
548     WiFi_Init();
549     Internet_Init();
550     OS_TimerCreate("AT_Service", 100, 1, AT_Service);
551     OS_WorkerCreate("WiFi_Init", Worker_ESPInitialise);
552     OS_Uart2SetLineReceivedCallback(ESP_LineReceived);
553     UART1_AsyncWriteString("\r\nMQTT Client...\r\n");
554     Beep(100);
555     OS_Start();
556 }
557
```



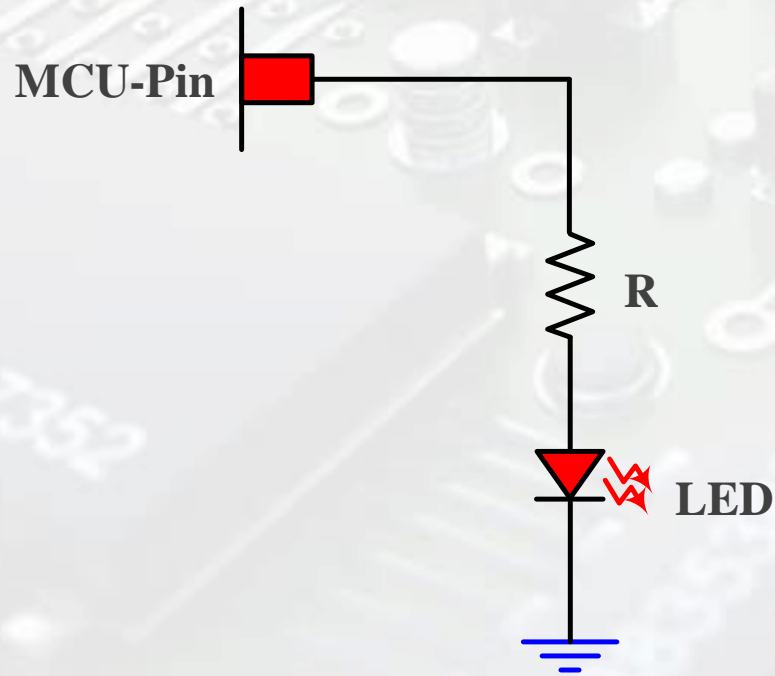
Learning by doing is the best way!



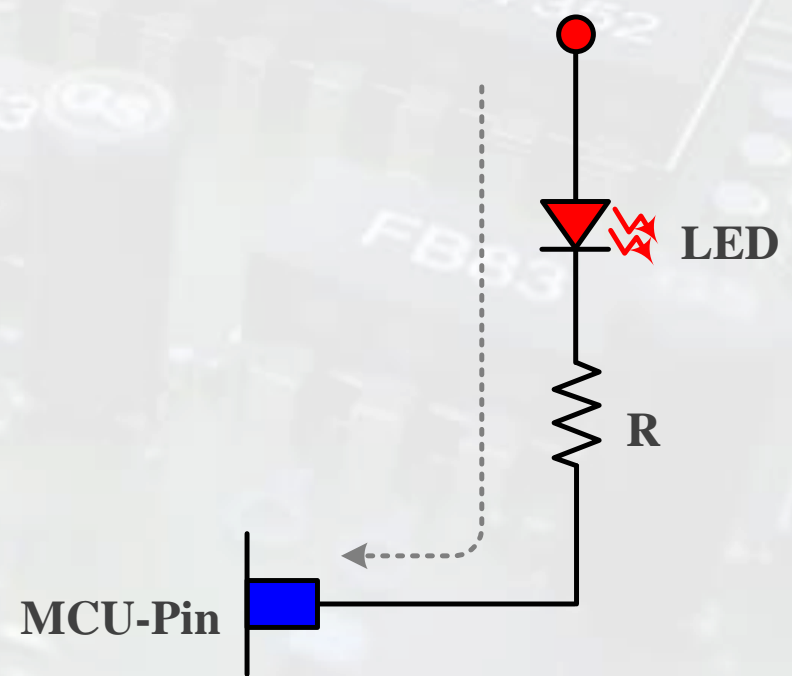


Basic Circuits

LED Circuits

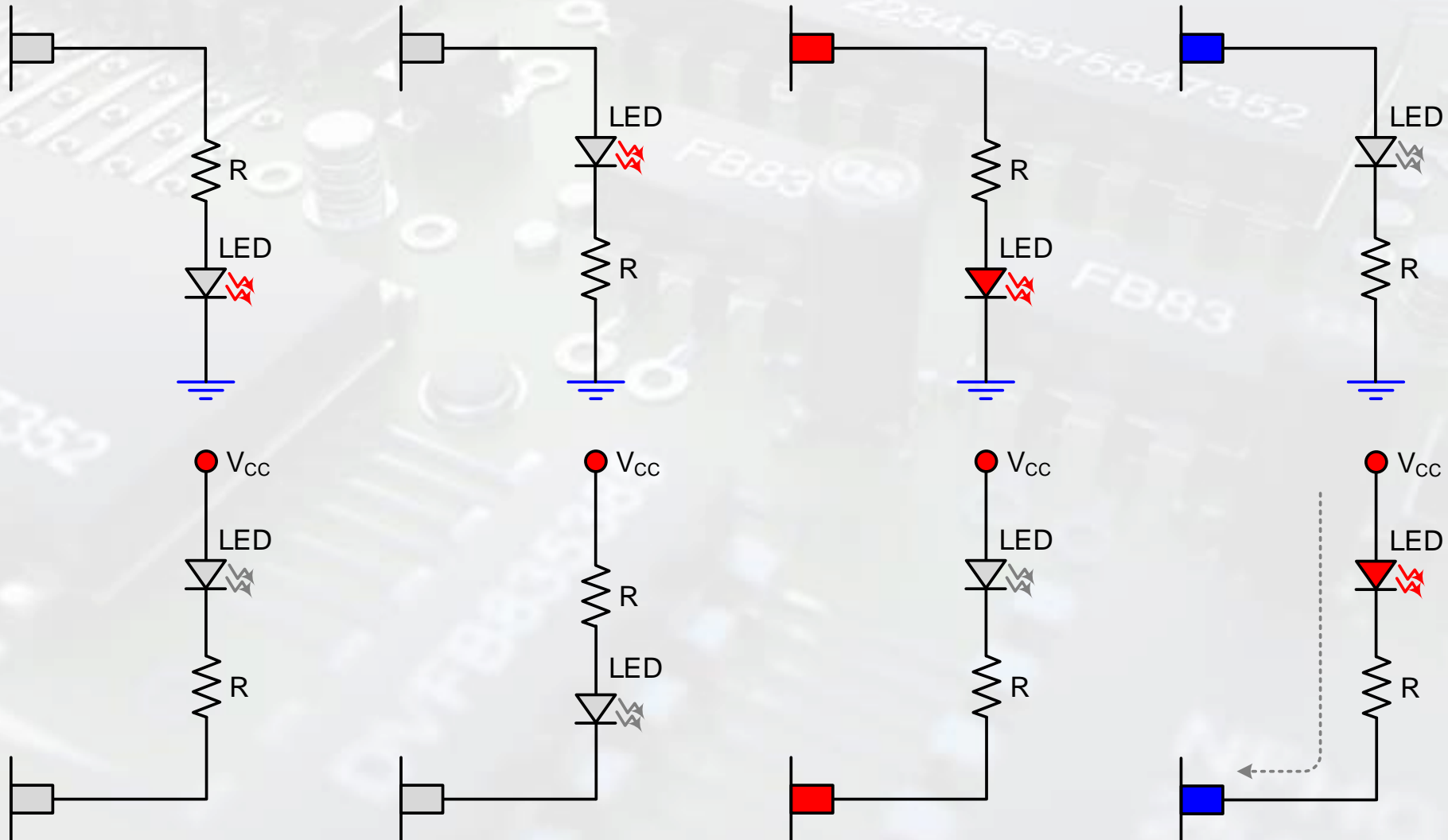


Active-High

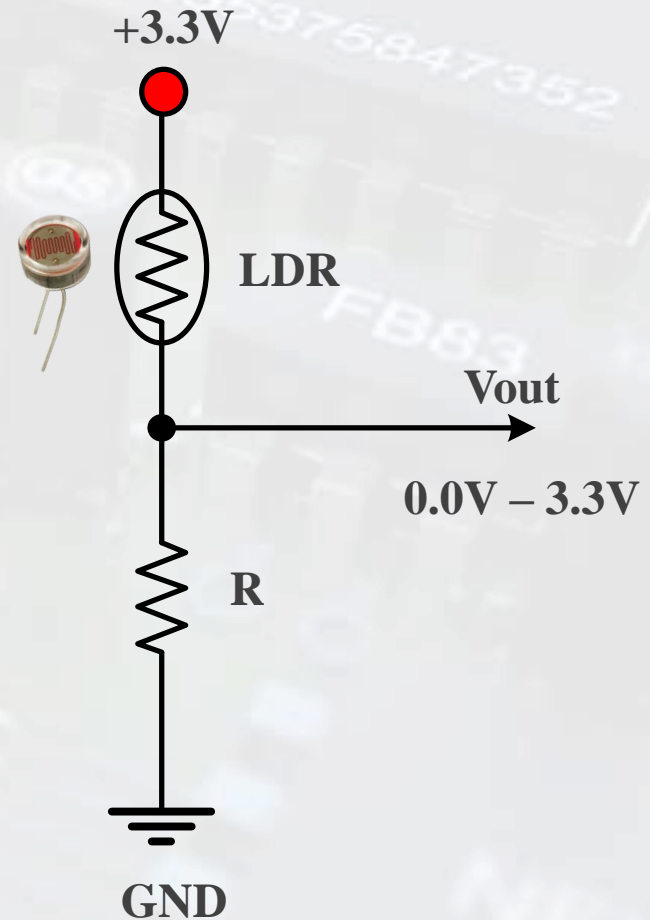
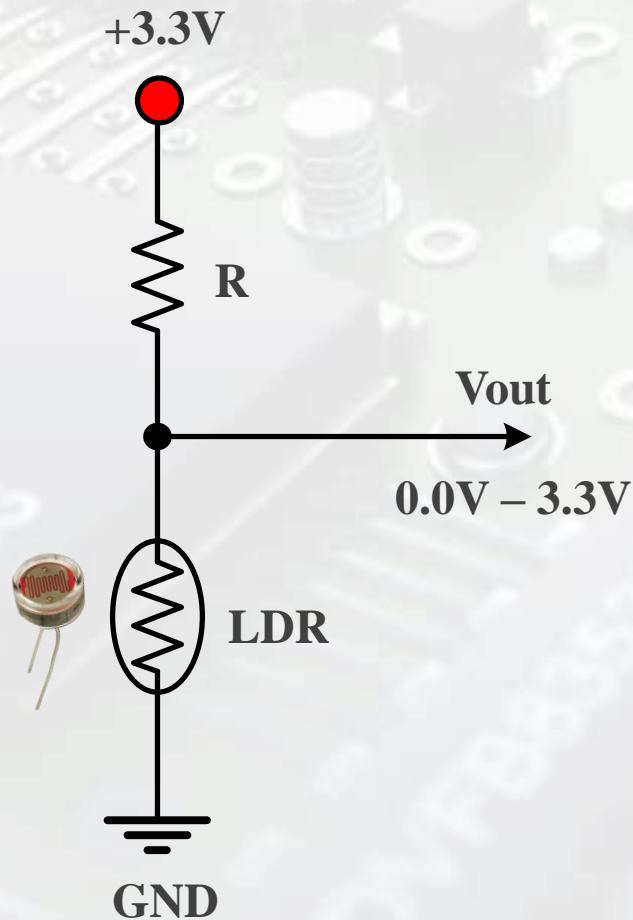


Active-Low

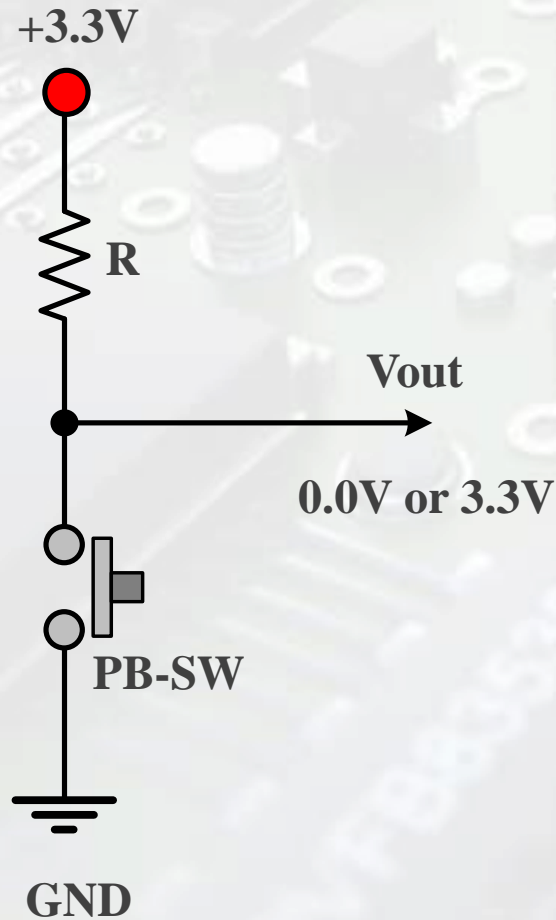
LED Circuits



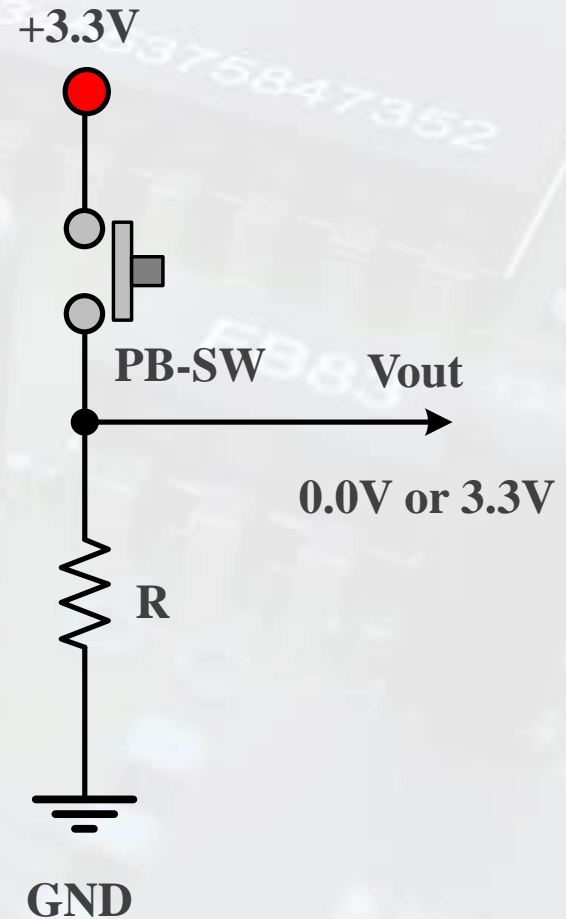
LDR (and other Variable Resistors) Circuits



Switch Circuits

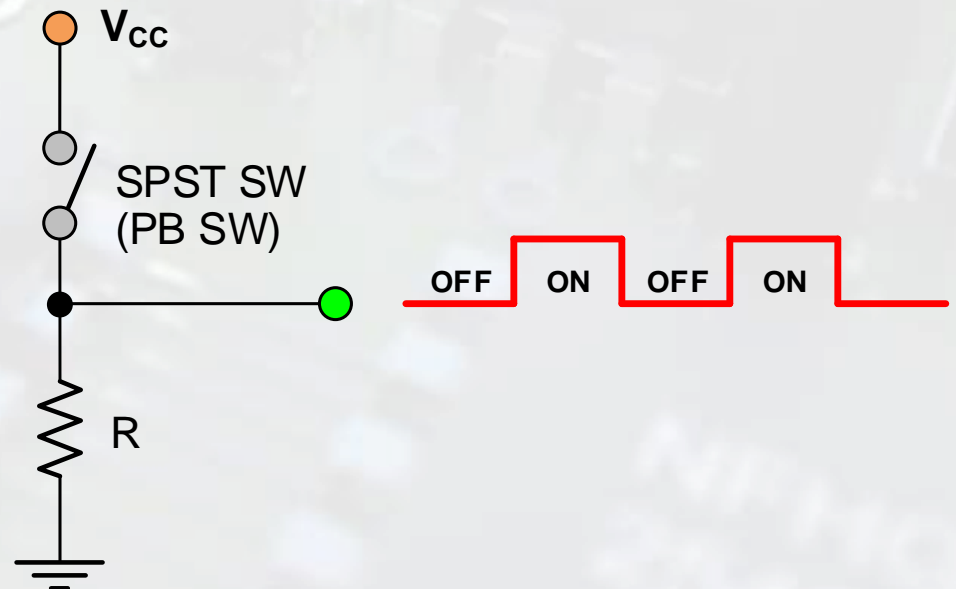
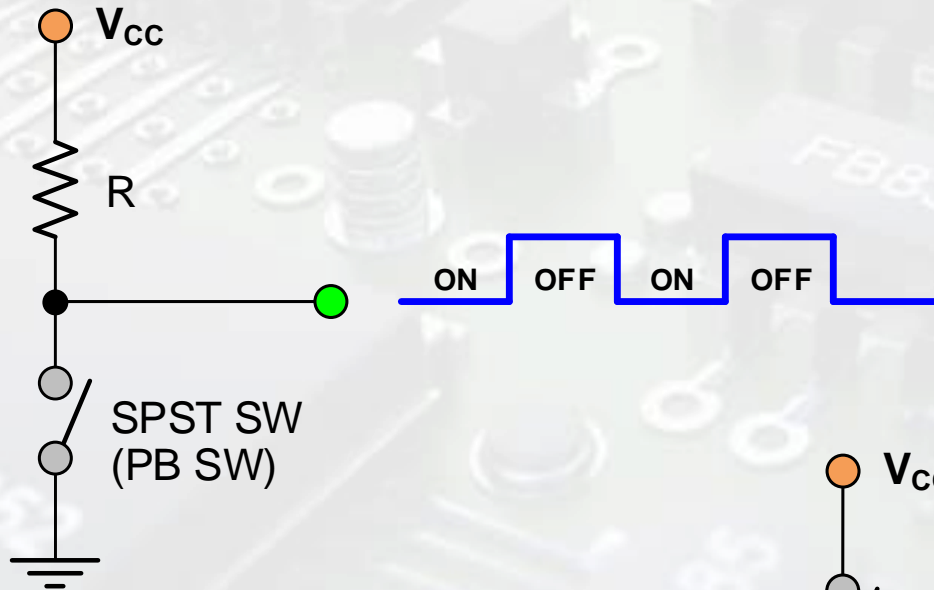


Active-Low

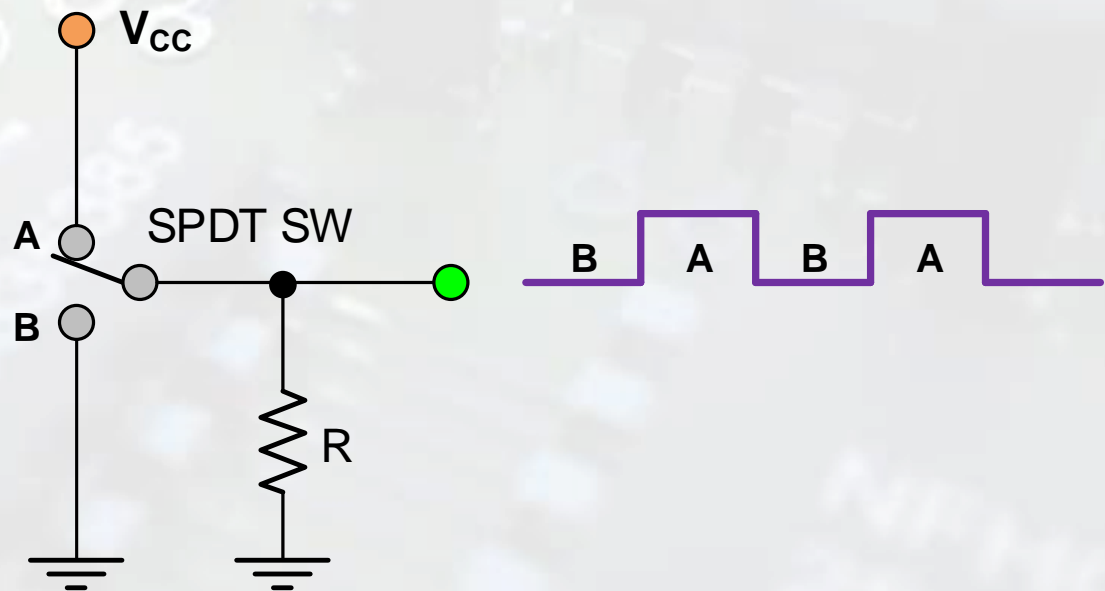
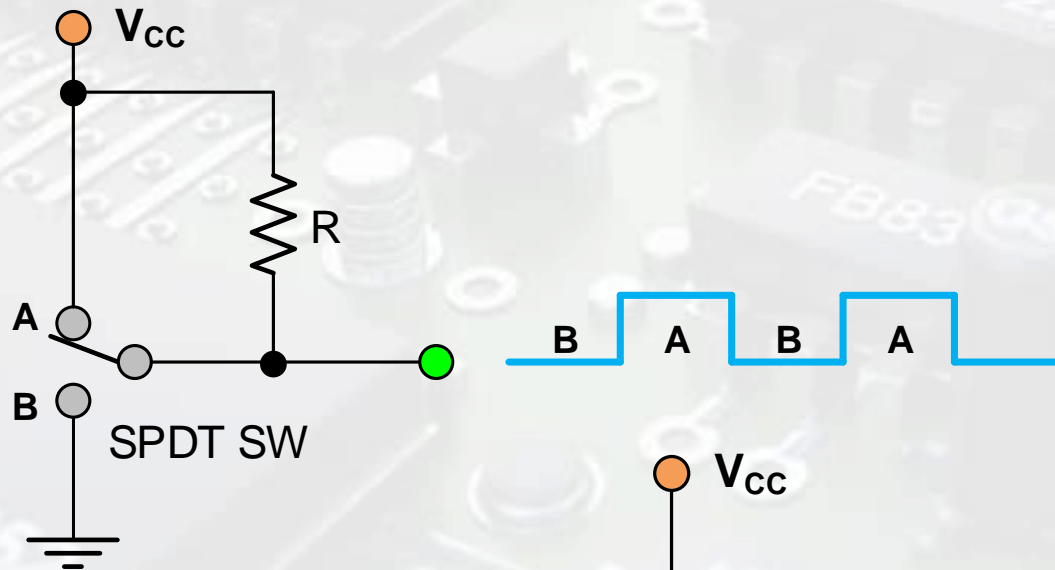


Active-High

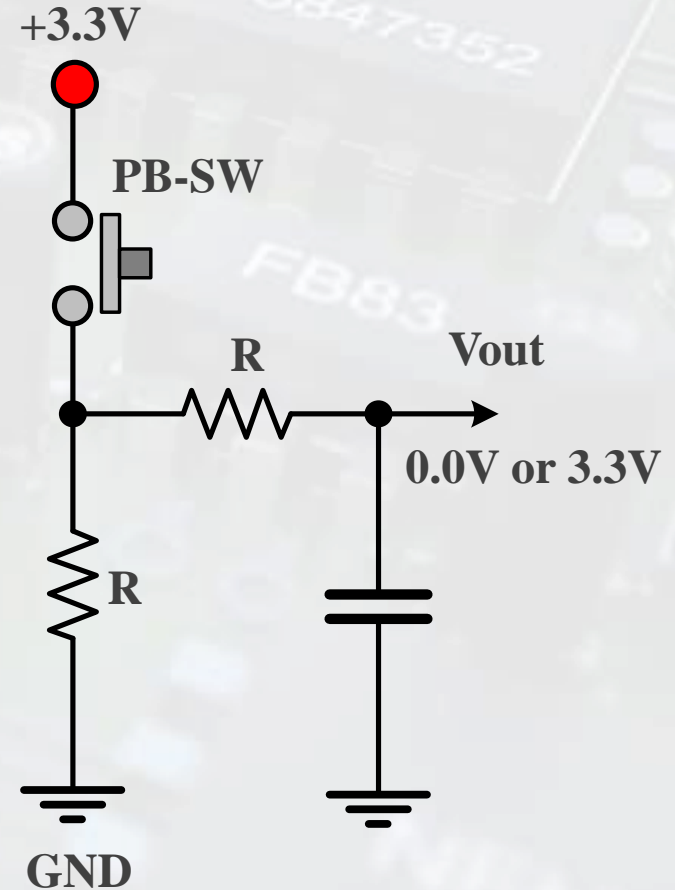
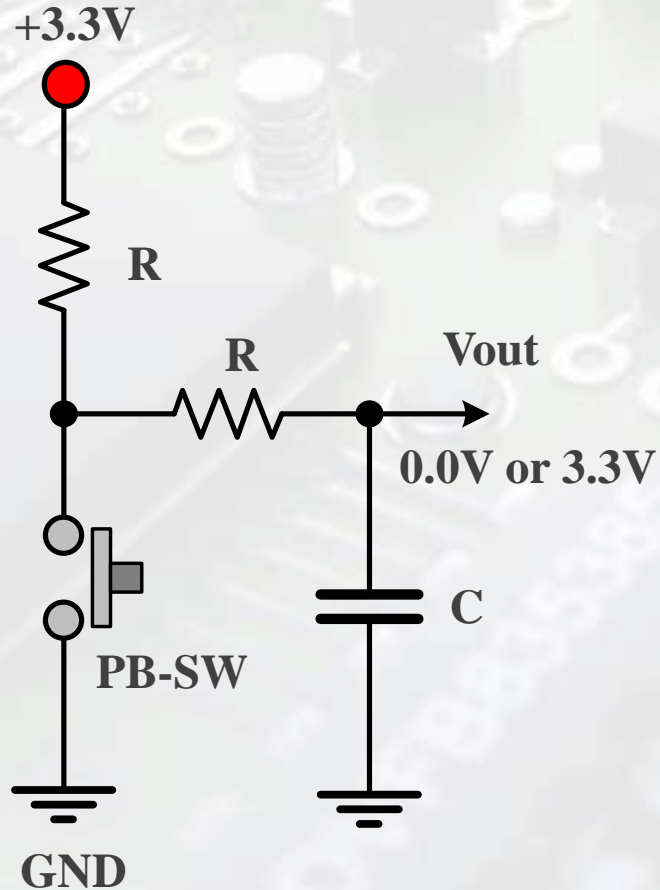
Single Pole Single Throw Switch



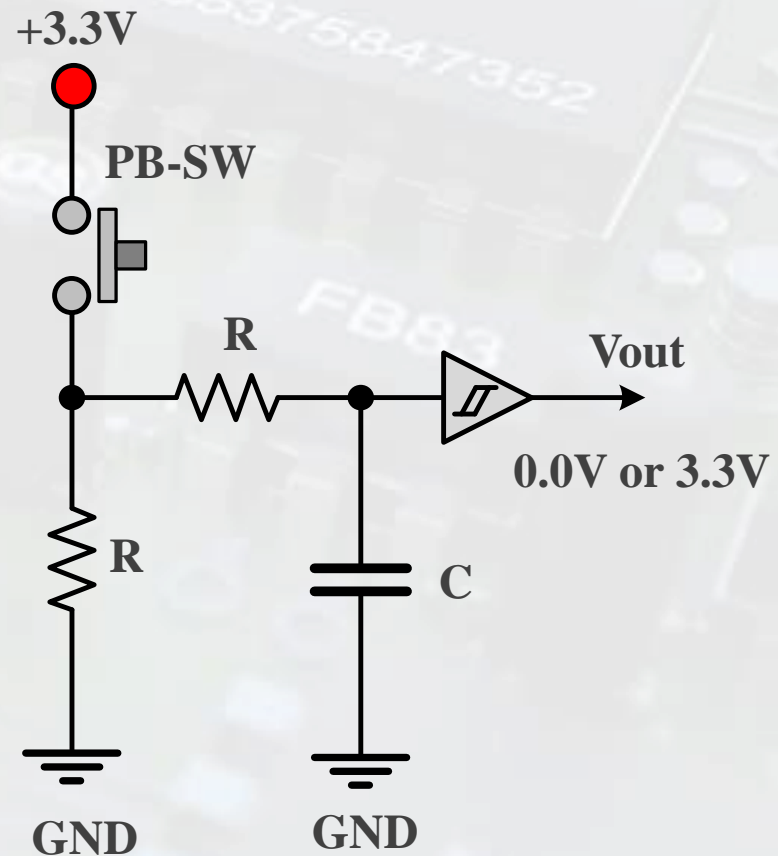
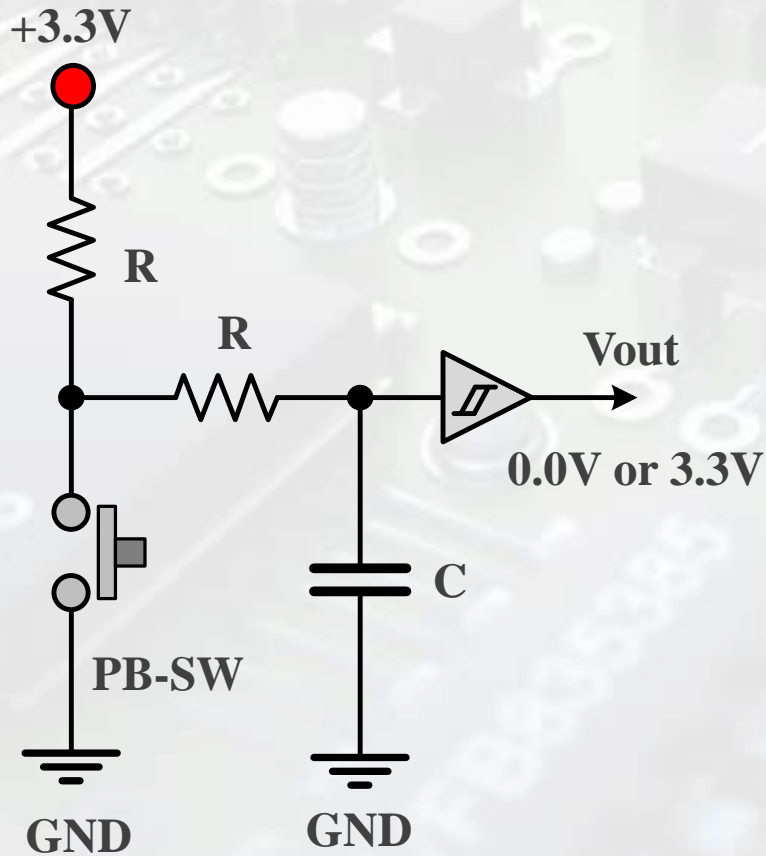
Single Pole Double Throw Switch



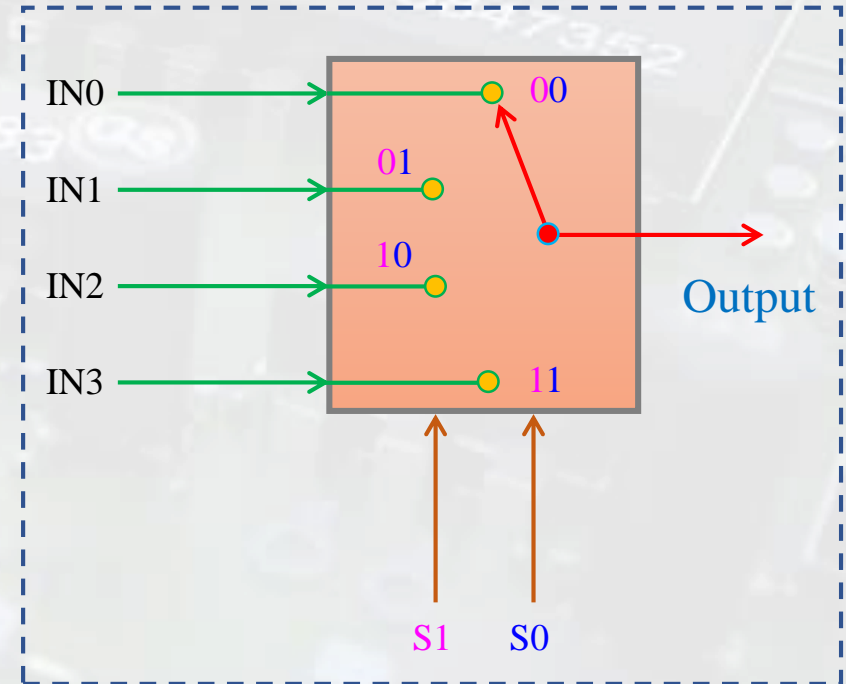
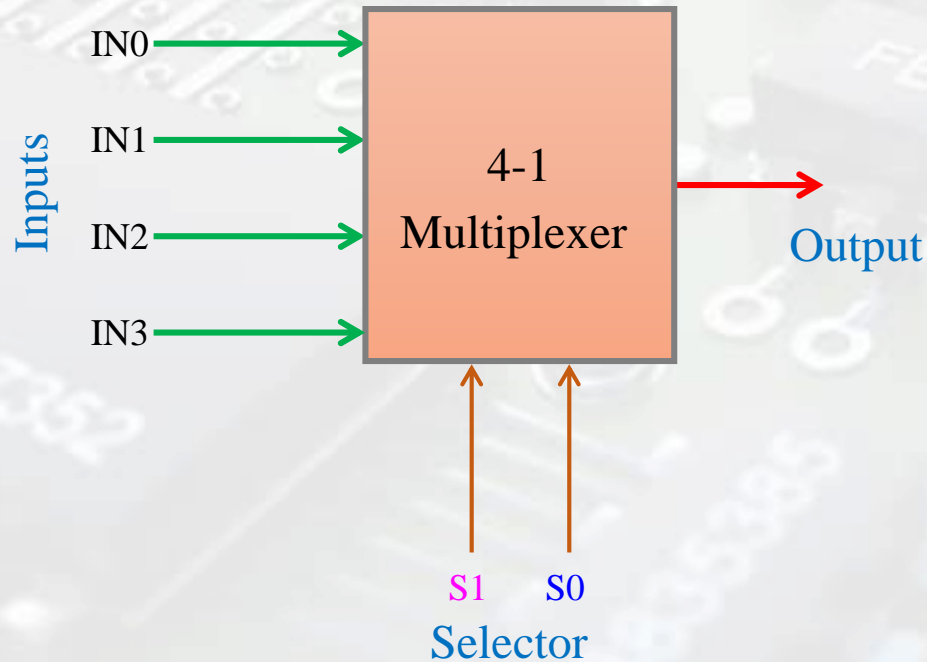
Switch and Capacitor Denouncing



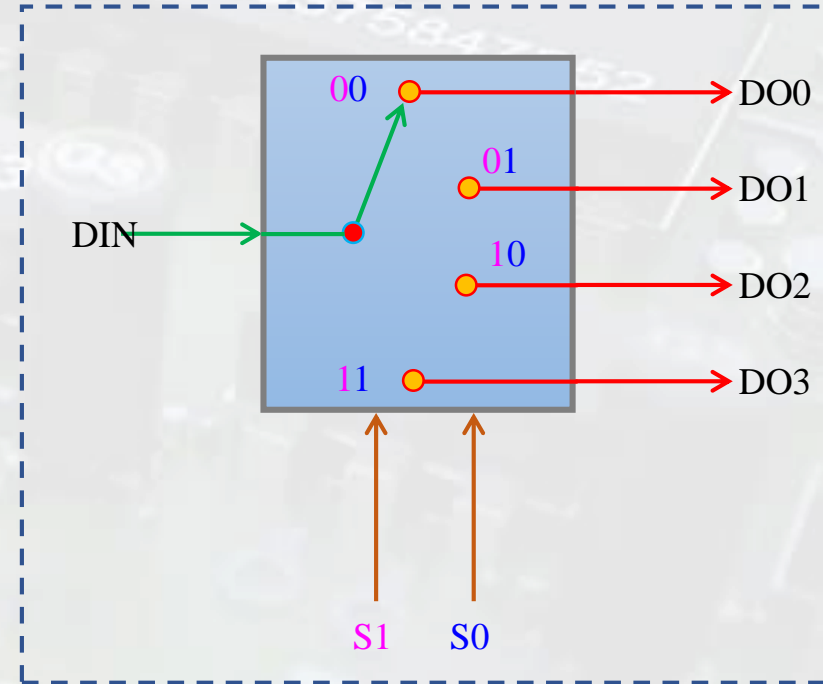
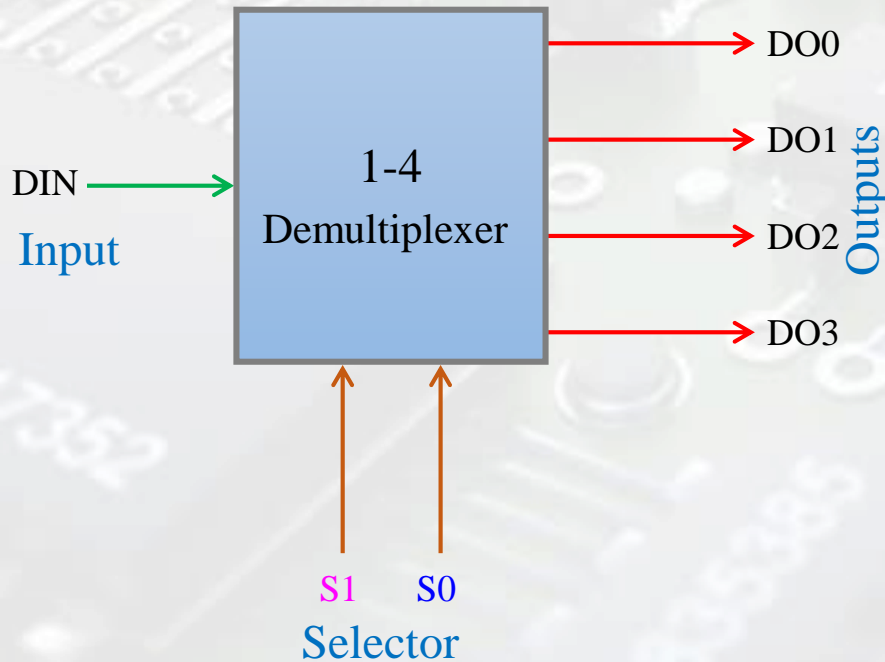
Switch and Schmitt-Trigger Denouncing



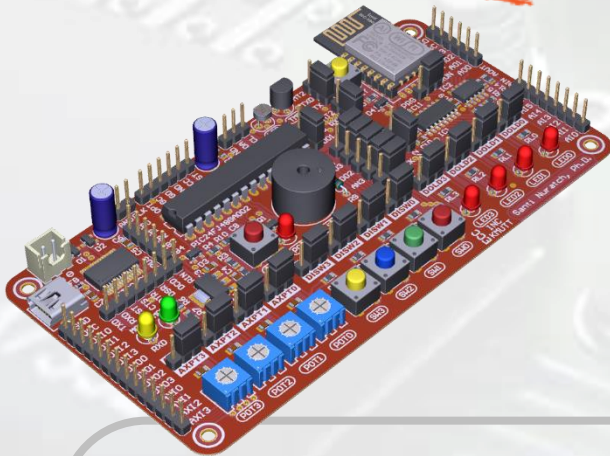
Digital and Analog Multiplexers



Digital and Analog Multiplexers



THANK YOU!



Santi Nuratch., Ph.D.

Embedded Computing and Control Lab. @ INC-KMUTT

santi.inc.kmutt@gmail.com, santi.nur@kmutt.ac.th

Department of Control System and Instrumentation Engineering,
King Mongkut's University of Technology Thonburi, **KMUTT**